BIOMETRIC BADGING SYSTEM

A dissertation submitted in partial fulfilment of

the requirements for the degree

of

BACHELOR OF *ENGINEERING* in 'Software

The Queen's University of Belfast

BY: IND

8TH MAY 2017

SCHOOL OF ELECTRONICS, ELECTRICAL ENGINEERING and COMPUTER SCIENCE

CSC3032 – SOFTWARE ENGINEERING PROJECT

Team Dissertation Cover Sheet

A signed and completed cover sheet must accompany the submission of the Software Engineering dissertation submitted for assessment.
Work submitted without a cover sheet will **NOT** be marked.

Team Name: Supervisor:

Project Title:

Team Members Names: Team Members Student Number:

Declaration of Academic Integrity

Before signing the declaration below please check that the submission:

1. Has a full bibliography attached laid out according to the guidelines specified in the Student Project Handbook
2. Contains full acknowledgement of all secondary sources used (paper-based and electronic)
3. Does not exceed the specified page limit
4. Is clearly presented and proof-read
5. Is submitted on, or before, the specified or agreed due date. Late submissions will only be accepted in exceptional circumstances or where a deferment has been granted in advance.

We declare that we have read both the University and the School of Electronics, Electrical Engineering and Computer Science guidelines on plagiarism - http://www.qub.ac.uk/schools/eeecs/Education/StudentStudyInformation/Plagiarism/ - and that the attached submission is our own original work. No part of it has been submitted for any other assignment and we have acknowledged in our notes and bibliography all written and electronic sources used.

Each Team Members signature *Date of submission*

Acknowledgements

The team would like to say thanks to Aaron Fulton and Neil Hamilton from Citi for all their advice during this project. We would also like to pay thanks to Fabian Campbell and the team at Liopa for meeting with us and letting us hear about their biometric product providing an insight to the market for a biometric system, and for the advice that really helped our product take shape. The team would also like to than Futronic-Tech for providing the API's for communicating with their fingerprint device, and SourceAFIS software library that provided the team will the necessary tools to produce a Biometric Fingerprinting Badging System.

Team

Team Name: InD

Team Members:

- Adam Farren
- Fergal O'Neill
- James Dickinson
- Lauren McGlennon

Abstract

This dissertation is concerned with the need for a biometric system to control access to a company. To eliminate the problems, current too the access control measures already in place within a company, in providing a system of high quality with convenience and security in mind.

This project focuses on the development of a biometric fingerprinting system.

The system we propose will be used to create a biometric badge for all current employees, and through the hiring process new employees and guests will be enrolled. The system will require a user to place their fingerprint on an available fingerprint scanner at points of interest in a company, (i.e. entry), that will then determine a user's access by searching a database for a biometric match.

Table of Contents

Problem Specification

Overview of Problem

A problem to any company and its employees is the ways and means of which it controls access to their buildings and certain areas with specific access levels. Majority of companies have an implemented control system that can make access difficult to a building or area for someone who does not hold the relevant means to enter. Presently this problem is widely solved in several ways through knowledge access control in the form of a pin or password, or using object access control which can take the form of a smart card or badge, but not limited to.

With current implemented solutions, it relies on an employee's need to hold with them a token or the knowledge to grant access to a building and within their privileges access to certain areas of the building, with its benefits to a company it is not without its problems. Knowledge access control such as pins or passwords allows ease of use, the ability to change frequently as a security measure if password has been compromised, pins or passwords can be forgotten, shared unintentionally, or hacked, this method of access control is not limited to a certain person and can pose a risk to the company's security.

Object control access such as access cards are unique to an employee in that no two cards are the same, they are convenient to use without the need to enter a pin or passwords providing a level of security to the employee and the company over the use of knowledge based solutions. The disadvantages to this is that access cards are a repeated cost to a company with the long-term contribution of issuing smart cards with consideration to newly enrolled employees, employees who have lost, stolen or damaged existing access cards. Access cards are susceptible to being forgotten, lost, and stolen. For an employee who has forgotten their access card on a day will have trouble with their access to the company, measures will be in place to confirm identification prior to access which can prove to be timely and inconvenient event. A lost or stolen access card if not reported increases the chances of a security breach to the company.

The solutions currently in place for access control that aim to provide security and convenience are in fact a source of vulnerability and expense if acquired by someone other than the intended employee.

Proposed Solution

We have purposed a biometric badging system to address the problem at hand, that eliminates an individual's need to carry with them a physical token, pin, or password for access controlled building entry and exit and or specific areas within, providing to any company a high-quality system that controls access with convenience and security in mind. Biometrics is defined as the use of a unique physical characteristics to identify individuals, we propose to use an individual's fingerprint marker as its considered the oldest and most widely used form of Biometric which is unique to the individual and immutable over an individual's lifetime.

The system we propose will be used to create a biometric badge for all current employees, and through the hiring process new employees and guests will be enrolled. The system will require a user to place their fingerprint on an available fingerprint scanner at points of interest in a company, (i.e. entry/exit), that will then determine a user's access by searching a database for a biometric match.

Requirements

1. As an employee, I can request my biometric data with the system.
2. As an employee, I can access the building with my fingerprint scan.
3. As an employee to the system I can be denied access to the building with my fingerprint scan.
4. As a manager, I can oversee employee's registration.
5. As a manager, I can search all registered employees.
6. As a manager, I can monitor employee movements.
7. As a manager, I can monitor footfall for areas within company.
8. As a manager, I can set employee access levels.
9. As a manager, I can revoke an employee's access.
10. As a manager, I can remove an employee's record.

Competition

Using a biometric marker to identify and grant access to buildings and secure areas for an individual is an area that very few companies are offering a solution to. Within our research there are a few companies with implemented biometric solutions, these vary for different uses other than security such as a fingerprint clock-in and -out system for payroll tracking

and canteen wallet accounts. The use of Biometrics in security is limited and with our proposal of a biometric fingerprint badging system we aim to fill this gap in the market by offering a secure and reliable system that will remove fears of lost or stolen access cards or of compromised security codes in access control.

Goal

The goals of this project are to replace the need for a physical token, pin, or password to gain access to a building by replacing with a biometric fingerprint scanner. By doing this we will increase security for companies to ensure that only the people with registered fingerprints can access buildings and secure areas within the limits of their access privileges. It will be more secure as every fingerprint will be unique to a single employee and will prevent employees from sharing pins and passwords and even their physical token with other people to allow them access and prevent people getting a hold of the pins, passwords, or badges by other means, such as theft. It will solve the problem of people needing to carry with them a physical badge on them always and be more convenient. The Biometrics solution will also stop the inconvenience for when an employee forgets or loses their badge thus eliminating the time which is then lost for the employee and company in trying to replace or issue a temporary badge for that day.

Constraints

Some of the constraints of this project are the hardware and software platforms to which the system will run on. We will need a server, fingerprint scanner as the client and a user interface to allow the employer to interact and change access levels etc. The performance and efficiency is a main constraint, we need to implement a fingerprint scanner(s) and system that utilises through communication ensuring a fast and seamless solution as possible, as we cannot run the risk of the system having people standing queueing, all whilst waiting for their fingerprint marker to be scanned and verified. We also need to ensure concurrency between scanners that all can work with the system, without the worry of multiple fingerprints being processed at a time with the risk of the system crashing.

Cost

The cost of a biometric security system depends on multiple factors that the client will have to consider. The proposed Biometric fingerprint badging system needs to consider in the cost to the client, the hardware needed to deliver the proposed solution. Fingerprint scanners are very reliable, and vary slightly between brands, ranging upwards of £50. Acquiring a fingerprint scanner consideration to the cost and the features it provides need to be consider, has more advance fingerprint scanners offer a wild range of features that further the security and quality of images acquire, a much sought after feature is a live finger detection which eliminates the possibility of fake fingerprints being used to gain access. The cost in developing the SDK (Software Development Kit) and adapting hardware to a system is very expensive, the client will have to consider what functionality they require and a costing quote for said functionality will need to be considered, a typical SDK can value at around £300. The cost for installation and implementation of hardware and software can to be very costly, with consideration to system maintenance, future updates to be considered. Although a biometric security system can provide a better level of security than other systems one major disadvantage is the cost. This issue comes down to how much the client is willing to spend without sacrificing the reliability of the system. The aim of the purposed system will be to offer to any client different choices of packages, which can range from a basic package offering the bare minimum to introduce a biometric badging system to their company or a premium product offering a more secure and functional system.

Programs to be adapted

The biometric system could potentially be adapted and integrated into other systems that are already in place, for example if the client's building has a smart card system in place to let employees buy food or drink then it could be possible to integrate the system to allow employees to purchase items using their biometric tag instead of a smart card. Another example would be allowing employees to enter the company car park using biometrics, this would increase security as only employees would only be able to access the car park. Another way in which the system could be adapted or integrated is through attendance tracking, this would allow managers to see when and where their employees are signing in or out.

Development Strategy

The System is to be designed using an Agile approach, this approach will help to schedule the development process as well as provide rapid changes to the specification or software when necessary. The team has decided on the Agile method as it allows easy scheduling of key milestones as well as flexibility if the specification needs changed.
The first semester will focus on research and designing of the system. A series of sprints have been organised for the second semester to begin development of the system. These sprints will consist of one week of coding, followed by two days of testing and finally three days reviewing the sprint and bug fixes. The sprint plan be a Gantt chart in figure 1 on the following page.

Each sprint will focus specific parts of the project, each sprint is described as follows:

Sprint 1

- App/WinForms Wireframes
- FingerprintFetch research

Sprint 2

- Server setup
- Database table creation
- Basic MySQL commands and validation
- Terminal creation
- FingerprintFetch creation
- Fingerprint matching research and creation

Sprint 3

- WinForms MySQL commands complete
- Validation of code in WinForms
- Integration of Fingerprint Fetch into Terminal
- Integration of Fingerprint Fetch into WinForms for Employee Registration
- Android app development

Sprint 4

- Integration of Fingerprint matching into Terminal
- Creation of printable reports in WinForms
- Validation of code

Sprint 5

- Bug fixes
- Code clean-up

Gantt Chart

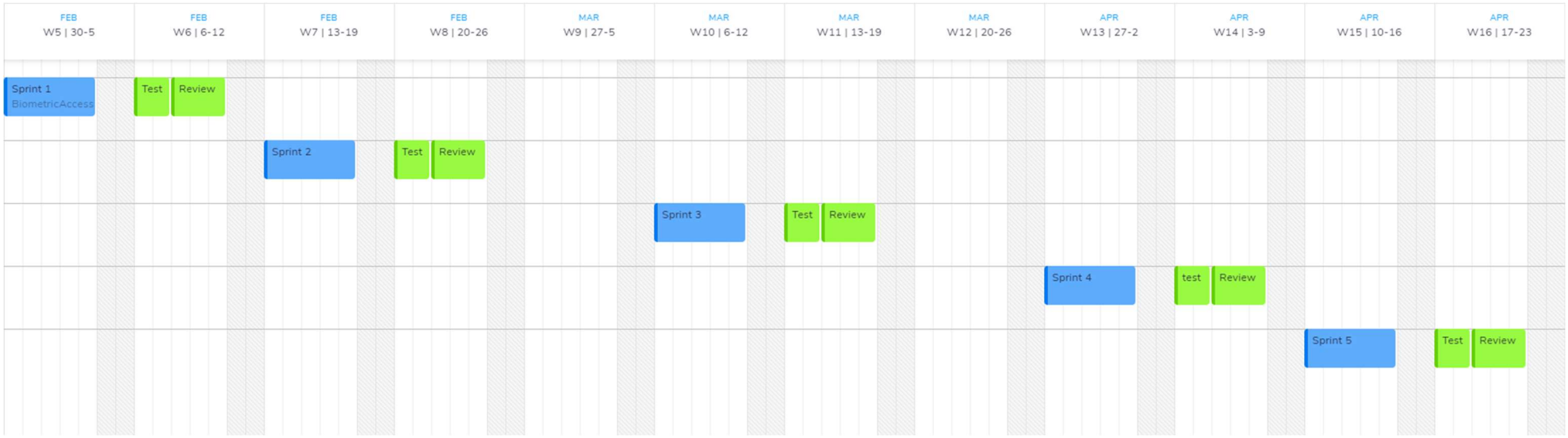


Figure 1: Gantt Chart of spirit plan

The Business Model Canvas

Designed for:

Designed by:

On:

Iteration:

Key Partners

Who are our Key Partners?
Who are our key suppliers?
Which Key Resources are we acquiring from partners?
Which Key Activities do partners perform?

End Users
Security Companies
Suppliers, Biometric sensor company.

Key Activities

What Key Activities do our Value Propositions require?
Our Distribution Channels?
Customer Relationships?
Revenue streams?

Fingerprint acquisition
Enrolment
Matching
Maintenance
Storing Data

Key Resources

What Key Resources do our Value Propositions require?
Our Distribution Channels? Customer Relationships?
Revenue Streams?

Physical hardware
Intellectual software, data
Humans – security experts
Financial, start-up costs

Value Propositions

What value do we deliver to the customer?
Which one of our customer's problems are we helping to solve?
What bundles of products and services are we offering to each Customer Segment?
Which customer needs are we satisfying?

Data
Security
Efficiency
Convenience
High-Quality

Customer Relationships

What type of relationship does each of our Customer Segments expect us to establish and maintain with them?
Which ones have we established?
How are they integrated with the rest of our business model?
How costly are they?

Training
Ease of use
Maintenance
Self-sustainability
Expandable

Channels

Through which Channels do our Customer Segments want to be reached?
How are we reaching them now?
How are our Channels integrated?
Which ones work best?
Which ones are most cost-efficient?
How are we integrating them with customer routines?

Awareness – Social Media
Incentives
Evaluation -feedback on the product

Customer Segments

For whom are we creating value?
Who are our most important customers?

Airports
Banks

Large Companies with access control

Small secure companies that would benefit from a biometric system

Engage with competitors

Cost Structure

What are the most important costs inherent in our business model?
Which Key Resources are most expensive?
Which Key Activities are most expensive?

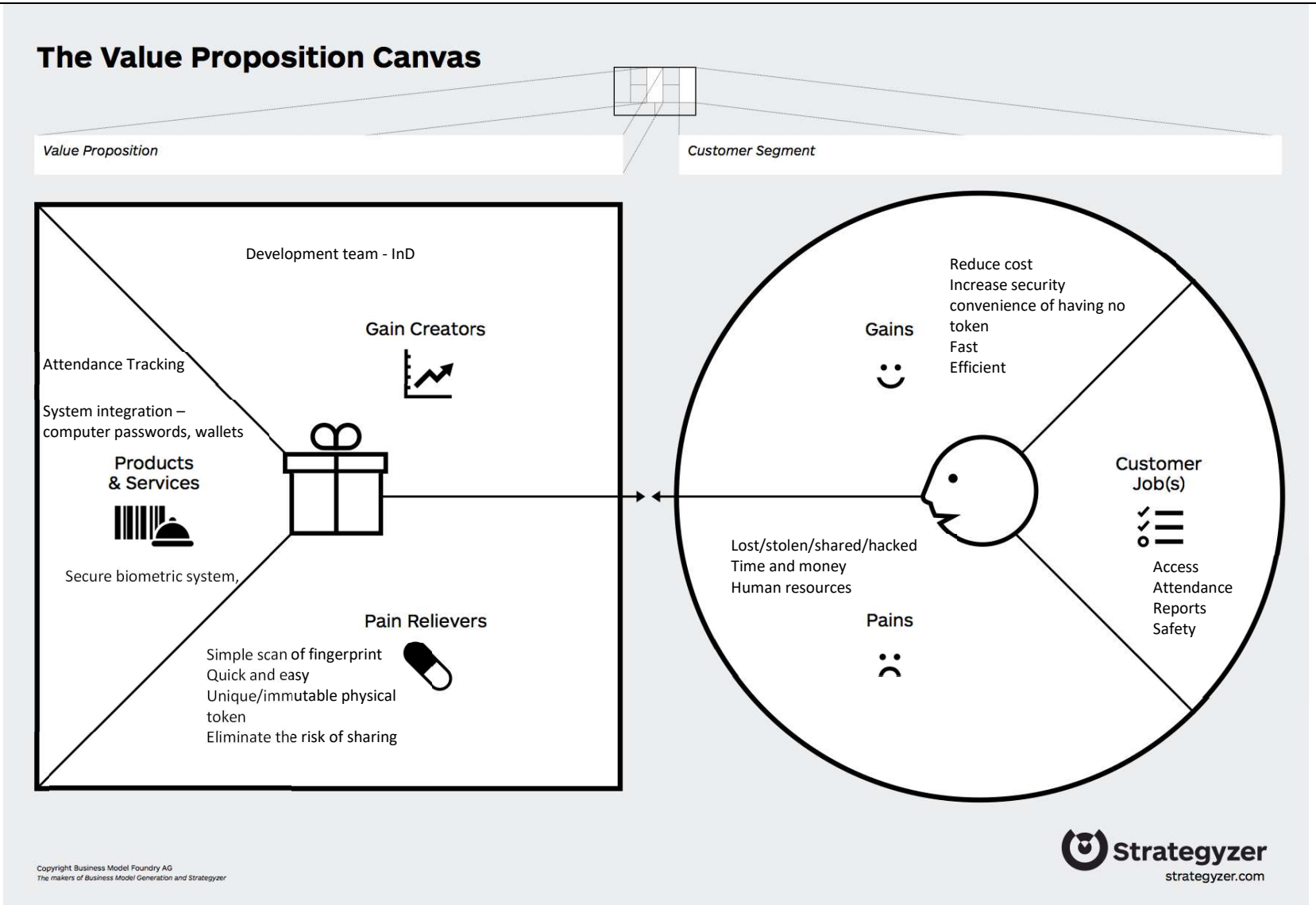
Analysis
Development
Rollout
Hardware/Software resources

Revenue Streams

For what value are our customers really willing to pay?
For what do they currently pay?
How are they currently paying?
How would they prefer to pay?
How much does each Revenue Stream contribute to overall revenues?

Solution packages,
Basic, mid-range, premium

www.businessmodelgeneration.com


The Value Proposition Canvas
Value Proposition
Customer Segment
Development team - InD
Gain Creators
Attendance Tracking
System integration – computer passwords, wallets
Products & Services
Secure biometric system,
Pain Relievers
Simple scan of fingerprint
Quick and easy
Unique/immutable physical token
Eliminate the risk of sharing
Gains
Reduce cost
Increase security
convenience of having no token
Fast
Efficient
Customer Job(s)
Access
Attendance
Reports
Safety
Pains
Lost/stolen/shared/hacked
Time and money
Human resources
Copyright Business Model Foundry AG
The makers of Business Model Generation and Strategyzer
Strategyzer
strategyzer.com

Specification

Data Model

There are four main components to our biometric solution, these are:

- A WinForms desktop application
- A MySQL database
- An android application
- A Restful PHP server

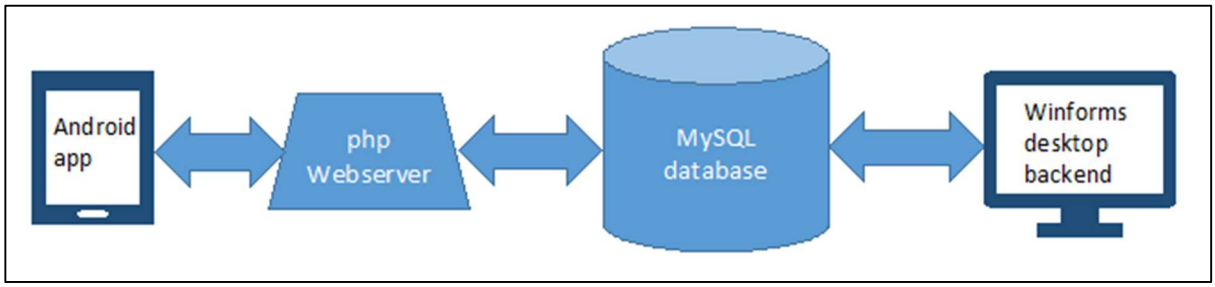


Figure 2 The structure of the communication between these various aspects of the project.

The WinForms application holds the core and essential functionalities we have proposed for a biometric fingerprinting system. The application handles fingerprint acquisition, template extraction and matching, which are essential to producing a biometric system. The application can communicate with the database directly, as illustrated in the diagram above. This allows for data to be inserted into the database through the enrolment process of any employee to the system. The application can declare SQL queries, execute them an in turn receive a response directly from the database, these include searching the database for fingerprint templates, and the production of analytical reports for administrative purposes, for these reasons the WinForms application is the main component for the proposed solution.

The MySQL database will act as the core of the system to which all other applications will connect to. The MySQL database will be stored on a PHP webserver to allow both the WinForms and mobile applications to connect to the database and retrieve, add, edit, or remove data from the database. The advantage of running the database on my PHP web server is that both the WinForms and the Mobile application will be able to view the same

data stored on the server almost immediately. The MySQL commands used by the applications to manipulate the database will also affect the database immediately after execution providing a fast and reliable system.

The android application takes a less direct approach to the way it sends and receives data from the MySQL database. The app makes web calls to a Restful PHP server which then queries the database and returns a response of the results back to the app. All the server functions are predefined within the app making calls to a specific URL to query the database and receives a response string which is encoded as JSON. This leads to the app having no direct contact with the database and limits the queries it can execute to the predefined queries that exist in the server functions.

Function Definitions

Web Server Functions

Login.php - This function on the server receives a request containing an email and password, it then checks these variables in the database and returns a Boolean value dependant on if it is a valid email/password combination along with details of the user.

SearchUsers.php - This server function receives a string containing a name or part of a name and performs a query that returns all users whose name is like the string. It then returns all results in a JSON array.

CreateMeeting.php - This server function receives details String details about a meeting to be created as well as the location id and inserts it into the Meetings tables in the database. It then returns a Boolean indicating if it was a success or failure.

UpdateMeeting.php - This server function receives a String containing updated notes for an existing meeting. It then updates the table with the new notes string and returns a Boolean indicating if it was a success or failure.

GetMeeting.php - This server functions receives a user an integer value containing a user id, it then queries the database to search for all meetings where this id is the owner and meetings where this user id is an attendee. It then returns details of the relevant meetings as a JSON array.

GetMeetingDetails.php - This server functions receives an integer value containing a meeting id, it then queries the database using this id. The results of this query are then returned as a JSON array.

GetMeetingAttendees.php - This server function receives an integer value containing a user id, it then queries the database to find all attendees of this meeting id. The results of this query are then returned as a JSON array.

InviteUser.php - This server function receives a meeting id and a user id, it then adds the user as an attendee of the meeting. A Boolean is then returned indicating if it was a success or a failure.

GetLocations.php - This server function does not receive any variables, it simply returns all location areas along with their respective location id as a JSON array.

UpdateProfile.php - This server function receives string values for updating user information, it receives the updating information and its relevant user id and performs an update using the string values received. It then returns a Boolean value indicating if the update was a success or failure.

Android Application Functions

MainActivity - This is the activity the user is presented with on start-up, it contains the login functionality and communicates with the Login.php server function. If a valid username/password combination is entered this activity will take the user to the main menu and pass along details of the logged-on user to the MainMenu activity.

MainMenu - This is the main navigation area for the user. From here the user can search for other registered users using the bar at the top which communicates with the SearchUsers.php server function or they can navigate to the other functional parts of the app using the 4 buttons presented. If the user navigates to another activity details of the logged in user is passed to the new activity.

MeetingsActivity - This activity is presented when the user presses the meeting button on the main menu. It contains a listview of all meetings that the logged in user is associated with and allows the user to navigate to the create meeting activity using the button on the top right. This activity communicates with the GetMeeting.php server function.

CreateMeetingActivity - This activity is accessible from the MeetingsActivity and allows the user to create a new meeting by entering relevant information and choosing a location from the drop-down box. This activity communicates with the CreateMeeting.php server function.

ViewMeetingActivity - This activity is accessible by selecting a Meeting from the ListView in the MeetingsActivity. It presents the user with details of the selected meeting and allows

them to edit the notes for the meeting. This activity communicates with the GetMeetingDetails.php and UpdateMeeting.php server function.

ProfileActivity - This activity is accessible by pressing the profile button on the MainMenu, it provides the user with a view of their account information and allows them to edit their information. This activity communicates with the UpdateProfile.php server function.

SearchUsersActivity - This activity takes the username from the MainMenu search users text field and presents the results of the search to the user. The user can then select an item from the list and will be taken to the ViewUserActivity for the user they selected. This activity communicates with the SeachUsers.php server function.

ViewUserActivity - This activity presents information on a searched user and allows them to be invited to any meeting that the logged in user is associated with. This activity communicates with the InviteUser.php server function.

WinForms Application Functions

Terminal - This form is where users will scan their fingerprint to enter/exit an area. The Terminal will use the fingerprint they have scanned, extract the fingerprint minutiae template, and compare it to the fingerprints saved to the server. Once a match is made the user will be granted access if their access level is equal to or greater than the required access level of the area they are attempting to access. Once a user has been granted access the Terminal will log their activity and save it to the database.

Menus - The MainMenu, EmployeeAdmin and LocationAdmin forms will act as navigation areas of the WinForms Application. Navigation buttons will be present to allow the user to open the functions/tools that will be available in the solution.

EmployeeMaintenance - This form is used to add, edit, or delete employees from the database. The form allows the user to enter key employee information and will also allow the user to upload the employee's photograph. It is from here that each new employee can register their fingerprint to be scanned and saved onto the system.

SearchEmployees - This form will be used to search for employees that are saved to the database. The user can either search by employee Id or by forename/surname. Searching by name will return any employees that meet the search parameters. With these results the

user can open the EmployeeMaintenance of the selected employee or export all returned results to Microsoft Excel with which the user can print.

ActivityReport - This form allows the user to review the access history of employees. Just like the Search Employees form the user can search by employee Id, forename, or surname. The user can also choose to search by location which will return the names of any employees who have access the selected area. The user will also can search by date between a specific time.

LocationMaintenance & AreaMaintenance -Both the Location Maintenance and Area Maintenance form are relatively similar. The user has the option to create, edit, or delete areas/locations. Locations can have multiple areas whereas areas can only be assigned to one location. For example, "The I.T Department" is a location which could have five Meeting Rooms, these rooms would be classed as areas.

FingerprintFetch - the fingerprint fetch function runs alongside the terminal and employee maintenance processes, this function provides the link between hardware and software, enabling fingerprint acquisition and returns it to the relevant process

Error Conditions

It is the aim of our development of the system that all conditions that could lead to an error are handled by the system and the users experience should never be interrupted. We hope to achieve this using appropriate validation on user input fields, prompting the user when something is incorrect and allow them to correct their input and by using try/catch statements throughout on aspects of the code that could produce an error.

From the perspective of the PHP web server, errors will be handled in a very simplistic way, if there is an issue with the variables sent by the android app and the server is unable to execute the query then an SQL error code is returned as a response to the app. This allows the app to then handle the error once it has received the response. Once the server receives a valid input from the client and executes the SQL statement it also makes use of a simple "success" Boolean variable that is returned to the client, as can be seen in the example below.

```
$response = array();
$response["success"] = false;

if(mysqli_stmt_affected_rows($statement) > 0){
    $response["success"] = true;
}
```

This variable allows the client to verify if the query has executed with the desired results and if not it allows the user to be prompted to retry the function. Using these two simple methods almost all errors can be handled efficiently by the app and we are able to prevent the user from being disrupted in their experience of the app.

In both applications; Android application and the WinForms desktop program, errors are handled primarily using try/catch blocks with the android application also making use of the success Boolean, as mentioned above. Using try/catch blocks in the code we can notice the primary causes of errors which are executing SQL statements directly on the database and parsing JSON replies from the web server. As both these errors are typically caused through issues with the users input, in the event of an incorrect user input appropriate error messages will be displayed to the user to highlight where the occurrence of an error resides.

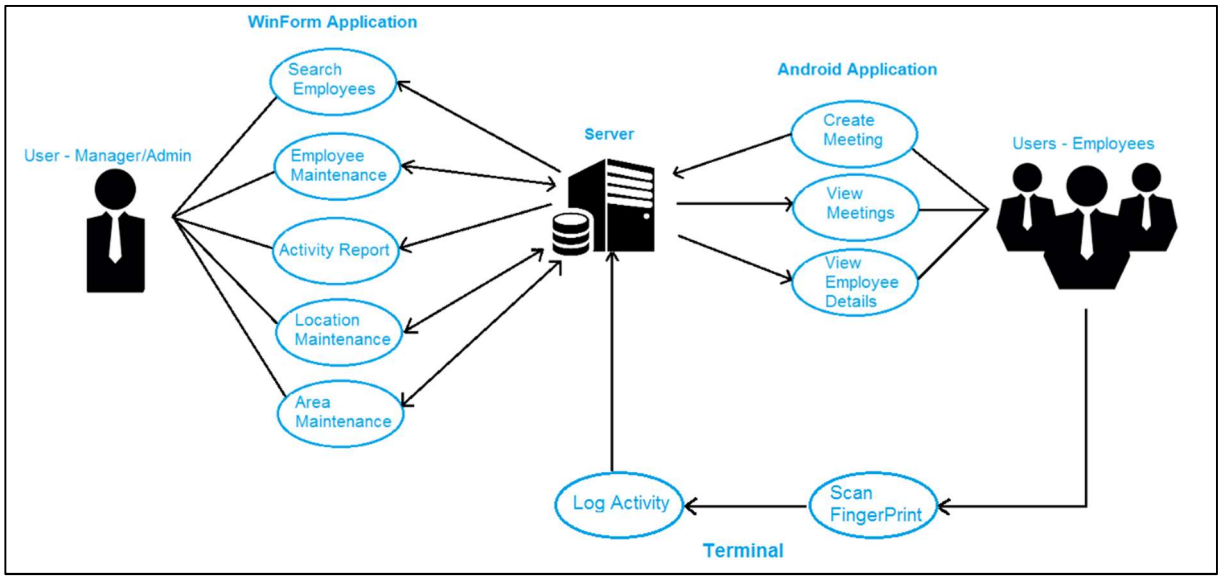
Assumptions & Constraints

- It is assumed that the android device used will be at least API version 15, the target API version of the application is 24. The application has been tested primarily on a Nexus 5 emulated device. It is also assumed that the device will have a working internet connection.
- It is assumed that the Desktop program will be run on the Windows operating system with windows 7 being the targeted minimum and the primary testing being done on Windows 10.

Design

Use Case Diagram

The Use Case Diagram (UML) for the project is shown below. It represents user interaction and connections between the Android, WinForms, and the Scan Fingerprint applications. It describes the relationship between the users and the various activities throughout the project. The Android Application will be used by employees to schedule meetings and book available rooms. The WinForms Application will only be used by Managers/ admins that wish to edit employee details or register new employees, or add/edit locations and areas as well as view reports on employee activities. The Scan Fingerprint application interfaces between the user and the WinForms, this process determines if the user is granted or denied access.



Software System Design

The biometric fingerprinting system key system components are the fingerprint acquisition development and the fingerprint matching development. The development of the fingerprint acquisition application will be completed using C++. The WinForms application will be developed in C#, but because the development team have chosen to development a fingerprint device that uses API's coded in C++ to enable communication between hardware and software, the conversion of code between C++ and C# will be a time-consuming task, the team have concluded that the WinForms application will call for the execution of the fingerprint acquisition application, which will be tasked to acquire and return the scanned fingerprint to the WinForms application. The fingerprint acquisition component will be called to run through employee registration, which will enrol an employee's scanned fingerprint and personal details to the database system. This component will also be called to run through the process of building entry/exit. The fingerprint matching component of the system will be developed in C# within the WinForms Application, and the component will be called to run through the process of an employee trying to gain access to a secure building or area. As mentioned above the fingerprint acquisition component will work on the same process, as the matching component requires a scanned fingerprint before execution.

Employee enrolment Sequence Diagram

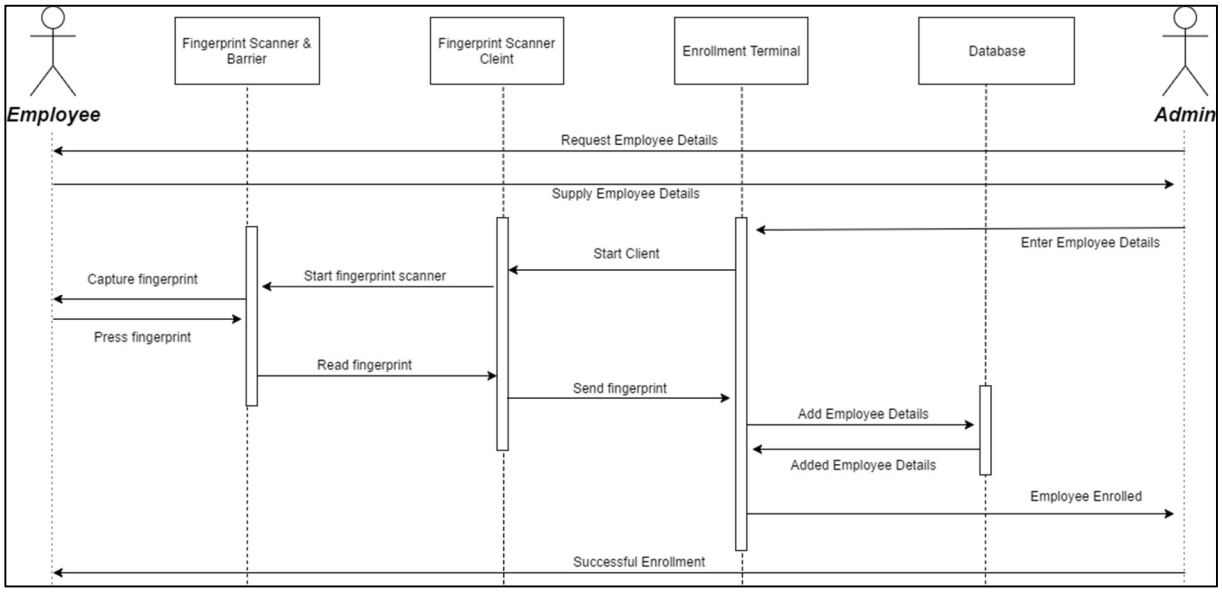


Figure 3 Employee fingerprint and details enrolment.

The admin user of the system begins the sequence of events for employee enrolment, with interaction with the employee, the admin request the employee's details and the employee provides. The admin then interacts with the purposed WinForms Application through the Enrolment Terminal, that enables the entry of details. When this event begins, it will call on the developed fingerprint scanner client to execute, starting communication between the fingerprint scanner and its client. The employee will then be required to press on the fingerprint scanner, and when an accurate fingerprint has been scanned the data will be read back to the fingerprint client and then onto the Enrolment terminal. The enrolment terminal will be developed to include communication to the database to add employees fingerprint and details, the database returns the success of the process to the admin and employee is notified.

Employee Access Control Sequence Diagrams

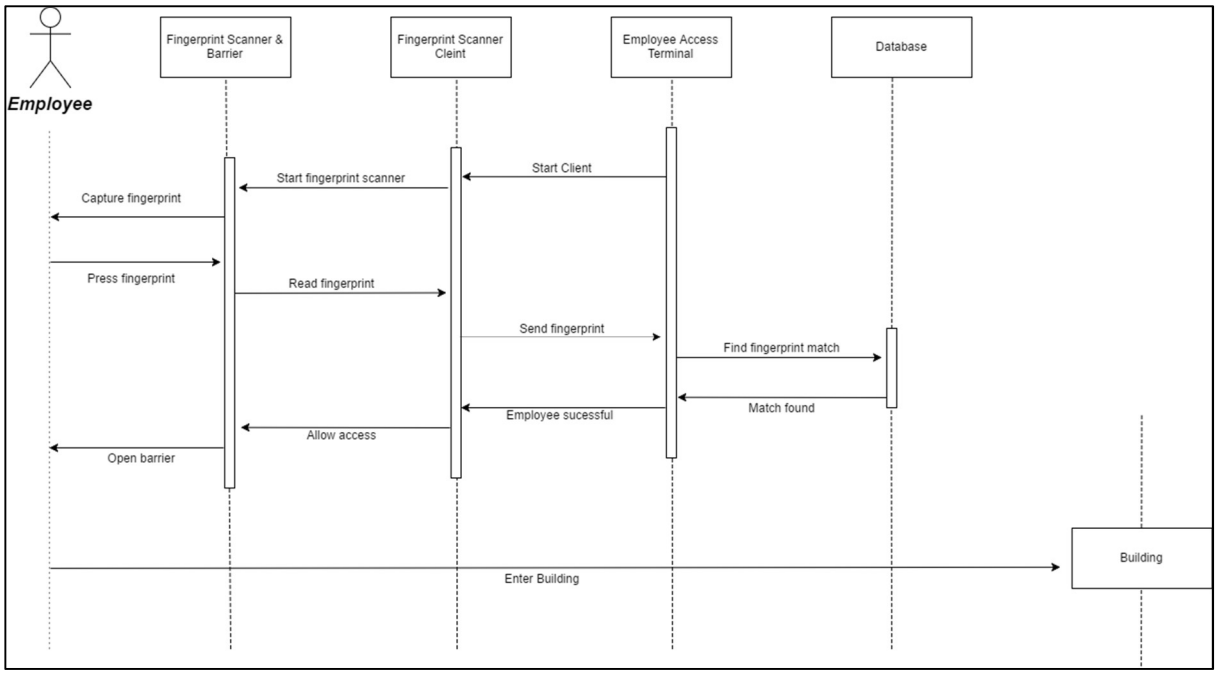


Figure 4 Sequence Diagram for Successful employee access

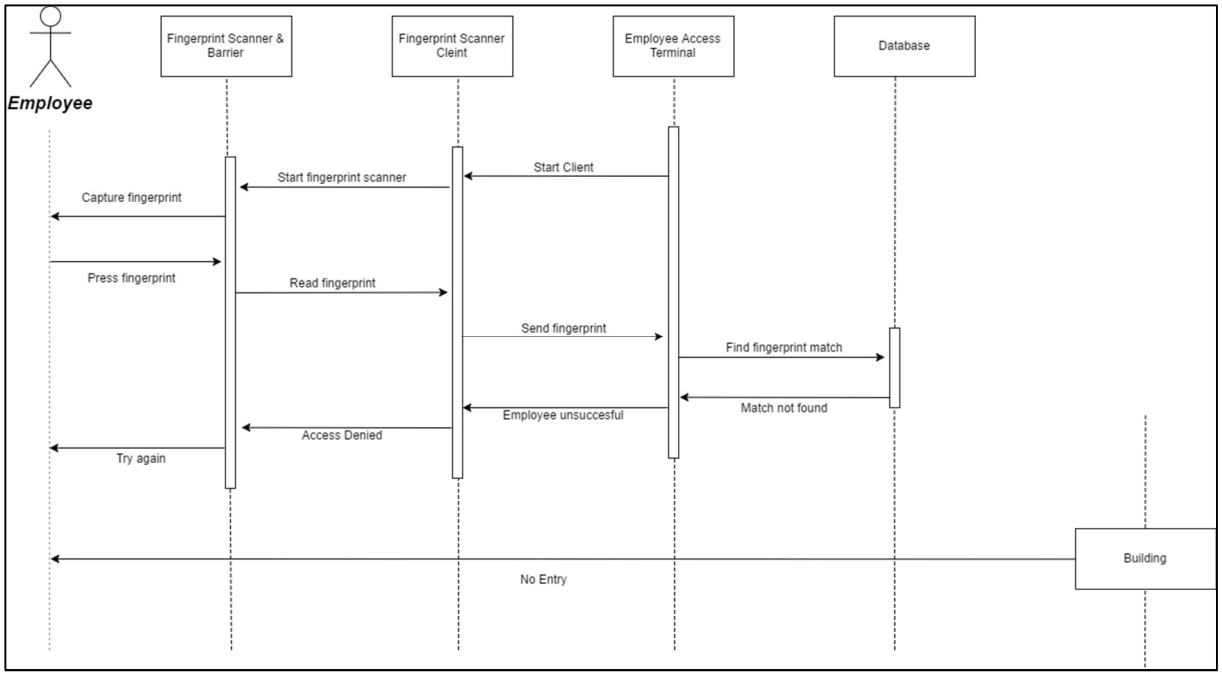
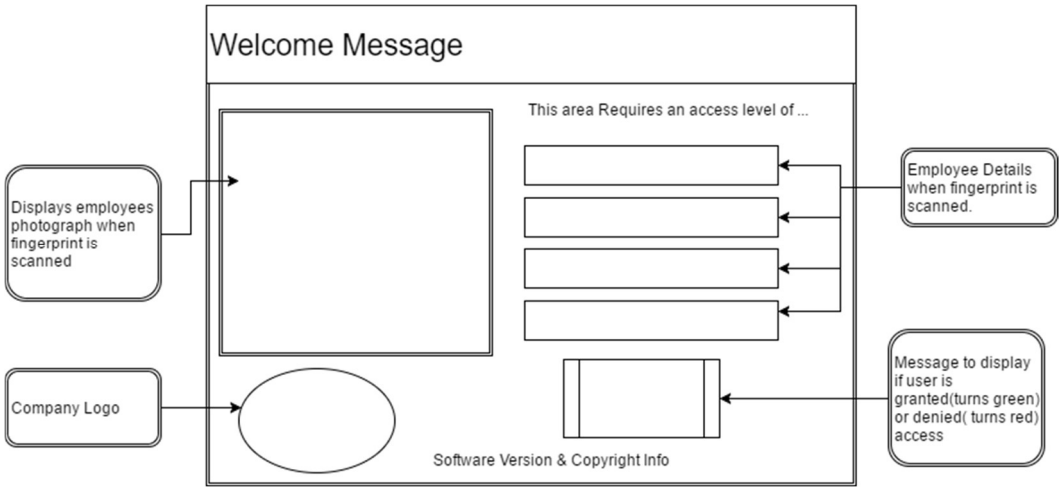


Figure 5 Sequence Diagram for denied employee access

The sequence of events for employee access control will begins on the Employee terminal, the terminal will start the execution of the fingerprint scanner client which is responsible for the communication between the fingerprint scanner and the fingerprint client. The fingerprint will then be ready to capture fingerprints. The employee will press their fingerprint on the scanner and their fingerprint will be read from the scanner to the employee terminal. The development of the employee terminal will at this stage call on the fingerprint matching functionality to search the database for fingerprints and compare the scanned fingerprint with each on in the database to determine a match. Figure 5, illustrates the sequence of a fingerprint match in the system. The Employee terminal will communicate to the fingerprint client of the fingerprint match and the fingerprint scanner will be notified and will signal success. Employee will then be able to access the building. Figure 6 illustrates that if a fingerprint match has not been found in the database, the employee terminal will notify the fingerprint client and in turn the fingerprint scanner and signal the result, the unknown will be denied access to the building.

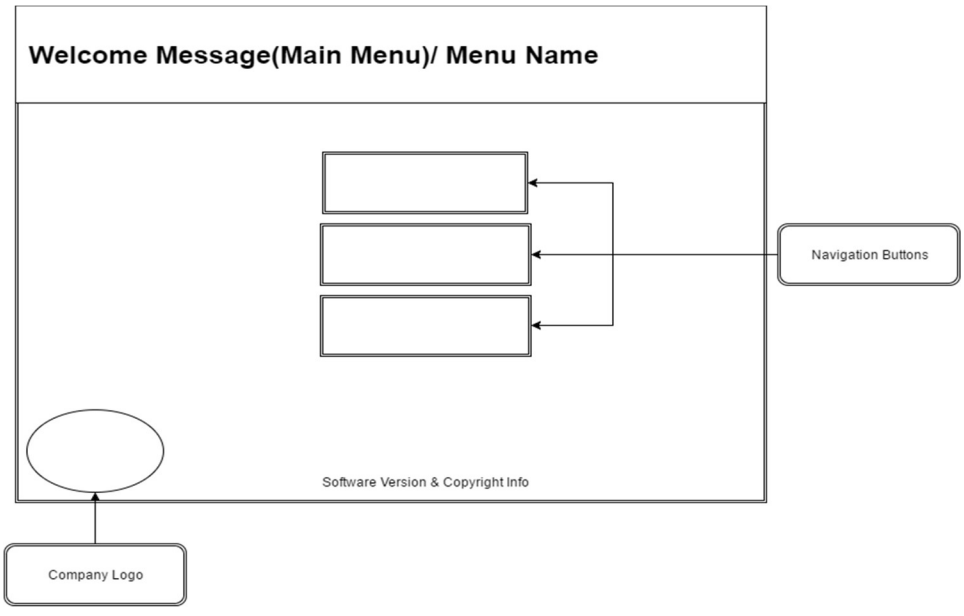
User Interface design

Biometric Access Terminal

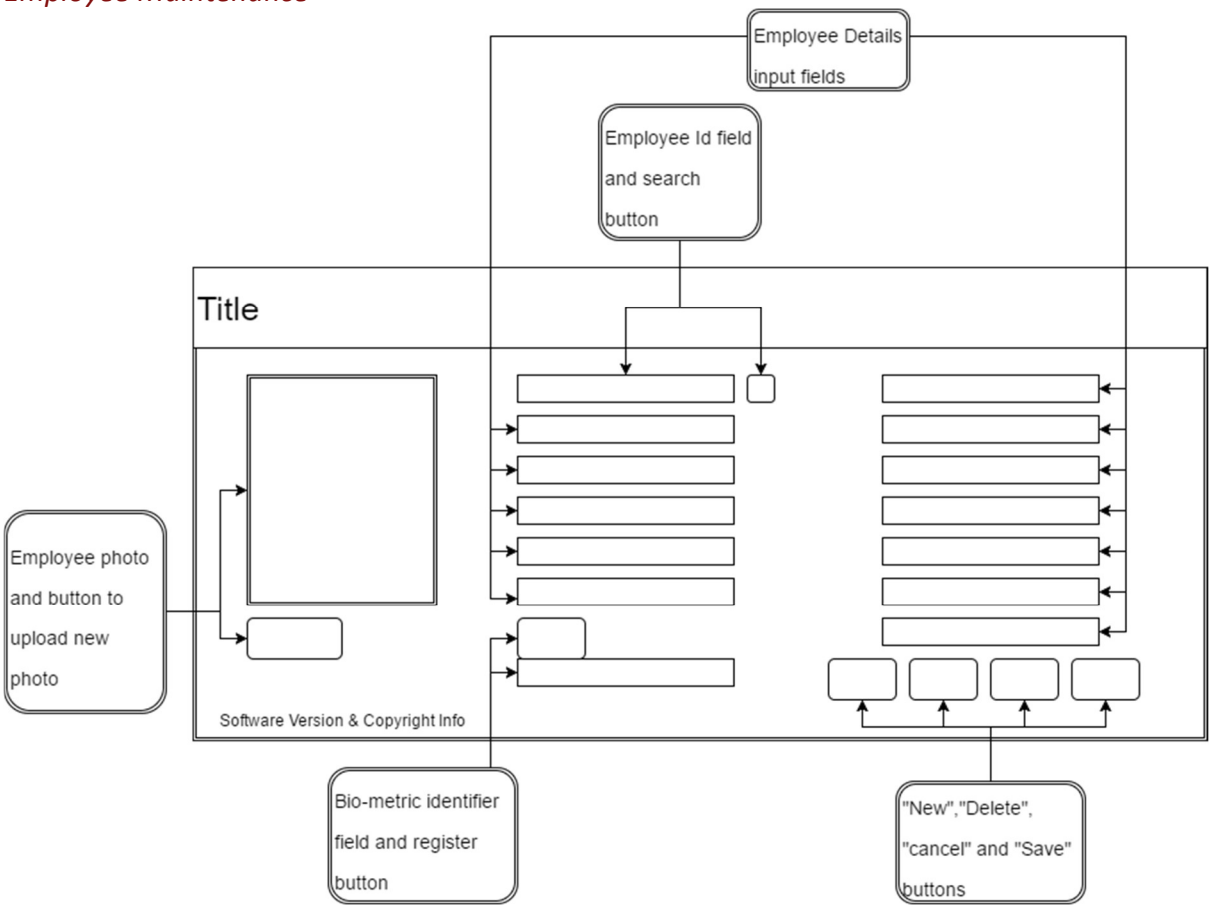


Menus

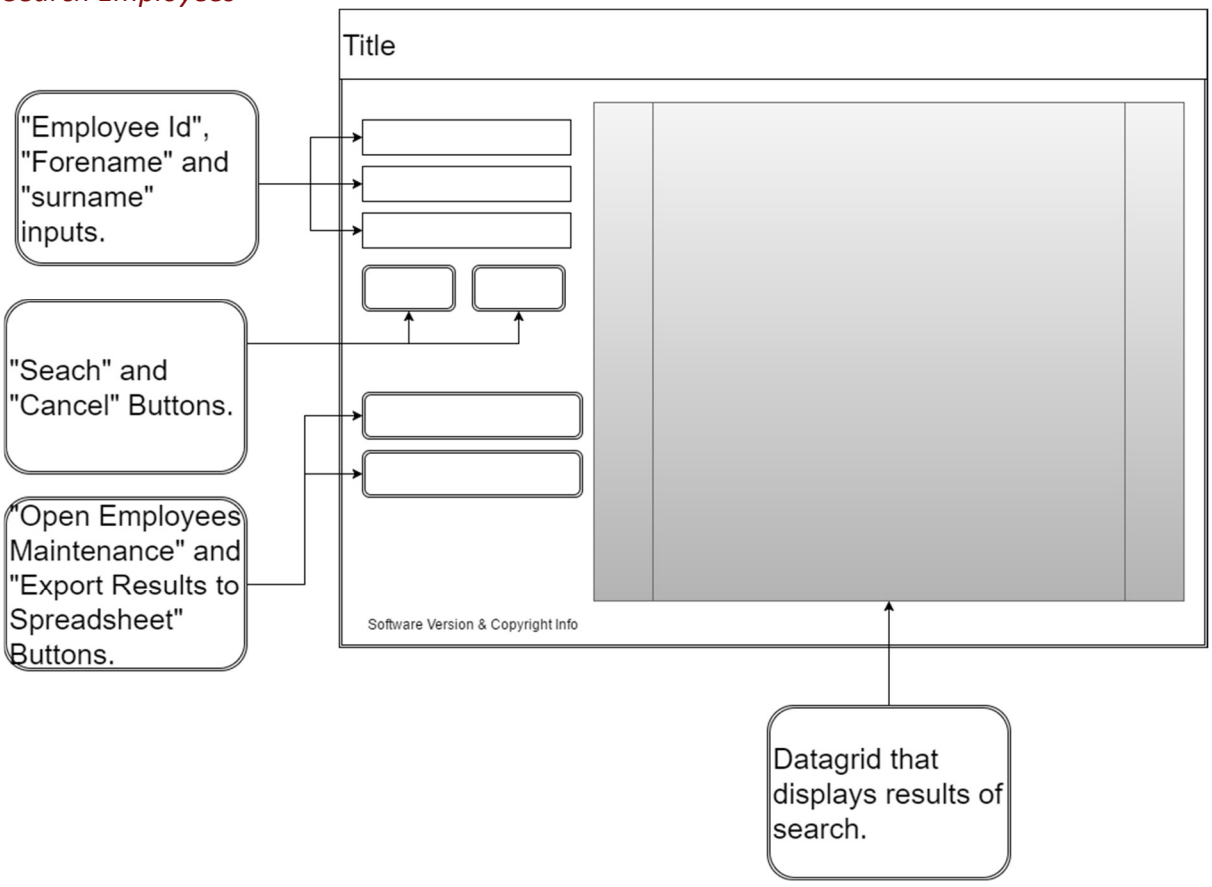
Mock-ups for MainMenu, EmployeeAdmin and LocationAdmin



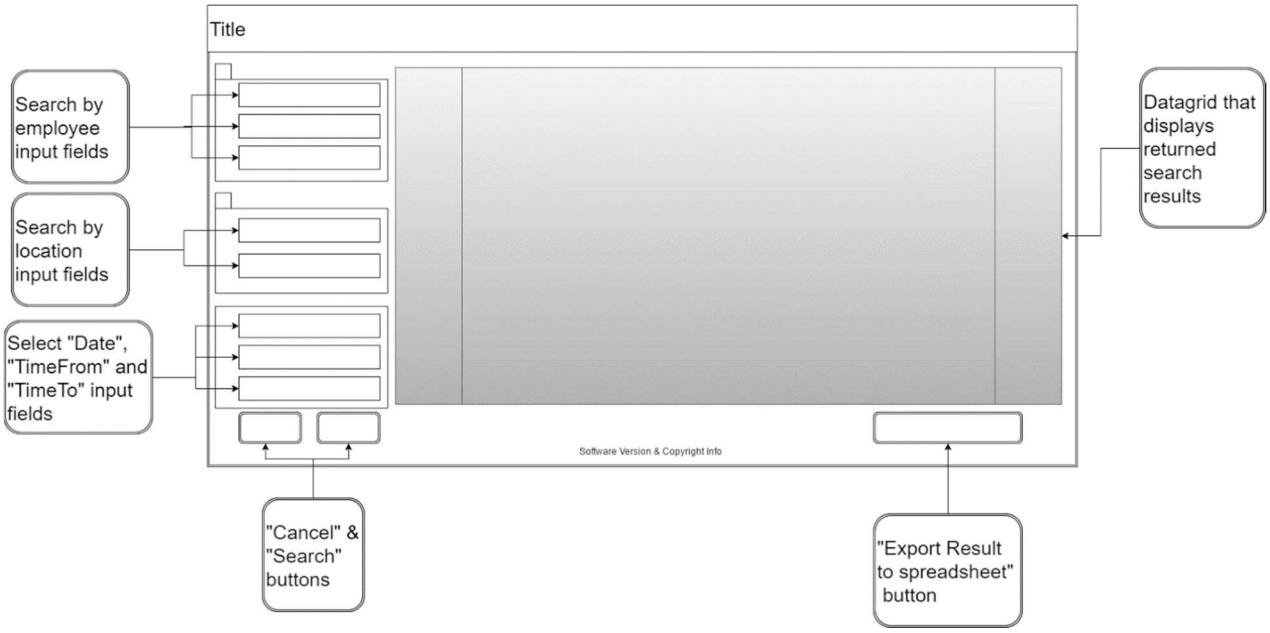
Employee Maintenance



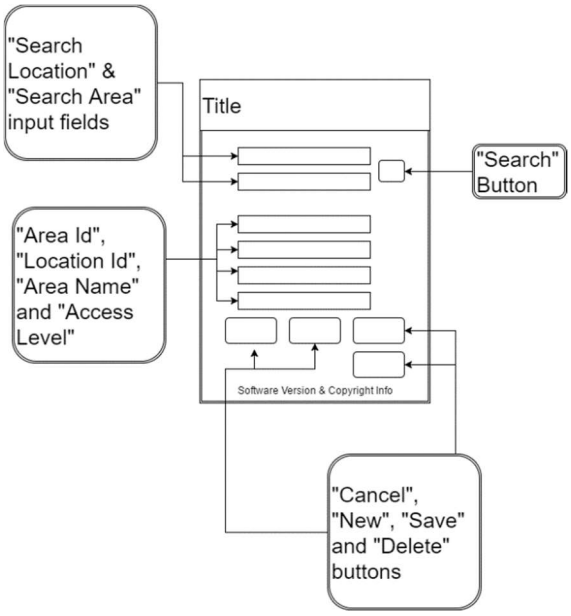
Search Employees



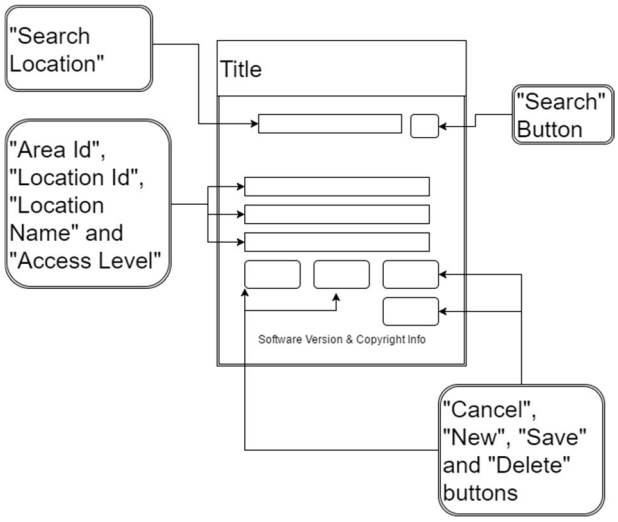
Activity Report



Area Maintenance



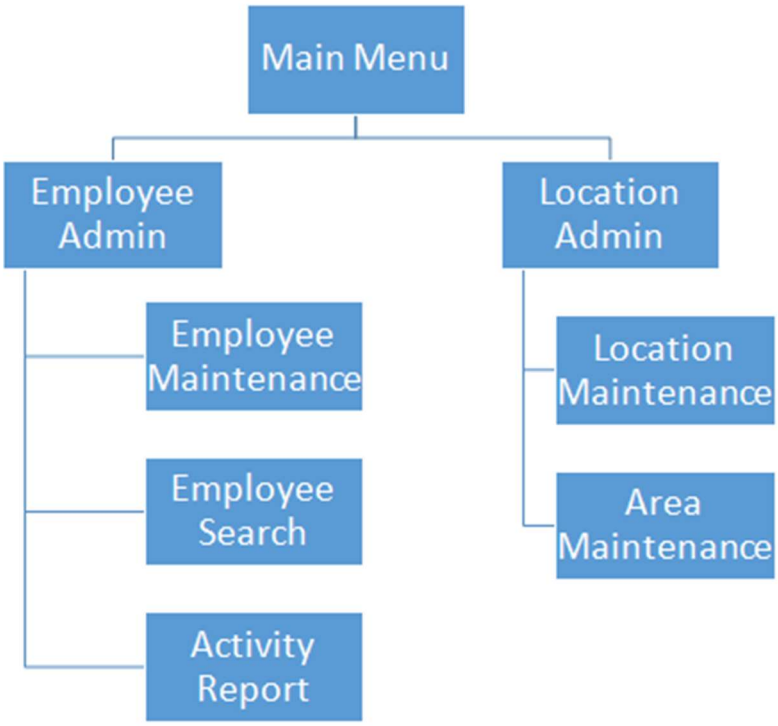
Location Maintenance



Database Tools and Reporting Software

The database tools and reporting software will be in the same software package and can be loaded onto any admin or manager's terminal within the company. Below are mock-ups of how the user interface will look for this package along with a short description of each field present.

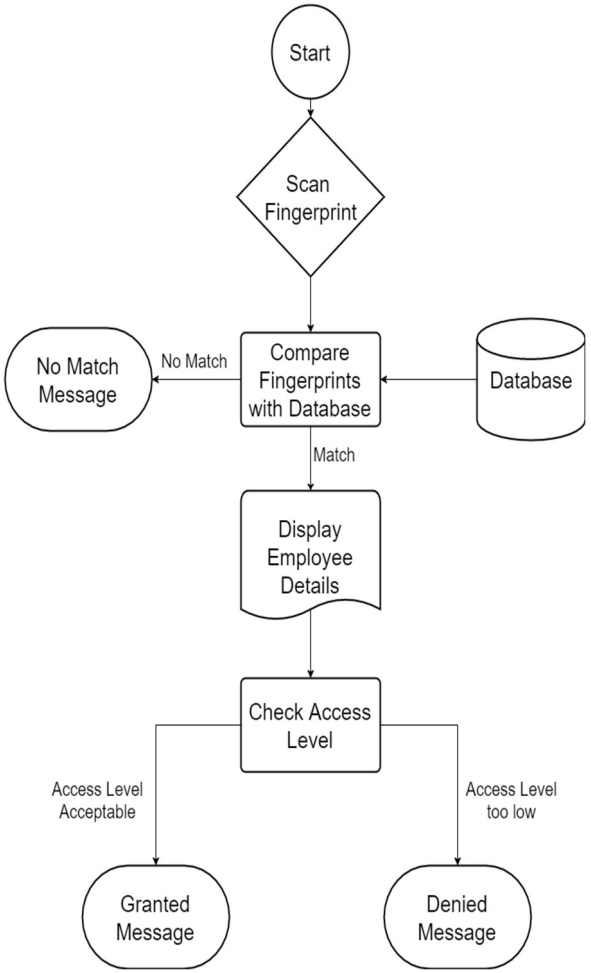
The Windows Form Application will be developed using C# and the UI can be seen in the diagrams above. The Windows form contains a set of tools that a manager or admin can use to manipulate data on the server. The form will also contain the ability to fetch data and publish them as printable reports below is a screen flow diagram that represents how each page in the Form Application will be connected.



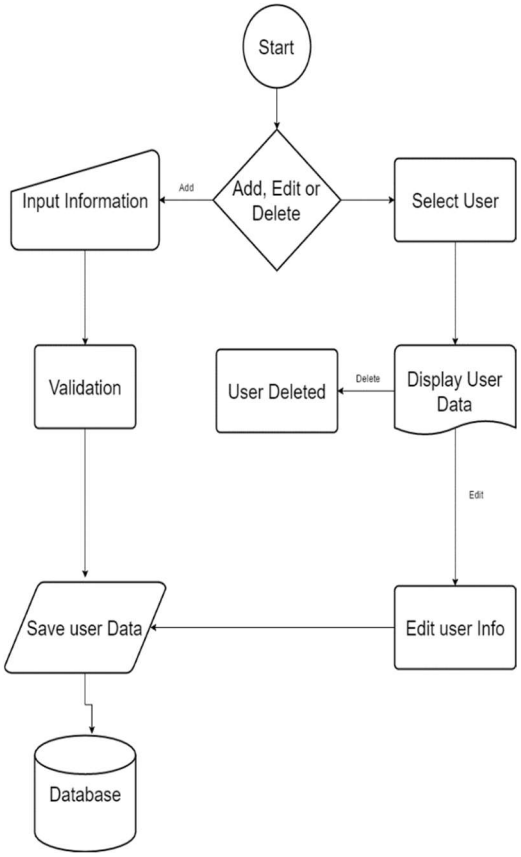
The Terminal, which employees will use to scan their fingerprints at each entrance will also be created using C#. The user will scan his/her fingerprint which will grant or deny access to the area they wish to enter/exit and then automatically refresh to allow the next user to scan their fingerprint.

As explained in the specification, the database will be hosted on the eeecs service and will be developed using MySQL. The Terminal, Windows Form Application and the mobile application will access this database using MySQL queries. The entire system relies heavily the server as the Windows Form and Mobile Applications will not be able to access the database without first connecting to the server.

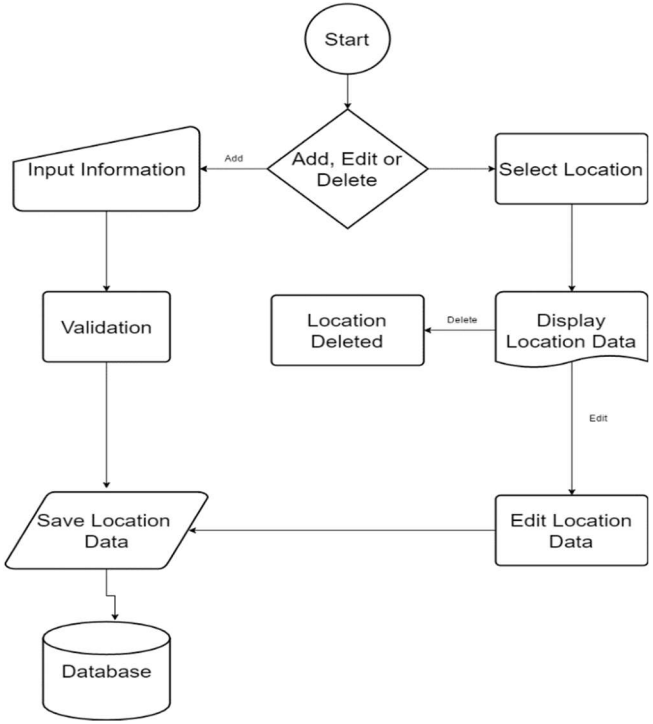
Terminal



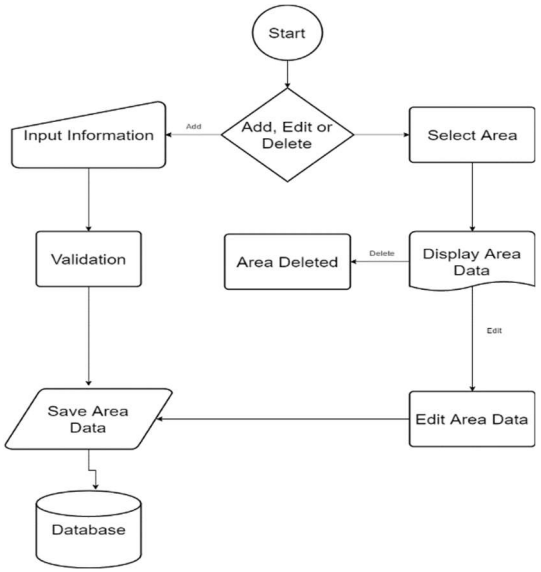
Employee Maintenance



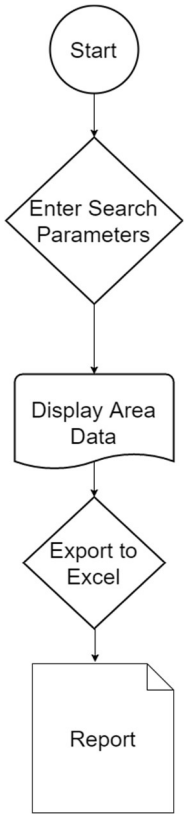
Location Maintenance



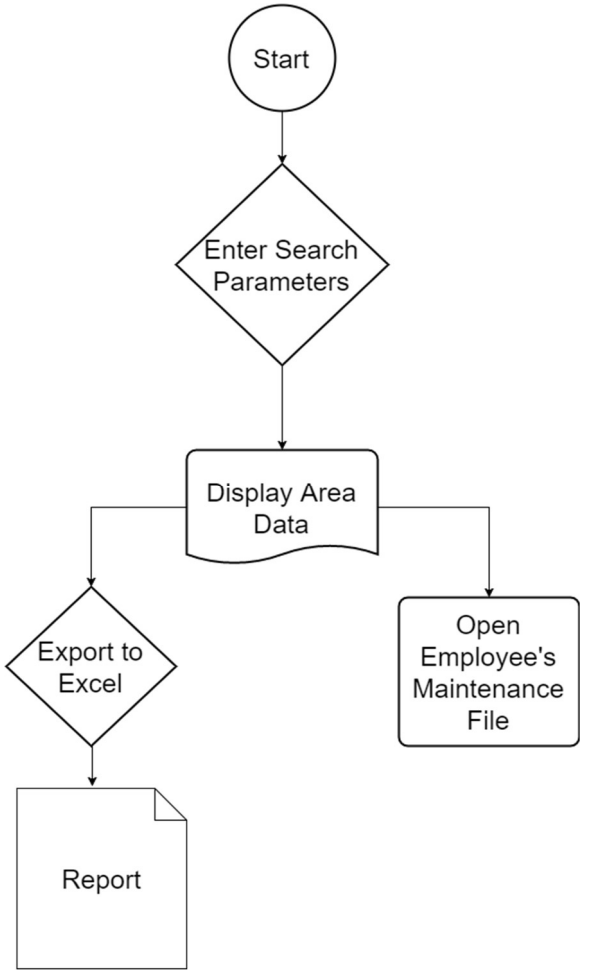
Area Maintenance



Activity Report



Search Employees



As illustrated below, the designs for each table on the database:

EmployeeDetails

Column	Type	Null	Default
Id	int(11)	No	
Forename	varchar(50)	Yes	*NULL*
Surname	varchar(60)	Yes	*NULL*
Email	varchar(50)	Yes	*NULL*
TelephoneNo	int(11)	Yes	*NULL*
AccessLevel	int(11)	Yes	*NULL*
BiometricMarker	longblob	Yes	*NULL*
AddressLine1	varchar(50)	Yes	*NULL*
AddressLine2	varchar(50)	Yes	*NULL*
AddressLine3	varchar(50)	Yes	*NULL*
City	varchar(50)	Yes	*NULL*
Postcode	varchar(50)	Yes	*NULL*
Photo	longblob	Yes	*NULL*
password	varchar(15)	Yes	*NULL*

EmployeeAccessHistory

Column	Type	Null	Default
AccessId	int(11)	No	
EmployeeId	int(11)	No	
EmployeeForename	varchar(60)	Yes	*NULL*
EmployeeSurname	varchar(60)	Yes	*NULL*
AreaId	int(11)	No	
AreaName	varchar(60)	Yes	*NULL*
TimeOfAccess	time	No	
AccessType	varchar(60)	No	
Date	varchar(60)	No	
LocationId	int(11)	No	
LocationName	varchar(60)	Yes	*NULL*

Locations

Column	Type	Null	Default
Id	int(11)	No	
LocationName	varchar(200)	Yes	*NULL*
AccessLevel	int(11)	Yes	*NULL*

LocationAreas

Column	Type	Null	Default
Id	int(11)	No	
LocationId	int(11)	No	
AreaName	varchar(200)	Yes	*NULL*
AccessLevel	int(11)	Yes	0

MeetingAttendees

Column	Type	Null	Default
Id	int(11)	No	
MeetingId	int(11)	No	
EmployeeId	int(11)	No	

Meeting

Column	Type	Null	Default
Id	int(11)	No	
MeetingTitle	varchar(50)	Yes	*NULL*
MeetingDescription	varchar(200)	Yes	*NULL*
MeetingNotes	varchar(1000)	Yes	*NULL*
MeetingDate	date	Yes	*NULL*
MeetingTime	time	Yes	*NULL*
LocationAreaId	int(11)	No	
MeetingOwnerId	int(11)	No	

User Interface design

Android UI Design

While designing the user interface for the android application the focus was keep the screen presented to the user uncluttered and to make sure information was presented to the user in a clear and concise fashion. The way this was achieved was by using a simple title bar that would contain the title of the page the user was currently viewing and contain any functionality that would take the user to a different activity, for example the search user and create meeting buttons are both contained in the title bar.

As can been seen in the example above this allows the main area of the user screen to focus presenting the information relevant to whatever task the user is trying to achieve while the title bar can be used as a way for the user to access any additional functionality related to their current task, e.g. creating a new meeting from the view meeting activity.

With the use of the title bar to provide the user access to functionality this allows the main area of the screen to focus on allowing the user to carry out whichever is the main function of the activity they are currently on. So, for example if the user is trying to create a new meeting they can focus on just the main area presented to them and it will contain only things related to creating a new meeting with all other functionality being in the title bar.

Software System Design

Android/Server

The basic flow of information between the android application and the database is the user will input any relevant information, then the app will take this information and send it to the appropriate URL on the PHP web server and wait for a response, this is done using the Volley

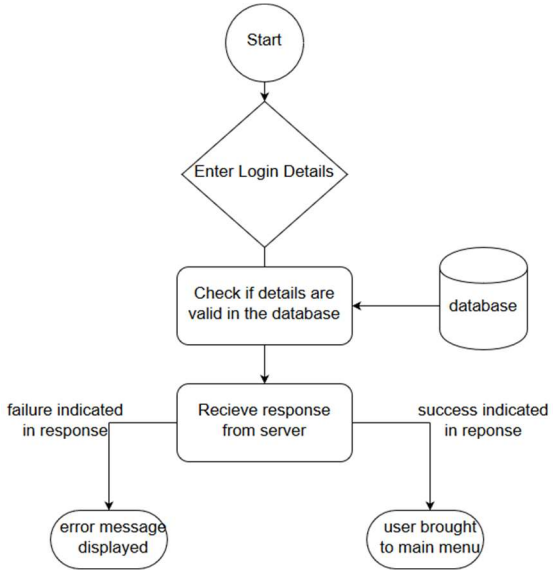
library for android. Once the web server receives this information it will perform the necessary SQL statements on the database and send a response back to the app based on the results of the SQL. Then the app will receive the response and will be able to determine the results of the action and proceed accordingly.

Typically, each activity in the application will make a call to the server in its OnCreate method to populate the user view with the relevant information. Once the view has been populated then the user will be able to edit and update any relevant text fields, this is done by making another call to the server on the press of a button which takes the new information input by the user and sends a request to the server to allow it to perform an update. The server response will also contain a Boolean value indicating whether the function has succeeded or failed which makes it easy for the app to identify when something has not worked and quickly inform the user to allow them to change any incorrect information that could cause an issue and try again.

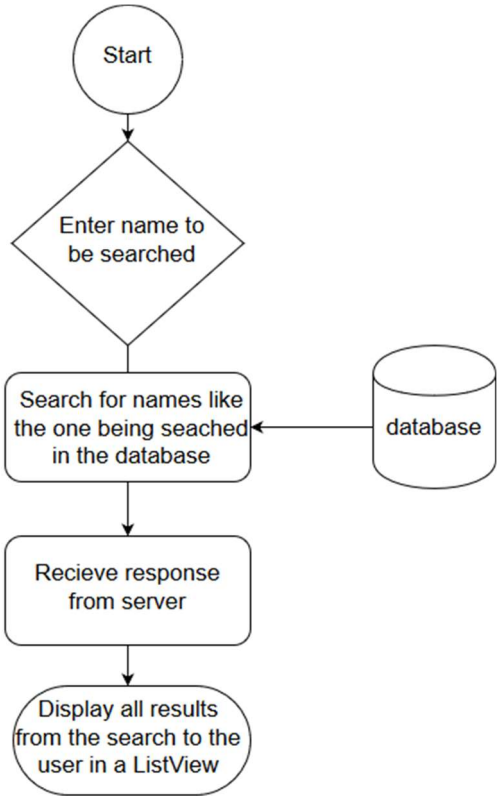
The chosen of communication between the app and server was the android Volley repository. Volley was chosen as it is extremely easy to set up and start using and provides all the necessary functionality required for the purposes of our system. Another reason Volley was chosen was because it is very efficient and allows requests to be sent with no delay that could impact the user experience in a negative way. Overall it provides a very simple and efficient solution.

The primary function of the Android application is to serve as a tool for the user to create and organise meetings within the company. The reason the app was designed around the use of the meeting functionality was to enable to user to have a convenient way to schedule meetings and invite other users without having to be sitting at their pc using the WinForms backend. The user can create a meeting using the app, add other users to the attendees list for their meeting allowing them to see the meeting along with details of the time/date and location and then the users associated with the meeting are able to create and edit notes for the meeting.

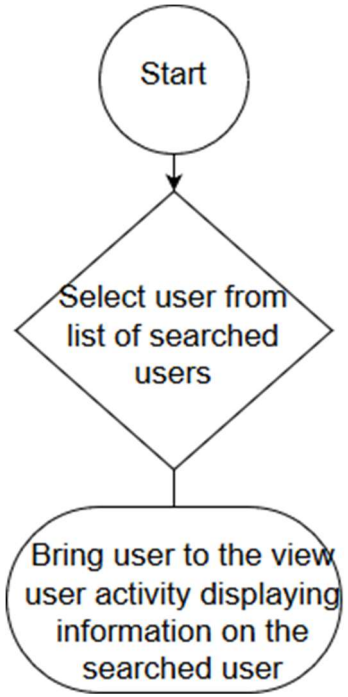
Login user



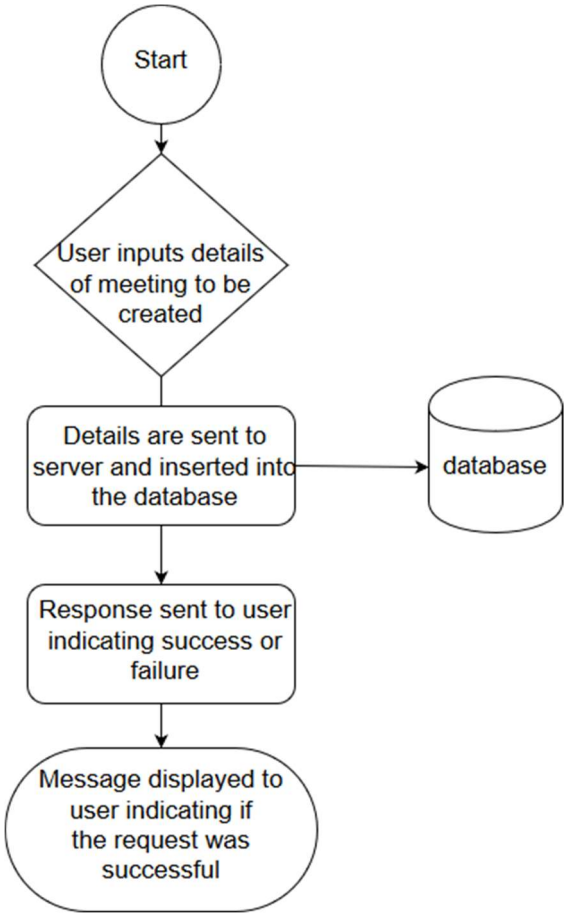
Search Users



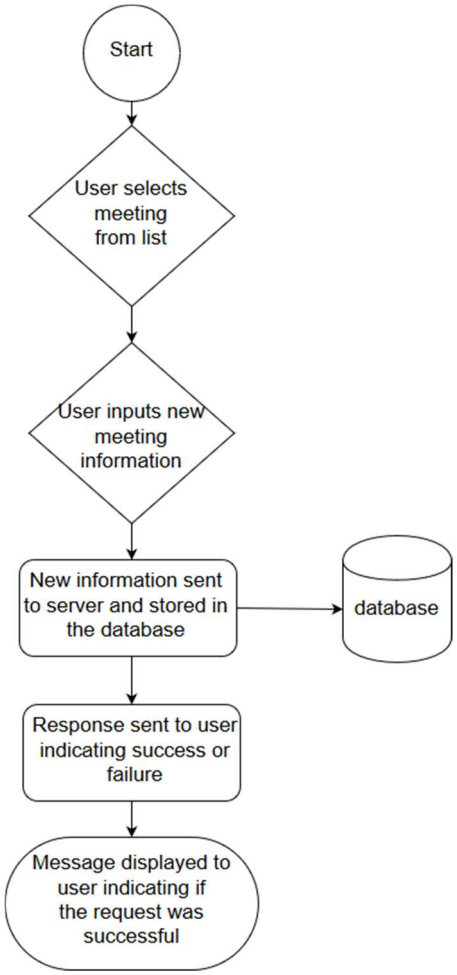
View User Details



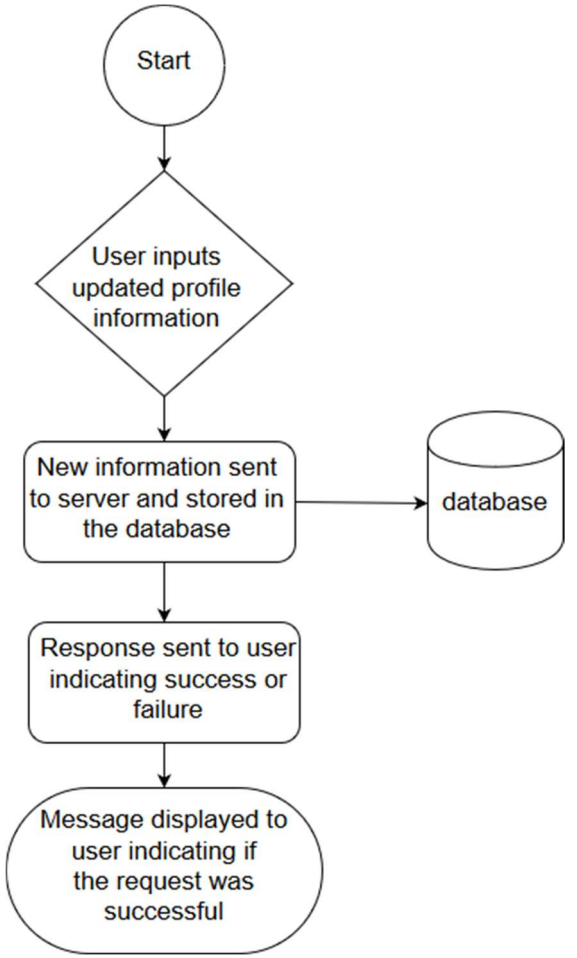
Create Meeting



Update Meeting



Update Profile



Implementation

This section covers the actual implementation and creation of the inD Biometric Project. It further describes the Development Methodology as well as highlights the most significant pieces of code developed. All source code for the inD Biometric Project can be found on the USB submitted alongside this dissertation.

Development Methodology

As mentioned in the Specification, an Agile approach was used in the development of the project. Using the user stories as well as the requirements a series of weeklong sprints were schedules. After each sprint two days of testing the code was done followed by a sprint review. It was essential that the Sprint Reviews were performed as it provided a report on areas of code to be improvement as well as to highlight any bugs that were reported during testing.

Agreed Coding Standards

The team have agreed on a set of Coding Standards to follow to create a consistent and clean look to the code. The conventions are as follows:

Naming Conventions

- Classes and Method names must use Pascal casing meaning each class/method must start with an uppercase letter in each word. For example, class EmployeeMaintenance.
- Variables and method parameters must use Camel casing and begin with a lowercase letter in the first word, all following words will have an uppercase first letter. For example, string inputString.
- Underscores (_) are not to be used in names.
- File name should match the class name. For example, for the EmployeeMaintenance class, the file name should be EmployeeMaintenance.cs.

Layout Conventions

- There is to be only one statement per line.
- There is to be only one declaration per line, however the declaration of variables of the same type on a single line is allowed.
- Parentheses must start on the line underneath a declaration of a method, class, or expression.
- Code that is contained inside parentheses must be indented if not done so automatically.

Commenting Conventions

- Comments must be written one line above the target code, it is not to be used on the same line.
- Comments must also have one space before the "//".
- Comments must begin with an uppercase letter and end with a full stop.
- Spelling and grammar must be correct.

Significant aspects of the fingerprint acquisition, template extraction and matching implementation

One of the significant aspects of the biometric fingerprinting system is communication between hardware and software. Through the communication that was required for fingerprint acquisition an executable file was developed that is called from the WinForms Application when a fingerprint image is required for the terminal process which grants or denies user access, or through the employee maintenance process which enrols the user to the system.

For this project, was developed for one fingerprint scanner brand, Futronic-Tech. The company supplied the development team with the necessary API's that were needed to develop the communication functionality to acquire a fingerprint from the device.

The second significant aspect of the biometric system proposed, is the template extraction and matching functionality. The template extraction focuses on Level 2 – fingerprint recognition which covers the extraction of fingerprint minutiae on an individual fingerprint to produce a template, fingerprint images are not directly compared.

Figure 6 Fingerprint Minutiae classification

SourceAFIS is a software library for human fingerprint recognition. It can compare two fingerprints 1:1 or search a large database for matching fingerprint. It takes raw fingerprint images on input and produces matching score on output, for development of the fingerprint template extraction and matching the development team have made use of this library.

FingerprintFetch Implementation

The development of fingerprint acquisition requires the inclusion of the API files provided by Futronic-Tech, which are (ftrScanAPI.dll, ftrScanAPI.h, ftrScanAPI.lib). These files were included into the solution though the configuration options, which enabled the additional library directories and dependencies to be declared, which declared a location pointer to the files to then be included in the program.

A simple processing directive “#include” allowed for the header file to referenced for usage with the program.

The FingerprintFetch program has been developed to include the communication between hardware and software, allowing fingerprint acquisition. When the program is executed, it creates an USB device handle object.

```
//Variables
FTRHANDLE dHandle;
PBYTE pBuffer;
FTRSCAN_IMAGE_SIZE ImageSize;
FTRIMGPARMS ImgParm;
FILE *fptr;
```

Figure 7 Program Variables

```
dHandle = ftrScanOpenDevice();
if (dHandle == NULL) {
    cout << "device not conntected";
}
else {
    cout << "device conntected \n";
    cout << "\n";
}
```

Figure 8 Device handle initialisation

From this handle, the image parameters can be determined which is necessary for producing a valid fingerprint image, such as height, width, and image size. The image size information is used to initialise a data buffer, which will hold the raw image data in of the type BYTE.

```
//set buffer size to size of image from device
ftrScanGetImageSize(dHandle, &ImageSize);
pBuffer = new BYTE[ImageSize.nImageSize];
```

Figure 9 Initialisation of the Data Buffer

Before the fingerprint is acquired the development, team had to consider the quality and security aspects of acquisition, we enabled the Live-Fingerprint-Detection feature on the device which eliminates the possibility of a fake fingerprint being used to gain access, also by enabling fingerprint image enhancement and background exclusion on the device we increase the quality of the fingerprint image received from the device.

```
//options - live fingerpirnt detection, improve image and eliminate background
ftrScanSetOptions(dHandle, FTR_OPTIONS_CHECK_FAKE_REPLICA, FTR_OPTIONS_CHECK_FAKE_REPLICA);
ftrScanSetOptions(dHandle, FTR_OPTIONS_IMPROVE_IMAGE, FTR_OPTIONS_IMPROVE_IMAGE);
ftrScanSetOptions(dHandle, FTR_OPTIONS_ELIMINATE_BACKGROUND, FTR_OPTIONS_ELIMINATE_BACKGROUND);
```

Figure 10 Security and Quality features being enabled.

The next part of the program was to enter a while loop on the condition that a fingerprint was not present, if fingerprint is present execution of the remaining code can resume. The retrieval of the fingerprint is then called returning a raw byte image stream to the buffer, at this point the handle to the fingerprint scanner device is closed, and will only reopen when the process is restarted for next fingerprint acquisition.

```
//loop while fingerprint is not present, then continue when it is.
    while (!ftrScanIsFingerPresent(dHandle, NULL)) {

    }
//function call that gets an image from the fingerprint device and saves it to a buffer
    ftrScanGetImage2(dHandle, 3, pBuffer);
//close communication the device when the image has been recieved
    ftrScanCloseDevice(dHandle);
```

Figure 11 Fingerprint not present loop, get image and device close.

The acquired fingerprint image is then returned to the WinForms, for enrolment on the system or for fingerprint matching to grant or deny system access.

FingerprintFetch.exe access from WinForms application.

The FingerprintFetch executable file has been developed in C++ as the API's provided by Futroinic_Tech prevented the development team from developing inside the WinForms application, call to the FingerprintFetch.exe inside the WinForms application has been developed to allow use of the FingerprintFetch.exe, illustrated below in figure.

```
ProcessStartInfo startInfo = new ProcessStartInfo();
startInfo.FileName = "\\FingerprintFetch\\Debug\\FingerprintFetch.exe";
startInfo.Arguments = "";
var proc = Process.Start(startInfo);
proc.WaitForExit();

fingerprintFilePath = "test10.bmp";
```

The ProcessStartInfo variable initialises a new process, to which in the next line adds a string reference to the process that we wish to start, that being the FingerprintFetch executable file. The process then is called to start, in the background now the process of fingerprint acquisition will have begun. The WinForms application then waits for the process to exit. With success of the process, and fingerprint image will then be available to the WinForms application.

Fingerprint Template Extraction and Matching

The fingerprint template extraction and matching function, does exactly that. It handles the execution of fingerprinting acquisition, fingerprint minutiae extraction and matching, the function iterates through the EmployeeDetails database table. And if there is a match returns the employees details on successful access.

Function declaration, that is called when the fingerprint terminal user interface is running.

```
public void getFingerprintandMatch()
```

Figure 12 getFingerprintMatch function

The function first executes the FingerprintFetch.exe, see process explanation above, a fingerprint acquisition take place.

The fingerprintFetch.exe will return a fingerprint image to this function, it is then initialised to a fingerprint object, which is then initialised to a person object, a person can have many fingerprints.

```
//Create Person Object
Person scannedPerson = new Person();

/*Create Fingerprint Object
 *Initalising with fingerprint image created from FingerprintFetch.exe
 */
Fingerprint scannedFP = new Fingerprint();
scannedFP.AsBitmap = new Bitmap(fingerprintFilePath);

//Add fingerprint object to Person object
scannedPerson.Fingerprints.Add(scannedFP);
```

Figure 13 Initialising a person and fingerprint object

Communication to the MySQL server is then executed, a command to select from the EmployeeDetails table is passed, the dataset is initialised and added to a data table variable.

```
using (var cmd = new MySqlCommand("Select * from EmployeeDetails ", conn))
 {
     MySqlDataAdapter daFinger = new MySqlDataAdapter(cmd);
     DataSet dsFinger = new DataSet("FingerSearch");
     daFinger.MissingSchemaAction = MissingSchemaAction.AddWithKey;
     daFinger.Fill(dsFinger, "Fingers");

     DataTable tblArea;
     tblArea = dsFinger.Tables["Fingers"];
```

Figure 14 MySQL communication to get EmployeeDetails table dataset.

The SourceAFIS software library contains the AfisEngine, this object holds the functionality to extract a template from the fingerprint and perform matching to determine if two fingerprint are from the same person.

The AfisEngine, Person and Fingerprint variables are declared ahead of extraction and matching.

```
//Fingerprint engine that extracts template and performs matching
AfisEngine abc = new AfisEngine();

//Person and fingerprint declaration for db BioMarker record
Person dbPerson;
Fingerprint dbFingerprint;
```

Figure 15Declartion of AfisEngine, Database person and their fingerprint object

An enhanced for loop is then initialised, the purpose of this is to iterate through each row on the data table, the fingerprint is extracted at each iteration which is then initialised to the fingerprint object previously declared, then the fingerprint object is initialised with the person object. On each iteration of the database table, this fingerprint, and person object represents and employee and their fingerprint biometric.

```
foreach (DataRow drCurrent in tblArea.Rows)
 {
    Byte[] byteBLOBData = new Byte[0];
    byteBLOBData = (Byte[])(drCurrent["BiometricMarker"]);

    MemoryStream stmBLOBData = new MemoryStream(byteBLOBData);


    //Person and Fingerprint intialsation
    dbPerson = new Person();
    dbFingerprint = new Fingerprint();

    //Fingerprint is set with bitmap from database and added to its person object
    dbFingerprint.AsBitmap = new Bitmap(Image.FromStream(stmBLOBData));
    dbPerson.Fingerprints.Add(dbFingerprint);
```

Figure 16 Iterate data table for employee fingerprint and initialise to fingerprint object, and initialise to the employee person object

Extraction of fingerprint minutiae is then carried out on the scanned fingerprint waiting to be matched and the fingerprint selected from the database.

```
//Fingerprints templates extracted for matching
abc.Extract(scannedPerson);
abc.Extract(dbPerson);
```

Figure 17 Fingerprint Minutiae extraction

The two fingerprints are then ready for matching, the AfisEngine calls a function to Verify fingerprints with the 2 person objects being passed as parameters to the function, the result is returned to a float object, which is the score on how similar the two fingerprints are. A Boolean match is declared and is initialised true or false on the matching threshold defined.

```
//Fingerprints are match and a score termines the similiarlites.
float score = abc.Verify(scannedPerson, dbPerson);

//Determine match on threshold 0
bool match = (score > 75);
```

Figure 18 Fingerprint match and similarity score

If a fingerprint match has been found in the database, the user's details are then returned to the fingerprint terminal interface and a break in the execution of the process is called, as to stop searching the database for a match the fingerprint scanner will flash green to notify the user of a successful match and access is granted. Else the loop continues until the end of the database table is reached, the system terminal will be notified of failed match and the fingerprint scanner will flash a red led to notify the user of access denied.

```
//if match set terminal field to show employee then break from loop else continue incremeting count
if (match)
{
    int id = (int) drCurrent["Id"];
    //fetch match
    textBox2.Text =((String)drCurrent["Id"]);
    textBox1.Text = ((String)drCurrent["Forename"]);
    textBox3.Text = ((String)drCurrent["Surname"]);
    textBox1.Text = ((String)drCurrent["AccessLevel"]);

    byteBLOBData = (Byte[])(drCurrent["Photo"]);
    stmBLOBData = new MemoryStream(byteBLOBData);
    pictureBox1.Image = new Bitmap(Image.FromStream(stmBLOBData), pictureBox1.Width, pictureBox1.Height);

    break;
}
```

Figure 19 Fingerprint match - Return details to terminal

```
ftrScanSetDiodesStatus(dHandle, 255, 0);
std::this_thread::sleep_for(std::chrono::milliseconds(200));
ftrScanSetDiodesStatus(dHandle, 0, 0);
```

Figure 20 Fingerprint scanner led function - Flash Green - Signifies access granted

```
ftrScanSetDiodesStatus(dHandle, 0, 255);
std::this_thread::sleep_for(std::chrono::milliseconds(200));
ftrScanSetDiodesStatus(dHandle, 0, 0);
```

Figure 21 Fingerprint scanner led function – Flash Red – Signified a failed access attempt

Opening EmployeeMaintenance From SearchEmployees

When a User searches for an employee they have the option to open the selected employee's maintenance form and allow them to edit data. This is done by first launching a blank copy of the EmployeeMaintenace from SearchEmployees, the EmployeeMaintenance form is set to always search for an Id from the SearchEmployeesForm during start-up.

```
private void button3_Click(object sender, EventArgs e)
{
    // Set employeeId so that employee maintenance can grab it to search using the selected Id.
    employeeId = dataGridView1.CurrentRow.Cells[0].Value.ToString();

    // Launch employee Maintence.
    this.Visible = false;
    EmployeeMaintenance searchEmployeeForm = new EmployeeMaintenance();
    searchEmployeeForm.ShowDialog();
    this.Visible = true;
}
```

Figure 22 SearchEmployee Form

```
// Checks if the user has opened the maintenance form from search employees.
public EmployeeMaintenance()
{
    // Grab the employee Id from SearchEmployees and use it  to search for employee.
    string checkIdFromSearch = SearchEmployee.employeeId;
    InitializeComponent();

    if (checkIdFromSearch != null)
    {
        // If the employee Id from searchEmployees is not null the form will search using the Id provided by SearchEmployees.
        textBox10.Text = checkIdFromSearch;
        Search(checkIdFromSearch);
    }

    // Clear any files in the Temporary photos to avoid errors.
    Array.ForEach(Directory.GetFiles(@"\BiometricDb\BiometricDb\WindowsFormsApplication1\TemporaryEmployeePhotos\"), File.Delete);
    Array.ForEach(Directory.GetFiles(@"\BiometricDb\BiometricDb\WindowsFormsApplication1\TemporaryFingerPhotos\"), File.Delete);


}
```

Figure 23 EmployeeMaintenance Form

Exporting results from SearchEmployees and ActivityReport to Microsoft Excel

When a user searches for employee activities or for employees in the company the results that are returned are loaded onto the datagrid box of the respective form. The user can then export the results of the datagrids to excel as a printable report.

```
    private void button4_Click(object sender, EventArgs e)
    {
        // Creating Excel Application.
        Microsoft.Office.Interop.Excel._Application app  = new Microsoft.Office.Interop.Excel.Application();

        // Creating new WorkBook within Excel application.
        Microsoft.Office.Interop.Excel._Workbook workbook =  app.Workbooks.Add(Type.Missing);


        // Creating new Excelsheet in workbook.
         Microsoft.Office.Interop.Excel._Worksheet worksheet = null;

       // See the excel sheet behind the program.
        app.Visible = true;

       // Get the reference of first sheet. By default its name is Sheet1.
       // Store its reference to worksheet.
        worksheet = workbook.Sheets["Sheet1"];
        worksheet = workbook.ActiveSheet;

        // Changing the name of active sheet.
        worksheet.Name = "Exported from Employee Search";

        // Add title to the report.
        DateTime dateTime = DateTime.UtcNow.Date;
        worksheet.Cells[1] = "Employee Search Report        " + dateTime.ToString("dd/MM/yyyy");
        worksheet.get_Range("A1", "A1").Font.Size = 14;
        worksheet.get_Range("A1", "A1").Font.Bold = true;

        // Storing header part in Excel.
        for(int i=1;i<dataGridView1.Columns.Count+1;i++)
        {
            worksheet.get_Range("A1", "D3").Font.Size = 12;
            worksheet.get_Range("A1", "D3").Font.Bold = true;
            worksheet.Cells[3, i] = dataGridView1.Columns[i-1].HeaderText;
            worksheet.Rows[3].Columns.Autofit();
        }

        // Storing Each row and column value to excel sheet.
        for (int i=0; i < dataGridView1.Rows.Count; i++)
        {
            for(int j=0;j<dataGridView1.Columns.Count;j++)
            {
                worksheet.Cells[i + 4, j + 1] = dataGridView1.Rows[i].Cells[j].Value.ToString();
                // Auto fit columns for any size of text.
                worksheet.Rows[i + 5].Autofit();
                // Apply border to cell to form a grid.
                worksheet.Cells[i + 4, j + 1].Borders.Color = System.Drawing.Color.Black.ToArgb();
            }
        }

        // Sets email Column to autofit text.
        worksheet.Columns["D"].AutoFit();
    }
}
```

Launching FingerprintFetch from EmlpoyeeMaintenance

When a user selects the Register fingerprint button on the EmployeeMaintenance form the FingerprintFetch project is called to collect the employee's fingerprint.

```
private void button_FingerReg_Click(object sender, EventArgs e)
{
        // Launch FingerprintFetch.
        ProcessStartInfo startInfo = new ProcessStartInfo();
        startInfo.FileName = "\\FingerprintFetch\\Debug\\FingerprintFetch.exe";
        startInfo.Arguments = "";
        var proc = Process.Start(startInfo);
        // Stall the code at this point and wait for FingerprintFetch to finish.
        proc.WaitForExit();
        // fetch the saved fingerprint Image from its location.
        fingerprintFilePath = "test10.bmp";
        // Load fingerprint image to PictureBox2 control.
        pictureBox2.Image = Image.FromFile(fingerprintFilePath);

        // Create a new Bitmap with the proper dimensions to fit inside the fingerprint picturebox.
        finalFingerImg = new Bitmap(pictureBox2.Image, pictureBox2.Width, pictureBox2.Height);

        // Center the new image.
        pictureBox2.SizeMode = (PictureBoxSizeMode.CenterImage);

        // Set the new image into the fingerprint picturebox.
        pictureBox2.Image = finalFingerImg;
        pictureBox2.Show();

}
```

Converting Images to Binary "Blob" Data

Images such as employee photos or fingerprint images are displayed in a pictureBox. When saving to the database however it is necessary to convert these images to a binary value to save space on the database as well as improve the speed of searches, these binary values are then stored on the database. Below is an example of how a user's photo is first converted then saved on the EmployeeMaintenance form:

```
if (pictureBox1.Image != null)
{
    //converts picturebox1(employee photo) image into blob data for storage on the db
    //opens a fileStream to begin read/write operations
    FileStream fsPhotoBLOBFile = new FileStream(photofilepath, FileMode.Open, FileAccess.Read);
    //create a byte variable
    Byte[] bytBLOBData = new Byte[fsPhotoBLOBFile.Length];
    //read the block og byte in the stream and assign it to the byte variable
    fsPhotoBLOBFile.Read(bytBLOBData, 0, bytBLOBData.Length);
    fsPhotoBLOBFile.Close();
    //set byte variable as parameter of the sql command
    cmd.Parameters.AddWithValue("@photo", bytBLOBData);
}
```

Converting Binary "Blob" Data to Images

When pulling Images from the database it is necessary to convert the Binary data stored on the database back into an image. Below is an example of how the employee photo is converted to an image and set in a pictureBox on the EmployeeMaintenance form.

```
Bitmap finalFingerImg,finalPhotoImg;
// Filepaths for photos temporaily saved to folders, these are cleared during cleanup.
string photofilepath = @"\BiometricDb\BiometricDb\WindowsFormsApplication1\TemporaryEmployeePhotos\employeePhoto.jpg";
```

```
    if (drCurrent["Photo"].ToString() != "")
    {
        // BLOB is read into Byte array, then used to construct MemoryStream, then passed to PictureBox.
        Byte[] byteBLOBData = new Byte[0];

        byteBLOBData = (Byte[])(drCurrent["Photo"]);

        MemoryStream stmBLOBData = new MemoryStream(byteBLOBData);
        pictureBox1.Image = Image.FromStream(stmBLOBData);

        // Create a new Bitmap with the proper dimensions.
        finalPhotoImg = new Bitmap(pictureBox1.Image, pictureBox1.Width, pictureBox1.Height);

        // Center the new image.
        pictureBox1.SizeMode = PictureBoxSizeMode.CenterImage;

        // Set the new image.
        pictureBox1.Image = finalPhotoImg;

        // Save the Image in a temporary folder.
        pictureBox1.Image.Save(photofilepath);

    }
```

As shown in the last line of code the image is finally saved in a temporary folder, this is to ensure that the system will reupload the Image once the user has either cancelled or edited information on the EmployeeMaintenance form. This temporary folder is wiped during each start-up of the form and during the cleanup() which resets the form when a user either saves or clears the form. Wiping the folder is done to avoid errors.

Significant aspects of Android application

Shown below is the login functionality that allows the user access to the app. As can be seen is a simple system that sends a login request to the server with email/password details and the server sends a response containing a Boolean value indicating if it was successful or not. If the login was a success the user is logged in, if not they are presented an error message.

```
bLogin.setOnClickListener((v) → {


        final String email = etUsername.getText().toString();
        final String password = etPassword.getText().toString();
        // This repsonse listener will listen for the response to the request send when the button is clicked
        Response.Listener<String> responseListener = (response) → {


                try {
                    // the response is converted to JSON and the success variable indicating whether the request failed or succeed is read
                    JSONObject jsonResponse = new JSONObject(response);
                    boolean success = jsonResponse.getBoolean("success");

                    // message dialog presented to the user if the request failed, otherwise the user is logged in and passed to the main menu activity
                    if(success){

                        Intent intent = new Intent(MainActivity.this, MainMenu.class);
                        // the response is sent with the intent to the main menu activity so it can read details of the current user
                        intent.putExtra("Json", response);

                        MainActivity.this.startActivity(intent);

                    }else{
                        AlertDialog.Builder builder = new AlertDialog.Builder(MainActivity.this);
                        builder.setMessage("Login Failed!")
                                .setNegativeButton("Retry", null)
                                .create()
                                .show();
                    }

                } catch (JSONException e) {
                    e.printStackTrace();
                }
        };
```

Here the functionality for creating a meeting is shown, the meeting details are read from the edit texts and sent to the server, then a response is read indicating if the meeting was created successfully or not.

```
bCreateMeeting.setOnClickListener((v) → {
        try {
            final String meetingTitle = etMeetingTitle.getText().toString();
            final String meetingDescription = etMeetingDescription.getText().toString();
            final String meetingDate = String.valueOf(String.valueOf(dpMeetingDate.getYear()) + "-" + String.valueOf(dpMeetingDate.getMonth()) + "-" + dpMeetingDate.getDayOfMonth());
            final String meetingTime = String.valueOf(tpMeetingTime.getCurrentHour()) + ":" + String.valueOf(tpMeetingTime.getCurrentMinute());
            final String ownerId = jsonResponse.getString("userId");
            String location = sChooseLocation.getSelectedItem().toString();
            final String locationId = location.substring(0, location.indexOf(" "));

            // This repsonse listener will listen for the response to the request send when the button is clicked
            Response.Listener<String> responseListener = new Response.Listener<String>() {
                @Override
                public void onResponse(String response) {


                    try {
                        // the response is converted to JSON and the success variable indicating whether the request failed or succeed is read
                        JSONObject jsonResponse = new JSONObject(response);
                        boolean success = jsonResponse.getBoolean("success");

                        // message dialog presented to the user based on whether the request was successful or not
                        if (success) {
                            AlertDialog.Builder builder = new AlertDialog.Builder(CreateMeetingActivity.this);
                            builder.setMessage("Meeting Created!")
                                    .setNegativeButton("Ok", null)
                                    .create()
                                    .show();
                        } else {
                            AlertDialog.Builder builder = new AlertDialog.Builder(CreateMeetingActivity.this);
                            builder.setMessage("Create Meeting Failed!")
                                    .setNegativeButton("Ok", null)
                                    .create()
                                    .show();
                        }

                    } catch (JSONException e) {
                        e.printStackTrace();
                    }
                }
            };
```

Shown below is the functionality for searching for users by name, a name is sent to the server and the server responds with all names in the database that are like the name sent. The app reads this response and places the names in a ListView displayed to the user which allows the user to click a name and view further details for that user/invite them to a meeting.

```
// This repsonse listener will listen for the response to the request send when the activity is created
Response.Listener<String> responseListener = (response) → {

        // as multiple results are sent from the server they need to be split into individual objects in order to be read and parsed so the response is split
        // and placed into an array, the split is done on the { character as this is the opening character of each new JSON array.
        // e.g. { "jsonexample1": "example"}{"jsonexample2: "example"} the two separate json objects are split and placed into a string array
        String[] parts = response.split("\\{");
        // this array will contain the json objects once they have been converted to json from the string
        JSONObject[] jsonParts = new JSONObject[parts.length];
        for (int i = 0; i < parts.length; i++) {
            if (i > 0) {
                // as the split function removes the { character from the first json object it needs to be replaced in order for the json conversion to work
                parts[i] = "{" + parts[i];
                try {
                    // the string array is traversed and each string is converted to json and placed in the json array
                    JSONObject json = new JSONObject(parts[i]);
                    jsonParts[i] = json;
                } catch (JSONException e) {
                    e.printStackTrace();
                }
            }
        }
        // this string array will contain the results of the search that the server responded with
        String[] usersList = new String[jsonParts.length - 1];
        // the json array is traversed and each item is placed into the string array to be used by the list view
        for(int i = 1; i < jsonParts.length; i++){
            JSONObject jsonPart = jsonParts[i];
            try {
                usersList[i - 1] = jsonPart.getString("userId") + "-" + jsonPart.getString("userForename") + " " + jsonPart.getString("userSurname");
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }

        // the array adapter to be used by the list view is created and set
        ArrayAdapter<String> arrayAdapter = new ArrayAdapter<~>(
                SearchUsersActivity.this,
                android.R.layout.simple_list_item_1,
                usersList );

        lvUsers.setAdapter(arrayAdapter);
};

// the request is created using the name that was passed through with the intent and sent using Volley
SearchUsersRequest usersRequest = new SearchUsersRequest(name, responseListener);
RequestQueue queue = Volley.newRequestQueue(SearchUsersActivity.this);
queue.add(usersRequest);
```

Testing Strategy

Testing was undertaken continuously throughout the entire development process. Any issues discovered were then immediately resolved and the regressively retesting to ensure that the issue was resolved. Throughout the testing process a multitude of testing techniques were used to ensure that as many bugs as possible were found and fixed. These techniques utilised are described below:

Black Box Testing

Black box testing is a way to test all high-level functionality. This was completed out at the end of each sprint and ensured that development did not progress whilst bugs and errors were present.

Acceptance Testing

Acceptance testing was carried out regularly and was completed by developers. The Acceptance Testing conditions and results are shown in Appendix 1.1.

Unit Testing

Unit tests were utilized throughout the Windows Form Application, especially testing to ensure that data entered by the user was correctly saving to the database on the server. The use of Visual Studio Test Explorer provided a flexible and efficient way to run unit tests and view their results in Visual Studio. With the use of this unit testing framework it is possible to create unit tests, run them, and report the results of these tests. Re-running unit tests provided the ability to test that code is still working correctly after changes were made.

Integration Testing

Integration Testing was completed every time a significant component was finished and was key to ensuring each new piece of code that was added to the system was compatible with each other. Integration testing was very important when connecting the server to both the Mobile and Windows Form applications. The connections to the server had to be carefully tested as both the Mobile and Windows Form applications will not be able to run without the server. If a bug were to occur here it could potentially disable both applications.

Evaluation & Conclusion

The project team feel that the system has been produced to meet all specifications and requirements, evaluating the Window Forms application and fingerprint acquisition, template extraction and matching functionality of the project to be a success.
The team feel that one of the key contributions to the success of this Biometric project was the involvement of Citi and their Upstart program. Regular meetings with the team's Mentors Aaron and Neil as well as project pitches and presentations to members of Citi offered us with the much need feedback and constructive criticism of the system which helped the team to re-evaluate components of the system as well as build upon the project in the best possible direction. The team met with Fabian Campbell and the team at Liopa, a local based competitor to our proposed system in the Biometric market, they provided us with much needed insight to Biometrics with their work and provided the team with fresh ideas and concepts that could be used to better the development team and in turn the project. The feedback from the members of Citi at every stage of development ensured the finished product was complete, polished, and ready for release.

Having chosen the biometric badging project at the beginning, I was excited about the prospect of working with a person's biometric marker, something I had never experienced before.
I spent a significant amount of time in the beginning in researching the subject of fingerprint biometric recognition which I found very interesting, from this I gained the knowledge of what was needed to acquire a fingerprint, extract features to be used for the matching process and I hope to be able to put this knowledge to good use in future endeavours.

My task within the group project was to develop code to acquire a fingerprint from a fingerprint scanner, I also was tasked with the fingerprint matching feature.
I began firstly with the fingerprint matching feature, through my research I came understand that once a fingerprint is acquired it goes through pre-processing image enhancement which is aimed at improving the quality of the fingerprint, this process is focused at improving the quality of the fingerprint image before a template is extracted. I chose to work on MATLAB for this part of development. MATLAB is a powerful mathematical and scientific coding platform, which provided me with the relevant libraries and functions need to improve an image. As I had never worked with MATLAB before I enjoyed learning it, I found it very easy to use but I found great difficulty in trying to improve the quality of an image to the standard required to extract a fingerprint template. Having met with Liopa, a local biometric company they advised me that the internet holds the relevant API's required for progress on my task, the code I developed in MATLAB was not used in the developed product for that reason as I came to use a fingerprint extraction and matching API.
In the project, I also took on the task of developing the fingerprint acquisition functionality from a fingerprint scanner. As the chosen fingerprint came supplied with the relevant API to develop communication between hardware and software I very much enjoyed the task of writing the code to enable this communication, and sending requests for a fingerprint image.
Throughout the project there were several difficulties;
From a project perspective, we didn't have a real-life client as compared to other groups, so we as a team created our own system requirement list and worked from that, as we were a part of the Citi upstart program the mentors at Citi, they at best tried to influence has a

client and for that we were grateful, influencing creative and technical thinking, and offering up suggestions directing on the right path.

As to be expected in any group work, you have the worry of being grouped with members that did not contribute as much as everyone else, as this was the case throughout the course of this project, communication, and engagement with certain members of the team was very poor. Ultimately, I felt that I had to take on more than enough tasks during the project development to compensate this lack off.

For a time, I felt the scope of my system components of fingerprint matching and acquisition was beyond me, but I took imitative in asking the fingerprint scanner company for technical support in communicating with their device, also by talking with technical professional in the field of biometrics the suggestions and insight from that contact, bettered my understanding in USB communication and fingerprint matching.

I successfully completed the components needed for the final system, I developed the fingerprint acquisition functionality in C++ and successfully with the help of Fergal could plug the execution of this within the WinForms application. From inside the WinForms application I developed the functionality to extract a match the fingerprint with success, I can conclude that the project was a success for me, slowly but surely things fell into place and we produced a fully operational biometric system, with all the functionality required.

Future Considerations

If in the future, the project was to be extended, I would like to consider other biometric recognition techniques, one that stood out through my own research was Gait recognition, which is the process of identifying a person by the way they walk, its suggested that a fingerprint is as unique to the individual as there walk. I would also like to consider the development for more than one attached fingerprint scanner to the system, due to hardware constraints we only had access to the one fingerprint device.

I believe that the project was a success, especially in the areas that I was responsible which were the Windows Form manager tools, Terminal and integration of Adam's fingerprint fetching and matching software.

Where the project was disappointing for me was the lack of a real client with whom the team could involve in the project and communicate with to receive feedback, for this the team had to rely mainly on the Citi mentors to act as our clients. I also felt that some members of the team also had trouble at first as they had no experience with the coding language or software that was used to create the system or the languages that they were developed with. As I had worked with C# and C++ before I did not have this issue and this worry was short lived as we began to share our experiences and knowledge with one another.

Another area that I personally had no experience with was using GitHub to save my code to, however through tutorials and the GitHub forums I quickly set up a repository and uploaded my work regularly. The link to the repository can be found in the appendices.

I have never been involved in an agile development approach and this new understanding of how a project can go from a concept, right through to a developed system will expand has expanded my skill sets. I have learned that software engineering is more than writing code; as project management, user engagement and testing are all equally as important. This newly acquired knowledge has provided me with new skills which will prove useful when working in industry.

Hardware and Software Evaluation

Regarding the Window Forms Application, I believe that key aspects to software, namely the Access Terminal and fingerprint registration on the Employee Maintenance tool should only be used when a fingerprint scanner is connected to the computer that is running the project as these forms rely heavily on the scanner. However other features of the Windows forms, such as viewing reports does not require this hardware. The terminal would also be installed at the entrance to each area rather than included in the Window Form Application. The idea for these would be that each entrance would have a raspberry pi with the software installed and a scanner plugged into it.

Future Enhancements

If the project was developed further I have some ideas that I would like to implement, these ideas can be found below:

- The implementation of an eWallet system. The idea for this would be that users could top up the wallet with money and use their fingerprint to pay for food or drink inside the company.
- Updates to the Activity report to allow users to search for employees who have attempted to access areas above their access level.
- UI improvements, such as smoother transitions between screens.
- Edit of the Employee maintenance to allow users to save employee information after they first scan their own finger to authorise the save.

James Dickinson - 40082743

I believe that the android application that I developed contains functionality that is valuable to the user. As this was my first time developing an android application from scratch there was a lot of trial and error on my part in getting things to work the way they were intended which slowed down developed quite significantly at the start. Although things did pick up some pace as the project progressed I feel that my lack of experience meant that the application is lacking in certain areas.

An area of the project I feel could be improved is more communication from the client as there was no direction or criticism offered. I feel that if we were given even a simple list of requirements for the system or feedback at each our presentations for the direction of development and where they would like improvements to be made it would have made the planning of development tasks much easier. As it was the project was very open ended and left for us to decide what direction to take development whereas I feel that criticism and feedback from a client could have helped guide the project and keep things more on track.

Hardware and Software Evaluation

I believe that although the android application offers functionality that is valuable to the user it could be improved. I feel the app could offer more to the user than just the meeting

functionality, things such as live location reporting or a simple message board would have added a lot of value to the user. Due to time constraints, I was not able to do everything that I wanted to achieve with the application.

Future Enhancements

If I was to continue to develop the application I feel there would be value in the following areas of development:

- Improvement to validation on the creation of meetings, as it stands there is no way to confirm that a location is available for the times the user selects.
- addition of a simple message board that can be viewed by all users of the organisation allowing them to post important information.
- addition of live location reporting allows locations to be viewed and for the user to see who is in a location

Lauren McGlennon - 40134508

I believe that overall the project was a success. the projects I have worked on previously have been more scrum and daily sprint based in how we get the stuff done. It was nice to can experience working in a different way. usually in my group projects I end up being project manager, so it was nice to take a step back and work as a normal team member. To see someone else project management skills and learn some tips, seeing different ways of doing things and communicating with the team. I had a bit of personal circumstances to deal with during the working of this project. Unfortunately, at times I feel I may have struggled to balance my circumstances and working on the project.

I found it hard to not be working with a real client and be given exact product requirements etc. as that is what I have previously been used to. Although this allowed the team and I to have more scope to choose where exactly we wanted to go on the project. However, it would have been nice to be able to work with someone and receive feedback on whether the project was to their satisfaction, if they wanted anything extra taken out, added etc. To be able to give monthly updates to them on where we were in the project and perhaps have them fill out feedback questionnaires.

Future Enhancements

A couple of things for the future if we had more time to work on the project:

The managers could set up an automatic report to run every month to send them an update on any new employees that have been added to the database and joined the company

Employers get an automatic notification via the app anytime someone tries to access a room or floor that they do not have access to

References

Exporting Datagrid view values to Microsoft Excel:
http://stackoverflow.com/questions/18182029/how-to-export-datagridview-data-instantly-to-excel-on-button-click

Converting Image to Binary "Blob" Data
http://stackoverflow.com/questions/9576868/how-to-put-image-in-a-picture-box-from-a-byte-in-c-sharp

Converting "Blob" Data to Image
http://stackoverflow.com/questions/14332862/how-to-convert-blob-to-image

https://github.com/F3rgal47/BiometricDb

This GitHub repository was the base of the volley requests used by the application.
https://github.com/tonikami/NEWLoginRegister

SoruceAFIS software library
https://sourceafis.angeloflogic.com/

Futronic-Tech supplied me with the APIs required for fingerprint scanner communication,
inquiry@futronic-tech.com

Appendices

What's included:

User Manual

Testing

USB memory pen – contains all project code.