Nicky Temperley

Nicky Temperley

E6155906

TMA03



# 1 Draft Project Report

## Problem Description

### 1.1.1 Background

I have been working for Northumberland County Council for 9 years now, and within it there are over 6000 employees across dozens of departments, each providing a role to the Council and the Community. One of the departments in the council is the planning department, who receive planning applications.

Planning is a customer facing Service. They deal with Planning applications from members of the Public, land owners and Developers. The type of planning applications can vary from small scale such as a simple householder extension, through to large scale residential developments, new commercial developments, opencast sites, etc. Planning Officers are given 'delegated authority' by the Chief Planning Officer which means they have authority to determine planning applications. Most applications are determined by a Planning Officer however some less straightforward or contentious applications may be referred to a Planning committee.

Planning also write Policies through consultation to guide future development of the County. This sets the framework for all planning officers to adhere to for all future applications for developments of towns and villages within the County.

Conservation areas also fall under the remit of Planning. The Conservation officers may be involved in a planning application to give opinion and guidance to a planning officer where a planning application/development is in a Conservation area. Likewise advise is also provided to residents who may wish to make changes to a historic building such as a grade II listed building.

Planning also deal with a number of planning breaches. This may be that building works are carried out without planning permission or not in compliance with their approved planning approval. The Enforcement officers will also get involved in reports from members of the public of issues such as untidy land, non-approved advertising, etc.

### 1.1.2 The Problem

The planning department have about 100 users who work out on site. Fiona Charlton, the client, is looking for the development of a system that will allow her and other managers to record which officers are on site, which site they are at, the time that they arrived and the time they estimated they will leave. The officer will time to leave the site, and if they haven’t checked in when they leave or before this time an alert needs to be sent to their line manager. See [Appendix 4.1](#_4.1_The_initial) for a brief description in the client’s words.

Enrich the body of the report by taking quotes and using them here.

Currently there is no standardised reporting procedures for this to happen, and all teams use a different way of recording the information. There is a system already available called Guardian24, but at over £5 per user per month, this isn’t a viable solution for the whole workforce and is only used for the most at-risk officers. Without a standardised solution it could be the case that a site officer has an accident and isn’t able to contact the office, and no-one would be aware that there was a problem.

The officers will need to be able to submit the check in and out times from their mobile phone, and the system must be able to handle a manual input by office staff in the case they are unable to use their phone for whatever reason.

### 1.1.3 My Proposed Solution

To solve the problem, I am planning on designing and implementing a system to record all the information required. I will use a Java GUI to make calls the SQL to the database, with a mobile app created in Cordova using HTML, CSS and JavaScript to communicate via a REST web service to the database.

The Java system will record the managers details, the officer’s details, the estimated and actual check in and out times (with the actual times being submitted through the mobile app or manually entered), the site location and any notes required for the site. Officers, or admin staff on their behalf, will selected the officer’s name, manually enter a site location, then enter estimated check in and out times for when they arrive and leave site. The actual times will be entered later through the mobile app. The GUI will check for users that have checked in at a site and haven’t checked out by the time they have estimated; in this instance, an alert will show on every running instance of the system. This will be a prompt for a manager to contact them and take further action if needed or update the system on their behalf.

The mobile interface will be simple, with only a dropdown box to select the site, and buttons for check in and check out. The user will be authenticated by the Google account that they are logged into on the phone, which is linked to their staff domain account, and the site dropdown will only show sites that they have previously typed into the system.

### 1.1.4 Alternatives to my proposal

|  |  |
| --- | --- |
| **Alternative** | **Justification** |
| Carry on the way they are currently working | Not really an option as it could be a health and safety risk if site operatives have an accident on site and no-one notices that they haven’t checked in |
| Use the app Guardian24 for ever officer | Not feasible due to the cost, it would work out over £500 per month |
| Ask a third party to develop a solution | Another cost implication, high initial cost although there shouldn’t be a monthly cost apart from maybe an annual licensing fee |
| Develop a system with different technologies to the one proposed | This is achievable, and up until recently it was a strong possibility. I have decided against this as Java is my strongest language and I feel I could deliver the best results with a language I am comfortable, rather than trying to learn a new language to achieve a suitable outcome. |
| The development team at Northumberland County Council to develop a solution | The development team are unable to develop a solution as it is outside the scope of the services they provide to the council. Resources may become available later, but currently this isn’t an option |

### 1.1.5 Resources Used

## Account of related literature

After further searching I came across an article on the IBM website called Introduction to Java Threads (Goetz, 2002). Although this is still quite outdated, I can’t find many resources on the matter, or not many newer ones anyway. I decided to read it and not only has it given me a better understanding of what threads are, but when to use them, when *not* to use them, and how to implement them. As of writing this TMA I haven’t implemented them, but I think to get around the database locking issue as mentioned in my project work, I think I need to introduce them. Having processes running on threads will stop the database locking as methods won’t run concurrently, so they won’t be accessing the database at the same time.

As also mentioned in my project work, a major issue I was having was connecting my Java program to my SQLite database, something I haven’t had to do as part of my OU studies thus far. After searching on the internet, I managed to find an article on SQLite Tutorial (n.d.), which gave me not only the download link for the driver, but also step by step instructions on how to add the driver as a library to my java project and is also the basis for the “Test Connection” method in my “DataConnect.java” class. There were other sources available for this technique, but I found these the most straightforward and user friendly.

My next issue was that I have never used SQL and Java together, so I didn’t know how to do SQL calls within the Java code. There were ample resources available, but to start with I went to the Java Docs homepage, specifically relating to the java.sql package. These guides are the ultimate resource, everything you could need is there; but personally, I found all of the information a bit overwhelming. I didn’t spend too long looking at the docs, but I did refer to them about specific methods if I came across something in my other research that I didn’t understand. For the bulk of my database interaction, I used a website I’m going to refer to as “programming notes” titled “Java Database (JDBC) Programming by Examples with MySQL“ (Hock-Chuan, n.d.). I found the notes here relevant, easy to use, and they were helpful in creating my own classes and adapting to suit my needs.

## Account of project work and its outcome

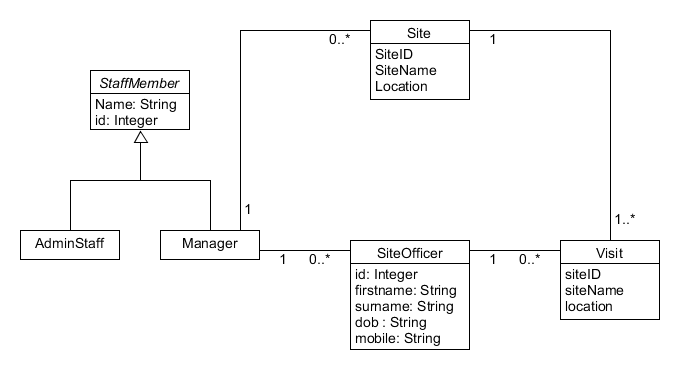
### Stage 1 – Data Collection and Project Definition

#### 1.3.1.1 Initial Client Meeting and system

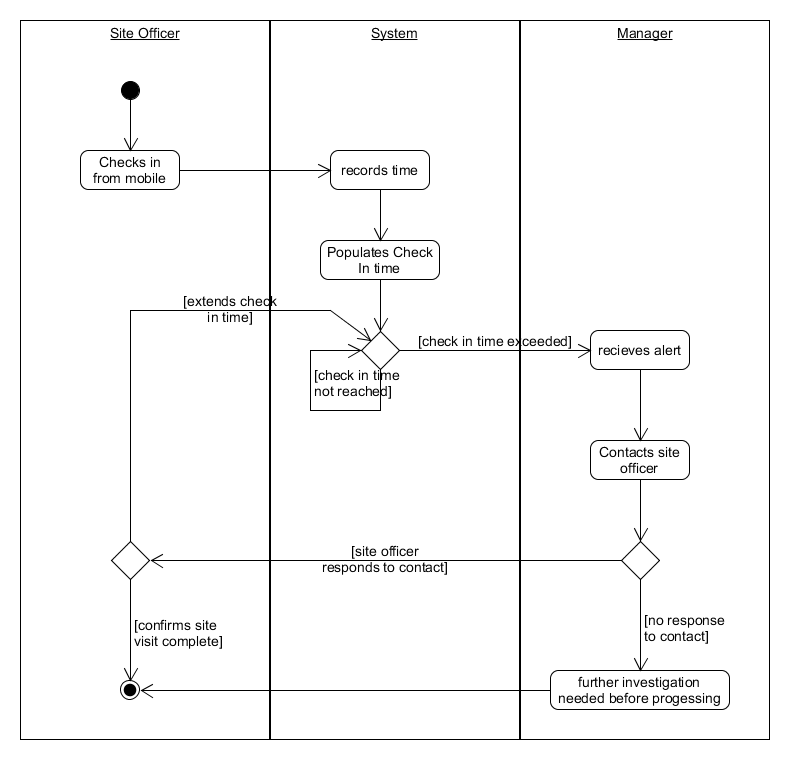
To start the project off I’ve had a meeting with Fiona Charlton on 07/03/19. It was my aim of this initial meeting to get some answers to some fundamental questions regarding the initial brief that was received in [Appendix 4.1](#_4.1_The_initial). The questions and her responses are in [Appendix 4.2](#_4.2_Initial_Meeting) .

Following this initial meeting I have managed to mock up some initial user interfaces (in Appendix 5) and I have planned a further meeting with Fiona on 24/03/2017 to get some feedback and talk her through the process of using the system. I still need some clarification on what the buttons along the bottom of the screen do (that I’ve labelled a-e) and if she needs this functionality included, and if its within the scope of this project. Once I’ve had the user interfaces confirmed I’ll be able to design them in Java using JSwing, then for the first iteration I’ll get the admin side of the system working.

Once the client is happy with the ideas so far, I’ll design the screens for adding and editing the site users and show the interactions between the different screens. At this moment, I imagine the classes needed and the associations between them will be:



After analysing the problem, my understanding is displayed in the following activity diagram:



At the end of the last TMA I had a list of questions that I wanted to ask my client. I have reproduced the list below with the response:

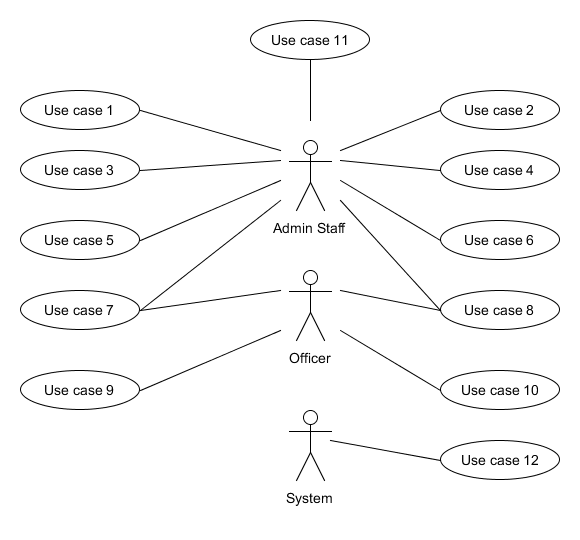
1. What is the role of the admin user? What do they do that the managers don’t and vice versa?  
   *There is no difference between the two users in regards to the function of the system, all staff will use the system, but admin staff wont be mentioned within the system. Site officers are admin staff that also perform site visits*
2. What happens in the event that the site user cant get a mobile signal, or use their phone for some reason.  
   *In this case, they tend to call or text their manager, or someone else in the office if they aren’t available. Even if a mobile signal cant be used for contact, they normally use a landline at whatever location they visit.*You may wish to create a message in your app that tells the user to call using a landline when data / mobile signal is not good.
3. What happens if a manager is not in the office when one of their team members go to site?  
   *If a manager is not available, due to annual leave etc, then another manager is appointed to that team in their absence. We’d quite like a way to move people teams in the system you’re developing to handle this if that’s at all possible.*
4. How would you expect to be able to manually edit the times of the site officers?  
   *We’d only need to edit the “actual” check in and out times, but I’d expect them to be available within the system that’s used on a desktop machine.*
5. If an estimated check out time is not specified, would it be helpful to use a default value?  
   *No, I’d like the officers to be forced to enter an estimated time please.*
6. What information do you store about the sites that the officers visit?  
   *We need to store their name, mobile number, car registration, next of kin name, and a contact number for the next of kin.*Is there any unique information you store about the users?  
   *Not at the moment, but we have the staff ID number that could be used as a unique identifier if that’s helpful?*
7. Is a site deleted once the officers have finished all their visits? Is it made inactive?  
   *The site details don’t need to be kept. We visit thousands of sites per year, so its not feasible to store all that data. We’re just using this system to view officers current visits, once they check back in, there no need to store the data about that visit.*

#### 1.3.1.2 The project so far

After reviewing all the information, and speaking with the client, I have developed to following use cases:

|  |  |  |
| --- | --- | --- |
| Reference | Name | Description |
| UC1 | Add a manager | Add a new manager to the system |
| UC2 | Edit a manager | Edit a currently stored manager from the system |
| UC3 | Delete a manager | Delete a currently stored manager from the system |
| UC4 | Add an officer | Add a new officer to the system |
| UC5 | Edit an officer | Edit a currently stored officer from the system |
| UC6 | Delete an officer | Delete a currently stored officer from the system |
| UC7 | Set estimated times | An officer sets the estimated visit times using the GUI |
| UC8 | View more details | Anyone can click “more details” on the GUI and see more details about the selected officer |
| UC9 | Check in from app | An officer uses the mobile app to check in when they get to site |
| UC10 | Check out from app | An officer uses the mobile app to check out when they get to site |
| UC11 | Manually edit actual times | If a user rings the office rather than use the app, the actual times need manually updated |
| UC12 | Alert the manager | If an Officer hasn’t checked in by the time the estimated check out time arrives, an alert will present on the screen |

From the use case, I have created a use case diagram denoting the three actors; Admin Staff, Officers and the System itself the show the responsibilities concerned:



### 1.3.2 Stage 2 - First Iteration

#### 1.3.2.1 The Protype Designs

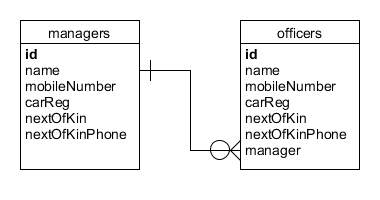
For the first iteration I have chosen to implement part of the GUI, specifically the part that allows new users to be added, edited or deleted; this will deal with UC1-UC6. I have initially developed the GUI as per [Appendix 4.3](#_4.3_Proposed_designs) and she has replied with the following comments:

* Could we get a button on the main screen, so admin staff can easily view all other details about the selected officer without having to go through the system looking for them
* The date field is irrelevant as we are only concerned with visits that are ongoing or happing on that day
* Could we have separate times on the page so we can see both the expected time on site and the actual time, that way we can see if someone has actually arrived on the site. This could be used in case someone has had an accident on the way to site. i.e. if they haven’t checked in around the time they were expecting, we can contact them to make sure everything is ok

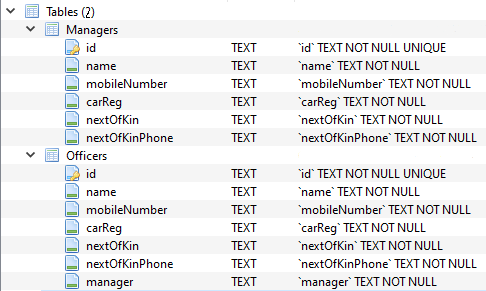
In response to the above points I have created the prototypes in [Appendix 4.4](#_Appendix_5.7_–) which include an amended front screen, and a new screen for the “more details” button. After showing these new designs to my client, she’s agreed to all the changes and has given the approval to develop them.

#### 1.3.2.2 SQLite Database

Since the first iteration is only concerned with the data concerning the site officers and the managers, my ERD is displayed below to reflect this:

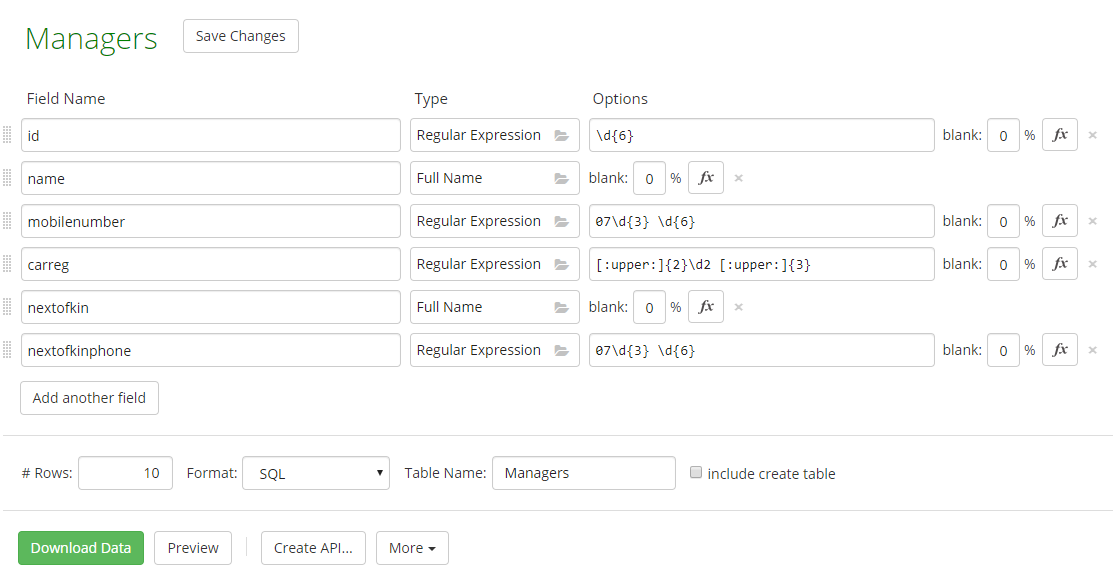


I have then used DB Browser for SQLite to create the database. This was done using the GUI, there is a create database button, from then it’s a simple as creating the tables using the buttons shown. My final table when created looks like this:



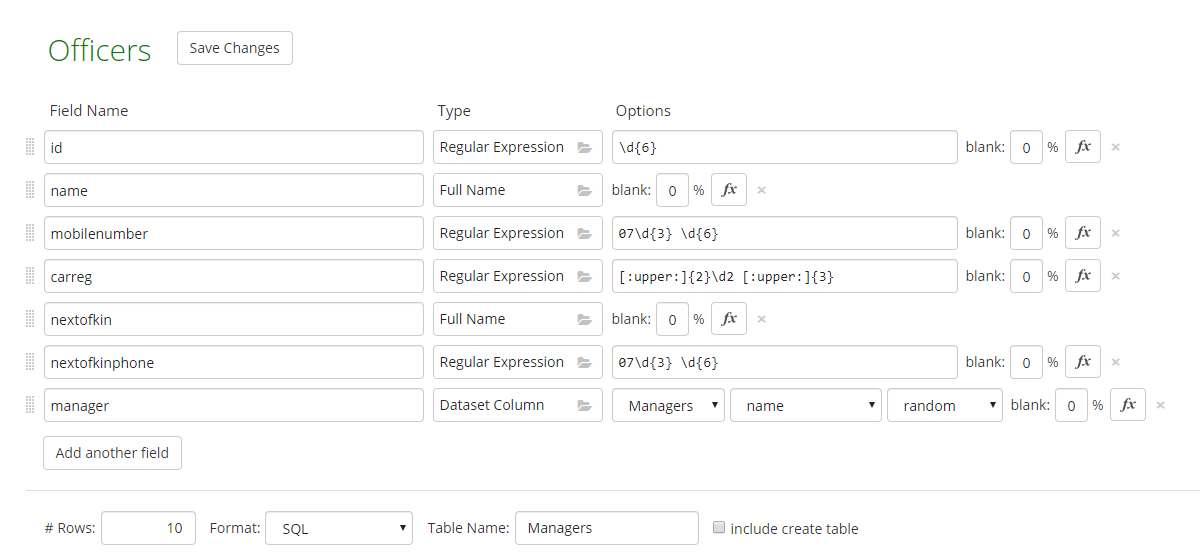
To populate the tables with test data I have used a website called Mockaroo. This website has allowed me to specify the types of data I want to create, and then provides the SQL for me to write to insert the data into the tables. The parameters I have used on the site to create the data are:

**Managers**



This gives me a 6 digit ID number (which matches our staff ID’s), a random name, a mobile number in the correct format, a car reg in the correct (post 2011) format, a random name for the next of kin and a phone number for them in the correct format.

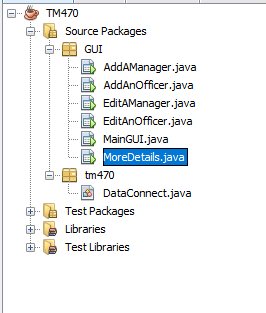
**Officers**



This gives me the same data as above, but with addition of a manager field, which randomly selects one of the previously created managers.

#### 1.3.2.3 Creating the GUI

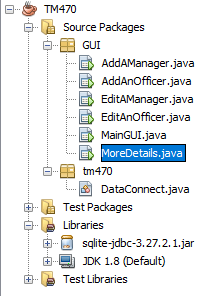
To create the GUI I have decided to use the JSwing elements of Java, as taught in M256. This will give me a drag-and-drop interface, which should allow me to fine tune the design to how its been agreed. I started by placing the elements on the page to recreate the main page and I’ll concentrate on the coding behind the objects later. I have created one main page, then an additional page to correspond to each of the buttons along the bottom of the page. All of these pages represent a Java Class, so that’s a total of 6 classes, one for each page. I have also created another class called ‘DataConnect.java’ which handles the connections to the database to make a total of 7 classes.



All the pages within the GUI package are available within [appendix](#_Appendix_5.10_–) 4.5. I have used the appendix to detail how the pages work together, and any other relevant notes are below the pictures.

#### 1.3.2.4 Problems so far

The only main problem I’ve faced up until this point was that I didn’t know how to connect my java pages to my SQLite database. Although my studies have taught me Java and SQL individually, there has never been any cross over between the two. After a bit of research on the internet, I found a tutorial (SQLiteTutorial.com, n.d.), and after downloading the SQLite JDBC driver I managed to add it to my project and this will allow me to connect my Java GUI with my SQLite database. My directory now looks like this:



#### 1.3.2.5 The code behind the pages

All of the classes I have created need code added to provide the required functionality. The below table lists the methods I will need are:

|  |  |  |  |
| --- | --- | --- | --- |
| **Class** | **Method** | **Activated by** | **Description** |
| MainGUI | btnTestConnectionMouseClicked() | Clicking on the ‘Test’ Connection’ button | Tests the connection to the database for testing purposes. To be removed in a later iteration |
| MainGUI | btnExitMouseClicked() | Clicking on the ‘Exit’ button | Displays a confirmation box asking if they want to close the system, if they click yes it closes, if they click no it returns to MainGUI |
| MainGUI | btnAddManagerMouseClicked() | Clicking on the ‘Add A Manager’ button | Displays the AddAManager screen |
| MainGUI | btnAddOfficerMouseClicked() | Clicking on the ‘Add An Officer’ button | Displays the EditAManager screen |
| MainGUI | btnEditManagerMouseClicked() | Clicking on the ‘Edit A Manager’ button | Displays the AddAnOfficer screen |
| MainGUI | btnEditOfficerMouseClicked() | Clicking on the ‘Edit An Officer Button’ button | Displays the EditAnOfficer screen |
| MainGUI | cmboOfficerMouseClicked() | Clicking on the ‘Officer’ dropdown menu | This will do an SQL call in the format “SELECT name FROM Officers” to populate the list of officers available |
| MainGUI | btnMoreDetailsMouseClicked() | Clicking on the ‘More details’ button | Displays the MoreDetails window |
| MainGUI | cmboOfficerActionPerformed() | Selecting a new officer from the officer drop down menu | Performs an SQL call in the format “SELECT \* FROM Officers WHERE name = ‘xxxx’”, where xxxx is the officer name selected in the list. |
| AddAManager | btnReturnMouseClicked() | Clicking on the ‘Return’ button | Returns to MainGUI |
| AddAManager | btnSaveMouseClicked() | Clicking on the ‘Save’ button | Performs an SQL call in the form “INSERT INTO Officers(id, name, mobileNumber, carReg, nextOfKin, nextOfKinPhone) values (?,?,?,?,?,?)", where the question marks refer to the values the user has entered on the screen |
| AddAnOfficer | btnReturnMouseClicked() | Clicking on the ‘Return’ button | Returns to MainGUI |
| AddAnOfficer | btnSaveMouseClicked() | Clicking on the ‘Save’ button | Performs an SQL call in the form “INSERT INTO Officers(id, name, mobileNumber, carReg, nextOfKin, nextOfKinPhone, manager) values (?,?,?,?,?,?,?)", where the question marks refer to the values the user has entered on the screen |
| EditAManager | btnReturnMouseClicked() | Clicking on the ‘Return’ button | Returns to MainGUI |
| EditAManager | btnDeleteMouseClicked() | Clicking on the ‘Delete’ button | Performs an SQL call in the form "DELETE FROM Managers WHERE id = \" + managerId + \" where the managerId is the manager’s ID that the user has selected in the form |
| EditAManager | btnUpdateMouseClicked() | Clicking on the ‘UPDATE button | Performs an SQL call in the form "UPDATE Managers SET name = xxxx. mobileNumber = xxxx, carReg = xxxx, nextOfKin = xxxx, nextOfKinPhone = xxxx WHERE id = \" + managerId + \" where the managerId is the manager’s ID that the user has selected in the form and xxxx are the details the user has entered on the form. |
| EditAnOfficer | btnReturnMouseClicked() | Clicking on the ‘Return’ button | Returns to MainGUI |
| EditAnOfficer | btnDeleteMouseClicked() | Clicking on the ‘Delete’ button | Performs an SQL call in the form "DELETE FROM Officers WHERE id = \" + id + \" where the id is the officers ID that the user has selected in the form |
| EditAnOfficer | btnUpdateMouseClicked() | Clicking on the ‘UPDATE button | Performs an SQL call in the form "UPDATE Officers SET name = xxxx. mobileNumber = xxxx, carReg = xxxx, nextOfKin = xxxx, nextOfKinPhone = xxxx, manager = xxxx WHERE id = \" + id + \" where the id is the officers ID that the user has selected in the form and xxxx are the details the user has entered on the form. |
| MoreDetails | btnReturnMouseClicked() | Clicking on the ‘Return’ button | Returns to MainGUI |

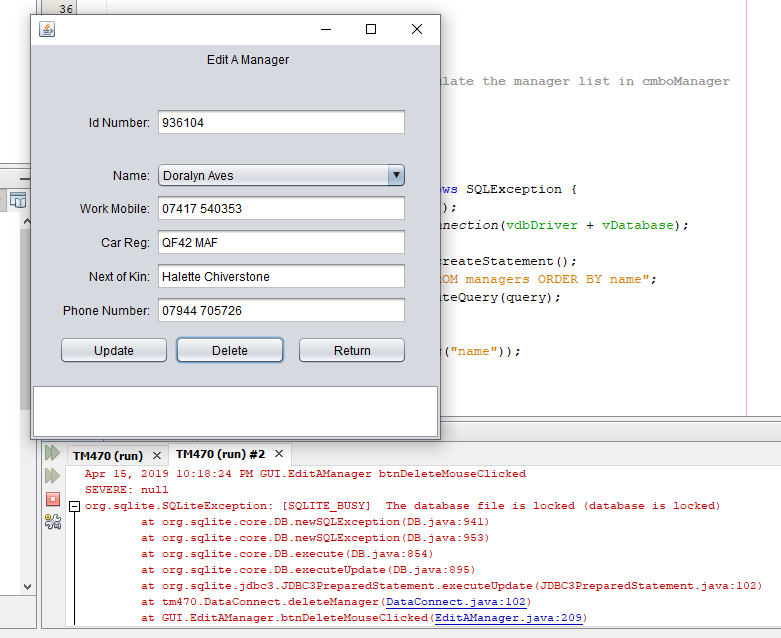
To help with the above methods, I will also create helper methods in the Data Connect class. These, and the full versions of my code can be found in [appendix 5.11](#_Appendix_5.11_–).

#### 1.3.2.6 UAT – User acceptance testing

To test my GUI I will ask my client Fiona to run the folling tests:

|  |  |  |  |
| --- | --- | --- | --- |
| **Reference** | **Name** | **Description** | **Outcome** |
| Test1 | testAddNewManager | To test the addition of a new manager, valid data entered by the user, expected success message | Success message generated, all fields cleared for next input |
| Test2 | testAddNewManager | To test the addition of a new manager, invalid data entered by the user, error message expected | Error message generated, as expected |
| Test3 | testAddNewOfficer | To test the addition of a new officer, valid data entered by the user, expected success message | Success message generated, all fields cleared for next input |
| Test4 | testAddNewOfficer | To test the addition of a new manager, invalid data entered by the user, error message expected | Error message generated, as expected |
| Test5 | testEditManager | To test deleting an existing manager, current manager selected, expected success message | Error message, database file has locked |
| Test6 | testEditManager | To test editing an existing manager, valid data entered by the user, expected success message | Not implemented yet due to error on previous method |
| Test7 | testEditManager | To test editing an existing manager, invalid data entered by the user, expected success message | Not implemented yet due to error on previous method |
| Test8 | testEditOfficer | Not implemented yet due to error on previous method | Not implemented yet due to error on previous method |
| Test9 | testEditOfficer | Not implemented yet due to error on previous method | Not implemented yet due to error on previous method |
| Test10 | testMoreDetails | After selecting a valid officer on MainGUI, all of their details should be available when the user clicks the ‘More Details’ button | Success. As expected, all the details of the currently selected officer are available when clicking on ‘More Details’ |

The tests above in Red failed, and I will be fixing this first as part of the next iteration. The error message showing on Test5 (with a random manager) is:



### 1.3.3 Stage 3 - Second Iteration

#### 1.3.3.1 Interview with the client

Before I start developing the mobile app for the officers to use on site, I arranged another meeting with Fiona to discuss the scope and usability of the project. Below are the questions I’ve asked with her responses:

Q. Can you clarify exactly what the difference is between an admin role in the office and a manager’s role. What can a manager do that an admin person can’t?

A. There is no difference, but the managers only should receive the alert.

Q. What would happen is the site officers listed manager was absent, or just away from their desk when an alert went off?

A. To avoid this, would we be able to get the screen to change to alert everyone that someone hasn’t checked in? Maybe we could install a tv in the office to display this screen only, so we could use it as a dashboard?

Q. Would the whole system not work better if that the case, designed as a dashboard? You could use it to display only the active visits, then you haven’t got to worry about the screen being cluttered with information that isn’t relevant for the purpose of this system?

A. Yes, that would be a much better idea, a dashboard only showing active visits.

Q. Would you need to interact with the new “dashboard” at all?

A. Yes, we would still need to be able to manually check in officers, as well as the normal admin functions like adding new officers etc.

Q. And what about the site information. Does this need to be retained?

A. It would be helpful if it could, although it isn’t really necessary. Maybe the site list could be controlled by the admin staff, so the officers don’t need to worry about it, they just select it when they go to site

Q. And lastly, how long does the normal site visit take? Do you need users to be able to check in for different times, or is there a minimum time that they need to contact the office?

A. Well, the policy at the moment is that they contact the office hourly when they are out on site. Maybe when they check in it could be set as 1 hour, then if they want to reset this they can just check in again to gain another hour.

From the discussion, I have noted the following points and will take them into consideration both when designed the app, but also in the third iteration when I finish the GUI:

* When a user checks in on site, they get one hour to either check out or check in again, restarting that hour.
* All sites can either be entered as new by the officers on site, or if returning to an existing site they can select that from a list.
* Sites will be deleted by office staff when officers hand them the file to say that inspections have finished at that site.
* The GUI will be redesigned as a dashboard, so I will be removing the check-out time fields and expected time fields and replacing them with a single “last checked in” box.
* The GUI will only show visits that are taking place at that time, and an alert will show up on screen if someone hasn’t checked in within the hour.
* Although site information can be deleted, a log of all check-ins and check-outs will be retained

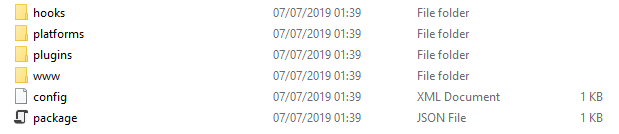
#### 1.3.3.2 Setting up my Cordova project

To develop my mobile phone app, I have used Apace Cordova. As I have previously used this in other subjects, I had it installed on my machine already so I wont go into the installation details again here. To create the project, I simply open a command prompt and type:

cordova create tm470app

which creates the project is the current working directory. I can then navigate to this file with cd tm470app

which puts me inside the project file. The default file structure should look like this when viewed with windows file explorer:

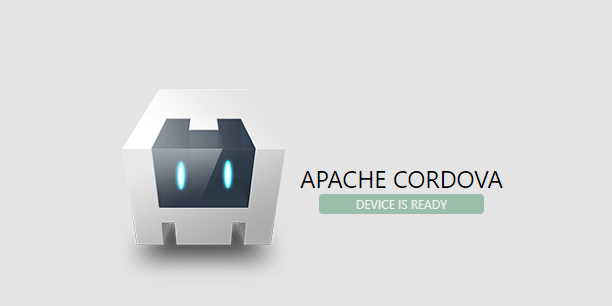


The www folder is containing the HTML, CSS and JavaScript files that I’ll use to develop the app. To run the file, I need to add ‘platforms’ that I wish it to run on, so I need to go back to the command line, navigate to the project directory and type:

cordova platform add android adds android as a platform – to run the app

cordova platform add browser adds the browser as a platform – used for testing

Running the app as it is shows



to confirm the app is working. Now I can start editing the www files to customise my app.

#### 1.3.3.3 Writing the code for the app

I’ll begin with the HTML file, to get the content on the screen that I want to show. I have deleted all the autogenerated code to start with a blank canvas; from there I know the screen has roughly got to take on the following form:

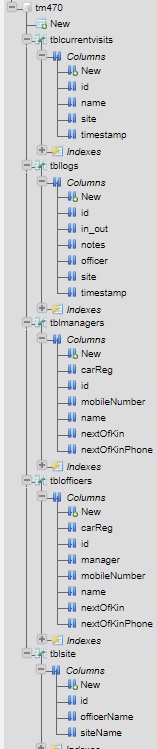
|  |
| --- |
| Logo |
| Select Username |
| Select Site |
| Enter Notes |
| Check In |
| Check Out |
| Status Update |

At this point it is worth noting that because I don’t have a mobile phone linked to a work account, I am unable to find the username by the account that’s logged into the device. To get around this I have decided to add a username dropdown box; the intention is that this will be removed in future iterations.

Once the HTML had been completed it was my intention to use the Javascript to connect to my SQLite database that I have on my pc, but until this point I hadn’t thought about how to do this. After doing a google search for “connect to SQLite Database with Cordova” I found two plugins that would appear to provide the functionality I needed, “cordova-plugin-sqlite-2” and “cordova-sqlite-storage”. After following multiple tutorials on how to get these to work, as well as one or two less documented plugins, I realised that this wouldn’t work how I had previously imagined; all of the plugins required you to copy the database to the device and work with that, rather than use the data stored in a central location which is what I needed.

After a bit more research I found out that Javascript, being a frontend language, can’t communicate with a backend service like a SQLite Database. From this point forward, I installed a program called XAMPP, which allowed the local host on my machine to act as a server. It is bundled with phpMyAdmin and MySql which I can use to connect the frontend to the backend. As long as the mobile phone and my machine were on the same local network, they could communicate.

#### 1.3.3.4 Using XAMPP – phpMyAdmin and MySQL

The first task I had to do was to import my database into MySQL running on the local host. Luckily, the package I was originally using (DB Browser) had an export facility so it was as simple as exporting the data from one and importing it into MySQL. Once I had the original data, I also create tables for the current site visits, the active site lists, and another to hold all of the logs. My database structure was now as follows:

tblcurrentvisits will hold a list of all the officers that are on site right now. When they check in with the app their details will be added to this table, and when they check out they will be removed.

tbllogs will store all activity (every check in and check out) that takes place, and be used as a reference of a visit taking place if needed.

tblmanagers (as previously described) will hold all the information for all of the team leaders.

tblofficers (as previouslt described) will hold all the information for all the site officers.

tblsite will hold a lost of all of the active sites. They can be added to from the mobile app or deleted from the GUI

#### 1.3.3.5 Writing the PHP to Link the App with the MySQL Database

This part of the process took by far the longest; even though the PHP I’ve written is quite simplistic, I’ve never really used it before so I had to pick everything up rom new. Luckily there is an abundance of resources available online that I was able to use a combination of to create the functionality I needed. Once XAMPP is running, it emulates a server on my local host. Within the installation folder there is a folder called “htdocs”; this acts as a sort of a file server, so any PHP files I place in there can be accessed from my app. The files I’ve created are:

|  |  |
| --- | --- |
| Filename | Function |
| db.php | This file create a connection to the database stored under the variable $con. This file will be referenced at the start of all the others so we can use the same connection |
| getNames.php | Used to retrieve the list of usernames from the officers table, activated when the user clicks the drop down list in the app. Executes the SQL “select name from `tblofficers` ORDER BY name ASC” |
| getSites.php | Used to retrieve the list of available sites from the site table, activated when the user clicks the drop down list in the app. Executes the SQL “select siteName from `tblsite` ORDER BY siteName ASC” |
| checkIn.php | This file has 3 functions:   1. If the site is new i.e. it has been typed by the user, add it to the tblsites 2. Add all of the data to tbllogs to record the action 3. Add the info into tblcurrentvisits to note the officer is currently on site |
| checkout.php | This file has two fucntions:   1. Add all of the data to tbllogs to record the action 2. Remove the corresponding entry in tblcurrentvisits to show the officer is not currently on site |

For the listing of all code, please see appendix 4.11

After all of the PHP code had been written, I was able to use jQuery, which is a javascript framework, to run the PHP files running on the local host. Before putting this app into production, the HTML in index.php would need to be edited. Currently it’s pointing to the IP address of my desktop machine, a new server would have to be created running phomyadmin and MySQL, and that IP address of that server would need to replace my IP. This would allow anyone with an internet connection running the app to connect to the server.

#### 1.3.3.6 Next Steps

# 2 Review



## Review of the current stage of project work

## Review Project Management

## Assess risks to project completion

## Review of personal development

# 3 References

# 4 Appendices

[4.1 The initial enquiry email from the client](#_4.1_The_initial)

[4.2 Initial Meeting with the Client](#_4.3_–_Initial)

[4.3 Proposed designs for the Java GUI](#_4.3_Proposed_designs)

[4.4 Agreed designs for the Java GUI](#_4.4_Agreed_designs)

[4.5 JSwing pages](#_Appendix_4.5_–)

[4.6 Java Code](#_4.6_–_Java)

[4.7 Mobile App Protype](#_4.7_Mobile_App)

[4.8 Code for the Mobile App](#_4.8_Code_for)

[4.9 Proposed Design of the Mobile App v2](#_4.9_Proposed_Design)

[4.10 Screenshots of the Mobile App](#_4.10_Screenshots_of)

## 4.1 The initial enquiry email from the client

|  |  |
| --- | --- |
| Northumberland County Council Mail | **Nicky Temperley <nicky.temperley@northumberland.gov.uk>** |

|  |
| --- |
| **FW: Lone Working Policy** 1 message |

|  |  |
| --- | --- |
| **Michelle Williams**<michelle.williams@northumberland.gov.uk> | 13 February 2019 at 12:11 |
| To: Nicky Temperley <nicky.temperley@gs.northumberland.gov.uk> | |
| |  | | --- | | Just checking with Joe Murphy/ Joanne Southern that anything has been done yet?  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ From: Fiona Charleton <[fiona.charleton@northumberland.gov.uk](mailto:fiona.charleton@northumberland.gov.uk)> Sent: 15 November 2018 16:32 To: Michelle Williams; Harri Bellizzi Subject: Fwd: Lone Working Policy  Hi Michelle/Harri  Sorry I know this is quite a bad time for you! I have been speaking with John Froud today about looking at the options of building a lone worker system for our Planning Officers who frequently go out on site. The system we use currently does not work so we need to review how we monitor our lone workers movements so we know where they are, who they are meeting and highlight when they haven't returned as expected.  I have spoken with Julie Seaton at Newcastle City Council. She is a former member of NCC staff and was very complimentary of the Newcastle system when she moved over 3 years ago. She has sent me a screenshot and brief overview as to how their system works. I have forwarded this on to IT for a call to be raised as to whether we can build something similar for our Officers. John has suggested I send this to both of you to look at the possibility of developing this. He is also keen to maybe widen this to other Services if we can build something that would work. Newcastle have been using this for a few years now and it does seem to work for them so we would be foolish to not investigate it when we are struggling with our current process.  Thanks Fiona  ---------- Forwarded message --------- From: Fiona Charleton <[fiona.charleton@northumberland.gov.uk](mailto:fiona.charleton@northumberland.gov.uk)<mailto:[fiona.charleton@northumberland.gov.uk](mailto:fiona.charleton@northumberland.gov.uk)>> Date: Thu, 15 Nov 2018 at 16:21 Subject: Fwd: Lone Working Policy To: John Froud <[john.froud@northumberland.gov.uk](mailto:john.froud@northumberland.gov.uk)<mailto:[john.froud@northumberland.gov.uk](mailto:john.froud@northumberland.gov.uk)>>   Hi John  Further to our discussion. Please see the below email which I mentioned I received from Newcastle City Council which details the home built system that they have been using with success for their Planning Officers.  I will update you when I know more from the call I have raised with IT. I will also send it separately to Michelle and Harri for them to look at.  Thanks for your help.  Fiona  ---------- Forwarded message --------- From: Seaton, Julie <[julie.seaton@newcastle.gov.uk](mailto:julie.seaton@newcastle.gov.uk)<mailto:[julie.seaton@newcastle.gov.uk](mailto:julie.seaton@newcastle.gov.uk)>> Date: Tue, 13 Nov 2018 at 12:26 Subject: RE: Lone Working Policy To: Fiona Charleton <[fiona.charleton@northumberland.gov.uk](mailto:fiona.charleton@northumberland.gov.uk)<mailto:[fiona.charleton@northumberland.gov.uk](mailto:fiona.charleton@northumberland.gov.uk)>>   Hi Fiona  Things are going OK here  We have an IT sytem that all officers must use when going out on site. They need to say where they are going and what time they will be back. It flashes an alert if they are not back by the time they stated and a manager must be notified. We monitor usage and it is a disciplinary offence if it’s not used. There are the odd occasion where people may go home straight from site and then we get them to call a ‘buddy’ back in the office when they are leaving the site – but it must be at a reasonable time when we know there will still be someone in the office who could raise the alarm if they don’t call.  All our pool cars (which staff use) have trackers on them so we could trace a car if we needed to.  This is a screen shot of the site visit database. Simon (our IT guru) put it together so he could possibly do something similar again.  Julie    -- Fiona Charleton Senior Technical Planning Officer Planning Northumberland County Council (01670) 622687   -- Fiona Charleton Senior Technical Planning Officer Planning Northumberland County Council (01670) 622687 | | |

## 4.2 Initial Meeting with the Client

Q. What are you expecting from this system as a minimum requirement

A. I would like a system to record who’s out on site, when they got there, when their expected to leave, and receive an alert if they haven’t left by this time. Id like to be able the add or delete staff as they change too, although that will be mainly be down to the admin staff.

Q. What information do you store about the officers

A. We store their first name, surname, a mobile number, and their date of birth in case we have two people with the same name.

Q. Do you need the users to log in for security, or are their Windows Credentials enough to verify their identity?

A. No, theres no need for them to log in really.

Q. Do you need managers to have a different level of access to the admin staff? If so, what is the difference?

A.

Q. Rather than tracking pool cars, do all the staff have company issued mobile phones?

A. Yes, all staff have a company issued mobile phone and they all have a voice and data package.

Q. How would you like to see the alert to managers? By email, screen notification, text message etc.

A.

Q. Do you need to record historic information, or once the site officer has confirmed they’ve left the site can the information be discarded?

A. Yes, we would need to save the information for a minimum of one year.

Q. On the screenshot you’ve provided, would the user complete the site they are visiting, or is that selected from a list of sites?

A.

Q. Do you need to import any previous information into the system, or will you be using this as a fresh start?

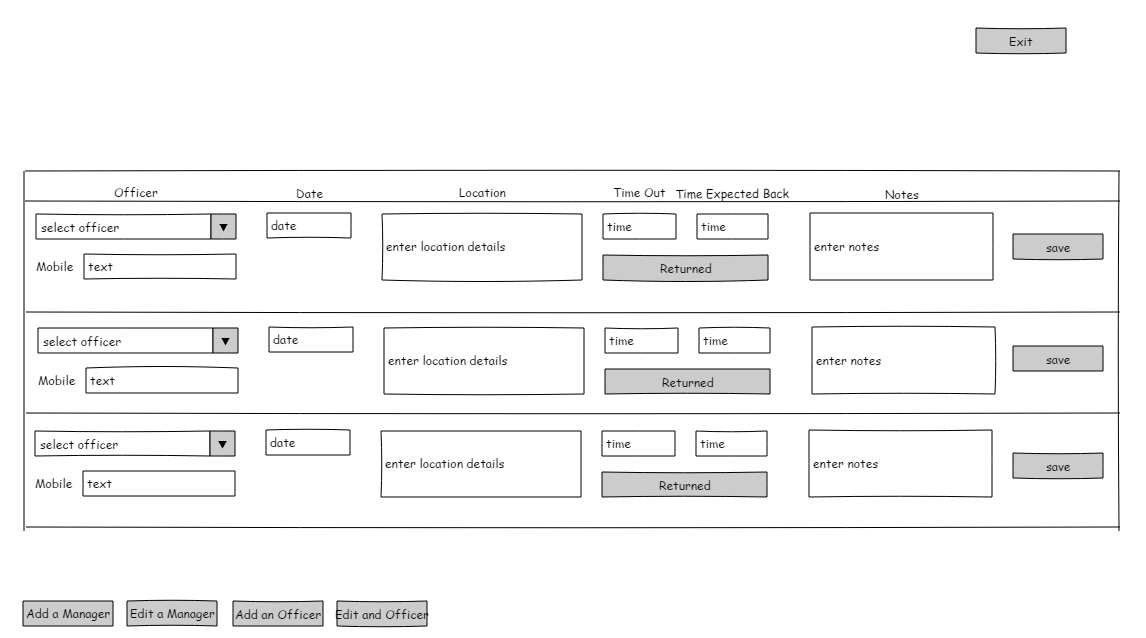
A. No, we’re going to use this system as new, no previous data will be imported into it, it’ll all be stored in its original state. If we need to refer back to it we’ll do it manually, there’s no need to link this new system to it.

Q. Do you a unique identifier for Officers and locations, or would a generic ID be enough?

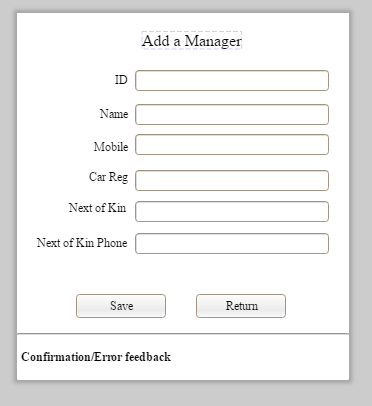
A. We use the staff payroll number as a unique ID for staff members, but the site ID isn’t relevant to anything we hold in our systems.

## 4.3 Proposed designs for the Java GUI

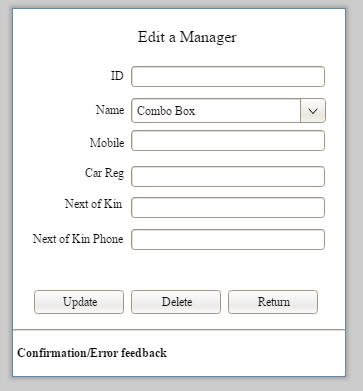
### 4.3.1 Main User Interface



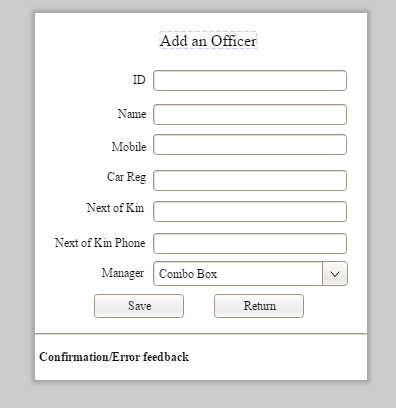
### 4.3.2 Add a manager



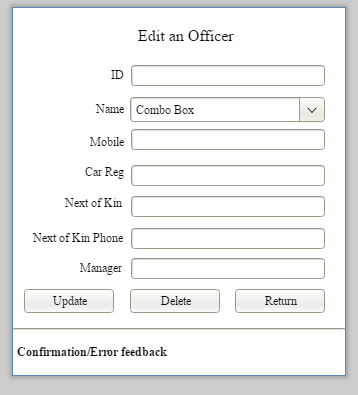
### 4.3.3 Edit a Manager



### 4.3.4 Add An Officer

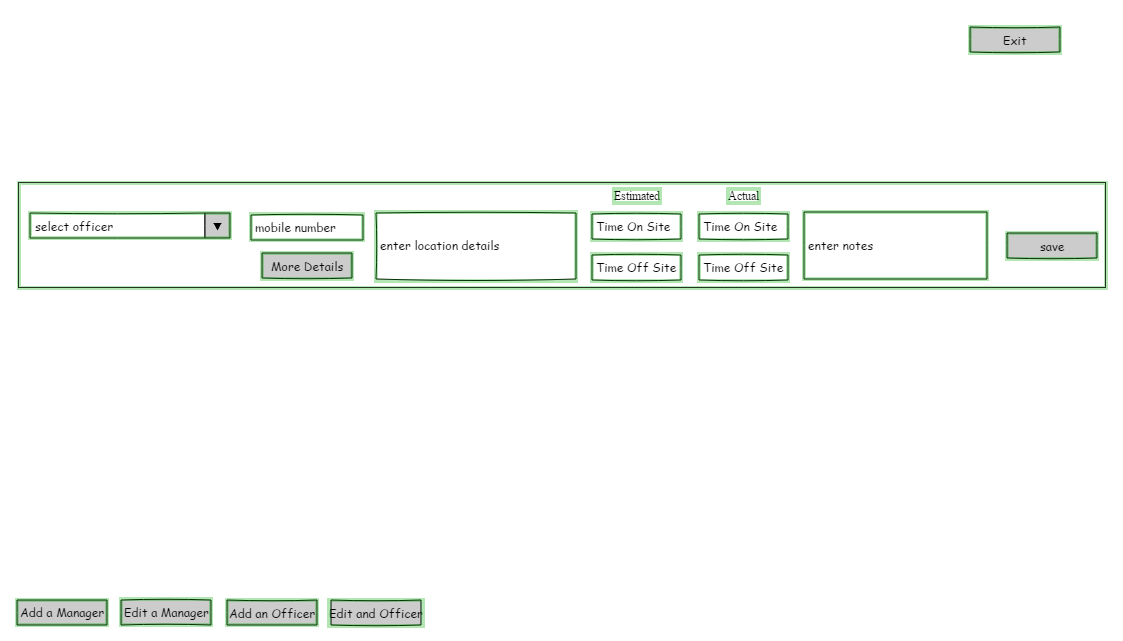


### 4.3.5Edit An Officer

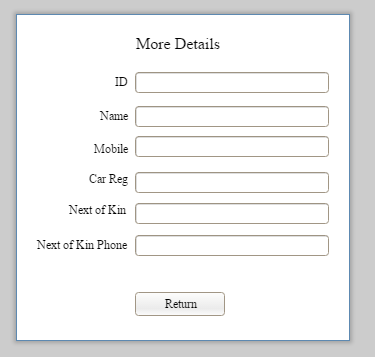


## 4.4 Agreed designs for the Java GUI

### 4.4.1 Main User Interface



### 4.4.2 More Details

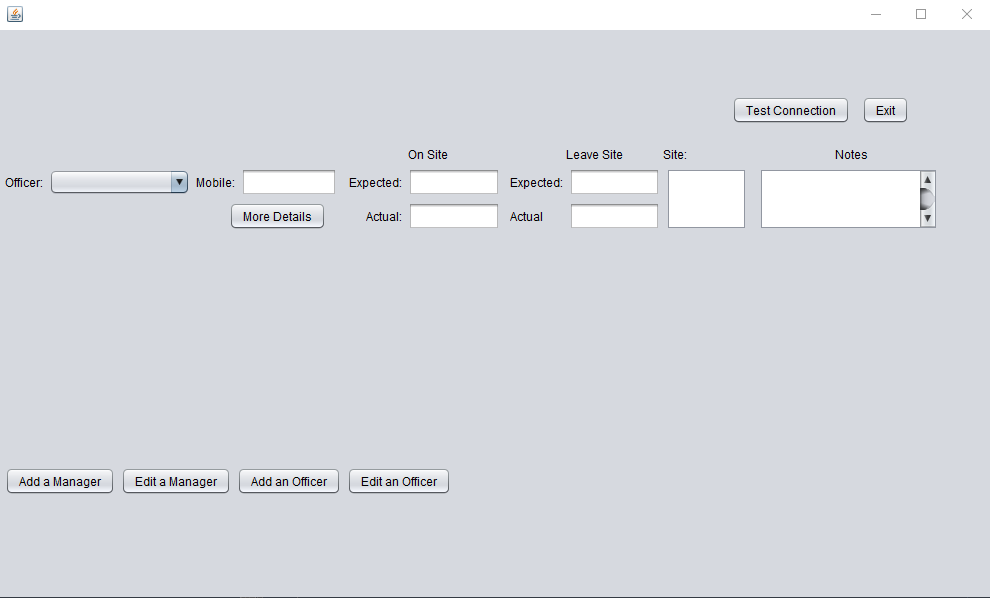


**All other pages for the GIU were agreed as proposed**

## 4.5 JSwing pages

### 4.5.1 MainGUI,java

Exit the system



EditAManager.java

AddAManager.java

Only here for testing purposes

EditAnOfficer.java

AnnAnOfficer.java

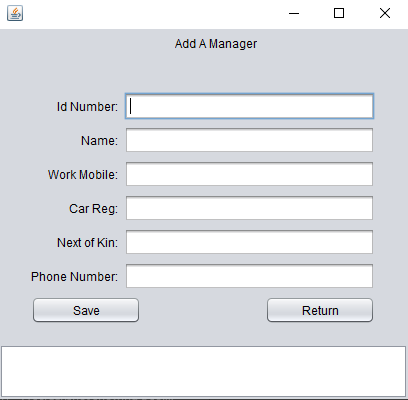
MoreDetails.java

For the purpose of this iteration, the users can:

* 1. Select an officer from the drop down box. This will populate the box with the officers name and the mobile field with their mobile number.
  2. Once an officer has been selected, the user can click on ‘More Details’ and see all the details held for that officer.
  3. Click ‘Add a Manager’ to go to the AddAManager.java page.
  4. Click ‘Edit a Manager’ to go to the EditAManager.java page.
  5. Click ‘Add an Officer’ to go to the AddAnOfficer.java page.
  6. Click ‘Edit an Officer’ to go to the EdditAnOfficer.java page.
  7. Click ‘Exit’ to close the system.

Please note, the ‘Test Connection’ button confirms connection to the database, and is only there for testing purposes. This will be remove in a later iteration.

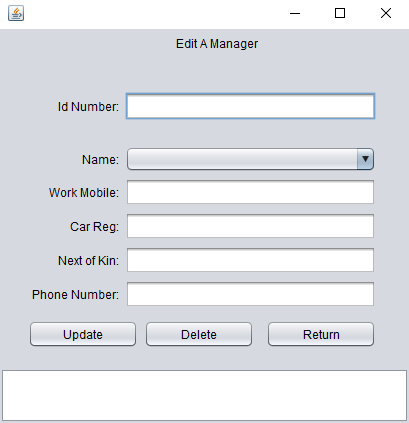
### 4.5.2 AddAManager.java



MainGUI.java

The user can either enter all the required details here, or click return to go back to the main page. Once all the details have been entered and the user clicks save, either a confirmation message or an error message is displayed in the white bar at the bottom.

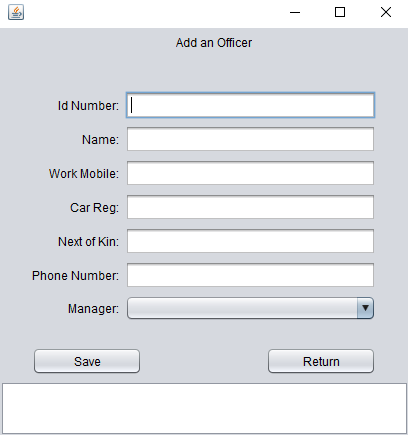
### 4.5.3 EditAManager.java



MainGUI.java

On this page the user has to select the Manager they wish to amend from the drop-down list. Once selected they can either change any of the details and click update to save; click delete to remove the user from the database or return to the main screen. Once all the details have been entered and the user clicks save, either a confirmation message or an error message is displayed in the white bar at the bottom.

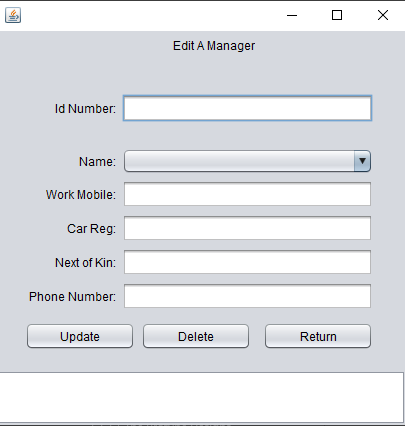
### 4.5.4 AddAnOfficer.java



MainGUI.java

This form has the same functionality as AddAManager.java with the addition of the Manager drop down box; this needs to be selected when adding a new Officer. The manager drop-down provides a collection of all managers who are currently on the system. Once all the details have been entered and the user clicks save, either a confirmation message or an error message is displayed in the white bar at the bottom.

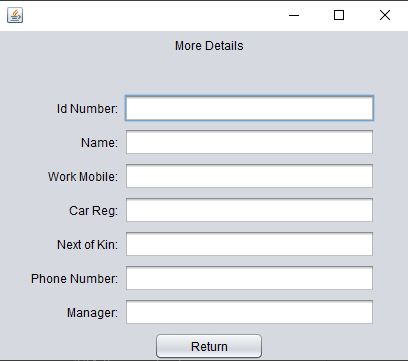
### 4.5.5 EditAnOfficer.java



MainGUI.java

Again, this form has the same functionality as EditAManager.java with the addition of the Manager drop down box.

### 4.5.6 More Details

  
  
This page has no functionality, the only option the user has is to click ‘Return’ and go back to the main page. The details cant be edited from this view either.

MainGUI.java

## 4.6 Java Code

### 4.6.1 MainGUI.java

**I have removed all system generated code. For the full code please see the attached program in the zip directory**

|  |
| --- |
| package GUI;  import java.sql.Connection; import java.sql.DriverManager; import java.sql.PreparedStatement; import java.sql.ResultSet; import java.sql.SQLException; import java.sql.Statement; import java.util.List; import java.util.logging.Level; import java.util.logging.Logger; import javax.swing.JOptionPane; import tm470.DataConnect;  /\*\*  \*  \* @author zy395907  \*/ public class **MainGUI** extends **javax**.**swing**.**JFrame** {   private void **btnTestConnectionMouseClicked**(java.awt.event.MouseEvent evt) {   DataConnect a = new DataConnect();   a.testConnection();  }   private void **btnExitMouseClicked**(java.awt.event.MouseEvent evt) {   int dialogButton = JOptionPane.YES\_NO\_OPTION;   int dialogResult = JOptionPane.showConfirmDialog(this, "Are you sure?", "Close the system", dialogButton);   if (dialogResult == 0) {    System.exit(0);   } else {    System.out.println("No Option");   }  }   private void **btnAddManagerMouseClicked**(java.awt.event.MouseEvent evt) {   AddAManager aManager = new AddAManager();   aManager.setVisible(true);  }   private void **btnAddOfficerMouseClicked**(java.awt.event.MouseEvent evt) {   AddAnOfficer anOfficer = null;   try {    anOfficer = new AddAnOfficer();   } catch (SQLException ex) {    Logger.getLogger(MainGUI.class.getName()).log(Level.SEVERE, null, ex);   }   anOfficer.setVisible(true);  }   private void **btnEditManagerMouseClicked**(java.awt.event.MouseEvent evt) {   EditAManager aManager = null;   try {    aManager = new EditAManager();   } catch (SQLException ex) {    Logger.getLogger(MainGUI.class.getName()).log(Level.SEVERE, null, ex);   }   aManager.setVisible(true);  }   private void **cmboOfficerMouseClicked**(java.awt.event.MouseEvent evt) {   cmboOfficer.removeAllItems();   DataConnect dbc = new DataConnect();   List < String > officers = null;   try {    officers = dbc.getOfficers();   } catch (SQLException ex) {    Logger.getLogger(MainGUI.class.getName()).log(Level.SEVERE, null, ex);   }   officers.forEach((officer) -> {    cmboOfficer.addItem(officer);   });  }   private void **btnMoreDetailsMouseClicked**(java.awt.event.MouseEvent evt) {   MoreDetails more = null;   try {    more = new MoreDetails();   } catch (SQLException ex) {    Logger.getLogger(MainGUI.class.getName()).log(Level.SEVERE, null, ex);   }   String officerName = (String) cmboOfficer.getSelectedItem();   more.setVisible(true);   try {    more.getDetails(officerName);   } catch (SQLException ex) {    Logger.getLogger(MainGUI.class.getName()).log(Level.SEVERE, null, ex);   }  }   private void **cmboOfficerPropertyChange**(java.beans.PropertyChangeEvent evt) {   System.out.println("Changed");  }   private void **cmboOfficerActionPerformed**(java.awt.event.ActionEvent evt) {   String vdbDriver = "jdbc:sqlite:";   String vDatabase = "D:/Program Files/SQLiteDatabaseBrowserPortable/Data/TM470.db";   Connection vConnection = null;   String name = (String) this.cmboOfficer.getSelectedItem();   if (name != null) {    try {     vConnection = DriverManager.getConnection(vdbDriver + vDatabase);     Statement stmt = vConnection.createStatement();     String query = "SELECT mobileNumber FROM Officers WHERE name = \"" + name + "\"";     ResultSet results = stmt.executeQuery(query);     txtMobile.setText(results.getString(1));    } catch (SQLException ex) {     Logger.getLogger(MainGUI.class.getName()).log(Level.SEVERE, null, ex);    }   }  } |

### 4.6.2 AddAManager.java

**I have removed all system generated code. For the full code please see the attached program in the zip directory**

|  |
| --- |
| package GUI;  import java.sql.\*; import java.util.logging.Level; import java.util.logging.Logger;  /\*\*  \*  \* @author zy395907  \*/ public class **AddAManager** extends **javax**.**swing**.**JFrame** {       private void **btnReturnMouseClicked**(java.awt.event.MouseEvent evt) {                                               this.setVisible(false);     }                                        private void **btnSaveMouseClicked**(java.awt.event.MouseEvent evt) {                                               try {             String id;             String name;             String mobileNumber;             String carReg;             String nextOfKin;             String nextOfKinPhone;             String vdbDriver = "jdbc:sqlite:";             String vDatabase = "D:/Program Files/SQLiteDatabaseBrowserPortable/Data/TM470.db";             Connection vConnection;              id = txtId.getText();             name = txtName.getText();             mobileNumber = txtMobileNumber.getText();             carReg = txtCarReg.getText();             nextOfKin = txtNextOfKin.getText();             nextOfKinPhone = txtNextOfKinPhone.getText();             vConnection = DriverManager.getConnection(vdbDriver + vDatabase);             PreparedStatement st = vConnection.prepareStatement("insert into MANAGERS(id, name, mobileNumber, carReg, nextOfKin, nextOfKinPhone) values (?,?,?,?,?,?)");             st.setString(1, id);             st.setString(2, name);             st.setString(3, mobileNumber);             st.setString(4, carReg);             st.setString(5, nextOfKin);             st.setString(6, nextOfKinPhone);             int a = st.executeUpdate();             if (a > 0) {                 txtAddManagerOutcome.setText((String)name + " has been added, you may \n"                     + "now add another manager or close this screen by clicking Return");                 txtId.setText("");                 txtName.setText("");                 txtMobileNumber.setText("");                 txtCarReg.setText("");                 txtNextOfKin.setText("");                 txtNextOfKinPhone.setText("");             }         } catch (SQLException ex) {             Logger.getLogger(AddAManager.class.getName()).log(Level.SEVERE, null, ex);             txtAddManagerOutcome.setText("There has been an error \n"                     + "please check the data above and try again");         }      } |

### 4.6.3 EditAManager.java

|  |
| --- |
| **I have removed all system generated code. For the full code please see the attached program in the zip directory**  package GUI;  import java.sql.Connection; import java.sql.DriverManager; import java.sql.PreparedStatement; import java.sql.SQLException; import java.util.List; import java.util.logging.Level; import java.util.logging.Logger; import tm470.DataConnect;  /\*\*  \*  \* @author zy395907  \*/ public class **EditAnOfficer** extends **javax**.**swing**.**JFrame** {      /\*\*      \* Creates new form AddAnOfficer      \* @throws java.sql.SQLException      \*/     public **EditAnOfficer**() throws SQLException {         initComponents();         cmboName.removeAllItems();         DataConnect dbc = new DataConnect();         List<String> managers = dbc.getManagers();         managers.forEach((manager) -> {             cmboName.addItem(manager);         });        }       private void **btnReturnMouseClicked**(java.awt.event.MouseEvent evt) {                                                this.setVisible(false);     } |

### 4.6.4 AddAnOfficer.java

|  |
| --- |
| **I have removed all system generated code. For the full code please see the attached program in the zip directory**  package GUI;  import java.sql.Connection; import java.sql.DriverManager; import java.sql.PreparedStatement; import java.sql.SQLException; import java.util.List; import java.util.logging.Level; import java.util.logging.Logger; import tm470.DataConnect;  /\*\*  \*  \* @author zy395907  \*/ public class **AddAnOfficer** extends **javax**.**swing**.**JFrame** {      /\*\*      \* Creates new form AddAnOfficer      \* @throws java.sql.SQLException      \*/     public **AddAnOfficer**() throws SQLException {         initComponents();         cmboManager.removeAllItems();         DataConnect dbc = new DataConnect();         List<String> managers = dbc.getManagers();         managers.forEach((manager) -> {             cmboManager.addItem(manager);         });        }       private void **btnReturnMouseClicked**(java.awt.event.MouseEvent evt) {                                                this.setVisible(false);     }                                         private void **btnSaveMouseClicked**(java.awt.event.MouseEvent evt) {                                              try {             String id;             String name;             String mobileNumber;             String carReg;             String nextOfKin;             String nextOfKinPhone;             String manager;             String vdbDriver = "jdbc:sqlite:";             String vDatabase = "D:/Program Files/SQLiteDatabaseBrowserPortable/Data/TM470.db";             Connection vConnection = null;                                       id = txtId.getText();             name = txtName.getText();             mobileNumber = txtMobileNumber.getText();             carReg = txtCarReg.getText();             nextOfKin = txtNextOfKin.getText();             nextOfKinPhone = txtNextOfKinPhone.getText();             manager = (String)cmboManager.getSelectedItem();             Connection con = DriverManager.getConnection(vdbDriver + vDatabase);             PreparedStatement st = con.prepareStatement("insert into officers(id, name, mobileNumber, carReg, nextOfKin, nextOfKinPhone, manager) values (?,?,?,?,?,?,?)");             st.setString(1, id);             st.setString(2, name);             st.setString(3, mobileNumber);             st.setString(4, carReg);             st.setString(5, nextOfKin);             st.setString(6, nextOfKinPhone);             st.setString(7, manager);             int a = st.executeUpdate();             if (a>0)             {                 txtAddManagerOutcome.setText((String)name + " has been added, you may \n"                     + "now close this screen by clicking Return");                 txtId.setText("");                 txtName.setText("");                 txtMobileNumber.setText("");                 txtCarReg.setText("");                 txtNextOfKin.setText("");                 txtNextOfKinPhone.setText("");;                 cmboManager.setSelectedIndex(-1);                              }         } catch (SQLException ex) {             txtAddManagerOutcome.setText("There has been an error \n"                     + "please check the data above and try again");             Logger.getLogger(AddAnOfficer.class.getName()).log(Level.SEVERE, null, ex);         }          } |

### 4.6.5 EditAnOfficer.java

|  |
| --- |
| **I have removed all system generated code. For the full code please see the attached program in the zip directory**  package GUI;  import java.sql.Connection; import java.sql.DriverManager; import java.sql.PreparedStatement; import java.sql.SQLException; import java.util.List; import java.util.logging.Level; import java.util.logging.Logger; import tm470.DataConnect;  /\*\*  \*  \* @author zy395907  \*/ public class **EditAnOfficer** extends **javax**.**swing**.**JFrame** {      /\*\*      \* Creates new form AddAnOfficer      \* @throws java.sql.SQLException      \*/     public **EditAnOfficer**() throws SQLException {         initComponents();         cmboName.removeAllItems();         DataConnect dbc = new DataConnect();         List<String> managers = dbc.getManagers();         managers.forEach((manager) -> {             cmboName.addItem(manager);         });        }       private void **btnReturnMouseClicked**(java.awt.event.MouseEvent evt) {                                                this.setVisible(false);     } |

### 4.6.6 MoreDetails.java

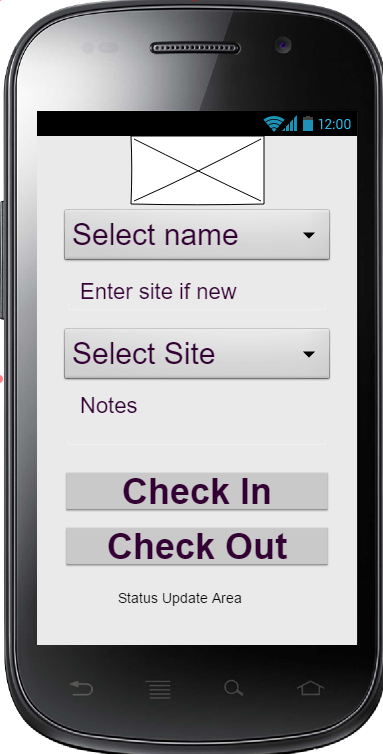
**I have removed all system generated code. For the full code please see the attached program in the zip directory**

|  |
| --- |
| package GUI;  import java.sql.Connection; import java.sql.DriverManager; import java.sql.PreparedStatement; import java.sql.ResultSet; import java.sql.SQLException; import java.sql.Statement; import java.util.List; import java.util.logging.Level; import java.util.logging.Logger; import tm470.DataConnect;  /\*\*  \*  \* @author zy395907  \*/ public class **MoreDetails** extends **javax**.**swing**.**JFrame** {      /\*\*      \* Creates new form AddAnOfficer      \* @param name      \* @throws java.sql.SQLException      \*/     public **MoreDetails**() throws SQLException {         initComponents();              }       private void **btnReturnMouseClicked**(java.awt.event.MouseEvent evt) {                                                this.setVisible(false);     }                                         private void **txtNameMouseExited**(java.awt.event.MouseEvent evt) {                                                  }                                          /\*\*      \* Get the details for the officer sselected in MainGUI      \* @param officerName      \* @throws SQLException       \*/     public void **getDetails**(String officerName) throws SQLException {         String vdbDriver = "jdbc:sqlite:";         String vDatabase = "D:/Program Files/SQLiteDatabaseBrowserPortable/Data/TM470.db";         Connection vConnection = null;                  txtName.setText(officerName);         vConnection = DriverManager.getConnection(vdbDriver + vDatabase);         Statement stmt = vConnection.createStatement();             String query = "SELECT \* FROM Officers WHERE name = \"" + officerName + "\"";             ResultSet results = stmt.executeQuery(query);              txtId.setText(results.getString(1));             txtMobileNumber.setText(results.getString(3));             txtCarReg.setText(results.getString(4));             txtNextOfKin.setText(results.getString(5));             txtNextOfKinPhone.setText(results.getString(6));             txtManager.setText(results.getString(7));                             } |

### 4.6.7 DataConnect.java

|  |
| --- |
| package tm470;  import java.sql.\*; import java.util.\*;  public class **DataConnect** {     String vdbDriver = "jdbc:sqlite:";  String vDatabase = "D:/Program Files/SQLiteDatabaseBrowserPortable/Data/TM470.db";  Connection vConnection = null;                           /\*\*          \* method to test database connection which returns a boolean value          \*/  public boolean **testConnection**(){    //try connecting to database, if successful show success message and return true boolean value  try {    vConnection = (Connection)DriverManager.getConnection(vdbDriver + vDatabase);    System.out.println("Connection to database successful!");    return true;      //catch will show unsuccessful message and return false boolean value     } catch (Exception ex) {    System.err.println ("Connection unsuccessful\n" + ex.toString ());    return false;  }     }                                                         /\*\*      \* return the list of managers to populate the manager list in cmboManager      \* on AddAnOfficer.java      \*      \* @return List of managers      \* @throws java.sql.SQLException      \*/     public List<String> **getManagers**() throws SQLException {         List<String> list = new ArrayList();         vConnection = DriverManager.getConnection(vdbDriver + vDatabase);         try {             Statement stmt = vConnection.createStatement();             String query = "SELECT name FROM managers ORDER BY name";             ResultSet results = stmt.executeQuery(query);             list.add(null);             while (results.next()) {                 list.add(results.getString("name"));             }                     } catch (SQLException e) {             System.out.println("Exception = " + e);         }         return list;     }          public List<String> **getOfficers**() throws SQLException {         List<String> list = new ArrayList();         vConnection = DriverManager.getConnection(vdbDriver + vDatabase);         try {             Statement stmt = vConnection.createStatement();             String query = "SELECT name FROM officers ORDER BY name";             ResultSet results = stmt.executeQuery(query);             list.add(null);             while (results.next()) {                 list.add(results.getString("name"));             }             stmt.close();         } catch (SQLException e) {             System.out.println("Exception = " + e);         }         return list;     }              public ResultSet **getManagersDetails**(String name) throws SQLException {         List<String> list = new ArrayList();         vConnection = DriverManager.getConnection(vdbDriver + vDatabase);         try {             Statement stmt = vConnection.createStatement();             String query = "SELECT \* FROM managers WHERE name = \"" + name + "\" ORDER BY name";             ResultSet results = stmt.executeQuery(query);                          return results;                      } catch (SQLException e) {             System.out.println("Exception = " + e);         }         //System.out.println(list);         return null;     }               public void **deleteManager**(String managerId) throws SQLException{             vConnection = DriverManager.getConnection(vdbDriver + vDatabase);             PreparedStatement st = vConnection.prepareStatement("DELETE FROM Managers WHERE id = \" + managerId + \"");             st.executeUpdate();                   }    } |

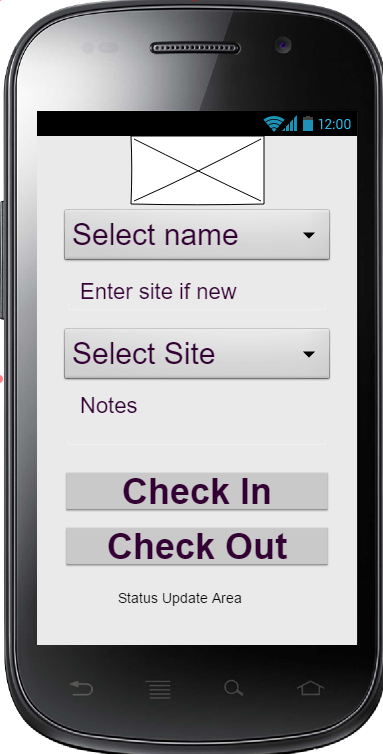
## 4.7 Mobile App Protype



## 4.8 Code for the Mobile App

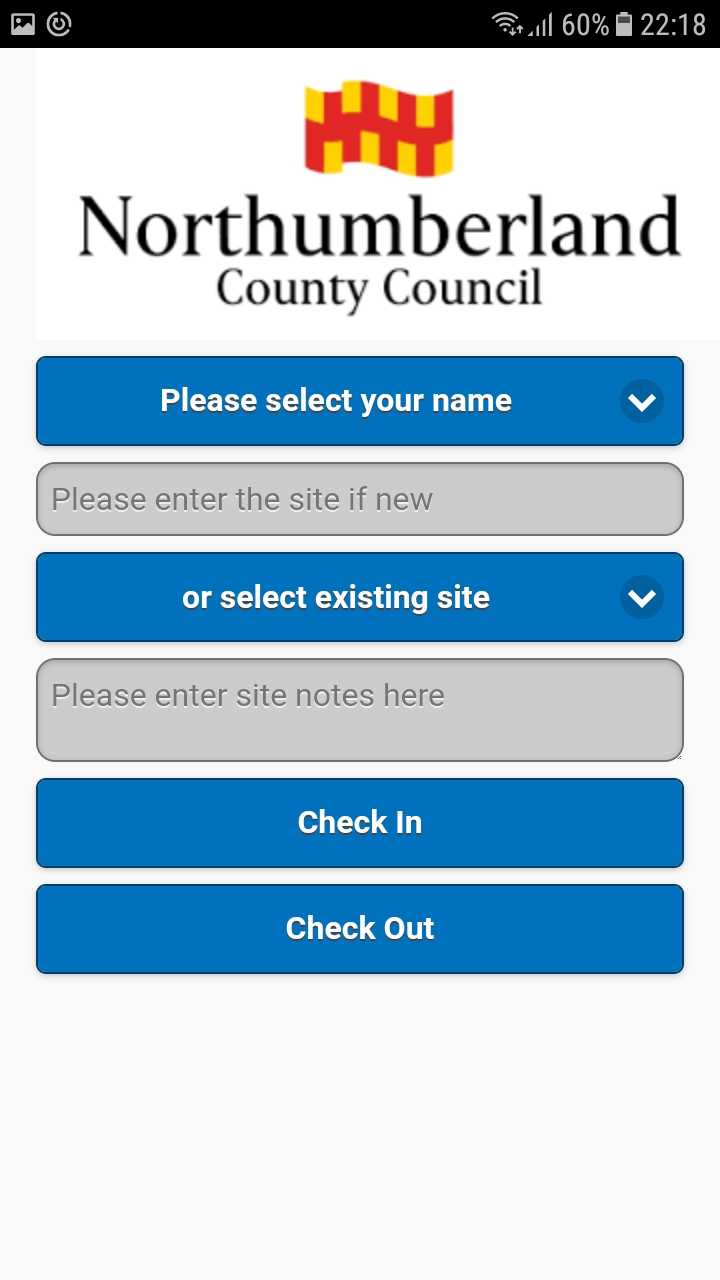
1. <html>
2. <head>
3. <meta charset="utf-8" />
4. <meta name="viewport" content="width=device-width, initial-scale=1">
5. <link rel="stylesheet" href="css/index.css" />
6. <link rel="stylesheet" href="css/themes/app.css" />
7. <link rel="stylesheet" href="css/themes/jquery.mobile.icons.min.css" />
8. <link rel="stylesheet" href="http://code.jquery.com/mobile/1.4.5/jquery.mobile.structure-1.4.5.min.css" />
9. <script src="http://code.jquery.com/jquery-1.11.1.min.js"></script>
10. <script src="http://code.jquery.com/mobile/1.4.5/jquery.mobile-1.4.5.min.js"></script>
11. <script type="text/javascript"> //function to insert officer names from db into dropdown list
12. $(document).ready(function () {
13. var url = "http://192.168.1.247/getNames.php"; //ip of local host over ethernet
14. $.getJSON(url, function (result) {
15. $.each(result, function (i, field) {
16. var name = field.name;
17. $("#name").append("<option value='" + name + "'>" + name + "</option>");
18. });
19. });
20. });
21. </script>
22. <script type="text/javascript"> //function to insert existing sites from db into dropdown list
23. $(document).ready(function () {
24. var url = "http://192.168.1.247/getSites.php"; //ip of local host over ethernet
25. $.getJSON(url, function (result) {
26. $.each(result, function (i, field) {
27. var siteName = field.siteName;
28. $("#existingSite").append("<option value='" + siteName + "'>" + siteName + "</option>");
29. });
30. });
31. });
32. </script>
33. <script type="text/javascript"> //function to check in
34. $(document).ready(function ()
35. {
36. $("#checkIn").click(function () {
37. var action = "";
38. var notes = "";
39. if ($("#site").val() === "" && $("#existingSite option:selected").val()!=="Please enter the site if new") {
40. var site = $("#existingSite option:selected").val();
41. action = "not new";
42. } else {
43. var site = $("#site").val();
44. action = "new";
45. }
46. notes = $("#notes").val();
47. var name = $("#name option:selected").val();
48. var url = "http://192.168.1.247/checkIn.php";
49. var dataString = "site=" + site + "&notes=" + notes + "&name=" + name + "&action=" + action;
50. if ($.trim(site).length > 0 & $(name) !== "Please select your name")
51. {
52. $.ajax({
53. type: "POST",
54. url: url,
55. data: dataString,
56. crossDomain: true,
57. cache: false,
58. success: function (data) {
59. if (data === "Success")
60. {
61. $("#status").html('Checked in Successfully');
62. $("#site").val("");
63. $("#notes").val("");
64. } else if (data === "Error")
65. {
66. alert("error");
67. }
68. }
69. });
70. } else {
71. alert("Please make sure you have selected your name, entered a new or existing site, and entered any notes if required");
72. }
73. });
74. });
75. </script>
76. <script type="text/javascript"> //function to check out
77. $(document).ready(function ()
78. {
79. $("#checkOut").click(function () {
80. var site = $("#existingSite option:selected").val();
81. var notes = $("#notes").val();
82. var name = $("#name option:selected").val();
83. var url = "http://192.168.1.247/checkOut.php";
84. var dataString = "site=" + site + "&notes=" + notes + "&name=" + name;
85. if ($(site) !== "existingSite" & $(name) !== "Please select your name")
86. {
87. $.ajax({
88. type: "POST",
89. url: url,
90. data: dataString,
91. crossDomain: true,
92. cache: false,
93. success: function (data) {
94. if (data === "Success")
95. {
96. $("#status").html('Checked out Successfully');
97. $("#site").val("");
98. $("#notes").val("");
99. $("#name").val("Please select your name");
100. $("#existingSite").val("Please select your name");
101. } else if (data === "Error")
102. {
103. alert("error");
104. }
105. }
106. });
107. }
108. return false;
109. });
110. });
111. </script>
113. <title>test</title>
114. </head>
115. <body>
116. <div class="app">
117. <img src="data:image/png;base64," alt="NCC Logo">
118. <select id="name">
119. <option value="name">Please select your name</option>
120. </select>
121. <input type="text" placeholder="Please enter the site if new" id="site">
122. <select id="existingSite">
123. <option value="name" >or select existing site</option>
124. </select>
125. <textarea id="notes" placeholder="Please enter site notes here"></textarea>
126. <button id="checkIn">Check In</button>
127. <button id="checkOut">Check Out</button>
128. <div id="status"></div>
129. </div>
130. </body>
131. </html>

## 4.9 Proposed Design of the Mobile App v2

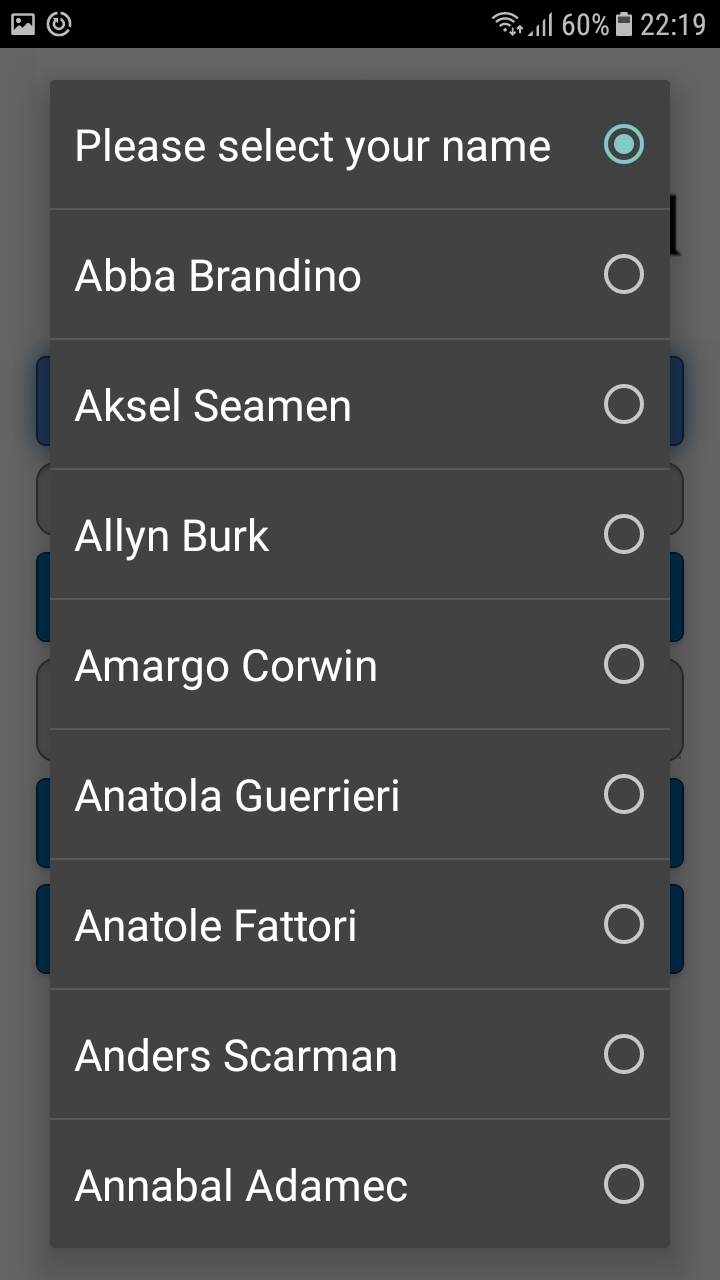


## 4.10 Screenshots of the Mobile App

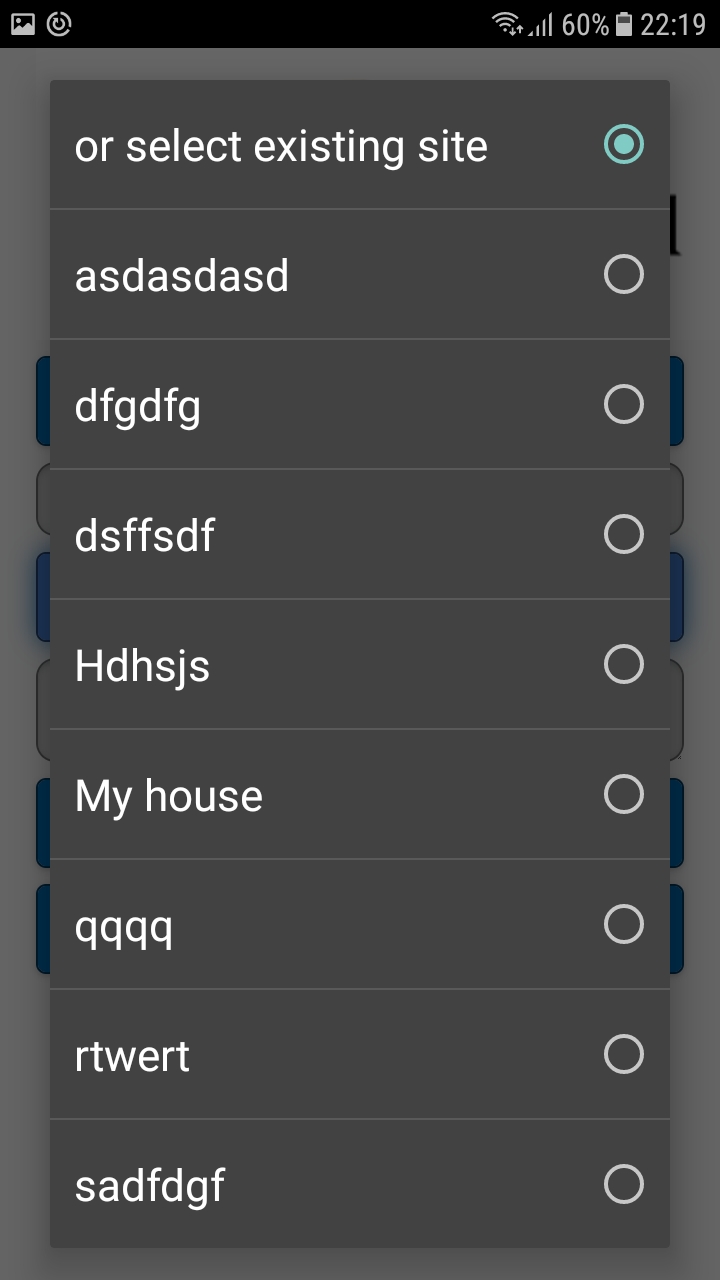
### 4.10.1 Main Screen



### 4.10.2 Menu Showing List of Names



### 4.10.3 Menu Showing Site List



## 4.11 PHP Code

### 4.11.1 db.php

1. <?php
2. header("Access-Control-Allow-Origin: \*");
3. $con = mysqli\_connect("localhost","root","","tm470") or die ("could not connect database");
4. ?>

### 4.11.2 getNames.php

1. <?php
2. include "db.php";
3. $data=array();
4. $q=mysqli\_query($con,"select name from `tblofficers` ORDER BY name ASC");
5. while ($row=mysqli\_fetch\_object($q)){
6. $data[]=$row;
7. }
8. echo json\_encode($data);
9. ?>

### 4.11.3 getSites.php

1. <?php
2. include "db.php";
3. $data=array();
4. $q=mysqli\_query($con,"select siteName from `tblsite` ORDER BY siteName ASC");
5. while ($row=mysqli\_fetch\_object($q)){
6. $data[]=$row;
7. }
8. echo json\_encode($data);
9. ?>

### 4.11.4 checkIn.php

1. <?php
2. include "db.php";
4. $site=$\_POST['site'];
5. $notes=$\_POST['notes'];
6. $name=$\_POST['name'];
7. $action=$\_POST['action'];
8. $event="in";

11. $sql = "INSERT INTO tbllogs (site, notes, officer, in\_out)
12. VALUES ('$site', '$notes', '$name', '$event')";
14. $sql2 = "INSERT INTO tblsite (siteName, officerName)
15. VALUES ('$site', '$name')";
17. $sql3 = "INSERT INTO tblcurrentvisits (site, name)
18. VALUES ('$site', '$name')";
20. if ($action === "new"){
21. $con->query($sql2);
22. }
24. if ($con->query($sql) === TRUE) {
25. ($con->query($sql3));
26. echo "Success";
27. } else {
28. echo "Error";
29. }
31. $con->close();
32. ?>

### 4.11.5 checkOut.php

1. <?php
2. include "db.php";
4. $site=$\_POST['site'];
5. $notes=$\_POST['notes'];
6. $name=$\_POST['name'];
7. $event="out";

10. $sql = "INSERT INTO tbllogs (site, notes, officer, in\_out)
11. VALUES ('$site', '$notes', '$name', '$event')";
13. $sql2 = "DELETE FROM tblcurrentvisits
14. WHERE name='$name'";
16. if (($con->query($sql) === TRUE) && ($con->query($sql2) === TRUE)) {
17. echo "Success";
18. } else {
19. echo "Error";
20. }
22. $con->close();
23. ?>