

An NFC Enabled Student Card System with API Integration

**Raissa Pululu**

**K00260175**

**Supervised by Brendan Watson**

A Final Year Project submitted in fulfilment of the requirements of the Technological University of the Shannon: Midlands Midwest, for the degree of Bachelor of Science (Honours) in Software Development

# **Acknowledgements**

I would like to thank and acknowledge the staff of TUS that have been there through my continuous growth during the past 4 years. The advice and feedback throughout the development of this project and the 4 years of this course has been incredibly valuable and shows how I have been able to get here.

I would like to thank the supervisor of the project Brendan Watson for supporting throughout this project. The aim of this project would not have been conceptualised nor achieved without his support.

I would like to thank my parents for their enormous efforts throughout my childhood to ensure I get to this point in my education. Their support is something I will always be indebted to.

I would like to finally thank the class of BSc Software Development 2024; it is incredible that so many of us are still here battling to the end. I could not be prouder of the work we have all put in to get here.

# **Abstract**

NFC and RFID are two of the main driving forces moving most of the world's everyday tasks, taking many users from physical to digital. In the last few decades, we have gone from the physical world to now a contactless world. These efficient ways of navigating the world have been long integrated into our lives such as Transport for Ireland’s Leap Card, payment transactions with contactless cards and mobile wallets such as the Apple Pay and Google Pay, even as far as BMW using the Apple Wallet to develop a digital card to unlock their cars.

This project outlines the in-depth research completed into NFC and RFID technologies and how it can be applied to a simple application with the use of a Raspberry Pi. The goal of this project is to model a student card system with the use of NFC enabled tags and readers that can communicate to a Raspberry Pi. The application should be able to complete simple tasks such as modelling unlocking a door, signing in students for attendance, add a new student to the system, top up the balance of each card to allow the user to pay for parking along with showing how we can create a sample API based on the project to showcase how the project can be used for many use cases.

**Table of Contents**

[**Acknowledgements** 2](#_Toc164810629)

[**Abstract** 3](#_Toc164810630)

[**Table of Figures** 6](#_Toc164810631)

[**Table of Tables** 7](#_Toc164810632)

[**Introduction** 8](#_Toc164810633)

[**Literature Review** 10](#_Toc164810634)

[**NFC Technology** 10](#_Toc164810635)

[**History and Evolution** 10](#_Toc164810636)

[**Mechanisms and Operations of NFC and RFID** 12](#_Toc164810637)

[**Applications** 17](#_Toc164810638)

[**NFC Enabled Student Cards** 18](#_Toc164810639)

[**Real World Applications** 18](#_Toc164810640)

[**API Integration** 22](#_Toc164810641)

[**Definition** 22](#_Toc164810642)

[**Challenges** 23](#_Toc164810643)

[**Future Trends** 24](#_Toc164810644)

[**Conclusion** 25](#_Toc164810645)

[**System Analysis and Design** 27](#_Toc164810646)

[**Introduction** 27](#_Toc164810647)

[**Analysis** 27](#_Toc164810648)

[**Hardware Requirements** 29](#_Toc164810649)

[**Software Requirements** 31](#_Toc164810650)

[**System Analysis** 32](#_Toc164810651)

[**Requirements Analysis** 32](#_Toc164810652)

[**User Analysis** 33](#_Toc164810653)

[**System Design** 34](#_Toc164810654)

[**Hardware Parts** 34](#_Toc164810655)

[**Hardware Design** 37](#_Toc164810656)

[**Architecture Design** 38](#_Toc164810657)

[**Interface Design** 40](#_Toc164810658)

[**Database Design** 41](#_Toc164810659)

[**Summary** 43](#_Toc164810660)

[**Implementation** 44](#_Toc164810661)

[**Introduction** 44](#_Toc164810662)

[**Technical Information** 45](#_Toc164810663)

[**System and Software Design** 45](#_Toc164810664)

[**Circuit Design** 47](#_Toc164810665)

[**Project Management** 48](#_Toc164810666)

[**Design Quality** 49](#_Toc164810667)

[**Program Reliability** 51](#_Toc164810668)

[**Software Reuse** 52](#_Toc164810669)

[**Design Decision** 53](#_Toc164810670)

[**System Architecture** 57](#_Toc164810671)

[**Overall Structure** 57](#_Toc164810672)

[**Assets Directory Structure** 58](#_Toc164810673)

[**Dependencies Directory Structure** 59](#_Toc164810674)

[**Modules Directory Structure** 60](#_Toc164810675)

[**Completed System** 65](#_Toc164810676)

[**Testing and Results** 73](#_Toc164810677)

[**Introduction** 73](#_Toc164810678)

[**Software Testing** 73](#_Toc164810679)

[**Test Cases** 75](#_Toc164810680)

[**Unit Tests** 75](#_Toc164810681)

[**Unit Test Results** 79](#_Toc164810682)

[**Functional Tests** 79](#_Toc164810683)

[**Test Conclusion** 85](#_Toc164810684)

[**Conclusion** 86](#_Toc164810685)

[**Project Summary** 86](#_Toc164810686)

[**Findings** 86](#_Toc164810687)

[**Improvements** 87](#_Toc164810688)

[**Future Work** 88](#_Toc164810689)

[**Final Thoughts** 88](#_Toc164810690)

[**List of Abbreviations** 90](#_Toc164810691)

[**References** 93](#_Toc164810692)

[**Appendix** 102](#_Toc164810693)

[**Appexdix A – Requirements Analysis Document** 102](#_Toc164810694)

[**Appendix B – Declaration of Authorship** 105](#_Toc164810695)

# **Table of Figures**

[Figure 1 NFC Reader Modes (STMicroelectronics, 2022) 15](#_Toc164809151)

[Figure 2 PN532 35](#_Toc164809152)

[Figure 3 PN532 35](#_Toc164809153)

[Figure 4. ARC122U 36](#_Toc164809154)

[Figure 5 RC522 37](#_Toc164809155)

[Figure 6 RC522 37](#_Toc164809156)

[Figure 7 Sample Circuit Diagram 38](#_Toc164809157)

[Figure 8 Sequence Diagram 40](#_Toc164809158)

[Figure 9 Tkinter Sample Interface 41](#_Toc164809159)

[Figure 10 PyQT Sample Interface 42](#_Toc164809160)

[Figure 11 EER Diagram 43](#_Toc164809161)

[Figure 12 Exploratory Software Development 47](#_Toc164809162)

[Figure 13 System Architecture 48](#_Toc164809163)

[Figure 14 A top level view of the physical circuit 49](#_Toc164809164)

[Figure 15 A screenshot of naming conventions in the project 51](#_Toc164809165)

[Figure 16 A screenshot of error handling 52](#_Toc164809166)

[Figure 17 A screenshot of error handling 52](#_Toc164809167)

[Figure 18 A screenshot of threads 53](#_Toc164809168)

[Figure 19 Sample code from I2C\_LCD\_driver.py 60](#_Toc164809169)

[Figure 20 API Code Snippet 61](#_Toc164809170)

[Figure 21 check\_attendance.py Code Snippet 62](#_Toc164809171)

[Figure 22 main.py Code Snippet 63](#_Toc164809172)

[Figure 23 main.py Code Snippet 63](#_Toc164809173)

[Figure 24 save\_user.py Code Snippet 64](#_Toc164809174)

[Figure 25 unlock.py Code Snippet 65](#_Toc164809175)

[Figure 26 A screenshot of the main system 66](#_Toc164809176)

[Figure 27 A screenshot of the parking system 66](#_Toc164809177)

[Figure 28 A screenshot of the top up system 67](#_Toc164809178)

[Figure 29 A picture of the LCD displaying messages 67](#_Toc164809179)

[Figure 30 A picture of the LCD displaying messages 68](#_Toc164809180)

[Figure 31 A screenshot of the API call for attendance tracking 68](#_Toc164809181)

[Figure 32 A screenshot of the department dropdown 69](#_Toc164809182)

[Figure 33 A screenshot of the Unlock Door method 70](#_Toc164809183)

[Figure 34 A screenshot of the Register User method 70](#_Toc164809184)

[Figure 35 A screenshot of the parking method 71](#_Toc164809185)

[Figure 36 A screenshot of the attendance system 71](#_Toc164809186)

[Figure 37 A screenshot of the top up method 72](#_Toc164809187)

[Figure 38 A screenshot of the top up method 73](#_Toc164809188)

[Figure 39 A code snippet of main.py unit tests 76](#_Toc164809189)

[Figure 40 A code snippet of parking\_sys.py unit tests 78](#_Toc164809190)

[Figure 41 A code snippet of top\_up.py unit tests 79](#_Toc164809191)

[Figure 42 A screenshot of the unit test results 80](#_Toc164809192)

# **Table of Tables**

[Table 1 Hardware Requirements 37](#_Toc164717469)

[Table 2 Software Requirements 38](#_Toc164717470)

[Table 3 Specification of PN532 41](#_Toc164717471)

[Table 4 Specification of ARC122U 42](#_Toc164717472)

[Table 5 Specification of RC522 43](#_Toc164717473)

[Table 6 Functional test table for the API in main.py 92](#_Toc164717474)

[Table 7 Functional test table for unlock.py in main.py 93](#_Toc164717475)

[Table 8 Functional test table for save\_user.py in main.py 93](#_Toc164717476)

[Table 9 Functional test table for check\_attendance.py in main.py 94](#_Toc164717477)

[Table 10 Functional test table for top\_up.py 95](#_Toc164717478)

[Table 11 Functional test table for parking\_sys.py 96](#_Toc164717479)

[Table 12 Functional test table for main.py 97](#_Toc164717480)

# **Introduction**

The 21st century is now being described as the digital era. Seamless, contactless, and wireless interactions are now becoming the new norm using technologies such as Wi-Fi, Bluetooth, NFC or Near Field Communication, the internet and so much more. Our methods of communication are changing every day, and it seems as though it is up to the regular individual to keep up with the overwhelming new standards as they rollout. Not only do we see these being implemented into our homes and workplaces, the public sector; public transport, mobile industries, automotive infrastructures have rapidly adopted these technologies to offer continuous ways of staying online using all the above but most importantly NFC and APIs (cj&co, 2023).

NFC in general has been the strongest force conducting real time and real-world applications when considering going digital or contactless. Payment transactions, transport, mobile payments are globally using NFC chips integrated onto physical debit/credit cards, NFC integrated basic cards, tags or even QR codes. By 2026, it is predicted that contactless payments are expected to rapidly increase by 221%, contactless cards expecting to increase by 119% in the same period which will come from majority of mobile payments. (Brophy and King, 2023). As of 2018, there are 3.4 billion active smartphones globally, two billion of these are NFC enabled devices meaning 20% of the global population are using NFC enabled devices. (NFC Forum, 2019).

Software as we know it today has become more expandable, developers are looking for easier ways to complete projects, apply the newest features and satisfy stakeholders and their end users. APIs have become a growing facilitator in the development of software. They are not only a flexible tool; they aim to assist the developers in creating fast, intuitive, and reliable systems. APIs grant us the ability to plug and play software with minimal effort, reducing the workload as the software has simply been implemented and wrapped around existing projects.

Statistics show that 90% of software developers integrate APIs into their work. 69% using third-party APIs and 20% using internal or private APIs. Principal factors considered for using APIs include performance, security, reliability, documentation, and scalability with performance being the highest factor at 72% and scalability being the lowest at 59%. Developers that create APIs find that integration with internal systems is crucial at 83% and speeding up development times the least important at 42%. (Kolya Hnatyuk, 2023).

Following these statistics, we can see that NFC and APIs are becoming global tools for all to use and that is how this final year project aims to take advantage of this technology and proposes an NFC enabled student card system with API integration that can be built around existing campus infrastructure. Systems like these already exist and the findings of these systems will be discussed in this paper. Solutions implemented by other developers see the system as not only an identification system but as solution tailored to enhance not the student experience and academia experience. Many of these systems can be transformed into APIs which allow the future developers and system administrators to easily implement innovative programs to existing solutions.

The goal of this Final Year Project is to develop and implement an NFC Enabled Student Card system with API Integration that will grow and improve student cards based on five use cases. The technology intends to offer a comprehensive solution for employees and students that includes faster class attendance tracking, paid parking, and a secure system with door access control. It will function as a proof-of-concept system with NFC hardware integration by utilising a Raspberry Pi as the microcontroller.

The goal of the project is to provide a system to relieve staff from tracking attendance during class times, have shorter lines for parking and have a secure system for door access control for specific areas on campus, offices etc.

The system would proceed to track attendance based on each scan of a student card with information including the name, the date and time of each student card and provide a callable API that would be available to the staff member. Depending on the permissions the administrator sets, the system would allow some cards to only access rooms. Without the use of a cash register, the card may be used to make purchases in parking.

This Final Year Project aims to cover NFC technology integration, with software, hardware, and model it on a smaller scale. Reviewing and understanding what NFC is, how it functions, and how to implement access control functions like door access control. Studying the use of NFC and how to use public Python libraries to obtain data from the readers and cards and how to store them. the data it gathers to send to specified users and how admins can view the information. As well as develop an API that can show that this application can be flexible and used in diverse ways for different purposes.

# **Literature Review**

## **NFC Technology**

This section will provide an overview of the historical implementation and impact Near Field Communication technology has provided over the course of the past decades and how it now used in our daily lives.

### **History and Evolution**

NFC or Near Field Communication is the main force behind your everyday payment facilitator such as Apple Pay, Google Pay and contactless card payments. Its main functionality is providing a wireless technology between devices within proximity of a range between 2 to 5 centimetres.

This technology can be dated as far back as the 1940s as an implementation for World War II using RFID (Radio Frequency Identification) as a means for radar technology marking the beginning of the new wave of communication which was especially significant for the Manhattan Project. This technology was used to detect and locate objects based on the reflection of radio waves, which was quickly adopted by the military. From this finding, RFID was born (Landt, 2005). Harry Stockman’s research on RFID technology has been noted as the first or earliest research published in 1948 in a paper “Communication by Means of Reflected Power”. The paper did state that there was tremendous research completed before him regarding the finding of this information stating, “Evidently considerable research and development work has to be done before the remaining basic problems in reflected-power communication are solved, and before the field of useful applications is explored.” (Stockman, 1948). As there was a lack of hardware during the time of the paper published by Stockman, it took over 30 years for this development to become a physical finding. Transistors, integrated circuits, microprocessors, communication networks and other developments were needed to find the solution to RFID and NFC (Landt, 2005).

The first real world application of RFID was developed between the 1960s and 1970s when companies Sensormatic, Checkpoint and Knogo began their development of EAS or Electronic Article Surveillance which would now be the grandfather of security tags for retail merchandise. They used microwaves or inductive technology which was cheap to create and the first commercially available RFID product (Landt, 2005). The 1970s there was increase of research towards RFID, Los Alamos Scientific Laboratory notably the lab that had managed most of the development of the first atomic bomb, Northwestern University and the Microwave Institute Foundation in Sweden all taking the reins in this development. Los Alamos delivered an important paper in 1975 on their researched named “Short-Range Radiotelemetry for Electronic Identification using Modulated Backscatter.”. This research demonstrated tags that could be operated from a 10-meter range while larger companies developed RFID technology such as Raytheon’s Raytag in 1973 and the RCA’s electronic development system in 1975 (Landt, 2005).

In the meantime, there was some news of an electronic toll collection in the works by the Port Authority of New York and New Jersey in partnership with General Electric, Westinghouse, Phillips and Glenayre. In the 1980s European countries had implemented RFID through their toll roads, animal tracking while the United Stated focused on transportation, personnel access, and animal tracking. The first commercial application of an RFID toll system after years of testing began in Norway in 1987 and the Dallas North Turnpike in 1989, the Port Authority of New York and New Jersey implemented RFID operations for buses passing through the Lincoln Tunnel. The 1990s signified the widespread use of RFID technology through electronic tolls with over 3 million RFID tags implemented on North American rail cars. Similar technology was being implemented in Europe, South America, Asia, and Australia (Landt, 2005). South Korea’s Seoul Bus Transport Association implemented the first contactless transport card called the UPass (Global Payments Integrated, 2014).

Retreating a few years, in the 1980s Sony and Phillips saw potential in the ability to use the technology to transfer data between devices. Both companies had used their development to transfer music files between devices during that time. (profylecard, 2021). Further on Sony and Phillips join to co-invent NFC which is then approved by the ISO in 2003 as a new standard. In the same year Nordea and Luottokunta released a prototype device that used NFC to make payments. The following year, Nokia modified the Nokia 6210 to be able to read RFID tags. Partnered with JC Decaux, the modified 6210 could read tags and send an SMS once the tag had been read. In the same year Nokia, Phillips and Sony curated the NFC Forum to aid in the development of NFC technology, they hoped it would create new standards in the mobile industry which was achieved now with over 160 members in the forum (Paragon ID, 2017).

The evolution of these developments and studies, companies began getting onboard the new burst of information about RFID and NFC. Nokia kept the mark strong by releasing the first NFC enabled phone in the same year as the first creation of the NFC tag in 2006. 3 years later peer-to-peer communications started to pop up allowing more wireless connections and allowing more functionality in mobile phones. Evolution continued as NFC enabled smartphone the Samsung Nexus S through Android was being sold to consumers (Paragon ID, 2017).

Contactless payments as we know today through our phones and physical cards began in 2008 when Visa American Express and MasterCard offered contactless payments for credit cards (Global Payments Integrated, 2014). Debit cards had followed suit the following year (Statista, 2020). Mobile payments are then launched in 2011 with Google Wallet and Android Pay aiming to decrease a physical wallet and complete all payments through your smartphone. Apple followed suit in 2014 with Apple Pay and the next year wearable devices were added into the mobile payment ecosystem, allowing devices such as the smartwatch to complete the same functionality as the smartphone (Global Payments Integrated, 2014).

In our everyday practical lives, we now see NFC integration not only on our mobile devices but even in our homes. Starting with security systems and how they may provide tags to enable and disable the alarm, entering homes and hotel rooms have had a long history of NFC enabled devices collaborating with each other.

### **Mechanisms and Operations of NFC and RFID**

NFC or Near Field Communication is an extension of RFID technology or Radio Frequency Identification. NFC enabled communication between devices in proximity i.e., short-range radio technology. This technology was standardised by the International Organization for Standardization in 2003 (Prohel, 2013).

The short-range, high frequency, low bandwidth and wireless communication occurs at 13.56 MHz high frequency with data transmission rates of up to 424 kbit/s with up to 20 centimetres, but 4 centimetres is practical which prevents eaves dropping on other NFC-enabled devices. There is usually a maximum of two devices used to complete the transfer of data using NFC. The first device is called the initiator which starts the communication, and the second device is called the target which responds to the requests from the initiator. (Rahul et al., 2015). With the use of electromagnetic induction loop antennas between two devices the process of request and response communication is processed. The standard frequency provided by ISO/IEC 180000-3 for NFC systems for 13.56 MHz can be used without a license (IONOS editorial team, 2023).

NFC provides 2 modes of communication: passive and active mode. Passive mode allows active NFC readers to read passive transponders. The antenna of the active device generated a high frequency magnetic field. If the transponder enters the range of the high frequency field, it is supplied with power but does not generate a feedback signal and is transmitted using a modulating inquiry signal. This can be used for contactless payments through debit/credit cards or mobile devices. Active mode occurs when both the initiator and target device can transmit NFC signals. These devices have their own energy source and generate their own high frequency during communication activating and deactivating fields when necessary or when waiting for a response (IONOS editorial team, 2023).

NFC can be used independently but the rest of the communication can be completed using Wi-Fi or Bluetooth e.g., wireless card readers. The target device may be able to manage multiple carriers such as NFC, Wi-Fi, and Bluetooth. The requesting device can choose which ever carrier is best suitable for both devices when it received the response message. This method of communication is preferred over Bluetooth as it consumes less power and does not require any physical paring compared to Bluetooth (Rahul et al., 2015).

NFC chips can support multiple Radio Frequency (RF) protocols and features such as Read/Write, Peer-to-Peer and Card Emulation as outlined in Figure 1.

1. Reader Mode: One device can function as a reader or writer while the other serves as a tag or card. The tag holds a small amount of data such as a unique identifier or other information. Tap & Pair represents the initiation of a connection between two devices such as when an NFC tag is attempting to connect to a Bluetooth printer. Tap & Exchange represents the exchange of data between the reader and NFC tag such as reading data from a smart poster using a smartphone.
2. Card Emulation Mode: The device emulates an NFC card or tag such as contactless payments where smart devices such as mobile phones or watches emulate debit/credit cards. Tap & Pay represents contactless payments from an NFC enabled device to a POS terminal.
3. Peer-to-Peer Mode: Two NFC enabled devices can communicate and exchange data with each other. Tap & Exchange represents the exchange of data like Reader Mode, but this occurs between two devices like smartphones to exchange contacts, photos, etc...
4. [A few cards with text and symbols

   Description automatically generated with medium confidence](https://www.st.com/content/st_com/en/support/learning/essentials-and-insights/connectivity/nfc/nfc-chips.html)Charging Mode: This is not a traditional NFC operation but represents NFC technology being used as a wireless charging mechanism. Approach & Charge represents an NFC enabled device being brought close to a wireless charger.

Figure 1 NFC Reader Modes (STMicroelectronics, 2022)

Due to communication occurring on two devices, there needs to be a security standard already implemented to reduce eavesdropping. Point-of-Sales (POS) terminals that use NFC to initiate contactless payments that use RFID communication protocols have implemented standards such as ISO/IEC 14443 (Badra and Badra, 2016).

The ISO/IEC 14443 is the international standard for proximity cards and contactless smart cards that are used in many NFC applications. It specifies the communication and security protocols for secure data transfer (ISO, 2023). It specified the physical characteristics, radio frequency power and signal interfaces along with initialisation and anti-collision protocols. This standard is implemented by Card Emulation Mode when NFC enabled devices emulate ISO/IEC 14443 cards in this mode such as digital wallets making it compatible with POS terminals. ISO/IEC 14443 is split into two types: Type A and Type B. NFC is compatible with both types allowing for broader range of interactions and supporting a large array of applications, particularly contactless payments (ISO, 2023).

Initially in 2001 when the ISO/IEC 14443 standard was released, it was structured into four distinct parts: physical characteristics, radio frequency power and signal interface, initialisation, and anti-collision and transmission protocol. These aimed to ensure consistent behaviour and interoperability(ability to operate and share data with another application) among proximity cards and readers. Over the years the specifications where refined or expanded on. In the 2010s, there was an enhanced security update applied to counter the rise of digital attacks. The standard received updates on encryption, authentication, and other security measures to ensure data being passed across contactless communication protocols were secure and could not be tampered with (ISO, 2023).

The ISO/IEC 18092 defines the communication modes for NFC Interface and Protocol (NFCIP-1) using conductive coupled devices operating at the specified frequency of 13.56 MHz. It defines the Active and Passive communication modes for NFCIP-1. This standard is implemented through Peer-to-Peer modes, allowing two-way communication between two NFC enabled devices. It encompasses the various layers of NFC communication including initialisation, data collision and data exchange (ISO/IEC, 2013). This standard ensures the compatibility with other contactless technologies such as FeliCa which is a contactless smart card system developed by Sony in Japan. It is widely used for public transportation including Suica and Pasmo cards. Universities and companies in Japan use FeliCa as their main system for NFC access control system and ID cards. FeliCa is also implemented into mobile phones in Japan to be used as digital smart wallets or public transport passes (Sony, 2023).

In 2004, the ISO/IEC 18092 standard “Near Field Communication Interface and Protocol(NFCIP-1) was released defining the communication modes for NFC enabled devices. From 2004, the adoption of this standard became widespread particularly in niche applications and electronics. In the 2010s, this standard was adopted on smartphones, tablets, smartwatches, and wearables on systems like Android and iOS. Security and performance improvements were implemented for payment transactions (ISO/IEC, 2013).

Apart from the security that are included in the standards provided by ISO/IEC, many developer/companies do lay their own security protocols on top of the pre-existing ones.

Tokenization is a method of replacing or masking sensitive data such as card information with randomly generated, unique card details used on digital wallets such as Apple Pay or Google Pay for e-commerce or in-store transactions on POS terminals. By doing this, it mimics the physical card and replacing it with the token. If said token is intercepted it cannot be used apart from the original transaction. During the token life cycle the Token Service Provider must control it with 4 different statuses:

1. Requested: Initial state of token on digital wallet
2. Active: The token is ready or already has been used
3. Suspended: The token is suspended by the issuer.
4. Ended: The token has permanently been disabled (HST Software Solutions, 2021).

Encryption is another method of security that it used to enhance the privacy of sensitive data being passed along NFC channels. To avoid eavesdropping and interceptor’s encryption can be used to encrypt the data during transmission and decrypted when received providing the receiver has the decryption key. Symmetric and asymmetric encryption can also be used where with symmetric both the sender and receiver use the same key for encryption and decryption, this is faster but a higher risk of exposure. Asymmetric using a public key for encryption and a private key for decryption which is mostly used in NFC transactions as the public key can be openly shared (Hendry, 2014).

SE or Secure Elements is a tamper-proof platform usually in the form of a chip, is capable of security hosting applications on their confidential and cryptographic data such as NFC enabled mobile payments on smartphones. This chip does not have permissions enabled for installation; all software comes preinstalled. Only trust applications such as digital wallets or POS terminals have read/write access to the chip. SE provides detection of hacking and modification attempts, creation of Root of Trust platform for encryption systems, storing private encryption keys and bank details, random number generator that is are secured by cryptography, generation of private and public keys for asymmetric encryption. This is all provided at a hardware level (Kaspersky, 2020).

Introduced around 2014, HCE or Host Card Emulation allows for NFC applications on smart devices without relying on access to the SE making mobile payments more versatile. HCE aimed to provide two solutions, card emulated by the cloud system or emulated by the mobile application. In comparison to SE, SE provided several functions as mentioned above like a cryptography check to validate the card information and verification where as HCE’s implementation would guarantee that any NFC data received by the processing app was delivered from the controller and would remove any other entities. By doing this, this would mean that the mobile device would have to be connected to a network to complete the authorisation of the transaction (Sims, 2014).

While all the above mentioned, intricate security onboarding protocols can be implemented onto NFC chips/devices, it is important to note that there is still the possibility of signal copying or cloning. Since NFC operated at a 13.56 MHz and allows two-way communication between devices and allows devices to read tags or act as one regarding card emulation it is possible for the data to be intercepted using specialised equipment or mobile applications designed for this purpose. Typically, the attacker would need to be close to the victim, centimetres away, they can then intercept the data. Once its intercepted, the data can be analysed using sophisticated tools as it may not be comprehendible immediately. With the captured and decrypted data, the attackers can emulate the same tag or reuse the signal emitted by the device to conduct unauthorised activities such as making fraudulent payments, identity, and data theft (Francis et al., 2012).

Devices such as Proxmark devices or the ChameleonMini can be used for RFID/NFC signal copying. The Proxmark3 can read and emulate a multitude of NFC and RFID tags as ca the ChameleonMini. These devices or similar are readily available on websites like Amazon starting at 40 GBP (Waldman, 2021). Projects like the ChameleonMini had started out as open source projects on GitHub in 2013 but now have become wide scale cloning tools coming in the form of credit cards. These devices could clone the UID(Unique Identifiers) and stores the data and can also attack RFID readers to sniff for keys and decrypt them but to be able to do this the device has to be near be able to complete the task (Person, 2022).

### **Applications**

In an era where technological advancements are never ending, NFC stands out as a pivotal enabler of seamless digital interactions. NFC has become the source of most of your daily interactions to hardware, making systems simple and efficient. From going from cash to cashless, from replacing physical public transport tickets to creating interactive access control systems.

The rise of digital wallets and mobile payments solutions have been fuelled by NFC technology. NFC-enabled smartphones allow to store their debit/credit card information on their devices by masking the card details but routing back to the account of where the information lies. Using POS systems like Square or SumUp, transactions can be completed without the need of a physical card. Due to the masking of the card details it adds a layer of security by using encrypted tokens. In 2022, 49% of global e-commerce transactions were completed by digital wallets while 20% were completed by 20% by credit card and 12% by debit card (Statista, 2023). Projected to surpass 1.30 billion proximity mobile payments in 2023, in 2019 0.95 proximity transactions were completed (Statista, 2019).

The public transport experience globally is fuelled in NFC technology. Dating as far back as the 1980’s with NFC tags for tolls and rail cars to collect payments for crossing borders and South Korea implementing the first public transport system using NFC in 1996. Transport for London implemented the Oyster card in 2003, allowing contactless public transport payments using an RFID system with a built-in inductor that reads the frequency being projected by the reader (Birbeck, University of London, n.d.). In 2012, TFL had implemented a contactless payment method for the underground and rail network allowing debit/credit cards to be used alongside the Oyster card and cash which has now the biggest payment form on TFL. Finland had implemented the same system on buses in 2021 in its city of Turku. By using their pre-existing ticketing system, they changed the model to match the requirements of payment service providers (Moore, 2023).

Traditional access control has relied on magnetic stripe cards or physical keys. With NFC, access control becomes more digital and more secure. Whether for offices, secures areas in airports, or even gym memberships, NFC devices can grant access based on the embedded data or permissions. These kinds of systems can provide logs of entry and exit times to enhance security. As predicted by market research, the access control market is expected to grow from $175.57 million in 2021 to $787.01 million by 2027 (The Insight Partners, 2020). Companies like Kisi Inc., Salto Systems, Proxy Inc., BlueID, Remotelock, YPTOKEY, Nok, Inc., Openpath, Inc., Unikey Technologies Inc, and Brivio Systems LLC are leading players in the marker that are implementing NFC in NFC access control systems (Research Union, 2023).

## **NFC Enabled Student Cards**

### **Real World Applications**

To get a complete understanding of this Final Year Project and compare it to the proposed system, we will delve into previous proposed systems or already implemented systems to see how versatile this technology can be.

As part of the proposed system for this final year project, the system should be able to manage door access control, class attendance tracking, paid parking all managed by an NFC integrated student card. The goal of this implementation is to create a unified, efficient experience for all students and faculty.

As discovered by Duke University, implementing NFC technology with student ID cards has been able to streamline campus life for all its students and staff. As well as using a physical student card, the university has opted for implementing this through the Apple Wallet with iPhone and Apple Watch to “improve the student experience.” They have integrated this as this now allows students to pay for lunch at dining halls through their student cards on their digital wallets which makes the card function as the home of the entire system. They have also implemented door access control through these digital wallets where only students specified to enter a chemistry lab has permissions to enter compared to a business student who would not need to enter that room thus the card would decline them from entering. Florida International University have implemented NFC tags and codes at their library for instant access to digital reading (Zimmerman, 2018).

Companies like Transact offer a multitude of customisable features, starting from having digital NFC enabled student cards that store encrypted student data per card which allows the student to make purchases on campuses which is integrated with pre-existing POS systems. They also provide door access control with can be integrated with security on campus which allows for an admin view where they can set and change permissions as they see fit (Transact, 2023).

Purdue University in West Lafayette Indiana, implemented a similar system like Duke University, allowing students to use their digital wallets as placeholders for the student cards. This digital NFC cards allow students and faculty to have assigned residential and campus access, pay for meal plans and laundry through the card, function as an attendance tracker for class and door access tracker for administrator logs. This is all with the use of the Transact system built around the pre-existing system at Purdue University to offer a smooth and seamless experience for everyone on campus (Purdue News Service, 2023).

The University of Hong Kong have implemented an attendance-taking and event enrolment using their in-house developed AIESEC HKU Tap-in app. Each student card can be scanned by the phone, it collects a string of characters is collected by the phone which contains each student which is now displayed on the app. Before it would take 30 seconds to register each person but now it takes two seconds to register students (Lung, 2015).

Previously implemented functionality by scholars who have designed a similar system start by creating a schema to manage all the data to complete the functionality. In terms of creating a class attendance system, tables such as ‘Course,’ ‘Student,’ ‘Lecturer,’ ‘CourseAttendance’ would exist to manager this functionality. To implement this, a web page as front end including HTML, JavaScript, CSS, MySQL, and PHP acting as back end was used. The lecturer would be able to log in and view which students were marked as present based of the NFC tags that were read. To simulate the physical hardware RiFiDi was used to mock the NFC tags. Two antennas would be used to cover the whole classroom and each student would be tagged (Pireva, Krenare R, Siqeca and Berisha, 2013).

Another attendance tracker implemented by students from the University of Bridgeport. Using an Arduino Mega to process tan NFC Module to read a NodeMCU to send IoT messages over Wi-Fi all with an NFC enabled card with the ISO/IEC 14443 standard, if the card is scanner by the NFC Module, the reader will send the ID number and desk to a cloud database to track attendance as well as the student information, course, and sign-in time. Each student ID card has a unique information based on 10-15 students in each class. When the NFC card is tapped on a reader during scheduled class time, the reader sends a message confirming it has received the data before sending the data to an Azure IOT Hub via MQTT message (Dixon and Abdel-shakour Abuzneid, 2020).

Implementing a smart access control for paid parking was implemented by students of the Department of Electrical Engineering, Hasanuddin University. Using a registered NFC student or staff card, it can open or close a parking gate automatically. The proposed parking gate by the students included a PN532 NFC RFID module, a microcontroller, NFC tag, PING sensor and ethernet shield. All information passed through this smart gate would be stores on a server including user data, access history entry and exist using a localhost. The process of interacting with the gate begins with the user tapping their tag or card on the reader of the gate, if the tag/card is registered and identified as having access to the gate, it is opened and closed automatically after the PING sensor has been detected. During this process, the database server is updated by ordering the users in chronological order of accessing the gate. This paper looks at this smart gate as a security measure on campus by only allowing certain users to have access to the gate (Mansur and Hasanuddin, Zulfajri B, 2018) .

Students of Electronic and Mechanical Engineering College, Fujian Polytechnic Normal University and Maynooth International Engineering College, Fuzhou University implemented a door access control system using OpenMV(microcontroller), Arduino an RC522(RFID reader), Esp9266 Wi-Fi module and Ali Cloud of Internet of Things. By developing a WeChat program, these students were able to read information between the users and electromagnetic lock when the door was opened and ensuing a non-contact unlock. The OpenMV was able to read and store information based on facial recognition by the connection module and power supply. The RC522 module was able to store and read information from an NFC enabled card through the connection module and power supply. The Esp8266 module was able to connect to the WeChat program through the Alibaba Cloud IOT so they it could interconnect the entire system to the campus network and so it could receive unlock and lock instructions. A magnetic lock is used to simulate the lock of the system. In this system one of two electrical connections is receiving a specified level while the other is grounded. The Arduino UNO board is connected to the computer and the lock is connected to the board; the Arduino has been coded to keep the lock closed. For facial recognition, the OpenMV is connected to the computer is coded to be able to write and modify face storage and face recognition code. Face recognition and radio frequency is set up as a level output and connected to the complete system by writing and modifying the code on the Arduino. The Wi-Fi module is then connected to the Arduino and connected to the Alibaba Cloud IOT and there is code written to interconnect the system to the campus network infrastructure (Zhou et al., 2022).

Students of State Polytechnic of Malang, Malang, Indonesia built an NFC based self-service canteen in a local boarding school. The reason for implanting such system was that students were exceeding the amount of money they could take daily from their residential financial managers which brought up a solution to tackle overspending and long lines of students wanting to withdraw money. Students also kept money in their dormitories which also raised a security concern. By creating a system that uses e-money transactions would eliminate all the issues brought up above. This application used NFC technology combined with a smart card and smartphone to make canteen purchases. The system included a web server and standalone transaction application which would manage the payment streams. The NFC/smart card would replace cash and hold a balance of money for each student. Each card would act a payment method for students and would complete transactions in the canteen (Reva Rikat Asih et al., 2022).

To ensure a safe and reliable NFC enabled student card, one of the best ways would be to use a Multi-Factor Authenticator. MFA has three main factor a PIN/password, device used and behavioural/physical biometrics. EU law states at least two of these must be used to use MFA. The students at Brno University of Technology, Brno, Czech Republic implemented a system using an NFC enabled android smartphone , SAM module in the form of a smart card and a Raspberry Pi 3. The system allows the terminal to connect to an electric door lock to unlock based on a successful authentication from the user. The user triggers the system by bringing the card to the reader, an Android application is triggers asking the user to authenticate themselves by either using a PIN or fingerprint which are encrypted in the backend. Once authenticated, there is a log of successful authentication with a timestamp which is also encrypted. They door then unlocks depending on if the authenticated user has permissions to use this door (Cvrček and Dzurenda, 2022).

## **API Integration**

### **Definition**

As defined by MuleSoft, an API is an Application Programming Interface which is a software intermediary that allows two applications to talk to each other (Frye and MuleSoft, 2023). They provide a simple interface to interact with a complex system like sending a mobile payment or changing the home temperature on your phone without needed to see or know the intricate details of its implementation. APIs can include functions and procedures. These then bridged the gap between the application on your device and other applications. They assume over 80% of internet traffic globally (Kong Inc, 2023).

Web services that allow you to log in with your Google or Facebook account to authenticate you as a user, use the API provided by the host to authenticate you. They operate by a client/server architecture. The application that sends the request acts as the client and the application receiving the response acts a service. In terms of using Google to authenticate you as a user for mobile gaming application, Google would act the server and the gaming app would function as the client (Amazon Web Services, Inc., 2023).

According to AWS there are four types of APIs:

1. SOAP APIs – Simple Object Access Protocol. Client and Server communicate through XML.
2. RPC APIs – Remote Procedure Calls. Client completes function on the server and the server sends the response to the client.
3. WebSocket APIs – Modern web API that uses JSON to pass data. Supports two-way communication between client and server.
4. REST APIs – Representational State Transfer defines set of functions like GET, PUT, DELETE etc. Stateless i.e., do not save client data (Amazon Web Services, Inc., 2023).

APIs have many benefits as they can be flexible depending on the application they are being used on. They provide automation as they can be managed by computers as there is less human effort required and workflows receive faster and more productive updates and can reach end users entirely. APIs enforce developers to reuse code as they do not need to continuously need to restart from scratch every time, they develop a new application as the API would specify how to assemble the software components (AirFocus, 2020).

### **Challenges**

APIs do come many with challenges. Software integration with APIs do require high level adoption of application languages and maintenance to meet with business requirements are a never ending task. As much as they do help add better functionality and improve services, they do come with their own challenges.

As data breaches become one of the most common security risks in 21-century technology, API technology remains vulnerable. As a lot of these APIs are always connected to a web application it has more opportunities to be susceptible to be targeted. Since these APIs can be complex the level of security and level of communication needs to be detailed from the beginning to avoid any attacks or misuse. A team or task force can be put together to manage the API. They would be responsible for managing security (Nguyen, 2023).

Many developers may be reluctant to implement new APIs as they may not be fully knowledgeable with new systems. Implementing new API would also require training which can become costly and not time effective. Professional or skilled developers may also be necessary to implement new solutions. Maintenance and upgrades should also be taken into consideration when planning to use a new API. It is important to note that APIs are not a one size fits all. Not all systems connect straight way meaning more time is spent planning and developing a system. Having a skilled developer experienced in the API can reduce the cost and time spent implementing a new API (Curoe, 2021).

Lack of documentation can also provide more issues for developers. Understanding the functionalities, data, and methods of accessing the API may not be available for the developers. These further delays implementation or updates of systems. Contacting the API developers may become last resort if errors or confusion arise during the integration process. There may be data incompatibility with a pre-existing and a new API. Data mapping tools are available to combat this and ensure that you can map the different formats to ensure data between APIs. APIs are usually version controlled which means they may operate differently or use different data structures. Any application using a version controlled API should also be version controlled to ensure that it is keeping up to date with the API. Regularly assessing the API integration should be a priority of a developer to ensure it is functioning correctly with the system as well as monitoring the API to ensure it is consistently online and its behaviour is normal (Jagaad Software House, 2023).

Scalability is also important when developing APIs. To avoid overloading on a service rate limiting is implemented but it can impact the functionality of the API during the traffic spikes. Having backwards compatibility is also important for people that use APIs to ensure they can roll back if their system does not work well with a newly updated API (Marcus, 2023).

## **Future Trends**

The NFC market was evaluated to have reached $21.1 Billion in 2022 (PR Newswire, 2023). NFC sales are set to reach $44.7 Billion by 2032 based on automotive and contactless payments (Future Market Insights, 2022). Companies such as Gemalto, Broadcom, Infineon, Inside Secure, NXP and MediaTek are dominating the NFC market. Gemalto leading the secure connectivity solutions with NFC offering services starting from SIM cards to mobile payment solutions (Reach Researcher, 2023).

Beamian, a hybrid event platform, project that NFC will take over event management by using it for ticketing and access control, reducing the need for paper tickets, speeding up the check in process by using wearable devices, mobile phones. NFC tags can be used around the venue for event information, maps, schedules. They can be used to capture attendees’ data for health purposes or safety (Carrilho, 2022).

NFC enabled devices such as thermometers, blood pressure monitors or glucose meters are predicted to be used for health monitoring, diagnosis, treatment, and prevention. NFC enabled test kits, patches and implants are now on the rise for treating conditions and illnesses (Linkedin.com, 2023).

In terms of APIs, as AI and ML are now becoming the most studied technologies, it is expected that APIs built to manage and manage these systems will now start trending as developers will look for systems to build their applications on top of to reduce development time and code manipulation. Since APIs can act a Software-as-a-Service model, developers are now treating it as a product rather than a method of communication between systems (Jaffery, 2022).

It has been noted that serverless architecture for cloud computing is becoming more popular. APIs will be more focused on as there is no need to have any server maintenance and further reduces costs. API management tools are now becoming more popular to help developer automate tasks or run tests. IOT devices are now becoming more complex with links between cars and smartphones or a smart home and a smartphone, developers are starting to opt for APIs that can act a foundation for their programs (Nguyen, 2022).

## **Conclusion**

This exploration of NFC technology, particularly within the context of NFC-enabled student cards and the implications of API integration, highlights overflowing amount of technology with potential and challenges. Starting with a foundational overview, we have explored NFC's progressive journey from its creation to its current state, acknowledging the pivotal developments that have contributed to its evolution.

The mechanisms and operations of NFC have been demonstrated, highlighting the efficiency of which this technology operates. NFC's versatility was further depicted through a diverse array of applications, from simple data transfer to complex, secure transactions.

Focusing on NFC-enabled student cards, we have examined the multifaceted functionalities they offer, ranging from access control to cashless payment systems, enhancing the educational ecosystem's efficiency and security. The implementation section shed light on the practical aspects, including the technical and logistical considerations vital for successful deployment.

Going into API integration, we discussed the role APIs play in today’s software systems and beyond. While APIs offer a framework for innovation and flexibility, the challenges they present—such as security risks and the need for rigorous management.

Future trends anticipate significant technological advancements, suggesting a trajectory of growth and transformation for NFC and API ecosystems. The potential applications reveal a world of possibilities, from market growth to implemented health systems...

In closing, NFC technology and API integration stands as a testament to the adaptability and potential of digital systems to reshape our interactions and infrastructures. As we continue to embrace these technologies, the importance of strategic implementation, continuous innovation, and consideration of privacy and security becomes increasingly evident. The future trends indicate not only technological progress but also a need for a conscientious approach towards embracing these advancements, ensuring they serve to better our institutions and society.

As this field continues to evolve, it will be imperative to monitor the developments and address the challenges discussed herein. With careful planning and foresight, the integration of NFC and APIs will continue to revolutionise the student experience and beyond.

# **System Analysis and Design**

## **Introduction**

The NFC Enabled Student Card System with API Integration is an innovative project designed to use Near Field Communication technology to streamline and secure campus operations and services. This system aims to not only enhance campus security and efficiency but also to foster a more connected, digitally enabled educational environment. Objectives include seamless access control, accurate attendance tracking, and facilitating cashless transactions. The scope of this system will extend to all areas of campus life impacted by these functionalities, ensuring a comprehensive integration of NFC technology. This section aims to cover the physical design, software design, coding, and software architecture. We will go over the required hardware and software, along with any APIs that can help this project go more smoothly.

## **Analysis**

When researching previous implementations of an NFC integrated systems for this Final Year Project, the developer examined several different projects to gain a sense of what previous projects looked like and what kind of hardware and software was used to accomplish even the most basic functions, such as reading or writing from an NFC card. According to my research, there are numerous factors to consider when attempting to construct a system like this, such as microcontrollers, NFC chips, software languages, APIs, and so on. User requirements are important to the design of our system. This project is committed to providing students, teachers, and administrative personnel with an intuitive experience. Reliable user authentication, efficient access management, and quick data retrieval are all functional needs. Non-functional criteria including system performance, security, stability, scalability, and usability are just as important. These will ensure that our system is not only functional, but also sturdy and easy to use. Taking all these aspects into account, this section will present a breakdown of the hardware required, components, software, and technologies that can assist in producing a functioning project in line with the proposed system.

1. The total cost of components, hardware and software licenses needed can play a part in deciding how to begin a project. Depending on whether you choose cheaper components or microcontrollers they may not be able to oversee what you intend them to execute. Same with software licences or even cloud storage may play a part. During the decision of the project, it is important to consider whether you would like to store the data on a cloud server or store it locally.
2. The user requirements are important as well. As there is one developer managing the project, it is important that they produce their own user requirements, and document and attempt ones that encapsulate the focus of the project.
3. Functional requirements including user authentication, access control, data retrieval must be detailed. Specified user requirements must match the functional requirements and they must work as the system is intended and documented to do.
4. Identifying non-functional requirements like performance, security, reliability, scalability, and usability should be considered if this is something that is intended to be created for an API. APIs must be able to manage the data load or usage specified or that does not offer a working product. As this is not a production level project, the non-functional requirements may be a bit relaxed.
5. The data model should be designed to represent all the identified models in the system. This along with the database design and data storage requirements should be analysed.
6. Since there will be sensitive data managed in the system, the system will have to have adequate security and encryption settings in place. As well as having a tamper proof system for the access control to prevent unauthorised access.
7. UI design for the administrative view must be user-friendly. Considering it will manage student cards, door authorisation, view logs etc… it must be laid out clear and understandable for administration to view.
8. Choosing hardware depends on the requirements needed to complete the project. Certain microcontrollers may only be able to manage a specific workload while another may be able to manage everything which also comes down to price. As well as any external modules chosen, it is important to consider that some modules will not be able to support all types of NFC cards/tags.

## **Hardware Requirements**

The chosen hardware components, including RFID Reader Writer and the Raspberry Pi 4 Model B, are selected for their reliability and compatibility with our system's requirements. These components are integral to ensuring efficient data processing and user interaction with the NFC system.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Hardware | Manufacturer | Quantity | Function | Reason for choosing | Connection | Cost |
| NFC RFID Reader Writer ACR122U ISO 14443A / B | Advanced Card Systems | 1 | A module managing the read and write to the NFC cards, will be able to talk to microcontroller | You can read and write from the module | USB A | £30.98 |
| Electronic Kit | Rk Education | 1 | Includes LED nodes, breadboards, capacitors, transistors etc.. | This will be used to connect the microcontroller to any external modules we have | N/A | £10.23 |
| NFC White Tags | cobee | 20 | NFC compatible cards | Can read and write from these cards | NFC/RFID | £7.16 |
| Raspberry Pi 4 Model B | Raspberry Pi | 1 | The microcontroller controlling the system | The microcontroller controlling the system | GPIO Pins | £66.10 |
| 5V 1 Channel Relay Board Module | BuxiuGK | 6 | Can function as a switch | Simple mechanism that does not require a lot of manipulation | GPIO Pins | £9.21 |
| DC12V Solenoid Lock | Tangxi | 1 | Type of lock used for electronic lock modules | It is a simple lock that can be connected to a breadboard and written through to lock and unlock. | GPIO Pins | £6.81 |
| LCD 1602 Module | FREENOVE | 1 | Display messages from microcontrollers | It can display any messaged when we execute functions through the microcontroller | GPIO Pins | £8.14 |
| NFC NXP RFID Module V3 Kit | Hailege | 2 | Supports NFC/RFID reading and writing | You can read and write from the module | GPIO Pins | £6.38 |
| Mfrc522 C522 RF IC Card Reader Sensor Module | AZDelivery | 3 | Supports NFC/RFID reading and writing | You can read and write from the module | GPIO Pins | £10.23 |
| SUM |  |  |  |  |  | **£155.01** |

Table 1 Hardware Requirements

## **Software Requirements**

The software stack will include robust database management systems and programming languages such as Python or Java. These tools are chosen for their flexibility and wide support in handling complex data structures and user interfaces essential for our NFC system.

|  |  |  |
| --- | --- | --- |
| Software Component | Purpose | Justification |
| Database Management System (e.g., MySQL, Amazon DynamoDB) | To store and manage user data, access logs, and transaction records | Provides robust, secure, and scalable data management capabilities |
| Programming Language (e.g., Python, Java) | For developing an NFC Enabled Student Card System the system backend and processing NFC data with API integration | Versatile and widely supported, suitable for backend development and NFC integration |
| Web Framework (e.g. Flask for Python) | To build the administrative web interface for system management | Facilitates quick development of secure and scalable web applications |
| API Development Tools (e.g., Swagger, Postman) | For creating and testing APIs used in the system | Streamlines the development and testing of RESTful APIs for system integration |
| Version Control System (e.g., Git) | To manage codebase changes and collaborate among development team members | Essential for tracking revisions and collaborating in a team environment |

Table 2 Software Requirements

## **System Analysis**

### **Requirements Analysis**

For this project, the NFC interface serve as the fundamental driving power of the entire concept, using it to demonstrate key principles of real-time data processing and user interaction. While real-time processing remains important, to ensure that the system is secure there can be an implemented slight delay to ensure that the system does recognise the scanned NFC tags as valid. The system could demonstrate a variety of different applications based on locations such as basic access control in classrooms to simple transaction processing for parking. The system might be able to manage basic authentication and simple transaction processing such as a balance reduction system for paid parking which can be implemented without using a complex financial solution in addition to the system.

Data management can focus on fundamentals aspects of storing and retrieving data efficiently without needing an elevated level of database management. Using a database system like MySQL or Amazon DynamoDB for data storage can alleviate any complexities. The database can demonstrate storing, retrieving data and logs.

As this is concept idea, the project should be able to demonstrate potential to show that it is scalable and dependable. The system architecture should be designed in a way that shows it can be scaled for example how we can integrate other NFC readers into the system in the future. The system should be dependable under normal usage conditions, but it should have redundancies or fail mechanisms implemented or documented. To ensure of a flexible system, the system should be coded using modular practices and follow all coding standards.

### **User Analysis**

The user groups of this system are designed for two: students and staff. These users have distinct needs and interactions with the system and understanding the different is crucial for the system design.

Many of the users if it were expanded to a production level software, would be students. Their interactions with the system are expected to be frequent but short such as using their student card for access to classrooms, labs etc…, or paying for parking. The system should allow for these interactions to happen smoothly and quickly to ensure that the system fast paced as intended. The staff group includes faculty such as lecturers and administrative staff. They will manage different areas of the system in a complex manner. For administration, they would be working with the system to restrict access, monitor logs and create new cards with new functionalities. The system would not be used as frequently for all staff compared to students but would still need to be as robust and intuitive as the student side.

Interactions need to be straightforward for both groups. Tapping the NFC card should be simple and consistent on all the different modules and should provide clear indications of successful or unsuccessful scans. The system should be able to adapt as per the user group and provide a tailored experience for each. The system should be capable of handling transactions in a matter of seconds to prevent bottlenecks.

## **System Design**

### **Hardware Parts**

Choosing the parts for the system depends on what is to be achieved. In relation to NFC/RFID modules some may be read only, some may be read and writable. For this project three different types of modules have been researched.

1. PN532 NFC NXP RFID Module

The PN532, is an NFC RFID module that supports reading, writing and peer-to-peer communication. It is a small sized module that is compatible with Arduino and Raspberry Pi. It supports I2C, SPI and HSU(High Speed UART) equipped with 5V TTL for I2C and UART, and 3.3V TLL for SPI.

|  |  |
| --- | --- |
| SPECIFICATION | |
| Operating voltage | 5V/3.3V |
| Interface | I2C, SPI, HSU |
| Compatibility | Arduino, Raspberry Pi |
| Supports | MIFARE 1k, ISO/IEC 14443-4 cards |
| Dimensions | 42.7mm x 40.4mm x 4mm |

Table 3 Specification of PN532A red circuit board with a white border

Description automatically generated

Figure 2 PN532

Figure 3 PN532

1. ACR122U USB NFC Reader

The ACR122U is a contactless smart card reader/writer using 13.56 MHz RFID technology which is compliant to the ISO/IEC 18092 standard for NFC. It supports MIFARE and ISO/IEC 14443 A and B cards and all types of tags.

|  |  |
| --- | --- |
| SPECIFICATION | |
| Protocol | USB CCID |
| Power Source | USB Port |
| Interface | 12C, SPI, HSU |
| Supports | ISO/IEC 18092 NFC, ISO 14443 Type A & B, MIFARE®, FeliCa |
| Dimensions | 98.0 mm x 65.0 mm x 12.8 mm |

Table 4 Specification of ARC122U

A white electronic device with a cord

Description automatically generated

Figure 4. ARC122U

1. RC522 Module

The RC522, is an RFID module with MIFARE transponders which is based off the Philips MF522-AN. It supports reading, writing and peer-to-peer communication. It is a small sized module that is compatible with Arduino and Raspberry Pi. It supports I2C.

|  |  |
| --- | --- |
| SPECIFICATION | |
| Operating voltage | 13,56MHz |
| Interface | I2C |
| Compatibility | Arduino, Raspberry Pi |
| Standard | ISO/IEC 14443A |
| Dimensions | 40 x 60mm |

Several blue electronic components

Description automatically generated with medium confidenceTable 5 Specification of RC522

Figure 5 RC522

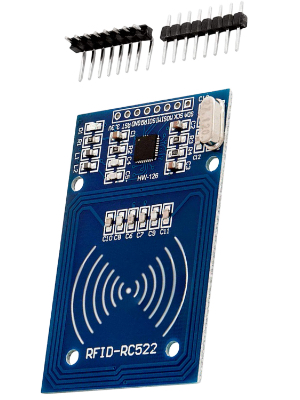


Figure 6 RC522

### **Hardware Design**

A diagram of a circuit board

Description automatically generatedIn Figure 4, we have provided a sample circuit diagram of what the hardware should look like when all the components are connected. It outlines the Raspberry Pi has as the heart of the system with the NFC Module connected to it which will read the tags and cards, the LCD display to display access messages, the relay module to act as an electronically operated switch controlled by the Raspberry Pi and would then control the power to the lock mechanism based on the validity of the NFC card. Due to the flexibility and compatibility of the RC522 chip, it has been chosen as the primary RFID module for this project. Compared to the other two modules discussed previously, it provides an extensive Python library that can be used to perform all the functionality needed for this project.

Figure 7 Sample Circuit Diagram

To setup the Raspberry Pi, an OS must be installed. To do this, we download the Raspberry Pi OS. Then we format an SD card using an SD formatting software and then drag over the downloaded OS into the SD card. We then power on the Raspberry Pi, insert the SD card, and install the software on the microcontroller.

### **Architecture Design**

This system's architecture consists of a combination of hardware and software elements, each of which is essential to its operation. The purpose of this architecture is to guarantee secure operations, easy-to-use interfaces, and effective data processing.

NFC/RFID readers are the primary contact points for the users. Since this is a model of a system, each reader will have a different function and model a classroom, canteen area etc…, which will enable users of the system to interact with their NFC enabled cards or tags. Microcontrollers such as an Arduino Uno or Raspberry Pi can function as intermediaries between NFC readers and the server. They will process the data coming from the reader and forward it to the server for further processing. The server will function as the system’s backbone. It will process the data from the microcontroller, interact with the database and manage user interfaces and API calls.

The servers will run on a reliable and secure operating system based on the demands of the system. A database management system like MySQL or Amazon DynamoDB can be used for data storage and management. Software developed on the server side will manage tasks such as user authentication, data processing and API management. All data processing between components must be encrypted to ensure data protection and security.

Based on this, in Figure 8, we have outlined a UML sequence diagram which depicts the interactions between various components of the system during specific use-case scenarios.

The scenarios and interactions identified are as follows:

* MainSystem: the user requests to unlock a door which is authenticated through an RFID reader, the user can register a new student or staff member through the reader, the system tracks attendance when a registered NFC card is used, and it is able to pull the recent attendance records.
* NFC to User Interactions: the user can top up their balance with a registered NFC card, and the user can log their parking with the balance being deducted from a registered NFC card.
* NFC Integration through Raspberry Pi: this represents hardware integration point between the reader and the system to perform all of the functionality that is required.
* Loop Sequence: the system loops to access information from the database that references the RFID\_UID of the NFC card and pulls the recent hour of attendance data.
* Register NFC Card: this flow allows for the registration of a new user.
* A screenshot of a computer screen

  Description automatically generatedUser Decisions: is where the user chooses what actions or functions, they wish to execute which is all done through the NFC system.

Figure 8 Sequence Diagram

### **Interface Design**

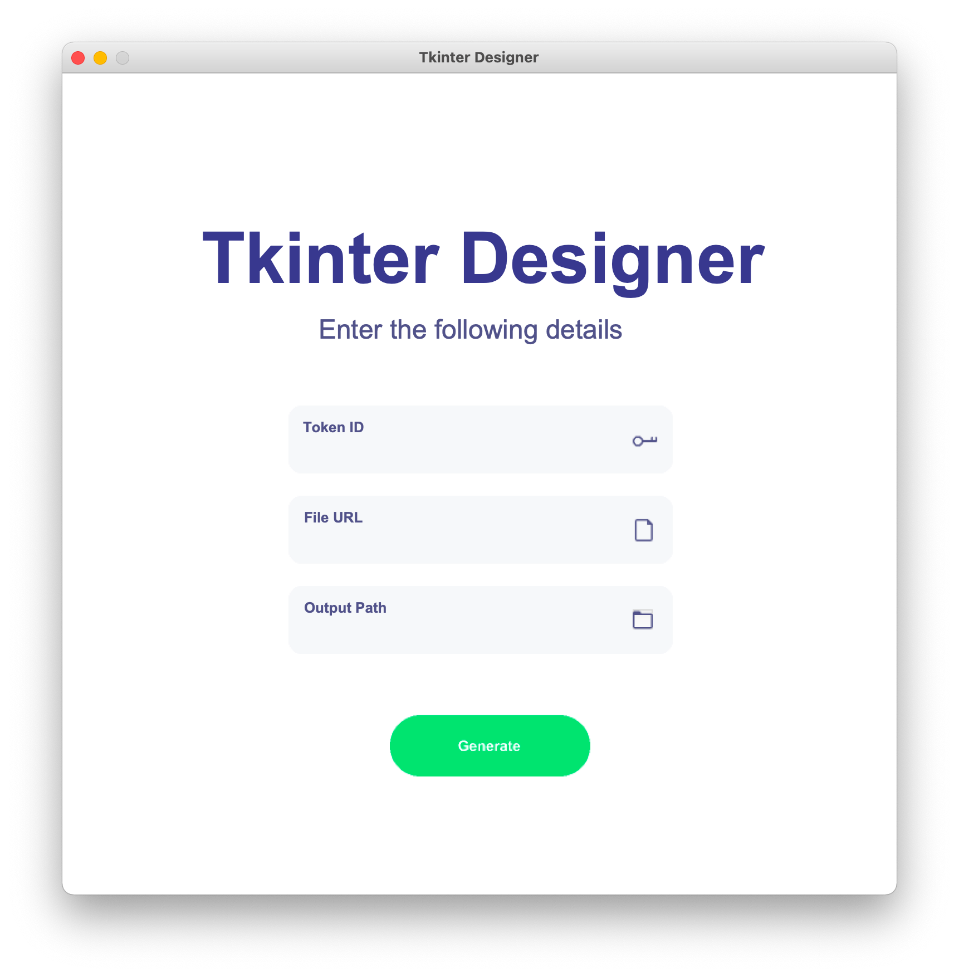
[](https://pythonrepo.com/repo/ParthJadhav-Tkinter-Designer-python-graphical-user-interface-applications#google_vignette)Ensuring that the UI design is clear, efficient, and intuitive for the administrators ensures that they can have a good user experience, reduce errors, and streamline system operations. Administrators should be able to manage access rights, view logs and conduct other necessary operations in an effortless manner. The UI should be laid out logically and ensure all features are easily accessible. It should have relevant information upfront such as recent logs or transaction records which should be presented in a well-organised table format. It should be easy to navigate between different sections with the use of icons and labels to guide the users through the system. The interfaces buttons should be clearly labelled and controls for adding unaccustomed users or updating new permissions should be clear from the rest of the operations. With these actions there should also be confirmation prompts to ensure the admins are aware of any changes they are making. To accelerate development process and since this is only a model project, we can Tkinter or PyQT for simple GUI design and implementation.

Figure 9 Tkinter Sample Interface

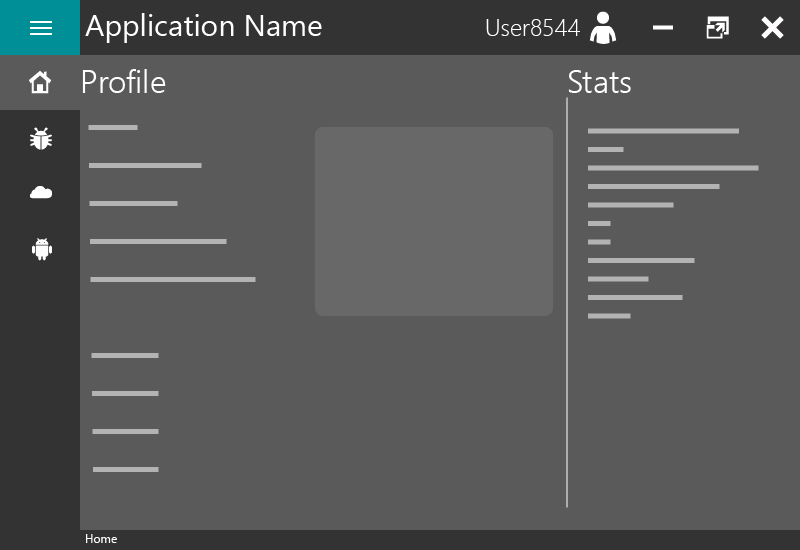
[](https://github.com/anjalp/Minimalistic-Flat-Modern-GUI-Template)

Figure 10 PyQT Sample Interface

### **Database Design**

As this application will be reading information from each NFC enabled card or tag and will be storing information based each user, the application will need a database management system to store all of this information. For this system, the Raspberry Pi will function as the server and the client, so all of the application does not need to run on a network unless you are trying to call the API from a different device. Since the complete system is relying on hardware integration and fast responses from a database, opting to use a local database on the microcontroller is best for this system. In a more centralised application, it would be more beneficial to use a cloud database as there would be multiple hardware points in a building.

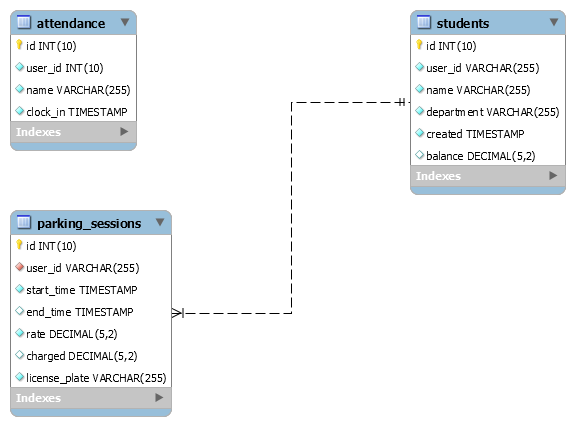


Figure 11 EER Diagram

The database design of the system is shown in Figure 11. As stated before, this entire database is being run locally on the Raspberry Pi. This database does use MariaDB which makes it suitable for a small-scale application such as this one. It is open source and free to use which reduces the overall cost of the system.

The database only consists of 3 tables with the students table being the reference point of where all the user information is stored. All registered students and staff go into the students table where they are identified by their user\_id which stores the RFID\_UID of their designated NFC card, their name and department which is necessary in distinct parts of the application. The created attribute stores the information of when the user was registered and the balance references, how much money the user has on their card which is used in the parking and top up applications.

The parking\_sessions table stores all the information relating to a parking session. It does take information from the students table based on the RFID\_UID of each card. From there it can use the balance in order to deduct money from the balance when a user pays for parking, as well as storing the charged amount and license plate.

The attendance table keeps track of the attendance. While there is no relationship between attendance and students, the application does abstract the name in the code and stores it there and compared the scanned RFID\_UID to one in the students’ table.

### **Summary**

The NFC Enabled Student Card System with API Integration project proposes using Near Field Communication technology to improve campus operations' usefulness and security. The system is aimed to serve students, lecturers, and administrative staff by providing smooth access management, accurate attendance tracking, and cashless transactions, encouraging a more connected, digital campus environment. The project considers both hardware and software requirements, with a focus on intuitive user experience and solid system performance. The NFC RFID Reader Writer and Raspberry Pi were chosen for their dependability and interoperability, while the software stack is based on strong database systems and programming languages like Python. The system's design emphasises modularity, scalability, and security, with an administrative user interface that provides a simple and effective management interface. To ensure project flexibility, an alternate design is being studied, which suggests using Arduino as a backup for the Raspberry Pi and additional NFC modules if necessary. This strategy ensures a durable and adaptive system capable of overcoming any implementation challenges, with the goal of achieving a decisive solution that effectively incorporates NFC technology into the fabric of campus life.

# **Implementation**

## **Introduction**

This chapter of the Final Year Project will discuss the implementation of the NFC Student Card System. We will outline the technical components, design decisions and practical implementation of this system. The NFC Student Card System is designed to improve and streamline processes for students, providing easy access control management and attendance tracking as well with API functionality. It combines hardware and software components to create a reliable and user-friendly experience for both students and staff.

The Technical Information section aims to provide an in-depth analysis of the technical foundations of the system. It will discuss the hardware elements such as NFC and RFID readers, RFID enabled tags and cards as well as the software architecture that enabled efficient data management and security. The design decision focuses on the crucial design choices that influenced the system, emphasising being adaptable and scalable to meet the need of potentially changing the system if required.

The System Structure section will provide a detailed explanation, including the arrangement of the folders and files that are essential for it functioning. This section will go through the system’s architecture which includes the logic on the sever side, the interfaces that clients communicate with, and the protocols that regulate NFC/RFID transactions. This section discusses the key components of the system which include main.py for the primary application logic, check\_attendance.py for attendance tracking, save\_user.py for assigning a new student to a new student card, unlock.py for access control functionality, top\_up.py to demonstrate updated the balance of the user, and parking\_sys.py to demonstrate the parking system. Their respective functions within the system are explained in detail.

The System Deployment and User Interaction section aims to provide an understanding of how the system is deployed and operated. It includes a demonstration of the interface that students use and the administrator dashboard that allows for system oversight. This section will provide visual examples and code snippets to provide a clear and detailed representation of the system’s functionality, encompassing student registration, daily access, and attendance tracking.

In conclusion, the Summary section will reflect on the process of implementing the NFC Student Card System. The section emphasises the system’s influence on simplifying campus operations and improving the student experience. It also suggests the possibilities for future improvements and adjustments in response to the changing technology environment and campus requirements.

## **Technical Information**

### **System and Software Design**

Adopting agile and flexible methodologies in software development is essential for meeting specific requirements in system development. This project adopts the Explorative Software Development Methodology. This approach deviates from the traditional linear models by promoting a more iterative approach and incremental approach that encourages continuous feedback and adaptation throughout the development lifecycle.

The Explorative Software Development Methodology is distinguished by its iterative cycles of exploration, prototyping, and refining. The initial phase of this methodology includes developers collaborating closely with stakeholders to define and understand the core requirements and objectives. In contrast to the rigid requirements phase in the Waterfall approach, this stage is unrestricted, enabling a thorough and flexible collection of requirements that can develop and change over the course of development. Following the initial research, the project progresses to a phase of fast prototyping. The primary objective is to create functional prototypes that encapsulate the identified features and functionalities. Following the prototyping phase, the system goes through a process of iterative testing and development. This phase resembles the system testing stage of the Waterfall methodology but is implemented continuously across each feature or module.

The verification phase entails the integration and validation of all system components to ensure that they meet the project’s goals and perform as expected. This phase is a repetitive process that takes place during the entire project life cycle, guaranteeing that the system is constantly matching the user requirements. Finally, the maintenance phase focuses on providing continuous support and improving the system. Given the explanatory nature of this methodology, maintenance not only involves bug fixing but also adapting and extending the system to meeting new requirements and incorporating new features as they become relevant. By utilising the Explorative Software Development Methodology, the project gains an advantage from a flexible and user-focused approach, ensuring the system is functional and dependable but in line with requirements (GfG, 2023).

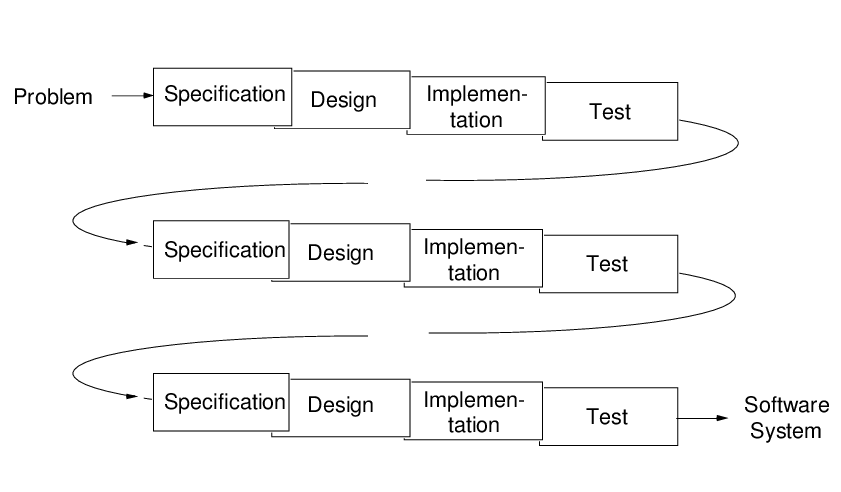
Alongside this development approach, software design is also necessary to produce a seamless combination of functionality, dependability, and user-friendliness. This can include and is not limited to architectural design, system definition, interface design, component design, data structure design and algorithm design. By following these, the system is designed to fulfil the requirements of the proposed system, enabling efficient access control management, attendance tracking and all the other proposed functionalities. Below in Figure 13 is an overview of the system architecture.

Figure 12 Exploratory Software Development

A diagram of a software system

Description automatically generated

Figure 13 System Architecture

### **Circuit Design**

The circuit design of this system is centralised around a Raspberry Pi connected to multiple peripheral devices for the applications proposed for the system. As shown in Figure 14, the Raspberry Pi serves as the central processing unit and acts as the primary controller for the system. The RC522 RFID reader is the unit used to process all the RFID functionality in the system such as access control, top up of balance. The LCD display is used to present messages to the user derived from the system such as user prompts or error messages. The relay module is linked to an electronic solenoid lock to regulate power to activate the lock mechanism. The buzzer is used to notify the user if they have not been granted access for the access control method. The power source is used for the relay to provide power to the lock. The wiring establishes the connections between the different components, and the GPIO pins on the Raspberry Pi, enabling the transmission of data and control signals.

A close-up of a computer

Description automatically generated

Figure 14 A top level view of the physical circuit

### **Project Management**

Within the field of software project management, the primary goals of following established timelines and remaining within the allocated budget are of the highest priority. Despite being motivated by academic and practical reasons; this project recognised the importance of quickly providing a fully functional system to end-users by a deadline. Punctuality was essential to prove the project's practical significance within the educational technology field.

Software development, especially in the field of educational technology, sometimes faces the difficulty of precisely predicting project deadlines, especially when it involves innovative technologies or approaches. Planning and schedule estimation in this case were flexible processes that adjusted as the project progressed. Strategic project milestones were set up to track the various stages of development and guarantee continuous development.

The project required a developer to manage several responsibilities: project management, design, programming, and testing. This integration of tasks required the developer to possess a strong sense of self-motivation and discipline, in addition to external supervision.

A significant milestone in the project was the creation and implementation of the user interface using Tkinter which is Python’s native GUI toolkit. This package facilitated the development of a clear user interface within the codebase itself, ensuring smooth integration with the underlying Python software. The integration of a graphical user interface into the system improved user interaction and experience.

The following developments included the creation of Python modules for managing users, tracking attendance (check\_attendance.py), and controlling access (unlock.py). These modules established communication with both the Raspberry Pi server and the local database. Determining the period of development was most noticeable for the components that interacted with hardware. Modules like the I2C LCD display (I2C\_LCD\_driver.py) and the SimpleMFRC522 package provided quicker solutions to building the project as they had come with predefined functionalities to interact with the hardware.

The project progressed through many stages, starting with its initiation, after the establishment of the Raspberry Pi server and the development of a local database, and concluding with the deployment of the user interface and functional modules. Each step was planned and implemented, resulting in reaching the proposed requirements of the project. Each achievement served as more than simply a point of evaluation, but as a chance to improve, acquire knowledge, and adjust, guaranteeing that the result not only achieved its original goals but also was strategically prepared for future improvements.

### **Design Quality**

The project placed a strong focus on the aspect of design quality. Factors like cohesion, interaction, comprehension, and flexibility have an impact on the quality of a software design. Cohesion is the extent in which the components inside a module are interconnected and contribute to a unified functionality. The Python modules were designed with a focus on achieving high unity. This means that each module, such as the user interface (main.py) or attendance tracking (check\_attendance.py), was specifically created to manage a single component of the system's operation and keep its concentration on that aspect.

Coupling, which measures the level of interconnection among various modules, was controlled to maximise the project places a high emphasis on software reliability, given its intended demographic of inexperienced users who may not have availability of technical assistance. To ensure dependability, the system was developed with the intention of reducing the likelihood of errors. The loose coupling played a crucial role in enabling the modification or replacement of modules with minimal errors for the rest. For example, changes made to the registering a user logic in the ‘save\_user.py’ script did not have any impact on the parking system procedures in ‘parking\_system.py.’

Simplicity is also essential; the code needed to be easily comprehendible to facilitate maintenance. This was accomplished by sticking to consistent naming conventions, creating thorough documentation, and keeping a manageable degree of complexity. Every Python file was given a logical name (e.g., ‘unlock.py’ for access control) that clearly indicated its purpose inside the system.

A screenshot of a computer

Description automatically generated

Figure 15 A screenshot of naming conventions in the project

Another important goal in the design was to provide adaptability, considering future requirements and the possibility of system evolution. The project's design facilitated the implementation of updates, whether in reaction to new needs or the addition of additional hardware components, with little difficulty.

The initial configuration and database scripts of the Raspberry Pi server were created with identical ideas in consideration. The bash scripts developed for database management were designed to be uncomplicated and modular, facilitating comprehension and modification as required. This approach prioritised clarity and simplicity over the utilisation of the most advanced or complex solutions, even when incorporating modern technologies.

The system's architecture was designed to embody these concepts, with every component, ranging from the Python modules to the Raspberry Pi server configuration, demonstrating robust internal unity, little external interconnection, high comprehension, and the ability to be easily modified. By adhering to these design qualities, the project established a strong basis for a resilient, sustainable, and expandable system.

### **Program Reliability**

The project places a high emphasis on software reliability, as the intended demographic of users may not have immediate technical help or will be very new to the system. The system interactions are designed to be user-friendly, including straightforward mouse clicks or typed inputs for specified functions, such as recording attendance or granting access. The minimalistic design minimises the probability of errors caused by users. It is essential to guarantee the dependability and user-friendliness of all features. If users experience problems or see the software as unreliable, it could impact their trust and willingness to interact with the system.

Integrating a strong error management is crucial for improving system dependability, particularly when the users of the system may not technically proficient. The project does provide comprehensive error logging to ensure that the user and developer understand what the errors are. The system provides user-friendly error messages that offer clear, non-technical explanations and meaningful recommendations to ensure the user know how to action the next time they run the application and avoid these failures.

A screen shot of a computer program

Description automatically generated

Figure 16 A screenshot of error handling

A computer screen shot of text

Description automatically generated

Figure 17 A screenshot of error handling

For this system, threading was essential to ensure a responsive user interface while managing background tasks such as database access and hardware interfacing. By running various tasks in separate threads, the system ensures that operations such as RFID scans and database changes do not block the main user interface, enabling it to remain responsive. This methodology significantly enhances user experience by minimising waiting times and enabling simultaneous processing of multiple tasks. Threading plays a key role in improving system performance and reliability ensuring that even during complex processes, the user interface remains responsive and efficient. (Bauer, 2017)

A screen shot of a computer program

Description automatically generated

Figure 18 A screenshot of threads

Using these design methods, the project guarantees a dependable and user-focused experience, which enhances the users' trust and creates an ideal environment for technological engagement, especially for individuals with minimal computer literacy.

### **Software Reuse**

The goal of the software produced in this project is to ensure its reusability across multiple scenarios. To accomplish this, the project follows specific design principles. It was created with a strong focus on modularity, using Python files as the basic components of the system. The naming rules were carefully deliberated to guarantee clarity and consistency throughout the codebase. The reusable components, including distinct functions and classes, were contained within separate modules for easy maintenance and enable potential reuse in various sections of the system as well as the future.

When designing our system, we took into consideration the built-in cross-platform capabilities of Python, ensuring that it was not just modular but also portable. The solution is meant to function across various computing settings and operating systems by utilising Python's adaptability and the Raspberry Pi's extensive connectivity. This forward-thinking perspective defines the project as a viable option for future adaptations and integrations, regardless of technology improvements.

Another factor to consider was the system's capacity to manage data in a manner that would ensure long-term accessibility and convertibility. The principles of data independence and future proofing were of importance. The system stores data, such as user information and access logs, in a standardised format that allows for simple export and manipulation if necessary.

Python scripts and the Raspberry Pi platform provide a promising opportunity for future research and development. With the constantly evolving technological environment and the development of new data management approaches, the project's organised approach and the decision to utilise widely supported technologies ensure that the data and software components may be easily adjusted or transferred to other formats and systems.

The project includes the fundamental concepts of modularity, portability, and future adaptation, which are crucial for ensuring the software's durability and ongoing significance in the ever-changing technology landscape.

### **Design Decision**

**Why a Python-Based Dashboard?**

The choice to use a Python-based dashboard (main.py) for the RFID functionality of the project was inspired by various practical factors. The RFID system has a separate dashboard interface that provides a streamlined controlled environment for users to interact with as well has the circuit of the hardware. The system enables the integration of immediate input and interactive components, which are essential to controlling and overseeing operations that rely on the use of RFID.

Using a Python-based dashboard customised for the Raspberry Pi allows improved communication and user involvement, surpassing the capabilities of a web-based interface, especially in offline environments or areas that lack internet connectivity. It enables the user to easily move through various features, hence improving their independence and the overall usability of the system.

The use of the dashboard method is advantageous due to Python's extensive collection of libraries and frameworks. This resilient support system allows for the development of a highly adaptable and user-friendly interface for RFID interactions. The modular structure of Python code facilitates the reuse and scalability of the dashboard, enabling it to adapt to the evolving needs of the system.

Additionally, the widespread knowledge of Python within the development community ensures that developers with different degrees of skill may effectively maintain and enhance the dashboard. As the system usage expands, the dashboard can be improved to incorporate other functionalities such as data visualisation, user administration, and system configuration choices.

**Benefits of a Dashboard Approach**

1. Centralised Control: The dashboard serves as the central hub for all RFID-related functions, providing users with a single point of interface with the system.
2. Improved User Experience: A carefully developed dashboard can significantly improve the user experience by providing information and controls in a structured and easily accessible manner.
3. The Python dashboard shows adaptability and extensibility, enabling seamless adjustment to new demands, such as the addition of additional hardware functionalities or the enhancement of the user interface.
4. Operational Independence: In contrary to web-based solutions, the dashboard functions independently of web connectivity, enabling its operation in diverse situations, such as remote or isolated places.
5. Community Support and Resources: Python's vast community provides developers with a diverse range of resources for resolving issues and improving the dashboard.

**Why an API?**

JSON (JavaScript Object Notation) and APIs (Application Programming Interfaces) are foundational technologies that facilitate the integration and compatibility of modern software systems. Unlike HTML (HyperText Markup Language) is primarily intended for organising and displaying material on the web, JSON is specifically created for data exchange, offering a compact, text-based format that is easily readable by human and machines. APIs, especially those that use JSON, provide seamless communication between different software components, systems, or services, allowing them to exchange data and functionality. This section examines the justification for using JSON and API technologies in this project.

The core principles underlying JSON and APIs is to promote decoupled architecture allowing the exchange of data and services can occur independently of their internal implementation. This separation enhances adaptability, expandability, and ease of maintenance across multiple systems. JSON can be used to serialise and transmit structured data across a network, especially between server and web applications. APIs, in contrast, establish the protocols and data structures that other systems can use to exchange information or use each other’s capabilities. Together, they facilitate a modular approach for system design, allowing for the development, modification, or substitution of components without impacting the entire system.

**Main Advantages of JSON and APIs**

1. Interoperability: JSON is independent of any specific computer language, meaning it can be generated and interpreted by various programming languages, making it a suitable format for data exchange across different systems. APIs that employ JSON further enhance interoperability by offering a standardised method for applications to interact.
2. Efficiency and Performance: JSON's efficiency and performance requires less bandwidth, leading to faster data transfer rates and better performance compared to other data interchange formats. APIs that are designed for JSON data exchange contribute to this efficiency, enabling real-time data processing and responsiveness.
3. Simplicity and Accessibility: JSON's format is straightforward, making it easy to write and understand without the need for specialised tools or parsers. APIs that communicate via JSON inherit this simplicity, allowing for quick development cycles and easier integration.
4. Flexibility: JSON data offers flexibility, allowing it to easily adapt to the specific needs of the application. APIs can use this flexibility to evolve over time, adding or modifying data fields without disrupting existing services.
5. Security: APIs can implement robust security protocols to protect data exchange, including authentication, authorisation, and encryption. When combined with JSON's simplicity, this ensures secure yet accessible data communication pathways.

Although JSON and APIs provide multiple benefits, it is also important to consider their limitations and challenges, such as security implications of open APIs, the need for thorough documentation, and potential performance issues with poorly designed API interfaces. However, with proper design and implementation strategies, JSON and APIs remain pivotal technologies that support the modular, interconnected, and efficient systems that define the current world of software.

**Disadvantages of a Python Based Dashboard**

While a Python based dashboard for RFID functionality has several advantages using Tkinter, like centralised access control, enhanced user experience, and operational independence, it also presents certain challenges that need to be addressed. It is vital to recognise these issues to have a full understanding of the technical decision to use this type of technology and minimise any issues that may arise.

**Challenges and Limitations of a Python based Dashboard**

1. Performance Constraints: Python’s ease of use and versatility can be compromised when compared to compiled languages such as C or C++. Where real time data is being processed and high speed interactions are occurring, Python may impose limitations especially on low-powered devices such as the Raspberry Pi or Arduino.
2. Resource Intensive: Python applications, particularly those with GUIs (Graphical User Interfaces), tend to require more resources compared to alternatives written in lightweight or native frameworks. This can impact the performance of the system especially in environments like embedded systems or IoT devices.
3. Security Vulnerabilities: Any software project imposes security risks including applications built in Python. Python libraries and frameworks that are open source can be vulnerable to attacks if there are not constantly updated secured.

While choosing a Python based dashboard for managing RFID functionalities present many advantages, it may also come with challenges that need to be accounted for. Understanding these limitations enables the development of more effective and secure ways for developing and implementing the dashboard ensuring it is useable and user-friendly as the project progresses.

**Disadvantages of JSON and APIs**

JSON and APIs also provide many benefits for modern software systems, including interoperability, efficiency, and flexibility but they also have their own issues and challenges that need to be addressed. These disadvantages are necessary for developers to identify to minimise issues that arise during the design and implementation phase.

**Challenges and Limitations of JSON and APIs**

1. Data Overhead: JSON being as lightweight as it is compared to other data exchange formats it still incurs some additional data due to its textual structure. This can have a considerable impact on performance and bandwidth utilisation, especially in application involving high volume transactions or the transmission of complex or deeply nested data structures.
2. Security Risks: APIs that can be publicly accessible pose vulnerabilities. Insufficient security measures, such as absence of rate limitation, encryption etc… can make APIs susceptible to data breaches, unauthorised access and DDOS attacks (Denial of Service).
3. Complexity in API Management: As systems grow, managing a collection of APIs can becoming more complex. Maintaining consistency, implementing API versioning without causing disruption to users and providing complex documentation for developers can demand substantial dedication and resources.

JSON and APIs are powerful tools for building interconnected software systems. Even though they provide many challenges, developers can counter this by designing fool-proof APIs with strong security measures and comprehensive documentation.

## **System Architecture**

The created system consists of various important components, each contributing to the overall functionality and user experience. The purpose of this section is to offer a brief overview of the system's design. This project uses Python scripts and a Raspberry Pi server to oversee and conduct its fundamental operations, adopting a distinct methodology.

### **Overall Structure**

The NFC Student Card System has an organisational structure that effectively separates and categorises its different functional components. This structure reflects the complex of the system operations and make it easier to navigate and maintain. The directory and file structure follows a hierarchical arrangement.

At the highest level, the system includes:

* The ‘NFC\_Enabled\_Student\_Card\_System root directory: This serves as the central location for the project related materials, including the project’s source code.
* The README.md file outlines the documentation of the project, how it functions, how to run the application, etc…
* The requirements.txt file includes all the Python dependencies that need to be installed in order for the project to function.

Within the root, the directory structure branches out into several key components:

* The ‘assets’ directory contains an image used in the README, and SQL script to create the database for the project, report.html is the web report for the output of the unit tests, and the style.css is for the unit test report.
* The ‘dependencies’ directory contains external dependencies.
* The ‘modules’ directory acts as the core of the system’s functionality. It contains the Python modules that constitute the system’s operations.
* The ‘tests’ directory hosts all the unit tests developed for the project.

**Assets Directory Structure**

The project's assets directory stores the essential files that may be miscellaneous but are required for some part of the functionality of the project. The content of this directory is separated from the main logic which any database scripts, images, web pages etc…

* The directory houses dependencies required documentation but not for the logic of the application. The ‘img.png’ is used in the README to provide a diagram of the circuit diagram.
* The ‘nfcstudentsys.sql’ is the database script used to create the tables and insert all the data for the project.
* The ‘report.html’ is used display the results of the unit tests, which displays whether the tests pass or fail. This has been generated by ‘pytest’
* The ‘style.css’ is used to style the report, which is also generated b ‘pytest’.

### **Dependencies Directory Structure**

The project's dependencies directory plays a crucial role in maintaining and storing the essential external libraries and drivers needed for the system to function properly. The layout and contents of this directory are intentionally selected to contain the external dependencies of the system, enabling a separation from the core application functionality.

* The directory houses essential drivers or dependencies required for hardware connection. The ‘I2C\_LCD\_driver.py’ which was obtained from vay3t’s implementation of Denis Pleic’s ‘RPi\_I2C\_driver.py’, provides a means of connecting and communicating with the LCF display via I2C communication protocol. It simplifies the direct contact with hardware by abstracting its complexity, offers a streamlined approach for the project.
* The ‘\_init\_.py’ file located in the dependency’s directory, serves as a marker that designates the directory as a Python package module. This allows the content of the directory to be imported and used modularly in other portions of the project. It signifies the directory’s role in the overall framework.

A screenshot of a computer program

Description automatically generated

Figure 19 Sample code from I2C\_LCD\_driver.py

### **Modules Directory Structure**

The modules directory in the project, is structured to contain the essential features needed for the system’s operations. The main logic and process flows are located here.

* The ‘\_init\_.py’ file located in the dependencies directory, serves as a marker that designates the directory as a Python package module. This allows the content of the directory to be imported and used modularly in other portions of the project. It signifies the directory’s role in the overall framework.
* A computer screen shot of a program

  Description automatically generatedThe API segment acts as the forefront for the API logic and includes the functionality of displaying students that signed in for attendance within the last hour.

Figure 20 API Code Snippet

* The check\_attendance.py’ module is responsible for the attendance tracking feature of the system. It manages the logic for recording and verifying the students that have signed in for class, interfacing with the RFID readers to collect and validate attendance data.

**A screenshot of a computer program

Description automatically generated**

Figure 21 check\_attendance.py Code Snippet

* The ‘main.py’ module houses all the functionality in one file to allow for flexibility in code and be wrapped around a GUI. In includes threading, control flows, setting up the environment etc…

**A computer screen shot of a program

Description automatically generated**

Figure 22 main.py Code Snippet

**A screenshot of a computer program

Description automatically generated**

Figure 23 main.py Code Snippet

* The ‘save\_user.py’ module contains the logic for adding new students to the system and assigning their department as it is crucial for the access control part of the system.

**A screenshot of a computer program

Description automatically generated**

Figure 24 save\_user.py Code Snippet

* The ‘unlock.py’ module contains the logic unlocking based of the authentication of the card presented.

**A screenshot of a computer program

Description automatically generated**

Figure 25 unlock.py Code Snippet

### **Completed System**

A screenshot of a computer

Description automatically generated

Figure 26 A screenshot of the main system

A screen shot of a parking system

Description automatically generatedFigure 27 A screenshot of the parking system

A screenshot of a computer

Description automatically generated

Figure 28 A screenshot of the top up system

A small electronic device with a blue screen

Description automatically generatedThe completed system works as 3 part system. The first application is where the users can either interact with the main system which hosts the API call, unlocking a door, registering a user, and logging attendance. The second system is where the user can log their parking session. The third system is where the user can update the balance to allow for more money for parking. All three systems have been developed using Python as the language and Tkinter as the GUI design. As well as the GUI, the user does have the hardware in front of them, the LCD will be displaying information to them as they progress through the application.

Figure 29 A picture of the LCD displaying messages

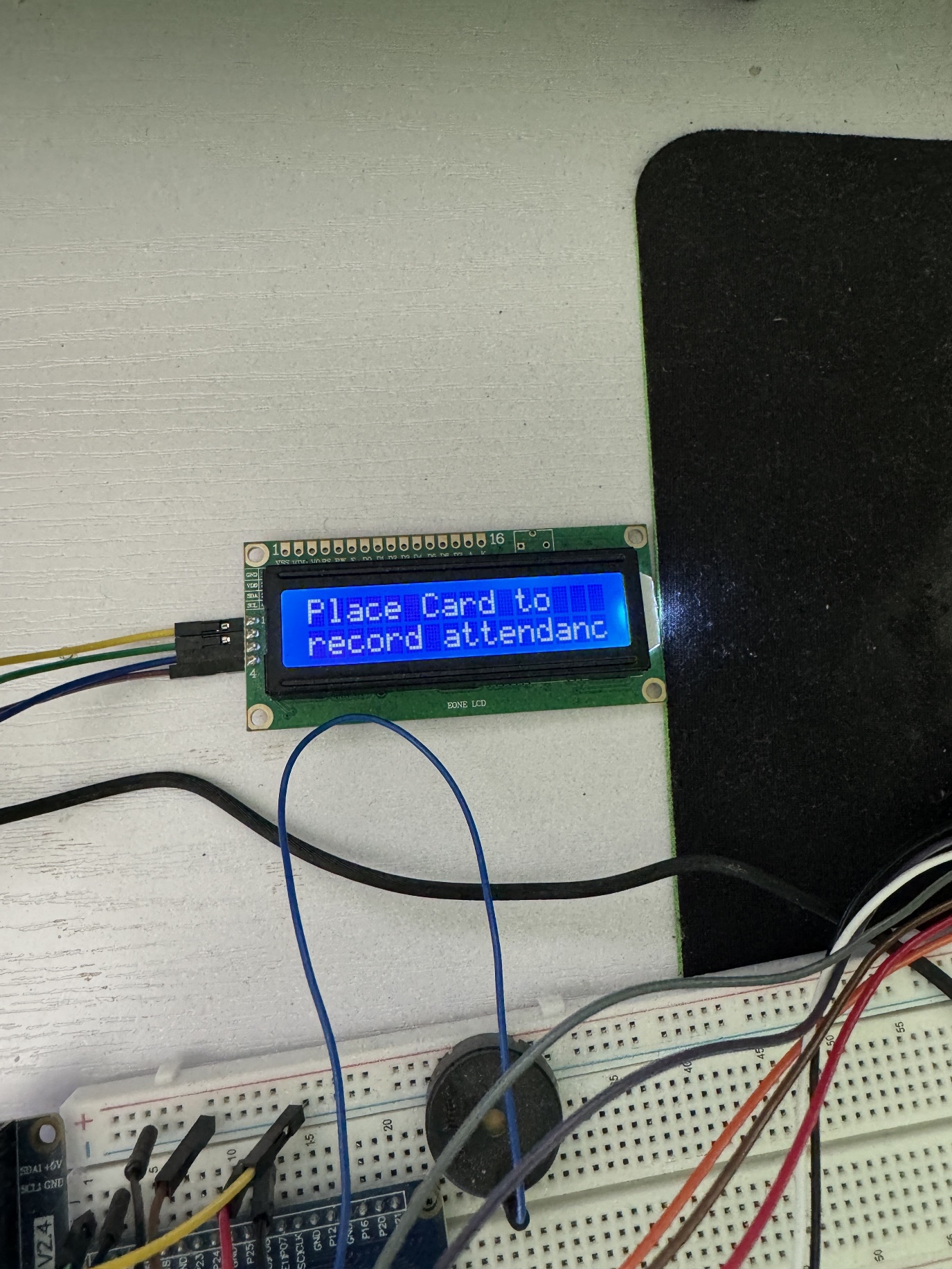
All the functionality for the user is laid out in one place for each program. There is no authentication required for the system, the user just has to click the buttons on the application in order to run the different methods. Each button corresponds to each method call and is clearly labelled. When the user runs main.py, the first thing they will see 4 buttons that are clearly labelled for their assigned methods, a drop down of the department, a text box to input names and an output box to display messages.

Figure 30 A picture of the LCD displaying messages

The first button on main.py is the button to start the API server and open the browser. It will take the last hour of recorded attendance from the database and display it on the browser. If there is nothing in the past hour to be called it will just return an empty page.

A screenshot of a computer

Description automatically generated

Figure 31 A screenshot of the API call for attendance tracking

From there the user can choose between, ‘Unlock Door,’ ‘Register User,’ or ‘Attendance.’ The first two methods do require inputs from the user. For ‘Unlock Door,’ the input required is the department. As default IT is set as the default department, but there is a dropdown menu that the user can select from in order to set the department.

A screenshot of a computer

Description automatically generated

Figure 32 A screenshot of the department dropdown

A screenshot of a computer

Description automatically generatedOnce the department is set the user clicks ‘Unlock Door’ and they are prompted to scan their card. Once the card is scanned, it matches the scanned RFID\_UID to one in the database and checks if the department matches the one that has been selected. If the RFID\_UID is found and the department for that user matches what has been selected in the application, a message is displayed informing that they have received access, as well as a buzzer in the hardware to notify the user that access have been granted.

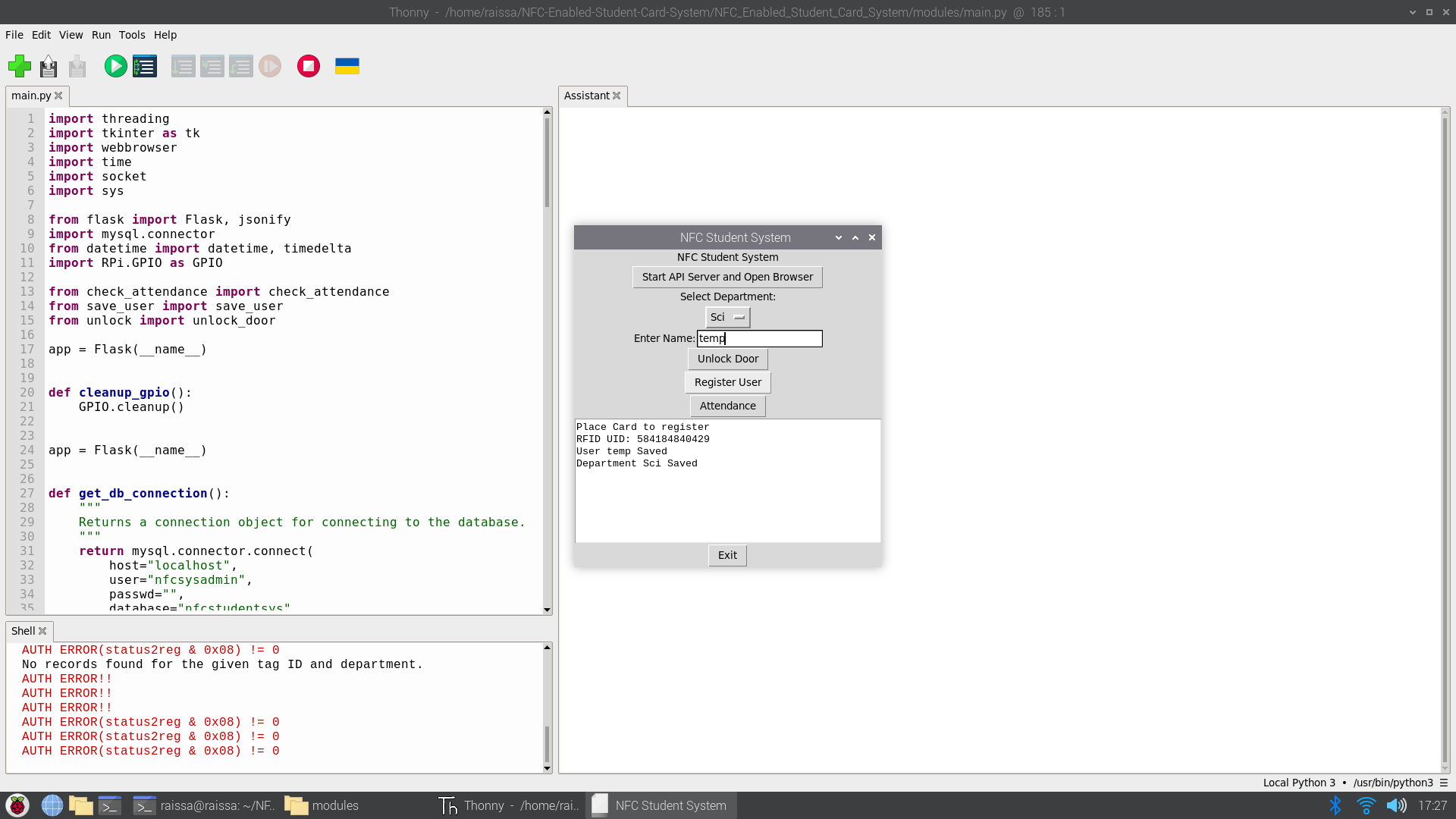
From there, the user can then set the department once again and then enter a name in the text box provided. The user then clicks ‘Register User’ which prompts the user to scan their card. It takes the scanned RFID\_UID and finds it in the database, if the RFID\_UID already has been assigned to a user in the database, the system displays a message that there is already a registered user. If there is not a registered user, the system will display a message that the user and their department has been found.

Figure 33 A screenshot of the Unlock Door method

Figure 34 A screenshot of the Register User method

From there, the user can click ‘Attendance.’ There are no inputs required from the user apart scanning the card. When the user clicks the button, they are prompted to scan their card to record attendance. If the user is found it responds that they have signed in, otherwise it will display that the user is not found.

A screenshot of a parking system

Description automatically generatedA screenshot of a computer

Description automatically generated

Figure 35 A screenshot of the parking method

Figure 36 A screenshot of the attendance system

For parking\_system.py, it required 3 inputs; the number of hours required for parking, the license plate, and the card to scan. Before the user clicks ‘Start Parking Session,’ the must input the number of hours for parking and the license plate. Once this information has been entered and the user clicks the button, it prompts the user to scan their card. Once the card is scanned, it completes the calculation and displays all the information related to parking. To the data base it saves the parking id, the RFID\_UID, the start time of the parking, end time of the parking, the rate charged and how much they have been charged as well as inputted license plate. The rate of charging is 50 cents for the first 2 hours and then increases to €1 per hour, which is then deducted from the balance of the user which is stored in the student’s table. The system does flag an error if there are no hours inputted or if the hours are less than 0.

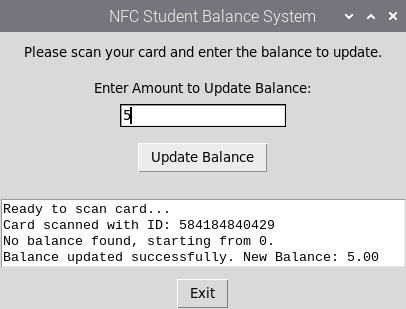


Figure 37 A screenshot of the top up method

For top\_up.py, it requires 2 inputs; the amount the user wishes to top up by and the card to scan. Similarly to the parking system, the user must input the amount they wish to update their balance by. Once that has been entered the user can click the button. It will then prompt the user to scan their card. Once it has been scanned, it will find whether or not the balance is null or has a preexisting balance. If the balance is null, it will inform the user that they are starting a new balance, or it will be added onto the current balance.

A screenshot of a computer

Description automatically generated

Figure 38 A screenshot of the top up method

All of the methods in this application are threaded meaning its able to execute multiple parts of the system at the same time. For example, when the API is executed, it does not interfere with the GUI and other processes, the user is able to continue to use the application without interference. When the user clicks the ‘Exit’ it ends all the processes, clears the GPIO pins and closes the window.

# **Testing and Results**

## **Introduction**

As part of developing any software or hardware testing is crucial in order to ensure that the project is suitable and meets the expectations to reach the market. Testing is the method used to verify and validate software to ensure it meets its technical requirements and to ensure it is free from defects and errors (GeeksforGeeks, 2017). Testing is the methodology used to exam software components and systems are performing as intended against predefined criteria. This chapter will demonstrate the project undergoing a comprehensive testing strategy, studying different testing components, test cases and automated testing. This application focuses on functional and unit testing as part of the system development to ensure the goals of the project were achieved as projected. During the development of the project, testing was actively done to ensure that usability and functionality was consistent throughout the code base to reduce errors in the code and ensure that they could be addressed as soon as possible. Even though a software system of high quality can still harbour bugs that may have been overlooked by a developer, it is important for the developer to understand and where the bugs may have come from and rectify the issue as soon as the bug is discovered. This chapter will discuss the results of the testing procedures, how bugs are addressed and how the system still maintains an elevated level of quality.

## **Software Testing**

As discussed before, testing is the method of verifying and validating a code base to ensure reliability and quality and that it achieves its functional and non-functional requirements. Verification is the process of checking if the software developed reaches a specification compared to validation which is the process of checking the requirements and expectations of the system. Both can be distinguished as validation requires code execution compared to verification that does not require it (Hamilton, 2024).

Functional requirements outline the products expected features, functionalities, and components. These can be operations, inputs, and outputs. These describe the capabilities the user will be physically using in the final product. Non-functional requirements outline how the product is expected to perform. These can be how useable, scalable, and maintainable the system is (Solutions, 2023). Both of these can be tested differently, functional requirements can be tested with unit tests and functional tests, whereas non-functional requirements can be tested with usability and performance testing.

The testing framework is centred around various methodologies designed to identify defects and verify that an application conforms to its outlined functional and non-functional requirements (Ahmad, 2024). Each methodology has a distinctive way to ensure the goal is achieved while maintaining quality and usability. They offer systematic frameworks for evaluating software functionality, dependability, and performance. For this system various testing methods were used to thoroughly examine its functionality and components.

During an SDLC or Software Development Life Cycle, testing is one of the stages that must be enforced to ensure the product being developed reaches the standards that have been outlined. This stage detects errors, verifies functionality, and reduces risk during the SDLC. By implementing some of the methodologies of the testing framework, developers are able to evaluate the quality of their code and optimise and improve their testing suites and code base.

When developing this system, multiple testing goals and objectives were addressed throughout the development of the project and used many testing methodologies. The testing began with outlining the user requirements and system specifications, providing a foundation for further testing. Functional testing was used to verify that the system’s features and functionalities satisfied user and system requirements. This involved designing use cases that simulated genuine user interactions and scenarios, guaranteeing the consistent performance of the application under different conditions.

In addition, unit testing methodologies were used to carefully examine the various components and units of code in isolation, confirming their accuracy and resilience with the outlined standards. They were created to cover a range of use cases to assess the relatability of the code base. By testing during the development of the project, defects in the system are found and are able to be addressed.

Functional testing is used to ensure that every feature and component of the system is conformed to user expectations, facilitating seamless interaction, and achieving the desired objectives and goals of a project. In other words, functional tests are completed to ensure that the functions are working as intended (Katalon, 2022).

Unit testing is used to ensure that individual units or components of a system are validated and working as intended. These units of code are isolated from the rest of the code base to ensure the functionality is operating as required (Hamilton, 2024b).

In this project, there are three systems to be executed; ‘main.py,’ ‘parking\_sys.py,’ and ‘top\_up.py.’ For each of these systems, there is a dedicated unit test file dedicated to testing individual components in each system. All of these unit tests were built in the systems native language of Python. Each of these tests could be ran in individually or together and would provide a pass or fail.

The functional testing was done based on the requirements of the project. These were separated as per each use case to ensure test coverage. For this test, this was documented in excel tables, with written sentences of each use case, inputs, expected output and actual output, as well if the test passed or failed.

## **Test Cases**

As discussed before, for both functional and unit testing there were separated by their use cases and systems respective to each testing framework. This section will discuss the test cases for the system and its results.

### **Unit Tests**

Figure 39 A code snippet of main.py unit tests

As shown in Figure 40, the first unit test suite is for ‘main.py’. The tests were developed using the Python packages ‘unittest’ and ‘pytest.’ To ensure that the tests would be able to pass without the direct need for a Raspberry Pi or the other hardware components, using unittest’s mock feature, the tests were able to be developed by simulating these components so that the tests can be run on any machine, as well as mocking the other dependencies such as ‘check\_attendance.py’, ‘save\_user.py’ and ‘unlock.py’. By doing this, the test is now able to be developed without the need of any of the dependencies outlined in the actual program. The fixture is able to set up the mocks and initialises the object for testing. From there the tests are as follows:

* test\_init(): Tests whether nfcsys\_obj’s master attribute is set up correctly.
* test\_start\_api\_and\_open\_browser(): Tests whether start\_api\_and\_open\_browser method calls the Thread class.
* test\_display\_message(): Tests whether display\_message method calls the insert method of the text widget.
* test\_unlock\_door\_threaded(): Tests whether unlock\_door\_threaded method calls the Thread class.
* test\_register\_user\_threaded(): Tests whether register\_user\_threaded method calls the Thread class.
* test\_check\_attendance\_threaded(): Tests whether check\_attendance\_threaded method calls the Thread class.

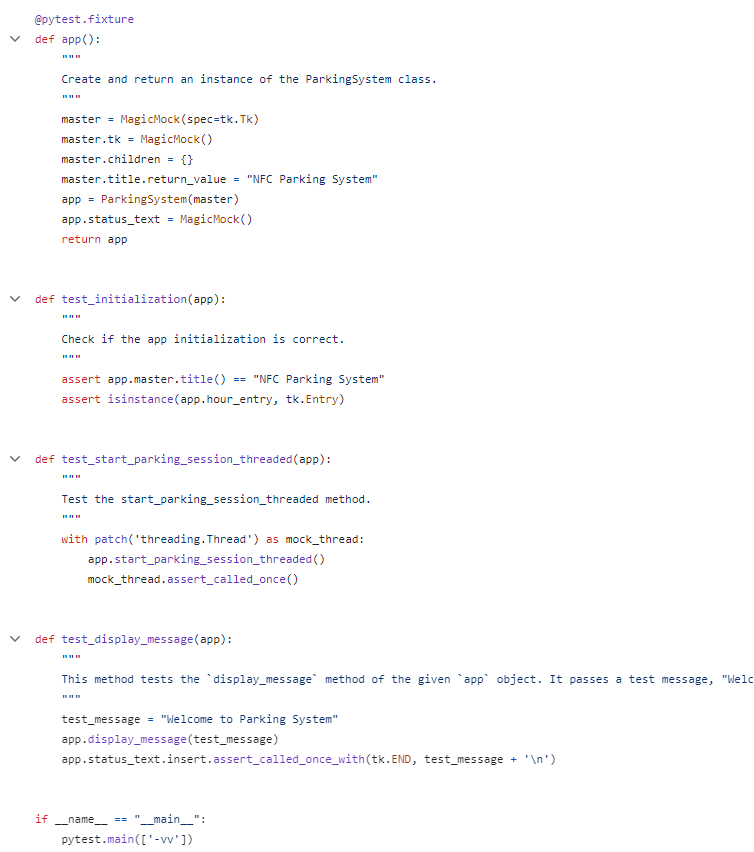
****

Figure 40 A code snippet of parking\_sys.py unit tests

As shown in Figure 41, this unit test suite is for ‘parking\_sys.py’. As well as the previous test suite, this test is using ‘unittest’ and ‘pytest’ and to avoid any dependency issues, the hardware is mocked to ensure test coverage as well as other software dependencies. The fixture is able to set up the mocks and initialises the object for testing. From there the tests are as follows:

* app(): Create and return an instance of the ParkingSystem class.
* test\_initialisaition(): Check if the app initialization is correct.
* test\_start\_parking\_session\_threaded(): Test the start\_parking\_session\_threaded method.
* test\_display\_message(): Tests the `display\_message` method of the given `app` object.

****

Figure 41 A code snippet of top\_up.py unit tests

As shown in Figure 42, this unit test suite is for ‘top\_up.py’. As well as the previous test suite, this test is using ‘unittest’ and ‘pytest’ and to avoid any dependency issues, the hardware is mocked to ensure test coverage as well as other software dependencies. The fixture is able to set up the mocks and initialises the object for testing. From there the tests are as follows:

* app(): Returns an instance of the NFCSYS application.
* test\_initialisation(): This method is used to test the initialisation of the tkinter application.
* test\_update\_balance\_threaded(): Test that the update\_balance\_threaded method starts a new thread to handle balance updates.
* def test\_display\_message(app): Test method to verify the behaviour of the display\_message method in the application.

### **Unit Test Results**

As part of the ‘unittest’ and ‘pytest’ framework they provide an automated testing method, which allows you to execute the tasks all at once and receive the results via the console or in HTML format. For visual purposes of understanding the test cases, this project has opted for the HTML, which provides a breakdown of if tests are passing or failing, and why they are failing. In a production level software, where the software would be developed and deployed on a CI/CD platform, these tests would be automated to run during a certain period to ensure that they are always passing but this project, we have saved the results to the project repository.

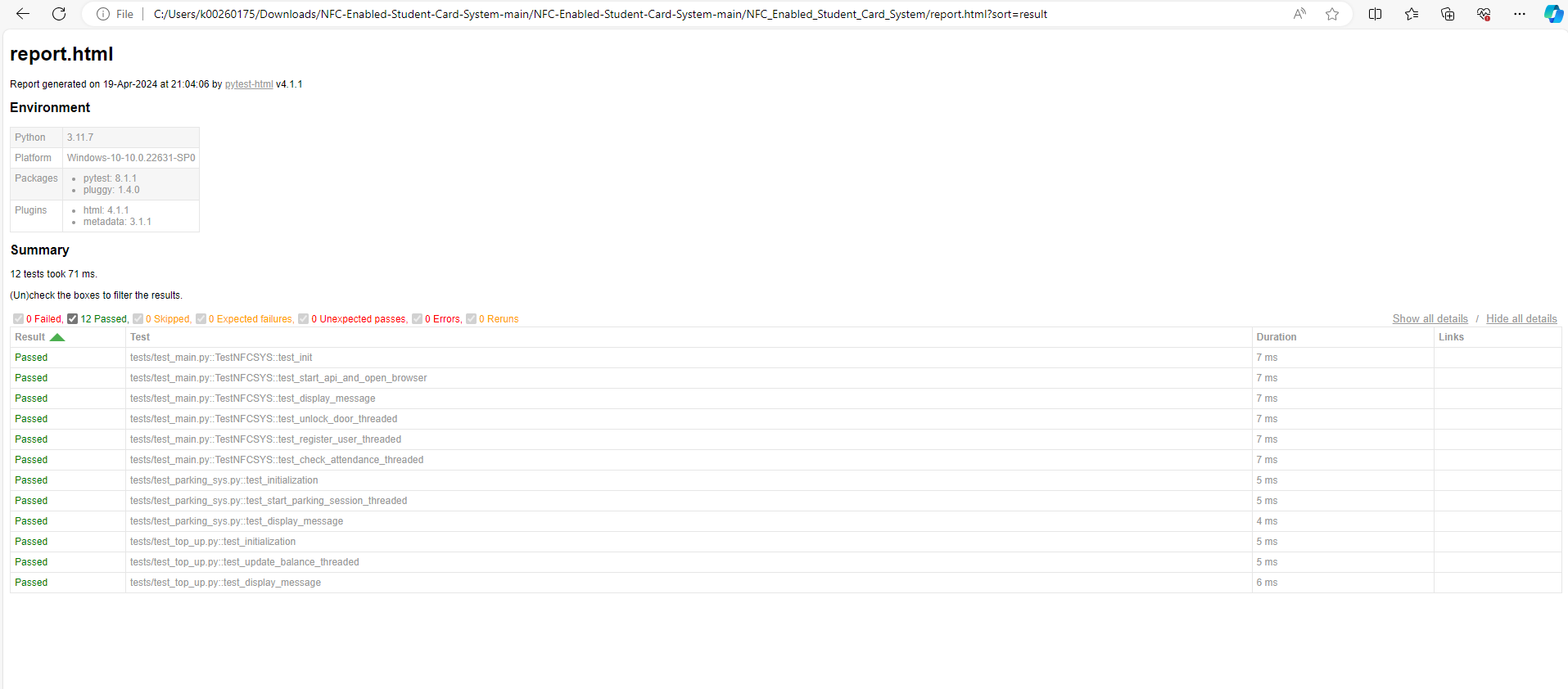


Figure 42 A screenshot of the unit test results

As shown in Figure 43, the tests all pass, and they are running as expected. This means that the tests have been correctly formatted to ensure that the functionality of the system is running as expected. Each one of the tests take a single unit of the code as part of the major code base and isolate it to ensure it is running as expected. Using mocks as a simulation tool ensures that the tests can pass without the need of hardware integrations, especially in the case if this project were to go on to a CI/CD platform.

### **Functional Tests**

As discussed previously, the functional tests were base to ensure that the expected outcomes of the system are occurring and if there are any failures, the developers should be able to rectify the code base as seen fit.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Input** | **Expected Output** | **Actual Output** | **Pass/Fail** |
| TC01 | Start Flask API and open browser | Click "Start API Server and Open Browser" | Browser opens displaying attendance data for last hour | Browser opens with JSON | Pass |
| TC02 | Fetch recent attendance records | GET request to /attendance/last\_1\_hours | JSON response with attendance records within last hour | JSON displaying records | Pass |
| TC03 | API server starts on system launch | Start system | Flask server running on specified port, ready to manage requests | Serving Flask app 'main' | Pass |
| TC04 | Able to access API on a different device | Start system and browser | Web browser opens automatically to specific API endpoint | JSON displaying records | Pass |

Table 6 Functional test table for the API in main.py

As shown in Table 6, the functional requirements for the API have all passed, the system is able to launch the flask API and open the browser when the user clicks the button, the API is able to retrieve the data from the database, as well and start the API server when the application is launched, and the user is able to access the API on a different device. All the tests passing shows this section of the system reaches all of its functional requirements.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Input** | **Expected Output** | **Actual Output** | **Pass/Fail** |
| TC01 | Unlock with valid user ID | User ID (valid), Department (Beauty) | Door unlocks, access logged | Access granted for department: Beauty | Pass |
| TC02 | Attempts unlock with invalid ID | User ID (valid), Department (IT) | Access denied, error message displayed | Access denied for department: IT | Pass |

Table 7 Functional test table for unlock.py in main.py

As shown in Table 7, the functional requirements for the unlock method of the main system have passed. The user is able to unlock a department with a valid user id with the set department that matches the department that has been set to that user as well as the user is unable to access a department that are not a part of. This shows that this part of the system reaches its functional requirements.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Input** | **Expected Output** | **Actual Output** | **Pass/Fail** |
| TC01 | Save new valid user | New user details | User saved, confirmation message | User Ed Saved Department Sci Saved | Pass |
| TC02 | Attempt to save duplicate user | Existing user details | Error message, no duplicate entry | RFID\_UID: 5841848645612 User already registered | Pass |
| TC03 | Update existing user information | Updated user details for existing user | User information updated | RFID\_UID: 5841848645612 User already registered | Fail |

Table 8 Functional test table for save\_user.py in main.py

As shown in Table 8, the functional requirements for the register a user method of the main system have not all passed. The user is able to register a new user, it returns the details that have been saved and database is updated. When the user tries to register the same NFC card again, the system informs the user that this user is already registered which is expected. However, there is no functionality for allowing the user to update an already registered user, which when the user tries to do it fails and displays the error that the user is trying to register the same user again. This means that this section of the system does not reach all of its functional requirements.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Input** | **Expected Output** | **Actual Output** | **Pass/Fail** |
| TC01 | Valid attendance mark | User ID (valid), Current time | Attendance recorded; confirmation displayed | Signed in Daniella | Pass |
| TC02 | Mark attendance with invalid ID | User ID (invalid) | Error message, no attendance record | User does not exist | Pass |
| TC03 | Repeated attendance mark | User ID (valid), Same day repeat entry | Message about already marked attendance | Signed in Daniella | Fail |

Table 9 Functional test table for check\_attendance.py in main.py

As shown in Table 8, the functional requirements for the attendance method of the main system have not all passed. The user is able to record their attendance, it does sign the user is as expected, if the user presents a card that has not been registered it returns that the user does not exist as expected but when the user tried to sign in again it does not inform the user that they have already signed in, it saves the record again which is not expected. This means that this section of the system does not reach all of its functional requirements.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Input** | **Expected Output** | **Actual Output** | **Pass/Fail** |
| TC01 | Valid top-up | User ID (valid), Amount: 50 | Account balance updated; confirmation sent | Balance updated successfully. New balance: €52.95 | Pass |
| TC02 | Top-up with invalid user ID | User ID (invalid), Amount: 30 | Error message, no balance update | User not found. Balance not updated | Pass |
| TC03 | Negative amount top-up | User ID (valid), Amount: -10 | Error message, no transaction | Invalid amount: Must be greater than 0 | Pass |
| TC04 | Exit application | None | Application closes, resources cleaned | Exits and shuts down | Pass |

Table 10 Functional test table for top\_up.py

As shown in Table 10, the functional requirements for the top up system have passed. The user is able to update their balance and they receive a confirmation is expected. The user does not have their balance updated if they present a card that is not registered which is expected. If the user inputs a negative value, their balance is not updated and provides, and error message as expected. When the user exits the program, the system shows down entirely as expected. This shows that the functional requirements of the system have been achieved.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Input** | **Expected Output** | **Actual Output** | **Pass/Fail** |
| TC01 | Start parking with valid data | User ID (valid), Hours: 3, License Plate | Parking starts, charge calculated, DB updated | Parking started for 3.0 hours. Charged €2.00. End time: 2024-04-22 03:05:02. Balance updated | Pass |
| TC02 | Start parking with invalid ID | User ID (invalid), Hours: 2, License Plate | Error message, no DB update | User does not exist | Pass |
| TC03 | Exit application | None | Application closes, resources cleaned | Exits and shuts down | Pass |
| TC04 | Negative input for hours | User ID (valid), Hours: -1, License Plate | Error message, no DB update | Invalid parking hours: Parking must be greater than 0 | Pass |

Table 11 Functional test table for parking\_sys.py

As shown in Table 11, the functional requirements for the parking system have not all passed. The user is able to input the hours they are parking and their license plate and scan a valid card and the data is saved, it passes as expected. The user is able to input valid hours and license plate but use an invalid card and it does not save the data, it passes as expected. The user is able to exit the program as expected and it shuts down all the resources and the window as expected. The user is able to input a negative number of hours with a valid card and the system does not allow negative values as expected. All the functional requirements of the system have been met.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Description** | **Input** | **Expected Output** | **Actual Output** | **Pass/Fail** |
| TC01 | Start system | System start command | All modules initialised; system ready | System starts | Pass |
| TC02 | Proper system shutdown | Shutdown command | Clean shutdown, resources released | Exits and shuts down | Pass |

Table 12 Functional test table for main.py

As shown in Table 12, the functional requirements for the main system have all passed. The system is able to start and initialise on startup and when the user clicks exit, the system shuts down all resources and closes the window as expected which meats the functional requirements of the system.

## **Test Conclusion**

This section concludes the testing chapter of the project. During the development of this project, testing has been an important to ensure that requirements of the project have been satisfied. With any software project, it is vital to guarantee that the software is either bug free or has minimal bugs and that the bugs that are present do not affect the overall functionality of the entire project. As well as that it important that developers are able to track bugs with documentation and have clear documentation of their code to demonstrate that they understand the code so that if any errors occur, they are able to pinpoint their code and action as appropriate. This shows that the developers of the project are working in alignment of the best practises of software development.

# **Conclusion**

## **Project Summary**

The system developed provides a proof of concept of integrating a simple card into the everyday life of students. It provides smaller scale system to allow for increased security, streamlined attendance tracking, a centralised student registry system as well as introducing a concept of a cashless society by integrating a paid parking system. If this project were to become a production level application, this would create a highly beneficial system not only for students but also for staff, it would decrease health and safety risks by having a digital record of all students on campus, diverging personnel from having access to areas on campus where should only be limited to a particular group of people, having directly access to a live attendance feed for lecturers. The students now have a streamlined approach to access rooms, sign into class and pay for parking. This project can be expanded and have more functionality added to it as seen fit by the university as there are endless use cases for this system.

## **Findings**

Choosing to develop this system in Python with the chosen hardware has proven itself to be a useful decision. Since the Raspberry Pi is quite a flexible device and many of the hardware dependencies were native to Python, developing the project was not as hard as expected. Most of the code was developed on a Windows machine and was pushed to a GitHub repository to ensure that the code was stored somewhere and then tested on the Raspberry Pi also eased the development of the project.

There are many repositories and documentation online that provide help on how to integrate the hardware to the Raspberry Pi, how to use their Python libraries in a development environment as well as the endless tutorials and forums used to debug any code issues that may have been discovered.

A limitation to this project would have been acquiring a Raspberry Pi of little RAM. This made the development process time consuming as the microcontroller would lag out a bit and take too long to launch programs. In a production level application, a microcontroller that has a higher level of RAM would be much more recommended compared to the one used here. Another limitation would have been the various RFID readers that were used during the development of the project. Between the PN532, ACR122U and RC522, the RC522 was the best choice for the development of this program. Not only did it have much more documentation and proof of application online, but its dedicated library was also the easiest to use compared to the two RFID readers. The library it provided had clear instructions and apart from soldering the pins made it quite easy to connect the chip to the circuit.

One of the main obstacles of this program was understanding how to use a microcontroller like the Raspberry Pi. As it is a Linux based system and it was the first-hand experience with developing solely on Linux, learning how to do commands through its command line was one of the main roadblocks during the project. Since the database was a local database on the machine, learning the Linux commands on how to use MariaDB was something that was not realised during the initial setup of the project.

## **Improvements**

Based on the results of the tests that were done for this application, it would be necessary to implement the missing features. This would ensure that the functional requirements were fully met and that it satisfies the user expectations. In terms of the project, it would be beneficial to make all three systems standalone applications to ensure that they do not need and IDE to run and it does make it more visually appealing to the user. Another improvement to be made would be to use a higher RAM Raspberry Pi to ensure that there is no lag when the system is running. Another improvement that could be made would be to deploy the database and application online so that the entire system could be wireless and could be deployed onto multiple circuits so that it could run a multifunctional application. It would apply to the real-world application where this could function across an entire university campus.

An improvement that could be made would be a top up mobile application built for iOS and Android devices so that users can update their balance on the go and also provide a log of what they have spent their money on and a time stamp to make it clear to the user. The administration system where the user accesses the API or registering the student could be password protected or prompt the user to scan their card and ensure that they are staff so that all the users of the system cannot access it.

**Future Work**

Considering all the improvements that could be made, the main piece of work that would expand this project would be to make a mobile able for the top up system and deploy the database onto a cloud database. That way the system would be able to take real world transactions and apply it to a real-world scenario.

Putting the project into a CI/CD platform and deploying it where it could run automated tests, allow for more cross development with multiple developers and more would allow for the system to enter a production level development environment. If the proposed software were presented to stakeholders or investors, having the software ready on a CI/CD platform it would showcase the capabilities of the software and that it ready for more functional requirements and can be deployed.

## **Final Thoughts**

This project has provided an opportunity to reflect on the process of examining the implementation of a flexible NFC enabled student card system with API integration by incorporating it into the everyday lives of students and academic staff. By meticulously organising, implementing, and analysing throughout the duration of this project lifecycle, it is clear that this project has provided valuable understanding of both its accomplishments and obstacles faced during the development.

Fundamentally, the effort goes beyond being a technological tool exercise; contains an objective to improve the effectiveness, safety and easy of campus life. Using Python programming and Raspberry Pi hardware, the project has effectively displayed the ability to combine multiple features such as access control, attendance tracking, paid parking into one cohesive platform. This strategy not only tackles practical requirements but also focuses on the innovation to be brought to educational settings.

Valuable lessons such as becoming more proficient in Python programming, understanding Linux based systems, learning how to solder from online tutorials and YouTube videos emphasise the importance of learning and acquiring more skills to move forward as a better software developer in the future.

Learning how to implement and build a circuit for the first time as well as learning how to implement these hardware pieces into a software program and provided great insight on how this program can be extended and built into production level software.

In conclusion, the aim and objective that was proposed in the beginning of the paper has been achieved and the main target features of the system have been implemented to address a proof of concept that could be taken to a broader scale and shown to stakeholders. Given the period provided to build and test the system, and the lack of experience for the developer with this kind of approach, it is clear that it has been a successful development process and the core objectives outlined have been reached.

# **List of Abbreviations**

AIESEC HKU - Association Internationale des Étudiants en Sciences Économiques et Commerciales, Hong Kong University

AI - Artificial Intelligence

API - Application Programming Interface

BMW - Bayerische Motoren Werke AG

CI/CD - Continuous Integration/Continuous Delivery

DDoS - Distributed Denial of Service

DELETE - HTTP method used to delete a resource

GPIO - General-Purpose Input/Output

GUI - Graphical User Interface

HCE - Host Card Emulation

HTML - Hypertext Markup Language

HSU - High-Speed UART (Universal Asynchronous Receiver-Transmitter)

I2C - Inter-Integrated Circuit

IEC - International Electrotechnical Commission

ID - Identifier

IoT - Internet of Things

ISO - International Organization for Standardization

IT - Information Technology

JSON - JavaScript Object Notation

LCD - Liquid Crystal Display

MHz - Megahertz

ML - Machine Learning

MQTT - Message Queuing Telemetry Transport

MFA - Multi-Factor Authentication

N/A - Not Applicable

NFC - Near Field Communication

NFCIP - NFC Interface and Protocol

NXP - NXP Semiconductors

OS - Operating System

PI - Principal Investigator or Perimeter Intrusion

POS - Point of Sale

QR - Quick Response

RAM - Random Access Memory

REST - Representational State Transfer

RFID - Radio-Frequency Identification

RPC - Remote Procedure Call

SAM - Secure Access Module

SD - Secure Digital

SDLC - Software Development Life Cycle

SE - Secure Element

SIM - Subscriber Identity Module

SMS - Short Message Service

SOAP - Simple Object Access Protocol

SPI - Serial Peripheral Interface

TFL - Transport for London

TTL - Time To Live or Transistor-Transistor Logic

UART - Universal Asynchronous Receiver-Transmitter

UID – Unique Identifiers

UML - Unified Modelling Language

USB - Universal Serial Bus

# **References**

Ahmad, N. (2024). Testing Methodologies: A Detailed Guide To Software Testing Methodologies. [online] www.lambdatest.com. Available at: https://www.lambdatest.com/learning-hub/testing-methodologies [Accessed 22 Apr. 2024].

AirFocus (2020). *What Is an API? API Definition, Examples, Benefits, Challenges & FAQ*. [online] Airfocus.com. Available at: <https://airfocus.com/glossary/what-is-an-api/> [Accessed 5 Nov. 2023].

Amazon Web Services, Inc. (2023). *What is an API? - Application Programming Interface Explained - AWS*. [online] Amazon Web Services, Inc. Available at: <https://aws.amazon.com/what-is/api/> [Accessed 5 Nov. 2023].

Badra, M. and Badra, R.B. (2016). A Lightweight Security Protocol for NFCbased Mobile Payments. In: *Procedia Computer Science*. [online] Elsevier, pp.705–711. doi:<https://doi.org/10.1016/j.procs.2016.04.156>.

Bauer, R. (2017). Understanding Programs, Processes, and Threads - Roderick Bauer - Medium. [online] Medium. Available at: https://medium.com/@rodbauer/understanding-programs-processes-and-threads-fd9fdede4d88 [Accessed 20 Apr. 2024].

Birbeck, University of London (n.d.). *Oyster -Smart Card Payment Systems on the London Public Transport Network*. [online] Available at: <https://titan.dcs.bbk.ac.uk/lo/fi/learningresources/session2/fit/oyster.pdf> [Accessed 2023].

Brophy, M. and King, M. (2023). *22 Contactless Payment Statistics for 2023*. [online] Fit Small Business. Available at: <https://fitsmallbusiness.com/contactless-payment-statistics/> [Accessed 14 Oct. 2023].

Carrilho, R. (2022). *5 Reasons Why NFC Technology Will Thrive In Future Events*. [online] beamian. Available at: <https://beamian.com/why-nfc-technology-will-thrive-in-future-events/> [Accessed 6 Nov. 2023].

cj&co (2023). *NFC Meaning: How This Tiny Tech Is Revolutionizing Our World | CJ&CO*. [online] CJ&COTM. Available at: <https://www.cjco.com.au/article/nfc-meaning/> [Accessed 28 Oct. 2023].

Curoe, M. (2021). *5 Common API Integration Challenges And How To Overcome Them.* [online] Redwoodlogistics.com. Available at: <https://www.redwoodlogistics.com/insights/what-are-some-of-the-barriers-facing-api-integration> [Accessed 5 Nov. 2023].

Cvrček, T. and Dzurenda, P. (2022). Access Control System Using Multifactor Authentication. In: *Proceedings II of the Conference Student EEICT*. Brno University of Technology, pp.190–193. doi:<https://doi.org/10.13164/eeict.2022.190>.

Cvrček, T., Dzurenda, P., Mansur, K., Hasanuddin, Zulfajri B, Pireva, Krenare R, Siqeca, J., Berisha, S., Francis, L., Hancke, G., Mayes, K., Markantonakis, K., Birbeck, ISO/IEC, Rahul, A., G, Gokul Krishnan, H, Unni Krishnan, Rao, S., Prohel, G., Tabuk, University of and and, E. (2020). ‘Digital era’: Impact on the economy and the education system (country analysis). *Utopia y Praxis Latinoamericana*, [online] 25(Extra10), p.170937053109325113352137190-193-140-52-144-60-46-120-11-711-93-186. doi:<https://doi.org/10.5281/zenodo.4155437>.

Dixon, J. and Abdel-shakour Abuzneid (2020). An NFC Based Student Attendance Tracking/Monitoring System Using an IoT Approach. [online] doi:<https://doi.org/10.1109/csci51800.2020.00201>.

Francis, L., Hancke, G., Mayes, K. and Markantonakis, K. (2012). *Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones*. [online] Available at: <https://eprint.iacr.org/2011/618.pdf>.

Frye, M.-K. and MuleSoft (2023). *What is an API? (Application Programming Interface) | MuleSoft*. [online] MuleSoft. Available at: <https://www.mulesoft.com/resources/api/what-is-an-api> [Accessed 5 Nov. 2023].

Future Market Insights (2022). *Near Field Communication (NFC) Market by Product Type, Application & Region - Forecast 2022 to 2032*. [online] Futuremarketinsights.com. Available at: [https://www.futuremarketinsights.com/reports/near-field-communication-nfc-market.](https://www.futuremarketinsights.com/reports/near-field-communication-nfc-market#:~:text=Near%20Field%20Communication%20(NFC)%20Market%20Overview%20(2022%2D2032)&text=The%20sales%20of%20NFC%20are,automotive%20and%20contactless%20payment%20sectors.) [Accessed 5 Nov. 2023].

Gapsalamov, A.R., Bochkareva, T.N., Akhmetshin, E.M. and Vasilev, V.L. (2020). ‘Digital Era’: Impact on the Economy and the Education System (Country Analysis): ‘Era digital’: impacto en la economía y el sistema educativo (análisis de país).. *Utopia y Praxis Latinoamericana*, [online] 25, pp.170–189. doi:<https://doi.org/10.5281/zenodo.4155437>.

GeeksforGeeks. Available at: https://www.geeksforgeeks.org/introduction-to-exploratory-style-of-software-development/ [Accessed 18 Mar. 2024].

GeeksforGeeks (2017). What is Software Testing? [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/software-testing-basics/ [Accessed 21 Apr. 2024].

GfG (2023). *Introduction to Exploratory Style of Software Development*. [online]

Global Payments Integrated (2014). *The History of Contactless Payments*. [online] Globalpaymentsintegrated.com. Available at: <https://www.globalpaymentsintegrated.com/en-us/blog/2020/09/15/the-history-of-contactless-payments> [Accessed 15 Oct. 2023].

Hamilton, T. (2024a). *Difference Between Verification and Validation with Example*. [online] Guru99.com. Available at: https://www.guru99.com/verification-v-s-validation-in-a-software-testing.html [Accessed 22 Apr. 2024].

Hamilton, T. (2024b). Unit Testing Tutorial: What is, Types, Tools, EXAMPLE. [online] Guru99.com. Available at: https://www.guru99.com/unit-testing-guide.html [Accessed 22 Apr. 2024].

Hendry, M. (2014a). Modes of operation. In: *Near Field Communications Technology and Applications*. Cambridge University Press, pp.31–46. doi:<https://doi.org/10.1017/cbo9781107446854.006>.

Hendry, M. (2014b). Security. In: *Near Field Communications Technology and Applications*. Cambridge University Press, pp.109–120. doi:<https://doi.org/10.1017/cbo9781107446854.012>.

Hendry, M. (2014c). What is NFC? In: *Near Field Communications Technology and Applications*. Cambridge University Press, pp.3–11. doi:<https://doi.org/10.1017/cbo9781107446854.002>.

HST Software Solutions (2021). *Tokenization of Cards - What it is, how it works and the main benefits -*. [online] HST Software Solutions. Available at: <https://www.hst.com.br/tokenization-of-cards-what-it-is-how-it-works-and-the-main-benefits/> [Accessed 28 Oct. 2023].

Hussein al ofeishat (2012). *Near Field Communication ( NFC )*. [online] ResearchGate. Available at: <https://www.researchgate.net/publication/316543104_Near_Field_Communication_NFC> [Accessed 14 Oct. 2023].

IONOS editorial team (2023). *Near-Field Communication (NFC)*. [online] IONOS Digital Guide. Available at: <https://www.ionos.com/digitalguide/server/know-how/nfc-near-field-communication/> [Accessed 24 Oct. 2023].

ISO (2023). *ISO/IEC 14443-1:2018*. [online] Iso.org. Available at: <https://www.iso.org/obp/ui/en/#iso:std:iso-iec:14443:-1:ed-4:v1:en> [Accessed 25 Oct. 2023].

ISO/IEC (2013). *Information technology — Telecommunications and information exchange between systems — Near Field Communication — Interface and Protocol (NFCIP-1)*. [online] *standards.iso.org*, Switzerland: ISO, p.52. Available at: <https://standards.iso.org/ittf/PubliclyAvailableStandards/index.html> [Accessed 25 Oct. 2023].

Jaffery, A. (2022). *5 API Trends to Look Forward to in 2023*. [online] Astera. Available at: <https://www.astera.com/type/blog/5-api-trends-to-look-forward-to-in-2023/.> [Accessed 6 Nov. 2023].

Jagaad Software House (2023). *5 Challenges in API Integration and How to solve or avoid them | JAGAAD Digital Solutions*. [online] Medium. Available at: <https://medium.com/jagaad/overcoming-5-common-challenges-when-integrating-apis-e1e10d865d22> [Accessed 5 Nov. 2023].

Kaspersky (2020). *Secure Element*. [online] Kaspersky.com. Available at: <https://encyclopedia.kaspersky.com/glossary/secure-element/> [Accessed 25 Oct. 2023].

Katalon (2022). What is functional testing? Definition, types & examples. [online] katalon.com. Available at: https://katalon.com/resources-center/blog/functional-testing [Accessed 22 Apr. 2024].

Kolya Hnatyuk (2023). *130+ API Statistics: Usage, Growth & Security*. [online] MarketSplash. Available at: <https://marketsplash.com/api-statistics/>[Accessed 14 Oct. 2023].

Kong Inc (2023). *What is an API? Use Cases and Benefits*. [online] Kong Inc. Available at: <https://konghq.com/learning-center/api-management/what-is-api> [Accessed 5 Nov. 2023].

Landt, J. (2005). *IEEE Xplore Full-Text PDF:* [online] Ieee.org. Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1549751> [Accessed 14 Oct. 2023].

Linkedin.com. (2023). *What are the latest trends and innovations in NFC technology and how can you keep up with them?* [online] Available at: <https://www.linkedin.com/advice/1/what-latest-trends-innovations-nfc> [Accessed 6 Nov. 2023].

Lung, N. (2015). *Does NFC make some smartphones smarter than others?* [online] TecHKU. Available at: <https://www.engineering.hku.hk/tecHKU/2015/02/09/does-nfc-make-some-smartphones-smarter-than-others/> [Accessed 5 Nov. 2023].

Mansur, K. and Hasanuddin, Zulfajri B (2018). *Implementation of NFC for Smart Gate Access Control in Campus Area*.

Marcus, K. (2023). *Five Common API Integration Issues and How to Avoid Them - Spiceworks*. [online] Spiceworks. Available at: <https://www.spiceworks.com/tech/devops/guest-article/avoiding-apiintegration-issues/> [Accessed 5 Nov. 2023].

Marforio, C., Ritzdorf, H., Francillon, A. and Capkun, S. (2012). Analysis of the communication between colluding applications on modern smartphones. In: *Proceedings of the 28th Annual Computer Security Applications Conference*. [online] New York, NY, USA: ACM, pp.51–60. doi:<https://doi.org/10.1145/2420950.2420958>.

Microsoft Devices Blog and Microsoft Devices Team (2012). *Nokia’s NFC phone history*. [online] Microsoft Devices Blog. Available at: <https://blogs.windows.com/devices/2012/04/11/nokias-nfc-phone-history/> [Accessed 15 Oct. 2023].

Moore, J. (2023). *How have other countries managed contactless bus payments while Ireland has not?* [online] TheJournal.ie. Available at: <https://www.thejournal.ie/contactless-bus-payments-public-transport-ireland-6077000-May2023/> [Accessed 25 Oct. 2023].

NFC Forum (2019). *Fresh SmartPhone Statistics and What They Mean For You, NFC And The World*. [online] Nfc-forum.org. Available at: <https://nfc-forum.org/news/2019-12-fresh-smartphone-statistics-and-what-they-mean-for-you-nfc-and-the-world/> [Accessed 14 Oct. 2023].

Nguyen, S. (2022). *11 API Trends to Watch for in 2023 - DreamFactory Software- Blog*. [online] DreamFactory Software- Blog - API Management, Enterprise Integrations, Data Security and More. Available at: <https://blog.dreamfactory.com/11-api-trends-to-watch-for-in-2022/> [Accessed 6 Nov. 2023].

Nguyen, S. (2023). *The Top Four Challenges With API Development - DreamFactory Software- Blog*. [online] DreamFactory Software- Blog - API Management, Enterprise Integrations, Data Security and More. Available at: <https://blog.dreamfactory.com/the-top-four-challenges-with-api-development/> [Accessed 5 Nov. 2023].

Organ, P. (2024). *The Hidden Costs: How Manual Attendance Tracking Damages Schools and Disrupts Parent Engagement | Orah Blog*. [online] Orah.com. Available at: <https://www.orah.com/blog/hidden-costs-of-manual-attendance> [Accessed 11 Mar. 2024].

Paragon ID (2017). *The history of NFC | Paragon RFID*. [online] Paragon-rfid.com. Available at: <https://www.paragon-rfid.com/en/the-history-of-nfc/> [Accessed 14 Oct. 2023].

Person, C. (2022). *The ChameleonMini is a skeleton key for RFID*. [online] The Verge. Available at: <https://www.theverge.com/23411372/chameleon-mini-rfid-nfc-attack-proxmark3-keyless-card-reader> [Accessed 28 Oct. 2023].

Pireva, Krenare R, Siqeca, J. and Berisha, S. (2013). RFID: Management system for students’ attendance. In: *IFAC Proceedings Volumes (IFACPapersOnline)*. IFAC Secretariat, pp.137–140. doi:<https://doi.org/10.3182/201306063XK4037.00057>.

Pireva, Krenare R, Siqeca, J., Berisha, S., Francis, L., Hancke, G., Mayes, K. and Markantonakis, K. (2012). *Practical Relay Attack on Contactless Transactions by Using NFC Mobile Phones*. [online] *IFAC Proceedings Volumes (IFACPapersOnline)*, IFAC Secretariat, pp.137–140. doi:<https://doi.org/10.3182/201306063XK4037.00057>.

PR Newswire (2023). *Near Field Communication (NFC) market size to grow by USD 33.1 billion from 2023 to 2030; Humungous preference for cashless payments to augment the global market trends - Facts & Factor*. [online] Yahoo Finance. Available at: <https://finance.yahoo.com/news/near-field-communication-nfc-market-213000434.html?guccounter=1> [Accessed 5 Nov. 2023].

profylecard (2021). *A history of NFC technology. — Profyle Card*. [online] Profylecard.com. Available at: [https://www.profylecard.com/a-history-of-nfc-technology/.](https://www.profylecard.com/a-history-of-nfc-technology/#:~:text=Near%20Field%20Communication%20(NFC)%20technology,transfer%20music%20files%20between%20devices.) [Accessed 14 Oct. 2023].

Prohel, G. (2013). *STMicroelectronics Introduction to NFC*. [online] Available at: <https://www.mouser.com/pdfdocs/STMicroelectronicsIntroductiontoNFC.pdf>.

Purdue News Service (2023). *Purdue launches Purdue Mobile ID for students, allowing them to get around campus with just a simple tap of their mobile device*. [online] Purdue.edu. Available at: <https://www.purdue.edu/newsroom/releases/2023/Q2/purdue-launches-purdue-mobile-id-for-students-allowing-them-to-get-around-campus-with-just-a-simple-tap-of-their-mobile-device.html> [Accessed 5 Nov. 2023].

Rahul, A., G, G.K., H, U.K. and Rao, S. (2015). Near Field Communication (NFC) Technology: A Survey. *International Journal on Cybernetics & Informatics*, 4(2), pp.133–144. doi:<https://doi.org/10.5121/ijci.2015.4213>.

Reach Researcher (2023). *LinkedIn*. [online] Linkedin.com. Available at: <https://www.linkedin.com/pulse/near-field-communication-nfc-market-size-2023-2030/> [Accessed 5 Nov. 2023].

Research Union (2023). *Decoding the Mobile Access Control Platform Market: A Deep Dive into the Latest Market Trends, Market Segmentation, and Competitive Analysis*. [online] Linkedin.com. Available at: <https://www.linkedin.com/pulse/decoding-mobile-access-control-platform-market-deep-dive/> [Accessed 25 Oct. 2023].

Reva Rikat Asih, Tania Izza Sholikhah, Nabila Dhiya Ulhaq, Rahman, T., Mil'atuttoyyibah, Moechammad Sarosa and Putri Elfa Mas’udia (2022). NFC (Near Field Communication)-Based Canteen Self Service Application at SMA Pomosda Nganjuk. [online] doi:<https://doi.org/10.1109/ieit56384.2022.9967855>.

Roberti, M. and Violino, B. (2005). *The History of RFID Technology*. [online] RFID JOURNAL. Available at: [https://www.rfidjournal.com/the-history-of-rfid-technology.](https://www.rfidjournal.com/the-history-of-rfid-technology#:~:text=The%20Germans%20discovered%20that%20if,the%20first%20passive%20RFID%20system).) [Accessed 11 Mar. 2024].

Sametinger, J. and Stritzinger, A. (1996). *Exploratory Software Development with Class Libraries*.

Sims, G. (2014). *Everything you need to know about Host Card Emulation*. [online] Android Authority. Available at: <https://www.androidauthority.com/everything-need-know-host-card-emulation-347626/> [Accessed 25 Oct. 2023].

Solutions, G. (2023). Functional and Non-Functional Requirements: The Ultimate Checklist with Examples. [online] Medium. Available at: https://medium.com/@growsolutions/functional-and-non-functional-requirements-the-ultimate-checklist-with-examples-cde16aba33d7 [Accessed 22 Apr. 2024].

Sony (2023). *Sony Corporation - FeliCa - Overview of FeliCa - What is FeliCa ?* [online] Sony.net. Available at: <https://www.sony.net/Products/felica/about/> [Accessed 28 Oct. 2023].

Stanford.edu. (2023). *NFC Technology*. [online] Available at: <https://cs.stanford.edu/people/eroberts/cs201/projects/2010-11/NFCChips/rfid.html> [Accessed 14 Oct. 2023].

Statista (2019). *Global proximity mobile payment users 2023 | Statista*. [online] Statista. Available at: <https://www.statista.com/statistics/557959/global-mobile-proximity-payment-users/> [Accessed 25 Oct. 2023].

Statista (2020). *UK: contactless card issuance 2007-2020 | Statista*. [online] Statista. Available at: [https://www.statista.com/statistics/859911/contactless-cards-in-issuance-in-the-uk/](https://www.statista.com/statistics/859911/contactless-cards-in-issuance-in-the-uk/#:~:text=Since%20contactless%20cards%20were%20first,debit%20cards%20also%20became%20available.) [Accessed 15 Oct. 2023].

Statista (2023). *E-commerce payment methods market share 2026 | Statista*. [online] Statista. Available at: <https://www.statista.com/statistics/1111233/payment-method-usage-transaction-volume-share-worldwide/> [Accessed 25 Oct. 2023].

STMicroelectronics (2022a). *What is an NFC Chip? - STMicroelectronics*. [online] STMicroelectronics. Available at: <https://www.st.com/content/st_com/en/support/learning/essentials-and-insights/connectivity/nfc/nfc-chips.html> [Accessed 25 Oct. 2023].

STMicroelectronics (2022b). *What is an NFC Chip? - STMicroelectronics*. [online] STMicroelectronics. Available at: <https://www.st.com/content/st_com/en/support/learning/essentials-and-insights/connectivity/nfc/nfc-chips.html> [Accessed 25 Oct. 2023].

Stockman, H. (1948). *IEEE Xplore Full-Text PDF:* [online] Ieee.org. Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1697527> [Accessed 14 Oct. 2023].

The Insight Partners (2020). *Mobile Access Control Platform Market to Grow at a CAGR of 27.1% to reach US$ 782.01 million from 2020 to 2027*. [online] The Insight Partners. Available at: <https://www.theinsightpartners.com/reports/mobile-access-control-platform-market> [Accessed 25 Oct. 2023].

Transact (2023). *Campus ID Solutions*. [online] Transact Campus. Available at: <https://transactcampus.com/solutions/campus-id> [Accessed 1 Nov. 2023].

University of Tabuk and and, E. (2020). *2020 International Conference on Computing and Information Technology (ICCIT1441).* [online] Available at: <https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9213758>.

Waldman, A. (2021). *Researchers hack Apple Pay, Visa ‘Express Transit’ mode*. [online] TechTarget. Available at: <https://www.techtarget.com/searchsecurity/news/252507530/Researchers-hack-Apple-Pay-Visa-Express-Transit-mode> [Accessed 28 Oct. 2023].

Zhou, M., Chen, N., Yang, Y., Pan, C. and Zhou, X. (2022). Intelligent Access Control System. *2022 7th International Conference on Communication, Image and Signal Processing (CCISP)*. [online] doi:<https://doi.org/10.1109/ccisp55629.2022.9974224>.

Zimmerman, E. (2018). *Near-Field Communication Technology Supports Contactless Student ID at Universities*. [online] Technology Solutions That Drive Education. Available at: <https://edtechmagazine.com/higher/article/2018/12/near-field-communication-technology-supports-contactless-student-id-universities-perfcon> [Accessed 1 Nov. 2023].

# **Appendix**

## **Appendix A – Requirements Analysis Document**

**Name: Raissa Pululu**

**ID: K00206175**

**Instruction:** Add information regarding your project under **each** of the RED headings below.

**Name of your project:**

The name of this project is An NFC Enabled Student Card System with API Integration

**Description of business problem being solved/purpose of the system:**

The purpose of this system is to design, build and test a Python based system functioning off a Raspberry Pi with a circuit based on an RFID reader module. The system aims to provide a proof of concept or smaller scale system that streamlines access control, attendance tracking, user registration, parking and cashless transactions all using an NFC enabled card or tag.

A fully developed system, built for handling actual educational institutions do exist and they do provide prebuilt solutions that just have to be implemented into the infrastructure, but this project provides a way to show that this system can be built in house, it would not be a huge cost to build it on a larger scale. The system is a three-subsystem project, which each sub system offering different functionalities. The first subsystem allows the user to do access control functionality, register a new student or staff member, track attendance as well as access the API, the second subsystem, allows the user to top up the balance of their card, and the third subsystem allows the user to use their balance to pay for parking. The system is connected to a local database stored on the Raspberry Pi, have multiple users registered to be able to use the system, allow them to use the system as authorised.

**Classes of ALL System Users**

When registering a user, you must predefine the department that they are joining. There are general users, but they are of different departments.

* **User**
  + **IT**
  + **Beauty**
  + **OP**
  + **Sci**
  + **Culinary**
  + **Eng**
  + **Staff**

**Numbered/Ordered List ALL Main User Functions**(for each user type):

**All users despite assigned department:**

1. **Start API Server and Open Browser** | Button | General user on main.py | Launches the API and takes the user to the browser to display JSON of attendance of the last hour
2. **Select Department** | Dropdown menu | General user on main.py | Displays dropdown for creating a new user or for access control
3. **Enter Name** | Text box | General user on main.py | Takes user input for registering a new staff or student by setting the name
4. **Unlock Door** | Button | General user on main.py | Takes user input unlocks a set room based on the department set and the department of the department from the scanned NFC card
5. **Register User** | Button | General user on main.py | Takes user input create a new student or staff based on the inputs provided from Select Department and Enter Name
6. **Attendance** | Button | General user on main.py | Takes user input to sign in for attendance
7. **Enter Parking Hours** | Text box | General user on parking\_sys.py | Takes user input for setting number of hours for parking
8. **Enter License Plate** | Text box | General user on parking\_sys.py | Takes user input for setting license plate for parking
9. **Start Parking Session** | Button | General user on parking\_sys.py | Takes user input to calculate parking and provide price
10. **Enter Amount to Update Balance** | Text box | General user on top\_up.py | Takes user input for setting amount to update balance
11. **Update Balance** | Button | General user on top\_up.py | Takes user input to update balance

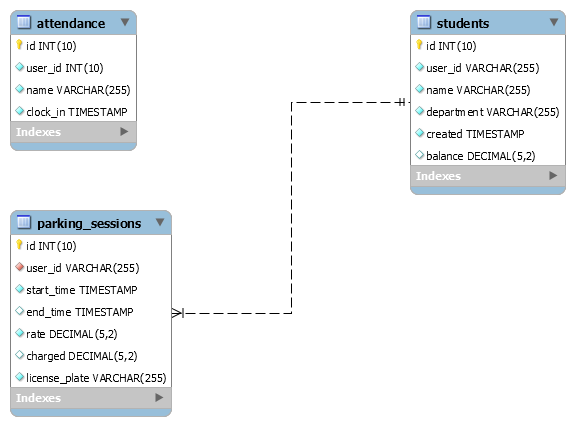
**Logical Architecture**A screenshot of a computer screen

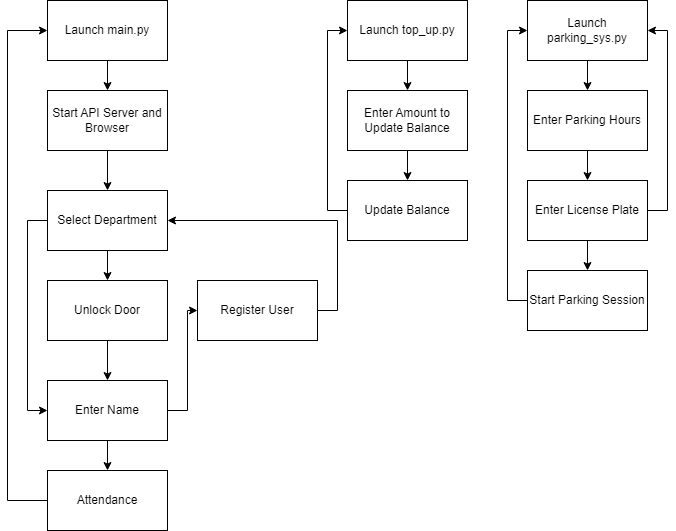
Description automatically generated

**Preliminary Database design: Provide Details of your initial database design**

**Database Name:** nfcstudentsys

**Database EER Model**



**Page Navigation**

## **Appendix B – Declaration of Authorship**

**BSc in Software Development – Declaration of Authorship**

I, Raissa Pululu, declare that the software development project titled An NFC Enabled Student Card System with API Integration submitted by me as part of my BSc in Software Development is entirely my own work, and I have not used unauthorised external assistance in its creation. I affirm that:

I have fully and independently developed the software code, design, and documentation contained within this project.

Any external sources, such as code snippets, libraries, or third-party resources, have been appropriately cited and referenced in accordance with academic citation standards.

This project is a result of my own creative and intellectual efforts, and I have not copied, borrowed, or utilised work produced by others without proper acknowledgment.

I have not used automated content generation tools, including AI-based code or text generation services, in the creation of this project.

Any collaborations or contributions from external individuals are acknowledged and appropriately credited in the project documentation.

The code and documentation presented here reflect my understanding of the concepts and principles learned during the course.

I understand that any breach of academic integrity, including plagiarism or submitting work that is not my own, is subject to disciplinary actions as outlined in the TUS academic policies.

[Signature] Raissa Pululu

[Date] 1/5/2024