National College of Ireland

Bachelor of Science Honours Computing

Software Development

2022/2023

Marie Elizabeth Hauroo

19140410

x19140410@student.ncirl.ie

MyNCI

Technical Report

Table of Contents

Table of Figures

Executive Summary

The aim of this technical report is to describe the development of the MyNCI application, an Android mobile application designed to enhance the student experience at the National College of Ireland (NCI) by creating an integrated platform for academic and logistical needs. The report outlines the technology that was used in the development process, including Android Studio, Firebase, GitHub, and PenUp. It also defines the architecture of the application, which follows a client-server model, a modular architecture, and in Android studio, an MVC (Model, View, Controller) approach. A comprehensive explanation of the functional requirements is included in the form of the use cases.

The report describes the implementation of the key features of the MyNCI application, including the Room Booking system, the Map function and Parking feature amongst others. Each feature is described in detail, with code snippets and screenshots provided to demonstrate the functionality. The report highlights the importance of Firebase in the implementation of these features and the app's modular architecture, with each component serving a specific function and working independently from each other.

After the development, the app is thoroughly tested and documented within this report alongside an insightful evaluation.

The report concludes that the MyNCI application serves as a productivity space that becomes an integral part of being a student at NCI. The conclusion also acknowledges some of the limitations of the project.

1. Introduction

1.1 Background

The choice to take up this project was based on the fact that, as NCI students, there is currently no college application available to us. In today's digital age, students expect to have all the information they need at their fingertips.

While NCI makes great use of Moodle and Microsoft Teams, these platforms are only used in specific situations and times. Most students will only check their Moodle page if they explicitly need to use it or refer from it, similarly with teams. The MyNCI application would be a one-stop-shop for students, by providing key functionalities such as room booking, an interactive map of the college, new feeds tailored to the student's course and quick link access to Moodle, Teams, and Timetables.

I undertook this project because I believe that students should have access to a simple tool that can help them manage their academic and logistical needs, all the while making their campus experience better.

1.2 Aims

The MyNCI application aims to enhance the student experience at NCI by creating an integrated platform for academic and logistical needs. MyNCI would enhance productivity and access to essential information, this would in turn enable students to manage their academic experience with ease.

The interactive map function of the application is designed to provide a user-friendly interface to help students navigate the campus with ease, particularly for new students who may find it challenging to ask for directions.

Additionally, the app aims to reduce the administrative burden on the reception staff by streamlining the process of booking rooms, making the feature available to all students in an easy and simple booking process.

Overall, the application aims to be a productivity space that becomes an integral part of being a student at NCI.

1.3 Technology

To develop this application in Java, I used several technologies, including Android Studio, Firebase, GitHub, and PenUp.

Android studio is one of the most popular platforms for developing Android applications and has support for a variety of languages. It provided me with an integrated development environment (IDE) to create and debug the application. The emulator offered by Android Studio is an excellent tool that has helped with the debugging and visual appraisal aspect of the development. [1]

Firebase was used as the backend service, and I implemented Firebase's database and Firebase Firestore, to store user data such as accounts, profiles, images, and bookings. Firebase Authentication was used for user registration and login functions. The database is where the majority of the app’s core functionality happen, specifically for the booking system and newsfeeds, and therefor is a crucial component of this project. [2]

GitHub is my code hosting platform of choice for version control throughout the development of the application. The use of repositories, branches and commits has made it easier to keep the code organised and in working condition. Pushing only fully working code to master meant that I always had a working project to fall back on when a something went wrong on my branches. [3]

Lastly, I used PenUp to draw and colour the maps and icons for the app's interactive map feature. I searched unsuccessfully for a free software that would allow me to produce the floor plans in a simple manner, given that the only references I has were from scans of the plans. But most floor-planning software were requiring measurements and other details that were unavailable to me. So, in the end I used Samsung’s PenUp to draw on the overlay of the scans to make my maps as accurate as possible.

1.4 Structure

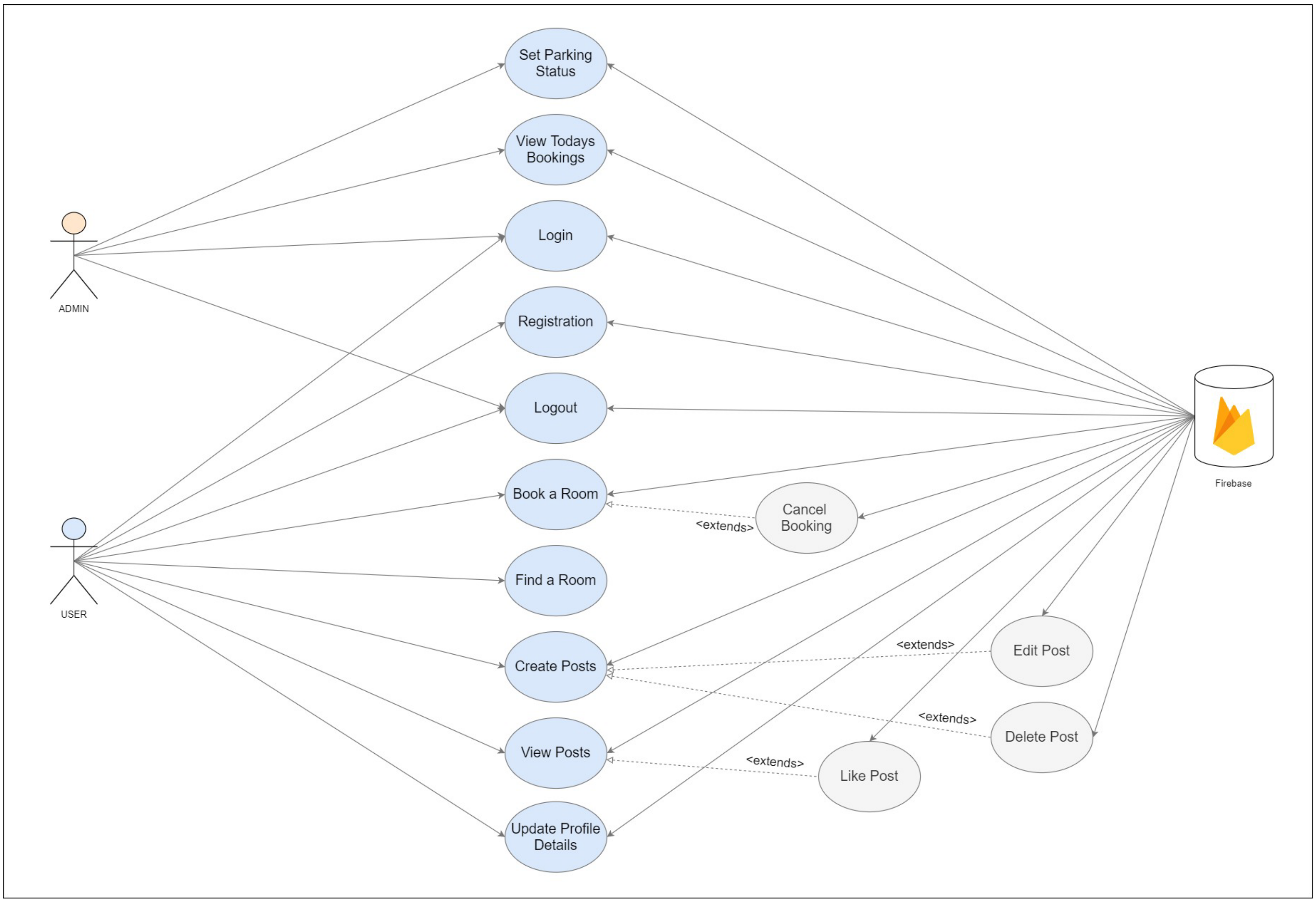
The high-level structure of this report is sectioned in four main parts; the Introduction, the System, the Conclusion, and the Further Development & Research. The System section is the most extensive segment of the report and covers all the requirements of the application (functional and non-functional), the Design & Architecture, the Implementation, the Graphical User Interface, and the Testing & Evaluation.

In each of the mentioned sections, multiple aspects of the project will be presented in both textual and visual representations in order to provide a well-rounded understanding of the logic and appearance of my project.

2. System

2.1 Requirements

Functional Requirements



MyNCI Figure 1 Use Case Diagram

2.1.1 Requirement 1: Register

Use Case Name	Register Account	Use Case ID	UC-1	Priority	High
Description	A user can only use the application if they have an account. Registration is mandatory.				
Actors	User				
Triggers	This use case starts when a user opens the application for the first time.				
Preconditions	A new user that does not yet have an account.				
Post-condition	A new account will be created for the user.				
Normal Flow	1. The user opens the application for the first time. 2. The system displays the register page. 3. The user fills in the required fields and click next. 4. The system displays the Setup page, prompting the user to add more details. 5. The user fills in their name, course code, student number and profile picture. 6. The system saves all of the information and saves it to database under "Users." 7. The system displays a success message and redirects the user to the Home page.				
Alternate Flow	**A1: User enters an invalid email:** 1. The user enters an email address that isn't an approved NCI email. 2. The system checks the email format. 3. The system displays an error prompting the user to use their NCI credentials.				
Exceptional Flow	**E1: System experiences an error while processing the registration.** 1. The user clicks the register button. 2. Due to a database connection error, the system fails to create the account. 3. The system displays the error message.				
Includes		Extends			
Notes	Only emails with the ending @student.ncirl.ie or @staff.ncirl.ie can be registered on this application.				

2.1.2 Requirement 2: Login

Use Case Name	Login	Use Case ID	UC-2	Priority	High
Description	A user must log into their account in order to use the application.				
Actors	User				
Triggers	The user opens the application.				
Preconditions	Must not be already logged in.				
Post-condition	User will be logged into their account and able to use the application.				
Normal Flow	1. The user opens the application. 2. The system displays the login page. 3. The user enters their email address and password and click the login button. 4. The system checks that there is an existing account for the email entered. 5. The system checks that the password entered matches the password saved to the account on firebase. 6. Once the checks pass, the user is logged in. 7. The system displays the home page of the application.				
Alternate Flow	**A1: Incorrect credentials:** 1. The user enters their email and the incorrect password. 2. The system checks the email and password and finds that they don't match. 3. The system does not log the user in. 4. The system displays a message indicating the password was incorrect.				
Exceptional Flow					
Includes		Extends			
Notes					

2.1.3 Requirement 3: Logout

Use Case Name	Logout	Use Case ID	UC-3	Priority	High
Description	A user must be able to terminate their session by logging out.				
Actors	User				
Triggers	The user clicks the "Logout" link on the side navigation bar.				
Preconditions	The user must be logged into the application.				
Post-condition	The session will be destroyed. The user is logged out.				
Normal Flow	1. The user goes to the side bar navigation and clicks "logout". 2. The system processes the logout request. 3. The system redirects the user to the Login page. 4. The user is now logged out.				
Alternate Flow					
Exceptional Flow	E1: Logout Error: 1. The user clicks the logout button. 2. The system attempts to clear any stored authentication tokens. 3. The system encounters an error, it fails to clear the tokens. 4. The system displays the error message. 5. The user is still logged in				
Includes		Extends			
Notes					

2.1.4 Requirement 4: Book a Room

Use Case Name	Book a Room	Use Case ID	UC-4	Priority	High
Description	This use case describes the sequence of steps the user must follow to book a room on the application.				
Actors	User				
Triggers	This use case starts when a user selects the “Book Room” tile on the Home page.				
Preconditions	The user must be logged into the application.				
Post-condition	The room has been booked for the selected timeslot by the user. Redirected to Home page.				
Normal Flow	1. The system shows the booking form when the user selects the “Book Room” tile. 2. The user selects the room they want from six choices, then click Next. 3. The user selects a reason for the booking. 4. The user selects a date on the calendar. 5. The system shows the available time slots for the given date. 6. The user selects a time slot and clicks Next. 7. The system displays the booking information, waiting for confirmation. 8. The user confirms the booking. 9. The System registers the booking to the user’s ID and stores the details to firebase. 10. The system sets the selected timeslot (room number, date, time) to *isBooked()* 11. Booking is confirmed.				
Alternate Flow	**A1: Unavailable date.** 1. If the date or time selected is not available, the system will loop back to step 4 of the main flow.				
Exceptional Flow	**E1: System experiences an error while processing the booking.** 4. The user confirms the booking. 5. Due to a database connection error, the system fails to save the booking to firebase. 6. The system displays an error message to the user. 7. The system redirects the user to step 1 of the main flow.				
Includes		Extends			
Notes					

2.1.5 Requirement 5: View and Cancel Bookings

Use Case Name	View and Cancel Bookings	Use Case ID	UC-5	Priority	Medium
Description	Users of the application have the ability to view and cancel their own room bookings.				
Actors	User				
Triggers	Going to the User Profile page.				
Preconditions	User must have previously made a booking.				
Post-condition	User can view and cancel their bookings.				
Normal Flow	1. The user navigates to their Profile page using the bottom navigation bar. 2. The system determines the current user ID and loads the relevant information. a. The current user data (Full name, Student Number, Course Code, Picture) b. A list of the current user’s bookings 3. The user can see all of their bookings. 4. The user clicks on a booking. 5. The system displays a cancellation window asking the user if they want to cancel it. 6. The user selects “yes.” 7. The system deletes the selected booking from the user’s page, and firebase. 8. The system makes the timeslot available again for booking.				
Alternate Flow	**A1: The user has made no bookings.** 1. The user navigates to their Profile page using the bottom navigation bar. 2. The system determines the current user ID and loads the relevant information. a. The current user data (Full name, Student Number, Course Code, Picture) 3. The system finds no booking for the user, therefore displays nothing.				
Exceptional Flow	**E1: The system encounters an error when cancelling the booking.** 1. On the profile page, the user clicks on a booking. 2. The system displays a cancellation window asking the user if they want to cancel it. 3. The user selects “yes.” 4. The system attempts to delete the selected booking from the user’s page, and firebase. 5. Due to a database connection error, the system is unable to complete the delete. 6. The system displays an error message of unsuccessful cancelation. 7. Redirected to the profile page and resume from step 2 of the normal flow.				
Includes		Extends	[UC-4] Booking a Room		
Notes					

2.1.6 Requirement 6: Admin – View Today's Bookings

Use Case Name	Admin – View Today's Bookings	Use Case ID	UC-6	Priority	High
Description	In order to prepare the rooms, an admin account holder should be able to view all of the bookings for the current date.				
Actors	Admin				
Triggers	Going to the Admin page.				
Preconditions	Must be logged in as Admin.				
Post-condition	Today's bookings are accessible to the admin.				
Normal Flow	1. Admin is logged into the application and directed to the Admin page. 2. The system checks today's date and retrieves all the bookings from firebase that have a matching bookingDate. 3. The system fetches the bookings and their relevant info (room number, timeslot, reason for the booking, name of the user who booked the room.) 4. The system displays today's bookings on the admin page.				
Alternate Flow	**A1: There are no bookings for today's date:** 1. The system checks today's date and retrieves all the bookings from firebase that have a matching bookingDate. 2. No bookings were found matching today's date, the system displays nothing in the bookings view.				
Exceptional Flow					
Includes		Extends	[UC-4] Booking a Room		
Notes					

2.1.7 Requirement 7: Find a Room

Use Case Name	Find a Room	Use Case ID	UC-7	Priority	High
Description	The user should be able to search for a specific room or office and be shown where in the building that room is situated.				
Actors	User				
Triggers	Navigating to the Maps tile on the Home page.				
Preconditions	User must be logged in.				
Post-condition	The user will be shown the correct map depending on their search.				
Normal Flow	1. The user navigates to the Maps tile. 2. The system loads the Maps page, and displays a map of the ground floor by default. 3. The user searches for a particular room, or office by typing the name/room in the search field 4. The system compares the entry in the search fields and autocompletes with some suggestions. 5. The user selects the room they are looking for. 6. The system reads the input in the search field and finds the corresponding map image to display in the main view.				
Alternate Flow	**A1: User selects a room from the dropdown menu:** 1. The user navigates to the Maps tile. 2. The system loads the Maps page and displays a map of the ground floor by default. 3. The user selects a room from the dropdown menu. 4. The system registers the selection and finds the corresponding map. **A2: User clicks one of the four floor buttons:**				

	1. The user navigates to the Maps tile. 2. The system loads the Maps page and displays a map of the ground floor by default. 3. The user clicks one of the four floor buttons. 4. The system displays the map of the selected floor.		
Exceptional Flow			
Includes		Extends	
Notes			

2.1.8 Requirement 8: Create Post

Use Case Name	Create Post	Use Case ID	UC-8	Priority	High
Description	User should be able to create a new post and select on which feed the post should appear. (General or Course)				
Actors	User				
Triggers	The user clicks the “Add Post” floating button on either the NCI Feed or the Course Feed.				
Preconditions	The user must be on one of the feed pages, NCI or My Course.				
Post-condition	A new post will be created, and viewable on the correct feed.				
Normal Flow	1. The user clicks on the floating button to show the pop-up window. 2. The user fills in the content of their post and selects the feed they want the post to appear on. 3. The user must select at least one feed but can select both if they wish. 4. The user clicks on the “create post” button. 5. The system saves the post and the relevant info to firebase (content, feed, author, time and date etc.) 6. The system displays a success message, and the post is added.				
Alternate Flow					
Exceptional Flow	**E1: Failure to create post.** 1. The user clicks on the “create post” button, after having completed the form. 2. Due to a database connection error, the system fails to save the post to firebase. 3. The system displays a message with the error that occurred.				
Includes		Extends			
Notes					

2.1.9 Requirement 9: Edit or Delete Post

Use Case Name	Edit or Delete Post	Use Case ID	UC-9	Priority	High
Description	A user should be able to edit or delete their own posts.				
Actors	User				
Triggers	The user clicks on one of their own posts.				
Preconditions	The user must have previously created a post.				
Post-condition	The selected post will be updated or deleted, depending on the action taken.				
Normal Flow	**Edit Post:** 1. The user clicks on one of their own posts. 2. The system displays a popup of the selected post. 3. The update and delete button are presented. 4. The user clicks “update” 5. The system displays the post content in edit mode. 6. The user makes the necessary changes and clicks “save”.				

	7. Changes are saved on firebase. 8. The system displays a success message. **Delete Post:** 1. The user clicks on one of their own posts. 2. The system displays a popup of the selected post. 3. The update and delete button are presented. 4. The user clicks “delete”. 5. The system displays a delete confirmation popup. 6. The user confirms the delete. 7. The system deletes the selected post from firebase and the application.		
Alternate Flow			
Exceptional Flow			
Includes		Extends	[UC-8] Create Post
Notes			

2.1.10 Requirement 10: View Feeds

Use Case Name	View Feeds	Use Case ID	UC-10	Priority	High
Description	The user should be able to view all the posts that are in the NCI feed and their own Course feed.				
Actors	User				
Triggers	The user navigates to either of the two feed pages.				
Preconditions	Posts must have been previously created by some users.				
Post-condition	Posts are visible and filtered according to feed.				
Normal Flow	General Feed: 1. The user navigates to the NCI feed. 2. The system filters through every post on the database that have the field isGeneral set to true. 3. The system displays all these posts to the NCI feed page. 4. The user is able to view every post on this feed. 5. The user is able to like posts. My Course Feed: 1. The user navigates to the My Course feed. 2. The system fetches the current user’s course code. 3. The system filters through every post on the database that have the field isCourse set to true. 4. From that result, the system filters out every post that have a matching course code to that of the current user. 5. The system displays all the filtered posts. 6. The user is able to see all the posts that are specified for this course. 7. The user is able to like posts on this feed.				
Alternate Flow					
Exceptional Flow					
Includes		Extends	[UC-8] Create Post		
Notes					

2.1.11 Requirement 11: Like Posts

Use Case Name	Like Posts	Use Case ID	UC-11	Priority	High
Description	A user should be able to like any posts on both the NCI feed and the Course feed.				
Actors	User				
Triggers	The user is on one of the two Feed pages.				
Preconditions	There must be previously created posts on the feeds.				
Post-condition	The like count on the post will increment by 1.				
Normal Flow	1. The user is on one of the feed pages. 2. The user clicks on the like (star) button on a post they like. 3. The system registers the click and increments the like count by 1. 4. The like button should now be in a selected state. (Blue)				
Alternate Flow	A1: User has already liked the post: 1. The user clicks on the like button of a post they've already liked. 2. The system sees that they have already liked the post. 3. The system decreases the like count by one. 4. The like button should now be in an unselected state (white)				
Exceptional Flow					
Includes		Extends	[UC-10] View Feeds		
Notes					

2.1.12 Requirement 12: Parking Status

Use Case Name	Parking Status	Use Case ID	UC-12	Priority	High
Description	This function keeps the user up-to-date with the parking status of the establishment. It is set by the administrator of the application.				
Actors	Admin, User				
Triggers	Admin: setting the parking flag on the admin page. User: Navigating to the Home page of the app.				
Preconditions					
Post-condition					
Normal Flow	Admin: 1. The admin is logged in. 2. On the admin page, the admin flicks the switch to indicate if the parking is full or not. 3. Depending on the state of the switch, firebase sets the ParkingFull to either true or false. User: 1. On every load of the home page, the system checks firebase for the state of ParkingFull (true/false) 2. Depending on the state of ParkingFull, the system displays the correct message. 3. The user is able to see if parking is currently full from the Home page.				
Alternate Flow					
Exceptional Flow					
Includes		Extends			
Notes					

2.1.13 Requirement 13: Update User Details

Use Case Name	Update User Details	Use Case ID	UC-13	Priority	Medium
Description	The user should be able to update some of their personal details: Course Code, Profile Picture, Password.				
Actors	User				
Triggers	The user navigates to the settings page.				
Preconditions					
Post-condition					
Normal Flow	Course Code : 1. On the settings page, the user clicks on the "Update Course Code" 2. The system displays a popup in edit mode, with the current course code preloaded. 3. The user makes the necessary changes and clicks save. 4. The system sends the updated course code to firebase to replace the old one. 5. The popup closes and the new course code should be visible in the header, and throughout the app. Profile Picture : 1. On the settings page, the user clicks on the "Update Profile Picture". 2. The system opens up the gallery picker on the user's device. 3. The user selects the desired picture. 4. The system saves the new picture to firebase to replace the old one. 5. The popup closes and the new picture should be visible in the header, and throughout the app. Password: 1. On the settings page, the user clicks on the "Change Password". 2. The system displays a popup with three fields. Current Password, New Password, Confirm New Password 3. The user fills in all three fields and clicks save. 4. When the save button is clicked, a. the system checks if the password entered matches the user's current password. b. the system checks if the new password and the Confirm new password match. 5. If the previous criteria pass, the password is updated on firebase.				
Alternate Flow					
Exceptional Flow					
Includes		Extends			
Notes					

Data Requirements

Here are the data requirements for the MyNCI application in order to provide an optimal experience to its users:

1. User data:
The application requires personal information such as name, NCI email, and password for user registration and authentication purposes.

2. Room booking data:
The application needs to store information about the rooms available for booking, their location, and the time slots when they are available.

3. User profile data:
The application needs to store user profile data, including name of the user, profile picture, course information, and any other relevant details.

4. Map data:
The interactive map feature requires data such as the building layouts, room locations, and other important points of interest on the campus.

5. Post data:
The application needs to store user-generated posts and their associated data, including the author, date, course, and post content.

6. Parking data:
The parking feature of the application requires real-time data on the availability of parking spaces on the campus. For this application, this will manually be toggled on/off by the Admin.

Usability Requirements

The MyNCI application is designed in such a way that it is intuitive to use. All of the main features available in the application are clearly displayed on the main page of the application with labels. MyNCI offers four main services. There is a searchable map of the college, the ability to book a time slot to use a room, an NCI announcement page, and a feed that shows all updates related to the user’s course. A user can choose to book a room in NCI, or they can view a full layout of the college building, with search capabilities for specific room to see where they need to go, and they can have access to their course news feed where updates can be made by themselves or other members of their cohort.

The user must be able to freely navigate between these pages, either through the tiles that are on the home page, or through the two navigation options (bottom navigation bar, side navigation bar). The bottom navigation bar is present on every page of the application to make it easy to access the commonly used pages, i.e.: Home, Profile, Booking. The side menu is displayed on the home page, and within this menu Settings, Moodle, Teams, and Timetable are made available, as well as the logout function. I reuse the code for the bottom navigation bar and the side menu throughout the application because the feature is necessary but repetitive.

Environmental Requirements

1. <u>Operating System:</u>
   The MyNCI application is designed for Android devices and requires Android 5.0 or later to run smoothly.

2. <u>Internet connectivity:</u>
   The application requires a stable internet connection to access and retrieve data from the Firebase database, as well as to display real-time information such as up-to-date bookings, feed, and parking status.

3. <u>Battery life:</u>
   The application should not significantly drain the battery of the user's device to ensure uninterrupted use throughout the day.

4. <u>Screen size:</u>
   The application should be optimized for various screen sizes to ensure that all content is visible and easily accessible.

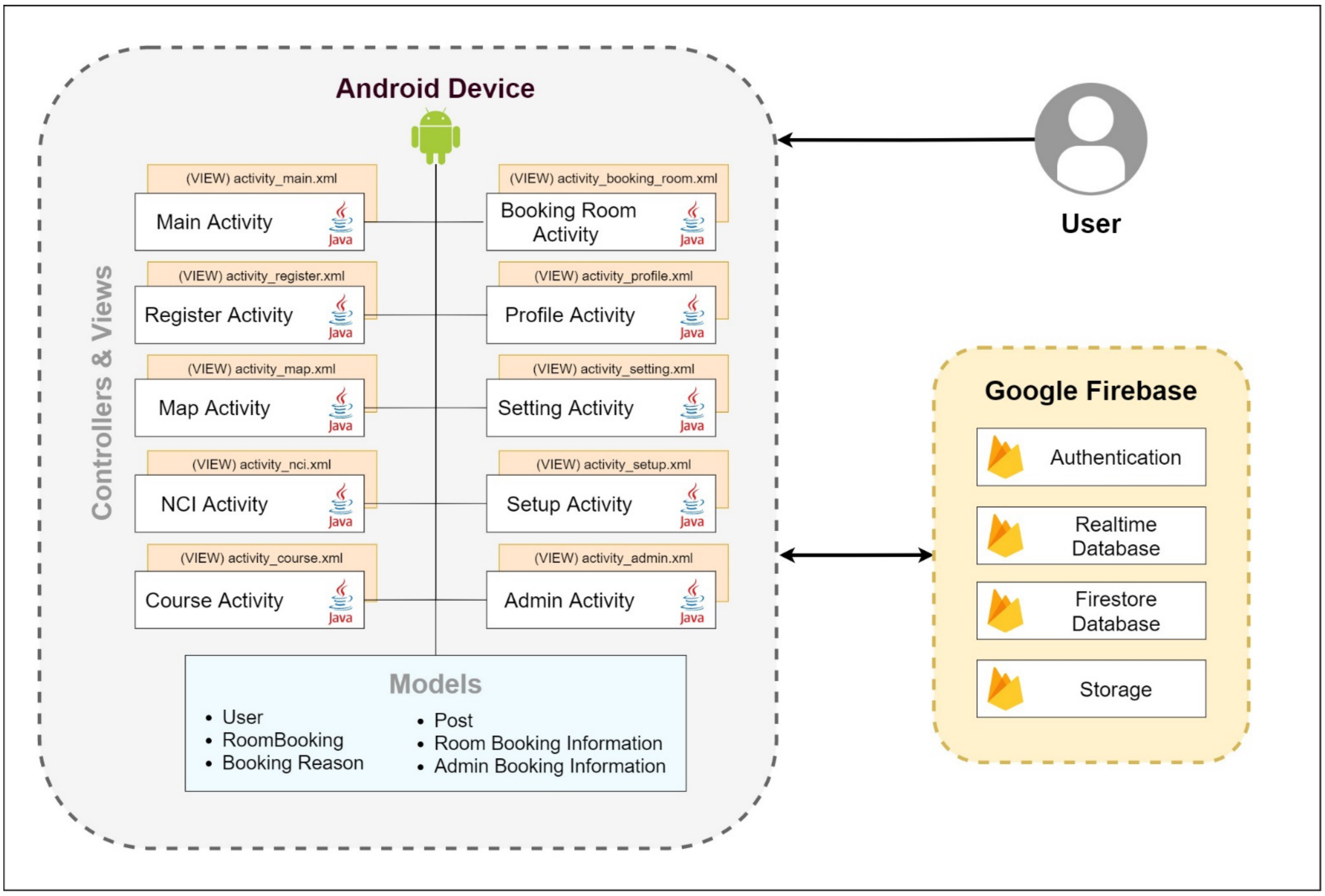
5. <u>Security:</u>
   The application should have secure authentication and data storage mechanisms to ensure the confidentiality and integrity of user data.

2.2 Design & Architecture

It is in the nature of most mobile application to have a dependency on databases and server-side monitoring, and the MyNCI application is no exception to that rule. In order to work smoothly and successfully, it is essential to have the proper architecture and design in place.

The MyNCI application is designed with a client-server architecture, with the Android app serving as the client and Firebase serving as the server. The Android app is built using the Model-View-Controller (MVC) architecture, separating the presentation logic from the business logic. This can be seen in the Activity java classes (Controller), which house all of the logic, being separated from the XML layout files (View), which sets up all the user interface. And the Models are where we handle the real-world business rules and communicate with the database. [4]

The main components used in the app include Firebase Authentication, Firestore, and Firebase Database. Using Google's Firebase allows for a sizeable delegation of the workload. Firestore is used as the primary database for storing user-generated data such as user profiles, bookings, and posts. This is particularly helpful for repetitive tasks of the application.



MyNCI Figure 2 Architecture Diagram (MVC)

The three primary algorithms of the MyNCI application are the interactive map function, the room booking system, and the creation of posts (CRUD functionality).

The Interactive Map functionality takes the user input, which can be a room number or name, and from that input the system shows the relevant room on the relevant floor on the map that is displayed to the user. The list of room names and images are hard coded within the application inside two array lists that can be found in Strings.xml

The Booking System is also another major feature of MyNCI and uses Firebase to check all the rooms that are available for reservations, and also checks if a selected room has already been booked by someone else on any given day.

Lastly the Posts functionality is another major algorithm used in this application to manage user-generated posts and display them on the appropriate feeds by using a filter to sort through all the posts according to the user's current course, and also checking which feed was selected at the time of creation. This is why Firebase is the central component of this application, it allows us to store the users' bookings and new posts created.

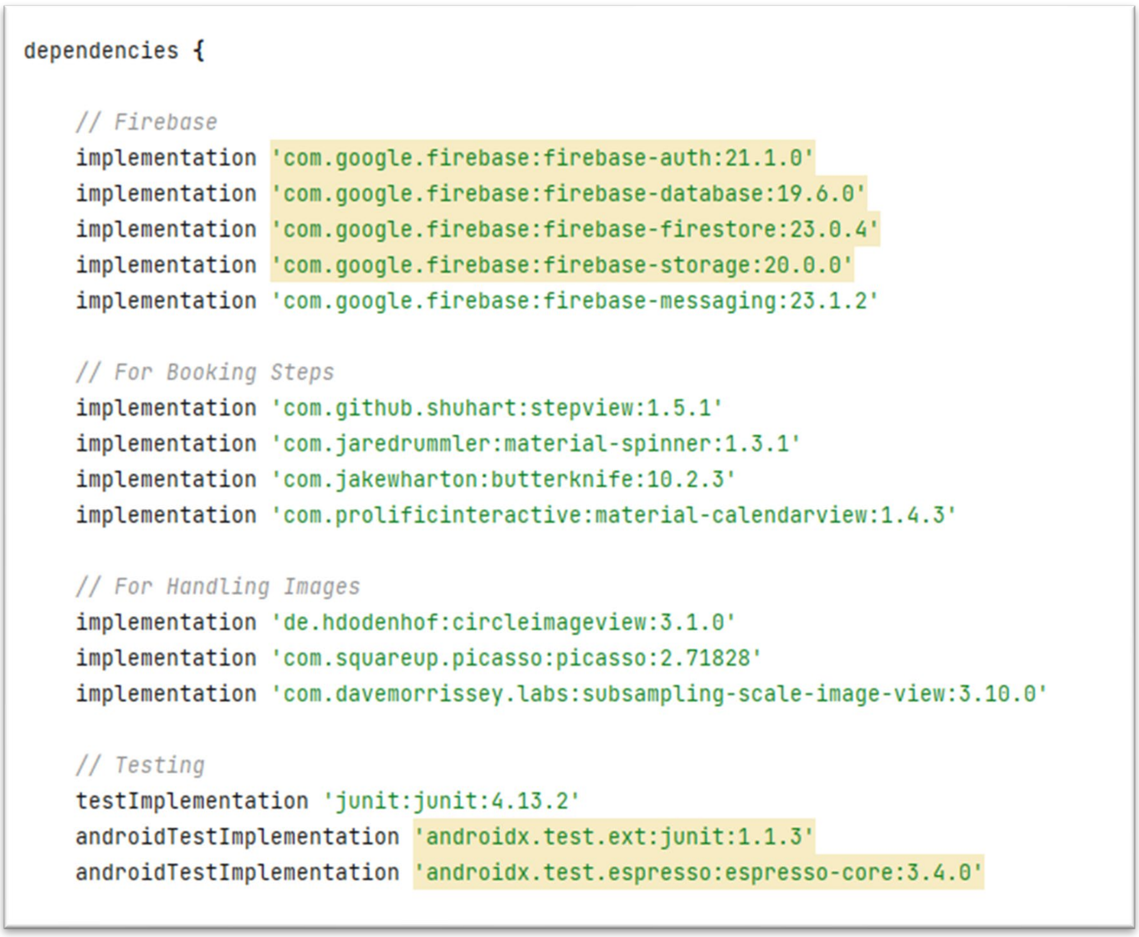
Overall, MyNCI is designed with a modular architecture, with each component serving a specific function and work independently from each other.

2.3 Implementation

This section will cover the implementation of the main components of the MyNCI application. It will go through a thorough explanation of the main logic alongside screenshots and code snippets. Each feature will have their own individual section.

The database plays a crucial part in the MyNCI app and is often the main foundation of most of the major functions in the application. This is why Firebase was chosen as the cloud database tool because it is robust, and has a unique combination of complexity, usability and flexibility. Firebase will be mentioned in several of the feature implementations.

Several other tools were used in order to meet the requirements of MyNCI, these include Picasso, Calendar View, and several others. Below are some noteworthy dependencies that were needed for the application.



```
dependencies {

    // Firebase
    implementation 'com.google.firebase:firebase-auth:21.1.0'
    implementation 'com.google.firebase:firebase-database:19.6.0'
    implementation 'com.google.firebase:firebase-firestore:23.0.4'
    implementation 'com.google.firebase:firebase-storage:20.0.0'
    implementation 'com.google.firebase:firebase-messaging:23.1.2'

    // For Booking Steps
    implementation 'com.github.shuhart:stepview:1.5.1'
    implementation 'com.jaredrummler:material-spinner:1.3.1'
    implementation 'com.jakewharton:butterknife:10.2.3'
    implementation 'com.prolificinteractive:material-calendarview:1.4.3'

    // For Handling Images
    implementation 'de.hdodenhof:circleimageview:3.1.0'
    implementation 'com.squareup.picasso:picasso:2.71828'
    implementation 'com.davemorrissey.labs:subsampling-scale-image-view:3.10.0'

    // Testing
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
```

MyNCI Figure 3 Noteworthy Dependencies used within the application.

Registration

For the registration functionality, Firebase Authentication was used, and it bears the heaviest load of the work. The Registration process has two steps for MyNCI. First there is the registration page which prompts the user for their name, NCI email, and password. And the second step is a Setup Page where the application gathers further information about the user, this includes their profile picture, student number and course code. The reason that it is done in two steps is so that the user details can be accessed from the Realtime Database and separating it from Firebase Authentication.

```
private void PerformAuthentication() {
    String email=emailTF.getText().toString();
    String password=pswdTF.getText().toString();
    String confirmPassward = confirmPswdTF.getText().toString();

    if(!email.matches(emailPattern)){
        emailTF.setError("Enter valid email");
    }else if (!email.endsWith(emailDomain)) {
        emailTF.setError("Email must end with " + emailDomain);
    }else if(password.isEmpty()){
        pswdTF.setError("Please Enter a Password");
    }else if(password.length()<6){
        pswdTF.setError("Password must have at least 6 characters");
    }else if(!password.equals(confirmPassward)){
        confirmPswdTF.setError("Both passwords do not match");
    }else{

        mAuth.createUserWithEmailAndPassword(email,
password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {
                if(task.isSuccessful()){
                    sendSetupActivity();
                    // show success message
                }else{
                    // show error message
                }
            }
        });
    }
}
```

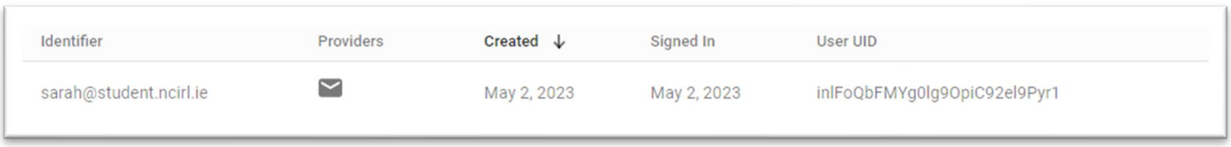
Code Snippet 1: Registration

The above code snippet shows the main method that performs the registration of a user. The method grabs the user's input from the text fields and runs them through a series of validations. Some validations worth pointing out are:

- The checks that the email used for registration should end with the correct `emailDomain` and follow the correct `emailPattern`, which in this case would be "xxxxx@xxxxx.ncirl.ie"
- The password entered in both password fields must match so that the user is certain that there was no mistake in their password creation.

Following the checks, the method moves onto the Firebase function that handles the creation of an account: `createUserWithEmailAndPassword()`

A successful registration results in a success message being displayed to the user and they are moved to the Setup page where they can add their profile picture, course code and student number. Every time a new account is created Firebase Authentication creates a new entry that looks like this:



Identifier	Providers	Created ↓	Signed In	User UID
sarah@student.ncirl.ie	✉	May 2, 2023	May 2, 2023	inlFoQbFMYg0lg9OpiC92el9Pyr1

MyNCI Figure 4 : Registration on Firebase Authentication

Login

The login function is essential to the application, because once a user is registered, they should be able to log into their account whenever they need to, from any android device they have at hand. Again, for this function, Firebase Authentication does a great deal of the work.

```
if(!email.matches(nciPattern)){
    emailTF.setError("Enter valid email");
}else if(password.isEmpty()){
    pswdTF.setError("Please Enter a
Password");
}else if(password.length()<6){
    pswdTF.setError("Password must have at
least 6 characters");
}else {

// if checks pass, then perform the login
// signInWithEmailAndPassword()

}
```

Code Snippet 3: Validations on Login

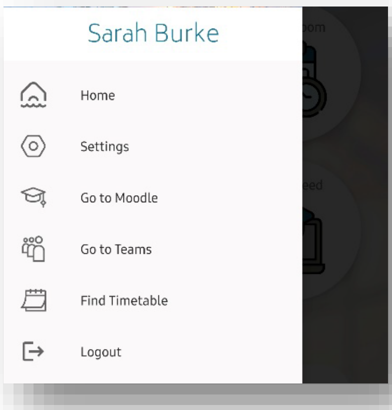
```
// inside of signInWithEmailAndPassword()

@Override
public void onDataChange(@NonNull
DataSnapshot dataSnapshot) {
    Boolean isAdmin =
dataSnapshot.getValue(Boolean.class);
    if (isAdmin != null && isAdmin) {
        sendAdminActivity(); // If the user
is an admin, send them to the AdminActivity
    } else {
        sendHomeActivity(); // If the user is
not an admin, send them to the HomeActivity
    }
}
```

Code Snippet 2: Checking for Admin at Login

The snippet on the left shows that the app checks that all the fields are passing the validations and no fields are left blank before moving onto the main Login method. The second snippet on the right is within the signInWithEmailAndPassword() method in the onSuccess() method. Here we are checking the account that we are trying to log into, and check is the account is an admin account. Depending on the condition of the if statement, the app should display the relevant page. If an admin is logging in, they are presented with the Admin page, but if it is just a normal user, then they should be directed to the Home page of the application.

Logout

This feature is simple but very important in the MyNCI application. Every user should be able to terminate their session if they wish. The logout button is located in the side navigation bar. The logout functionality only requires a short bit of code that is again using Firebase Authentication, but it is crucial and that is why the logout function is included. After the user has logged out, the app should display the Login page.



GUI 1: Side Navigation, Logout

```
case R.id.nav_logout:

    mAuth.signOut();
    SendUserToLogin();

    break;
```

Code Snippet 4: Logout

Room Booking

The Room Booking functionality inside MyNCI is one of the core features of the application and is rightfully made up of an extensive amount of code. For the implementation, the most relevant snippets will be shown. The Booking function occurs over three steps, and in order to implement this, fragments were used alongside of a step view to allow the user to navigate to each of the booking steps.

The first fragment is responsible for showing the six rooms that are available for bookings. These rooms and their details are set on Firebase Firestore under “BookableRooms” and include a description, roomNumber, and size. The rooms are fetched from Firestore and displayed in a recycler view, and each item has an OnClickListener(). When a room has been picked, we save that item as `selectedRoom`, and pass that variable onto the next fragment. It is also worth noting that the “Next” button is set to “disabled” until a room has been selected.

```
private void setupRecyclerView(List<RoomBooking> rooms) {
    RoomAdapter.OnRoomClickListener roomClickListener = new
RoomAdapter.OnRoomClickListener() {
        @Override
        public void onRoomClick(int position) {
            selectedRoom = rooms.get(position);
            nextButton.setEnabled(true);
        }
    };
    roomAdapter = new RoomAdapter(rooms, roomClickListener);
    roomsRecyclerView.setAdapter(roomAdapter);
}
```

Code Snippet 5: Fragment 1, Select a Room

The second fragment is where the core of the Booking Information is set. Here the user must choose a reason for booking the `selectedRoom`, and they pick the date that they want to book. Once they select a date, the timeslots for that day are displayed. There are six timeslots, and depending on whether the timeslot is available, the user can select the timeslot they need. Only when the user has selected all three (reason, date, and timeslot) then the "Next" button is enabled.

```
private void fetchBookingsAndUpdateTimeslots(String selectedDate) {
    String selectedRoomNumber = ((BookingRoomActivity)
getActivity()).getSelectedRoom().getRoomNumber(); // fetching room number from frag 1

    db.collection("RoomBookings")
            .whereEqualTo("room_number", selectedRoomNumber)
            .whereEqualTo("bookingDate", selectedDate)
            .get()
            .addOnCompleteListener(task -> {
                if (task.isSuccessful()) {
                    List<TimeslotBooking> bookedTimeslots = new ArrayList<>();
                    for (QueryDocumentSnapshot document : task.getResult()) {
                        TimeslotBooking timeslotBooking =
document.toObject(TimeslotBooking.class);
                        bookedTimeslots.add(timeslotBooking);
                    }
                    updateTimeslotsForSelectedDate(selectedDate, bookedTimeslots);
                } else {
                    Log.d("Fragment2Booking", "Error getting documents: ",
task.getException());
                }
            });
}
```

Code Snippet 6: Fetching the timeslots of the date selected (from the calendar) for the room that was selected in Fragment 1

The last fragment is the confirmation step, this is where all of the booking information selected so far is presented to the user for confirmation. We pull all the details from the last two fragments and display them in the view. These details are also staged for uploading to Firestore under the collection "RoomBookings" once the user confirms their choice.

```
private void initializeData() {
    selectedDateString = getArguments().getString("selectedDateString");
    selectedTimeslotString = getArguments().getString("selectedTimeslotString");
    selectedTimeslot = (TimeslotBooking)
getArguments().getSerializable("selectedTimeslot");
    bookingReason = (BookingReason) getArguments().getSerializable("bookingReason");
    selectedRoom = (RoomBooking) getArguments().getSerializable("selectedRoom");
}
```

Code Snippet 7: Fetching all the information set in the previous fragments (Booking Steps 1 & 2)

```
frag3BtnConfirm.setOnClickListener(v -> {
    if (selectedTimeslot != null && bookingReason != null && selectedRoom != null) {
        String bookingID = UUID.randomUUID().toString(); // creating random booking id

        if (roomID != null ) {
            if (firestore != null) {
                Map<String, Object> bookingData = new HashMap<>();
                bookingData.put("bookingID", bookingID);
                bookingData.put("currentUserName",
frag3UserNameTxt.getText().toString());
                bookingData.put("room_number", frag3RoomNumber.getText().toString());
                bookingData.put("BookingReason", bookingReason.name());
                bookingData.put("roomSize", frag3RoomSize.getText().toString());
                bookingData.put("bookingDate", selectedDateString);
                bookingData.put("bookingTime", selectedTimeslot.getBookingTime());
                bookingData.put("userID", currentUserID);

                firestore.collection("RoomBookings").add(bookingData)
                        .addOnSuccessListener(documentReference -> {
                               // show success message
                               // redirect to home page
                        })
                        .addOnFailureListener(e -> { //show error message });
                saveTimeslotToFirebase();
            } else { //show error message }
        } else { //show error message }
    }
});
```

Code Snippet 8: Saving the booking information as an object and save it to Firestore.

In Code Snippet 8, we can see the final stage of the Booking function. When the user clicks the "Confirm" button, we save all of the relevant information including the user's name, and a unique booking ID, in a new object called BookingData. The unique booking ID is essential for when we want to cancel a booking in the future. We then use bookingData object and add it to our "RoomBookings" collection on Firebase.

After the confirm button has been clicked, and the booking succeeds, the app redirects to the home page and the new booking is now available on firebase and the timeslot for the selected room is set to isBooked(true).

Room Search

Initially the idea for Room Search was to have one image for each floor and depending on which room was selected, a coloured frame would appear in the right spot on the image of the corresponding floor. And while this method would have probably been more efficient, the actual implementation proved to be quite difficult as the exact X and Y values would have to be accurately calculated for every single room on every single floor.

Since the plans available to me were not easily usable, they needed to be redone in a less cluttered fashion. I could not manage to find a suitable and free software that would allow me to recreate the plans in a clearer and outlined image. In the end, I used Samsung's PenUp drawing app to trace over the maps that I procured, and only traced the outlines of the rooms and adding the room numbers. It was a lengthy process, and not as professional as I was seeking, but it served the purpose.

It is also worth noting that the plans that I received were from before some structural changes in the building, and while I went to the college to try and map the rooms to the best of my ability, access to certain rooms are rightfully limited, therefore the accuracy of some of the offices may be slightly lacking.

As a result, the solution used for the Room Search functionality was to have an image of the floor plan for every single room in the building, where in each image, one room is shaded in. To implement this, a list of all of the rooms (names) in NCI were saved in a string array: "map_room_names_for_search" and a list of all the images were saved in another array: "map_images". It is paramount that the order of both lists match each other as the same index number will be used on both to perform the search.

These arrays were saved on the application itself inside the strings.xml.

```xml
<string-array
name="map_room_names_for_search">
  <!--Ground Floor-->
  <item>Ground Floor</item>
  <item>Ground: Library</item>
  <item>GROUND: Kelly Theatre</item>
  <item>GROUND: Fusion </item>
  <item>GROUND: Career Dev </item>
  <item>GROUND: Reception</item>
  <item>GROUND: Student Services</item>
  <item>GROUND: Maths|Dev Support</item>
    <!--1st Floor-->
    <!--2nd Floor-->
    <!--3rd Floor-->
    <!--4th Floor-->
</string-array>
```

Code Snippet 9: Room Names

```xml
<string-array name=
"map_images">
    <!--Ground Floor-->
    <item>nci_0_0_ground</item>
    <item>nci_0_2_library</item>
    <item>nci_0_1_kelly_theatre</item>
    <item>nci_0_3_fusion</item>
    <item>nci_0_4_centre</item>
    <item>nci_0_5_reception</item>
    <item>nci_0_6_service</item>
    <item>nci_0_7_support</item>
    <!--1st Floor-->
    <!--2nd Floor-->
    <!--3rd Floor-->
    <!--4th Floor-->
</string-array>
```

Code Snippet 10: Room Images

With the array lists set up so that each room name has a corresponding image, the next step is to set up the search function. The user can search for a room in two ways, they can either browse through the list of rooms in the dropdown option, or they can search by room name. When the user has selected the desired room, then the corresponding image is displayed in the `imageBox`. The image box is a zoomable view, allowing the user to pinch into the map for a closer look.

```java
private void updateImageView(String selectedItem) {
    int index = -1;
    for (int i = 0; i < map_room_names_for_search.length; i++) {
        if (map_room_names_for_search[i].equals(selectedItem)) {
            index = i;
            break;
        }
    }

    if (index >= 0) {
        imageBox.setImage(ImageSource.resource(map_images[index]));
    }
}
```

Code Snippet 11: Updates the image displayed in the Image Box based on the selected item.

Create Post

Creating a post is made available through a floating button that is situated in each of the two feed pages (General feed, Course feed). Once the user clicks on app post, they are prompted to fill in their text content and select which feed they would like their post to appear on, they must select atg least one feed, but can also select both. If they select the Course feed, then the post will be visible in the Course feed of every user that has a matching course code e.g.: BSHCSD4. If they select the General feed, then their post will be visible to every single user of the MyNCI app.

Creating posts is another function that is heavily reliant on Firebase, as every post created is added to the "Posts" database with relevant info such as a timestamp of the post, the author of the post and the course code of the author. The main implementation occurs within `createNewPost()`.

```
private void createNewPost(String content, boolean isGeneral, boolean isCourse) {
    if (user != null) {
        DatabaseReference userRef =
FirebaseDatabase.getInstance().getReference("Users").child(user.getUid());
        userRef.addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                if (snapshot.exists()) {
                    final String firstName =
snapshot.child("firstname").getValue(String.class);
                    // doing the same for last name and course code
                    SimpleDateFormat sdf = new SimpleDateFormat("HH:mm  |  dd MMM yyyy");
                    String currentDateandTime = sdf.format(new Date());

                    DatabaseReference postsRef =
FirebaseDatabase.getInstance().getReference("Posts");
                    String postId = postsRef.push().getKey();


                    Post createdPost = new Post(postId, userId, fullName, course,
content, currentDateandTime, isCourse, isGeneral, 0, new ArrayList<String>());


postsRef.child(postId).setValue(createdPost).addOnCompleteListener(new
OnCompleteListener<Void>() {
                        @Override
                        public void onComplete(@NonNull Task<Void> task) {
                              if (task.isSuccessful()) {//show success message }
                              else {//show error message}
                        }
                    });
                }
            }
          @Override
          public void onCancelled(@NonNull DatabaseError error) {//show error message}});
    }
}
```

Code Snippet 12: Creating New Post and saving to Firebase.

Parking

Ideally the Parking feature would have been a database-centred feature as well, where each parking spot would have their entry pre-stored inside a collection on Firestore, similar to "BookableRooms." Then each individual spot would have a "isTaken()" Boolean that would be set to true when someone was parked in that spot. But in order to implement this in this manner, there would need to be sensors installed at each parking spot in the NCI building which would be something above my implementation capacity and budget.

Therefore, the parking feature in MyNCI consists of a simple switch scenario. It is a much more simplified version than desired, but still serves the crucial purpose of allowing users to check the car park status before arriving to the NCI building and rerouting if the car park is currently full.

Currently at NCI, when the car park is full, the car park entry barrier is closed, and so the only way for people to know that the car park is full is once they get there, which can sometimes be frustrating. With this feature, the only added job to the parking attendant would be to log into the MyNCI application as Admin and switch the parking toggle to "full."

This feature is implemented mostly on the Admin Page, which houses most of the logic for the parking status.

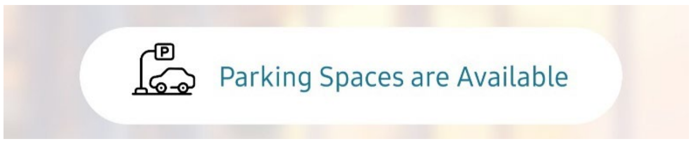
```
// This listener will trigger whenever the isFull field's value changes in the
database.
parkingRef.child("isFull").addValueEventListener(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
        Boolean isFull = dataSnapshot.getValue(Boolean.class);
        if (isFull != null && isFull) {
            parkingSwitch.setChecked(true);
        } else {
            parkingSwitch.setChecked(false);
        }
    }

    @Override
    public void onCancelled(@NonNull DatabaseError databaseError) {
    }
});

// when the switch is toggled on, set it to FULL
parkingSwitch.setOnCheckedChangeListener(new
CompoundButton.OnCheckedChangeListener() {
    @Override
    public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
        parkingRef.child("isFull").setValue(isChecked);
    }
});
```

Code Snippet 13: Parking Listeners on the Admin Page

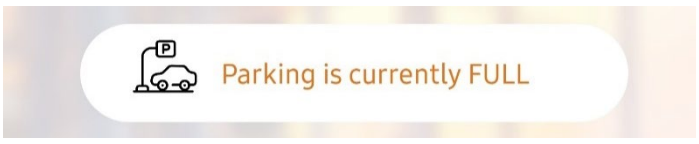
Then on the Home page of the application is where we display the information to the users. A simple text view, that changes in content and colour, according to the parking status is made visible on the Home page of MyNCI. This makes it easily accessible and distinguishable when opening the app with no need for complex navigation, as the likelihood of users checking the parking status while they are in the car is high.

```
private void ParkingToggle() {

    // Read parking state from the database and set the UI accordingly
    parkingRef.child("isFull").addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
            Boolean isFull = dataSnapshot.getValue(Boolean.class);
            if (isFull != null && isFull) {
                parkingStatus.setText("Parking is currently FULL");

parkingStatus.setTextColor(getResources().getColor(R.color.orange_pumpkin));
            } else {
                parkingStatus.setText("Parking Spaces are Available");

parkingStatus.setTextColor(getResources().getColor(R.color.blue_cerulean));
            }
        }

        @Override
        public void onCancelled(@NonNull DatabaseError databaseError) {
            // Log any errors or handle them as needed
            Log.w(TAG, "Failed to read parking status:",
databaseError.toException());
        }
    });
}
```

Code Snippet 14: Retrieving the Parking Status from Firebase and displaying the appropriate message on the Home Page.



MyNCI Figure 6: Parking not Full (Screenshot)

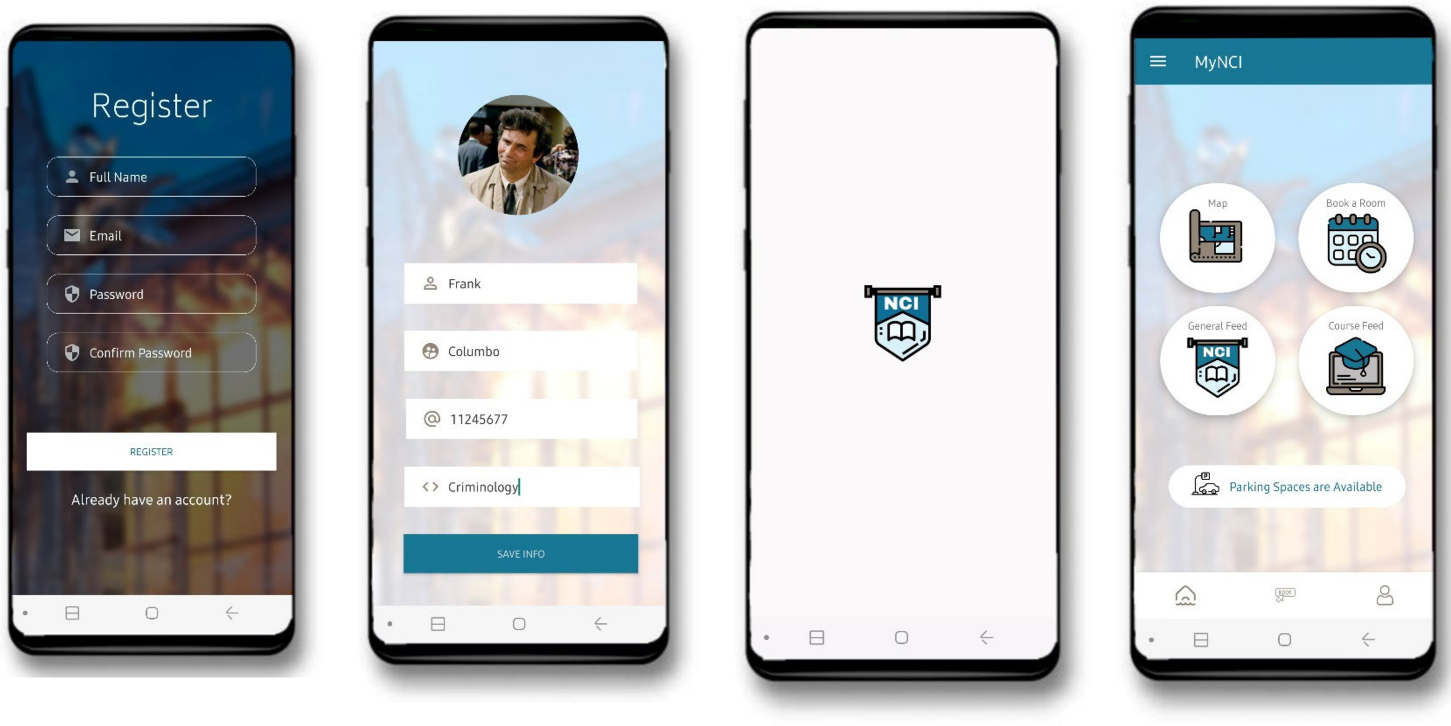


MyNCI Figure 5: Parking is Full (Screenshot)

2.4 Graphical User Interface

This section of the report will demonstrate each of the screens that can be found within the MyNCI application.

For any new user of the application, the flow will go from the Registration page to the Setup Page, then to the Splash Screen, and finally to the Home Page.



Frank

Columbo

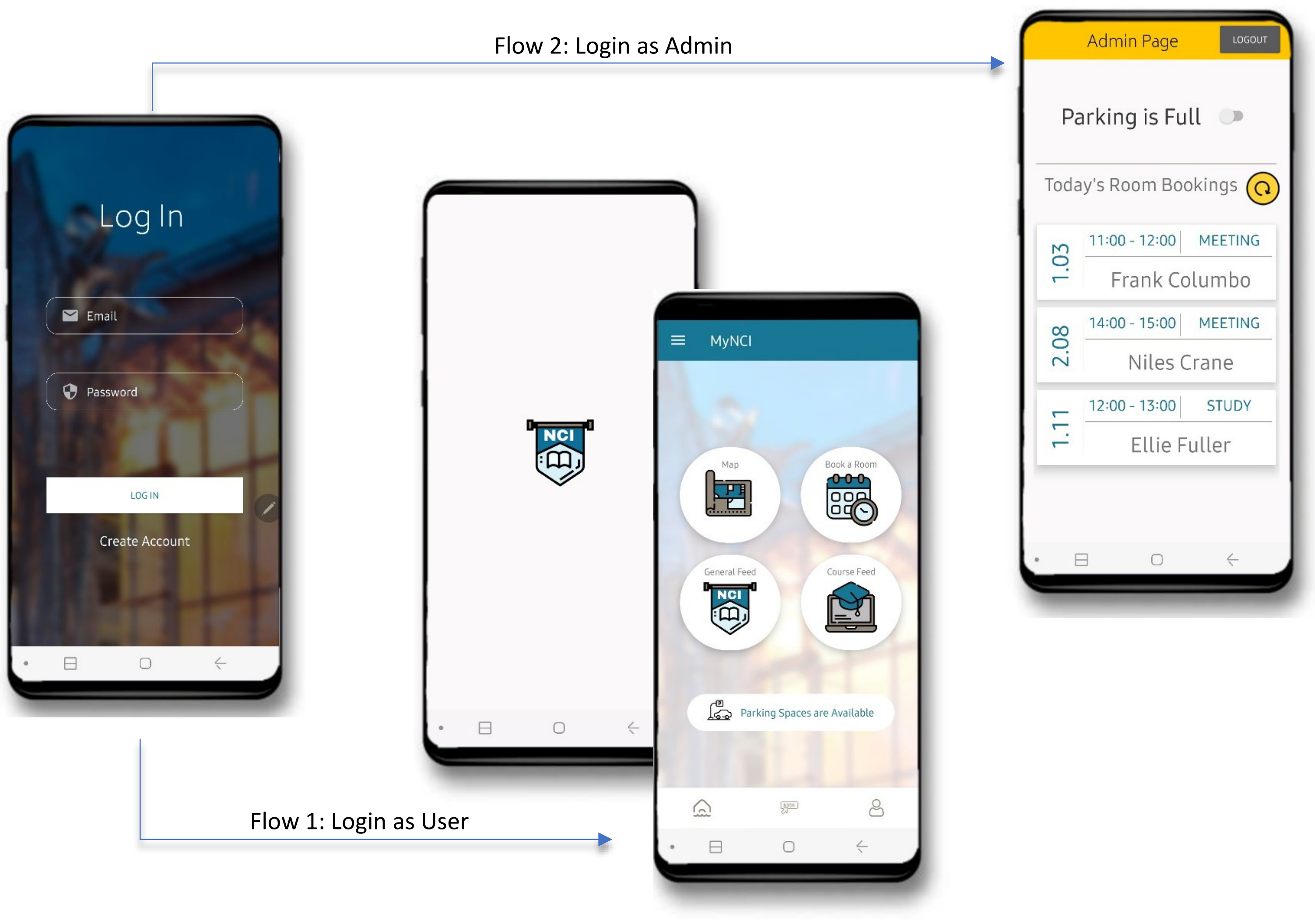
11245677

Criminology

GUI 2: Registration Flow.
From left to right: Register Page, Setup Page, Splash Screen (1 second), Home Page

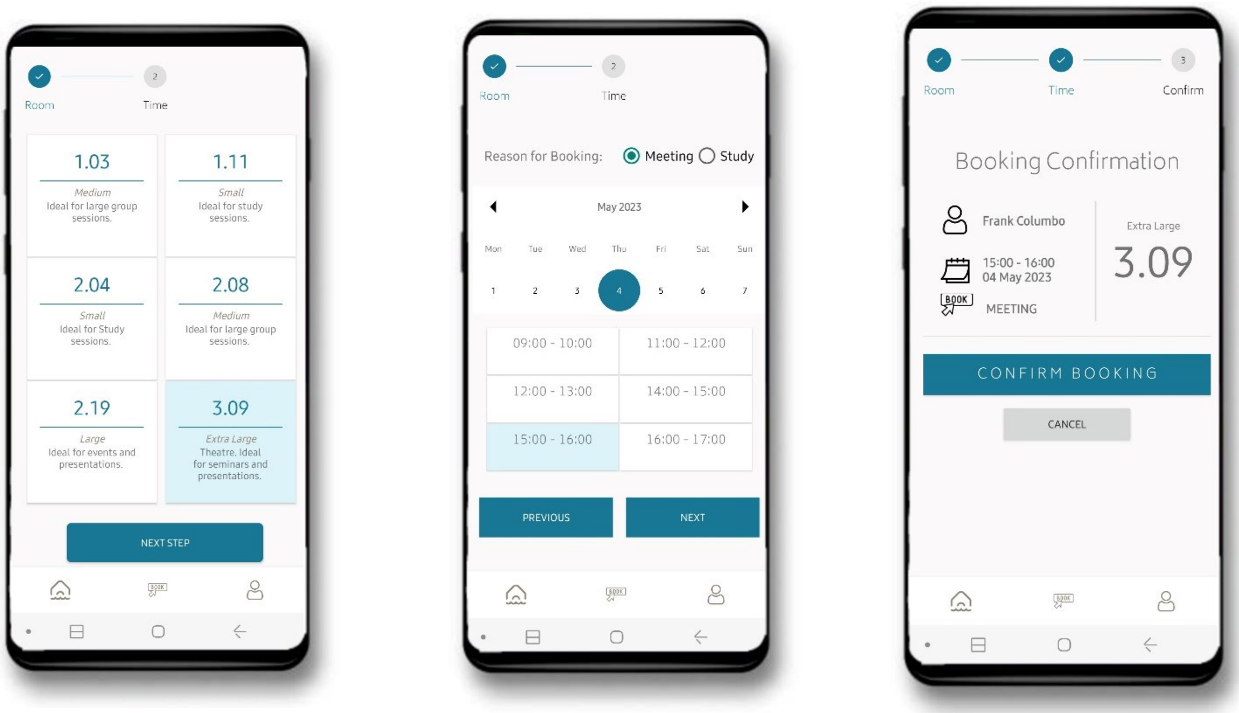
A user that is already registered will go from the Login page to the Splash screen, and then finally to the Home page. Then depending on the user's selection, the GUI for the Maps, Room Booking, General Feed, Course Feed, My Profile, Side Navigation, and Settings page will be available.

Alternatively, an Admin user will see the Login Page, then the Admin page.



GUI 3: Login Flow
Flow 1 Login as User: Login Page, Splash Screen (1 second), Home Page
Flow 2 Login as Admin: Login Page, Admin Page

As mentioned previously, the Room Booking feature occurs over three steps, beginning with the room Selection once the user clicks on the "Booking" icon on the Home page.



1.03 — *Medium* — Ideal for large group sessions.
1.11 — *Small* — Ideal for study sessions.
2.04 — *Small* — Ideal for Study sessions.
2.08 — *Medium* — Ideal for large group sessions.
2.19 — *Large* — Ideal for events and presentations.
3.09 — *Extra Large* — Theatre. Ideal for seminars and presentations.

Reason for Booking: [✓ Meeting] Study

May 2023

Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	2	3	4	5	6	7

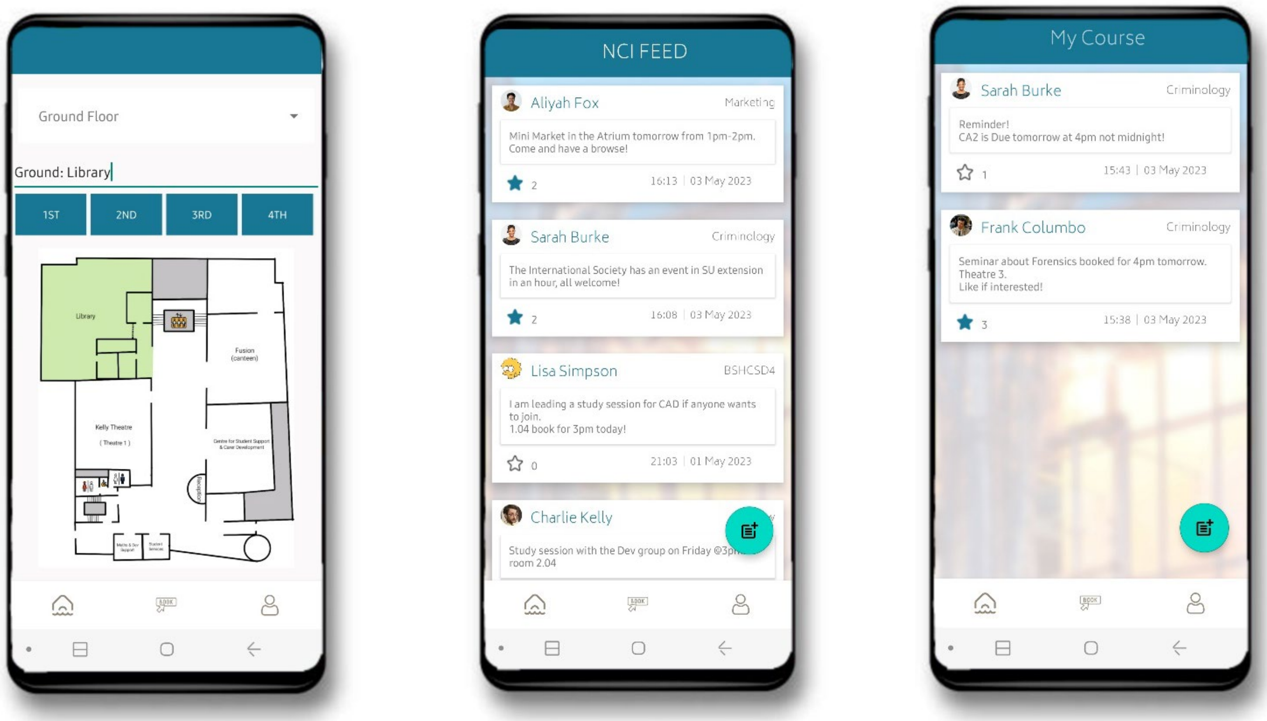
09:00 - 10:00	11:00 - 12:00
12:00 - 13:00	14:00 - 15:00
15:00 - 16:00	16:00 - 17:00

Booking Confirmation

Frank Columbo
15:00 - 16:00
04 May 2023
MEETING
Extra Large
3.09

GUI 4: Booking Steps
From left to right: Step 1 – Choose Room, Step 2 – Booking Details, Step 3 – Confirm Booking

The other main features in the application are the Map, General Feed, and Course Feed pages, all of which can be accessed directly from the Home page by selecting the respective icon.



Aliyah Fox — Marketing
Mini Market in the Atrium tomorrow from 1pm-2pm. Come and have a browse!
2 — 16:13 | 03 May 2023

Sarah Burke — Criminology
The International Society has an event in SU extension in an hour, all welcome!
2 — 16:08 | 03 May 2023

Lisa Simpson — BSHCSD4
I am leading a study session for CAD if anyone wants to join.
1.04 book for 3pm today!
0 — 21:03 | 01 May 2023

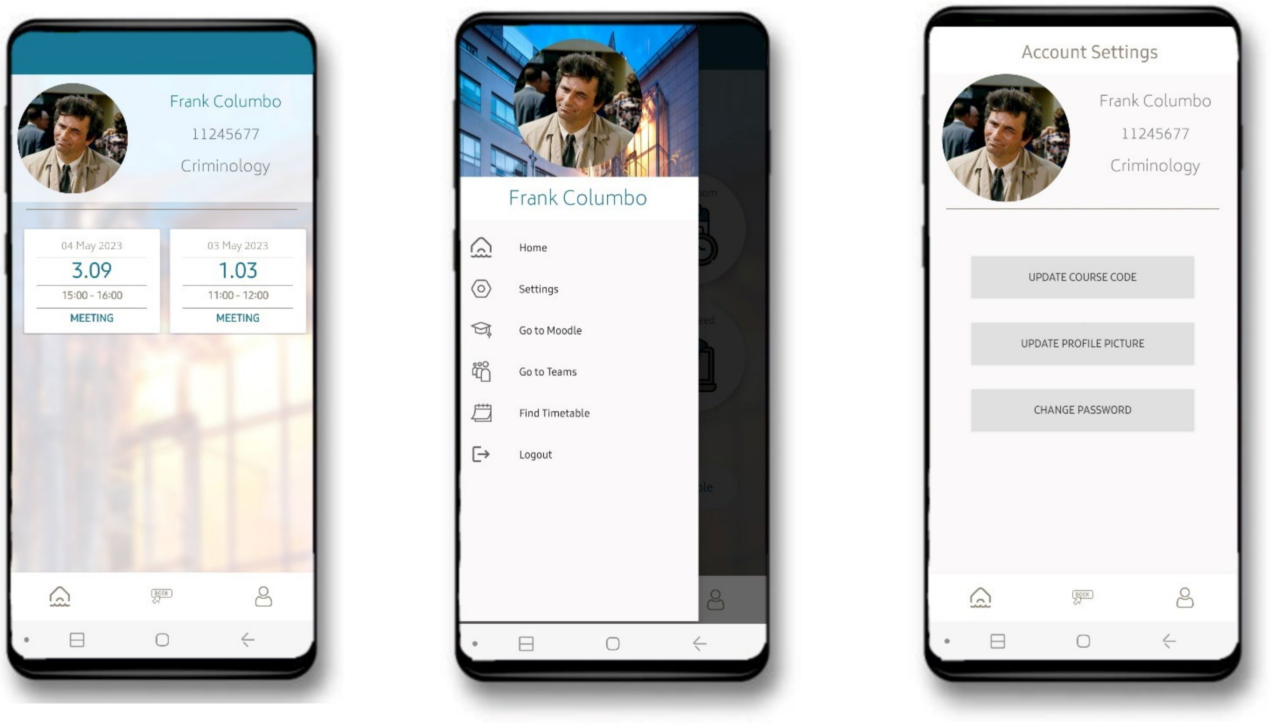
Charlie Kelly
Study session with the Dev group on Friday @3p… room 2.04

Sarah Burke — Criminology
Reminder!
CA2 is Due tomorrow at 4pm not midnight!
1 — 15:43 | 03 May 2023

Frank Columbo — Criminology
Seminar about Forensics booked for 4pm tomorrow.
Theatre 3.
Like if interested!
3 — 15:38 | 03 May 2023

GUI 5: From left to Right: Maps page, General Feed page, Course Feed

Additionally, the Profile page can be accessed from anywhere in the app as it is located in the bottom navigation bar, alongside Home, and Room Booking. The profile page is where the user's bookings are displayed. Lastly, the side navigation menu houses the Settings page, Logout function, and external links to: Moodle, Teams and the student Timetable.



GUI 6: From left to right: Profile Page, Side Navigation Menu, Settings Page

2.5 Testing & Evaluation

Unit and Integration Testing

I have carried out several tests for the MyNCI application and those include the unit and Integration Testing.

I have performed unit tests for the models such as the `User`, `BookingRoomInformation`, and `AdminInformation` models. A unit test is a type of software testing that focuses on verifying the individual components, or units, of a software system. The purpose of unit testing is to isolate and test each unit of code independently to ensure that it behaves as expected. [5]

For example, let's take the `BookingRoomInformationTest.`

```java
public class RoomBookingInformationTest {
    private RoomBookingInformation bookingInfo;

    @Before
    public void setUp() {
        bookingInfo = new RoomBookingInformation("Meeting", "2023-05-10", "123456",
"2:00 PM - 4:00 PM", "John Smith", "Small", "Room A", "1", "7890");
    }

    @Test
    public void testGetBookingReason() {
        assertEquals("Meeting", bookingInfo.getBookingReason());
    }

    @Test
    public void testSetBookingReason() {
        bookingInfo.setBookingReason("Study");
        assertEquals("Study", bookingInfo.getBookingReason());
    }

    @Test
    public void testGetBookingDate() {
        assertEquals("2023-05-10", bookingInfo.getBookingDate());
    }

    @Test
    public void testSetBookingDate() {
        bookingInfo.setBookingDate("2023-05-11");
        assertEquals("2023-05-11", bookingInfo.getBookingDate());
    }

    @Test
    public void testGetBookingID() {
        assertEquals("123456", bookingInfo.getBookingID());
    }

    @Test
    public void testSetBookingID() {
        bookingInfo.setBookingID("654321");
        assertEquals("654321", bookingInfo.getBookingID());
    }     // more similar tests for the rest of the variables…
}
```

Code Snippet 15: Unit Test - BookingRoomInformationTest

The `RoomBookingInformationTest` is a Junit class that is responsible for testing the functionality of the RoomBookingInformation class in the MyNCI app. The RoomBookingInformation class represents the booking information for a room booking, and the tests are making sure that the getters and setters of this class are working properly.

In the setup method, an instance of the RoomBookingInformation class is created with some dummy data to be used for the testing. Each test method focuses on testing a specific getter and setter method of the RoomBookingInformation class.

For example, the testGetBookingReason method checks that the getBookingReason method returns the expected booking reason, and the testSetBookingReason method checks if the setBookingReason method correctly updates the booking reason. The same tests are run for the other variables like booking date, bookingID, time, user name, room size, room number, timeslot, and user ID.

The other unit tests follow the same kind of logic, and those tests are there to make sure that the models are functioning as they should and allowing users to retrieve and update information accurately. The test methods all use assertions provided by the JUnit testing framework to compare the expected values with the actual values obtained from the Model object. Unit Tests provide confidence in the model behaviour and contribute to the overall quality of the MyNCI App.

As for integration testing, an example of it used in my application is in the NavigationBarTest. Integration testing is a software testing method that checks the interaction and integration between different components, modules, or systems of an application. These tests are the second level of software testing process after unit tests. Integration tests make sure that the integrated features work together seamlessly, communicate well with each other, and produce the expected results. [6]

```java
@RunWith(MockitoJUnitRunner.class)
public class NavigationBarTest {

    @Rule
    public ActivityScenarioRule<HomeActivity> activityScenarioRule =
            new ActivityScenarioRule<>(HomeActivity.class);

    @Test
    public void testNavigationBar() {
        onView(withId(R.id.navigation_view)).check(matches(isDisplayed()));

        // Click the profile button
        onView(withId(R.id.profile_bottomnav)).perform(click());

        // Check that the ProfileActivity is loaded
        onView(withId(R.id.profile_header)).check(matches(isDisplayed()));
    }
}
```

Code Snippet 16: Integration Test - NavigationBarTest

The `NavigationBarTest` is another JUnit test class that focuses on testing the navigation bar functionality of the HomeActivity. Because the navigation bar is visible on every page of the app, and is responsible for the majority of the app's flow and navigation, it is very important to test if this feature is working as expected.

Here, the test method testNavigationBar() checks whether the navigation bar contents are displayed correctly and if the navigation to the profile screen works as expected when the icon is clicked. It does so by checking the ID of the navigation view and if it is visible on the screen using the onView and onDisplayed methods. Then it performs the action of clicking the profile icon to trigger the navigation to the Profile page. Once that is done, it is assumed that the app has navigated to the Profile page so we run a check by searching for an element that we know should be present on the Profile page, which in this case is the profile_header. If the profile header is displayed, it indicates that the navigation to the profile screen was successful.

SMART Testing

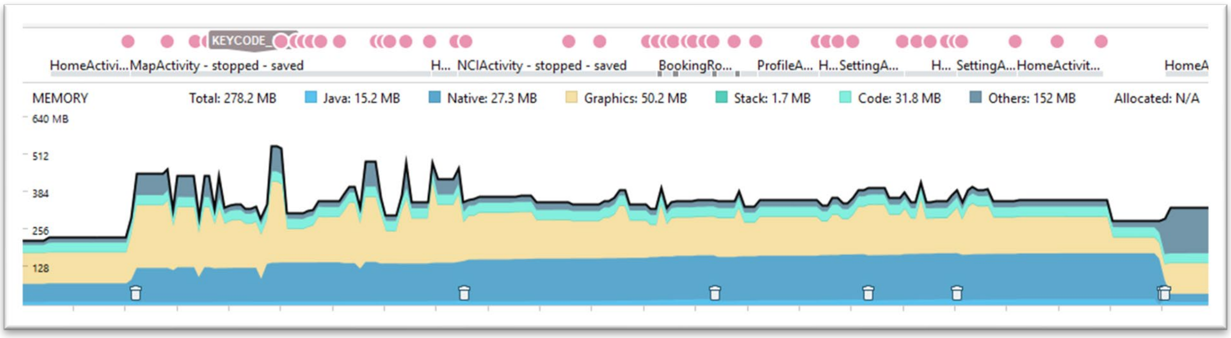
Performing SMART testing on the MyNCI application meant that I was verifying that the app met the following requirements of being Specific, Measurable, Achievable, Relevant and Time-bound. SMART testing is an effective approach to ensure the quality and reliability of an application. [7]

Specific:

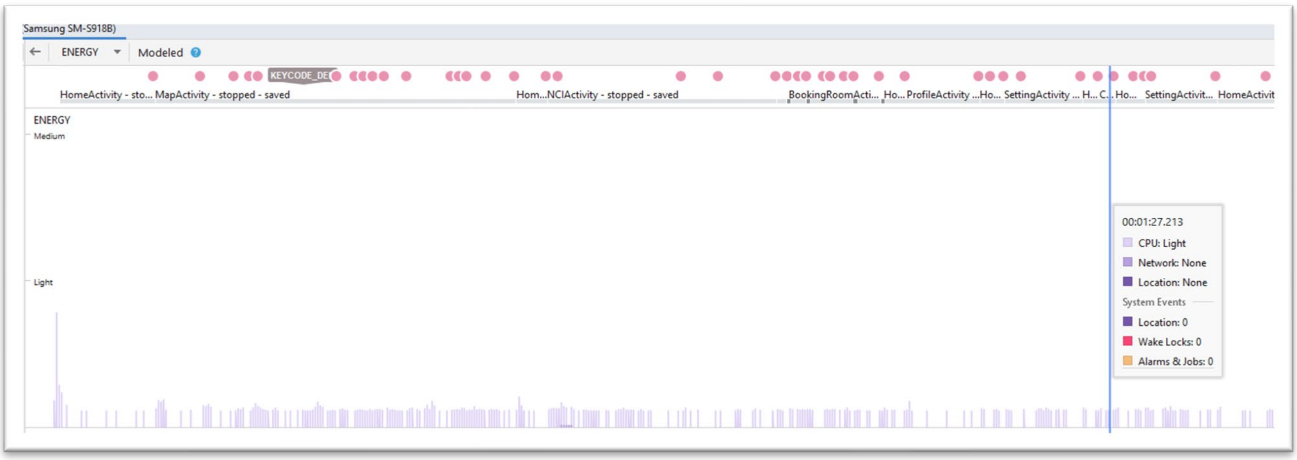
This included physical testing of the features of the application on my Samsung phone. I tested every feature (Booking Room, Search for a Room, View Filtered feeds, Create posts for specific feeds, update the user's course code etc.) The goal was to make sure that every feature was meeting the objectives and scope of their respective use cases.

Measurable:

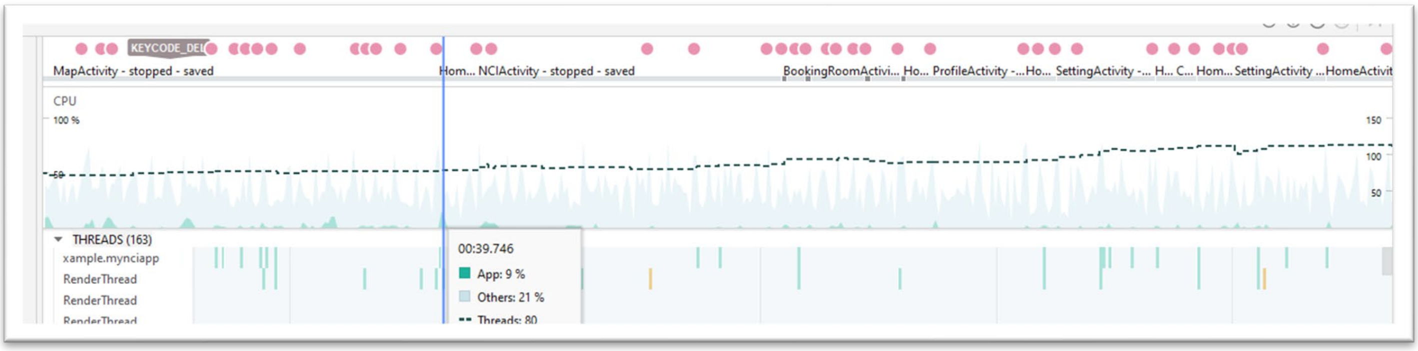
I ran some performance tests using Android Studio's Profiler tool to gather the performance of the MyNCI application. The tests included measuring the Memory, Energy and CPU usage. The tests were run on my mobile phone, a Samsung S23 Ultra.



MyNCI Figure 7: Performance Test - Memory


Samsung SM-S918B)
ENERGY
Modeled
KEYCODE_DE
HomeActivity - sto... MapActivity - stopped - saved
Hom...NCIActivity - stopped - saved
BookingRoomActi... Ho... ProfileActivity ...Ho... SettingActivity ... H... C... Ho... SettingActivit... HomeActivit
ENERGY
Medium
Light
00:01:27.213
CPU: Light
Network: None
Location: None
System Events
Location: 0
Wake Locks: 0
Alarms & Jobs: 0

MyNCI Figure 9: Performance Test - Energy


KEYCODE_DE
MapActivity - stopped - saved
Hom... NCIActivity - stopped - saved
BookingRoomActivi... Ho... ProfileActivity -...Ho... SettingActivity -... H... C... Hom... SettingActivity ...HomeActivit
CPU
100 %
50
150
100
50
THREADS (163)
xample.mynciapp
RenderThread
RenderThread
RenderThread
00:39.746
App: 9 %
Others: 21 %
Threads: 80

MyNCI Figure 8: Performance Test - CPU

We can see from figures 7, 8 and 9 that the performance of the app is having a very small impact on the device's RAM. The energy test in figure 8 is constantly coming up as "light" usage, and on the CPU test graph, the MyNCI app (which is represented by the darker green isobars) is taking up a very small percentage of the overall CPU usage, averaging at 9%.

<u>Achievable:</u>
Having the mobile application to directly test the app made the testing much more efficient and attainable. Each feature was tested as they were being built during the sprints. This made it easier to see the immediate result of the work and permit to fix errors as they arose.

<u>Relevant:</u>
The main focus of the tests was to ensure that the app was fulfilling its purpose and meeting the Software Specification Requirements.

<u>Time-bound:</u>
As the manual test of the app were occurring throughout the entire lifecycle of the project, it made it easier to complete features fully, before moving onto the next one with confidence.

I also set milestones and deadlines for each phase of the app development and testing, in the hope to allow for sufficient time for bug fixes, retesting and refinements.

Usability Testing

Evaluate MyNCI's user interface for accessibility and ease of use. The requirements are:

- Ask someone to use the application as a first-time user.
- Test the app on different android devices.
- Record the feedback.

Test Case:

1. Register an account.
2. Search for a room in the college.
3. Book a room + cancel the booking.
4. Create two posts. (One for General, one for Course Feed).
5. Update your profile picture.
6. Comments.

Samsung Galaxy S21+

1	Registration was standard
2	Quick and easy to find a room
3	Nice step by step process for booking. Cancellation was simple.
4	Posts created easily and on the right feed. Like function, but no comments?
5	Simple and seamless.
6	Great app! Very easy to use.

Google Pixel 6

1	Easy registration, set up step is a nice touch.
2	Love this feature! Worked really well!
3	Booking a room was super-fast!
4	Adding a post was simple, page is intuitively laid out.
5	Standard, no issue.
6	This app has great potential, love the parking status and the quick links to Moodle!

Sony Xperia 10 iv

1	As expected, good validations.
2	Works well, no ground floor button, (only 1-4 floors) had to search for it.
3	Nice UI, handy feature.
4	Slight glitch when liking posts, profile images seem to refresh on every "like"
5	As expected.
6	Good app idea. Overall, very easy to use.

Overall, the feedback received, both from end users and the performance tests I performed, have been positive and very helpful feedback. It shows that the application has met the requirements set out and delivers the expected product.

3. Conclusions

MyNCI has been designed in a way to provide an integrated platform for students’ productive and logistical needs, all with the aim to enhance the student experience at NCI. The app is designed in an intuitive and simplistic manner making it all the more attractive to users.

Using Firebase as the main cloud database for this project provides a robust and flexible backend service, which allows for repetitive tasks of the application to be handled easily and in an effective manner. Additionally, the modular architecture of the project allows each feature to work independently of each other but as a compliment to the app, and also providing scalability for future development.

The main advantage of this project is that it streamlines and simplifies various aspects of campus life for students, making it easier for them to navigate and manage their academic needs. By offering features such as room booking, interactive maps of the college, news feeds tailored to the student's courses, and quick link access to Moodle, Teams, and Timetables, MyNCI can become a one-stop-shop app for anyone involved in the National College of Ireland.

However, there are also limitations to the project. The Room Search functionality had to be implemented with images of each floor plan for every single room in the building, making the process more time-consuming and less efficient. The maps were also hand drawn and, as a result, not the most aesthetically pleasing. The Parking feature, while providing an important service to users, is a simplified version of what could have been a more advanced database-centred feature.

Despite these limitations, the MyNCI application is a valuable tool for anybody affiliated with the National College of Ireland, providing easy access to important information and streamlining various aspects of campus life. It is a project that I stand proudly behind, and I look forward to further improve upon it in the future.

4. Further Development & Research

With additional time and resources, there are certain aspects that could be improved upon. The main one being that the application should be also implemented for IOS platforms, as it makes up a significant percentage of the user’s demographic and only then will MyNCI truly be considered the go-to app for NCI goers.

Another area that could see some helpful improvements would be the Maps area. With the proper collaboration with the logistic department, cleaner and more accurate plans of the building can be procured, including the Research building which was not made available to me. And with a more accurate floor plan, and additional time to work on it, then mapping room points over each floor plans according to the room name would be a more achievable task.

Again, another improvement that could only be achieved through external collaboration would be the addition of parking sensors in the building. This would allow for a much more accurate reading of the parking status right down the exact number of available slots. This addition would also reduce the administrative task as once all the parking spots are full, then the trigger could be automatically set, without the need to switch the toggle.

Since MyNCI is made in a modular fashion, the door is open to new innovative features that can easily be added in the future. For example, some features that I had in mind that wouldn't have been possible due to the time constraint were to add specific feeds for all the different societies that are present at NCI, and users could subscribe to their desired ones.

And my favourite idea which proved too logistically difficult to implement at the time would be to add a Biometric feature to MyNCI. This feature would allow users to scan their student cards on the application and save its data, and this would allow them to use their phones in lieu of their student card to access rooms or log attendance, much like with Google wallet.

5. References

[1] "Android Studio," [Online]. Available: https://developer.android.com/.

[2] "Google's Firebase," [Online]. Available: https://firebase.google.com/.

[3] GitHub, "Wikipedia," [Online]. Available: https://en.wikipedia.org/wiki/GitHub.

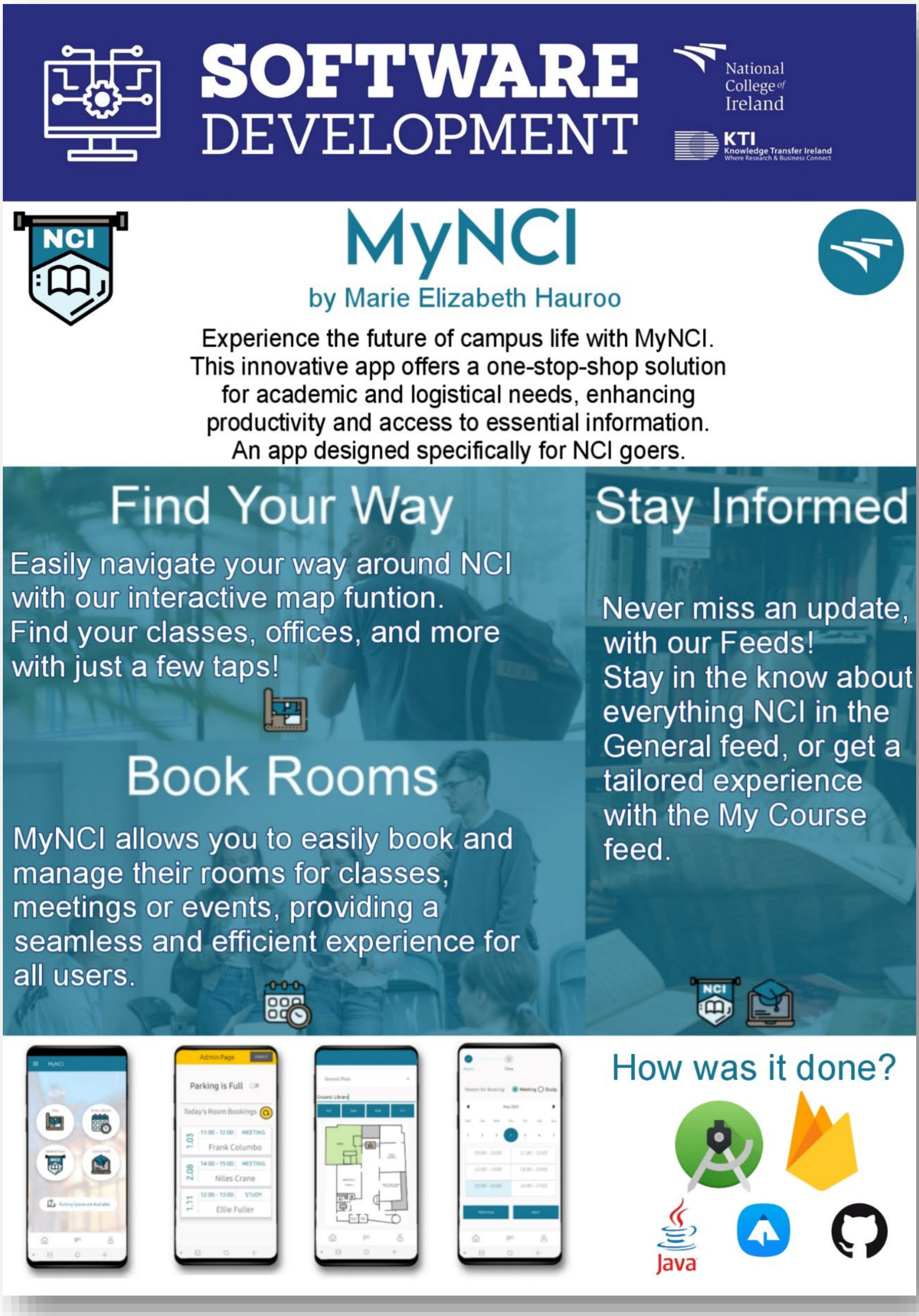
[4] R. MISHRA, "Geeks for Geeks," [Online]. Available: https://www.geeksforgeeks.org/mvc-model-view-controller-architecture-pattern-in-android-with-example/.

[5] S. Bear, "What is Unit Testing?," [Online]. Available: https://smartbear.com/learn/automated-testing/what-is-unit-testing/.

[6] J. T. Point, "Integration Testing," [Online]. Available: https://www.javatpoint.com/integration-testing.

[7] Wikipedia, "SMART criteria," [Online]. Available: https://en.wikipedia.org/wiki/SMART_criteria.

6. Poster



SOFTWARE DEVELOPMENT

National College of Ireland

KTI Knowledge Transfer Ireland Where Research & Business Connect

MyNCI

by Marie Elizabeth Hauroo

Experience the future of campus life with MyNCI. This innovative app offers a one-stop-shop solution for academic and logistical needs, enhancing productivity and access to essential information. An app designed specifically for NCI goers.

Find Your Way

Easily navigate your way around NCI with our interactive map funtion. Find your classes, offices, and more with just a few taps!

Book Rooms

MyNCI allows you to easily book and manage their rooms for classes, meetings or events, providing a seamless and efficient experience for all users.

Stay Informed

Never miss an update, with our Feeds! Stay in the know about everything NCI in the General feed, or get a tailored experience with the My Course feed.

How was it done?

7. Appendices

7.1 Project Proposal

6.1.1 Objectives

For my final year project, I am proposing to make a mobile application for National College of Ireland. This application will be accessible to both NCI students and staff, and be fronted with a login page, that will only allow initial registration for people with NCI credentials. The main goal of this project is to give users access to an interactive map of the college, and to be able to book study rooms in the college. The app will also provide immediate access to real-time updates via the feeds, all inside one application that is only a touch away.

MyNCI will be a blend of a social platform and a productivity application, as it will be made up of several "cards" each holding a different function. Each of the cards will be displayed on the Home page for easy access. Some of the proposed productivity features of the MyNCI application are to have a "Maps" card that will display the floor plan of the college for each floor level, making it easier to navigate through the building. This feature will also allow the user to search for a specific room, and be directed to the right floor and location of the room requested. The other main tile on the home page is the "Book a Room" card where users can choose between two rooms to book a time slot.

The application will also have feed-like tiles such as "General NCI", "My Course". This is where MyNCI transitions into a social platform because these feeds will each allow the user to create posts, likes and comments within them. The feeds are going to serve as the place to go for the latest updates in the selected forum. For example, "MyCourse" will be filtered for each student so that they see only posts that are relevant to their course. This will be done using a hashtag to categorize the posts. Then students can post information that is related to their own course, this could be an announcement from the student rep, or someone sharing a helpful article etc.

The application will also have a profile page for each user displaying their names, their NCI email address, their course code (if they are a student), and some optional fields such as links to their LinkedIn or GitHub accounts.

6.1.2 Background

When choosing the idea for my final project, I knew I wanted to work on something that I knew I would use myself. This was mainly because I feel it would be much more effective as I would be able to come up with the product requirements in a much more realistic way, since I would essentially be one of the users at the end of production.

As a final year student in 2022, I have only really spent one full semester on campus and every other semester thereafter has been delivered online due to the pandemic, so even though I am in final year, several aspects of the college are still somewhat new to me. Coming back into the college this semester, I was very aware that this would be daunting to everyone, particularly those who have had their entire college experience online prior to this semester. To be able to have a full floor plan readily accessible to students will make navigating the college a much less daunting task.

MyNCI will set out to be a nucleus point of information for all things NCI. Students are often in need of using a room either to carry out a group project session or student meeting etc. Having the “Book a Room” feature eliminates the middleman of the transaction, and as a result reduces admin work for reception.

The feeds feature will be where most of the up-to-date information will be available, be it in the “General NCI” feed - where people can post updates on upcoming fire-drills, pop-up stands in the Atrium etc. Or in the “MyCourse” feed, where anyone in the user’s cohort can post helpful links, deadline reminders, class rep discussions etc. The idea for the feeds came from my experience at using “Slack” during my internship. It was great to have different channels that were categorized so that you knew exactly where to ask for specific information.

Having all this information inside a single application makes it easier for everyone, but especially for those who are more reserved in nature that might have missed some information because they are too shy to ask. This application will hopefully make navigating the college much easier for everyone.

6.1.3 State of the Art

The concept of having a mobile application for a college establishment isn’t necessarily a new one, several colleges and universities have implemented their own, but according to my research, National College of Ireland does not at present have a mobile application for their students and staff beyond their Moodle page and Teams, there doesn’t seem to be an application that offers the features that I am proposing.

Trinity, UCD, and DCU are part of the few Irish universities that I have found to have their own mobile application on the mobile app market. These apps vary in what they offer, and as expected, I would need the valid credentials in order to delve deeper into their features.

Also, while some of those applications have a “plan” of their college/campus, I’m proposing to make that feature more complex, where the user should be able to type in the room number/name they are looking for and the app will apply that search and highlight specifically where on the given floor that room is situated.

Theoretically, my project idea doesn’t deviate too far from what is already on the market, but rather that it sets out to reach an untouched target audience, the members of the National College of Ireland.

6.1.4 Technical Approach

This project is fairly big on scale for what I have set out to achieve, therefore it is very important that I take the necessary measures to plan it properly. I believe that the best approach for undertaking this project is to implement an Agile methodology, specifically scrum.

When I was doing my internship last year at Workday, I was on a team that used a scrum-based approach. I am planning on using my experience from the internship to track my Project using Trello and the scrum methodology.

The Scrum framework is made up of sprints of approximately one or two weeks where the focus is to complete a segment (aka Epic) of the final product. During a sprint, each "feature" (aka story) goes through the following stages: refinement, backlog, development, testing, and review, before being added to the "done" lane.

To identify the requirements necessary for the project, I will refer to my initial inspiration for undertaking this project, the main real-world problems I wanted to solve when I came up with the idea. From this source alone, there are several features that I have proposed to implement. And as I am going to be one of end users of this application, I feel that it will be slightly easier to put myself in the client's shoes.

However, I am also planning on researching some other college apps that offer a similar services to see what popular features are succeeding in the current market. This will help me to figure out what the most important features (functional and non-functional requirements) of my project needs to be. This will also help me to get an idea of the general flow and layouts that these apps usually consist of.

When it comes to breaking down the requirements into project tasks, my approach is to dissect the requirements into blocks of work that, when completed, can be tested properly. For example, the first project task that I would do would be designing the separate pages of the application in Android studio, with no functionality attached. Then I am planning on tackling each page of the app, one function at a time. This could be the login feature, or profile page etc. Each page has functions attached to them, and those I will code in turn and testing them as I go.

Some important milestones that I have set for this app are as follows:

- UI (design and code all the pages – no functionality)
- Register/Sign-In (Firebase integration)
- Navigation of the "cards" to the correct pages
- NCI Maps Tile (with search capabilities)
- Booking a Room
- Create a post (in the filtered feeds)
- User Profile page

6.1.5 Technical Skills

The MyNCI application will be a native mobile application for Android, and so for this application, I will be using Android Studio to deliver the final product. A native application is a platform specific app that will run on a specific operating system, in this case Android OS. While building a native application is limiting, these types of applications commonly offer better user-experience and are better in performance. The implementation language will be Java and Kotlin.

The MyNCI app will be a hybrid of static and dynamic databases. The static will be all of the core information that are hard coded onto the app, but mostly the dynamic storage is to accommodate the data provided by the user. This will take many forms for example, user profiles, posts, and comments.

This app will require various libraries during implementation, at the moment I am planning on using Firebase for the database handling, and because I will be using Android Studio, I am also anticipating using libraries that will allow for authentication, image cropping (Picasso) to name a few that I know of so far. Over the course of implementation, I believe this would become a more intensive list as I learn more about the technology.

The feeds section of this app will have to be sorted so that the newest posts appear at the top of the feed, and the oldest at the bottom. This will require some sort of stacking algorithm. I also plan on filtering out the posts so that they only appear on the correct feeds, for example a post that is intended for “MyCourse” should not appear on the “General NCI” feed. In order to accomplish this, each post will need to be given a specific key that I can filter out by (eg: #BSCHSD4).

6.1.6 Special Resources Required

A. Trello Account
B. Firebase Account
C. Wireframe creating tools
D. Image editing Software
E. Android Studio IDE
F. GitHub

6.1.7. Project Plan

As mentioned previously, I have chosen to use the scrum methodology for my product management. I am using a Trello board to keep track of my Epics and stories, but I have also produced a provisional Gantt chart as a visual timeline. As the project continues to take shape, I expect to make several additions to the Gantt chart.

During the process of creating this product, I have categorized the workload into 5 main categories:

1. Project Conception & Initiation
2. Project Definition & Planning
3. Design & Development
4. Implementation & Testing
5. Submissions

Project Conception & Initiation

This starting phase includes the conception of the project idea and refinement of the project pitch. It is expected to be mostly completed by the first half of November.

Project Definition & Planning

During this phase I plan on completing tasks such as the research into technologies that will best suit my project, creating the wireframes for my application and the general design/theme I want the app to have. I am also including the tracking of my documentation in this category. Most of the tasks in this category should be completed in October and November.

Design & Development

This category is one that I expect to be spending the most task on, since it is the crux of the application production. This category will be made up of many tasks and subtasks as I work my way through production.

I have started to organize my tasks in my Gantt chart with an estimation of how long I am expecting to spend on each task. For example, the first task is to build out all the pages of the application (with no functional features). This should not take any longer than two weeks to complete. I have this listed on my Gantt chart to be due by the 2nd week of November.

The Development phase of this project will take up the majority of my timeline, I've attached it below to outline how I expect my implementation to go.

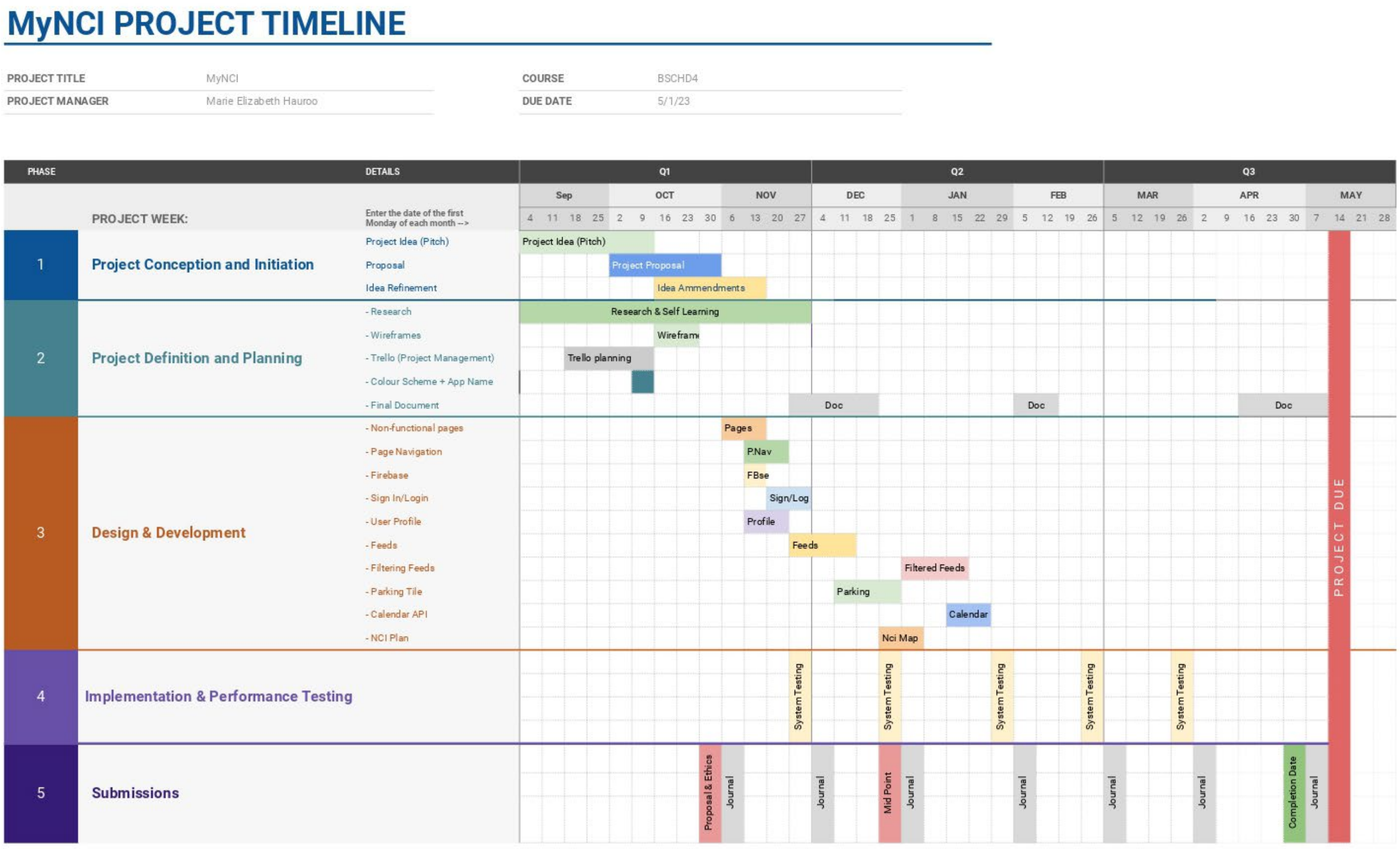
Implementation & Performance Testing

I am going to be working with a scrum methodology, meaning that each sprint (dev task) will go through the entire Product Life Cycle (Brainstorm, refinement, development, test, launch). But even so, I have added some system testing dates onto my timeline so that I set side time at the end of every month until submission to do some system testing to see how the application is performing at that point in time.

Submission

This gets a category of its own on my Gantt chart simply because it is visually easier to see when a certain component is due for submission. There are town main submissions during this module, the Mid-Point presentation and the Final Submission. But in between I also have to complete my monthly reflections, so I have also added them to the chart.

I have also set myself a development due date that falls two weeks before the entire project is due. For my own productivity, I would like to have the application completely finished by the last week in April, so that I don't run the chance of a last-minute code breakdown and if I do, then the two weeks will act as a safety net. The last two weeks will then be dedicated to getting all the documentation ready for submission, including the video presentation.



6.1.7. Testing

In order to do some testing of the application, I will be using the tools that are provided by Android Studio. My testing phase will be carried out at the end of each “story” from my project plan. This will allow me to catch errors and bugs much more quickly and efficiently than if I were to do my testing further apart.

The plan is to carry out tests with each new feature that gets completed, making sure that it is fully working as designed before moving on to the next piece of work.

The helpful thing about using Android Studio to make this application is that I can use Android Emulators with my Android device to get a hands-on experience of how the app is developing. This gives a very real look into how the application looks, feels, and flows.

As well as testing out the features on the application, database tests will also have to be performed. Once the database is setup, I will test that when data is being entered onto the application that the changes are being persisted into the database. The information that has already been given should always be available on the app unless it has been deleted or removed by the user.

My plan is to evaluate how the various components of the application interact together at each step of the implementation process, but still carry out monthly system tests to regularly test the application as a whole.

7.2 Reflective Journals

October

Student Name:	Marie Elizabeth Hauroo	**Course:**	BSHCSD4
Student Number:	19140410	**Supervisor:**	Lisa Murphy

Achievements:

This month was all about getting the project idea primed and ready for pitching. It took me quiet a while to come up with my project idea, I was bouncing between various options and only came up with an idea that I was happy with three days before the pitch video was due.

This didn't leave me with much time to flesh out all the details that I wanted to put across in my pitch video, which was due in the second week of the month. But once submitted, I began to delve deeper in what I wanted my project to do. As we had to wait for the pitches to be approved, I used the time to start making my planning board on Trello. To the best of my estimation, I started to plan out a loose timeline to follow.

Part of that planning included the start of the visual design process. I selected my colour palette for the application as well as the general look and feel I would like the app to have. I have selected my icons and digitally coloured them to match the colour palette.

I signed myself up to a Udemy course on using Android Studio and spent a significant amount of time this month getting some practice using the IDE.

Reflection:

As I look back on this past month, I think that I could have been more prepared for the project idea. It would have been ideal if I had been able to come up with my idea sooner so that I could have been better prepared for my pitch.

While I had started on the visual aesthetic that I wanted the app to have, I haven't yet done a full mock-up of wireframes for the application, I was waiting for the pitch approval, when in hindsight I could have gotten started on those.

Next Actions:

At the time of writing this journal, I have just received my feedback on my pitch, and have now been appointed my supervisor. My pitch was approved with amendments, so the beginning of next month will start with a meeting with my supervisor to see what amendments should be made to my proposal.

November

Student Name:	Marie Elizabeth Hauroo	**Course:**	BSHCSD4
Student Number:	19140410	**Supervisor:**	Lisa Murphy

Achievements:

I have had to go back to the drawing board and refined my idea. It hasn't been a great month for progress on the project, but I managed to source out some materials for the mapping component of the project.

I have also run into issues with the lack of current documentation for android studio and firebase which has greatly delayed the progress.

Time has also been a major constraint this month as I was struggling to balance all of the other assignments that I had due right up until the end of November.

Reflection:

This month has been a difficult time to navigate through my project. Because our supervisors were only assigned at the very end of October, and my project idea was approved but with amendments. The initial meeting with my supervisor brought up many changes in the direction I initially wanted my project to go.

Next Actions:

Researching is still a major part of my project planning at the moment. And getting a prototype ready for the midpoint presentation is top priority. I plan on having the login feature complete, as well as navigation within the app without functionality. And lastly, I hope to have the main components of the "map of NCI" functioning.

There is a significant amount of work to be done in the next three weeks.

December

Student Name:	Marie Elizabeth Hauroo	**Course:**	BSHCSD4
Student Number:	19140410	**Supervisor:**	Lisa Murphy

Achievements:

December has been a busy month. I spent a lot of time on the documentation and actual implementation of my project for the mid-point submission. I had to flesh out and finalise my functional requirements as a result. I manage to complete a good portion of the documentation which also was a good guide for what direction I was going to take the project. I have decided to stick with Android studio as my main technology and began the coding process of the application.

Reflection:

I managed to get some basics of my application started, so I have connected my app to a database (Firebase) and have a functioning login/logout and register function. I also have the main pages set up but with minimal functionality. I have some helpful navigation, including a side bar menu. Navigating between the pages is functioning, but the more complex part of the application still needs to be implemented.

I met my deadline for the midpoint submission with a solid portion of my documentation completed, this documentation will of course be reviewed and updated through the coming months, but at least there is a good starting ground.

The next big challenges will be to do the booking a room function and the interactive floor maps of the college.

Next Actions:

I still need to do some more research on how exactly I will implement the booking and map functionalities.

I will have to keep the documentation up to date as well and watch out when/if some of the requirements are changed slightly or if I add something different.

January

Student Name:	Marie Elizabeth Hauroo	**Course:**	BSHCSD4
Student Number:	19140410	**Supervisor:**	Lisa Murphy

Achievements:

This month since we had a pretty busy start of the year and TABAs keeping me busy, I have made less progress than the previous few months as I was trying to complete other pieces of work as well as taking the necessary time off. I have mostly spent my project time watching tutorial videos on how to handle a booking option using android in the hope of finding a method that will suit my project.

Reflection:

The physical progress has been minimal this month, but I have been brainstorming some of the best ways to accomplish my main features and researching for resources to help me do so. I have also done some minor coding changes to fix some of the design and UI look of my application to make it look slightly more polished.

Next Actions:

This coming month I expect to make more substantial progress and hit at least one of my milestones, that is to get the booking a room function implemented. I am also going to be looking deeper into how to implement the interactive map.

February

Student Name:	Marie Elizabeth Hauroo	**Course:**	BSHCSD4
Student Number:	19140410	**Supervisor:**	Lisa Murphy

Achievements:

I began implementing the booking feature this month. With the help of a tutorial, I have tried to use the knowledge gained to build up my booking feature. Firebase is becoming a real backbone to this feature as the rooms that are available for bookings are set out in firebase. The structure of the Rooms collection is very specific, and proper planning is needed to have a sensical flow.

Added the extra setup page after registration. This page allows the user to save more detailed information about themselves, including their student number and course code. The profile image is not working at the moment as the method I was planning to use is deprecated.

I added some validations for registration and logging, only allowing for NCI email addresses.

Reflection:

During the implementation of the booking feature, I have run into many dependency errors. I have had to refactor the entire application to AndroidX instead of the .com package. Still having issues with the calendar package that I am hoping to use for the booking date step.

The booking feature will require some further planning as I feel it is not currently flowing in the most intuitive way. I can see problems arising with double-booking in the current way that I am implementing it.

Next Actions:

Finding alternative solutions for the two packages that are causing me errors; pictures (deprecated method) and a different calendar package that will allow for horizontal view, and only one week at a time.

I will keep working on the booking feature, and hopefully move on to the map feature as soon as it is completed.

I have found in some of my research that the way to go about this function would be to have the map and calculate the coordinates of each room on the picture and save those x and y values to use for displaying a marker over the correct room. It sounds complex, so I hope to get started on it soon to see if it is attainable, or search for an alternative.

March

Student Name:	Marie Elizabeth Hauroo	**Course:**	BSHCSD4
Student Number:	19140410	**Supervisor:**	Lisa Murphy

Achievements:

Honestly, this month there has been almost no progress on my project. There has been several deadlines for the other modules that have taken up the majority of my time.

I worked a little bit more on my booking feature and got to the booking date step but not past it. I have instead opted to work on some of the documentation such as redoing the use cases etc. I have also made some UI changes to the application and decided to change my colour scheme from purples to blues.

I also added some extra navigation to the side bar, allowing the user to easily go to Moodle, Teams and timetable. It's a mere redirection, but handy to have at hand.

Reflection:

This month has not been as productive as it could have been. But I recognise that the other modules needed more immediate attention. Just because I haven't spent a lot of time physically working on it, doesn't mean I haven't thought about the project.

I have come up with a possible alternative for my mapping of the college, it would be time consuming and maybe not as efficient, so I will try the coordinates option first.

Next Actions:

Next month will be the completion of the other modules, and while there are still assignments to be submitted, I know that the second half of the month will be solely dedicated to the project. I am hopeful that I will make great strides progress-wise in the project.

April

Student Name:	Marie Elizabeth Hauroo	**Course:**	BSHCSD4
Student Number:	19140410	**Supervisor:**	Lisa Murphy

Achievements:

I have decided to re-start the entire booking feature from another angle. The previous way was too complex to keep track of on firebase and was making it almost impossible to cross-reference check for timeslots that are already taken. As a result, people could essentially book the same timeslots for the same room when the only difference was the "reason for booking". Additionally, because of the way it was set out on firebase, to delete (cancel) a booking, the retrieval was far too convoluted. It required multiple nested loops that was very inefficient.

So now I have split the booking steps into only two main steps, choosing the room number first, and then on the second page, picking the reason, date and time. I have also changed when the data is getting sent to Firebase, with the previous way it was at every step, which made the collections nested into themselves multiple times. Now I carry the data from step to step, and only push it to Firebase on the confirmation step, along with a booking Id that makes the bookings easily retrievable. I completed the booking step, and I'm please with how it works. I also completed showing the user's bookings on their own profile page, and cancelling them.

I also opted for my alternative option for the mapping function. I tried to do the first way (mapping x and y values over the correct spots on the image), but it was very difficult to get an accurate position, and took too long for just one, considering that I would have to do this for over 50 rooms, it became less possible.

So now I have set out to retrace the plans that I received from Facilities, as they are over detailed and busy plans. Also, some rooms have since been modified, so I spent some time in the building trying to map out the rooms as accurately as possible. Now I am creating a map image for every room with the selected room “coloured” in, a lengthy process, but I am more confident in getting it to work.

I also decided to include the parking feature in my application. I had originally dismissed it as the way I wanted it to work would need sensors to be installed in the NCI parking. But I decided that even just a simple flag would be a useful add on to the application. So, I designed the Admin version of the application which included the parking switch.

Towards the end of the month, I started implementing the CRUD posts for the feed feature and got as far as getting the posts to appear on the feed, now I want to implement the filtering.

Reflection:

I have made great progress this month. Having the time to solely focus on this project has allowed me to be very productive and make great advancement.

It was a difficult decision to redo the booking feature knowing that time was of the essence, but I could not accept delivering a booking feature that couldn’t prevent double-booking. This new method is much more efficient, and I used my experience from the first implementation to improve upon it. I am much happier with the final result and could now move onto the next features.

I was also happy to include the parking feature, it was a simple enough implementation, but a valuable feature.

At this stage I feel like I have the main functions of my application. I have the logic for the search of the map implemented with “dummy” images until I complete drawing the maps for all the rooms of the college, I just have to import them in the right place now.

Next Actions:

Finishing filtering the feeds and posts feature. I also want to add any finishing touches possible, including allowing the user to edit their details on the settings page, adding a like function to the posts (maybe). Admins should also be able to view all of the bookings for the current day, so I want to add a recycler view on the admin page with that filter.

Focusing on the documentation will be the main focus of the coming two weeks as well as making the video.

May

Student Name:	Marie Elizabeth Hauroo	**Course:**	BSHCSD4
Student Number:	19140410	**Supervisor:**	Lisa Murphy

Achievements:

This month was the final leg, and my project has come along nicely. I have managed to finalise most of my apps loose ends and have spent a lot of time on the documentation and video components for the submission. Most of the coding was completed towards the first week or so of this month.

Reflection:

This month has felt cathartic, and really satisfying to see the final product come to fruition. I do feel that if I had started the work a little earlier in the first semester, I could have avoided the stress of feeling rushed for the last two months.

But I feel really proud of the product I am submitting, and I know that I gave it my best shot. I have given this project a humongous amount of my time and attention over the last nine months, and now it is complete and ready for submission, which makes me really happy.

Next Actions:

Even though it has been a stressful few months working on this app, I have thoroughly enjoyed working on it and do plan to keep tweaking it after submission.