



Mobile Attendance Taking application using NFC



A REPORT SUBMITTED TO MANCHESTER METROPOLITAN
UNIVERSITY FOR THE DEGREE OF BACHELOR OF SCIENCE

IN THE FACULTY OF SCIENCE AND ENGINEERING



2024

By

Taha Gorkem Sarac

Department of Computing and Mathematics

Abstract

This dissertation presents the development of a Mobile Attendance Taking application which uses Near Field Communication (NFC), designed to improve the efficiency and accuracy of tracking student attendance in higher education settings. By integrating NFC technology into the mobile application, this project aims to overcome the limitations of traditional attendance methods such as ID cards coupled with card readers. The study conducts a thorough review of existing attendance technologies and methodologies and the technologies to be used, establishing the foundation for a system that combines Microsoft SQL Server, ASP.NET Core, and React Native to create a robust, cross-platform solution. The implementation covers system design, database management, API creation, deployment to the server, and the mobile application development, culminating in a thorough testing phase including unit, speed, and user testing to validate usability. Evaluation of the system highlights its advantages over conventional methods and identifies potential limitations of the system such as its security. The conclusion reflects on the project's academic and practical applications, suggesting future directions for enhancing security features and expanding system capabilities. This work aims to contribute to the field by demonstrating the practical application of NFC and mobile technologies in educational settings, showing the acceptance of the solution, thus showing its potential for broader applicability in various contexts.

Table of Contents

Abstract	1
Declaration	7
Acknowledgements	8
List of Figures	9
Abbreviations	12
1. Chapter 1 - Introduction	15
1.1 Background	15
1.2 Aims and Objectives	17
1.3 Report structure	18
2. Chapter 2 – Literature Review	20
2.1 Attendance registration in academic settings	20
2.2 Attendance registration methods	23
2.3 Internet of Things	25
2.4 RFID and NFC technology	29
2.5 Similar work on attendance registration	32
2.5.1 NFC-Based Attendance Systems	32
2.5.2 Biometric and RFID-Based Solutions:	33
2.5.3 Machine Learning and Facial Recognition Technologies:	33
2.5.4 General Challenges and Technological Implications	34

2.6	Development methodology	35
2.6.1	Waterfall.....	35
2.6.2	Scrum	35
2.6.3	Extreme Programming (XP)	36
3.	Chapter 3 – Requirements.....	37
3.1	Functional Requirements	37
3.1.1	User Requirements.....	37
3.1.2	Admin Requirements	38
3.1.3	Mobile Application Requirements	38
3.1.4	API Requirements	39
3.1.5	Database Requirements.....	39
3.2	Non-Functional Requirements	40
3.2.1	Performance	40
3.2.2	Reliability.....	40
3.2.3	Usability	40
3.2.4	Security	40
4.	Chapter 4 – Design.....	41
4.1	Development Methodology	41
4.2	Version control	43
4.3	Choosing the NFC technology.....	44
4.4	Designing the system architecture	46

4.5	Designing the database	47
4.5.1	Comparing various databases	47
4.5.2	Choosing the DBMS	48
4.5.3	Entity-Relationship Diagram	49
4.6	Designing the API	50
4.6.1	Comparing various API types and frameworks	50
4.6.2	Choosing the API framework.....	52
4.6.3	Api endpoints	53
4.7	Designing the Mobile application.....	54
4.7.1	Mobile development frameworks	54
4.7.2	Choosing the appropriate framework.....	55
4.7.3	User interface and User experience	56
5.	Chapter 5 – Implementation.....	63
5.1	Creating the database	63
5.2	Stored procedures.....	69
5.2.1	sp_attendance	69
5.2.2	sp_get_attendance	71
5.3	Creating the API.....	74
5.4	Setting up the domain and hosting.....	84
5.4.1	Setting Up the Subdomain for the API	84
5.4.2	Exporting and Importing the Database	86

5.4.3	Publishing the ASP.NET Core API	87
5.5	Creating the mobile application	88
5.5.1	Deploying the mobile application.....	100
6.	Chapter 6 – Testing	101
6.1	Unit testing.....	101
6.2	Speed and efficiency testing	102
6.3	Black box testing.....	104
6.3.1	Testing Procedure and Encountered Bugs:	104
6.4	User testing	106
6.4.1	Methodology	106
7.	Chapter 7 – Evaluation.....	108
7.1	Achievement of Objectives and requirements	108
7.2	System evaluation	111
7.2.1	Comparative Advantages	111
7.3	User feedback.....	113
7.3.1	Findings:	113
7.3.2	Accuracy of gathering user feedback.....	118
7.4	Limitations	119
7.4.1	Buddy Punching with Cheap Phones:.....	119
7.4.2	Security Weaknesses:	119
7.4.3	Reverse Engineering and Location Spoofing:	119

7.4.4	Deployment Constraints on iOS:	120
7.4.5	Database Shortcomings:.....	120
8.	Chapter 8 – Conclusion.....	121
8.1	Conclusion	121
8.2	Next steps.....	123
8.3	Reflection.....	124
	References.....	126
	Appendices.....	132
	Appendix A	132
	Appendix B	134
	Appendix C	140

Declaration

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work. This work has been carried out in accordance with the Manchester Metropolitan University research ethics procedures and has received ethical approval number: 64882

Signed: Taha Gorkem Sarac

Date:16/05/2024

Acknowledgements

I would like to thank my supervisor, Dr Kristopher Welsh for his guidance throughout the Synoptic Project unit and throughout my final year of university in general. I would also like to thank my friends for their feedback and testing of the project.

Finally, I'd like to thank my family for their continuous support during my academic career.

List of Figures

Figure 1: Philips hue light bulb.....	26
Figure 2: Philips hue mobile application	27
Figure 3: The developer's Philips hue lightbulb, dimmed automatically because of the late hour of this section being written	27
Figure 4: System architecture design diagram.....	46
Figure 5: Entity Relationship Diagram for the Attendance database.....	49
Figure 6: Login screen design.....	57
Figure 7: Student home screen design	58
Figure 8: Student timetable screen design	59
Figure 9: Admin home screen design.....	60
Figure 10: Manage table screen design, consistent across all tables	61
Figure 11: Manage a particular row screen design	62
Figure 12: Creation of the attendance database	64
Figure 13: Course table design view.....	64
Figure 14: Unit table design view	65
Figure 15: Classroom table design view	65
Figure 16: Student table design view.....	66
Figure 17: Student Timetable table design view.....	66
Figure 18: Timetable table design view	67
Figure 19: Timetable Content table design view	67

Figure 20: Attendance table design view	68
Figure 21: ASP.NET Core Web API project creation	74
Figure 22: ASP.NET Core Web API project configuration.....	74
Figure 23: Program.cs file after deleting example code	75
Figure 24: Dapper and Microsoft.Data.SqlClient libraries installed	75
Figure 25: Connection string	76
Figure 26: All the API models representing the tables	76
Figure 27: Course endpoint in swagger UI.....	77
Figure 28: API Endpoints.....	78
Figure 29: API website configuration	84
Figure 30: Let's encrypt CMD interface	85
Figure 31: Displaying the API website on the Let's encrypt CMD interface	85
Figure 32: ASP.NET Core project publish screen.....	87
Figure 33: Updated connection string.....	87
Figure 34: Manage course screen in the mobile application.....	97
Figure 35: Student attendance page in the mobile application	98
Figure 36: Location fetching time test results.....	102
Figure 37: Useability rating by users	113
Figure 38: Student interface rating by users	114
Figure 39: Admin interface rating by users.....	115
Figure 40: System use instead of the current system rating.....	116

Figure 41: System use alongside the current system rating	117
---	-----

Abbreviations

Abbreviation	Meaning
IoT	Internet of Things
AI	Artificial Intelligence
ID	Identification
NFC	Near Field Communication
RFID	Radio Frequency Identification
App	Application
API	Application Programming Interface
SQL	Structured Query Language
IT	Information Technology
OS	Operating System
UI	User Interface
UX	User Experience
JS	JavaScript
DBMS	Database Management System

HVAC	Heating, Ventilation, and Air Conditioning
FDA	Food and Drug Administration
ISO	International Standards Organization
EPC	Electronic Product Code
ISM	Industrial, Scientific, and Medical
XP	Extreme Programming
UUID	Universally Unique Identifier
ERD	Entity-Relationship Diagram
REST	Representational State Transfer
HTTPS	Hypertext Transfer Protocol Secure
SOAP	Simple Object Access Protocol
XML	Extensible Markup Language
AWS	Amazon Web Services
RDS	Relational Database Service
CRUD	Create, Read, Update, Delete

IDE	Integrated Development Environment
XAML	Extensible Application Markup Language
MAUI	Multi-platform App User Interface
CLI	Command Line Tool
JDK	Java Development Kit
USD	United States Dollar

1. Chapter 1 - Introduction

1.1 Background

With how fast technology has advanced over the past few decades, it is undeniable that systems once used only for specialized cases have become widespread and utilized by the public. As cutting-edge technologies see increasingly broader use, the average person has become more knowledgeable in technology than ever before, with virtually everyone claiming at least a basic understanding of how various technologies work.

However, this technological knowledge among the population has given rise to a concerning trend of generalization and oversimplification. Highly specialized concepts and systems are often rebranded or miscategorized in an effort to render them more acceptable to a mainstream audience. An example of this issue can be seen in recent products and software inaccurately marketed as "Artificial Intelligence (AI)" despite having no major connection to Artificial Intelligence beyond the intention to capitalize on the buzzword's popularity.

Another instance of this generalization issue, particularly relevant within the technology industry, involves the misunderstood concept of the "Internet of Things" (IoT). While many who work in the Information Technology (IT) industry could confuse the IoT with the conventional internet, the global network facilitating communication between billions of computers and electronic devices. However, these people would not be wrong completely, as the Internet of Things is indeed based on the same principle of connecting “things” via a network. To be precise, the Internet of Things is the network of physical objects or “Things” such as sensors and their ability to exchange data with each other or various other devices over the Internet.

In the realm of academics and various organizational settings, the recording of attendance is a crucial task. It forms the backbone of administrative efficiency and academic integrity, playing a critical role in the evaluation of institutional effectivity. Historically, the process of tracking attendance has been a manual endeavour, involving sign-in sheets and more recently physical tokens of presence such as identification (ID) cards. While traditional and widespread, these methods come with their own challenges as they demand significant time and administrative

oversight, which is susceptible to human error, and can be intentionally manipulated. Such as the cases of “buddy punching”, where a peer would register attendance for their friends using their cards. Thus, compromising the accuracy of the records maintained.

Thanks to the widespread use of these technologies, such as the IoT, is the advance and steady progression it brings alongside it, which presents an opportunity to enhance everyday concepts such as attendance registration. Among the abundance of innovations, Radio Frequency Identification (RFID) and Near Field Communication (NFC) have emerged as systems that have been integrated into everyday devices such as mobile phones. The existence of these technologies within the public can allow the creation of systems for simple attendance registration and tracking within an institution, through the use of electronic tags and readers. Such a digital upgrade not only streamlines the process, greatly reducing the time spent on attendance registration but also significantly lowers the possibility of human error and unethical practices such as “buddy punching”.

Leveraging these advances, the proposed Mobile Attendance Taking Application aims to improve managing and documenting the presence of individuals within educational contexts. The core functionality of this application is to use the versatility of IoT, particularly the convenience and accessibility of NFC technology. By doing so, the application seeks to offer a simple, user-centred method for attendance recording. Aimed at benefiting both the students, educators, and institutions, as it simplifies the process into a digital interaction, transforming smartphones into virtual attendance registers.

This application promises two main benefits: for students, it ensures their attendance is logged even in the absence of their ID cards, therefore accommodating a simple and reliable way to ensure their continued attendance. For educators, it minimizes the administrative burden, allowing them to redirect their focus on their core responsibilities rather than tedious tasks. Furthermore, by digitizing attendance, the system inherently enhances data integrity and security, providing a reliable information on students and their attendance habits.

1.2 Aims and Objectives

The primary aim of this project is to develop a simple and intuitive mobile app that utilizes NFC technology to communicate with classroom attendance devices or tags. By allowing students to register their attendance using their mobile phones, even if they forget their ID cards, the application relieves the administrative burden on lecturers and minimizes instances of students attempting to register attendance on behalf of their peers.

Objectives:

- Determine the appropriate software and framework for the development of the mobile application, to ensure the delivery of a high-quality product.
- Design and implement a user-friendly interface that ensures a seamless experience for users.
- Develop a robust and efficient database system to manage student information and track their attendance records securely.
- Develop a reliable and secure API to facilitate seamless communication between the database and the mobile application.
- Integrate a secure log-in system for students to authenticate with their university accounts.
- Incorporate NFC technology for attendance registration within the mobile app, leveraging IoT capabilities for an effortless attendance registration process.
- Evaluate and select the most suitable NFC technology, ensuring compatibility with the application.
- Conduct comprehensive usability testing and gather feedback from users, including students, and lecturers to refine the application's functionality and user experience.
- Evaluate and optimize application performance, addressing any identified issues or bugs based on user feedback, ensuring a seamless and efficient attendance tracking experience.

By leveraging the power of IoT and NFC technology, this project aims to modernize attendance tracking in educational institutions, paving the way for a more efficient learning environment.

1.3 Report structure

Chapter 1, Introduction – Sets the stage for the research, detailing the technological advancements and their impact on society, with a particular focus on the education sector. It elaborates on the background of IoT, the motivation behind adopting NFC, and the specific aims and objectives of the project.

Chapter 2, Literature review – Gives foundational knowledge and context necessary for understanding the project. It explores the importance of attendance in various settings, followed by an examination of different attendance methods. The report then extends into the details of IoT, RFID and NFC Technologies which provides technical insights necessary for the project. The chapter concludes with a review of Similar Projects.

Chapter 3, Design – Addresses the Design aspects of the project. Starting with the choice of technology, followed by the overall System Design. Subsequent sections discuss the selection and design of the Mobile App Framework, the API Framework, and the Database.

Chapter 4, Requirements – Outlines the Requirements gathered for the development of the system. This section not only specifies the functional and non-functional requirements but also considers additional elements necessary for a comprehensive system design.

Chapter 5, Implementation – Focuses on the Implementation of the system. Describes and showcases the process of Creating the Database, API, and transferring them to the Server. It also covers the steps involved in creating the Mobile Application and connecting the API to it. Additionally, the process of deploying the mobile application is also shown.

Chapter 6, Testing – The testing processes are described and detailed. It includes various testing methodologies such as Unit, Efficiency, Black Box, and User Testing. This chapter ensures that the system meets all requirements and functions as expected under different scenarios. Feedback from this phase produces further refinements.

Chapter 7, Evaluation – The following chapter evaluates the system components, revisits the requirements to see if they were met, evaluates the user feedback results, and discusses the limitations of the current system. Additionally, the feasibility of Deploying to other operating systems and considerations regarding the project's security are explored.

Chapter 8, Conclusion – The final chapter reflects on the project and provides a Conclusion. It encapsulates the challenges faced and the solutions implemented. The reflection section contains the thoughts of the author/developer and the knowledge gained throughout the project lifecycle. This chapter not only concludes the report but also provides a reflective look at the project's impact and potential future revisions, suggesting the next steps that need to be taken.

2. Chapter 2 – Literature Review

This chapter aims to provide a comprehensive background and context for understanding the key concepts and other work related to attendance registration in academic settings, with a specific focus on leveraging emerging technologies like the Internet of Things (IoT), RFID, and NFC.

2.1 Attendance registration in academic settings

The importance of attendance within the context of education has multiple key factors, extending beyond the confines of the academic environment. It is a concept which can be affected by many key variables such as the institution itself, student's personal factors and the type of course. Because of its centrality to the educational experience, institutional reputation, and subsequent professional pathways for the students, it is a matter that requires attention from both the student's themselves and the institutions. The importance of attendance for students is highlighted by multiple studies, displaying that regular attendance is a significant indicator of academic success (Credé, Roch and Kieszczynka, 2010) (Macfarlane, 2013).

Furthermore, irregular attendance can often be the causation of academic failure and in more severe cases, full academic withdrawal (Sanders, Mair and James, 2016). Several studies highlight the ramifications of poor attendance rates such as decreased performance in studies and potential threats to both the mental well-being and future employment prospects of the students (Macfarlane, 2013). Following cases of academic failure or withdrawal not only affects the student, but the institution itself as well.

The negative affects the institutions suffer because of the previously mentioned attendance issues have been increasingly observed and documented, raising concerns over student participation and engagement (Oldfield *et al.*, 2017). This trend has significant implications for institutions and lecturers, as the retention rates of the students tends to heavily affect the reputation and credibility of the institution and those who work for it. This is caused by the impact they have on the financial stability of the universities since retention rates are becoming increasingly critical in quality assurance and performance evaluation for most (Yorke and Longden, 2007).

The correlation between attendance rates and academic performance is apparent at various levels of education. As recorded by the Department for Education (2023), students who maintain a high level of attendance at GCSE and A Level are consistently among the highest achievers. Such findings are not merely coincidental as they illustrate a pattern where engagement and success are linked. In higher education, where the coursework is more rigorous and centred around self-study, the importance of regular attendance becomes even more pronounced. This importance of attendance in higher education has been underscored by the comprehensive analysis conducted by Allen and Webber (2010), which reveals crucial observations into the direct and indirect benefits of consistent class attendance has on student performance. This research is particularly significant as it builds upon and critiques earlier studies such as Marburger (2006), which suggested that the impact of attendance policies on academic performance was minimal. This article is also particularly relevant to this project as the fact that Allen and Webber conducted their research within the UK context is a significant aspect that differentiates the outcomes of their study from Marburger's, as mentioned in their findings. The impact of regional differences on this research shows how geographical and educational differences can affect the effectiveness of attendance policies and their influence on academic performance.

Allen and Webber's findings challenge the claim of Marburger by demonstrating a more substantial correlation between attendance and exam scores. Their study indicates that each additional seminar attended can noticeably increase a student's exam performance, indicating a direct benefit of active participation in educational activities. This is a crucial finding, as Marburger's study had suggested only a slight negative impact on performance of 2% when mandatory attendance policies were removed, implying a relatively insignificant role for attendance in a student's achievement. Furthermore, Allen and Webber expand on the context of attendance by exploring how structured learning environments and clear, transparent attendance policies not only improve attendance trends but also significantly enhance a student's performance. They argue that these policies make the benefits of attendance clear, thus encouraging students to participate more actively in their timetabled sessions. This approach contrasts with Marburger's conclusions, which did not fully capture the potential of well-implemented attendance policies in enhancing academic outcomes.

However, the debate continues over the motivations behind students' choices to attend or miss classes, with conclusions still elusive (Clark *et al.*, 2011; Kelly, 2012). The decision to attend is influenced by a number of reasons and inclinations, some within the influence of HE institutions and educators, meanwhile others are dictated by students' individual circumstances and life choices (Kelly, 2012). Oldfield (Oldfield *et al.*, 2017) offers a categorization of reasons for non-attendance as the following: encompassing feelings of isolation, negative perceptions of teaching staff, a fundamental approach to education, external pressures that preoccupy the students' time and importantly, the ease and appeal of attending lectures for students.

This emphasizes that there are subtler factors influencing students' attendance. It is essential to adopt an inclusive and situational approach when it comes to such a crucial subject, considering the personal and institutional elements that jointly shape the attendance behaviour. These factors also include personal traits, such as maturity and sociocultural background (Halpern, 2007); logistical issues like travel and weather conditions (Longhurst, 1999); the need for part-time employment (Kelly, 2012); and mental health challenges (Batı *et al.*, 2012).

Institutional factors also play a significant role for the students' attendance patterns. The effectiveness and simplicity of attendance policies and the strategies for early identification of withdrawal are critical in shaping attendance patterns. While explicit attendance policies may improve attendance rates (Snyder *et al.*, 2014), they must also accommodate students' individual circumstances. The educational dimension includes the mode and quality the session, online access to learning material, and the ease of attending the sessions. Thus, it is crucial that institutions address and enhance these factors within their control to better accommodate the diverse needs of students. In the context of this project, the goal is to simplify the attendance registration process, making it more inclusive and reliable, therefore reducing the burden on students who already have made the effort to physically attend lectures.

2.2 Attendance registration methods

Accurately tracking student attendance is a crucial aspect of administration for institutions. Throughout the years, various methods have been employed, each with their own advantages and limitations. Being the oldest known attendance recording system, the traditional paper-based approaches, such as roll-call and sign-in sheets remain common through numerous institutions because of their simplicity and the institutions desire to keep their working systems in place. However, these methods have significant drawbacks in terms of time efficiency, being prone to human errors, and being an administrative burden to educators.

Roll-call, where instructors verbally call out names to record presence, is extremely time-consuming, especially for large classes and is an unnecessary burden for educators. Sign-in sheets passed around the room are less disruptive but still slow and can serve as a distraction (Honglei, Song and Yang, 2016). Both methods require labour in the form of data entry to convert paper records to digital formats for record keeping and analytics. Simple human errors like misreading names, damaging the sign-in sheets, or students signing for others can render the attendance records inaccurate (Patel and Swaminarayan, 2014).

To address these issues, many institutions, such as MMU itself, have investigated electronic attendance tracking systems using token-based ID cards. Token-based solutions like ID cards and RFID card readers allow students to quickly self-register by presenting their ID cards (Younis *et al.*, 2012). While this method is far more efficient than paper-based methods, it requires an up-front investment for the system to function. Firstly, this method requires at least one dedicated RFID reader within each classroom. These RFID readers also require a constant and stable connection to the university network to function. In addition to this, each student has to possess an ID card containing RFID, which would add onto the initial cost. It would also cost the institution further if these cards were to be lost, which could be deducted from the students.

As a more recent method, biometric attendance systems identify individuals within the system directly through inherent physical or behavioural traits like fingerprints (Basheer and Raghu, 2012), voiceprints (Honglei, Song and Yang, 2016), or facial recognition (Honglei, Song and Yang, 2016). This method has several advantages over traditional methods such as eliminating the burden of managing physical token and possessing a much higher degree of resistance

against impersonation attempts. Similar to the token-based systems however, this method has an even higher cost of installation in terms of specialized hardware. In addition, fingerprint or voice capturing can still be disruptive during classes, which makes its cost harder to justify.

Distinctive to the other two biometric methods, facial recognition in particular is a highly efficient and fully automated approach. Classroom cameras can continuously capture images that are processed to identify students' faces against an enrolment database (Lukas *et al.*, 2016). While this method is convenient, facial recognition's accuracy remains imperfect due to environmental factors like lighting, angles, image quality and the need for large training datasets (Lim *et al.*, 2017). This method not only increases the cost even further as facial recognition has greater hardware requirements, it also could cause the students to be distracted or disturbed by the fact that they would be constantly recorded. Similarly, all biometric systems would necessitate the collection and storage of sensitive personal data, such as fingerprints, voiceprints, or facial scans, into the institution's system during enrolment. While this can initially be inconvenient, the primary concern is ethical. Not all students may feel comfortable providing their biometric data due to potential privacy breaches. Such perceptions are often rooted in fears of data leaks, which could lead to severe consequences. This ethical consideration is crucial when institutions decide to implement biometric systems.

2.3 Internet of Things

The concept of a network of smart devices has its roots tracing back to 1982 when a modified Coca-Cola vending machine at Carnegie Mellon University became the first ARPANET-connected device, capable of reporting its inventory and the temperature of its newly loaded drinks (Carnegie Mellon University, 2024). The mid-1990s saw several corporations propose early solutions similar to Internet of things, including Microsoft with its 'at Work' technology and Novell with its 'NEST' technology. Finally, in 1999, the term "Internet of things" was coined by Kevin Ashton of Procter & Gamble (Ashton, 2009), who later worked at MIT's Auto-ID Centre. Ashton, who preferred the term "Internet for things" instead (Day, 2015), emphasized the importance of radio-frequency identification (RFID) technology as a cornerstone for IoT, mentioning how he envisioned a future where computers could manage all individual items.

As a modern-day definition, the Internet of Things (IoT) refers to any network of interconnected physical devices that are equipped with sensors, software, or other technologies to collect and exchange data, therefore enabling these objects to interact with the internal workings of other devices or their external environments, without ever needing human intervention. Simple examples of this concept include thermostats and systems for monitoring and controlling Heating, Ventilation, and Air Conditioning (HVAC), which facilitate the operation of smart homes. However, the impact of IoT extends far beyond household environments. It plays a pivotal role in enhancing the overall efficiency and ease of life features across various sectors such as education, transportation, healthcare, industrial automation, and emergency management where human decision-making may prove challenging. IoT empowers the devices used in these sectors to acquire and analyse data, facilitating communication and coordination between them. This not only enables them to execute complex tasks but also transforms them from conventional hardware to intelligent devices. This transformation is driven by a number of underlying technologies, including embedded systems, communication technologies, sensor networks, and Internet protocols. Together, these technologies enable the creation of smart systems that perform specific functions within certain industries, known as vertical markets, while the broader, technology-driven capabilities such as data analytics and ubiquitous computing (or pervasive computing) contribute to domain-independent services, or horizontal markets (Vasseur and Kaufmann, 2010).

Throughout the approaching years, IoT is anticipated to significantly expand its influence in both domestic and business sectors, enhancing quality of life and contributing to global economic growth. For instance, smart homes equipped with IoT technologies can offer considerable conveniences in day-to-day life, such as automatically opening garage doors upon a resident's arrival, brewing coffee, and controlling climate systems, opening televisions, and other appliances. An example closer to this dissertation's context is the use of a Philips Hue smart light bulbs in the author's home, which is programmed to automatically switch on in the morning and off at night and has other similar functionalities which can be activated using the Philips hue mobile application, illustrating the practical benefits of IoT integration in daily life.



Figure 1: Philips hue light bulb

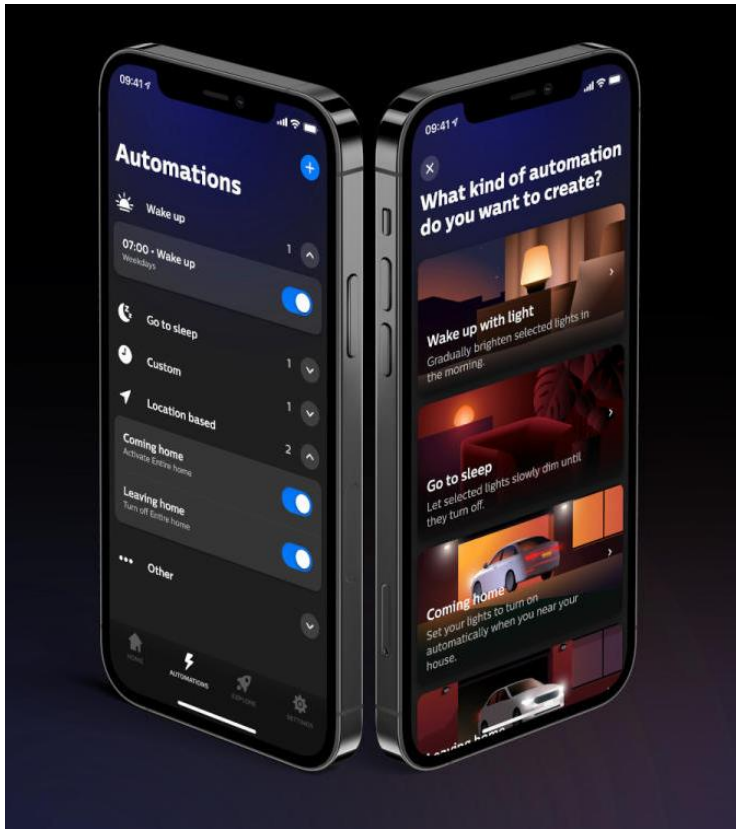


Figure 2: Philips hue mobile application



Figure 3: The developer's Philips hue lightbulb, dimmed automatically because of the late hour of this section being written

To capitalize on this potential, there must be a proportional growth in new technologies, innovations, and services to meet market demands and customer expectations. Economic growth driven by IoT-based services is considerable, especially in the healthcare and manufacturing sectors. These areas are projected to have the largest economic impact, with healthcare applications like mobile health and telecare, facilitating wellness, prevention, diagnosis, treatment, and monitoring services via electronic media, is expected to generate between \$1.1 trillion and \$2.5 trillion annually for the global economy by 2025. Overall, the annual economic impact of the IoT could range from \$2.7 trillion to \$6.2 trillion by 2025 (James Manyika *et al.*, 2013). These statistics display the rapid and potentially substantial growth of the IoT services and related industries in the near future. This progression offers a unique opportunity for traditional equipment and appliance manufacturers to evolve their products and services into smart systems.

2.4 RFID and NFC technology

One of the most common technologies used in IoT, Radio Frequency Identification (RFID), uses tags attached to objects to transmit data to RFID readers through electromagnetic fields. These tags, whether passive or active, are integral to various applications, ranging from simple inventory management to complex supply chain solutions. RFID tags are categorized mainly into active and passive types, each suited for different uses based on their power sources and mechanisms. Active tags are equipped with their own power sources, typically batteries, allowing them to transmit signals over greater distances and store more data. This makes them indispensable for tracking large items over long distances, such as in freight logistics. As opposed to active tags, passive tags do not possess an in-built power supply. Instead, they draw power from the reader's signal, which powers the tag's internal circuitry to send information back to the device. These tags are significantly cheaper, sometimes costing as little as pennies, and are much smaller, making them common in retail and inventory tracking (Weinstein, 2005).

As mentioned before, RFID has countless applications in numerous industries. For example, RFID's uses in supply chain management demonstrates its utility. In such cases, RFID tags track products from production through to delivery, improving efficiency and security. Unlike traditional barcodes, RFID does not require a line-of-sight to be scanned, can read multiple tags, and covers a larger range. These capabilities allow for real-time, automated updates throughout a product's journey, facilitating on-time delivery models that can significantly reduce overhead costs and improve service delivery in these sectors. Additionally, RFID technology has extended use in healthcare. Hospitals use RFID tags to monitor newborns to prevent catastrophes such as abductions or misplacements. Even official government branches such as the U.S. Food and Drug Administration (FDA) has approved using RFID to authenticate prescription drugs (Food and Drug Administration, 2008), enhancing the security and integrity of the pharmaceutical supply chain. Furthermore, RFID technology is used in the field of security for various identification and access control systems. For instance, buildings can employ RFID-tagged ID cards for entry access, and similar technology is embedded in credit cards and public transit fare systems to streamline transactions and to enhance user convenience.

Standards also play a crucial role in the widespread adoption and functionality of RFID systems. Several international standards, such as those from the International Standards Organization (ISO) and the Electronic Product Code (EPC) from EPCglobal, regulate various aspects of RFID technology, including encoding (International Standards Organization, 2023), data storage, system protocols, and the air interface. These standards ensure compatibility and functionality across different systems and borders, facilitating global implementation.

However, pure RFID technology has not advanced in every aspect, as the information capacity of passive RFID tags is generally around 256 bits of data, which is sufficient for basic identification and historical tracking, but it still is a major shortcoming of RFID technology. However, thanks to technological advancements, a new form of wireless communication called Near Field Communication (NFC) has emerged to overcome some of the shortcomings of traditional RFID systems while enhancing capabilities.

Near-Field Communication (NFC) is a technology built on the foundations of RFID, developed to enhance proximity communication with a focus on security and especially on user convenience. The official development of NFC technology was led by Philips and Sony, who recognized the potential for a secure, quick, and versatile communication standard. In September 2002, these two companies agreed to establish a technological specification (Sony Corporation and Philips Corporation, 2002), laying the groundwork for what would become NFC. This collaboration led to the formal approval of NFC as an ISO/IEC standard in December 2003, marking a major milestone in its development (Lange and Steck, 2014). By 2004, the NFC Forum was established by Nokia, Philips, and Sony with the goal of promoting the technology and ensuring the compatibility of NFC devices across different manufacturers. This association played a critical role in defining the standards and protocols that support NFC functionality today.

NFC technology operates at a frequency of 13.56 MHz within the globally available unlicensed radio Industrial, Scientific, and Medical (ISM) band, with data transfer rates between 46kbit/s up to 1.7Mbp/s. It adheres to international standards such as ISO/IEC 18000-3 for air interfaces. The hardware design of NFC involves an inductive coupling, where electromagnetic coils in NFC-enabled devices interact, allowing for three modes of communication: card emulation, reader/writer, and peer-to-peer. However, unlike the extended range of RFID, NFC technology generally has a range of less than 4 centimetres (NFC Forum, 2024). While this range can be

an issue in some industrial use cases, it acts as a form of security for public and commercial use.

Throughout the early-2010s, NFC technology was integrated into mobile devices with the reveal of Samsung nexus S and began to see practical applications, particularly in mobile payments and electronic ticketing systems. Following the increased integration of NFC technology in Android devices, Apple introduced support for NFC in its iPhone lineup starting with the iPhone 6 in 2014. While later iPhone iterations and iPhones operating systems provided the mobile phones with the ability to freely read and write to other NFC devices, it was initially only used for Apple Pay. This development by Apple was in response to Google's Google Pay service in android devices, highlighting the competitive landscape that NFC technology has fostered in the market since its launch. To give a more recent example, in 2020, nearly one-third of the adult population (32%, or around 17.3 million people) had registered for mobile payments, displaying an increase of 7.4 million users (75%) from the previous year (UK Finance, 2021). Among those registered, 84% carried out at least one transaction. Furthermore, half of these users made payments on a bi-weekly basis or more frequently. Consistent with recent trends, research indicates that younger demographics are more likely to adopt mobile payment solutions like Apple Pay, Google Pay, or Samsung Pay compared to older individuals, showing the increasing widespread use of such services.

Throughout the years, NFC has been integrated into various other consumer applications, such as Identity Tokens. In this field, NFC-enabled devices or tags can serve as electronic identity documents, similar to those used in ID cards, and as keycards for fare cards, transit passes, car keys, and access badges. The short range and robust encryption support of NFC make it a more secure and private option compared to traditional RFID systems, which makes it an ideal technology for this project considering its academic and educational context.

2.5 Similar work on attendance registration

The progress of attendance registration systems has been aided by the integration of advanced technologies aimed at improving accuracy, enhancing security, and improve user convenience. Several journal articles have been reviewed to find a number of innovative solutions, each tailored to tackle specific challenges within the domain of attendance registration in educational and organizational environments. These solutions span from NFC-based systems to sophisticated biometric and RFID implementations, and applications utilizing machine learning and facial recognition technologies.

2.5.1 NFC-Based Attendance Systems

Dixon and Abuzneid's (Dixon and Abuzneid, 2020) work on an NFC-based attendance system integrated with IoT highlights its ability to manage large-scale student populations efficiently. The IoT elements of the project were managed using MQTT protocol for lightweight messaging in a machine-to-machine context. This system not only facilitates quick check-ins but also significantly reduces possibilities for impersonation through secure NFC tags and readers that connect to a centralized cloud database, viewable via a web interface. However, this web interface also makes attendance data readily available to unauthorized personnel. The primary challenge here is the infrastructure investment required for widespread NFC reader installation and the dependency on continuous internet connectivity for the devices.

Tombeng's group (Tombeng *et al.*, 2023) has designed an NFC system that enhances real-time data processing, which is claimed to be crucial for dynamic educational settings. This system utilized student ID's and the NFC reader technology through the mobile devices for quick registration and a cloud computing services for its data storage. Firebase framework was selected for managing data in real-time and the web interfaces were developed using JavaScript, for enhanced user interaction. However, the system's functionality is based on the singular device that the attendance registration application is running on, and it still requires the students to carry around ID's which does not overcome the "buddy-punching" issue.

Mazumdar's groups' (Islam Mazumdar *et al.*, 2022) project featured a peer-to-peer NFC-based application, emphasizing ease of use. The system used Android Studio for the applications development, with peer-to-peer NFC technology where students would use their smartphones as cards to tap into the lecturer's tablet for attendance. One significant limitation of this project is its exclusive support for Android devices, which poses ethical concerns given that younger people in the U.K. are more likely to own iPhones (Kunst, 2024). This restriction could exclude a substantial portion of the student population and the application would need to be re-created in a framework that supports iPhones. Furthermore, the system depends on a single device which is controlled by the lecturer. This setup not only jeopardizes the process in case of an issue with the lecturer's device but also burdens the lecturer themselves with additional administrative tasks, potentially distracting them from their primary responsibilities.

2.5.2 Biometric and RFID-Based Solutions:

Memane's group (Memane *et al.*, 2022) advocates the creation of a fingerprint-based attendance system noted for its focus on security and precision. The system uses fingerprint scanners to capture data and managed it through a custom application developed in C#. The attendance data was then integrated into an existing ERP system, which highlights the system's capability to interface with broader institutional software. While fingerprint recognition is marginally less vulnerable to impersonation, it raises significant privacy issues and require direct physical interaction, which might not be ideal in scenarios demanding hygiene (e.g., post-pandemic contexts). The system also requires for all students to have their fingerprints scanned as well as the installation of fingerprint reader machines. These requirements contribute to the high initial cost of implementing the biometric system.

2.5.3 Machine Learning and Facial Recognition Technologies:

Narkhede's group (Narkhede *et al.*, 2023) introduces an AI-based system that employs facial recognition to automate attendance taking. The technology stack included Python for scripting, OpenCV for image processing, and Haar Cascades for face detection. The system was capable of real-time face recognition using webcams and integrated this data with a web application

built using Flask to allow administrative monitoring and reporting. Furthermore, the system featured capabilities such as generating charts and reports from the attendance data for lecturers as well as the ability to notify students via email when their attendance rates were below a set percentage. However, the system faces challenges such as variability in environmental conditions affecting image capture, like lighting and angles, and substantial privacy concerns due to the sensitive nature of biometric data collection by the institution. The system also faces issues similar to biometric methods as it has an even higher installation cost and still requires the faces of students to be recorded beforehand.

2.5.4 General Challenges and Technological Implications

Across these technologies, while automation reduces manual input and human error and enhances the reliability of attendance data, each method introduces similar challenges:

Technological Dependency: The necessity for specialized equipment (NFC readers, biometric scanners, cameras) and the associated implementation costs can cause issues for some institutions.

Privacy and Security: Systems that use personal biometric data would have to ensure robust protections to maintain user trust as well as having to comply with privacy regulations. These concerns would not be taken lightly by institutions as any breach in sensitive student data can spell disaster for any institution.

Accessibility and Equity: Dependence on a singular type of personal technology, like android smartphones, may exclude students with different preferences and would require them to acquire a device compatible with the current systems.

In conclusion, the proposed mobile application using NFC project should aim to mitigate these issues by balancing technological needs, privacy concerns, and user convenience to ensure a successful end product.

2.6 Development methodology

Selecting an appropriate methodology in software development is crucial as it guides the approach teams take to planning, designing, and implementing the project. Each methodology comes with its own set of principles, advantages, and disadvantages for different project needs and team sizes. Below is a summary of several popular methodologies and how they are typically used in various contexts.

2.6.1 Waterfall

The Waterfall methodology is a traditional, sequential design process, often used when the scope and requirements are unlikely to change, or, if the project does not have an established methodology. It follows a linear model that progresses through phases such as requirements, design, implementation, testing, and maintenance. Although simple and easy to implement accurately, the Waterfall model presents several disadvantages which makes it frowned upon. Its inflexible nature leads to limited end user involvement, resulting in feedback and testing phases that occur late in the development cycle. This delay puts the project at risk as the potential misunderstandings with user requirements are only discovered towards the end of the project (Gumiński, Dohn and OLOYEDE, 2023).

2.6.2 Scrum

Scrum is a methodology that promotes iterative development and team collaboration, where projects are divided into small units known as sprints. This methodology is highly flexible and promotes continuous feedback with the end-users, allowing the team to stay on point. It is built around a self-organizing team that includes roles like the Scrum Master, Product Owner, and the Development Team. Another key practice in scrum are the daily “stand-up” meetings where members of the team share updates to help the team and the project manager assess the progress they have made. Scrum facilitates regular adjustments based on stakeholder feedback, making it ideal for projects requiring flexibility and constant adaptation to changes (Gumiński, Dohn and OLOYEDE, 2023). However, it does have disadvantages such as scope creep due to its iterative nature and lack of a definite end-date. The success of a project under Scrum heavily relies on the cooperation of team members, requiring experienced team members for effective

execution. Additionally, the daily "stand-up" meetings can sometimes frustrate team members if seen as unproductive.

2.6.3 Extreme Programming (XP)

Extreme Programming (XP) is a framework that aims to produce high quality software, and a higher quality of life for the development team. Its key practices include frequent releases in short development cycles, which introduces checkpoints where new requirements can be adopted, pair programming, unit testing all of the code, and simplicity in code. Extreme Programming, similar to scrum, encourages end user involvement, believing that the requirements may change during the project. XP's focus on technical practices, as well as its emphasis on testing during the development makes it highly adaptive to varying requirements to create an accurate product. However, XP has its own disadvantages. It demands significant effort, requiring persistence, and creative thinking to continually adapt the system to meet the user's varying requirements. Furthermore, XP can be highly stressful for team members due to the pressure of deadlines, where coding and testing must be completed immediately, hence the term 'extreme' in Extreme Programming.

3. Chapter 3 – Requirements

This chapter details the comprehensive requirements for the development of the proposed mobile Attendance Registration project. The project will have two separate interfaces and thus requirements, an admin interface, and a student interface. However, there will be requirements that are shared among the different types of users when it comes to points such as accessibility or reliability. The requirements are categorized into Functional Requirements and Non-Functional Requirements, prioritized using the MoSCoW method: Must Have, Should Have, Could Have, and Won't Have.

3.1 Functional Requirements

3.1.1 User Requirements

Must Have:

User Authentication: Students and staff must be able to securely log in to the mobile application using their university credentials, or an imitation of their credentials for safety purposes.

NFC Attendance Registration: Students must register attendance by tapping their NFC-enabled mobile devices against the NFC tags within classrooms.

Real-time Attendance Updates: The system must update and display the attendance status and records in real-time within the mobile app.

Should Have:

Timetables: Users should be able to view their daily or weekly timetables within the app to view their upcoming classes.

Could have:

Notifications: Users could receive notifications for attendance confirmation, changes, and reminders for upcoming classes.

3.1.2 Admin Requirements

Must Have:

Attendance Monitoring: A screen within the admin dashboard to view the attendance data of individual students.

Viewing Database Tables: A screen for each table of the database within the admin dashboard for the admins to view the records.

Managing Database Tables: A screen for admins to insert, alter or delete individual rows of records from any of the tables within the database.

Creating NFC Tags: A screen for the admins to create or edit the information on the NFC tags.

Should Have:

Automatic Conflict Errors: Admins should not be allowed to edit or insert conflicting information on the management screens.

Could have:

Analytical Tools: Tools for analysing attendance patterns and generating reports to give insight as to how students behave.

3.1.3 Mobile Application Requirements

Must Have:

NFC Support: The app must support NFC functionality to communicate with the NFC tags placed in classrooms.

User Interface: Intuitive user interface for both students and staff, optimized for devices.

Should have:

User Experience: Features such as gestures, sound effects, haptic feedback, vibration etc. when appropriate buttons or actions are taken.

Could Have:

Offline Functionality: Ability to record attendance in offline mode, with data synchronization when the device reconnects to the internet.

3.1.4 API Requirements

Must Have:

Data Handling: API endpoints must handle requests for user authentication, data retrieval, and updates securely and efficiently.

Compatibility: API must be compatible with both the mobile app on both android and apple operating systems.

Should Have:

Futureproofing: API should also be compatible with web interfaces and desktop applications to be integrated with existing systems or for future applications.

3.1.5 Database Requirements

Must Have:

Functionality: The relationship between the tables and the stored procedures must work flawlessly with all the edge cases accounted for.

Scalability: The database must be scalable to accommodate increasing data volumes as the number of students grow.

Should have:

Backup and recovery: The database should have a backup and recovery plan in place to ensure data is not permanently lost in the event of a system failure or other catastrophic events.

3.2 Non-Functional Requirements

3.2.1 Performance

Must Have:

Response Time: The system should respond to user interactions within a few seconds (a maximum of 10 seconds) under typical usage conditions.

3.2.2 Reliability

Must Have:

Uptime: The system must guarantee 99% uptime outside of scheduled maintenance periods, which should be outside university hours.

Data Integrity: Mechanisms must be in place to ensure the accuracy and completeness of attendance data.

3.2.3 Usability

Must Have:

Ease of Use: The system must be user-friendly, requiring minimal or no training for students to perform essential functions such as registering attendance.

Should Have:

Accessibility: The application should be accessible to users with disabilities or special needs.

3.2.4 Security

Must Have:

Authentication: The system must implement strong authentication mechanisms to prevent unauthorized access within the mobile application.

4. Chapter 4 – Design

The design phase is a crucial aspect of any software development process, as it lays the foundation for translating conceptual requirements and constraints into a concrete design that guides the following implementation efforts. This following chapter details development methodology and version control, followed by the key design decisions and architectural considerations undertaken for the proposed system. This is achieved by justifying the choices made from the range of technologies and frameworks as well as displaying necessary information and design schemes about the system.

4.1 Development Methodology

Various software development methodologies were explored and evaluated within the literature review, each offering distinct advantages depending on project scope and team composition. The selection process for the most appropriate development approach for this project must consider that it is being undertaken by a single developer, and that the project's requirements may change due to the developer's initial unfamiliarity with frameworks such as React Native. Additionally, significant modifications to how the project functions may be necessitated based on feedback from end users, further influencing the need for a flexible and adaptive development methodology.

Traditional methodologies such as the Waterfall model were considered for its simplicity; however, its sequential phase dependency and inflexibility regarding the requirements were deemed unsuitable given the varying nature of the project and the challenge it presents in accommodating changes later in the development process. Similarly, Scrum, which is typically favoured for its collaborative team-based structure and its ability to adapt to changes, was eliminated due to the solitary nature of this project as it does not involve a team of developers or meetings that define Scrum projects.

Therefore, Extreme Programming (XP) was selected as the development methodology. XP advocates for frequent releases in short development cycles, which improves productivity and introduces a checkpoint at which new requirements to be adopted. This iterative approach is particularly beneficial for a single developer as it allows for continuous feedback and

adaptation of the project without the overhead of managing a larger team. Furthermore, XP emphasises a high level of involvement and communication with the client, and in this case the students, which ensures that the project is aligned closely with expectations of the students and the end product is suitable for its intended audience.

4.2 Version control

Version control is essential for managing the project's development, particularly as the developer was learning React Native. Although there are numerous version control frameworks, Git and GitHub will be used to ensure version control and backup as the developer had worked with them before.

Git will track changes in the codebase, allowing the developer to revert to previous versions when issues arose. This is crucial given the potential for errors due to learning a new framework, or because of clashing libraries within the project.

The workflow will involve local development with frequent commits, creating branches, testing changes, and merging stable code back into the main branch. Regular pushes to GitHub ensured that the latest changes were backed up and available to use across multiple devices.

Using Git and GitHub will allow the developer to experiment and learn without fear of breaking the project, ensuring a stable and manageable development process. The screenshot of the commits can be seen in the appendix A.

4.3 Choosing the NFC technology

As the aim of this project is to overcome the attendance registration problem using NFC technology within mobile phones, selecting the appropriate type of NFC technology is crucial. The primary options considered for the system include peer-to-peer NFC, NFC scanners, NFC emitters, and NFC tags. Each of these technologies offers distinct advantages and functionalities, but for the purposes of this project, the choice must align with cost-effectiveness, ease of implementation, and system efficiency as to be distinguishable from other modern attendance registration methods.

Peer-to-peer NFC technology allows two NFC-enabled devices to exchange data by bringing them into close proximity, such as two mobile phones. While this is ideal for applications requiring two-way communication, it is less suitable for a context where the interaction can be accomplished using unidirectional communication, as is the case with attendance registration.

NFC scanners and NFC emitters are more complex systems compared to other systems. NFC scanners read information from NFC tags or devices, and emitters can actively send data to other NFC-capable receivers. These technologies, while powerful, are typically more expensive and require more sophisticated infrastructure and maintenance in the long term. They also necessitate specialized hardware per classroom, which could complicate deployment in a setting like a university with numerous classrooms. A solution could have been procured to have the mobile phones emit the student information to NFC readers. However, such an approach would be an underutilization of the mobile phones' capabilities.

NFC tags occur to be the most practical choice for this project for several reasons. Firstly, NFC tags are the cost-effective technology when it comes IoT devices. They are inexpensive to purchase in bulk, which is an essential consideration given the need to equip numerous classrooms across numerous buildings across a campus. Secondly, their implementation is straightforward; NFC tags do not require an independent power source or complex setup, they can be easily affixed to classroom entrances and activated with minimal technical support using the mobile application itself, which will be discussed later on. Additionally, should any of these NFC tags become damaged or otherwise rendered unusable, they can be replaced easily and inexpensively.

Furthermore, the use of NFC tags aligns well with the project's architectural design, where the bulk of the processing power and complex operations are managed by a central server. This setup not only simplifies the physical infrastructure at the points of user interaction, but also enhances scalability and maintainability. By directing the computational workload to the server, the system can handle data processing and attendance analysis more efficiently, leveraging the powerful backend to manage large quantities of data and support complex queries that might arise from widespread use throughout the institution.

4.4 Designing the system architecture

The system architecture for the project is structured to support scalability, maintainability, and efficient communication between various software components. The architecture is divided into three main layers: the database layer, the application programming interface (API) layer, and the mobile application layer. This separation of concerns ensures that each component can be developed, tested, and maintained independently while still functioning cohesively as a part of the larger attendance system.

The overall architecture is designed as a three-tier model where the mobile application acts as the front end, and the API acts as the link between the mobile app and the database. The server will be hosted on a private hosting environment, ensuring secure and reliable access. The mobile application will be communicating with the API over HTTPS, providing a secure channel to transmit data. The API then interacts with the database to retrieve or store data, maintaining a centralised data repository that ensures data integrity and security.

The architecture is designed to be scalable, accommodating increases in user numbers or data volume without significant changes to the system. The use of cloud-based private hosting allows for flexible resource management and scalability. However, it is important to mention that the scalability of the system is closely linked with the hardware specifications of the hosting server, which in this case will not be in the scale required for an entire institution. The separation into distinct layers also facilitates easier maintenance and updates, as changes in one layer can be made independently of the others.

The diagram showing the system architecture can be seen below:

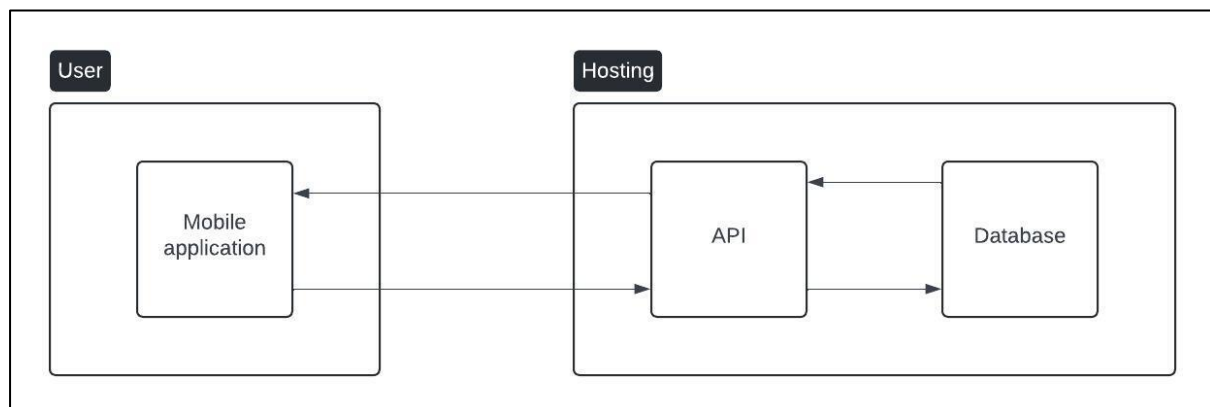


Figure 4: System architecture design diagram

4.5 Designing the database

The database for a large and crucial system such as attendance registration firstly requires determining the proper structure that will hold the system's data in an organized manner, ensuring that the system functions efficiently, securely, and is scalable. This design phase will involve the following steps: understanding the data requirements, determining the relationship structure, outlining the diagrams, and selecting the appropriate database management system (DBMS) software.

4.5.1 Comparing various databases

The initial step in designing the database is to clearly define the data requirements of the application. For an attendance registration system, this includes student information, classroom or building details, course and unit details, timetable sessions, and attendance records. This design only entails information relevant to the scope of this project, as a general university database would have more detailed information on each one of the mentioned records.

Secondly, the choice of relationship structures will depend on the nature of the data and the operational requirements of the application. Below are the three main database types:

1. **Relational Databases (using SQL):** These traditional databases store data in predefined schemas and tables. They use Structured Query Language (SQL) for managing and manipulating the data. SQL databases are highly structured, making them suitable for applications that require complex transactions and where data integrity and accuracy are critical. Examples include Microsoft SQL Server, MySQL, and PostgreSQL.
2. **Non-Relational Databases (using NoSQL):** NoSQL databases are more flexible in terms of how their data model's function. They are designed to handle large volumes of unstructured or semi-structured data and are ideal for rapid development and handling applications that require scalability across distributed architectures. Examples include MongoDB and Cassandra.
3. **Object Databases:** These databases store data in the form of objects, similar to object-oriented programming. Object databases are directly integrated with the language features, allowing objects to be stored and retrieved without conversion to a relational model. Examples include IBM Db2 and ObjectStore.

As attendance systems inherently deal with highly structured data, their data structure fits well into the relational model. A relational database allows for defining these entities and their attributes clearly in tables. Relationships between these entities (e.g., students attending courses on specific dates) are crucial. Relational databases excel in maintaining strong referential integrity through foreign keys (e.g., using course ids to link students to that course). This ensures that relationships are consistently maintained without data anomalies (e.g., a student being linked to a non-existent course) and allows for cascading updates.

4.5.2 Choosing the DBMS

Given the requirements and goals of the project, it was essential to choose a DBMS that aligns with the operational needs as without a stable database, the attendance system would simply fail to perform to an ideal standard. After reviewing various options, Microsoft SQL Server 2019 was chosen as the preferred system for several reasons:

- **Familiarity and Experience:** The developer has previous experience with relational databases and especially with Microsoft SQL Server. This familiarity minimizes the probability of errors during implementation of the database, as attempting to build a database while simultaneously learning its framework could lead to fundamental structural issues. Additionally, it reduces the learning curve and development time.
- **Relational Database Benefits:** Microsoft SQL Server is a powerful relational database management system known for its robust transactional support, which is crucial for maintaining the integrity of the attendance data. Its comprehensive SQL support ensures that complex queries and data relationships can be handled efficiently without issues.
- **Stability and Features:** SQL Server 2019 is one of the latest stable releases available during the project timeline. While SQL Server 2022 offers brand-new features, the stability and extensive documentation of SQL Server 2019 makes it a safer choice for this project, due to its stability without the risks associated with newer versions.
- **Cost and Licensing:** SQL Server offers a free edition that is more than sufficient for the data needs of this project and could even accommodate the scale of a real-life university records. This makes it a cost-effective option without the licensing expenses associated with other commercial database management systems like OracleDB.

- **Integration with Microsoft Technologies:** The hosting environment anticipated for deploying the application is expected to run Microsoft Windows Server OS, as it is the most common software to run on hosting. Staying within the Microsoft ecosystem ensures optimal performance of the DBMS, reducing potential compatibility issues.
- **Security Features:** SQL Server provides comprehensive security feature, including encryption, robust authentication mechanisms, and Universally Unique Identifier (UUID) generation, ensuring that student attendance data is securely managed and protected against unauthorized access.

4.5.3 Entity-Relationship Diagram

The next step, entity relationship modelling, is a crucial step where the defined data requirements are translated into a conceptual model. For the attendance system, the following Entity-Relationship Diagram (ERD) outlines the key entities and the relationships in between:

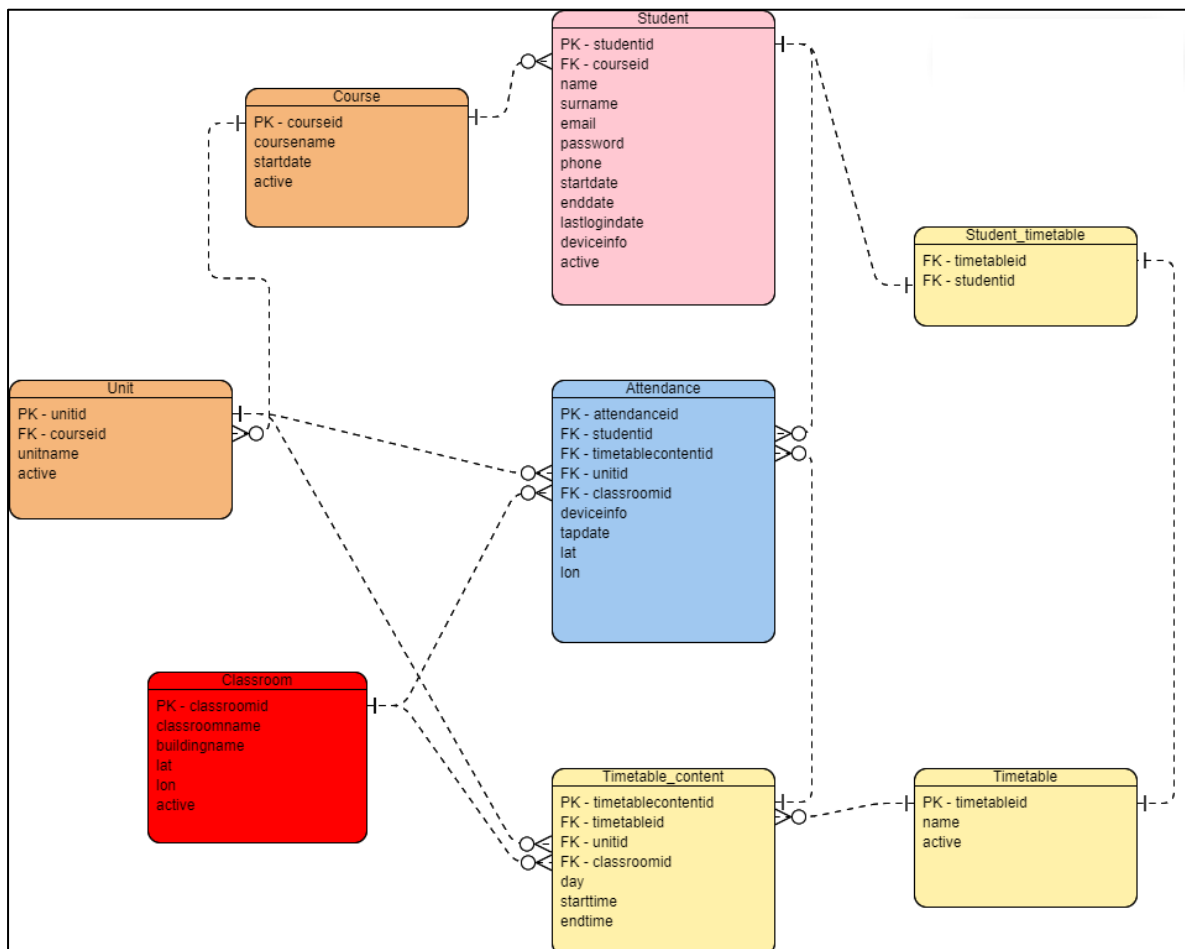


Figure 5: Entity Relationship Diagram for the Attendance database

4.6 Designing the API

An Application Programming Interface (API) serves as a crucial intermediary in software development, facilitating communication between different software components. In a three-tier architecture, the API plays a crucial role by connecting the presentation layer (frontend) with the logic layer (middle tier) and the data layer (backend). This separation allows the system to be modular, making scalability, and maintainability easier. By providing endpoints that expose specific functionalities, the API allows the frontend to interact with the backend without needing direct access to the database, ensuring a secure and efficient exchange of data.

4.6.1 Comparing various API types and frameworks

There are multiple approaches to designing and implementing APIs, each with its own advantages, disadvantages and use cases. Some common methods include:

- **Representational State Transfer (REST):** RESTful APIs use standard Hypertext Transfer Protocol Secure (HTTPS) methods, such as GET, POST, PUT, DELETE, and are known for their statelessness, scalability, and simplicity. They are resource-oriented, with each endpoint representing a specific resource within the system.
- **GraphQL:** This query language for APIs provides a more flexible and efficient way to fetch data. Clients can request exactly what they need, reducing over-fetching and under-fetching of data. However, it can be more complex to implement and maintain compared to other methods.
- **Simple Object Access Protocol (SOAP):** SOAP APIs are protocol-based and use Extensible Markup Language (XML) to format their messages and requests. They are highly standardized and support complex operations, thus they tend to be used within enterprise applications.

While designing the API, it is also essential to consider various modern cloud services that offer API hosting and backend functionalities:

- **Firebase:** Firebase is a cloud computing service by Google, which offers real-time databases, API services, and authentication within one system. Its flexibility makes it

excellent for rapid prototyping and development. It is particularly useful for web applications that require real-time data synchronization.

- Amazon Web Services (AWS): AWS provides a comprehensive set of cloud services, including API Gateways, Lambda, and Relational Database Service (RDS). It is known for its scalability, reliability, and vast array of tools for various specific needs. AWS supports serverless architectures, microservices, and large-scale applications. It is widely used in enterprise environments due to its global infrastructure.
- Microsoft Azure: Azure offers robust cloud services, including API Management, App Services, and SQL Databases. It integrates well with Microsoft technologies, making it a strong candidate for projects leveraging a Microsoft stack. Azure provides a range of unique services from virtual machines to AI and machine learning tools, making it versatile for different types of enterprise applications.

Although there were a variety of API options and cloud computing services to choose from, the REST API was a clear choice for a few reasons. Firstly, cloud services can become expensive, particularly as the application scales. Services like AWS and Azure offer a pay-as-you-go model, but the costs can add up quickly, especially for data storage, and API calls. As the project does not have an extensive budget, using one of these services would prove too costly, especially in development stages. Additionally, managing cloud services requires a deep understanding of the platform. Learning to navigate and effectively use these services can be time-consuming. For developers without prior experience, this learning curve can increase the likelihood of misconfiguring the service, which is not ideal given the context of this project. Oppositely, RESTful APIs are popular due to their simplicity, and scalability, making them ideal for small projects that may grow over time. Additionally, RESTful APIs are well documented, ensuring ample resources and guides for development and troubleshooting. RESTful APIs also provide a clear structure for building and maintaining endpoints, which is crucial for an attendance recording system that would regularly interact with multiple database tables.

However, the choice does not end there as there are several frameworks available for building REST APIs, each suited to different needs and development environments. Popular frameworks include:

- **Laravel:** A popular PHP framework known for its elegant syntax and robust features. It offers extensive built-in tools for authentication, routing, and data handling.
- **Express.js:** A minimalist Node.js framework that is lightweight and flexible. It provides a high-performance, unopinionated approach to API development.
- **Django:** A high-level Python framework that encourages rapid development and clean, pragmatic design. Django is powerful and comes with numerous built-in features.
- **FastAPI:** A modern, high-performance framework for building APIs with Python 3.6+ based on standard Python type hints. It is highly performant and user-friendly.
- **Flask:** A lightweight Python framework that is easy to set up and use. It is great for small projects but might lack the scalability and features for bigger projects.
- **ASP.NET Core:** A cross-platform, high-performance framework for building modern applications. ASP.NET Core is an improved version of ASP.NET, offering better performance, more flexible deployment, and a unified programming model.

4.6.2 Choosing the API framework

After evaluating various API frameworks, ASP.NET Core is chosen for several reasons:

- **Performance:** ASP.NET Core is known for its high performance and scalability, which is crucial for handling potentially large amounts of attendance data and concurrent requests.
- **Cross-Platform:** Unlike the traditional ASP.NET framework, ASP.NET Core is cross-platform, allowing deployment on Windows, macOS, and Linux servers.
- **Familiarity and Expertise:** The developer's familiarity with C# and ASP.NET ensures a smoother development process, reducing the learning curve and potential errors associated with less familiar frameworks.
- **Integration with Microsoft Ecosystem:** Given that the hosting environment will be running Microsoft Server Edition, using ASP.NET Core ensures seamless integration with other Microsoft products and services, such as SQL Server, Azure, and Active Directory.

- Security: ASP.NET Core provides robust security features, including built-in support for authentication and authorization, data protection, and HTTPS enforcement.
- Community and Support: ASP.NET Core has a strong community and extensive documentation, providing ample resources for troubleshooting and learning.

By choosing ASP.NET Core, the project leverages the developer's existing skills and ensures compatibility with the intended hosting environment, ultimately providing a robust and high-performance API for the attendance registration system.

4.6.3 Api endpoints

The API for the attendance recording system is designed to provide endpoints for various database tables and operations. The primary functions of the API will include:

- Authentication Endpoints: Handle user login, both for admins and students.
- Admin Endpoints: Create, Read, Update, Delete (CRUD) operations for all the tables.
- Attendance Endpoints: Record and retrieve attendance data, including marking attendance and generating attendance reports.
- Timetable Endpoints: retrieve timetable data for the student requesting it.

4.7 Designing the Mobile application

Designing the mobile application for the attendance registration system requires good attention to design as this will be the part of the system the users will view and judge. This design process involves selecting the right development framework, creating an intuitive user interface, and ensuring operational integration with the backend. This section will discuss the considerations and decisions made in choosing the development framework, as well as the design process for the user interface and experience.

4.7.1 Mobile development frameworks

Several popular mobile development frameworks were considered for the project, each with its strengths and weaknesses. Here are the main contenders:

Swift: Swift is the programming language developed by Apple for iOS and macOS applications. It offers powerful performance and seamless integration with Apple's ecosystem. Swift is meant to be a replacement to Objective C, which was the previous main language used to develop IOS applications, but still allows for integration with it and C++.

Limitations: Developing with Swift restricts the application to iOS devices, which is not suitable for a university setting where students use both iOS and Android devices.

Android Studio: Android Studio is the official Integrated Development Environment (IDE) for Android development, using Java or Kotlin.

Limitations: Similar to Swift, developing an app solely with Android Studio limits the application to Android devices, missing out on iOS users.

Flutter: Flutter, developed by Google, allows for the creation of natively compiled applications for mobile, web, and desktop from a single codebase. Flutter is developed using the dart language. It offers a rich set of pre-designed widgets and a fast development process.

Strengths: Flutter's cross-platform capabilities and extensive preinstalled UI toolkit make it a strong contender.

Microsoft Multi-platform App User Interface (.NET MAUI): Microsoft's mobile development framework, .NET MAUI, is the evolution of Xamarin, enabling developers to create cross-platform applications for Android, iOS, macOS, and Windows using a single codebase written in C# and Extensible Application Markup Language (XAML).

Strengths: .NET MAUI has seamless integration with other Microsoft technologies such as Azure, which makes it indispensable for a Microsoft tech stack.

React Native: React Native, developed by Facebook, enables developers to use React with native platform capabilities. It allows for building cross-platform apps with a single codebase using JSX, which is an XML-like extension to the JavaScript language. which even includes both the view and functionality in the same file, enabling a seamless development environment.

Strengths: Extensive guides, documentation, a wealth of libraries, and popularity. Most importantly, React Native offers NFC libraries essential for the project's requirements.

4.7.2 Choosing the appropriate framework

As context, when the project was still in its early design stages, the developer was inexperienced with mobile development. However, the developer planned to learn React Native through the "Mobile Computing" unit to smooth out the development process. Unknown to the developer, the curriculum was switched to .NET MAUI, making the plans of learning the framework through the unit obsolete. Despite the shift in the "Mobile Computing" unit to .NET MAUI, React Native remained the chosen framework for several reasons:

- **Extensive Documentation and Support:** React Native has a vast number of resources, tutorials, and community support available, which is invaluable for both learning and troubleshooting as the developer is inexperienced with mobile development.
- **Rich Ecosystem of Libraries:** The framework contains numerous libraries, including those specifically for NFC functionality, which are crucial for this attendance registration project. React native libraries also boast extensive documentation similar to the framework itself.
- **Cross-Platform Capabilities:** React Native's ability to develop applications for both iOS and Android ensure that the application can be used by all university students and system administrators, regardless of their device.

- **Popularity and reliability:** React Native has been released in 2015 and has been widely used in the industry ever since. With many successful applications built on it such as Amazon shopping, Discord, and Pinterest, proving it a reliable choice for years to come.

4.7.3 User interface and User experience

Creating an effective user interface (UI) and user experience (UX) design is critical for ensuring that the application is user-friendly and meets the needs of both students and administrators. This is also a crucial aspect in any applications development as no matter the capabilities of the project, its appearance is the aspect that dictates most of the opinions regarding it. Below are the design principles of the mobile application:

Simplicity: The application design will focus on simplicity to minimize the learning curve for new users. Essential features like logging in, registering attendance, viewing attendance records or timetables will be simple to access.

Consistency: Consistent use of colours, fonts, and interface style helps users quickly familiarize themselves with the app's interface. Functionalities of components such as tables and entry fields should also be consistent as not to confuse the users.

Accessibility: The application must be accessible to all users, including those with disabilities. This involves using readable fonts, high-contrast colour schemes, and providing alternative text for images if they are used.

Usability: Enhancing user experience through alerts, sound effects, haptic feedback, and vibrations can significantly improve the application's usability. These features provide an immediate response to user actions, ensuring that users are aware of the outcomes of their interactions and creating a more interactive experience.

The initial wireframe designs for the various screens of the mobile application's UI, created with careful attention to the aforementioned design principles, are displayed below:

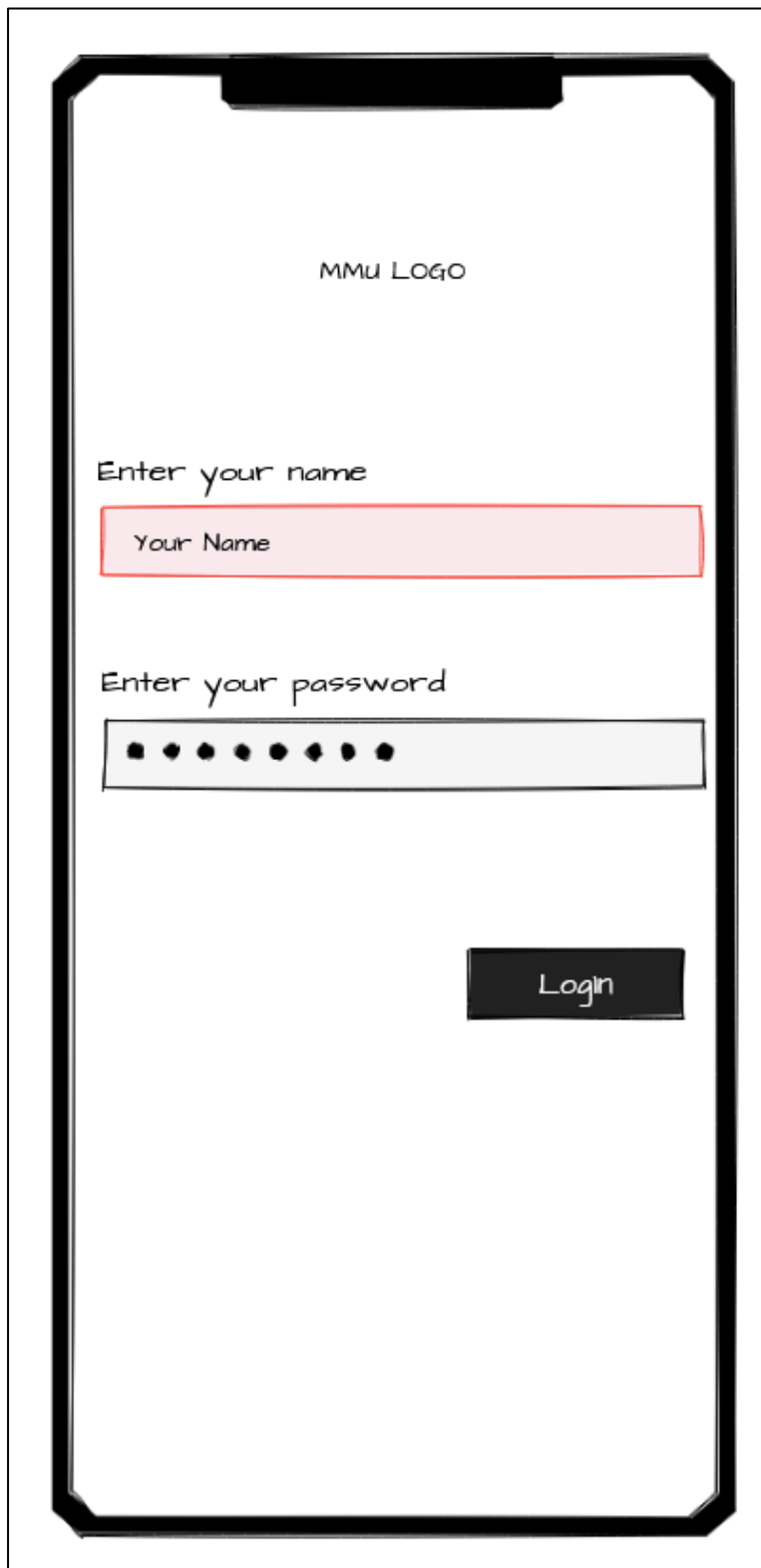


Figure 6: Login screen design



Figure 7: Student home screen design

Timetable

monday

lesson 1

time and place

>

tuesday

lesson 2

time and place

>

wednesday

lesson 3

time and place

>

thursday

Figure 8: Student timetable screen design

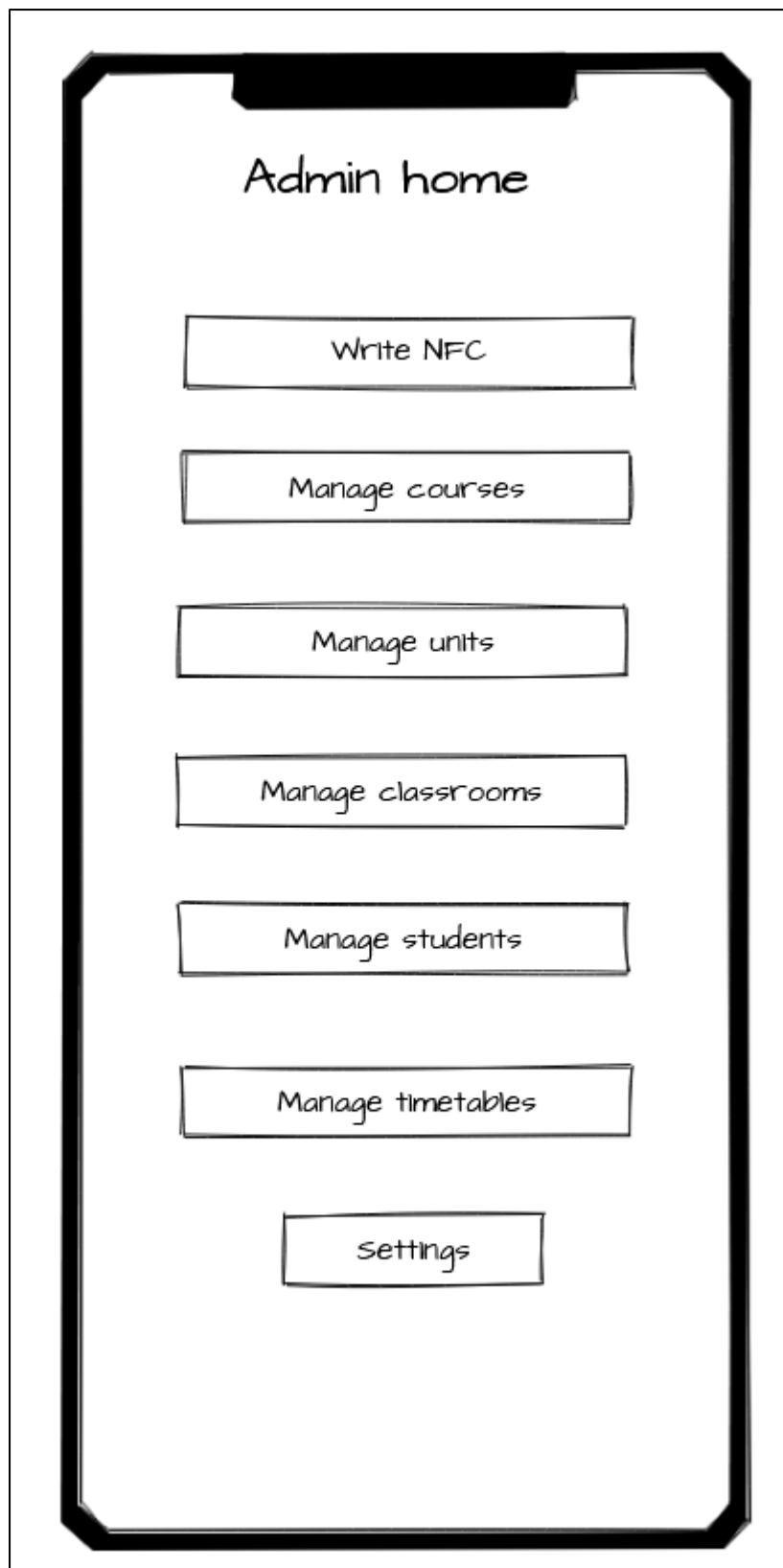


Figure 9: Admin home screen design

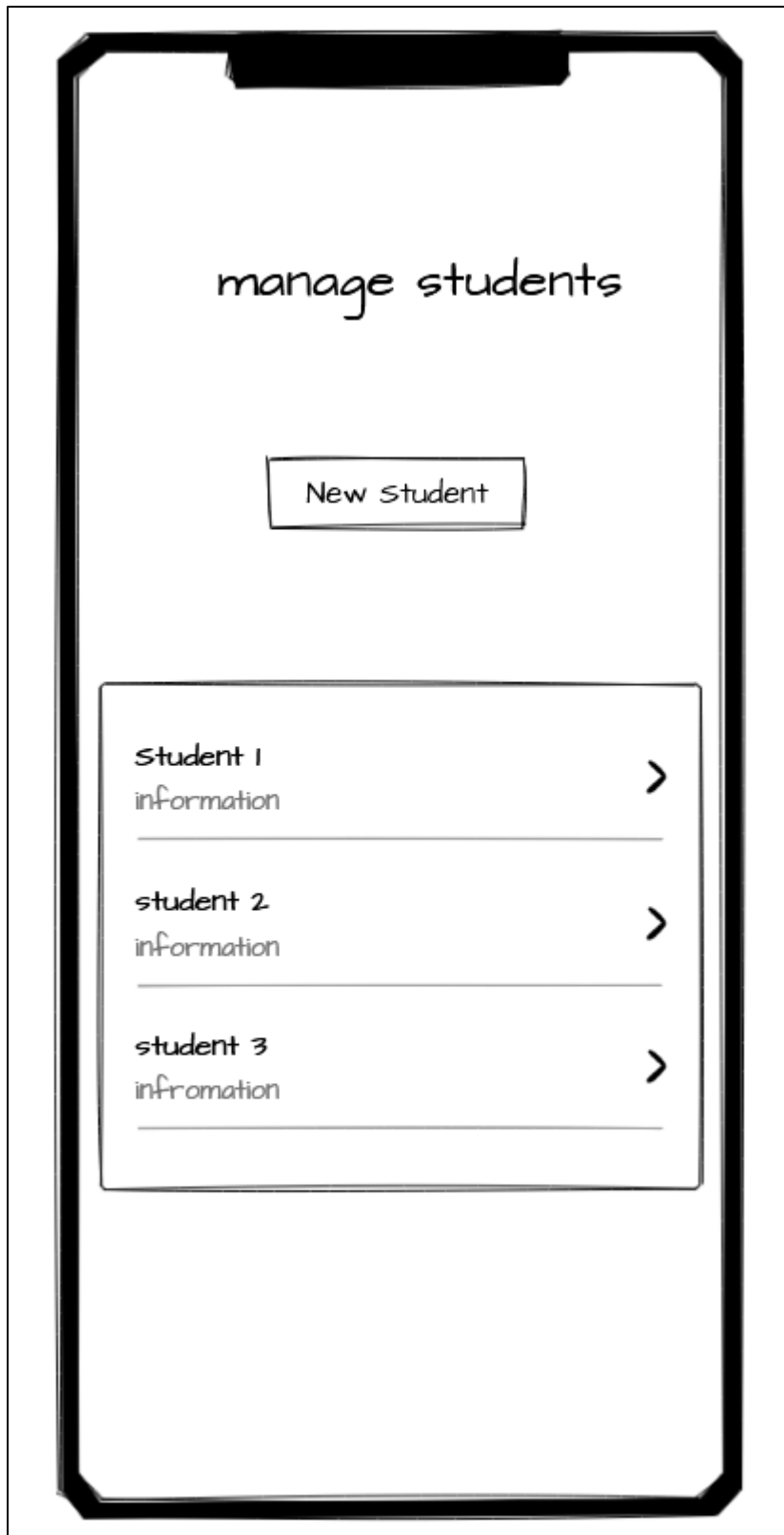


Figure 10: Manage table screen design, consistent across all tables

A hand-drawn sketch of a mobile application screen titled "Manage student". The screen is enclosed in a thick black border. At the top, the title "Manage student" is written in a casual, handwritten font. Below the title, there are six identical rows, each consisting of the word "field" followed by a light gray rectangular input box containing the text "info". At the bottom of the screen, there are two rectangular buttons: one labeled "Save" and one labeled "Delete".

Figure 11: Manage a particular row screen design

5. Chapter 5 – Implementation

This following chapter details the implementation phase of the mobile attendance registration system, where the design specifications outlined in the previous chapter were translated into a functional code. This phase involved overcoming various technical challenges and making decisions to ensure a robust and efficient system.

5.1 Creating the database

The creation of the database for the project is the foundation of the entire system as it cannot function without a place to store the attendance and student records. Utilizing Microsoft SQL Server 2019, the database was structured to support robust data management and retrieval processes that are central to the system's operations.

After Microsoft SQL Server 2019 was downloaded and installed onto the development computer, the Attendance database was created.

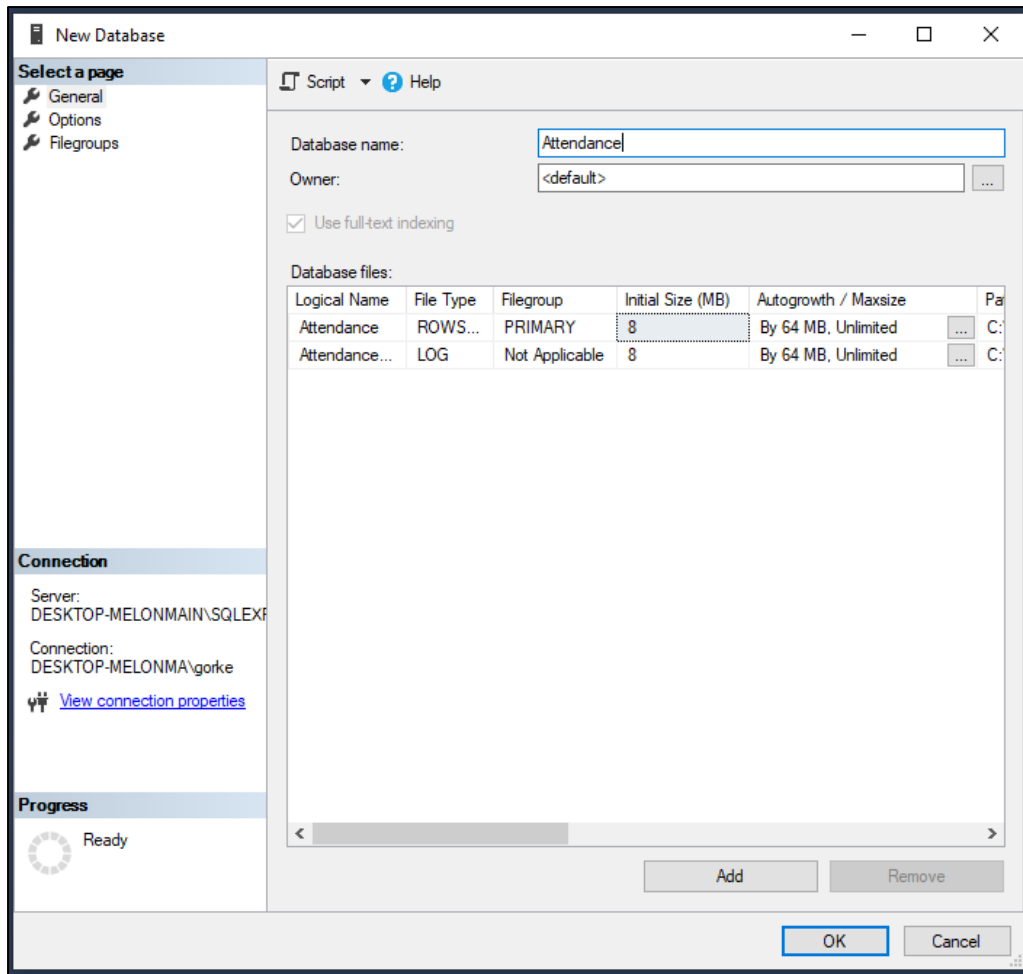


Figure 12: Creation of the attendance database

Following the initial setup of the database, the following structure was established with the creation of several key tables:

Courses: Stores information about the courses offered within the institution, including course ID, name, and course start date.

	Column Name	Data Type	Allow Nulls
►	courseid	int	<input type="checkbox"/>
	coursename	nvarchar(100)	<input type="checkbox"/>
	active	bit	<input type="checkbox"/>
	startdate	datetime	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Figure 13: Course table design view

Units: Each course consists of several units, and this table captures details about these units, such as unit ID, unit name and related course.

	Column Name	Data Type	Allow Nulls
►	unitid	int	<input type="checkbox"/>
	unitname	nvarchar(50)	<input type="checkbox"/>
	courseid	int	<input type="checkbox"/>
	active	bit	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 14: Unit table design view

Classrooms: Holds data about the physical locations where classes are held, including classroom ID, building, room number.

	Column Name	Data Type	Allow Nulls
►	classroomid	int	<input type="checkbox"/>
	classroomname	nvarchar(50)	<input type="checkbox"/>
	buildingname	nvarchar(50)	<input type="checkbox"/>
	active	bit	<input type="checkbox"/>
	lat	float	<input checked="" type="checkbox"/>
	lon	float	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Figure 15: Classroom table design view

Students: This table stores essential information about each student, such as student ID, name, contact details, and program enrolment date. It acts as a central entity that interacts with multiple other tables.

	Column Name	Data Type	Allow Nulls
▶	studentid	int	<input type="checkbox"/>
	uuid	uniqueidentifier	<input type="checkbox"/>
	name	nvarchar(50)	<input type="checkbox"/>
	surname	nvarchar(50)	<input type="checkbox"/>
	email	nvarchar(100)	<input checked="" type="checkbox"/>
	phone	nvarchar(11)	<input checked="" type="checkbox"/>
	courseid	int	<input type="checkbox"/>
	startdate	datetime	<input checked="" type="checkbox"/>
	enddate	datetime	<input checked="" type="checkbox"/>
	active	bit	<input type="checkbox"/>
	lastlogindate	datetime	<input checked="" type="checkbox"/>
	deviceinfo	varchar(MAX)	<input checked="" type="checkbox"/>
	password	nvarchar(100)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Figure 16: Student table design view

Student Timetable: Links the Students table to the Timetable table using IDs.

	Column Name	Data Type	Allow Nulls
▶	studentid	int	<input type="checkbox"/>
	timetableid	int	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 17: Student Timetable table design view

Timetable: Enables the timetable content to be grouped into named groups with ID's.

	Column Name	Data Type	Allow Nulls
▶	timetableid	int	<input type="checkbox"/>
	name	nvarchar(50)	<input type="checkbox"/>
	active	bit	<input type="checkbox"/>
			<input type="checkbox"/>

Figure 18: Timetable table design view

Timetable Content: Contains detailed information about each scheduled session within the timetables, such as the date, time, and classroom. It ensures that the timetable structure is flexible and can accommodate changes.

	Column Name	Data Type	Allow Nulls
▶	timetablecontentid	int	<input type="checkbox"/>
	timetableid	int	<input type="checkbox"/>
	unitid	int	<input type="checkbox"/>
	day	int	<input type="checkbox"/>
	starttime	time(7)	<input type="checkbox"/>
	endtime	time(7)	<input type="checkbox"/>
	classroomid	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Figure 19: Timetable Content table design view

Attendance: This critical table records each student’s attendance data. It logs entries every time a student checks in using their NFC-enabled device, referencing their ID, the specific classroom, the location of the device and the timestamp of entry as well as other identifiable information.

	Column Name	Data Type	Allow Nulls
▶	attendanceid	int	<input type="checkbox"/>
	studentid	int	<input type="checkbox"/>
	courseid	int	<input type="checkbox"/>
	unitid	int	<input type="checkbox"/>
	classroomid	int	<input type="checkbox"/>
	tapdate	datetime	<input type="checkbox"/>
	deviceinfo	nvarchar(MAX)	<input type="checkbox"/>
	lat	float	<input checked="" type="checkbox"/>
	lon	float	<input checked="" type="checkbox"/>
	timetableid	int	<input checked="" type="checkbox"/>
	timetablecontentid	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

Figure 20: Attendance table design view

The implementation of the database has generally stayed true to the original design outlined in the project’s design section with the addition of a few informational columns. This adherence was crucial for maintaining consistency across the development process and ensuring that the back-end infrastructure supports the application's functional requirements reliably. Relationships between the tables were also carefully established using foreign keys to maintain referential integrity and to successfully facilitate complex queries that involve multiple tables.

5.2 Stored procedures

In the implementation of the database for the attendance system, two stored procedures play a crucial role: `sp_attendance` and `sp_get_attendance`. These procedures manage the core functionalities of registering attendance and retrieving attendance data for the students.

5.2.1 `sp_attendance`

The `sp_attendance` stored procedure is designed to record a student's attendance. When a student registers their attendance, the procedure checks if the student is in the correct classroom at the appropriate time. It also ensures that the student has not already tapped in for that session on the same day. This is crucial to prevent duplicate records and ensure data integrity. The procedure uses multiple parameters, including student ID, classroom ID, device information, and geographic coordinates (latitude and longitude), to log attendance accurately.

Here's the code for the `sp_attendance` stored procedure:

```
ALTER PROCEDURE [dbo].[sp_attendance](
    --input parameters
    @studentId int = 1,
    @classroomid int = 1,
    @deviceinfo nvarchar(MAX) = '',
    @lat float,
    @lon float
)
AS
BEGIN

    --local variables declared here
    Declare @courseid int = 0
    declare @unitid int = 0
    declare @timetableid int = 0
    declare @timetablecontentid int=0

    SELECT @timetablecontentid=timetable_content.timetablecontentid,
    @timetableid=timetable_content.timetableid, @courseid=unit.courseid,
    @unitid=unit.unitid FROM timetable_content
    JOIN student_timetable ON timetable_content.timetableid =
    student_timetable.timetableid
    JOIN unit ON timetable_content.unitid = unit.unitid
    WHERE student_timetable.studentid = @studentId AND day =
    DATEPART(DW,GETDATE()-1) AND timetable_content.classroomid = @classroomid
    AND starttime< cast(GETDATE() as time)
    AND endtime> cast(GETDATE() as time)
```

```

if (@timetablecontentid>0)
BEGIN

    declare @InsertedBefore int = 0;

    SELECT @InsertedBefore = count(*) FROM attendance
    JOIN timetable_content ON attendance.timetablecontentid =
timetable_content.timetablecontentid
    WHERE attendance.studentid = @studentId
    AND attendance.classroomid = @classroomid
    AND attendance.timetablecontentid = @timetablecontentid
    AND DAY(attendance.tapdate) = DAY(GETDATE())
    AND MONTH(attendance.tapdate) = MONTH(GETDATE())
    AND YEAR(attendance.tapdate) = YEAR(GETDATE())

    if(@InsertedBefore>0)
    BEGIN
        select 409 AS ID, 'Already tapped in!!' as MSG
    END
    ELSE
    BEGIN
        INSERT INTO attendance (studentid, courseid, unitid,
classroomid, tapdate, deviceinfo, lat, lon, timetableid, timetablecontentid)
        VALUES (@studentId, @courseid, @unitid, @classroomid,
GETDATE(), @deviceinfo, @lat, @lon, @timetableid, @timetablecontentid)
        select 200 AS ID, 'Inserted new row' as MSG
    END

END
ELSE
BEGIN
    select 404 AS ID, 'Invalid Classroom or time!!' as MSG
END

END

```

5.2.2 sp_get_attendance

The sp_get_attendance stored procedure retrieves the attendance rate and other attendance information for a student within a specified date range. It calculates the total number of lessons, attended sessions, and missed sessions, and computes the attendance percentage. This procedure is essential for generating reports for the students. It also could be used by admins for monitoring student attendance trends over time in future implementations.

Here's the code for the sp_get_attendance stored procedure:

```
ALTER PROCEDURE [dbo].[sp_get_attendance](
    --input parameters
    @studentId int = 1,
    @startdate datetime,
    @enddate datetime
)
AS
BEGIN

    declare @lessonCount int=0;
    declare @attendanceCount int=0;

    delete report where studentid=@studentid

    declare @daydiff int = DATEDIFF(day, @startdate,@enddate)

    DECLARE @i int = 0
    DECLARE @day int = 0
    --DECLARE @oncount int = 0

    DECLARE @lessonday int
    DECLARE @st time
    DECLARE @et time
    DECLARE @unitid int
    declare @currentDay datetime
    declare @missedCount int=0

    DECLARE day_cursor CURSOR FOR
    SELECT day as lessonday,starttime,endtime,unitid from timetable_content c join
    student_timetable s on c.timetableid=s.timetableid and studentid=@studentid

    --select tapdate,cast(tapdate as date) , * from attendance where studentid=1

    OPEN day_cursor;

    FETCH NEXT FROM day_cursor INTO @lessonday,@st,@et,@unitid

    -- Check @@FETCH_STATUS to see if there are any more rows to fetch.
    WHILE @@FETCH_STATUS = 0
    BEGIN
```



```

WHILE @i < @daydiff
BEGIN
    select @currentDay=DATEADD(day, @i, @startdate)
    SELECT @day= DATEPART(dw,@currentDay)-1

    if ((@day>=1) AND (@day<=5)) AND (@day=@lessontday)
    BEGIN

        declare @st_ datetime = CAST(cast(@currentDay as date) AS DATETIME) +
cast (@st as datetime)

        declare @et_ datetime = CAST(cast(@currentDay as date) AS DATETIME) +
cast (@et as datetime)

        declare @notAttended int=0;
        select @notAttended=count (*) from attendance where
studentid=@studentid and tapdate between @st_ and @et_

        if (@notAttended=0)
        BEGIN
            set @missedCount = @missedCount+1;

            --print @unitid
            --print @currentDay

            Insert into report(studentid,unitid,date) values (@studentid
,@unitid,cast(@currentDay as date))

            END

            SET @lessonCount = @lessonCount +1

        END

        SET @i = @i + 1
    END
    SET @i =0

    -- This is executed as long as the previous fetch succeeds.
    FETCH NEXT FROM day_cursor
    INTO @lessontday,@st,@et,@unitid
END

--select @lessonCount

CLOSE day_cursor;
DEALLOCATE day_cursor;

select @attendanceCount =count(*) from attendance where studentid=@studentid and
tapdate>=@startdate and tapdate<=@enddate

if (@lessonCount>0)
BEGIN

```

```
select @lessonCount as lessonCount,@attendanceCount as attendanceCount ,
@missedCount as missedCount, ((@attendanceCount * 100) / @lessonCount) as
percentage
END

END
```

5.3 Creating the API

To implement the API for the mobile NFC attendance registration project, ASP.NET Core was used due to its robust performance, scalability, and seamless integration with other Microsoft technologies. This section outlines the process of creating the API, including the technologies used, the structure of the program, and the specifics of the endpoints.

Firstly, the project was created on visual studio 2022 by selecting “ASP.NET Core Web API” project with the following options:

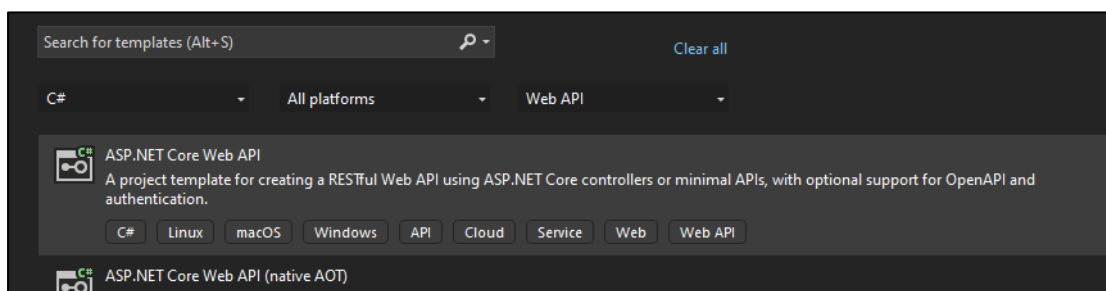


Figure 21: ASP.NET Core Web API project creation

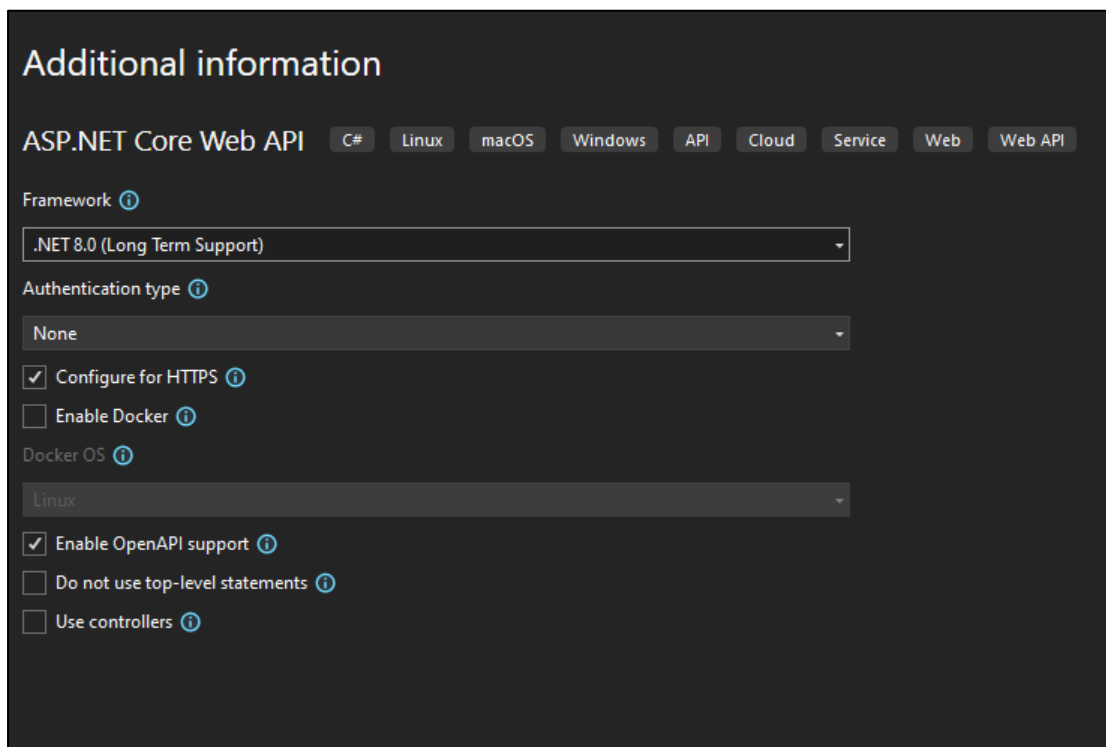


Figure 22: ASP.NET Core Web API project configuration

After creating the project, unnecessary example code was removed to have the following boilerplate structure:

```
var builder = WebApplication.CreateBuilder(args);

// Add services to the container.
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();

var app = builder.Build();

// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.Run();
```

Figure 23: Program.cs file after deleting example code

With the base file ready to code, Dapper and Microsoft.Data.SqlClient libraries were installed through the NuGet package manager:

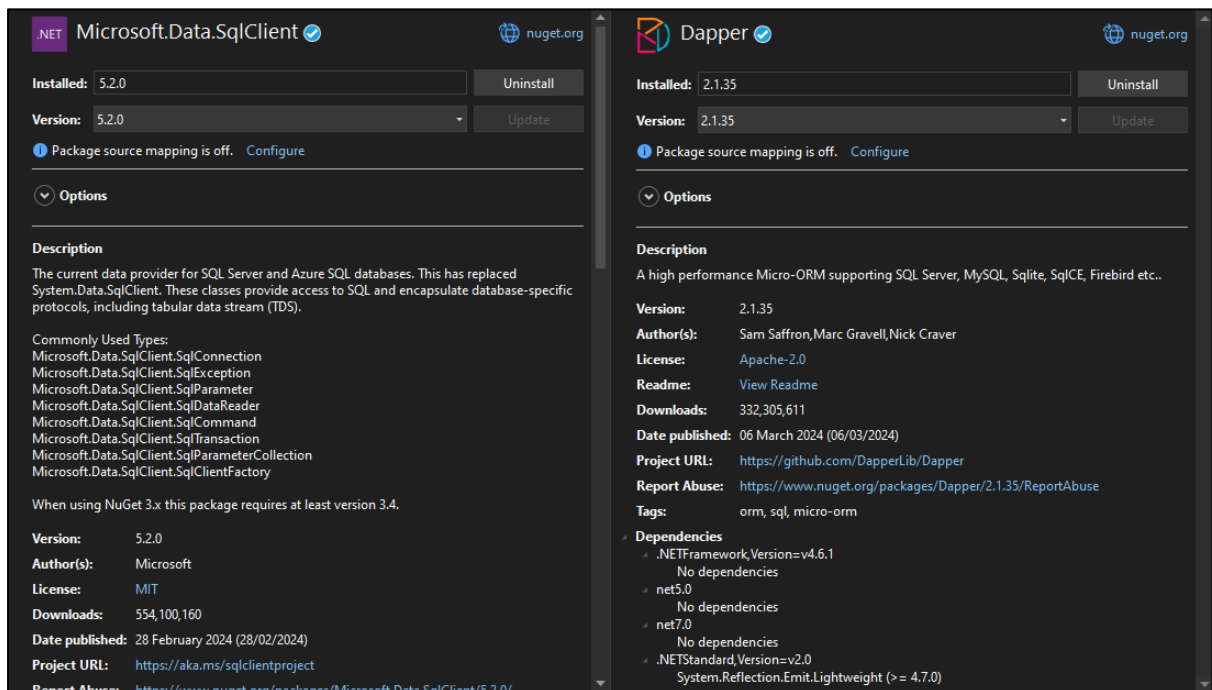


Figure 24: Dapper and Microsoft.Data.SqlClient libraries installed

With the necessary libraries installed, the connection string for the local database was added to the project in the appsettings.Development.json file:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=DESKTOP-MELONMA\\SQLEXPRESS;Database=Attendance;Trusted_Connection=True;Encrypt=False"
  },
}
```

Figure 25: Connection string

Afterwards, the models for the tables were created for the CRUD operations in the following manner with the appropriate fields and variable types:

```
namespace AttendanceAPI.Models
{
    public class Course
    {
        public int courseid { get; set; }
        public string coursename { get; set; }
        public bool active { get; set; }
        public DateTime startdate { get; set; }
    }
}
```

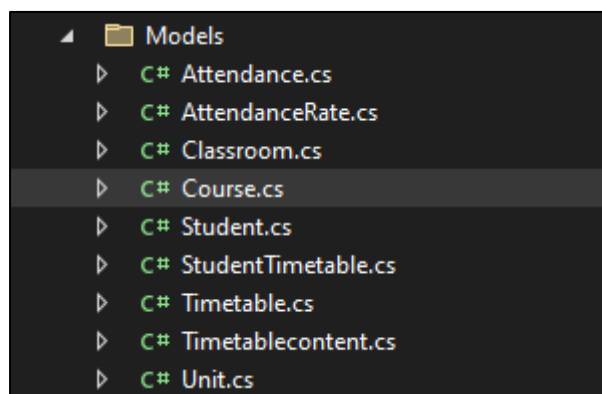


Figure 26: All the API models representing the tables

After the models were created, a GET endpoint for the tables were created in program.cs to test the functionality of the API:

```
app.MapGet("", async (IConfiguration configuration) =>
{
    //establishes connection to database
    var connectionString =
configuration.GetConnectionString("DefaultConnection")!;

    using var connection = new SqlConnection(connectionString);

    //declare queries as strings
    const string sql = "SELECT courseid, coursename, active, startdate FROM
course";

    //dapper function, converts type to a course type
    var courses = await connection.QueryAsync<Course>(sql);

    //returns the server code
    return Results.Ok(courses);
});
```

After launching the project, the course endpoint seemed to be functional ,which can be seen down below:

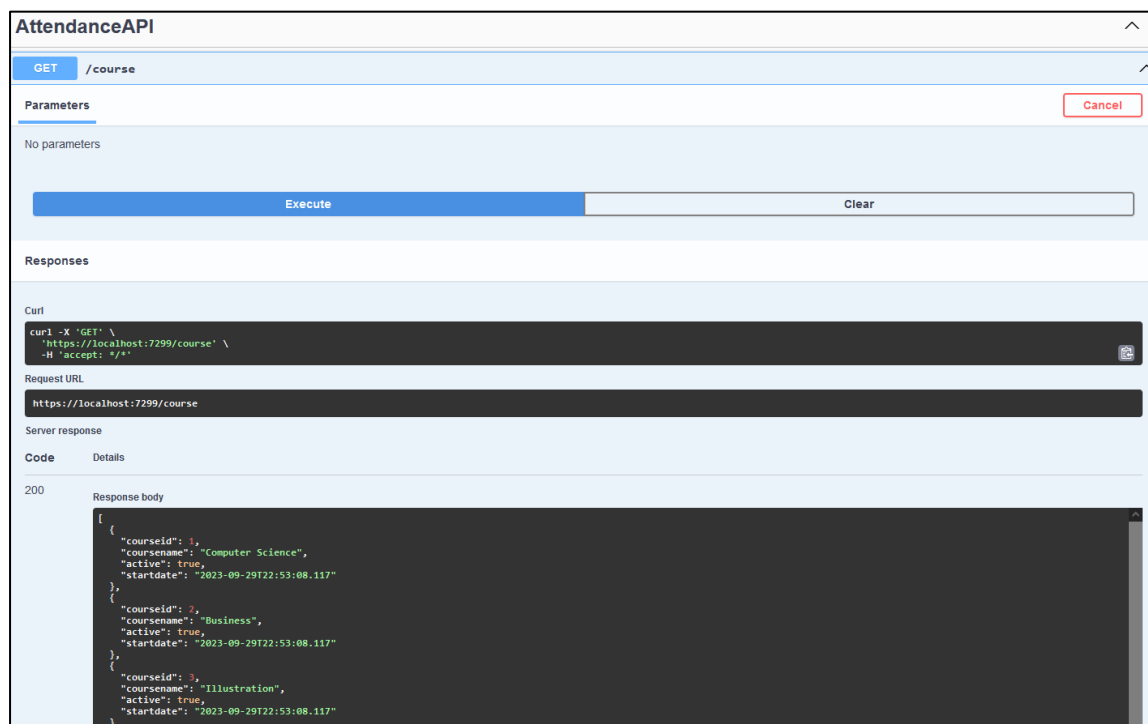


Figure 27: Course endpoint in swagger UI

Before adding and testing any more endpoints, a folder to store the endpoints was created to modularize the code. The folder, the course endpoint and the program.cs can be seen below:

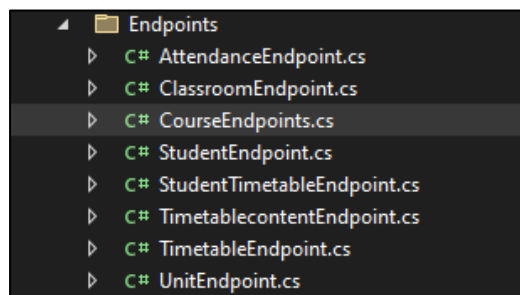


Figure 28: API Endpoints

```
using AttendanceAPI.Models;
using AttendanceAPI.Services;
using Dapper;
using Microsoft.Data.SqlClient;

namespace AttendanceAPI.Endpoints
{
    public static class CourseEndpoints
    {
        //start

        public static void MapCourseEndpoints(this IEndpointRouteBuilder
builder)
        {
            //start

            builder.MapGet("courses", async (IConfiguration configuration) =>
            {
                //establishes connection to database
                var connectionString =
configuration.GetConnectionString("DefaultConnection")!;

                using var connection = new SqlConnection(connectionString);

                //declare queries as strings
                const string sql = "SELECT courseid, coursename, active,
startdate FROM course";

                //dapper function, converts type to a course type
                var courses = await connection.QueryAsync<Course>(sql);

                //returns the server code
                return Results.Ok(courses);
            });
        }
    }
}

//Using the endpoints in program.cs instead
app.MapCourseEndpoints();
```

A helper service for creating the SQL connections were created to simplify the code even further and to focus on writing the queries instead of the connections in each endpoint. The SQL connection factory code, the modified program.cs and endpoint can be seen below:

```
using Microsoft.Data.SqlClient;

namespace AttendanceAPI.Services
{
    public class SqlConnectionFactory
    {
        //represents the connection string
        private readonly string _connectionString;

        //Constructor
        public SqlConnectionFactory(string connectionString)
        {
            _connectionString = connectionString;
        }

        //Returns a sql connection
        public SqlConnection Create()
        {
            return new SqlConnection(_connectionString);
        }
    }
}
```

```
//Takes care of sql connection factory in the program.cs
builder.Services.AddSingleton(ServiceProvider => {
    var config = ServiceProvider.GetRequiredService<IConfiguration>();

    var connectionString = config.GetConnectionString("DefaultConnection") ??
    throw new ApplicationException("The connection string is null");

    return new SqlConnectionFactory(connectionString);
});
```

```
//Simplifies the queries
builder.MapGet("courses", async (SqlConnectionFactory factory) =>
{
    //establishes connection to database, using nugget package
    using var connection = factory.Create();
```



```

//declare querys as strings
const string sql = "SELECT courseid, coursename, active, startdate FROM
course";

//Dapper function, converts type to a course type
var courses = await connection.QueryAsync<Course>(sql);

//Returns the server code
return Results.Ok(courses);

});

```

As a last improvement, all endpoints within an endpoints file were grouped to simplify the routes. Afterwards, all the CRUD operations were added to the endpoints. The final version of the course endpoints file with the GET, POST, PUT and DELETE mapped for the course can be seen down below:

```

using AttendanceAPI.Models;
using AttendanceAPI.Services;
using Dapper;
using Microsoft.Data.SqlClient;

namespace AttendanceAPI.Endpoints
{
    public static class CourseEndpoints
    {
        //start

        public static void MapCourseEndpoints(this IEndpointRouteBuilder
builder)
        {
            //start

            //create a routetrough
            var group = builder.MapGroup("course");

            //Simplifies the queries
            group.MapGet("", async (SqlConnectionFactory factory) =>
            {

                //establishes connection to database, using nugget package
                using var connection = factory.Create();

                //declare querys as strings
                const string sql = "SELECT courseid, coursename, active,
startdate FROM course";

                //Dapper function, converts type to a course type
                var courses = await connection.QueryAsync<Course>(sql);

```

```

        //Returns the server code
        return Results.Ok(courses);
    });

    //fetch single course from database using its ID
    group.MapGet("{courseid}", async (int id, SqlConnectionFactory
factory) =>
    {
        using var connection = factory.Create();

        //pass as paramater
        const string sql = "SELECT * FROM course WHERE courseid = @ID";

        var course = await
connection.QuerySingleOrDefaultAsync<Course>(sql, new {ID = id});

        //the ? and : act as a if statement
        return course is not null ? Results.Ok(course) :
Results.NotFound();
    });

    //Create a course
    group.MapPost("", async (Course course, SqlConnectionFactory
factory) =>
    {
        using var connection = factory.Create();

        const string sql = "INSERT INTO course (coursename,
startdate,active) VALUES (@coursename,@startdate, @active)";
        await connection.ExecuteAsync(sql, course);

        return Results.Ok();
    });

    //Edit a course
    group.MapPut("{courseid}", async (int ID, Course course,
SqlConnectionFactory factory) =>
    {
        using var connection = factory.Create();

        course.courseid = ID;

        const string sql = "UPDATE course SET coursename = @coursename,
startdate=@startdate, active=@active WHERE courseid = @courseid";

        await connection.ExecuteAsync(sql, course);

        return Results.NoContent();
    });

    //Delete a course
    group.MapDelete("{courseid}", async (int id, SqlConnectionFactory
factory) =>
    {
        using var connection = factory.Create();

        const string sql = "DELETE FROM course WHERE courseid =
@courseid";

```

```

await connection.ExecuteAsync(sql, new { courseid = id });
        return Results.NoContent();
    });
    //end
}
//end
}
}

```

Rest of the API endpoints were coded same way the course endpoint was with the appropriate values and variables for the exception of the Attendance endpoint. The attendance point utilized the two stored procedures aforementioned in the database section, which needed parameters to be sent differently from default queries. Those two attendance endpoints can be seen below:

```

//register student attendance
group.MapPost("Attendance/{studentid, classroomid, deviceinfo, lat, lon}",
async (int _studentid, int _classroomid, string _deviceinfo, float _lat, float
_lon, SqlConnectionFactory factory) =>
{
    var spName = "sp_attendance";
    var parameters = new DynamicParameters();

    parameters.Add("studentId", _studentid, DbType.Int32,
ParameterDirection.Input);
    parameters.Add("classroomid", _classroomid, DbType.Int32,
ParameterDirection.Input);
    parameters.Add("deviceinfo", _deviceinfo, DbType.String,
ParameterDirection.Input);
    parameters.Add("lat", _lat, DbType.Double, ParameterDirection.Input);
    parameters.Add("lon", _lon, DbType.Double, ParameterDirection.Input);

    using var connection = factory.Create();

    var attendance = await
connection.QueryFirstOrDefaultAsync<Attendance>(spName, parameters,
commandType: CommandType.StoredProcedure);

    //the ? and : act as a if statement
    return attendance is not null ? Results.Ok(attendance) :
Results.NotFound();
});

```

```

//fetch student attendance rate
group.MapPost("GetAttendance/{studentid, startdate, enddate}", async (int
_studentid, DateTime _startdate, DateTime _enddate, SqlConnectionFactory
factory) =>
{
    var spName = "sp_get_attendance";
    var parameters = new DynamicParameters();

    parameters.Add("studentId", _studentid, DbType.Int32,
ParameterDirection.Input);
    parameters.Add("startdate", _startdate, DbType.DateTime,
ParameterDirection.Input);
    parameters.Add("enddate", _enddate, DbType.DateTime,
ParameterDirection.Input);

    using var connection = factory.Create();

    var attendance = await
connection.QueryFirstOrDefaultAsync<AttendanceRate>(spName, parameters,
commandType: CommandType.StoredProcedure);

    //the ? and : act as a if statement
    return attendance is not null ? Results.Ok(attendance) :
Results.NotFound();

});

```

5.4 Setting up the domain and hosting

As part of the implementation process, establishing a reliable server environment and securing a domain were essential steps for deploying the database and the API as the mobile application would not be able to communicate with it otherwise. Fortunately, the developer already had access to a pre-installed server and hosting service running on Microsoft Server which had Microsoft SQL Server 2019 installed, streamlining the process significantly as there would be no additional expenses for it. Furthermore, the developer had previously acquired a domain name for their website, providing a convenient platform for setting up necessary subdomains.

5.4.1 Setting Up the Subdomain for the API

To host the API, I created a subdomain called “nfcapi.gorkemsarac.com/swagger/index.html” on the existing domain of <https://gorkemsarac.com>. This subdomain is dedicated solely to the API, ensuring clear separation between the website and API endpoints. The image displaying the subdomain can be seen down below:

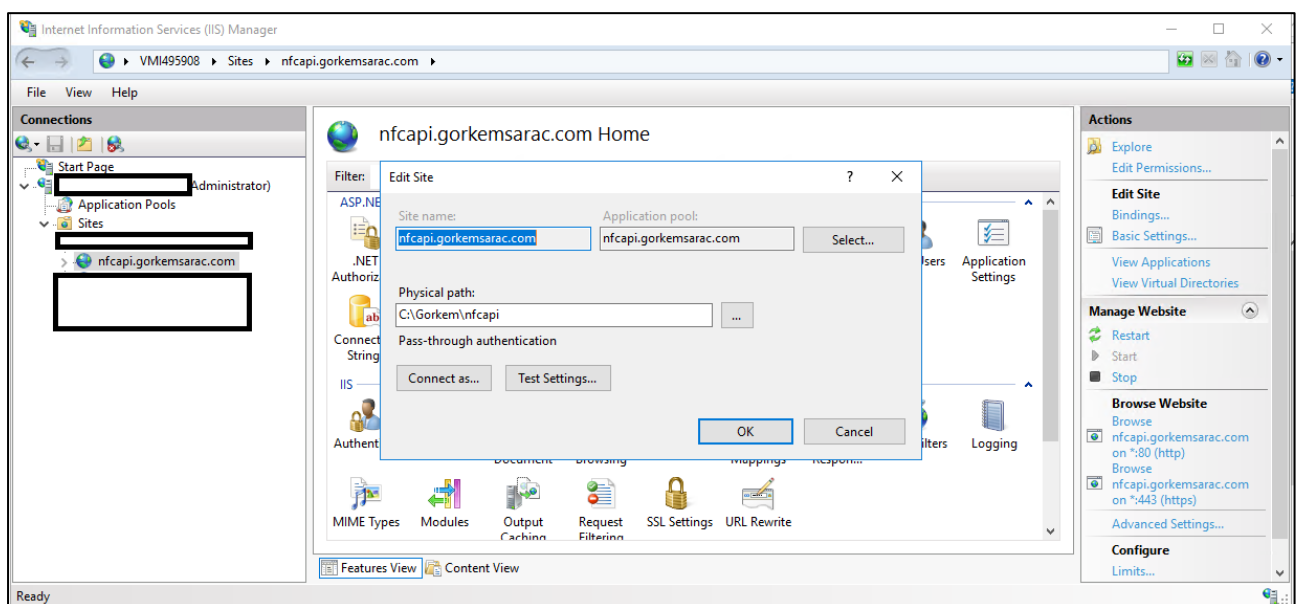


Figure 29: API website configuration

Furthermore, ensuring the security of the API was a top priority. Thus, the necessary certificates were generated using Let's encrypt to convert the subdomain address from HTTP to HTTPS. Let's encrypts interface with the API interface can be seen down below:

```
Administrator: Command Prompt - wacs.exe
C:\Win-Acme\win-acme.v2.1.13.978.x64.pluggable>wacs.exe
Found 2 files older than 120 days in cache path 'C:\ProgramData\win-acme\acme-v02.api.letsencrypt.org\Certificates'

A simple Windows ACMEv2 client (WACS)
Software version 2.1.13.978 (RELEASE, PLUGGABLE, 64-bit)
ACME server https://acme-v02.api.letsencrypt.org/
IIS version 10.0
Running with administrator credentials
Scheduled task looks healthy
Please report issues at https://github.com/win-acme/win-acme

N: Create certificate (default settings)
M: Create certificate (full options)
R: Run renewals (0 currently due)
A: Manage renewals (4 total)
O: More options...
Q: Quit
```

Figure 30: Let's encrypt CMD interface

```
Please choose from the menu: a

Welcome to the renewal manager. Actions selected in the menu below will be
applied to the following list of renewals. You may filter the list to target
your action at a more specific set of renewals, or sort it to make it easier
to find what you're looking for.

3: [IIS] nfcapi.gorkemsarac.com, (any host) - renewed 1 time, due after 2024/5/29 23:29:22

Currently selected 4 of 4 renewals

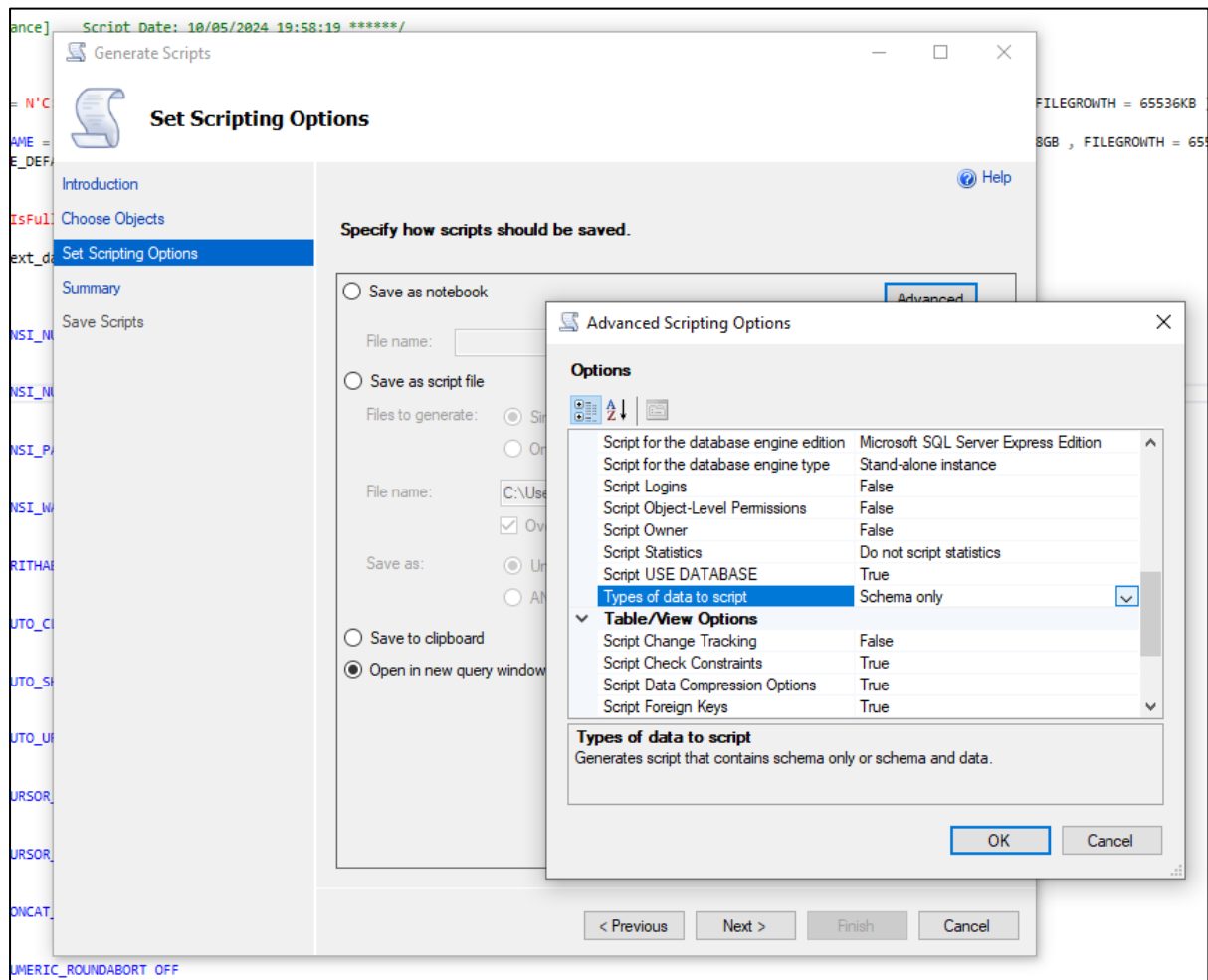
F: Apply filter
S: Sort renewals
D: Show details for *all* renewals
R: Run *all* renewals
U: Analyze duplicates for *all* renewals
C: Cancel *all* renewals
V: Revoke certificate(s) for *all* renewals
Q: Back

Choose an action or type numbers to select renewals: _
```

Figure 31: Displaying the API website on the Let's encrypt CMD interface

5.4.2 Exporting and Importing the Database

The next step involved exporting the local development database into a creation script. The creation script was executed on the Microsoft SQL Server instance on the hosting environment was, effectively migrating the local database to the remote server. The export screen can be seen down below:



5.4.3 Publishing the ASP.NET Core API

With the database set up, the ASP.NET Core API project was published in Visual Studio. These published files were then transferred to the hosting environment using a remote desktop connection, as it provided a simple way to access the server. The published version of the project can be seen down below:

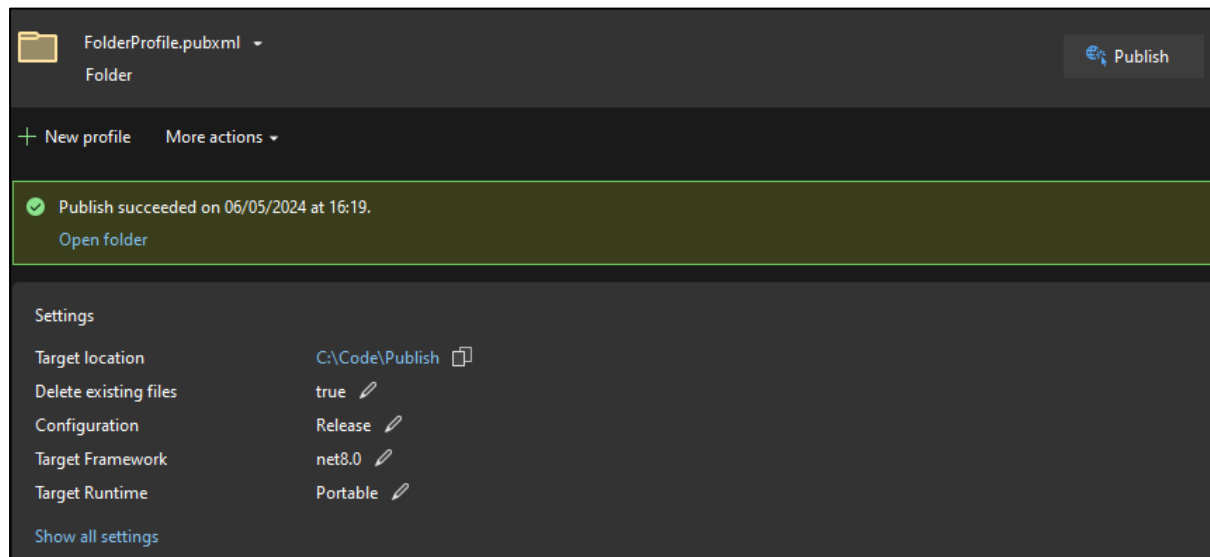


Figure 32: ASP.NET Core project publish screen

The only file to be changed was the “appsettings.json” file in which the connection string was changed to access the server database instead of the local database, which can be seen below:

```
{
  "ConnectionStrings": {
    "DefaultConnection": "Server=VMI495908\\SQLEXPRESS;Database=Attendance;Trusted_Connection=SSPI;Encrypt=false;TrustServerCertificate=true"
  },
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft.AspNetCore": "Warning"
    }
  }
}
```

Figure 33: Updated connection string

After successfully deploying the API, thorough testing was performed to ensure that all endpoints were functioning correctly. The testing process and results will be further detailed in the following Testing chapter.

5.5 Creating the mobile application

Unlike other parts of the implementation process where the project can be easily created by downloading the singular appropriate software and creating the project on it, react native required several components to run which was the first challenge the developer met. To set up a react native project, Node.js was needed first as well as a development environment, which in this case was Visual Studio because of its popularity and the developers experience with it. Additionally, there were two main ways of creating a React Native project, Expo Command Line Tool (CLI) or React Native CLI. Expo was chosen to create the project as it was a simpler process, which would prove to be the wrong decision later on. After installing Node.js, creating the blank expo project by using the command “`npx create-expo-app appname`”, and installing the Expo app on the mobile phone, the development environment was ready.

The project began with basic experimentation, such as creating buttons and implementing simple functionalities. The initial phase was challenging due to the developer's limited experience with functional programming, including concepts like arrow functions, states, asynchronous functions, callbacks, `useState`, and `useEffect` hooks.

After some experimentation, the navigation system was to be installed next using “`npm install @react-navigation/native`”. However, it became apparent that installing libraries required working on a React Native CLI project rather than an Expo project. Consequently, the developer switched to a React Native CLI project to facilitate the installation of necessary libraries using npm. However, doing so was not easy as it required Java Development Kit (JDK) to be installed using Chocolatey. After the JDK , android studio had to be installed on the development environment to Configure the `ANDROID_HOME` environment variable within the Windows OS. Lastly, the project was again created using the command “`npx react-native@latest init appname`”. Unlike Expo, however, react native CLI required either the setup of a virtual device or the physical phone to be plugged into the computer while the project was running.

After the switch, navigation in between screens was implemented using React Navigation, allowing the creation of multiple screens such as Student Home, Timetable, Admin Home, and Admin Write to NFC Screen, which replaced the experimentation code in `app.jsx`, which was the default initial launch screen for the application. During this phase, styles used to design the

view components were being re-used in every screen. To streamline the styling process, a `globalStyles.js` file was created to manage global styles, reducing the redundancy of copying styles across different components. An early version of the navigation code in `app.jsx` and the `globalstyles` can be seen down below:

```
import React from 'react';
import {NavigationContainer} from '@react-navigation/native';
import {createStackNavigator} from '@react-navigation/stack';

//Screens
import LoginScreen from './screens/LoginScreen';
import StudentHome from './screens/StudentScreens/StudentHome';
import AttendanceRate from './screens/StudentScreens/AttendanceRate';
import Timetable from './screens/StudentScreens/Timetable';
import AdminHome from './screens/AdminScreens/AdminHome';
import ManageNFCTags from './screens/AdminScreens/ManageNFCTags';
import ManageClassrooms from './screens/AdminScreens/ManageClassrooms';
import ManageStudents from './screens/AdminScreens/ManageStudents';
const Stack = createStackNavigator();
const App = () => {
  //START
  //log to show app has started fine
  console.log('Application Started');
  <NavigationContainer>
    <Stack.Navigator>
      <Stack.Screen name="Login" component={LoginScreen} />
      <Stack.Screen name="StudentHome" component={StudentHome} />
      <Stack.Screen name="AttendanceRate" component={AttendanceRate} />
      <Stack.Screen name="Timetable" component={Timetable} />
      <Stack.Screen
        name="SettingsScreenStudent"
        component={SettingsScreenStudent}
      />
      <Stack.Screen name="AdminHome" component={AdminHome} />
      <Stack.Screen name="ManageNFCTags" component={ManageNFCTags} />
      <Stack.Screen name="ManageClassrooms" component={ManageClassrooms} />
      <Stack.Screen name="ManageStudents" component={ManageStudents} />
      <Stack.Screen
        name="SettingsScreenAdmin"
        component={SettingsScreenAdmin}
      />
    </Stack.Navigator>
  </NavigationContainer>;
};

export default App;
```

```
import {StyleSheet} from 'react-native';

//Stylesheet
export const globalStyles = StyleSheet.create({
  //CONTAINERS//
  //Main container
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
    backgroundColor: 'lightblue',
  },

  //Aligns items
  aligned: {
    flexDirection: 'row',
    backgroundColor: 'lightblue',
    alignItems: 'center',
    justifyContent: 'center',
  },

  //TEXT//
  midTitle: {
    fontSize: 48,
    color: 'black',
    textTransform: 'uppercase',
  },

  midText: {
    flexDirection: 'row',
    backgroundColor: 'lightblue',
    alignItems: 'center',
```

```

        justifyContent: 'center',
    },

    //Text within the button
    buttonText: {
        fontSize: 18,
        color: '#fff',
        fontWeight: 'bold',
        alignSelf: 'center',
    },

    //BUTTONS//
    //Button
    button: {
        marginTop: 15,
        elevation: 8,
        backgroundColor: '#009688',
        borderRadius: 10,
    },

    //Border raidus has to be half of width and height
    circleButton: {
        marginTop: 30,
        marginBottom: 20,
        borderWidth: 1,
        borderColor: 'rgba(0,0,0,0.2)',
        alignItems: 'center',
        justifyContent: 'center',
        width: 200,
        height: 200,
        backgroundColor: '#fff',
        borderRadius: 100,
    },

    //INPUT//
    //Text input field in middle
    inputMid: {
        flexGrow: 1,
        width: 200, // Set a fixed width
        alignItems: 'center',
        margin: 10,
        padding: 10,
        height: 40,
        borderColor: 'black',
        borderWidth: 2,
    },
  },
});

```

One critical library required for the project was the NFC library, which was installed using the following npm command: “npm --save react-native-nfc-manager”. Before working with NFC on android phones however, NFC permissions were added to the AndroidManifest.xml file to request the use of NFC technology within the phone. Implementing NFC reading technology involved writing functions to detect and read data from NFC tags. Similarly, functionality for writing to NFC tags was developed later on for the admin screen, enabling data to be stored on NFC tags and read from them from a single application. An early version of the NFC read and write code can be seen below:

```
//NFC
const [tagData, setTagData] = useState(null);

useEffect(() => {
  initNfc();
  return () => {
    NfcManager.stop();
  };
}, []);

const initNfc = async () => {
  try {
    await NfcManager.start();
  } catch (ex) {
    console.warn('NFC init error', ex);
  }
};

//Reads the nfc tag
const readTag = async () => {
  try {
    await NfcManager.requestTechnology(NfcTech.Ndef);
    const tag = await NfcManager.getTag();

    if (tag.ndefMessage && tag.ndefMessage.length > 0) {
      // Extract payload from the NFC tag
      const payload = tag.ndefMessage[0].payload;

      // Convert payload from bytes to string
      const payloadString = String.fromCharCode.apply(null, payload);

      // Update state with the payload text
      setNfcData(payloadString);
    }
  }
}
```

```

    } catch (ex) {
      console.warn('Error reading NFC tag', ex);
    } finally {
      const StudentHome = ({navigation}) => {
        readTag();
      };

      //NFC DATA
      const [nfcData, setNfcData] = useState(null);

```

```

//Writes to the NFC tag
const writeTag = async () => {
  let result = false;

  try {
    //STEP 1, Request nfc usage
    await NfcManager.requestTechnology(NfcTech.Ndef);

    //Get tag data
    let tdata = setTagText(className, courseName, unitName);
    //Convert and encode the text message to bytes
    const bytes = Ndef.encodeMessage([Ndef.textRecord(tagText)]);
    const bytes = Ndef.encodeMessage([Ndef.textRecord(tdata)]);

    //IF the nfc is ready, write the bytes to the tag
    if (bytes) {
      await NfcManager.ndefHandler //STEP 2, Await nfc handler
        .writeNdefMessage(bytes); //STEP 3, Write message from bytes
      result = true;
    }
  } catch (ex) {
    //Catch any errors
    console.warn(ex);
    Alert.alert('Error', ex);
  } finally {
    //STEP 4 , Close off the nfc request
    NfcManager.cancelTechnologyRequest();

    //Notify the user of successfull write
    Alert.alert('Success', 'NFC data has been written to the tag');
  }

  return result;
};

```

The next major step was developing the login screen to interact with the API for user authentication instead of using hard-coded values. The ability to have API calls within the application was a crucial step as the entire system had now been connected, allowing the addition of all other required functionalities. The application was then modified to fetch a dropdown list of classrooms, populating a dropdown list for selection when writing data to NFC tags. The dropdown list was retrieved from the “react-native-dropdown-select-list” library. This setup was integral to ensuring that the NFC tags contained relevant information for the attendance system and made the NFC writing process simpler. Every other part of the mobile application that has API calls uses similar code to fetch data. The relevant code for the login functionality can be found below:

```
const authenticate = async () => {
  try {
    // Define the query parameters
    const params = {
      _Email: username,
      _Password: password,
    };

    // Convert the query parameters into a query string
    const queryString = Object.keys(params)
      .map(
        key =>
          `${encodeURIComponent(key)}=${encodeURIComponent(params[key])}`,
      )
      .join('&');

    // Make the GET request with the query string appended to the URL
    const response = await fetch(
      `https://nfcapi.gorkemsarac.com/student/Login/{email, password}?${queryString}`,
    );

    if (response.status === 200) {
      // Login successful
      const data = await response.json();
      console.log('Login successful:', data);
      navigation.navigate('StudentHome'); // Navigate to the home screen
    } else if (response.status === 404) {
      // Login failed
      console.log('Login failed: Invalid credentials');
      Alert.alert('Error', 'Invalid username or password');
    }
  }
}
```

```

} else {
    // Handle other status codes if necessary
    console.log('Login failed:', response.statusText);
    Alert.alert('Error', 'An error occurred while logging in');
}
} catch (error) {
    console.error('Error logging in:', error);
    Alert.alert('Error', 'An error occurred while logging in');
}
};

```

On the admin side, a “manage.js” screen was created to view any table in the database, as all the information within the database could be shown in a list. This screen Simplified the code as without it, 8 individual screens would be needed. This was achieved by returning the specific table item required for the view. To further polish the admin screen, a view within the manage screen was developed for viewing a single student's attendance, providing detailed insights into individual attendance records. A code snippet for how the return view functionality was achieved can be seen below:

```

useEffect(() => {
    const fetchDataFromName = async () => {
        try {
            let attendanceLink = pagename;

            if (pagename === 'attendance') {
                let isnum1 = /^d+$/ .test(sendstudentid);
                let _id;
                if (sendstudentid !== '' && isnum1) {
                    _id = sendstudentid;
                } else {
                    _id = '1';
                }

                //Define the query parameters
                const params = {
                    id: _id,
                    rownum: 1000,
                };

                //Convert the query parameters into a query string
                const queryString = Object.keys(params)
                    .map(

```



```

key =>
    `${encodeURIComponent(key)}=${encodeURIComponent(params[key]
)}}`,
    )
    .join('&');
attendanceLink =
    'attendance/AttendanceRecords/{studentid, rownumber}?' +
    queryString;
}
//Getting data from api
const response = await fetch(
    `https://nfcapi.gorkemsarac.com/${attendanceLink}`,
);
if (response.status === 200) {
    //Success
    const data = await response.json();
    // Group the data by type
    const listData = [];

    switch (pagename) {
        case 'course':
            listData.push({
                data: data.map(item => ({
                    courseid: item.courseid,
                    coursename: item.coursename,
                    active: item.active,
                    startdate: item.startdate,
                })),
            });
            break;

            //Other cases for tables

            default:
                break;
        }

        return listData;
    } else {
        console.log('Error fetching data (in response): ',
response.status);
    }
} catch (error) {
    console.error('Error fetching data (in fetch): ', error);
    return [];
}
};

```

However, the CRUD (Create, Read, Update, Delete) functionalities required an individual screens for each table as they had to have different fields and data types, which would need to be checked individually. Although they were different screens, their logic stayed the same in

general as they all had inserting, updating, and deleting capabilities. An image of the course table manage screen can be seen below:

← ManageCourses

MANAGE COURSE: 2

Course ID:

Course name:

Start date:

Active:

Figure 34: Manage course screen in the mobile application

Additionally, an attendance rate viewing page for the students was implemented, utilizing the “sp_get_attendance” stored procedure within the database to display attendance statistics over a specified period. This page also utilized the already existing list view structure in other screens to display the last addended sessions. The attendance screen can be seen down below:

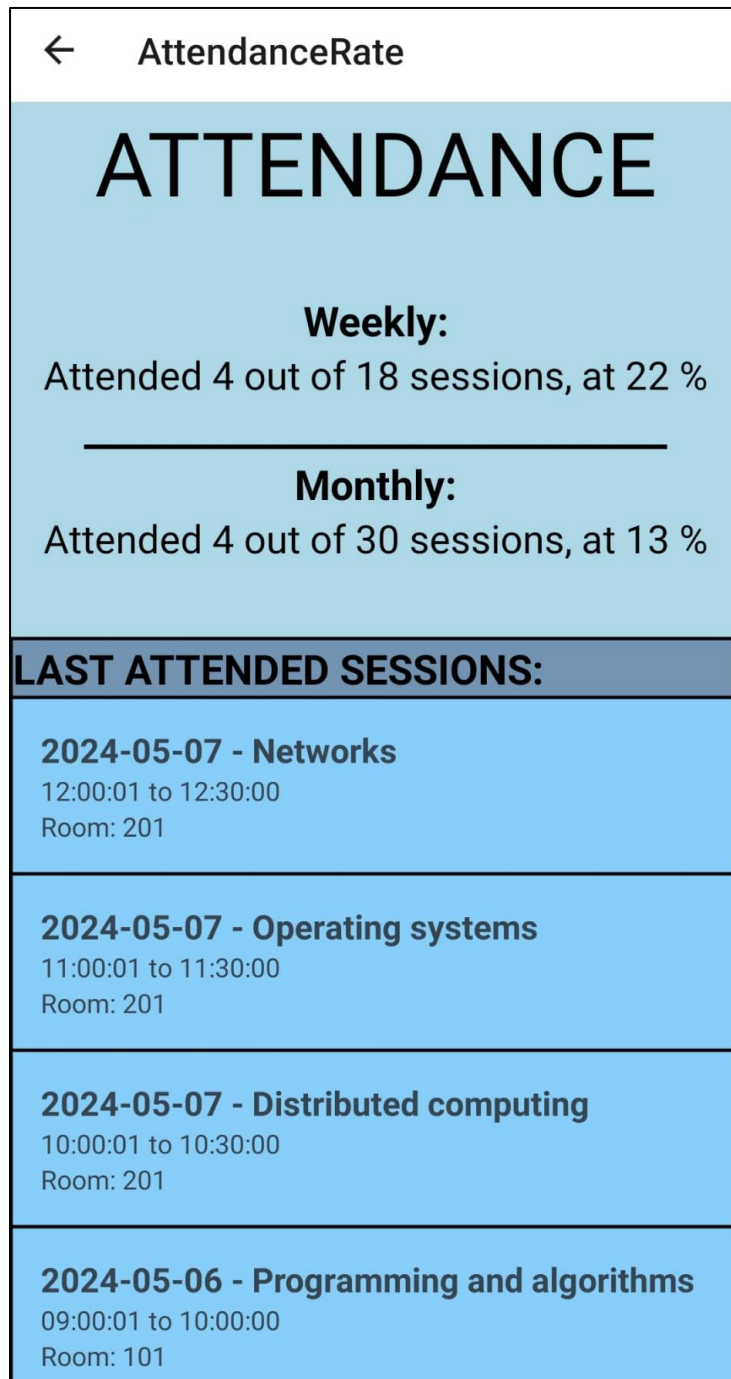


Figure 35: Student attendance page in the mobile application

Finally, to enhance the user experience, sound, vibration, and haptic feedback were added to error and success alerts as well as to button interactions through the creation of custom buttons. The “react-native-sound” library was used for sound , but vibration was already integrated into React Native. The code for the custom button can be seen below:

```
import React from 'react';
import {Alert, TouchableOpacity, Vibration} from 'react-native';
import Sound from 'react-native-sound';

//The sound effect
const clickSound = new Sound('buttonclick.mp3', Sound.MAIN_BUNDLE, error =>
{
  if (error) {
    console.log('Failed to load the sound', error);
    Alert.alert('failed to load sound');
  }
});

const CustomClickButton = ({onPress, ...props}) => {
  const handlePress = () => {
    //Play the click sound effect
    clickSound.play();

    // Trigger haptic feedback
    Vibration.vibrate(10);

    //Call the onPress function provided by the parent component
    if (onPress) {
      onPress();
    }
  };

  return <TouchableOpacity onPress={handlePress} {...props} />;
};

export default CustomClickButton;
```

5.5.1 Deploying the mobile application

Once the React Native application was fully developed, the next crucial step was preparing it for deployment. Unlike during development, the app cannot run independently on mobile devices without being connected to the development environment. Therefore, the mobile app had to be deployed into an APK which involved several steps. Although before these steps, a release branch was created in the Git version control system to avoid potential errors during deployment and maintain the stability of the project.

Following the React Native deployment guide, the first step was to generate a signing key. This was done using the Windows key tool utility using the following command in the terminal:

```
keytool -genkeypair -v -storetype PKCS12 -keystore my-upload-key.keystore  
-alias my-key-alias -keyalg RSA -keysize 2048 -validity 10000
```

This command generated a key named "my-upload-key.keystore". This keystore file was then securely copied into the project's Android directory to be used for signing the application.

Next, the Gradle variables were configured with the key information in the ~/.gradle/gradle.properties file. This included the keystore location, alias, and passwords.

After setting up the Gradle variables, the next step was to configure the app's build settings for signing. The signing configuration was added to the android/app/build.gradle file to ensure that the APK produced was correctly signed, ready for release.

With the signing configuration in place, the project was cleaned to ensure no remnants of previous builds could cause issues using the “./gradlew clean” command. Following the cleanup, the APK was built using the command “./gradlew assembleRelease”.

This process generated a release APK that was located in the android/app/build/outputs/apk/release directory. The APK was now ready to be distributed and installed on Android devices.

The next chapter, dedicated to testing, will detail the procedures and methodologies used to ensure the released app's functionality and stability in real-world scenarios using real mobile devices. Furthermore, screenshots of the final product will be included in the appendix C to provide a visual representation of the application's user interface and features.

6. Chapter 6 – Testing

Rigorous testing is an indispensable phase in the software development lifecycle on any project, ensuring the quality, reliability, and correct functioning of the system. This chapter outlines the comprehensive testing strategies and methods used to validate the mobile attendance registration system's various components, including the database, API, and mobile application. However, while testing encompassed all elements of the system, further emphasis was placed on the mobile application as it serves as the primary interface for end-users to interact with the attendance registration process.

6.1 Unit testing

Throughout the development of the mobile application and the corresponding API, unit tests were implemented to verify the correctness of individual units of code. During the creation of the system, each module, function, and feature was subjected to unit testing. These tests were designed to validate that each component behaved as expected under various conditions. For instance, in the React Native mobile application, unit tests were written for components such as buttons, text inputs, navigation flows, and API interaction functions. These tests checked for correct rendering, appropriate state changes, and successful data retrieval from the API.

The unit testing process also involved creating test cases that mimicked real-world usage scenarios. For example, tests were conducted to ensure that the login functionality accurately authenticated users, the NFC reading and writing processes correctly interacted with NFC tags to display the correct results, and the attendance recording system accurately logged student attendance data. The problematic results, such as text input fields accepting empty values or faulty navigation pages, were promptly fixed.

In addition to front-end testing, the back-end API, developed using ASP.NET Core, also underwent extensive unit testing. Each CRUD endpoint was tested to ensure it returned the expected responses for various input scenarios. This included testing the database interactions, such as querying attendance records and updating student information. The most common error within the API testing were variable type clashes. For example, the timetable content held the session start and end times in the time format, however there were no matching date types

within the ASP.NET core API, including the timespan format, thus the API was fixed by using type conversions in such cases.

Before the deployment of the system, a comprehensive suite of unit tests was run to confirm that all components functioned. This pre-deployment testing phase was crucial to identify and resolve any last issues that could compromise the system's stability and performance. By running these tests, the developer ensured that the system was reliable and functional, ready to be deployed.

6.2 Speed and efficiency testing

Speed and efficiency testing is paramount in ensuring that a mobile application provides a smooth and responsive user experience, especially to students as time lesson should not be wasted by waiting for the application. During the development of the mobile application, various components were tested to ensure that their loading times were within acceptable limits. This testing was essential to identify and rectify any performance bottlenecks that could degrade the user experience.

One of the primary areas of focus during speed and efficiency testing was the NFC registration functionality. The process of reading NFC tags involves several steps, including fetching the user's geolocation to record the exact position where the NFC interaction occurs. Initially, the application used a popular community geolocation library to obtain the device's location. However, it was observed that the geolocation fetching process often took an excessively long time, sometimes up to 40 seconds to a minute. To address this issue, an alternative geolocation library, “react-native-get-location”, was into the application instead. The new library reduced the geolocation fetching time to a few seconds, significantly enhancing the speed and efficiency of the NFC registration functionality. The location fetching time tests can be seen down below:

Device / location fetching time (in seconds)	Community location test library 1	Community location library test 2	Get location library test 1	Get location library test 2
Xiaomi mi 9	20 seconds	55 seconds	7 seconds	6 seconds
Xiaomi Redmi note 10 pro	13 seconds	32 seconds	3 seconds	3 seconds
Xiaomi poco x5 pro	6 seconds	13 seconds	2 seconds	8 seconds
Samsung s22 pro	12 seconds	4 seconds	1 second	2 seconds

Figure 36: Location fetching time test results

In addition to the NFC registration process, other parts of the mobile application were also subjected to speed and efficiency testing. This included testing the loading times of different screens, such as the login screen, student timetable, and admin interfaces as well as the load times of the manage screens within the admin interface. The application was tested on various devices, mainly including the ones mentioned in the table above, with different hardware capabilities to ensure consistent performance across the board. There are no recorded results to be presented as the aforementioned functionalities were completed within seconds across all the devices, not necessitating a replacement.

6.3 Black box testing

Black box testing, a method where the internal structure of the system being tested is unknown to the tester, was applied to further evaluate the system. This testing approach focuses on examining the functionality of the application without looking into its internal workings or the code itself. The aim was to verify whether the system met the basic requirements and to identify any functional errors in user interactions and the data handling processes.

6.3.1 Testing Procedure and Encountered Bugs:

The black box testing process involved a tester from a computer science background, but without the knowledge of the inner workings of the system. Test cases were developed to cover all functionalities, including input validation, component interactions, and data integrity checks. During this phase, several significant types of bugs were identified:

Input Field Issues: The tester found that certain input fields were unresponsive or did not validate the input correctly, leading to errors in data entry. For example, student timetable page did not accept the input of secondary key ID's which prevented them from inserting new rows. The input validation logic was adjusted to ensure that all data entered into the system adhered to expected formats and constraints. Regex was used to achieve client-side validation, and error alert notifications were added to inform the user about their mistakes.

Cascading Deletion Issues: A critical issue was discovered in the management of units and courses. Deleting a course or unit inadvertently caused cascading issues in related tables. This issue also unexpectedly affected the login system, as it caused issues when the system attempted to relevant course or unit information, failed, causing the login system to alert the user with an error. To resolve the cascading deletion problem, the database was modified to include appropriate foreign key constraints with "ON DELETE SET NULL" and "ON DELETE RESTRICT" actions. This adjustment prevented unintended deletions and maintained data integrity. Additionally, the application logic was updated to handle these constraints properly, ensuring that the user is told about the consequences of the delete action.

Time and Date Field Errors: Several problems were encountered with time and date fields, particularly concerning format consistency across different views and any time format going below seconds. These inconsistencies led to confusion and errors in the inserting and updating functionalities. These issues with time and date fields were addressed by constraining the date and time fields throughout the application to fit their formats. The system was configured to parse user input into the correct format in case of incorrect submissions to avoid errors within the database system.

6.4 User testing

User testing is an essential phase in the development lifecycle, providing critical insights into the usability and functionality of the application from the end-users' perspective. For the mobile attendance registration project, user testing was conducted to ensure that the application met the needs and expectations of its intended users.

The primary objectives of user testing were to assess the ease of use, identify any usability issues or bugs that were not detected during previous testing, and importantly to evaluate user satisfaction with the application to gather feedback for further improvements and refinements.

6.4.1 Methodology

User testing involved a total of 13 unique testers, comprising both individuals who had previously tested the system and those who had never interacted with it before. This mix provided a comprehensive evaluation from both experienced and new users, ensuring a well-rounded assessment of the application. The participants were shown the participant informant sheet and signed a consent form which can be seen in appendix B.

Experienced testers had interacted with earlier versions of the application and provided insights into improvements made and any outstanding issues. While the new testers had no prior experience with the application, offering fresh perspectives on usability and overall functionality.

The participants were then given access to the mobile application and asked to perform specific tasks designed to evaluate various aspects of the system. These tasks included:

- Using both a student and an admin account to log in.
- Navigating through different screens such as the student home, timetables, student attendance, and admin home.
- Utilizing the NFC functionality and the NFC tags to register attendance.
- Choosing and writing classroom data to NFC tags.
- Viewing the admin management screens for the tables and performing CRUD operations on the records.
- Viewing attendance records for a specific student.

After completing the tasks, participants were asked to fill out a questionnaire designed to gather quantitative and qualitative data on their experience. The questionnaire covered various aspects of the application, including overall usability, the student and admin user interfaces, and their likelihood to use the project instead or in addition to the existing attendance system. The full questionnaire can also be seen in appendix B.

The feedback from user testing was instrumental in identifying areas for improvement and validating the application's overall effectiveness. The results and feedback gathered will be discussed in more detail in the following evaluation section. This user feedback was crucial for guiding future enhancements and ensuring the application continues to meet user requirements effectively.

7. Chapter 7 – Evaluation

This following chapter will provide a comprehensive assessment of the NFC-based mobile attendance registration system, examining how well the project met its initial objectives and requirements. This chapter then explores the advantages over existing and other similar attendance systems. It will also incorporate user feedback gathered through extensive testing to provide insights into the user experience and satisfaction levels. Lastly, this chapter identifies the limitations encountered during the development and deployment phases, offering a different perspective on the system's performance and areas for potential improvement.

7.1 Achievement of Objectives and requirements

Achievement of objectives:

The first objective, determining the appropriate software and framework for the mobile application, was thoroughly addressed within the design chapter by selecting React Native. This framework was chosen for its extensive documentation, support, libraries, and compatibility with both iOS and Android platforms, which was crucial given the range of devices used by university students.

The design and implementation of a user-friendly interface were prioritized to ensure seamless user experience. The application's UI was developed with a focus on intuitive navigation and accessibility, which was reflected in the positive feedback from user testing, which will be further discussed in the user feedback section.

Developing a robust and efficient database system was another key objective. Microsoft SQL Server 2019 was chosen for its stability, scalability, and integration with the existing Microsoft-based environment. This choice facilitated the stable management of student information and attendance records as the developer already had experience with the software.

The creation of a reliable and secure API was accomplished using ASP.NET Core. This API facilitated seamless communication between the database and the mobile application, ensuring data integrity and security. The API endpoints were designed to handle requests efficiently,

with a strong focus on compatibility and futureproofing for potential web or desktop applications, which could use the same API without any changes.

The integration of a secure login system was successfully implemented, allowing students to authenticate using their credentials (not real credentials, as it would not be ethical to ask testers for their real identifications). This feature ensured that only authorized users could access the application, maintaining the security of student data.

Incorporating NFC technology for attendance registration was a critical component of the project. The application leveraged IoT capabilities to enable effortless attendance registration by tapping NFC-enabled mobile devices against classroom tags. Admins could also write the appropriate classroom information onto the tags. The NFC technology selected was compatible with the application, providing a reliable and user-friendly attendance registration method.

Comprehensive usability testing and feedback collection from students were conducted. This feedback was instrumental in refining the application's functionality and user experience, leading to iterative improvements.

Performance evaluation and optimization were ongoing processes throughout the development phase. Identified issues, such as the location fetching time, were promptly addressed, ensuring a seamless and efficient attendance tracking experience.

Achievement of requirements:

The system's functional requirements were met during the early stages of development. User authentication was securely implemented through the login page, and the NFC attendance registration worked effectively with the NFC tags. Real-time attendance updates were achieved in the student attendance page, providing users with immediate updates on their attendance rates and last attended sessions. The student screen also included a timetables screen for them to view their upcoming sessions. However, notifications outside the app were not implemented as it would require a set of functionalities checking the students' timetables in the background. This feature was not added because of time constraints and a focus on testing the main functionalities of the system.

For the admin requirements, the system included screens for, viewing, and managing database tables, as well as creating or editing NFC tags. The application also has warnings and

restrictions in place to prevent conflicting information. These functionalities were implemented to ensure comprehensive administrative control over the attendance data. Although the app also contains a page for the admin to view a student's attendance data, a page with more analytical tools to generate reports and provide attendance patterns have not been implemented yet. This again is because of time constraints as generating an attendance pattern report is dissimilar to other functionalities of the app.

The mobile application requirements were also mostly fulfilled. The app supported NFC functionality, had an intuitive user interface that was functional different devices and screen sizes, and included user experience enhancements such as sound effects, haptic feedback, and vibration for buttons and alerts. However, the mobile app does not have the ability to record attendance in offline mode, as it would be too much of a security risk. This also should not be an issue due to the fact that the university has free WIFI across the campus.

The API requirements were addressed by ensuring secure data handling and compatibility with both mobile operating systems, as well as being efficient enough not to warrant additional waiting times for the system. The API was designed to be scalable and future-proof, capable of integration with web and desktop applications in later upgrades.

The database requirements focused on flawless functionality and scalability. The relationship between tables and stored procedures worked seamlessly, with SQL server backups in place to protect against data loss. Furthermore, the Microsoft SQL server 2019 supports up to 524,272 terabytes of data (Ray *et al.*, 2023), meaning the database is scalable to accommodate any number of students.

Non-functional requirements such as performance, reliability, usability, and security were also adequately met. The system responded to user interactions within acceptable time frames after the location library changes, maintained consistent uptime, ensured data integrity through entry restrictions, and implemented strong authentication mechanisms. Additionally, the system was user friendly as 11 of the 13 users who tested the project had never seen it before yet praised it to be easy to use, which will be further discussed in the user feedback section.

7.2 System evaluation

The Mobile Attendance project was an ambitious undertaking aimed at enhancing the efficiency and accuracy of attendance tracking within academic settings through the use of NFC technology. This system evaluation critically examines the functionality of the completed product and the results achieved, focusing on the system's architecture.

The system comprises three main components: the database, the API, and the mobile application. Together, these form a complex, yet complete infrastructure designed to handle the demands of a modern educational environment. The database serves as the central repository for all data, the API facilitates secure and efficient communication between the database and the mobile application, and the app provides a user-friendly interface for both students and administrators to use.

7.2.1 Comparative Advantages

The NFC system introduces several enhancements over traditional paper and card-based systems. It eliminates the need for physical cards, reducing material costs and the inconvenience of card loss or damage for the students. Additionally, the NFC setup reduces the potential for impersonation attempts within attendance entries, a common issue with card systems as well as the traditional sing-in sheets. Not only does it reduce the likelihood of these attempts, but it also allows the students who do so to be easily picked out from the rest.

Implementing the NFC system is also cost-effective compared to other biometric or card-based systems explored within the literature review. As the NFC tags are inexpensive, especially when bought in bulk as the institution would, and the mobile app can be deployed on existing smartphones of the students, reducing the need for additional hardware investments and helps reduce e-waste as less complicated devices are needed.

The new system also incorporates features not available in the current mobile application of the university, such as real-time data retrieval within the application for timetables and attendance rates, enhanced security measures, and a more intuitive user interface. These improvements significantly enhance the user experience for students and administrators alike.

In the broader context of higher education and academia, thanks to the system's design, it is very suitable for it to be adapted to other universities or academic institutions. Its compartmentalized architecture enables customization to meet specific needs without extensive redevelopment. The scalability of the database and the flexibility of the API mean that the system can accommodate increases in user numbers and data volume with minimal adjustments. Furthermore, the adjustments needed for the system can be centralized as the components that need to be altered can be done within the hosting server.

Overall, the Mobile Attendance project portrays an advancement in attendance management technology. It successfully meets its initial objectives, providing a robust, user-friendly, and cost-effective solution that enhances the accuracy and efficiency of attendance tracking. Further development and refinement based on feedback would ensure its readiness for wider deployment, not only in academic scenarios, but also organizations and businesses as well.

7.3 User feedback

In this study, a total of 13 participants were involved in providing feedback on the NFC-based mobile attendance registration system. The user feedback process engaged 13 unique participants who interacted with the application and subsequently filled out a detailed questionnaire. The participants were a mix of individuals who had previously tested the system and new users who were experiencing it for the first time. This combination provided a comprehensive evaluation from both seasoned users and fresh perspectives.

7.3.1 Findings:

The feedback gathered through the questionnaires was predominantly positive, with participants highlighting several key strengths of the application:

Most participants felt that the mobile application was easy to use as the average useability rating was 9.3 out of 10. The specific responses can be seen in the following response chart below:

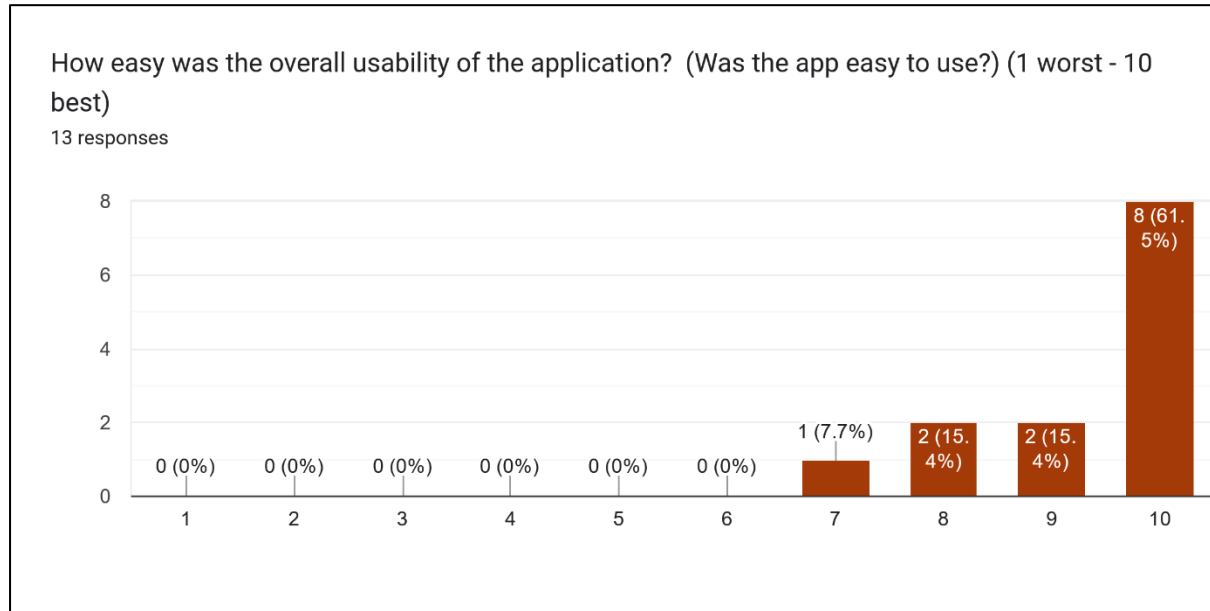


Figure 37: Useability rating by users

Similarly to the useability of the application, the rating of the student interface was at an average of 8.9 out of 10 with one user rating it 4. The response chart can be seen down below:

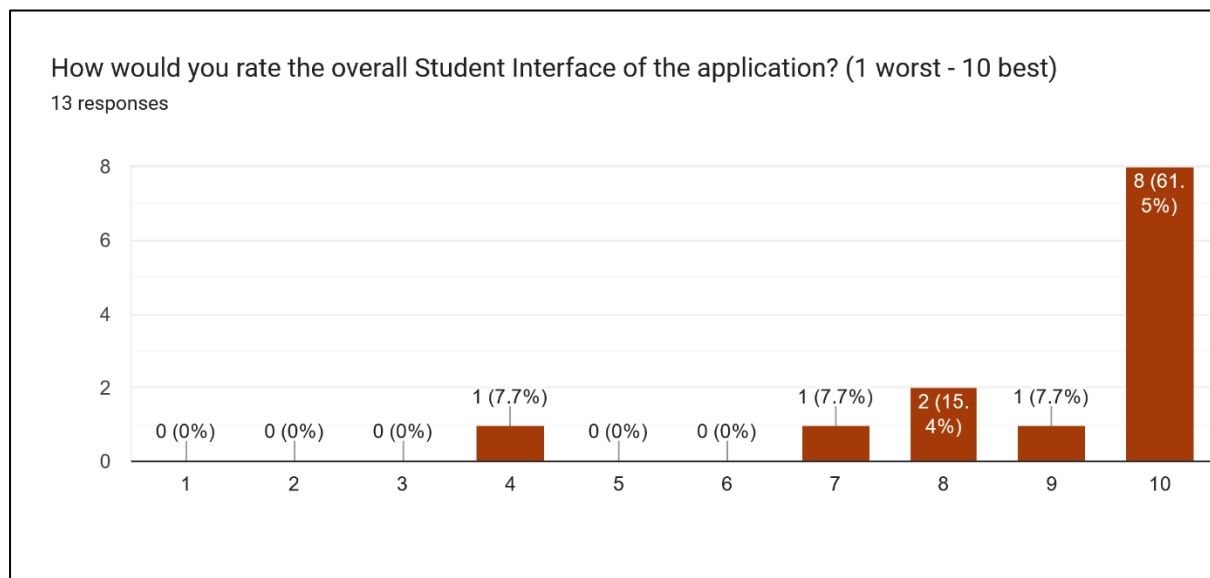


Figure 38: Student interface rating by users

Furthermore, this question also had an optional text box underneath asking their reasonings behind their rating for the student interface. The general consensus along the responses was that the interface was simple and easy to use, with some suggestions on how to improve it. The specific responses of the students regarding their reasonings can be seen down below:

- There were not a lot of options for the students.
- it was easy to use and understand and very impressive.
- It had a clean design, it rejected false logins, detected the NFC of the chip and it gave visual feedback when actions were completed.
- Easy to use and understand but some small formatting problems when using on mobile or when opening a keyboard such as text being hidden.
- simple and straight to the point .
- its good.
- simple and easy .
- All features in the app actually stay in the app e.g. timetable.
- Buttons were very clear, and navigation of the UI was easy to understand .
- I'm easily impressed.
- Interface was clean and looked good.

- Clean however, it could be more visually appealing.
- Simple and effective.

Similarly, the users were asked to rate the overall admin interface, which had a slightly better reception at an average of 9.3 out of 10. The response chart can be seen down below:

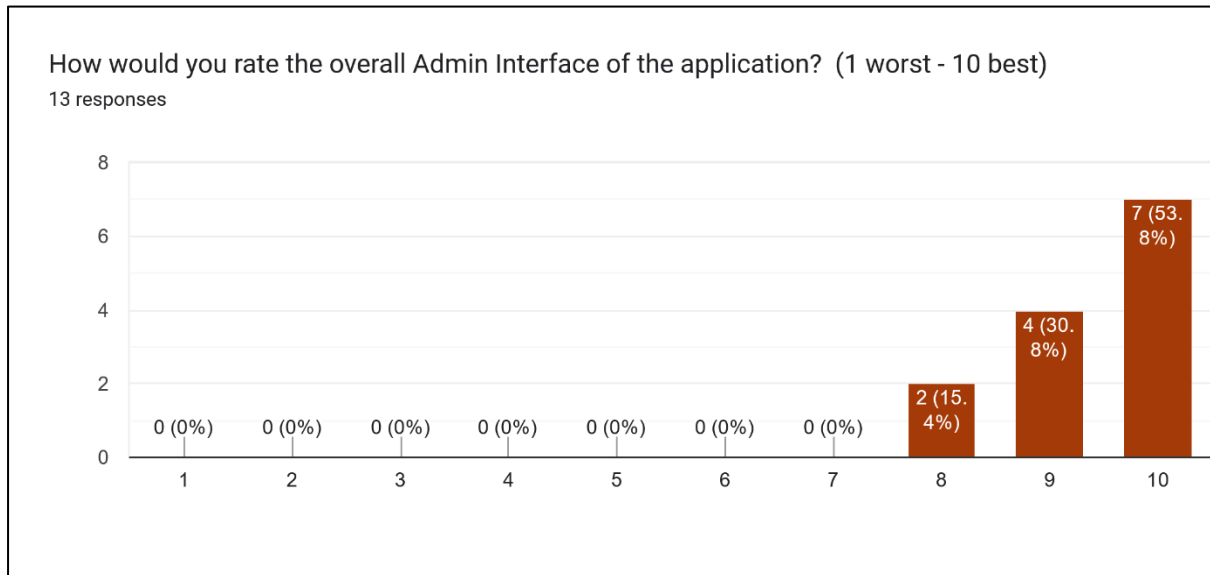


Figure 39: Admin interface rating by users

The users were again asked to justify their ratings on the admin interface like last time. The general consensus was again that the admin interface was easy to navigate and had the necessary functionalities. There were some negatives they mentioned such as loading times for the student timetable, minor overlaps, and clashes. The specific responses can be seen below:

- It has pretty much everything that an admin interface should have.
- edit student timetables took time to load.
- Invalid information was rejected, for example a unit was rejected when it did not match a pre-existing unit code value, The menus were straightforward and intuitive.
- also easy to use like the student interface but also experiences some similar minor overlapping and clashing.
- again, it is simple and straight to the point and provide all the options needed.
- its good.
- can easily navigate through the application.
- admins have full control.

- The admin interface was also very easy to navigate, titles and buttons were very clearly marked and responsive.
- It was intuitive and easy to use.
- The options were clear, and the data was easy to see.
- Simple and consistent.

Next, the users were asked how likely they would use this application instead and alongside of the card reader system if they could both be used in lectures/labs. The responses to these two questions were important as this had technically been the goal of the entire project, to replace or to provide an alternative to the existing attendance systems. Thankfully, the rating of the users using the application instead of the existing system was an average of 8.9 out of 10 with one user however, rating it 4. Similarly, the users were slightly more likely to use the application alongside their ID cards with an average rating of 9.0 out of 10. These responses, although not too strongly, are still evidence of the fact that this project has achieved its goals as the users are accepting of the application, wanting to use it instead or alongside with the current ID system. These two response charts can be seen down below:

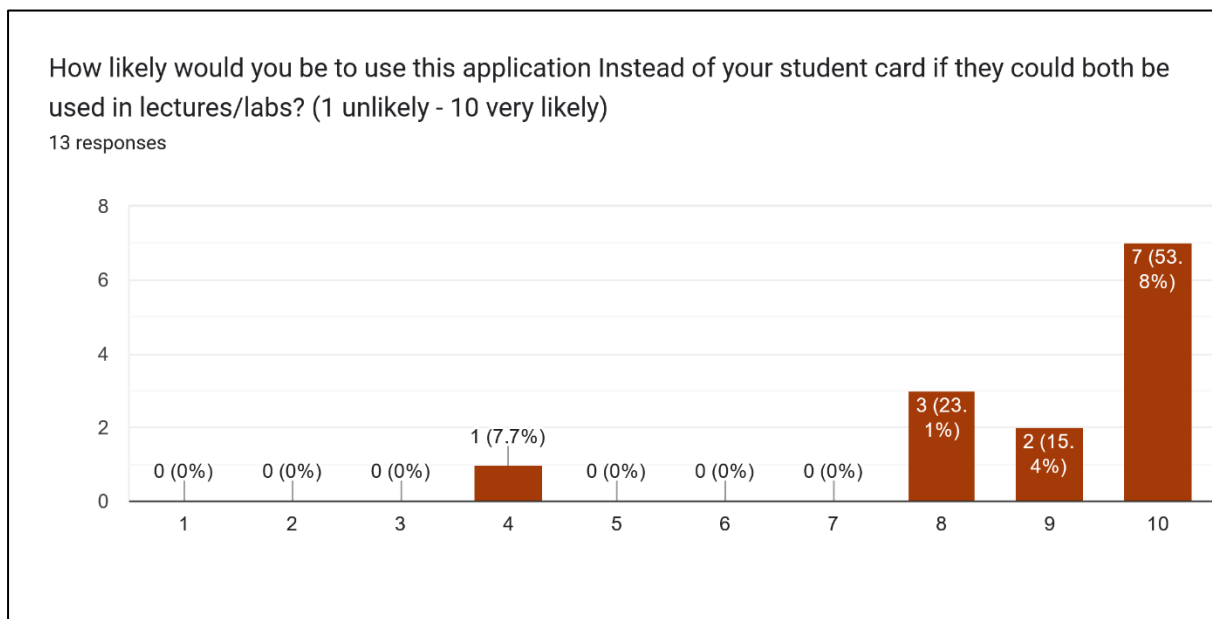


Figure 40: System use instead of the current system rating

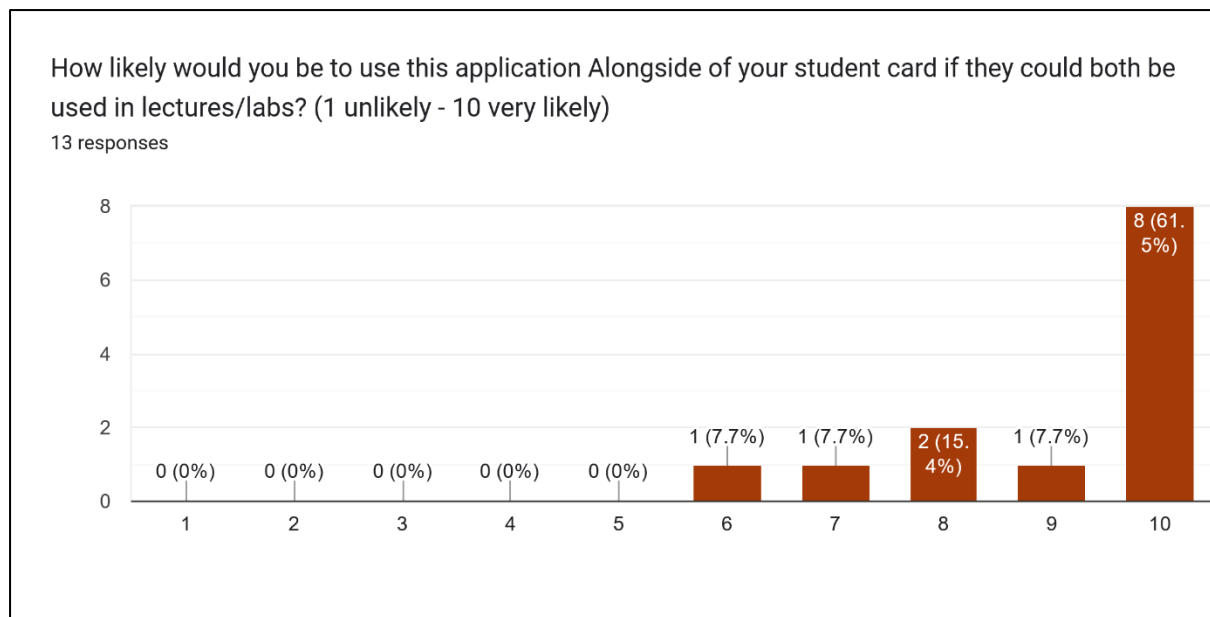


Figure 41: System use alongside the current system rating

Furthermore, users were asked if they encountered any errors, and 4 out of the 13 users said they did. Below are the users who encountered a bug, explaining it:

- The program crashed a few times, the first occurrence was when I tried to scan the NFC chip, second occurrence was when I tried to write an NFC device to the database
- Nothing in terms of functionality but some small display errors as already stated, like when opening a keyboard etc.
- when scanning the app closed itself
- On the admin interface the "Course ID" text box did not let you input anything.

Errors such as the course id text box and the display errors were fixed promptly. However, as two users have mentioned before, there persists a “bug” in the application where upon reading the NFC tag twice without enabling it will quit the application. However, this “bug” was later found out not be an actual bug, but a functionality of the android operating system. Confirmed by our testing, in cases where an NFC tag is scanned twice within an android application when that application does not have the NFC reader on, the android OS will take over and manage to launch whatever is in that NFC tag, effectively quitting the application since the NFC tag does not have a link or a web address but information about the classroom. This was confirmed to be in other applications such as Google pay and Bee network. There might be a way to overwrite the NFC reading functionality of an android system, but the developer was not able

to find a solution to block this feature within the android system, as neither could a corporation like google.

Lastly, the users were asked if they had any additional comments or feedback regarding the system, and the non-empty responses, which are mostly regarding the UI, can be seen down below:

- Change the colour scheme because it looks ugly.
- The black circle was not immediately intuitive as the button for logging in, maybe some bevel or a traditional rounded rectangle button shape would make this more obvious.
- Having an IOS version of the app

UI feedback was partially implemented as the colour scheme was later improved, as well as changes to make the NFC registration button clearer to understand. The application was tested on android devices by the users, thus the feedback regarding having an IOS version, which will be further discussed in the following limitations section.

7.3.2 Accuracy of gathering user feedback

Although it was not on the questionnaire, the participants were great at traversing through the application screens and functionalities, both in student and admin interfaces. However, one question that stands out is if this was caused by the overall design of the application, or if it was because these students were fairly adept at using mobile applications, since they were all computer science or cyber security students. To add onto the question, could this familiarity be caused by the fact that all testers were between the ages of 19 to 22, making them generally adept at using technological applications. To properly answer these questions, a wider range of testers from a diverse range of backgrounds would be needed to confirm the results. Nonetheless, as there are more students in higher education below the age of 24 than over, the data would still be relevant. Furthermore, although their feedback was not recorded on questionnaires, at least 5 separate lecturers who were shown the application understood the concept and use of the application and were generally pleased with the overall interface. What they had issues with, or rather suggestions for, were more on the side of security, authorization, impersonation, and ethics.

7.4 Limitations

While the NFC-based mobile attendance registration system presents a numerous improvement over traditional methods, it is not without its limitations. These limitations span several aspects, including security vulnerabilities, potential misuse, deployment constraints, and database shortcomings.

7.4.1 Buddy Punching with Cheap Phones:

One notable limitation is the potential for "buddy punching," where students can bypass the attendance system by using a cheap secondary phone instead of their ID cards. A student could give this phone to a friend, who could then register their attendance on their behalf. This undermines the system's integrity and accuracy. To mitigate this, the system could incorporate multi-factor authentication, such as fingerprint verification to ensure that the person tapping the NFC tag is indeed the rightful student. However, this does lead to the same ethical issues mentioned in the literature review chapter, leaving the integration of such biometric verification methods open for debate to question the institutions if they value accuracy over privacy.

7.4.2 Security Weaknesses:

The system's current security measures are relatively basic, primarily relying on secure logins. This could be strengthened by implementing token-based authentication, which adds an additional layer of security. Tokens are more secure than traditional sessions as they are harder to hijack and can be set to expire, reducing the risk of unauthorized access. Moreover, the system would need to undergo regular security audits to identify and fix vulnerabilities if it were to be deployed in a real institution.

7.4.3 Reverse Engineering and Location Spoofing:

NFC tags could potentially be reverse-engineered, and their data could be replicated by students with the necessary knowledge. Additionally, the system's reliance on GPS coordinates for location verification is vulnerable to spoofing, where fake latitude and longitude data can be generated. The combination of these methods would allow students to register attendance

remotely, bypassing the physical location requirement. To counteract this, the system could use advanced location verification methods, such as triangulating Wi-Fi signals. However a better solution would be to encrypt the data on the NFC tags themselves, which would be decrypted on the server to further prevent reverse engineering. This feature was not implemented in the current system because of the time constraints, as data encryption and decryption are a very distinct field compared to creating mobile applications, which the developer focused on learning instead.

7.4.4 Deployment Constraints on iOS:

Currently, the application is only deployed on Android devices via APK files. Deploying the application on iOS presents additional challenges, primarily due to the need for an Apple developer account to enable NFC and location permissions. This requirement adds to the cost of the deployment process, which this project lacked the funds for as an Apple developer account costs 99 United States Dollars (USD) per year (Apple Inc., 2024). In a real-life deployment scenario, an apple developer account can be created, and the necessary permissions can be easily obtained for the creation of the iOS application.

7.4.5 Database Shortcomings:

The existing database schema has certain limitations as the database does not encompass every field of information a higher education institution might store. For instance, courses do not have associated departments, and the capacity of classrooms for specific units is not specified. Enhancing the database structure by adding these attributes, as well as other relevant information such as faculty details, unit leaders, and room capacities, can provide a more comprehensive data set, which thankfully can be added onto the existing system in future updates without breaking other functionalities within the system.

8. Chapter 8 – Conclusion

As the developer reflects upon the journey of conceptualizing, designing, and implementing the NFC-based attendance registration system, this final chapter serves as a culmination of the experiences, learnings, and aspirations for the future.

8.1 Conclusion

This dissertation has been a culmination of extensive research, careful planning, careful implementation, and thorough testing. The aim of this project was to develop a reliable, user-friendly, and efficient NFC-based mobile attendance registration system that would leverage modern technologies to streamline the attendance process, first in academic settings but also to create a system that could be configured to be utilized in other institutions as well.

From the start, this project has aimed to address the various challenges associated with traditional attendance systems and specifically, the currently employed student ID system. The primary goal was to create a seamless experience for both students and administrators while ensuring data integrity, security, and real-time updates. The development process was guided by a set of functional and non-functional requirements, which were adhered to as far as possible.

One of the most crucial decisions was selecting the appropriate technologies for the system. The choice of ASP.NET Core for the API, and Microsoft SQL Server for the database was mainly driven because of familiarity with these tools. React Native however, was chosen for its cross-platform capabilities and primarily extensive library support. Initially, this project was planned to be developed alongside the Mobile Computing unit, using lessons from that class to develop the app. However, the curriculum unexpectedly changed to .NET MAUI. While it was acceptable to develop the project in .NET MAUI, it would be hard to do so due to how new it was, without any proper tutorials. This posed a big problem as the developer was new to mobile app development and integrating NFC technology into applications. React Native's vast tutorials and guides thankfully provided the necessary knowledge to successfully develop the mobile application.

The system was challenging to implement, especially with new concepts such as functional programming, asynchronous functions, and state management in React Native. Despite the initial setbacks, the development progressed smoothly thanks to the extensive tutorials and code available for React Native. The API development using ASP.NET Core also benefited from tutorials, which were very easy to follow because of the developer's prior experience with C# and ASP.NET.

Throughout the development, Git and GitHub were indispensable for version control. They provided a safety net, allowing the development of the application without a fear of losing progress. Version control was used from the start since the developer was wary of losing progress made within a new framework.

The testing phase was meant to be very thorough, thus had four stages, unit testing, speed and efficiency testing, black-box testing, and user testing. Each functionality was rigorously tested during and after the implementation to ensure the system performed as expected. The user testing phase, which involved 13 unique participants, was the most useful phase of testing as it helped identify more nuanced areas for improvement not so obvious perhaps to a developer.

Evaluating the system against the initial objectives and requirements, it is evident that the project has generally been a success. Furthermore, the mobile application was easy to use for all testers, who voiced their satisfaction with the product, mentioning how they would rather use this mobile application instead of the current system. It was also important to acknowledge the limitations as the system is not perfect. However, suggestions on how to overcome these limitations were clearly laid out which proves the system would be perfected if given more time and resources, thus could be deployed to a real university. In conclusion, this system with further refinements can significantly enhance the attendance management process in educational settings.

8.2 Next steps

While the finished system demonstrates significant improvements over traditional methods, there are several areas to be fixed and enhanced. If provided with more time, increased funding, or a development team, the following steps could be taken to address the current limitations.

Security Enhancements: As highlighted in the limitations section, the current security measures are relatively basic. Implementing token-based authentication to the API calls of the system would protect the system against unauthorized access to admin functionalities and vulnerabilities regarding the data being intercepted. Furthermore, incorporating data encryption on the NFC tags themselves would greatly lower the risk of reverse engineering and impersonation attempts, thus providing a fully secure system against “buddy punching”.

Comprehensive Database Restructuring: The existing database schema could be expanded to encompass a broader range of information relevant to academic institutions. This includes incorporating details such as departments, classroom capacities, faculty information, unit leaders, student details, and any other pertinent data. A more comprehensive data model could enable integrations with other educational institutions.

Cross-Platform Deployment: Currently, the mobile application is only available for Android devices. Allocating funds towards obtaining an Apple developer account and addressing the necessary permissions would enable the deployment of the application on iOS devices. This cross-platform availability would ensure accessibility for a wider user base, further supporting the idea of cross institutional integration.

If provided with the opportunity to revisit this project from the beginning, several lessons learned through the development of this project could be applied. For instance, having more frequent user testing and feedback even from the design stages, could have led to earlier identification of usability concerns. Additionally, investing more time in the system architecture and having more feedback from people like lecturers who can spot specific flaws to have a secure system would mitigate some of the limitations encountered during later stages. Overall, having more constant feedback from everyone and focusing more on finer details would be the key to achieving a perfect product.

8.3 Reflection

Before I selected my final project, I knew I wanted to do something that would be impactful as my final piece of work within my academic career. I knew I wanted to create a product, a product that I could put on my CV to display as my masterpiece. Although I wanted to create a major project that would use recent technologies such as machine learning or augmented reality, I also wanted to achieve a high grade and produce fully complete product. Given these factors, it was crucial for me to strike a balance between the project's complexity, in terms of both learning and developing it, and ensuring the scale of the product was substantial enough to qualify as my final academic endeavour. I was also determined to develop a product that was not just another clone of the template projects we had been shown. I wanted for my project to serve a real purpose and hold the potential for real-world use. Thus, I decided to address one of the most common issues I saw during my time as a student lab demonstrator: the inefficiencies and faults of the current attendance system. This motivated me to develop the NFC-based mobile attendance registration system. The project was an opportunity to learn new technologies, including mobile app development and working with IoT technologies such as NFC tags while also allowing me to experience the role of a full-stack developer, handling all aspects of the system's development independently.

When it came to developing the project, I had the upmost confidence in my choice of tech stack. I choose Microsoft SQL Server for the database due to my familiarity with it. For the backend, I chose ASP.NET Core, not only because of my experience with it but also because it is a framework that is gaining traction in the industry. Finally, I selected React Native for the mobile application development as I would be concurrently learning it in the mobile computing unit. Additionally, React Native is still highly regarded in the job market, making it a valuable skill to acquire. However, the unit unexpectedly shifted to focus on .NET MAUI, presenting a significant setback. Although I was more than open to learning .NET MAUI, the lack of documentation and tutorials made it a daunting task. This challenge reaffirmed my decision to use React Native, which offered extensive libraries with an ample number of tutorials to follow. Looking back, I do not regret my choice as I now know not one, but two mobile frameworks, putting me at a very advantageous position to when and if I apply to a mobile development job.

One of the most valuable aspects of this project was the hands-on experience with various technologies. I had to get comfortable with functional programming, JavaScript, the React framework and ASP.NET CORE. Each step, from uploading the database to the hosting server to implementing the API endpoints, was a learning curve. It was not just about writing code, it was about understanding the principles of creating an entire system from scratch, which has been an invaluable experience.

As I finished my project and started presenting it to other people, both to gather opinions and during user testing phases, it was really reassuring to know that my hard work was appreciated as people always had nice things to say about the system or gave me things to further think about. This feedback is important, as mentioned in the limitations section of the evaluation chapter, the system still lacks certain elements necessary for real-world application. Nonetheless, I firmly believe that with further refinements to fix these issues, this project could be successfully deployed in real-life scenarios. Furthermore, I think that with some alterations, this system could have other applications beyond the educational context. It could be adapted to be used in business environments to monitor employee attendance or even in healthcare settings to track patient movements and, to track comfort levels or patient requests. I am excited about the potential improvements and expansions to this system, which I could even launch a startup based on it.

References

Allen, D.O. and Webber, D.J. (2010) 'Attendance and exam performance at university: a case study', *Research in Post-Compulsory Education*, 15(1), pp. 33–47. Available at: <https://doi.org/10.1080/13596740903565319>.

Apple Inc. (2024) *Enrolment - Support - Apple Developer, Enrolment*. Available at: <https://developer.apple.com/support/enrollment/> (Accessed: 13 May 2024).

Ashton, K. (2009) *That 'Internet of Things' Thing - RFID Journal, That 'Internet of Things' Thing*. Available at: <https://web.archive.org/web/20130415194522/http://www.rfidjournal.com/articles/view?4986> (Accessed: 5 May 2024).

Basheer, M. and Raghu, C.V. (2012) 'Fingerprint attendance system for classroom needs', in, pp. 433–438. Available at: <https://doi.org/10.1109/INDCON.2012.6420657>.

Batı, A. *et al.* (2012) 'Why do students miss lectures.? A study of lecture attendance amongst students of health science', *Nurse education today*, 33. Available at: <https://doi.org/10.1016/j.nedt.2012.07.010>.

Carnegie Mellon University (2024) *The 'Only' Coke Machine on the Internet, The 'Only' Coke Machine on the Internet*. Available at: https://www.cs.cmu.edu/~coke/history_long.txt (Accessed: 5 May 2024).

Clark, G. *et al.* (2011) 'Attendance and Performance: Correlations and Motives in Lecture-Based Modules', *Journal of Geography in Higher Education*, 35, pp. 199–215. Available at: <https://doi.org/10.1080/03098265.2010.524196>.

Credé, M., Roch, S.G. and Kieszczynka, U.M. (2010) 'Class Attendance in College: A Meta-Analytic Review of the Relationship of Class Attendance With Grades and Student Characteristics', *Review of Educational Research*, 80(2), pp. 272–295.

Day, P. (2015) 'Peter Day's World of Business Podcast'. (The World of Business). Available at: <https://www.bbc.co.uk/programmes/p02s1gwx> (Accessed: 5 May 2024).

Department for Education (2023) *Why is school attendance so important and what are the risks of missing a day? - The Education Hub*. Available at: <https://educationhub.blog.gov.uk/2023/05/18/school-attendance-important-risks-missing-day/> (Accessed: 30 December 2023).

Dixon, J. and Abuzneid, A. (2020) 'An NFC Based Student Attendance Tracking/Monitoring System Using an IoT Approach', in *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*. *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 1082–1086. Available at: <https://doi.org/10.1109/CSCI51800.2020.00201>.

Food and Drug Administration (2008) *Technologies for Prescription Drug Identification, Validation, Track and Trace, or Authentication; Request for Information, Federal Register*. Available at: <https://www.federalregister.gov/documents/2008/03/20/E8-5599/technologies-for-prescription-drug-identification-validation-track-and-trace-or-authentication> (Accessed: 5 May 2024).

Gumiński, A., Dohn, K. and OLOYEDE, E. (2023) 'Advantages and disadvantages of traditional and agile methods in software development projects – case study', *Scientific Papers of Silesian University of Technology Organization and Management Series*, 2023. Available at: <https://doi.org/10.29119/1641-3466.2023.188.11>.

Halpern, N. (2007) 'The impact of attendance and student characteristics on academic achievement: Findings from an undergraduate business management module', *Journal of Further and Higher Education*, 31, pp. 335–349. Available at: <https://doi.org/10.1080/03098770701626017>.

Honglei, R., Song, Y. and Yang, S. (2016) 'An automated student attendance tracking system based on voiceprint and location', in. Available at: <https://doi.org/10.1109/ICCSE.2016.7581583>.

International Standards Organization (2023) 'ISO 28560-2:2023 Information and documentation RFID in libraries Part 2: Encoding of RFID data elements based on rules from ISO/IEC 15962'. International Standards Organization. Available at: <https://www.iso.org/standard/85149.html> (Accessed: 5 May 2024).

Islam Mazumdar, A.T. *et al.* (2022) ‘NFC-based Mobile Application for Student Attendance in Institution of Higher Learning’, in *2022 1st International Conference on AI in Cybersecurity (ICAIC)*. *2022 1st International Conference on AI in Cybersecurity (ICAIC)*, pp. 1–5. Available at: <https://doi.org/10.1109/ICAIC53980.2022.9896975>.

James Manyika *et al.* (2013) *Disruptive technologies: Advances that will transform life, business, and the global economy*. McKinsey & Company, p. 176. Available at: https://www.mckinsey.com/~/media/mckinsey/business%20functions/mckinsey%20digital/our%20insights/disruptive%20technologies/mgi_disruptive_technologies_full_report_may2013.pdf (Accessed: 5 May 2024).

Kelly, G. (2012) ‘Lecture attendance rates at university and related factors’, *Journal of Further and Higher Education*, 36, pp. 17–40. Available at: <https://doi.org/10.1080/0309877X.2011.596196>.

Kunst, A. (2024) *iPhone ownership by age in the UK 2023*, Statista. Available at: <https://www.statista.com/forecasts/1284304/iphone-user-share-in-the-united-kingdom-by-age> (Accessed: 5 May 2024).

Lange, P.I.B. and Steck, T.J.P. (2014) *Near Field Communication Its adoption process and technology acceptance*. Master Thesis. Lund University School of Economics and Management. Available at: <https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=7aa5bc6bbf60ec6cc788067daf4daef0e7150b3> (Accessed: 5 May 2024).

Lim, J.H. *et al.* (2017) ‘Automated classroom monitoring with connected visioning system’, in *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*. *2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pp. 386–393. Available at: <https://doi.org/10.1109/APSIPA.2017.8282063>.

Longhurst, R.J. (1999) ‘Why Aren’t They Here? Student absenteeism in a further education college’, *Journal of Further and Higher Education*, 23(1), pp. 61–80. Available at: <https://doi.org/10.1080/0309877990230106>.

Lukas, S. *et al.* (2016) ‘Student attendance system in classroom using face recognition technique’, in, pp. 1032–1035. Available at: <https://doi.org/10.1109/ICTC.2016.7763360>.

Macfarlane, B. (2013) ‘The Surveillance of Learning: A Critical Analysis of University Attendance Policies’, *Higher Education Quarterly*, 67. Available at: <https://doi.org/10.1111/hequ.12016>.

Marburger, D. (2006) ‘Does Mandatory Attendance Improve Student Performance?’, *Journal of Economic Education*, 37, pp. 148–155. Available at: <https://doi.org/10.3200/JECE.37.2.148-155>.

Memane, R. *et al.* (2022) ‘Attendance Monitoring System Using Fingerprint Authentication’, in *2022 6th International Conference on Computing, Communication, Control and Automation (ICCUBEA)*. 2022 6th International Conference on Computing, Communication, Control and Automation (ICCUBEA), pp. 1–6. Available at: <https://doi.org/10.1109/ICCUBEA54992.2022.10010791>.

Narkhede, N. *et al.* (2023) ‘Facial Recognition and Machine Learning-based Student Attendance Monitoring System’, in *2023 3rd International Conference on Intelligent Technologies (CONIT)*. 2023 3rd International Conference on Intelligent Technologies (CONIT), pp. 1–7. Available at: <https://doi.org/10.1109/CONIT59222.2023.10205631>.

NFC Forum (2024) *NFC Technology - Exploring the Fundamentals and Applications of Near Field Communication*, NFC Forum. Available at: <https://nfc-forum.org/learn/nfc-technology/> (Accessed: 5 May 2024).

Oldfield, J. *et al.* (2017) ‘Psychological and demographic predictors of undergraduate non-attendance at university lectures and seminars’, *Journal of Further and Higher Education*, 42, pp. 1–15. Available at: <https://doi.org/10.1080/0309877X.2017.1301404>.

Patel, U. and Swaminarayan, P. (2014) ‘Computer Science and Management Studies Development of a Student Attendance Management System Using RFID and Face Recognition: A Review’, *International Journal of Advance Research in Computer Science and Management*, 2.

Ray, M. *et al.* (2023) *Editions and supported features of SQL Server 2019 - SQL Server, Microsoft Build*. Available at: <https://learn.microsoft.com/en-us/sql/sql-server/editions-and-components-of-sql-server-2019?view=sql-server-ver16> (Accessed: 12 May 2024).

Sanders, L.D., Mair, C. and James, R. (2016) 'Detecting uncertainty, predicting outcome for first year students', *Journal of Applied Research in Higher Education*, 8(3), pp. 346–359. Available at: <https://doi.org/10.1108/JARHE-10-2015-0076>.

Snyder, J.L. *et al.* (2014) 'What Is the Influence of a Compulsory Attendance Policy on Absenteeism and Performance?', *Journal of Education for Business*, 89(8), pp. 433–440. Available at: <https://doi.org/10.1080/08832323.2014.933155>.

Sony Corporation and Philips Corporation (2002) *PHILIPS AND SONY ANNOUNCE STRATEGIC COOPERATION TO DEFINE NEXT GENERATION NEAR FIELD RADIO-FREQUENCY COMMUNICATIONS*, *Sony Group Portal - Sony Global Headquarters*. Available at: <http://www.sony.com/en/SonyInfo/News/Press/200209/02-0905E/index.html> (Accessed: 5 May 2024).

Tombeng, M.T. *et al.* (2023) 'Integrated Attendance System using NFC Technology', in *2023 5th International Conference on Cybernetics and Intelligent System (ICORIS)*. *2023 5th International Conference on Cybernetics and Intelligent System (ICORIS)*, pp. 1–6. Available at: <https://doi.org/10.1109/ICORIS60118.2023.10352289>.

UK Finance (2021) *UK PAYMENT MARKETS SUMMARY 2021*, p. 8. Available at: <https://www.ukfinance.org.uk/sites/default/files/uploads/SUMMARY-UK-Payment-Markets-2021-FINAL.pdf> (Accessed: 5 May 2024).

Vasseur, J.-P. and Kaufmann, M. (2010) *Interconnecting Smart Objects with IP*. Morgan Kaufmann.

Weinstein, R. (2005) 'RFID: a technical overview and its application to the enterprise', *IT Professional*, 7(3), pp. 27–33. Available at: <https://doi.org/10.1109/MITP.2005.69>.

Yorke, M. and Longden, B. (2007) 'Retention and Student Success in Higher Education, edited by Mantz Yorke and Bernard Longden. The Society for Research into Higher Education, Open

University Press, 2004', *Interchange*, 38, pp. 93–95. Available at: <https://doi.org/10.1007/s10780-007-9016-1>.

Younis, M. *et al.* (2012) 'Design and Implementation of a Scalable RFID-Based. Attendance System with an Intelligent Scheduling. Technique', *Wireless Personal Communications*, 71. Available at: <https://doi.org/10.1007/s11277-012-0929-3>.

Appendices

Appendix A

Full list of Git Commits for the project hosted on GitHub:

Commits		
main	All users	All time
Commits on May 6, 2024		
made changes	DravenMolten committed last week	382a93e
Commits on May 5, 2024		
sound works	DravenMolten committed 2 weeks ago	262c73b
made sound work	DravenMolten committed 2 weeks ago	8c412e9
Commits on Apr 17, 2024		
Fixes on content, colour	DravenMolten committed last month	f8ba37e
Changed location tech, Set colours	DravenMolten committed last month	1ba6a37
Reapply "Made admin attendance page"	DravenMolten committed last month	9439766
Revert "Made admin attendance page"	DravenMolten committed last month	cb6adce
Made admin attendance page	DravenMolten committed last month	842ff68
Commits on Apr 15, 2024		
Alert on delete, UI changes, Most manage works	DravenMolten committed last month	ee2b315
Commits on Apr 13, 2024		
Added the individual manage screens	DravenMolten committed last month	4c8ba47
Commits on Apr 12, 2024		
Added a screen to manage every table	DravenMolten committed last month	476c48f
Commits on Apr 10, 2024		
Added timetable screen	DravenMolten committed last month	ec6a84b
Sending attendance, location, and device info	DravenMolten committed last month	d580a0b
Commits on Apr 9, 2024		
Added dropdown list to nfc screen	DravenMolten committed last month	45a66c8
Commits on Apr 7, 2024		
Changed globals to globalstylejs	DravenMolten committed last month	f108b3f
Commits on Apr 6, 2024		
Log in works	DravenMolten committed last month	786cfe0
Commits on Mar 29, 2024		
Added logo and styles	DravenMolten committed 2 months ago	165ba0b
Commits on Mar 27, 2024		
Writing nfc data from text	DravenMolten committed 2 months ago	908ca83
Fixed pages, added writing	DravenMolten committed 2 months ago	5c5fb3a
Commits on Mar 25, 2024		
Added more login measures	DravenMolten committed 2 months ago	d3096cc
Commits on Mar 22, 2024		
Can read the nfc data	DravenMolten committed 2 months ago	8342c43
Added text to signal nfc status	DravenMolten committed 2 months ago	d39e6e0
Commits on Mar 21, 2024		
Added the ability to read nfc data	DravenMolten committed 2 months ago	a76d894
Added every other screen and installing packages	DravenMolten committed 2 months ago	ee0cfd6
Commits on Mar 20, 2024		
Added comments	DravenMolten committed 2 months ago	f14b6db
Second commit of the app maybe	DravenMolten committed 2 months ago	b51c9e0
first commit for the app	DravenMolten committed 2 months ago	3f9e609
Commits on Mar 19, 2024		
Initial commit	DravenMolten committed 2 months ago	f6cf79e

Appendix B

Participant information sheet:

Participant Information Sheet

Invitation Paragraph:

I would like to invite you to take part in this research study. Before you decide, it's important for you to understand why the research is being conducted and what your involvement will entail. Please take the time to read the following information carefully. Feel free to ask questions if anything is unclear or if you would like more information. Take the time you need to decide whether you would like to participate

What is the Purpose of the Study?

The purpose of this study is to develop and evaluate a mobile attendance application that utilizes Near Field Communication (NFC) technology. This application aims to streamline the attendance tracking process in various settings such as educational institutions, workplaces, and events.

Why Have I Been Invited?

You have been invited to participate because you are a potential user of the mobile attendance application since you are a student at MMU. We are seeking participants from different years and courses to ensure the application meets the needs of various user groups.

Do I Have to Take Part?

Participation in this research study is entirely voluntary. You have the right to decline participation or withdraw from the study at any time without providing a reason.

What Will Happen to Me If I Take Part?

If you choose to take part in the study, you might be asked to install the mobile attendance application on your smartphone and test it. You will also be asked to provide feedback regarding things such as the User Interface and accessibility of the app.

Expenses and Payments?

There are no expenses or payments associated with participation in this study.

What Will I Have to Do?

Your responsibilities could include installing the mobile attendance application, testing it, and will include providing feedback on your experience with the application.

What are the Possible Disadvantages and Risks of Taking Part?

Potential risks associated with participation include the loss of personal data in the event of any issues with the database. However, we will take measures to secure your data and minimize such risks.

What are the Possible Benefits of Taking Part?

While we cannot guarantee direct benefits to you, your participation will contribute to the development of a novel attendance tracking solution that may benefit future students and various other users.

Participant consent form:

Centre Number:

Study Number:

Patient Identification Number for this trial:

CONSENT FORM

Title of Project: **Mobile attendance application using NFC**

Name of Researcher: **Taha Gorkem Sarac**

Introduction:

Thank you for your interest in participating in our project titled "Mobile Attendance Application Using NFC." Before you decide to participate, it is important that you understand the purpose of the project and what your involvement will entail. Please read the following information carefully and feel free to ask any questions before proceeding.

Please initial
all boxes

1. I confirm that I have read and understand the information sheet dated **22/02/2024** (version **1** for the above study. I have had the opportunity to consider the information, ask questions and have had these answered satisfactorily.
2. I understand that my participation is voluntary and that I am free to withdraw at any time without giving any reason, without my medical care or legal rights being affected.
3. I understand that participants might be asked to install the mobile attendance application on their smartphones. The participants can decline to do so. The application will utilize NFC technology to facilitate the attendance tracking process. Data collected during the study will be anonymized and used for research purposes only.
4. I understand any input or feedback received from the participants will will be anonymized and used for research purposes only.
5. I agree to take part in the above study.

☐☐☐☐☐

User questionnaire:



Mobile attendance app questionnaire

This is the end-user questionnaire for the "Mobile attendance application using NFC" project made by Taha Gorkem Sarac. By filling in this questionnaire you are accepting the terms and conditions of the consent form that you should have signed. See the participant information sheet for extra information on the project and your answers.

thgrkmsrc@gmail.com [Switch accounts](#)



Not shared

* Indicates required question

How easy was the overall usability of the application? (Was the app easy to use?) *
(1 worst - 10 best)

1	2	3	4	5	6	7	8	9	10
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

How would you rate the overall Student Interface of the application? (1 worst - 10 * best)

1	2	3	4	5	6	7	8	9	10
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

What was your reasoning behind your rating of the student interface?

Your answer

How would you rate the overall Admin Interface of the application? (1 worst - 10 * best)

1	2	3	4	5	6	7	8	9	10
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

What was your reasoning behind your rating of the Admin interface?

Your answer

How likely would you be to use this application Instead of your student card if they could both be used in lectures/labs? (1 unlikely - 10 very likely) *

1	2	3	4	5	6	7	8	9	10
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

How likely would you be to use this application Alongside of your student card if they could both be used in lectures/labs? (1 unlikely - 10 very likely) *

1	2	3	4	5	6	7	8	9	10
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Have you encountered any bugs or issues while testing out the application? if so, what?

Your answer

Do you have any suggestions or additional feedback about the application?

Your answer

Submit

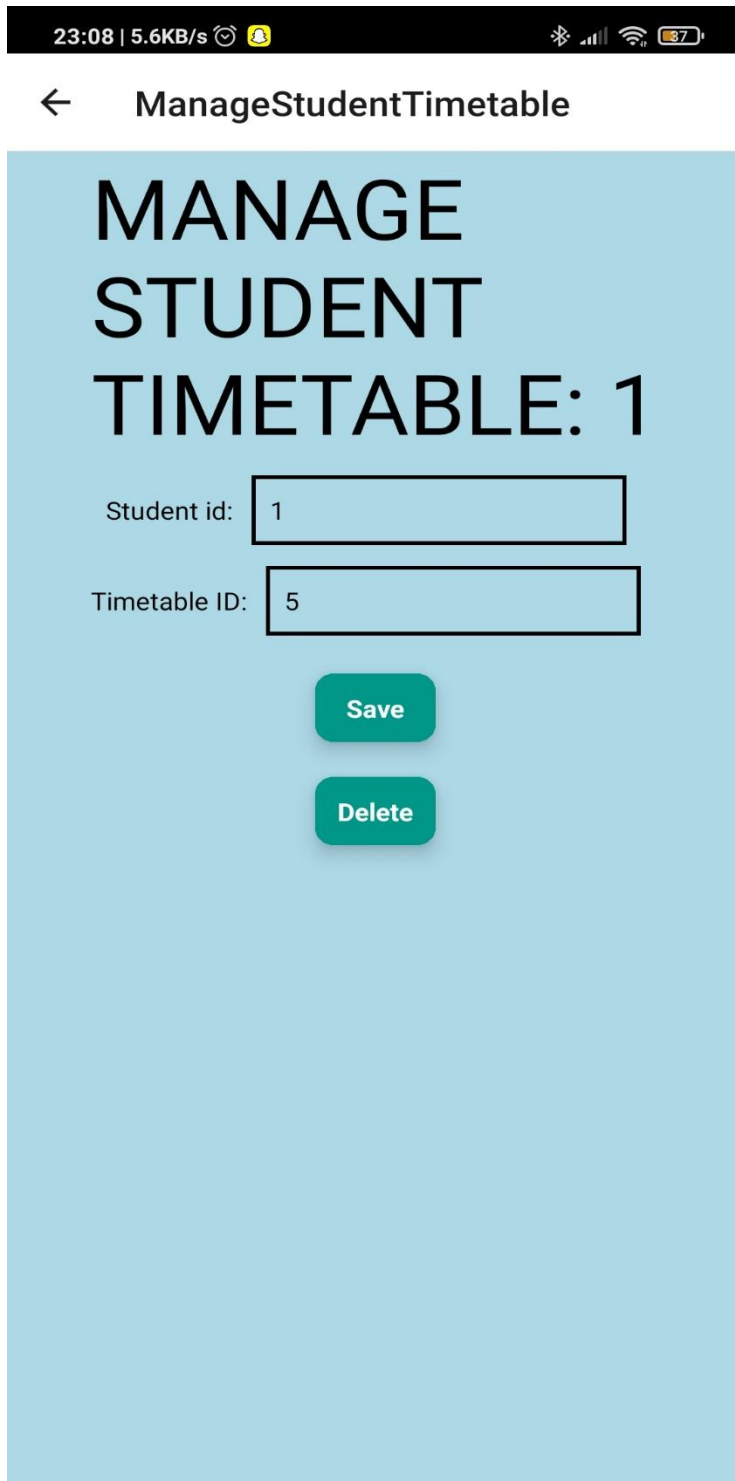


Page 1 of 1

Clear form

Appendix C

The screenshots of all the different pages within the mobile application:



The screenshot shows a mobile application interface for managing student timetables. At the top, there is a status bar with the time 23:08, a data speed of 5.6KB/s, and various connectivity icons. Below the status bar is a navigation bar with a back arrow and the title "ManageStudentTimetable". The main content area has a light blue background and features the title "MANAGE STUDENT TIMETABLE: 1" in large, bold, black capital letters. Below the title, there are two input fields: "Student id:" with the value "1" and "Timetable ID:" with the value "5". At the bottom of the form, there are two green buttons: "Save" and "Delete".

23:08 | 5.6KB/s

← ManageStudentTimetable

MANAGE STUDENT TIMETABLE: 1

Student id: 1

Timetable ID: 5

Save

Delete



← Manage

MANAGE STUDENT

student id: 1

timetable id: 5

student id: 2

timetable id: 1

student id: 3

timetable id: 3

student id: 4

timetable id: 4

student id: 5

timetable id: 2

Insert Student timetable

← ManageT timetable

MANAGE TIMETABLE: 1

Timetable ID:

1

Timetable name:

computer science group 1

Active:

true

Save

Delete

← Manage

MANAGE TIMETABLES

ID: 1

Name: computer science group 1
Active: true

ID: 2

Name: computer science group 2
Active: true

ID: 3

Name: business group 1
Active: true

ID: 4

Name: biomedical science group 1
Active: true

ID: 5

Name: admin group
Active: true

Insert Timetable

← ManageT timetableContent

MANAGE TIMETABLE CONTENT: 1

T timetable content ID:

1

T timetable id:

1

Unit id:

6

Day:

1

Start time:

09:00:01

End time:

10:00:00

Classroom id:

1

Save

Delete

← Manage

MANAGE TIMETABLES

ID: 1

Timetable id: 1

Unit: Artificial intelligence

Classroom id: 1

Day: 1

Start date: 09:00:01

End date: 10:00:00

ID: 2

Timetable id: 1

Unit: Distributed computing

Classroom id: 2

Day: 1

Start date: 10:00:01

End date: 11:00:00

ID: 3

Timetable id: 1

Unit: Operating systems

Classroom id: 1

Day: 1

Start date: 11:00:01

End date: 12:00:00

Insert Timetables content

← ManageStudents

MANAGE STUDENT: 5

Student ID:

5

student name:

gorkem

student surname:

sarac

Email:

gorkem@stu.mmu.ac.uk

Phone:

05714144110

Course id:

2

Start date:

2023-09-23T00:00:00

End date:

2026-07-17T00:00:00

Last login date:

2024-05-13T23:06:05

Device info:

xiaomi-mi9-android-10

Password:

1234

Active:

true

Save

Delete

[←](#) Manage

MANAGE STUDENTS

ID: 1

Name: adminname

Surname: adminsurname

Email: admin

Phone: 07402341055

Course: 1

Start date: 2000-01-01T00:00:00

End date: 2099-01-01T00:00:00

Login:2024-05-13T23:06:38.863

Device:

Pass:admin

Active: true

ID: 2

Name: john

Surname: doe

Email: q

Phone: 07502344410

Course: 2

Start date: 2023-09-23T00:00:00

End date: 2026-07-17T00:00:00

Login:2024-05-11T14:09:12.563

Device:

Pass:q

Active: true

[Insert Student](#)

← ManageClassrooms

MANAGE CLASSROOM: 1

Classroom ID:

1

Classroom name:

101

Building name:

John Dalton

Latitude:

53.472095

Longitude:

-2.241826

Active:

true

Save

Delete

← ManageUnits

MANAGE UNIT: 1

Unit ID:

Unit name:

Course ID:

Course name:

Active:

Save

Delete

← Manage

MANAGE UNITS

ID: 1

Unit: Programming and algorithms
Active: true

ID: 2

Unit: Databases
Active: true

ID: 3

Unit: Networks
Active: true

ID: 4

Unit: Operating systems
Active: true

ID: 5

Unit: Distributed computing
Active: true

ID: 6

Unit: Artificial intelligence

Insert Unit

← ManageCourses

MANAGE COURSE: 1

Course ID:

Course name:

Start date:

Active:

Save

Delete

← Manage

MANAGE COURSES

ID: 1

Course: AdminCourse

Active: true

Start date: 2000-01-01T00:00:00

ID: 2

Course: Computer science

Active: true

Start date: 2023-09-26T00:00:00

ID: 3

Course: Business

Active: true

Start date: 2023-09-29T00:00:00

ID: 4

Course: Biomedical science

Active: true

Start date: 2023-09-20T00:00:00

Insert Course

← ManageNFCTags

WRITE NFC TAGS

status:awaiting-start the nfc

TAP TO WRITE TO TAG

Select the classroom to write onto the NFC tag

Select option ▼

John Dalton-101

John Dalton-102

John Dalton-201

John Dalton-202

John Dalton-301

[← AttendanceRate](#)

ATTENDANCE

Weekly:

Attended 0 out of 6 sessions, at 0 %

Monthly:

Attended 4 out of 36 sessions, at 11 %

LAST ATTENDED SESSIONS:

2024-05-07 - Networks

12:00:01 to 12:30:00

Room: 201

2024-05-07 - Operating systems

11:00:01 to 11:30:00

Room: 201

2024-05-07 - Distributed computing

10:00:01 to 10:30:00

Room: 201

2024-05-06 - Programming and algorithms

09:00:01 to 10:00:00

Room: 101



StudentHome

WELCOME GORKEM SARAC

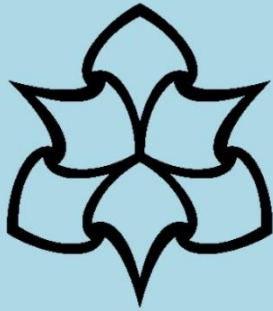


[Check Attendance Rate](#)

[View Timetable](#)

Login

Welcome to the MMU attendance app!



**Manchester
Metropolitan
University**

Username:

Password:

Login

App made by Taha Gorkem Sarac
Version 1.0.0

← Manage

MANAGE CLASSROOMS

ID: 1

Room: 101
Building: John Dalton
lat: 53.472095
lon: -2.241826
Active: true

ID: 2

Room: 102
Building: John Dalton
lat: 53.472095
lon: -2.241826
Active: true

ID: 3

Room: 201
Building: John Dalton
lat: 53.472095
lon: -2.241826
Active: true

ID: 4

Room: 202
Building: John Dalton

Insert Classroom

← Manage

MANAGE ATTENDANCE

Attendance ID: 5

Student id: 5

Student name: gorkem

Unit: Networks

Tap date: 2024-05-07T12:19:06.48

Device info: Xiaomi_MI 9_Android_10

Lat: 53.47077941894531

Lon: -2.240291118621826

Attendance ID: 4

Student id: 5

Student name: gorkem

Unit: Operating systems

Tap date: 2024-05-07T11:23:02.057

Device info: Xiaomi_MI 9_Android_10

Lat: 53.47188949584961

Lon: -2.2401692867279053

Attendance ID: 3

Student id: 5

Student name: gorkem

Unit: Distributed computing

Tap date: 2024-05-07T10:01:39.893

Device info: Xiaomi_MI 9_Android_10

Student ID:

5