COMP 2000

Contents

Introduction

This document explores the development process of a Java Mobile application as part of the COMP2000 coursework. The app is intended for use on mobile devices, such as Android phones. To see the source code for the application please see Figure 1.

The report starts by introducing the scenario and context of the project – here the project vision is explored, and the reasoning behind developing the project is stated. This planning stage will be further developed in the LSEP section, where the end users and any issues that they may have with the app are considered – such as the security of their data, and how they will use the app. The Design section oversees the creation of diagrams that convey the requirements, and the layout of the GUIs. This section heavily discusses user testing and how that shaped the project. Finally, the report will delve into how the project was coded and implemented, with evidence of it working, and a discussion of issues undergone during development and how these have affected the project and will affect the developer in the future when working on other projects.

GitHub Repository	https://github.com/Plymouth-University/comp2000-main-assignment-ORG4N/tree/main/Assignment%202/ProjectApp
Video Submission	https://www.youtube.com/watch?v=kdV02ISDBss

Figure 1 – Resources

Background

This section of the report discusses the intent of the application – such as what is it and who the intended users are.

As a student I wish to enter my project details
As a student I wish to check my project details have been uploaded.
As a student I wish to edit my project details.
As a student I wish to delete my project details

Figure 2 – User stories

As seen in Figure 2, the Mobile application has four simple requirements, with each requirement being established from the perspective of a *student*. The user stories describe a management application wherein student projects can be created, edited, and deleted. Students wish to use the application alongside their studies so that they can keep track of all the projects that they are working on or have completed.

When a student creates a project, they are expected to supply the following information:

- Student's full name
- Project title, description, and year
- Poster image

The application should therefore allow students to create a project with the above information, and then display a student's projects back to them on a project dashboard. For privacy and security reasons, it can be assumed that a student would only want to see their own projects – and not their peers' projects.

The app could potentially be developed for Classroom usage. Although out of scope of the assignment and not a requirement, teachers could be described as another stakeholder, and they could fulfill the role of Admin wherein they can view all their student's projects.

Legal, Social, Ethical, and Professional Issues

This section of the report highlights any significant considerations that must be made during the design and development processes of the app. These considerations are important as they shape the project to be inclusive and law abiding. It is necessary for developers to focus on their users and how the app can impact them – especially the negative consequences if not made correctly.

Privacy and Security

As the app will be storing and accessing personal information, such as a student's full name and their associated student reference number (id) it is important to consider what could happen if another user gathered this information. If an individual used another student's information to access their dashboard a potential negative consequence could be plagiarism, wherein the individual could steal the other student's ideas for projects. Likewise, the infiltrator could also delete or edit existing projects and sabotage the victim's work.

The app could combat this by incorporating a login system with username and password functionality. This, however, is mildly outside of the scope of the assignment – but has still been considered as the main approach to combatting the above issues, if the app was intended to be deployed in a professional, and not academic, context.

Integrity

Some factors of integrity, such as: maintaining unique identifiers for each entity, and ensuring that uploaded images are all the same datatype, are handled by the API. However, the application itself does ensure that data integrity has been handled. First, and most importantly, a user can only delete a project if it is their own. This stops malicious act of non-consensually deleting other students' projects. Secondly, the code ensures that when a HTTP PUT/POST request is occurring, the data is of a correct datatype.

Another aspect of integrity is input validation. The app, however, does not provide this and a user could potentially input a year longer than four digits (as an example). In a future iteration of the application, input validation would be incorporated, and toasts would be used to relay to a user when an input is in an incorrect format.

Usability and Accessibility

A major concern with the design of the application is that it should be inclusive of all different types of users, including those that suffer from disabilities. Examples of disabilities that the application should aspire to help are motor and visibility disabilities. In both instances, the application could be designed to be bigger so that elements are more visually clear and easier to click on. Brighter colors for specific elements also bring more attention to them and allow for a clearer GUI and the next action that is intended for the user to take is highlighted.

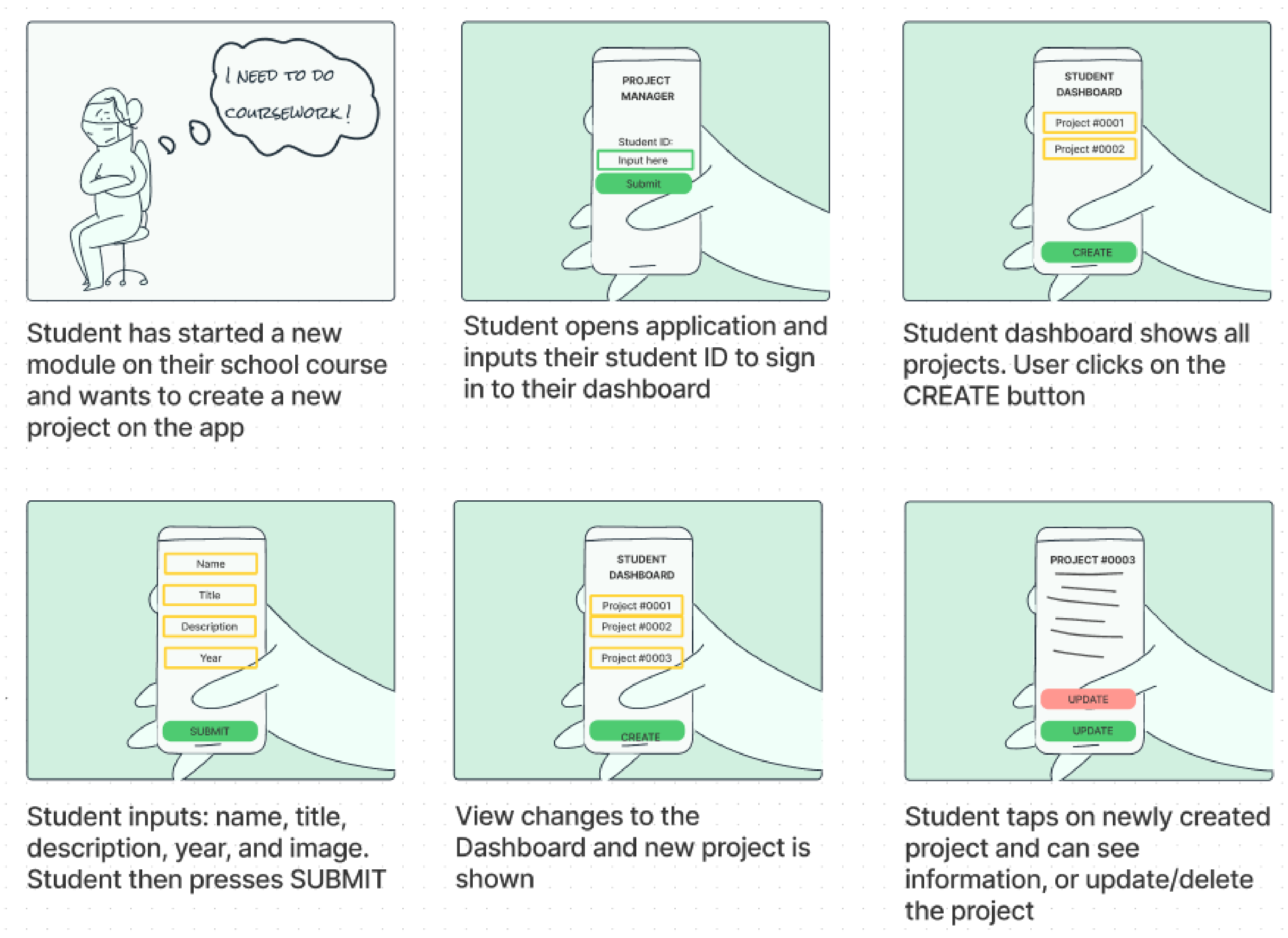
The application that has been developed follows both above solutions as colors are colored and large and easy to press. Text is made bold when it is important, and the delete button is especially differentiated from other components on the screen as it is bright red (associated with consequences or danger).

Popups/Toasts can also be used to make the application responsive – user actions can be explicitly confirmed or denied through notifications. Without these notifications, users may struggle to know if an action has been carried out or denied. The developed app has a notification for the delete process, however notifications for other functionality has yet to be implemented.

Design

This section of the report visualizes the requirements in terms of diagrams and low-fidelity prototypes. Usability testing has been carried out on these prototypes to ensure that they are sufficient to be implemented and turned into high-fidelity prototypes.

At the start of this stage, the following storyboard was made to portray the user's actions when using the application:



Student has started a new module on their school course and wants to create a new project on the app

Student opens application and inputs their student ID to sign in to their dashboard

Student dashboard shows all projects. User clicks on the CREATE button

Student inputs: name, title, description, year, and image. Student then presses SUBMIT

View changes to the Dashboard and new project is shown

Student taps on newly created project and can see information, or update/delete the project

Figure 3 – Low-fidelity storyboard

The storyboard in Figure 3 shows a rough layout of the early app design. Buttons were designed to be very colorful and span the width of the screen so that they can easily be pressed.

The storyboard in Figure 3 was then used to develop a Figma prototype (see Figure 4) which better described the layout of each View. Both prototypes were designed for mobile devices like smart phones.

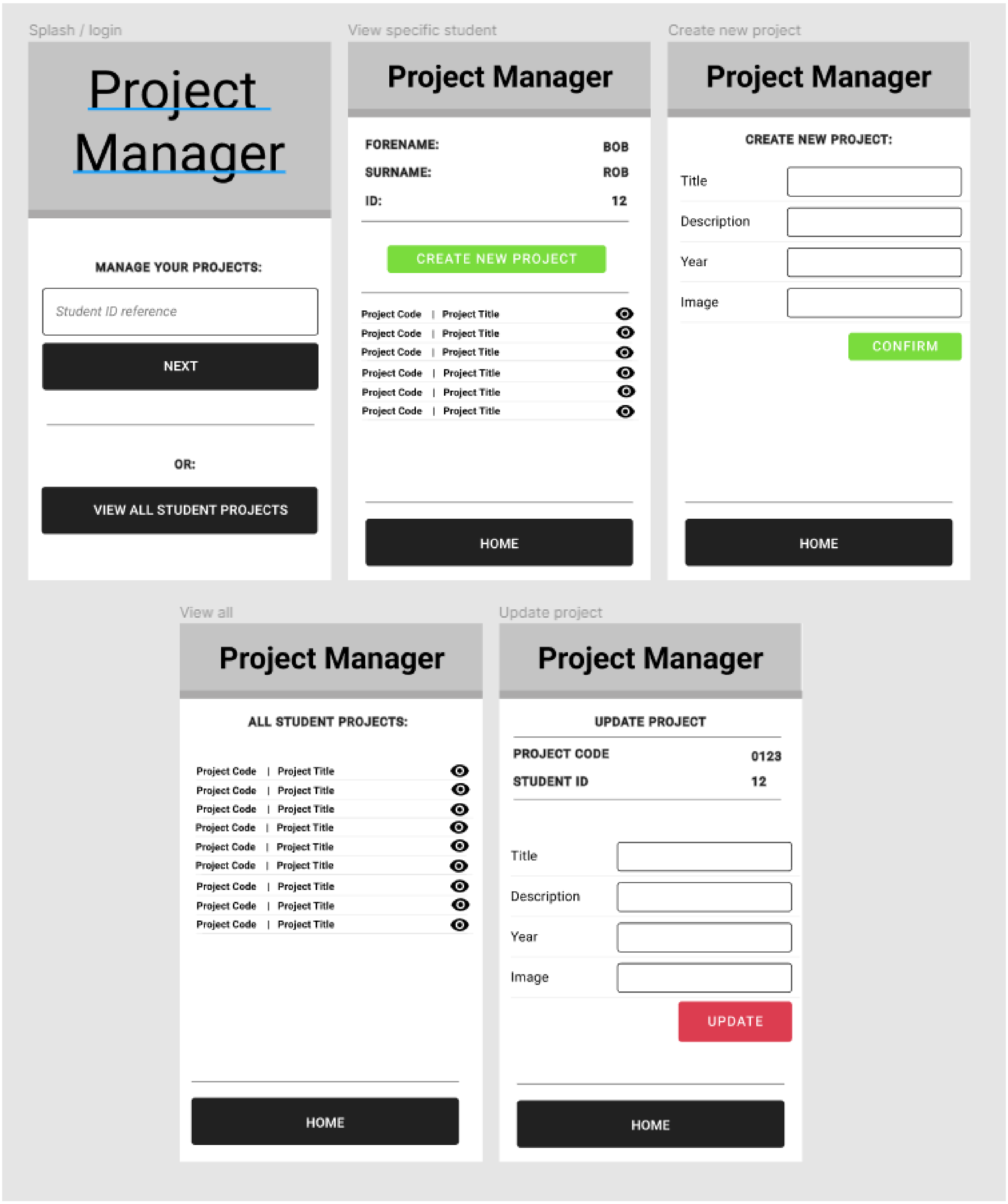

Splash / login
Project Manager
MANAGE YOUR PROJECTS:
Student ID reference
NEXT
OR:
VIEW ALL STUDENT PROJECTS
View specific student
Project Manager
FORENAME: BOB
SURNAME: ROB
ID: 12
CREATE NEW PROJECT
Project Code | Project Title
Project Code | Project Title
Project Code | Project Title
Project Code | Project Title
Project Code | Project Title
Project Code | Project Title
HOME
Create new project
Project Manager
CREATE NEW PROJECT:
Title
Description
Year
Image
CONFIRM
HOME
View all
Project Manager
ALL STUDENT PROJECTS:
Project Code | Project Title
Project Code | Project Title
Project Code | Project Title
Project Code | Project Title
Project Code | Project Title
Project Code | Project Title
Project Code | Project Title
Project Code | Project Title
Project Code | Project Title
HOME
Update project
Project Manager
UPDATE PROJECT
PROJECT CODE 0123
STUDENT ID 12
Title
Description
Year
Image
UPDATE
HOME

Figure 4 – Figma prototype

The Figma prototype shows each of the Views of the application and specifies the View layouts. Once again, color has been used to highlight buttons and buttons are big so that they can easily be pressed. Each view is titled accordingly so that the user can easily navigate the app. Input fields are also labeled so that it is simple to know what to input.

When the student is on the project dashboard screen, they can see all their projects as well as their personal information: forename, surname, ID. They will access a project by clicking it on the list. The Figma prototype therefore provides the functionality outlined in the user stories.

The Figma differs from the storyboard as it allows for connections between elements to be made and therefore buttons can be made operable. Because of this, the Figma is ideal for user testing and was used in the testing process rather than the storyboard.

Feedback and testing are further discussed in Evaluation. This feedback shaped the following High-Fidelity prototype, which is the same version as the app located on the GitHub repository.

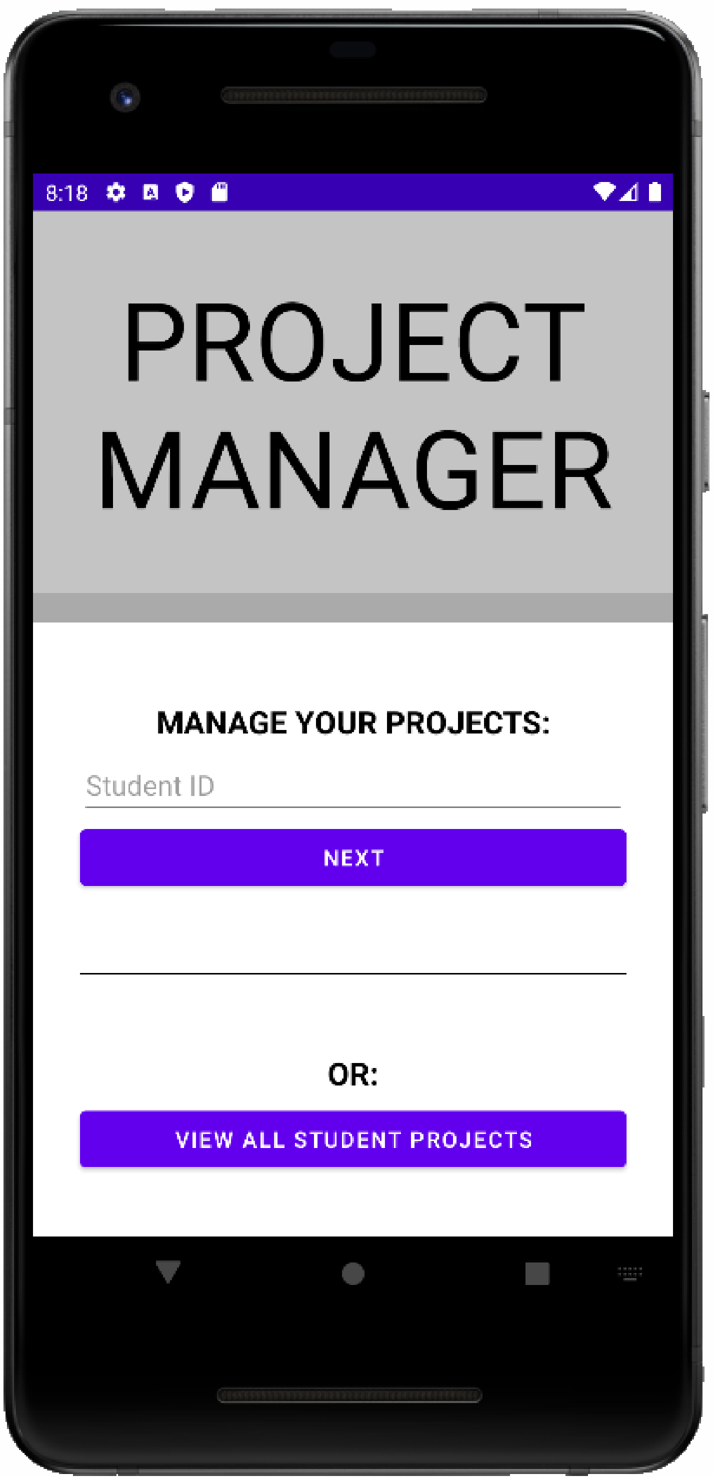
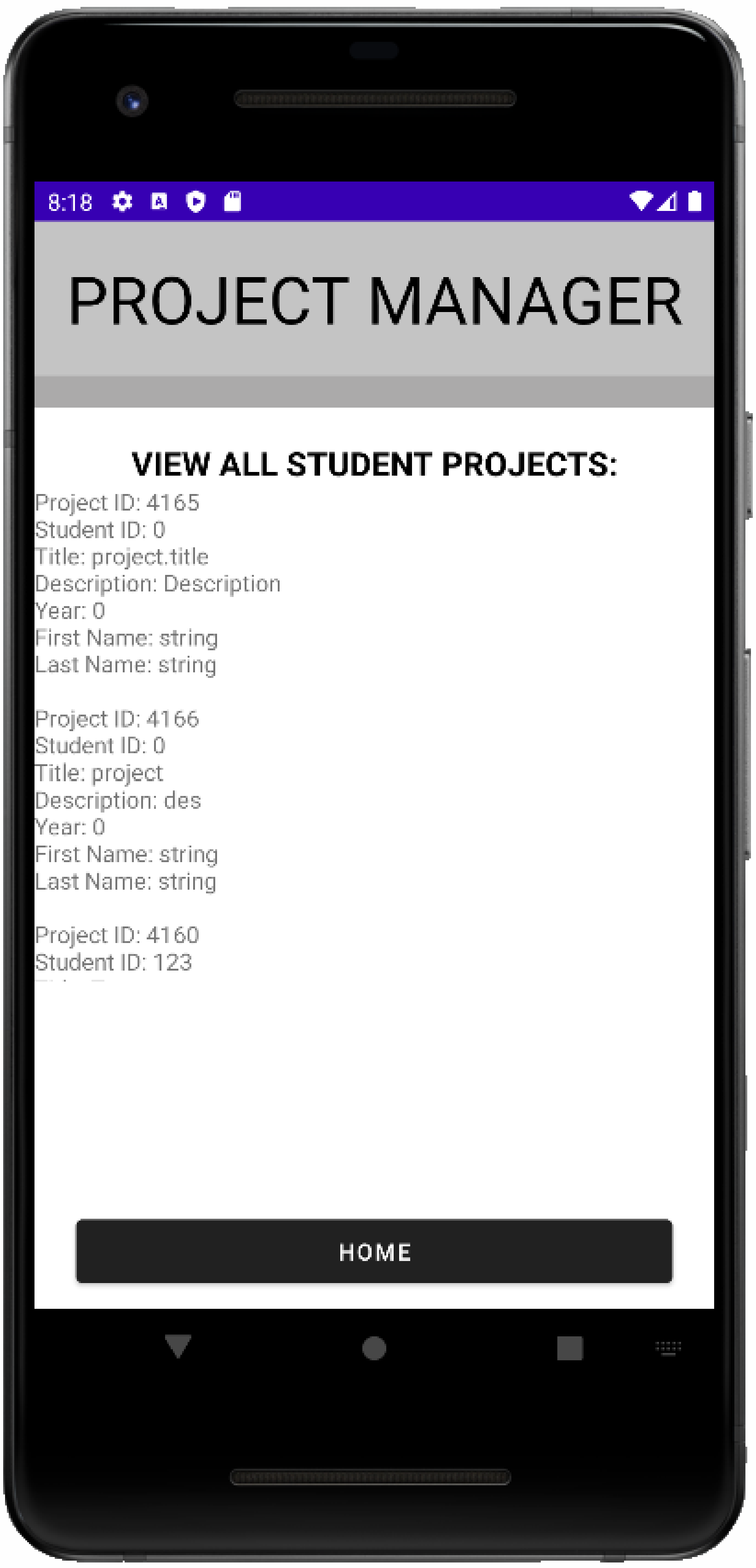


Figure 5 – Home/Login

Figure 5 has two buttons – one takes the user to the dashboard, the other shows them a view of all student projects. Besides from the colors, this View is identical to the Figma prototype.



8:18

PROJECT MANAGER

VIEW ALL STUDENT PROJECTS:

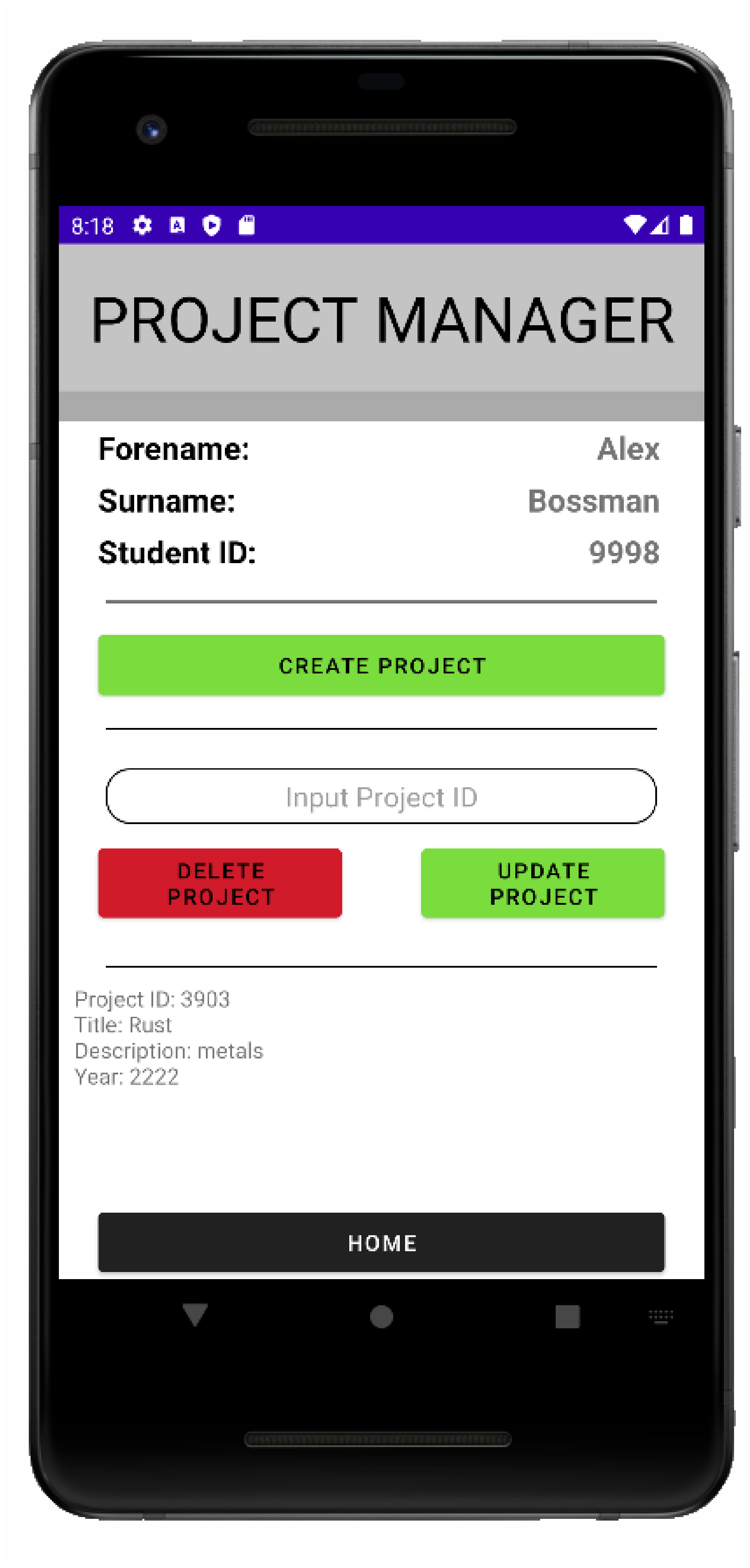
Project ID: 4165
Student ID: 0
Title: project.title
Description: Description
Year: 0
First Name: string
Last Name: string

Project ID: 4166
Student ID: 0
Title: project
Description: des
Year: 0
First Name: string
Last Name: string

Project ID: 4160
Student ID: 123

Figure 6 – View all

A collection of all student projects is found on this view. However, due to time constraints this view was not styled to be identical to the Figma prototype. However, this is not significant as this view is not part of the project's requirements.


8:18

PROJECT MANAGER

Forename: Alex
Surname: Bossman
Student ID: 9998

Project ID: 3903
Title: Rust
Description: metals
Year: 2222

Figure 7 – Dashboard

The dashboard has been simplified from the storyboard as the delete and update have been incoroporated onto this view, reducing the complexity of the app design and making it easier for users to delete/update. Users are expected to input the project id of the project they want to manage before they press one of the two buttons.

This image does not convey it, but if there is a image associated with the record, it will be displayed underneath.

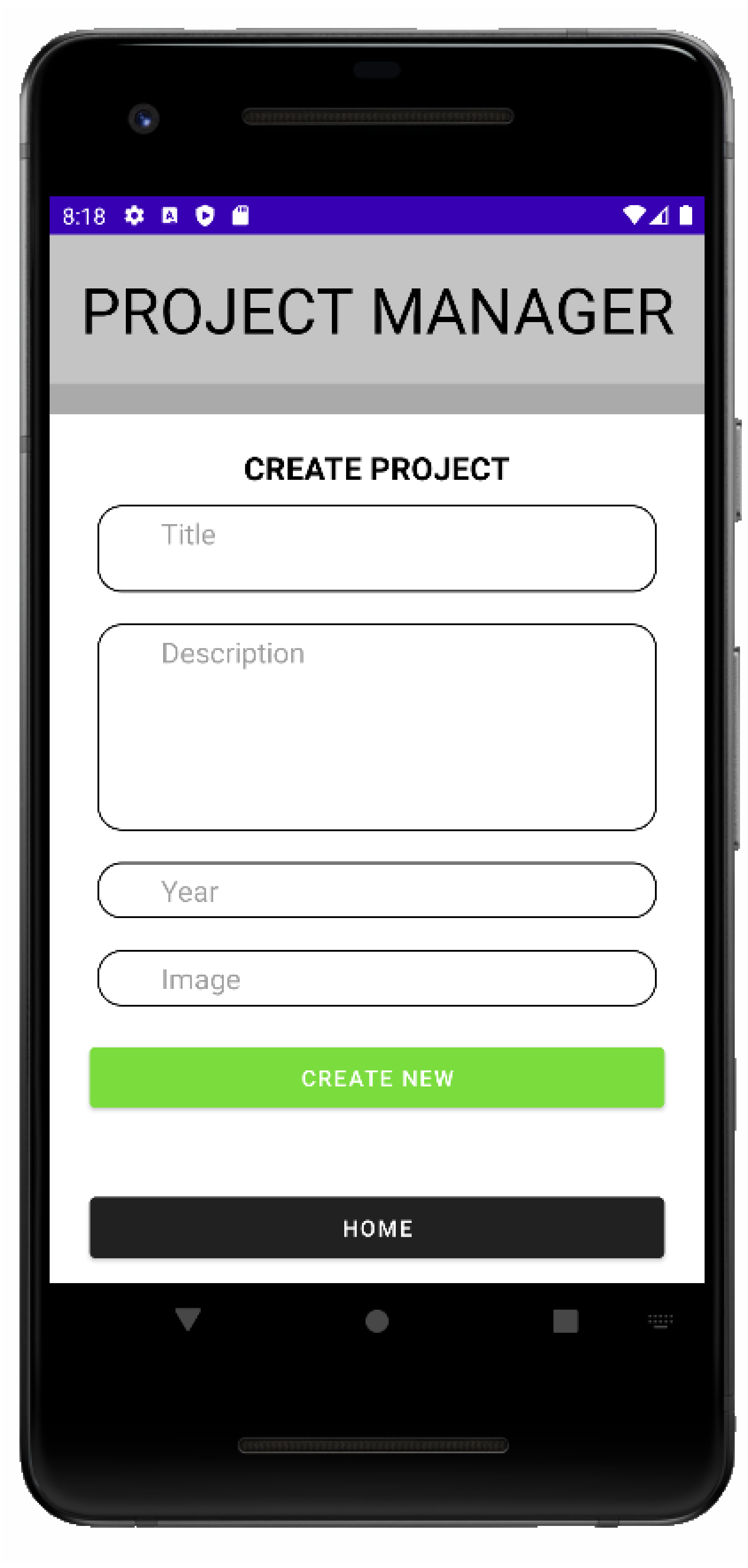


Figure 8 – Create project

This view is also identical to the storyboard, except for the description input field being bigger. This is so that when the user inputs their data, it is easy to read and modify once written.

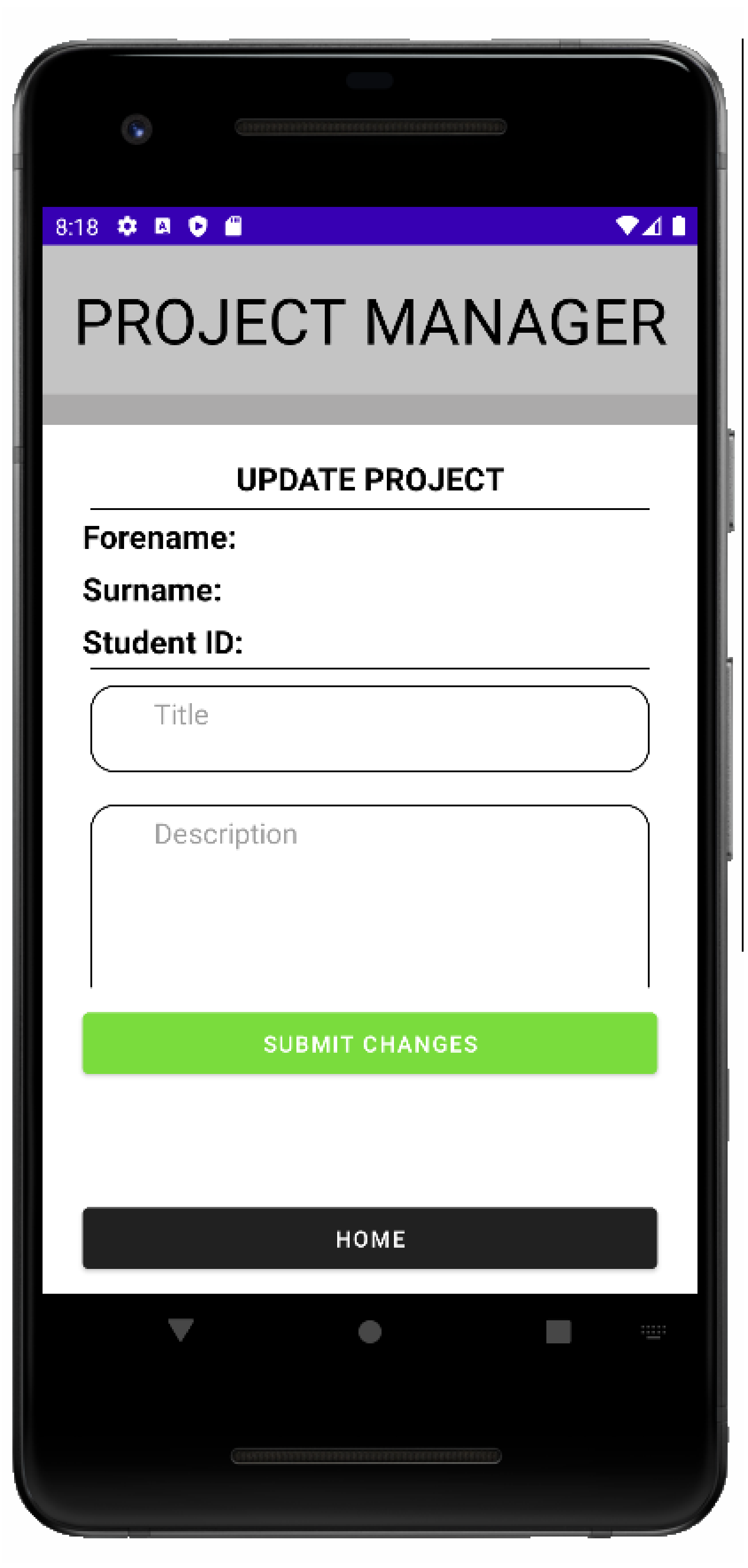


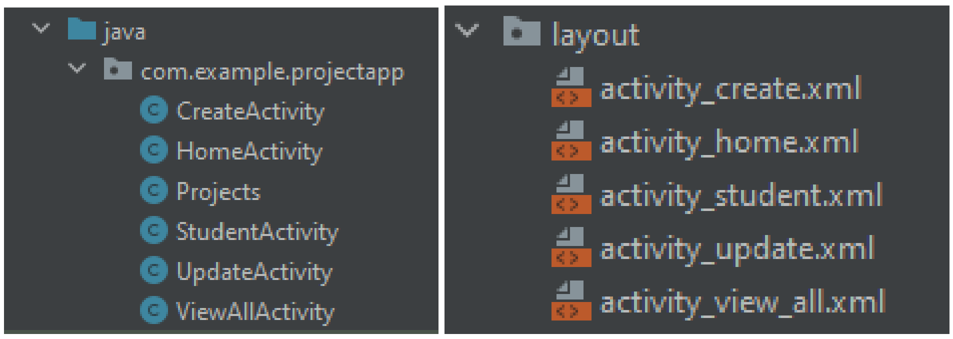
Figure 9 – Update project

This view is also like the storyboard's update view. However, so that all the input fields fit on the screen, the user can scroll through them.

Overall, the Figma prototype and High-Fidelity prototype are very similar to each other and not many changes have been made – largely due to the layout being simple and effective.

Implementation

This section will explore how the project was implemented and discuss design principles.


java
com.example.projectapp
CreateActivity
HomeActivity
Projects
StudentActivity
UpdateActivity
ViewAllActivity

layout
activity_create.xml
activity_home.xml
activity_student.xml
activity_update.xml
activity_view_all.xml

Figure 10 – Activities (classes) and layouts (XML)

Each of the app's views have been built using XML. There are 5 layout views in total for the following aspects of the app:

- Home
- Dashboard
- View all
- Update
- Create

Each XML layout has it's associated Activity which is a Java class. Alongside these Activities is the ***Projects*** model. This model is used to store objects when they are fetched from the HTTP GET. Each XML layout is comprised of a LinearLayout and its children Text or Edit vies. For example, Figure 11 shows a portion of the XML for the home layout.

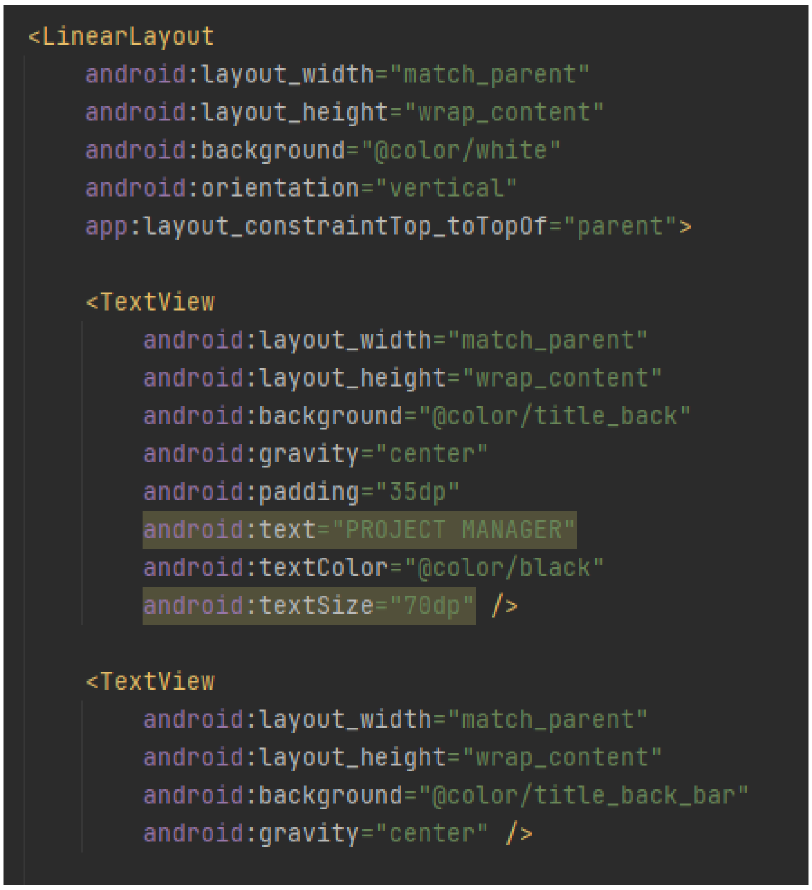
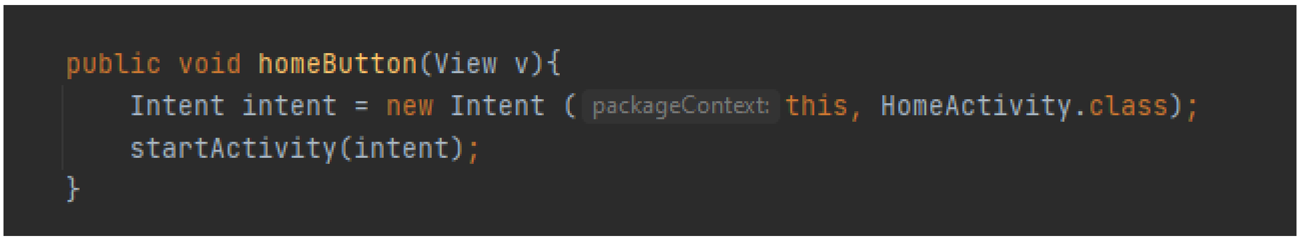

```xml
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@color/white"
    android:orientation="vertical"
    app:layout_constraintTop_toTopOf="parent">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/title_back"
        android:gravity="center"
        android:padding="35dp"
        android:text="PROJECT MANAGER"
        android:textColor="@color/black"
        android:textSize="70dp" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="@color/title_back_bar"
        android:gravity="center" />
```

Figure 11 – Part of activity_home.xml

https://github.com/Plymouth-University/comp2000-main-assignment-ORG4N/blob/main/Assignment%202/ProjectApp/app/src/main/res/layout/activity_home.xml

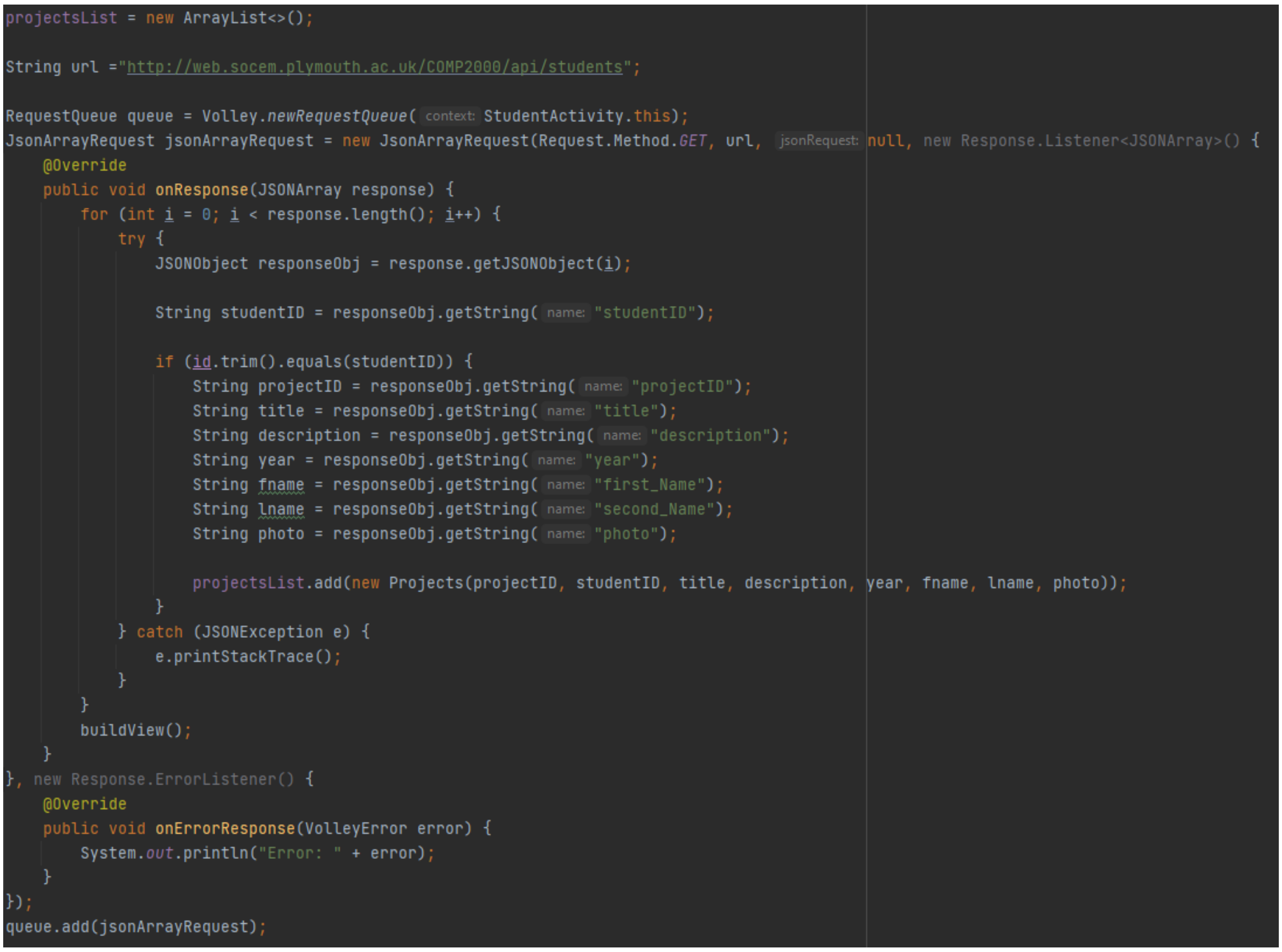


```java
public void homeButton(View v){
    Intent intent = new Intent ( packageContext: this, HomeActivity.class);
    startActivity(intent);
}
```

Figure 12 – Function to change activity

Figure 12 shows how the functions to change activities are typically written in the app. These functions are found within each of the Activity files.

The HTTP GET request was implemented as follows (Figure 13):



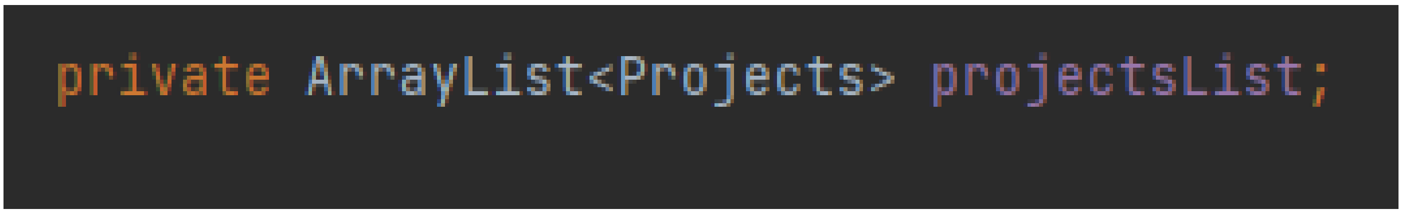
```java
projectsList = new ArrayList<>();

String url ="http://web.socem.plymouth.ac.uk/COMP2000/api/students";

RequestQueue queue = Volley.newRequestQueue( context: StudentActivity.this);
JsonArrayRequest jsonArrayRequest = new JsonArrayRequest(Request.Method.GET, url,  jsonRequest: null, new Response.Listener<JSONArray>() {
    @Override
    public void onResponse(JSONArray response) {
        for (int i = 0; i < response.length(); i++) {
            try {
                JSONObject responseObj = response.getJSONObject(i);

                String studentID = responseObj.getString( name: "studentID");

                if (id.trim().equals(studentID)) {
                    String projectID = responseObj.getString( name: "projectID");
                    String title = responseObj.getString( name: "title");
                    String description = responseObj.getString( name: "description");
                    String year = responseObj.getString( name: "year");
                    String fname = responseObj.getString( name: "first_Name");
                    String lname = responseObj.getString( name: "second_Name");
                    String photo = responseObj.getString( name: "photo");

                    projectsList.add(new Projects(projectID, studentID, title, description, year, fname, lname, photo));
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
        buildView();
    }
}, new Response.ErrorListener() {
    @Override
    public void onErrorResponse(VolleyError error) {
        System.out.println("Error: " + error);
    }
});
queue.add(jsonArrayRequest);
```

Figure 13 – StudentActivity.java

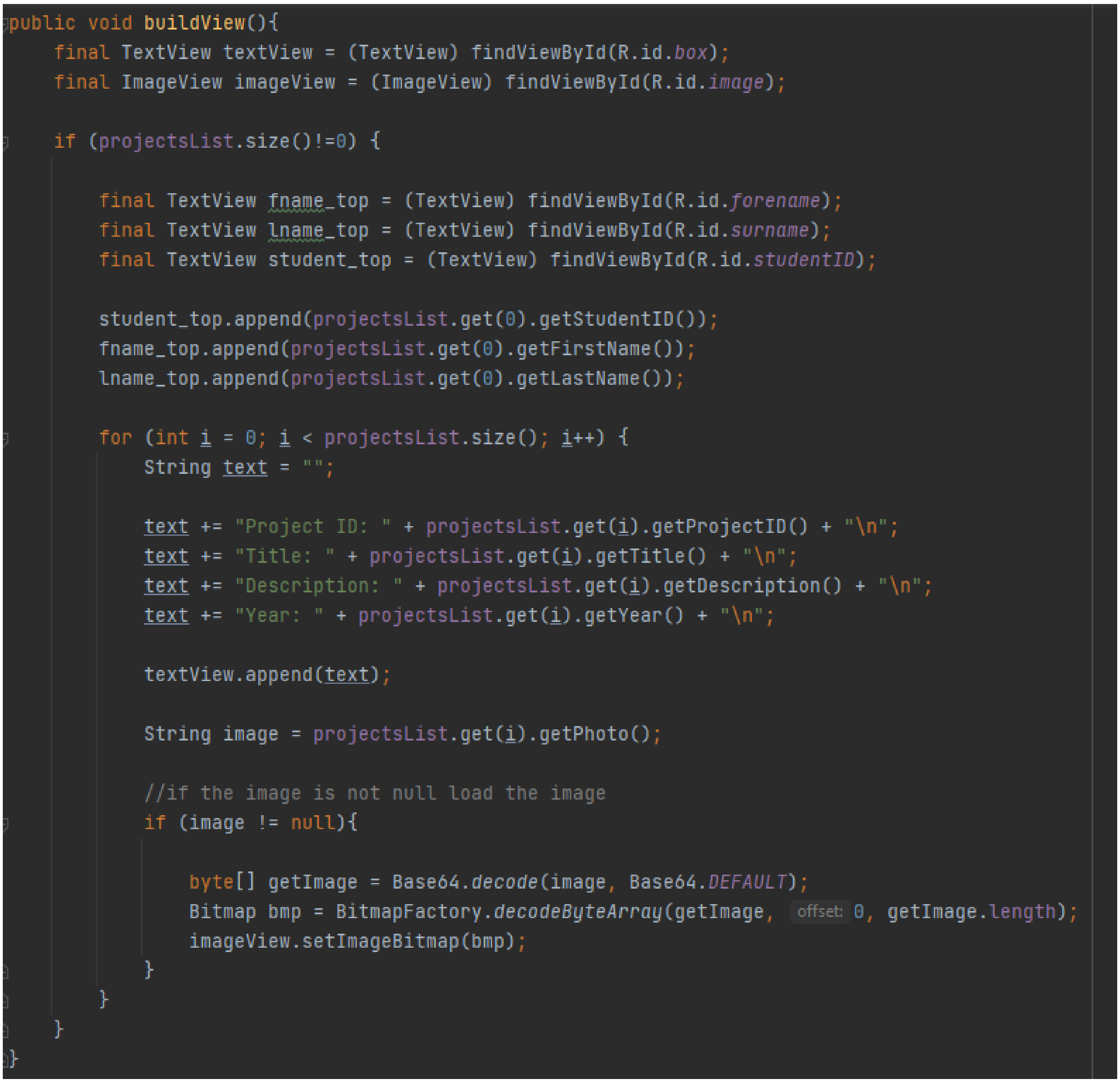
https://github.com/Plymouth-University/comp2000-main-assignment-ORG4N/blob/main/Assignment%202/ProjectApp/app/src/main/java/com/example/projectapp/StudentActivity.java

Using Volley, the above code calls a JsonArrayRequest and uses the GET method to interact with the API stored within the URL variable. When the method receives a response, the Array response is iterated through and each of the array's elements are manipulated and stored into a ArrayList that stores elements of type Projects. Projects is a model that can be found in Figure 15.

The buildView() method is called after each object from the API has been stored into local memory. This method outputs the object to the screen (Figure 14). buildView() is found within the same Java file.



```java
private ArrayList<Projects> projectsList;
```

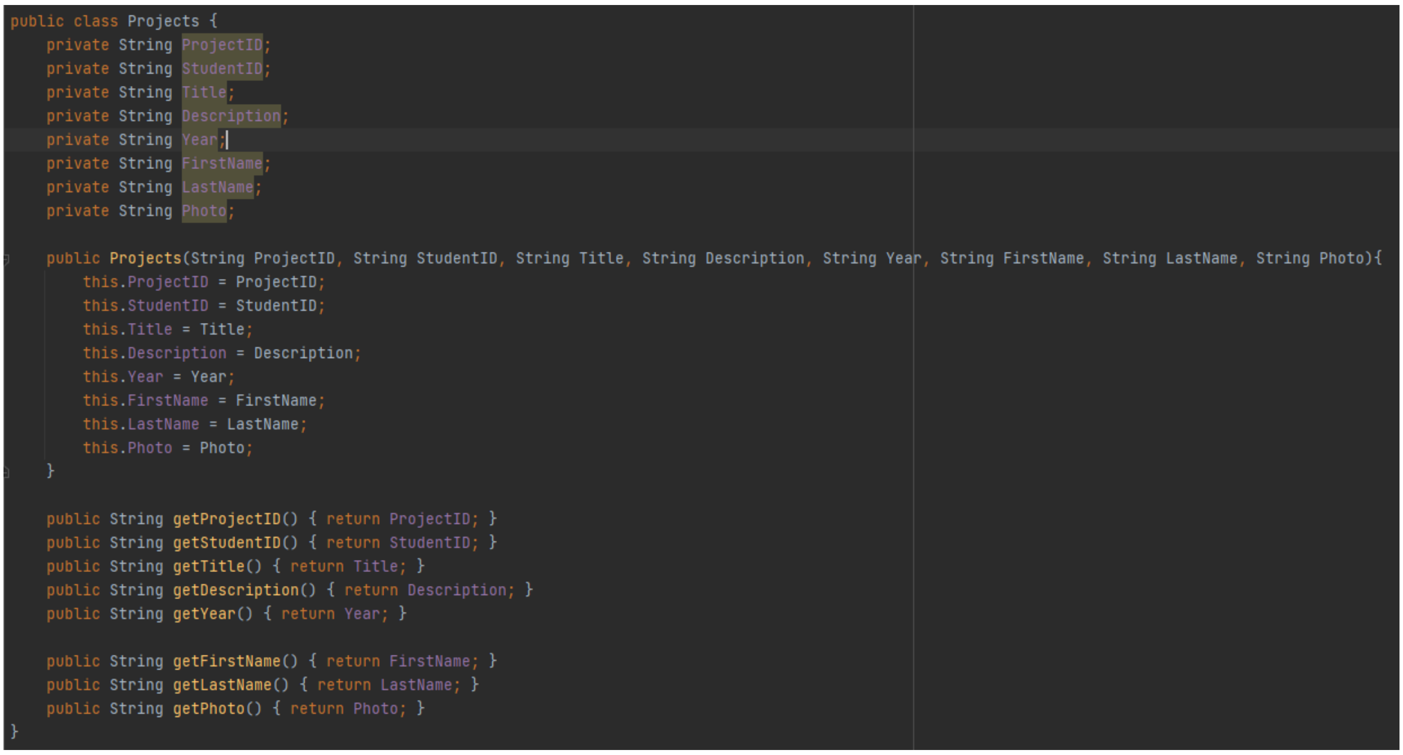


```java
public void buildView(){
    final TextView textView = (TextView) findViewById(R.id.box);
    final ImageView imageView = (ImageView) findViewById(R.id.image);

    if (projectsList.size()!=0) {

        final TextView fname_top = (TextView) findViewById(R.id.forename);
        final TextView lname_top = (TextView) findViewById(R.id.surname);
        final TextView student_top = (TextView) findViewById(R.id.studentID);

        student_top.append(projectsList.get(0).getStudentID());
        fname_top.append(projectsList.get(0).getFirstName());
        lname_top.append(projectsList.get(0).getLastName());

        for (int i = 0; i < projectsList.size(); i++) {
            String text = "";

            text += "Project ID: " + projectsList.get(i).getProjectID() + "\n";
            text += "Title: " + projectsList.get(i).getTitle() + "\n";
            text += "Description: " + projectsList.get(i).getDescription() + "\n";
            text += "Year: " + projectsList.get(i).getYear() + "\n";

            textView.append(text);

            String image = projectsList.get(i).getPhoto();

            //if the image is not null load the image
            if (image != null){

                byte[] getImage = Base64.decode(image, Base64.DEFAULT);
                Bitmap bmp = BitmapFactory.decodeByteArray(getImage, offset: 0, getImage.length);
                imageView.setImageBitmap(bmp);
            }
        }
    }
}
```

Figure 14 – StudentActivity.java

https://github.com/Plymouth-University/comp2000-main-assignment-ORG4N/blob/main/Assignment%202/ProjectApp/app/src/main/java/com/example/projectapp/StudentActivity.java

This method is essentially a variety of string constatations and then setting the text property of a view to the large string being made.

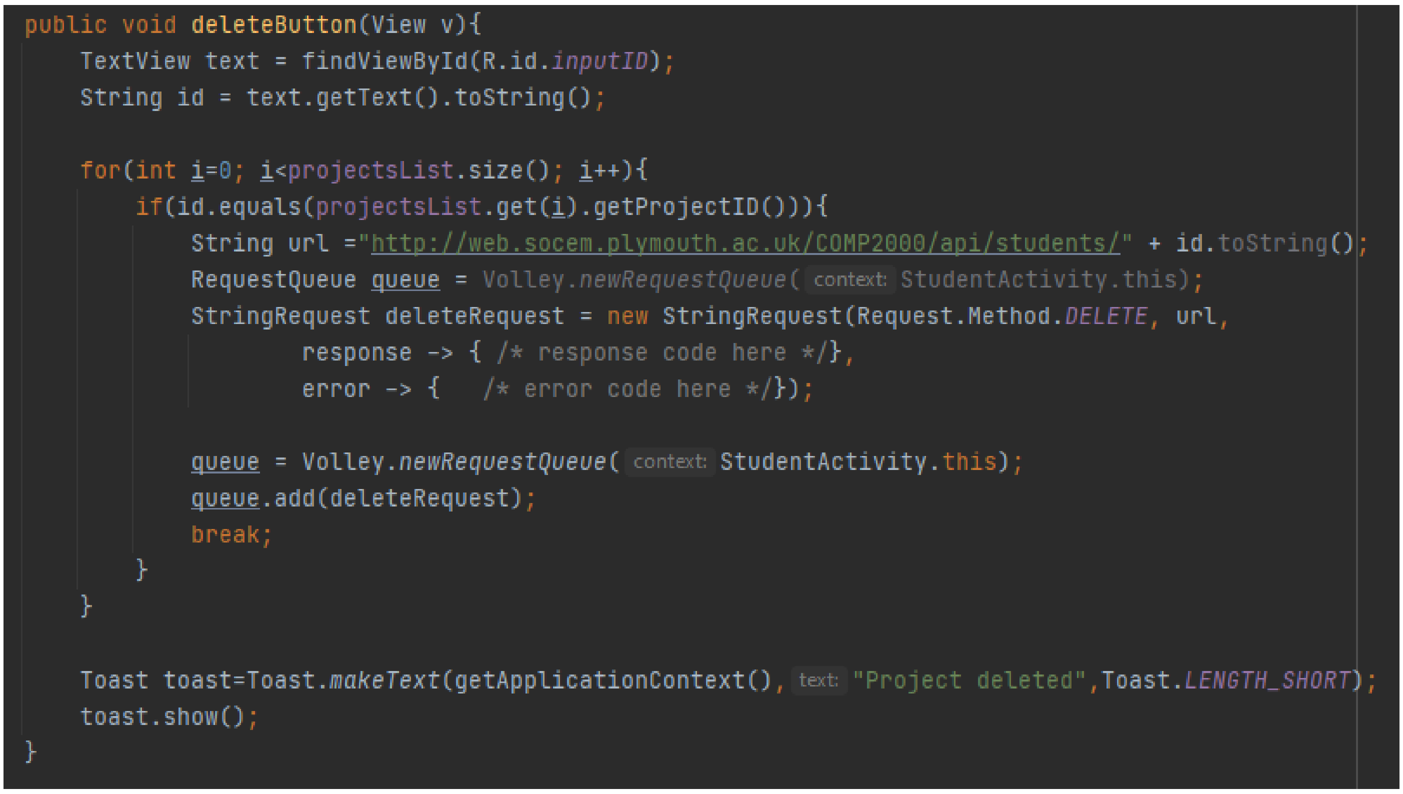


```java
public class Projects {
    private String ProjectID;
    private String StudentID;
    private String Title;
    private String Description;
    private String Year;
    private String FirstName;
    private String LastName;
    private String Photo;

    public Projects(String ProjectID, String StudentID, String Title, String Description, String Year, String FirstName, String LastName, String Photo){
        this.ProjectID = ProjectID;
        this.StudentID = StudentID;
        this.Title = Title;
        this.Description = Description;
        this.Year = Year;
        this.FirstName = FirstName;
        this.LastName = LastName;
        this.Photo = Photo;
    }

    public String getProjectID() { return ProjectID; }
    public String getStudentID() { return StudentID; }
    public String getTitle() { return Title; }
    public String getDescription() { return Description; }
    public String getYear() { return Year; }

    public String getFirstName() { return FirstName; }
    public String getLastName() { return LastName; }
    public String getPhoto() { return Photo; }
}
```

Figure 15 – Projects.java

https://github.com/Plymouth-University/comp2000-main-assignment-ORG4N/blob/main/Assignment%202/ProjectApp/app/src/main/java/com/example/projectapp/Projects.java

This model defines the attributes in the model, and then a constructor, and then finally methods that can be used to get the variables.

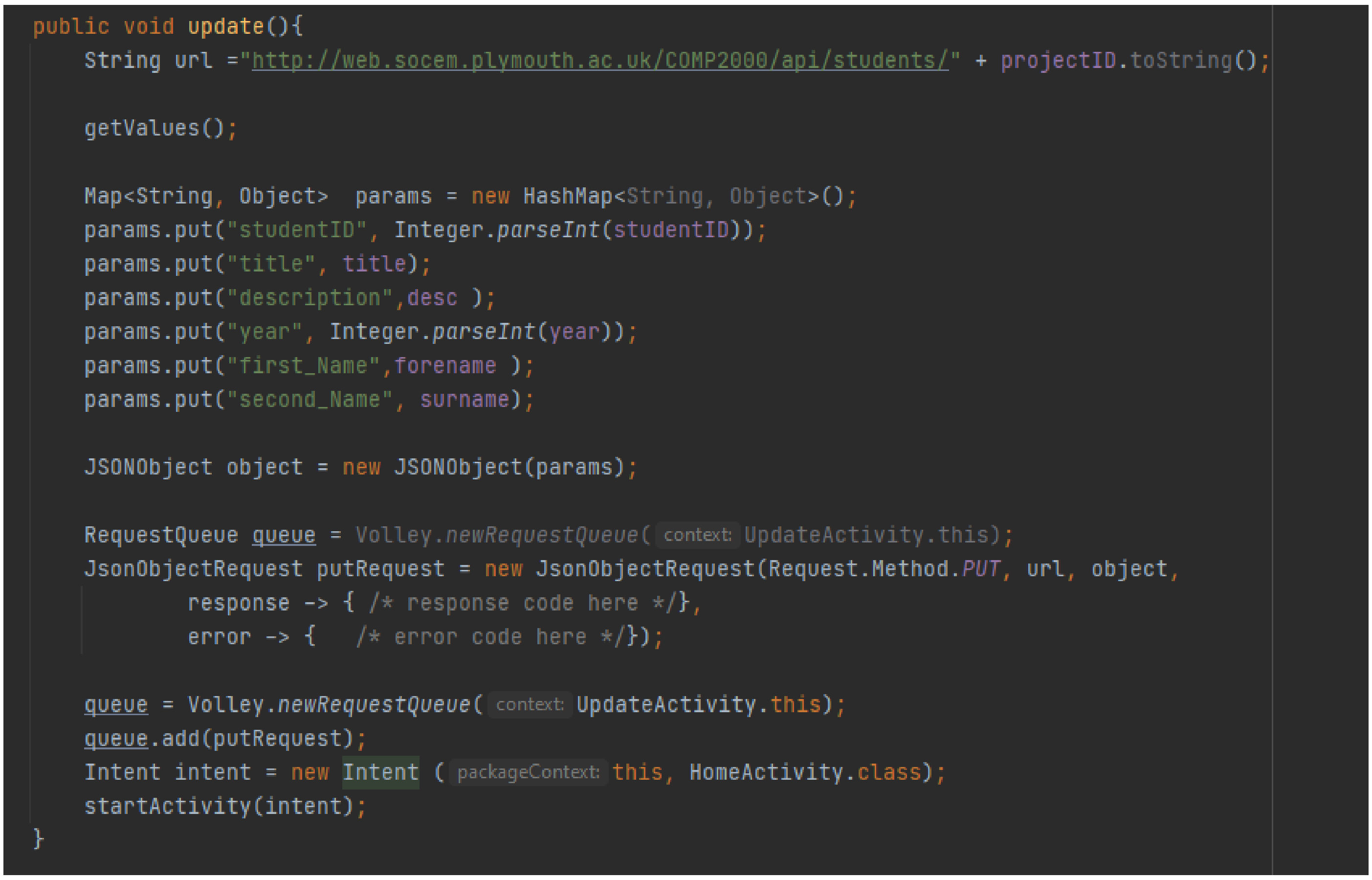


```java
public void deleteButton(View v){
    TextView text = findViewById(R.id.inputID);
    String id = text.getText().toString();

    for(int i=0; i<projectsList.size(); i++){
        if(id.equals(projectsList.get(i).getProjectID())){
            String url ="http://web.socem.plymouth.ac.uk/COMP2000/api/students/" + id.toString();
            RequestQueue queue = Volley.newRequestQueue( context: StudentActivity.this);
            StringRequest deleteRequest = new StringRequest(Request.Method.DELETE, url,
                    response -> { /* response code here */},
                    error -> {   /* error code here */});

            queue = Volley.newRequestQueue( context: StudentActivity.this);
            queue.add(deleteRequest);
            break;
        }
    }

    Toast toast=Toast.makeText(getApplicationContext(), text: "Project deleted",Toast.LENGTH_SHORT);
    toast.show();
}
```

Figure 16 – StudentActivity.java

https://github.com/Plymouth-University/comp2000-main-assignment-ORG4N/blob/main/Assignment%202/ProjectApp/app/src/main/java/com/example/projectapp/StudentActivity.java

The DELETE request is also issued from within the StudentAcitivity. Here it is shown that a DELETE request is issued, and the URL is modified so that a specific record is deleted. At the end of the method, a toast is displayed for responsive messaging. The conditional IF statement ensures that the record is only deleted if the project is part of the student's individual projects.

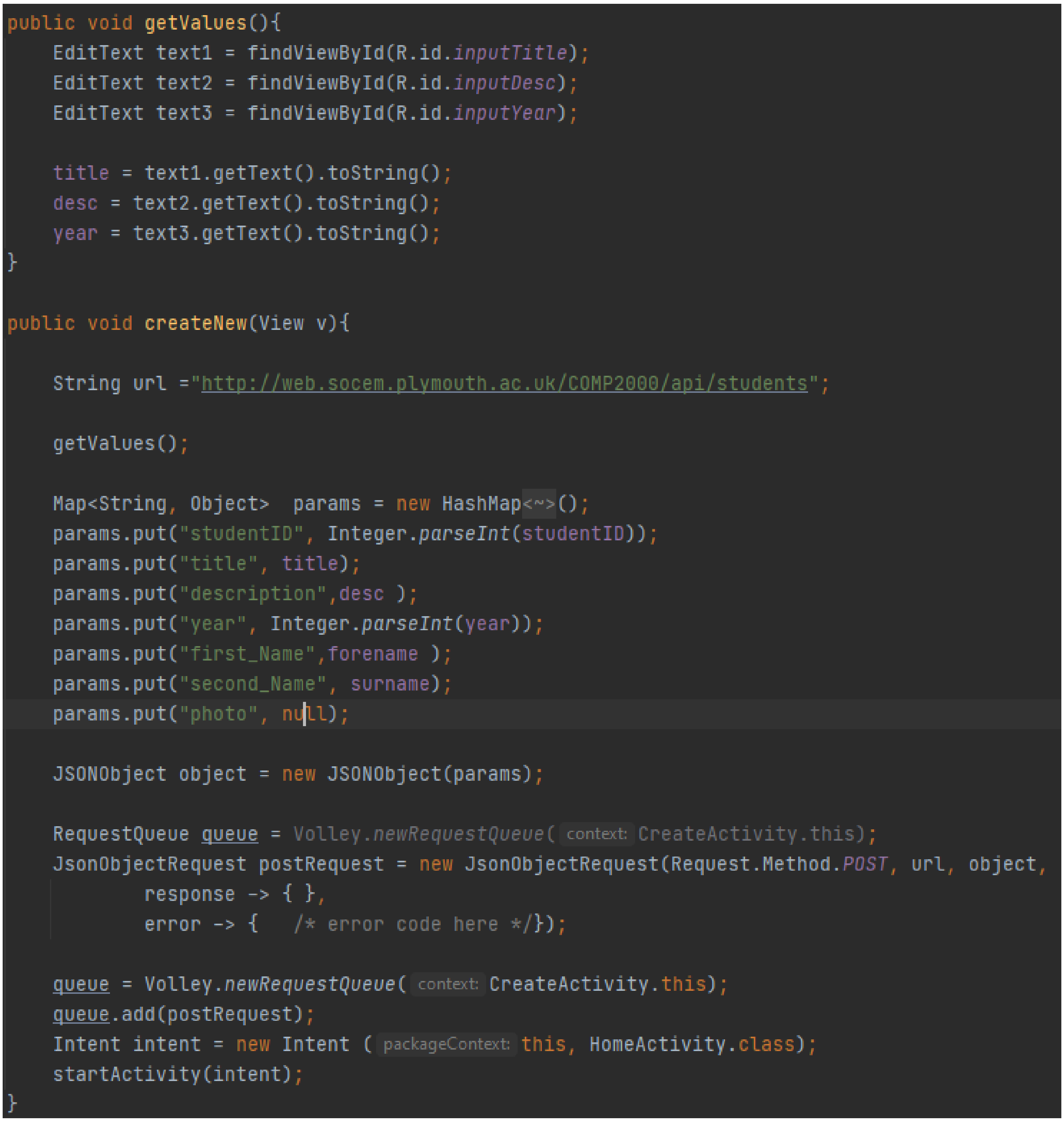


```java
public void update(){
    String url ="http://web.socem.plymouth.ac.uk/COMP2000/api/students/" + projectID.toString();

    getValues();

    Map<String, Object>  params = new HashMap<String, Object>();
    params.put("studentID", Integer.parseInt(studentID));
    params.put("title", title);
    params.put("description",desc );
    params.put("year", Integer.parseInt(year));
    params.put("first_Name",forename );
    params.put("second_Name", surname);

    JSONObject object = new JSONObject(params);

    RequestQueue queue = Volley.newRequestQueue( context: UpdateActivity.this);
    JsonObjectRequest putRequest = new JsonObjectRequest(Request.Method.PUT, url, object,
            response -> { /* response code here */},
            error -> {   /* error code here */});

    queue = Volley.newRequestQueue( context: UpdateActivity.this);
    queue.add(putRequest);
    Intent intent = new Intent ( packageContext: this, HomeActivity.class);
    startActivity(intent);
}
```

Figure 17 – UpdateActivity.java

https://github.com/Plymouth-University/comp2000-main-assignment-ORG4N/blob/main/Assignment%202/ProjectApp/app/src/main/java/com/example/projectapp/UpdateActivity.java

Like the GET request, this PUT request uses the Volley library. The difference between the two methods is that Figure 16 uses the PUT method and is sending the variable ***object*** to the API to store. Also, the URL is modified to ensure that the record in the database being edited is the one that the user specifies. The variable object is constructed by creating a HashMap called ***params*** and storing key, pair values that match the structure of the API. This HashMap is converted into the object that the PUT request uses.



```java
public void getValues(){
    EditText text1 = findViewById(R.id.inputTitle);
    EditText text2 = findViewById(R.id.inputDesc);
    EditText text3 = findViewById(R.id.inputYear);

    title = text1.getText().toString();
    desc = text2.getText().toString();
    year = text3.getText().toString();
}

public void createNew(View v){

    String url ="http://web.socem.plymouth.ac.uk/COMP2000/api/students";

    getValues();

    Map<String, Object>  params = new HashMap<~>();
    params.put("studentID", Integer.parseInt(studentID));
    params.put("title", title);
    params.put("description",desc );
    params.put("year", Integer.parseInt(year));
    params.put("first_Name",forename );
    params.put("second_Name", surname);
    params.put("photo", null);

    JSONObject object = new JSONObject(params);

    RequestQueue queue = Volley.newRequestQueue( context: CreateActivity.this);
    JsonObjectRequest postRequest = new JsonObjectRequest(Request.Method.POST, url, object,
            response -> { },
            error -> {   /* error code here */});

    queue = Volley.newRequestQueue( context: CreateActivity.this);
    queue.add(postRequest);
    Intent intent = new Intent ( packageContext: this, HomeActivity.class);
    startActivity(intent);
}
```

Figure 18 – CreateActivity.java

https://github.com/Plymouth-University/comp2000-main-assignment-ORG4N/blob/main/Assignment%202/ProjectApp/app/src/main/java/com/example/projectapp/CreateActivity.java

The create request is as outlined above. It is very similar to the PUT request. However, the URL is not modified, and the POST method is used instead.

Evaluation

The following test plan was created and is relevant for when testing both the Figma prototype and the final High-level prototype developed within Java.

Scenarios: As a student I wish to enter my project details As a student I wish to check my project details have been uploaded. As a student I wish to edit my project details. As a student I wish to delete my project details	
Scenario Specific Goals	1. To ensure that students can create, delete, edit, and view projects 2. To ensure that information is displayed clearly and effectively
Quantitative Measurement List	• Time taken to complete the task • Difficulty in completing task (usability) • Number of issues uncovered
Task List / Procedure	1. Input student ID 2. View existing projects 3. Create new project 4. Update that project 5. Delete that project
Qualitative Measurement List	• Accessibility • Navigability
Potential Observation of Users	Observe: • Which elements are slow/difficult to interact with? • Which elements are intuitive/easy to interact with? • Body language and facial expressions (Confused? Comfortable?)
Test Setup Details	1. Prepare questionnaire to input answers 2. Prepare workstation (eliminate distractions) 3. Reintroduce context of the prototype formally 4. Sign consent form 5. Proceed with testing when ready 6. Follow up with questionnaire 7. End session

Figure 19 – Usability test plan

Due to time constraints, only the Figma prototype was tested, and when testing the user, the above specification was used. The individual being tested was asked to sign the following permission (Figure 20) form before any testing occurred.

Usability Prototype Test consent form:

Your involvement within the testing process will include:

- Performing certain tasks using a given prototype
- Providing feedback on your experience using this prototype

All information used within this study will remain private and will only be used for the purposes of improving the prototype in accordance with the feedback required. Any identifiable information provided will be anonymised – at no time will this type of information be shared. At any time, if you wish, you can refuse to continue taking part in the study.

By signing below, you consent to carrying out testing for the prototype, and you understand what is required from you.

Signature: …………………………………… Date: ……………………………………

Figure 20 – Consent form

The following feedback was acquired from testing one individual:

Test Subject Questionnaire Results:

- What was your overall opinion of the app?
  - Very clear and intuitive to use. Would definitely use in real life.
- What were your favourite parts about the app?
  - Buttons are colourful and it is easy to know what to do
- What were your least favourite parts about the app?
  - Response times when getting data is slow
- What are your opinions on the UI (colour, layout etc.)?
  - Layout is clear, colour is good. Text is good size. S
- Could you ever see yourself using this app in real life?
  - Yes
- Are there any features you think aren't needed?
  - No
- Are there any features you think need to be added?
  - Sign in features so that only I can access my projects
- If you could change any existing part of the app, what would it be?
  - I want to be able to set tasks and goals for my project outside of the description – better management.
- If you have any disabilities, how was your experience using the app?
  - N/A

Figure 21 – Questionnaire results

This method of testing was preferred over other methods because it was enacted in a controlled environment and questions are tailored to being open and informative as well as qualitive. The mixture between these two types of questions allows for good quality feedback to be received. The feedback, however, was not applicable and went outside of the scope of the assignment. Where the test subject said they liked the layout though, this gave reason to keep the design the same and implement it almost identically in the high-level prototype.

Summary

Overall, the project went smoothly. An app was developed that could CREATE/DELETE/UPDATE records in a database that relate to a student's project. However, due to time constraints not much iteration could be performed over the duration of the project and user testing was minimal. Also, a big feature of the API was image uploading but the final version of the app did not have this incorporated.

In the future, this app could be developed to have this image upload functionality, as well as incorporating a login functionality and potentially allowing for more info to be stored on projects, such as deadlines, and tasks to do.