ASHESI UNIVERSITY

HOSTELDESK: MULTIMEDIA INCIDENT REPORTING APPLICATION

APPLIED PROJECT

B.Sc. Computer Science

Jefferson Reuben Nii Adama Welbeck

2021

ASHESI UNIVERSITY

HOSTELDESK: MULTIMEDIA INCIDENT REPORTING APPLICATION

APPLIED PROJECT

Applied Project submitted to the Department of Computer Science, Ashesi University College in partial fulfillment of the requirements for the award of Bachelor of Science degree in Computer Science.

Jefferson Reuben Nii Adama Welbeck

2021

DECLARATION

I hereby declare that this applied project is the result of my original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

…………………………………………………………………………………………..

Candidate's Name:

Jefferson Reuben Nii Adama Welbeck

……………………………………………………………………………………………

Date:

13th May 2021

..…………………………………………………………………………………………

I hereby declare that preparation and presentation of this applied project were supervised following the guidelines on supervision of applied project laid down by Ashesi University College.

Supervisor's Signature:

……………………………………………………………………………………………

Supervisor's Name:

Mr. David Sampah

……………………………………………………………………………………………

Date:

13th May 2021

….…………………………………………………………………………………………

Acknowledgements

I thank God for making it possible to complete this applied project successfully. I express my many thanks to Mr. David Sampah for his guidance and support through every stage of the project. I also appreciate the support of my family and friends for their encouragement and help in seeing this project through.

Abstract

University hostels owned by individuals cannot afford to hire multiple personnel to manage the hostels' operations, from booking students to maintaining facilities. Instead, they employ individual persons to handle the day-to-day activities in the hostel since these owners cannot be available all the time. Due to the manager's many responsibilities, they often focus on general problems and neglect issues reported by students creating frustration and dissatisfaction. With the many issues occurring around the hostel, these managers often either find it challenging to keep track of these problems and/or forget these problems unless reminded by students.

The project seeks to design a hostel front desk incident reporting application based on data obtained from a survey questionnaire administered to students and managers of hostels.

The HostelDesk, an incident reporting application, should assist managers receive reports from students and attend to multiple issues promptly. Again, it can help students report their issues promptly without waiting for personnel. This paper seeks to improve the communication gap between students and managers in solving problems in hostels.

Table of Contents

List of Figures

Chapter 1: Introduction

Universities provide students with facilities and services that promote a good learning environment for them. Among these services is the provision of accommodation for students usually managed by public or private institutions. The provision of accommodation services does not come without complaints from students, who mainly use such facilities. In addressing these complaints, larger institutions use helpdesk systems. These institutions pay and hire personnel to handle the integration and implementation of these systems. Due to its complex nature, training has to be provided for staff to use the system. Private hostel owners on the other hand are unable to invest in application systems to handle these challenges easily. They mostly hire individuals to handle all operations in the hostel manually. This chapter expands on the problem of poor maintenance practices and miscommunication between students and managers in resolving student complaints and challenges in hostels.

1.1 Background

Universities represent a utopia where students can learn to foster relationships and live independently by being responsible for their well-being and studies. These academic institutions provide students with the necessary education and skills to succeed. According to Hassanain, higher education is intended to help students attain intellectual competence, enliven personal character, and aid in forming patterns of behavior, thought, and imagination that creates a fulfilling living experience [5]. For students to get the whole experience, as Hassanain stated, campus housing is an integral component for universities. Campus housing, such as hostels, bring students from different cultures and fields to share experiences and create memories.

Most campus housing facilities differ, but they all provide students with the basic facility requirements such as electricity supply, gas, beds, wardrobes, shared or individual toilet, bathroom and kitchen, among others. These facilities offer students comfort and satisfaction, and create an environment where they can live and learn. However, due to increased student enrollment in most universities, the facilities available at the universities are most often unable to accommodate all students.

Aside, most facilities created for a limited number of people per room now contain more students than their intended capacity [5]. The limited space leads to more inadequate and dilapidated facilities and create overcrowding in these hostels, leading to poorer services and management of these facilities. With the increase in student population, some private companies and individuals have invested in hostels accommodation to address this issue [5]. Though these private hostels tackle overpopulation, students continue to deal with poor management, facilities and services.

1.2 Problem Statement

Universities provide an environment for students to acquire knowledge and create new relationships by living on and around campus. Students select hostels from various options, with each hostel having a leverage over the other in terms of pricing, space, facilities, and services. Literature outlines five different forms of hostel management, namely [5]:

- Private hostel managed by the private sector property managers
- Institutional hostels managed by private sector property managers
- Institutional hostels managed by institutional property managers
- Institutional hostels managed by students
- Institutional hostels in a built, operate and transfer (BOT) arrangement

The two forms in Ashesi are private hostels managed by the private sector, which are off-campus hostels and institutional hostels managed by institutional property managers, each of which are run by separate entities.

Most hostels off-campus are owned by private individuals, each providing different facilities in terms of accommodation and services. Most of these private owners employ individuals to manage the issues and affairs of these hostels. These individuals are responsible for handling and resolving any problems that may occur, from assigning rooms to students through payments to maintaining facilities and equipment. Moreover, these hostels provide a few services like study rooms and shops where students can purchase goods and study. A key issue in managing hostels is maintenance and ensuring that maintenance services are prompt to enhance students' satisfaction. Maintenance is defined by Abdul Lateef et al. as the procedures taken to watch over the building after completion to run and function as expected [4]. Wahab and Basari also define it as the technical and administrative activities to fulfill their occupants' needs [4]. It is important to preserve facilities and equipment to improve student satisfaction. Moreover, communicating these problems for prompt addressing also remains a challenge.

Other issues like room space and accommodation where the room size does not match the number of students for that room also pose a challenge. For instance, some hostels have rooms that are small in size and still accommodate more students to increase profit. In other situations, students would have to find a way of booking their rooms quickly. This is due to the rush that occurs at the beginning of school semester, the absence of adequate information on room availability and the requirement for ready payment by students. These factors make it difficult for most students to secure their rooms. Furthermore, maintaining these facilities for future use by both managers and students at all times remain a critical issue. Repairs or maintenance works are unplanned, hostel managers mostly react only when complaints are persistent, to the extent that some tenants finally end up being responsible for the repairs [5]. Students sometimes pay not only for the accommodation but maintenance and services as well - complaints by students on repairs; changes made within the facilities, and provision of other amenities such as water, electricity, and shops. Other problems like broken toilets seats, internet, water supply, electricity or broken furnishes are either ignored or take a long time to fix, putting the burden on some students. The absence of an effective communication and information system between students and hostel managers makes it difficult to report and resolve these issues.

This paper proposes the development of a mobile application that allow managers to receive complaints or incidents that have occurred around and within the hostels. Students will be able to report on issues and provide details in the form of images and videos. Reporting these problems through the application will improve communication between students and managers of private hostels, aside addressing students' complaints speedily. Also, managers will be notified by the application on the issues existing in the hostel to improve maintenance.

1.3 Related Work

"Housing creates a foundation for stable communities, social inclusion and an enabling environment for students to study" [1]. In a university like Kwame Nkrumah University of Science and Technology (KNUST), the increase in student population makes it difficult for the school to handle accommodation for its students [1]. Due to unavailable hostel space, some students seeking accommodation have no choice but to move to private hostels around the campus. These hostels are registered with the school and are inspected to ensure they provide safety for the students. For this to be done, the university has a unit that hold meetings with hostel operators and conducts physical inspections of these hostels [1]. Though physical inspections are conducted, they only focus on the quality of the building and its features like the size of rooms and the equipment available, not the quality of services like daily or monthly maintenance of facilities. Alkandari refers to housing satisfaction as the "degree of contentment experienced by an individual or a family member with regards to the current housing situation" [1]. Student satisfaction with these hostels vary from space, facilities, price, and management. Though all these are factors, hostel management, maintenance, and services are the focus of this study.

Maintenance involves providing services like repairs and replacements to keep a building safe and in good condition. This practice of maintaining and repairing problems improve customer satisfaction and living. However, due to overcrowding, it has become increasingly difficult for management to keep up with these maintenance practices. A survey conducted by Oladiran focused on using a post-occupancy evaluation to analyze the accommodation of student hostels in universities. From research conducted on accommodation availability and user satisfaction, the author found that some issues that reduced student satisfaction were noiselessness, the lack of cleanliness, space, comfortability, ventilation, and water supply [3]. These issues show the need for management to focus on management practices to help in improving student satisfaction. KNUST is not the only tertiary institution facing such problems. Universities in other countries like Nigeria also have similar issues where it has been observed that toilets and water closets, were not functioning in some hostels at the University of Lagos. [3]. These issues reduce user satisfaction and comfortability while living in the hostel.

Many systems and applications have been created through research to handle technical and administrative incidents and issues in offices within organizations. According to Masongsong and Damian [2], "helpdesk is a customer support center in an organization that provides information, administrative and technical support to users, with the view of solving problems that users encountered in the course of using the organizations' resources and facilities." Most organizations use helpdesk systems to improve incident management and maintenance supports with both customers and departments within these organizations. These organizations have technical concerns that exist in the everyday work environment. With the helpdesk system, these concerns are monitored and resolved quickly, in addition to keeping records of these problems. Helpdesk systems vary, with each system having an advantage over the other in terms of pricing and services provided. These systems, albeit focus on the same goal of improving overall customer satisfaction and enhancing organizational growth. Helpdesk systems are mainly implemented for large organizations and are complex; however, for small organizations handling hostel property and services, a simplified application may assist managers to facilitate reporting of issues by students, acknowledgement of the issues and addressing them.

1.4 Aim

This project aims to improve communication and information flow between students and managers in the resolution of issues and challenges in private hostels. Also, it seeks help managers monitor the issues around the hostel more efficiently, quickly address them and improve hostel maintenance. This will enable students to quickly notify hostel managers of any problems with facilities or necessities and ultimately improve students' overall satisfaction with their hostel choice.

Chapter 2: System Requirements

2.1 Scope

The application is limited to both students and hostel managers, to allow for easy and effective communication of issues and challenges in the private hostels. Users may be able to report minor and significant issues around the hostel to the manager. Managers may also view these complaints and address them quickly or request speedily for more details. The requirement analysis conducted provides more context into the design of the system based on responses. The results in Appendix A shows the problems student face in their various hostels, contacting managers on these problems, the duration it takes to solve the problem and their satisfaction with the way problems are resolved.

2.2 Scenarios

Scenario 1

James is a student at Ashesi and lived in one of the hostels off-campus. Since he started living there, he never encountered any problem until he was learning in his room when the fan suddenly stopped working. He could not focus due to the heat in the room. He tried looking for the manager to report the issue but could not find him. He asked his roommate, who showed him the application. He was able to report his problem on the application. After receiving the alert, the caretaker contacted an electrician. His fan was fixed so he could continue learning in his room.

Scenario 2

Kojo is the manager at Ceewus hostel. The hostel owner has tasked him to manage hostel services and facilities alongside other employees. He has many responsibilities, and due to this, the managers' attention is focused on more important tasks first before attending to the other issues raised by students. Moreover, he often forgets specific reports made by students due to his busy schedule and only remembers them when pressure is applied. He needs an efficient way to handle incidents to get them done quickly. He is introduced to the hostel incident application by a student who sees his frustration. He tells the tenants of the hostel about the system, and in no time, they begin to use it. The manager gets reports without having to visit the student's rooms.

Scenario 3

Sarah is a final-year student who has lives at Hosanna. Though she lived in the same room since her first year, she realized that the socket beside her bed had always been faulty. She reported it to the manager in her first year, who assured her that her problem would be fixed. When she resumed school for her second year, she noticed that her socket was still faulty. The manager had a lot of responsibilities to attend to and forgot to inspect the problem. She needs the socket to charge her electronic devices for school, however since her socket is faulty, she has to go to campus to charge and return to the hostel. She downloaded the incident mobile application and lodged a complaint which the caretaker successfully received. The manager was, therefore, able to fix her socket.

2.3 Use Case

This section focuses on the actions of the users when interacting with the application.

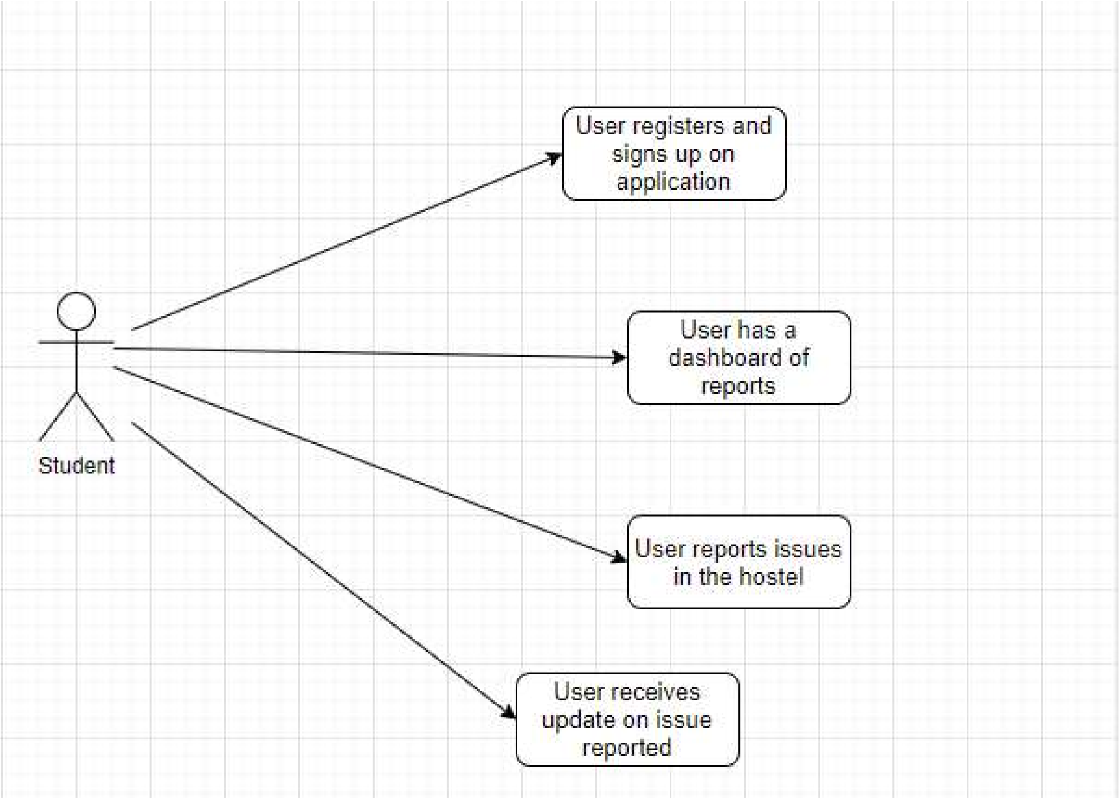

User registers and signs up on application
User has a dashboard of reports
Student
User reports issues in the hostel
User receives update on issue reported

Figure 2.1: Use case (Student)

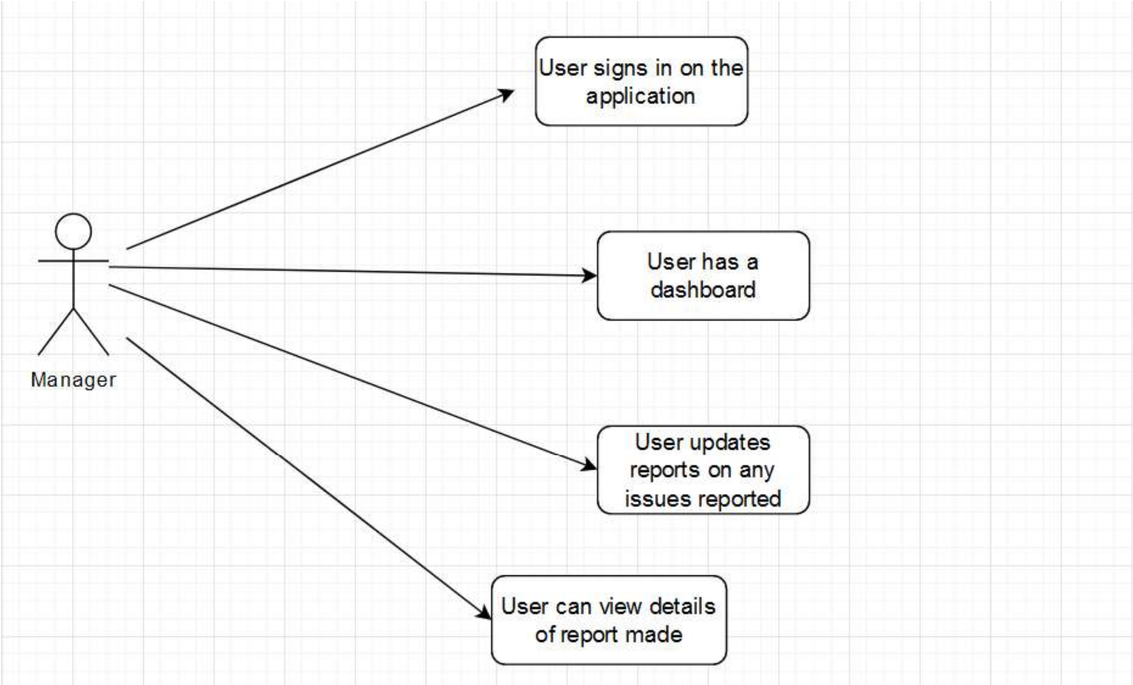

User signs in on the application
User has a dashboard
Manager
User updates reports on any issues reported
User can view details of report made

Figure 2.2: Use case (Manager)

2.4 Design Constraints and Assumption

In designing the software, a few assumptions were made:

- The user has to be connected to the internet to use the application

2.5 Functional Requirements

This section focuses on the functional requirements for the incident application. Since there are two stakeholders involved in the use of this application, the users would be divided into two categories, the student requirements and requirements for the manager.

2.5.1 Student Requirements

- The user should be able to register and sign up.
- The user should be able to send reports to the caretaker.
- The user should be able to take pictures and videos.
- The user would be able to track the issues.

2.5.2 Manager requirements

- The administrator would register this user.
- The user should be able to receive reports of issues through the application on the dashboard.
- The user should be able to view images, and videos of the issues reported.

2.6 Non-Functional Requirements

Availability – The system can be used at any time by both students and managers to send and receive reports.

Usability- The user should be able to use the system easily

Manageability - The system can be monitored and managed easily by the caretaker to keep the application performing at all times.

Chapter 3: System Design and Architecture

3.1 The Design of the Architecture

The proposed solution was developed using the three-layer architecture consisting of the presentation layer, the application layer, and the storage or database layer. The architecture involved using the Model, View, and Controller for easy use and navigation. The view presents an interface to end-users, the model manages the application’s data, and the controller serves as a link between the model and view where any changes made from the view effect a change in the model.

3.1.1 The Presentation Layer

The front-end layer of the application allows users to interact with the system through an interface. The application has multiple interfaces for both students and managers. HTML (HyperText Markup Language) and CSS is used to develop the front-end layer of the application. The interfaces consist of the sign-up and sign-in page, the dashboard for the manager, a page for students to file complaints, a list of reports, and a media page to upload and view images and videos. *Figure 3.1* shows the students interface prototype for the proposed solution while *Figure 3.2.1* and *Figure 3.2.2* shows the interface prototype for managers

Students (*Prototype*)

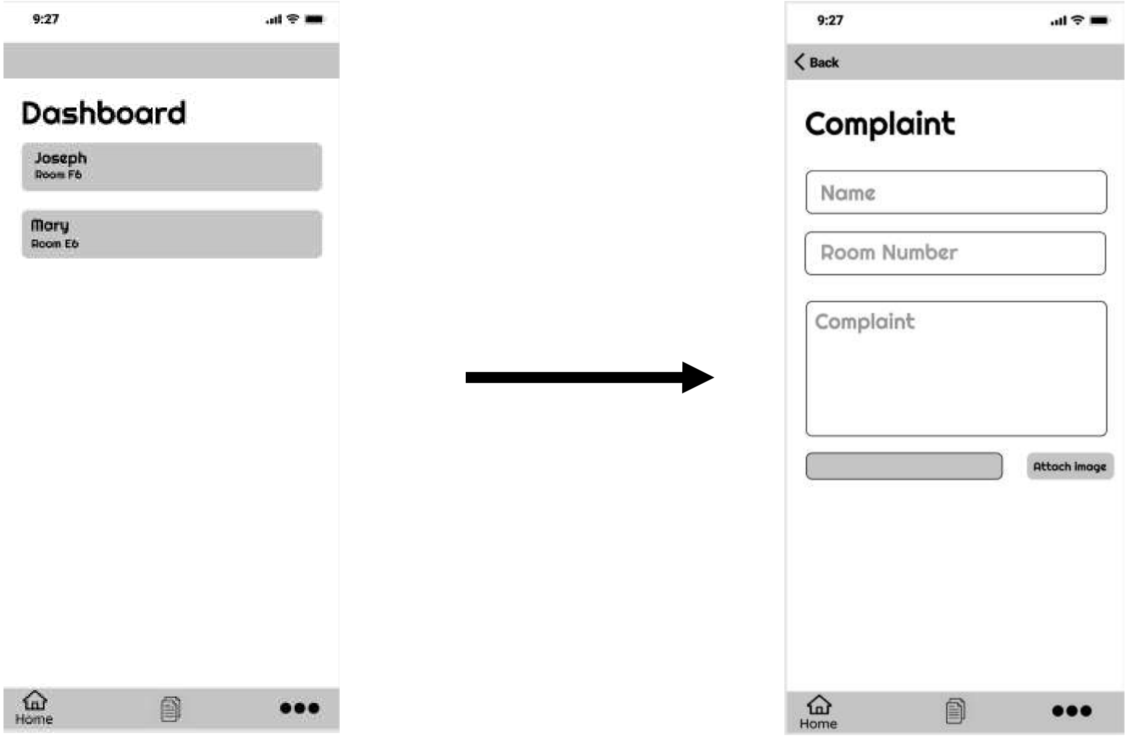


Figure 3.1: Student's Complaint and List Page

Manager (*Prototype*)

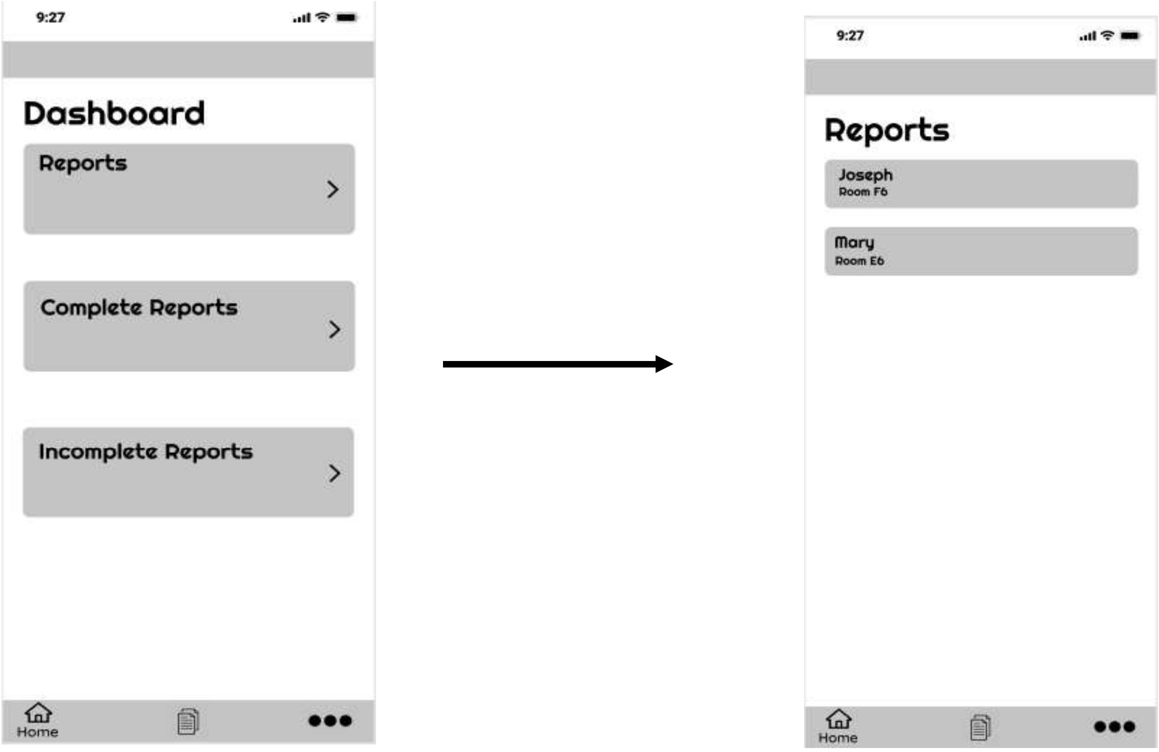
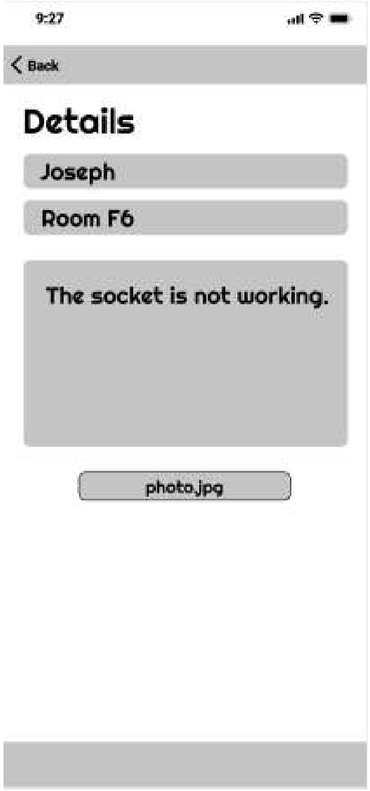


Figure 3.2.1: Manager Dashboard and Report list



Joseph

Room F6

The socket is not working.

Figure 3.2.2: Details Page

3.1.2 The Application Layer

The application layer, also known as the functional process logic, performs functions on the application allowing the user to send their complaint to the manager and upload pictures and videos. JavaScript was used in developing this layer, specifically Angular JS. This JavaScript language is integrated into the Ionic framework due to its core functionalities. This language helped build the back-end of the application in the browser during the testing stages to ensure the application works without any errors. The application developed is hybrid and can be used across multiple platforms.

3.1.3 Database Layer

The database layer is responsible for storing user's data through firebase cloud storage. The database is responsible for keeping both student's and manager's data, including user's login details, images, and reports made by students. The student's data is stored when the student signs up on the application. The student then moves to the complaint page to file a report, and once it is complete, the issue is stored on the database. The database being implemented is a data model NoSQL database from Firebase.

The data is stored in documents that are organized in a collection. Within the collection, each user's report is represented as a document. These documents contain the user's report details. The report includes information on the name of the issue, the student's room number, any details of the problem, the date the problem began, and the status. Images and other large-size files cannot be stored on the firestore database and stored on firebase storage in the firebase console. *Figure 3.3* shows the No-SQL data model.

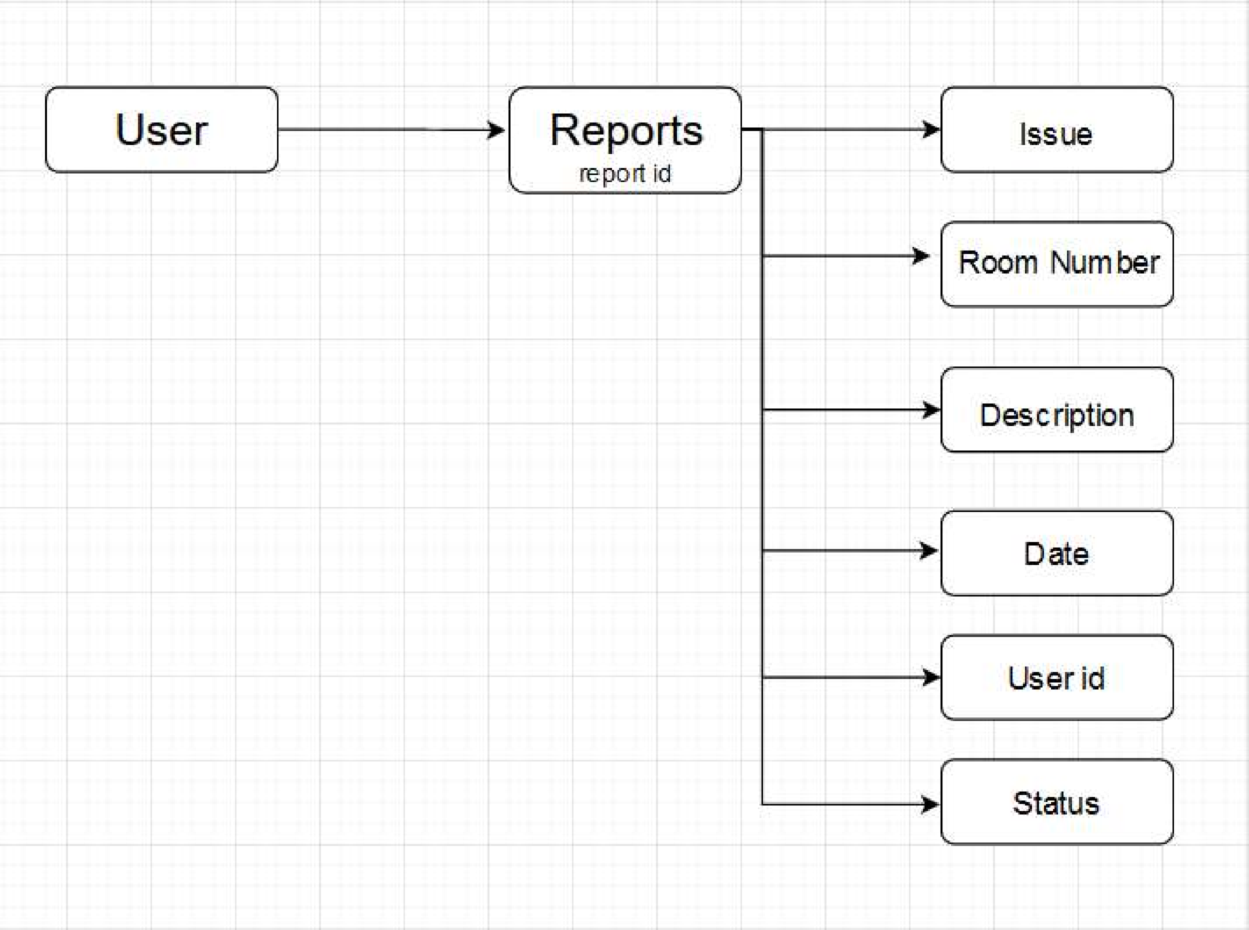


Figure 3.3: No-SQL Data Model (Firebase)

3.2 System Modules

The incident management application comprises two modules: the student module and the manager's module, alongside activities involving both the student and the manager. The application allows students to create accounts with their information. On the other hand, the manager has login details and does not need to register an account. The manager is able to view the reports made by students on the account.

3.3 Activity Diagram

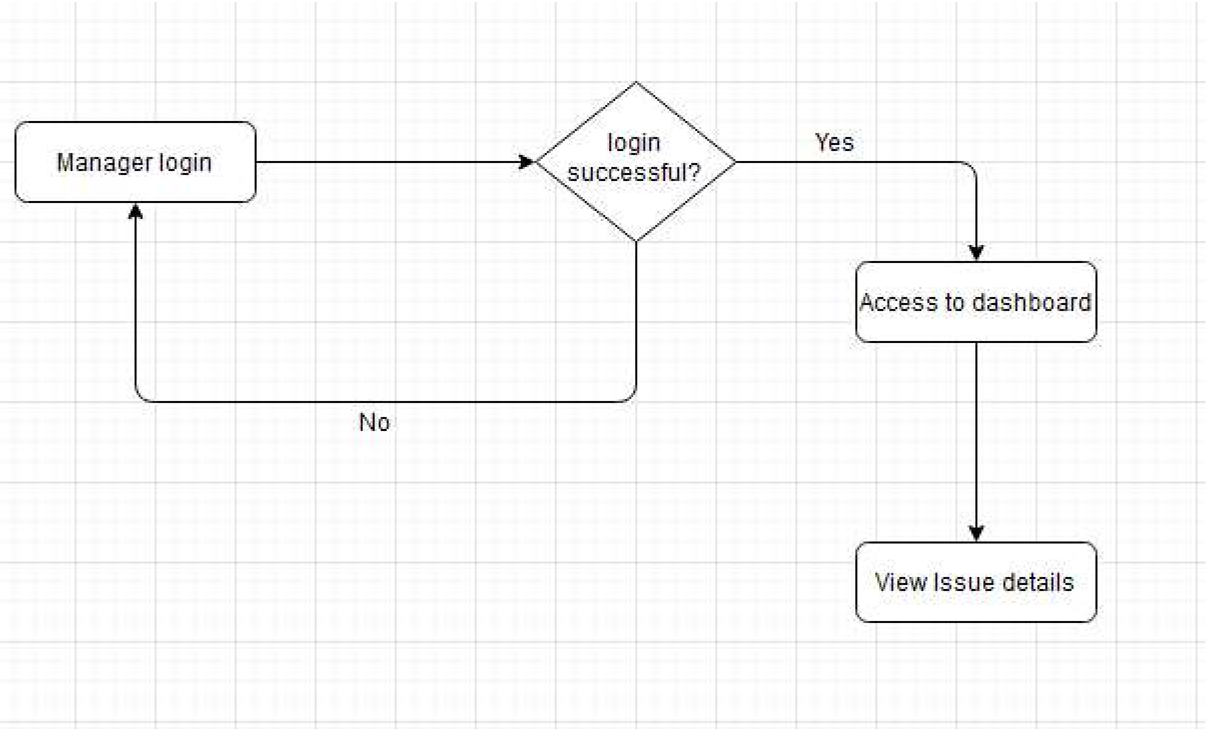


Figure 3.4: Activity Diagram of Manager for viewing reports.

Figure 3.4 shows the activities of the caretaker when using the application. The user creates the account and enters login credentials to access the account. If the user's login is unsuccessful, the user is prompted to input the details again. If the user is successful, the user is redirected to the dashboard to view the complaints made by students. The dashboard contains a general list of reports, reports that have been completed, and incomplete reports that need to be resolved. The user can also view any media files from the database.

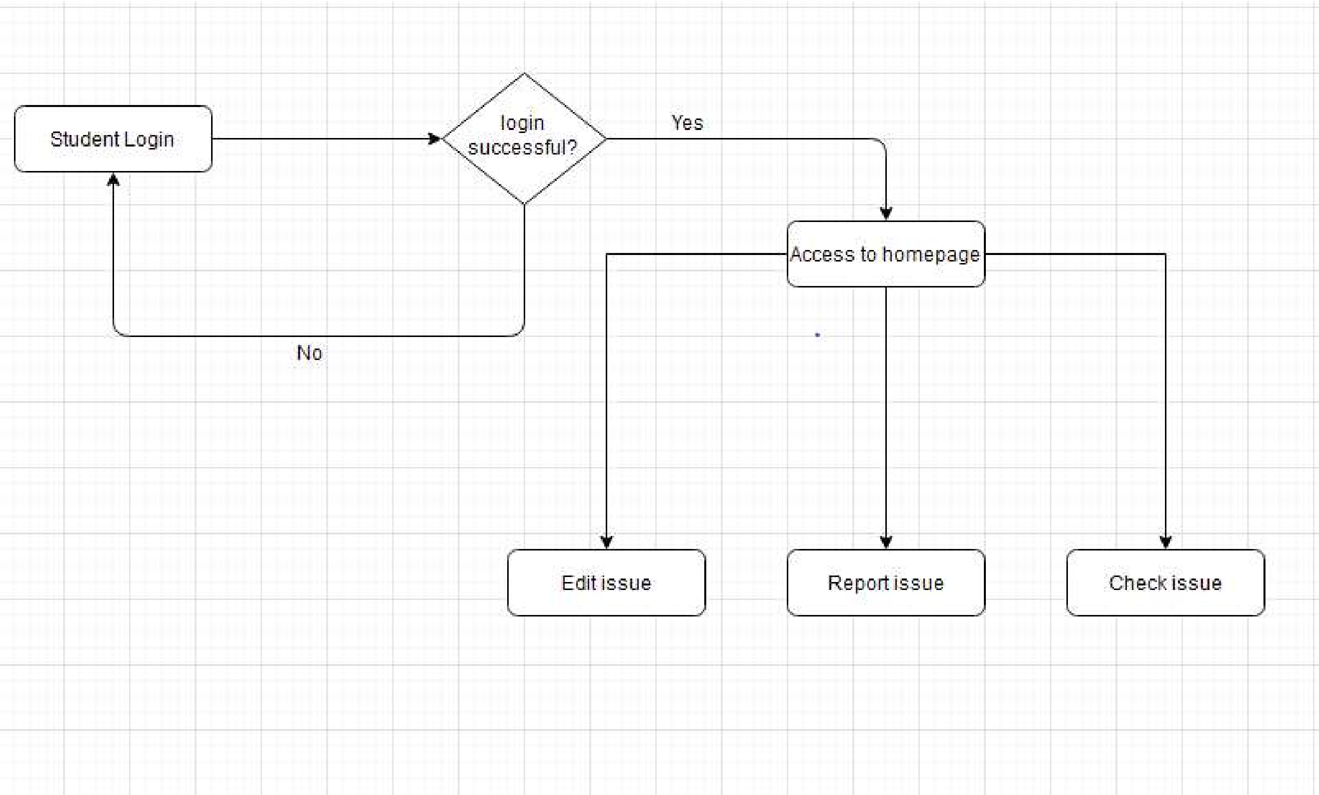


Figure 3.5: Activity Diagram of Student for sending reports

Figure3.5 shows the student's activities when using the application. The user registers by providing the necessary information. Once the information is stored and verified, the user is redirected to the login page to sign in to the account. If the user details are unsuccessful, the user is prompted to try again. If the user's credentials are successful, the user is taken to the home page, which will display the user's filed complaints. The user puts in the name, room number, details of the problem, and the date to file an issue. The status is marked incomplete when the user sends the report. Once the user is done, the report is sent to the manager.

3.4 Languages

3.4.1 Javascript

JavaScript is a programming language used for web and mobile development. JavaScript was used for the application layer, which contains the logic and functions of the application. HTML and CSS was used to handle the presentation layer of the application. The Javascript framework that was used for the proposed solution is Angular JS. This language is integrated into the ionic framework used due to its flexibility and ability to develop applications that can be used across different platforms and in browsing applications. Also, this framework provides various functionalities and libraries that can be used on native platforms.

3.4.2 NoSql

NoSQL is a non-relational database used for the storage and retrieval of data. This database is necessary because the data is stored in the cloud and retrieved easily in real-time. The type of NoSQL database used is document-based, containing the complaint document in a collection. This type of database is used for the firebase database and will be integrated and implemented to provide data to the user in real-time.

3.4.2.1 Firebase

Firebase is a platform that helps in the storage and development of applications. This platform provides tools and services that help in developing mobile applications. Firebase can be used to store data, and for this project, the NoSQL database was integrated, providing access to libraries and methods. Also, the platform offers an authentication service that makes it easier to store and verify user login data once the user’s information is stored. This firebase console is integrated with services like authentication, cloud firestore, and firebase storage, allowing users to access the reports and media files stored.

3.5 Libraries

3.5.1 Ionic

Ionic is an open-source hybrid mobile development platform for mobile developers. This framework was chosen because it allows the application to be used on multiple platforms. Also, it is easy to learn and has inbuilt tools that make application development faster, easier and reduces the time needed to develop the application. For this project, the framework was used for developing and deploying the application.

3.5.2 Cordova

Cordova enables the development of hybrid mobile applications. It helps in accessing native device functions using JavaScript alongside the Ionic framework. The framework is built on Cordova, allowing hybrid applications to access and use native functions and libraries on native platforms.

3.6 Tools

3.6.1 Node JS

Node JS is an open-source platform JavaScript runtime environment that executes JavaScript. This platform is essential in installing the Ionic framework used for developing the proposed solution.

3.6.2 Visual Studio Code

Visual Studio Code is a free code editor for developing applications. It provides development tools and services that make developing an application more flexible. Most frameworks can be integrated with Visual Studio for developing applications. The proposed solution was developed using Ionic on this code editor.

3.6.3 Browser Tools

Browser tools provide an environment for testing and debugging. The application runs on the browser, allowing the developer to view the entire application's entire flow, test the functionalities, and debug any errors. Using the tools provided by the browser enables developers to solve any problems before building and deploying the application. Browsers like Chrome and Mozilla Firefox have browser developer versions that provide these tools for running ionic applications.

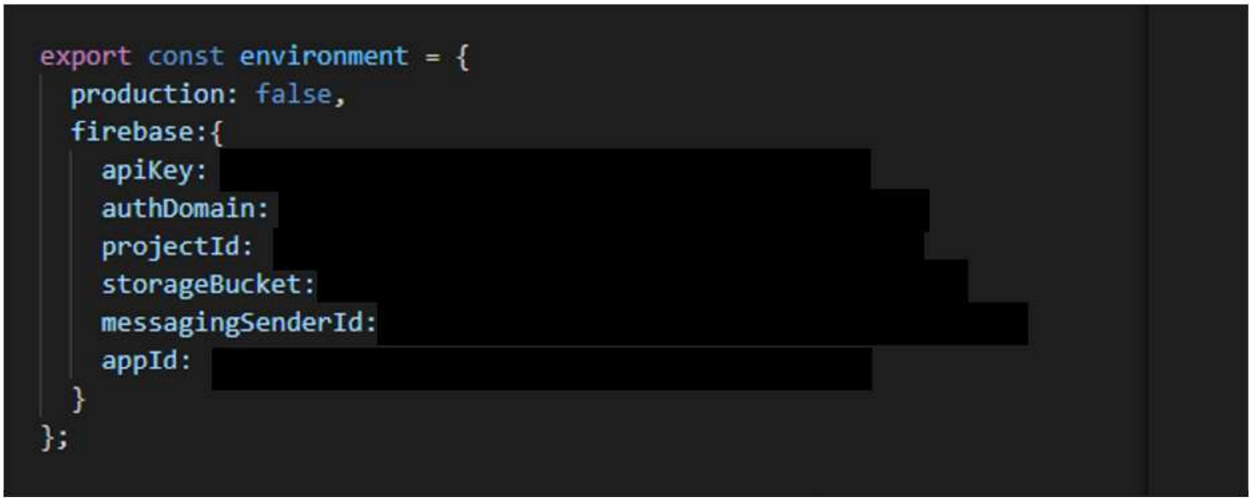
Chapter 4: Implementation

4.1 Implementation Techniques

The development of the application was broken down into sections beginning with the registration and login pages. Next, the student section which contains the form page for making a complaint and displaying the problems reported. The last section was the manager's section which shows the issues reported and allows the user to change the status of the issue and view any media files sent by students. Using a modular approach makes it easier to implement and debug.

4.2 Firebase Integration

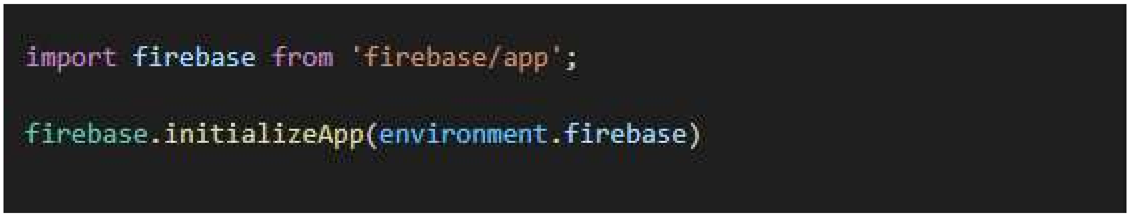
Firebase storage was integrated with the application before authentication and storage services were implemented. In integrating Firebase with the application, Firebase was installed first. A database is created on the firebase console, generating a web API, Application Programming Interface, to connect the application. The API was then copied from the firebase console on the website and is inserted in the *environment.ts* file in *Figure 4.1*.



```
export const environment = {
  production: false,
  firebase:{
    apiKey: [redacted]
    authDomain:
    projectId:
    storageBucket:
    messagingSenderId:
    appId:
  }
};
```

Figure 4.1: environment.ts file

The API was imported and initialized in the *app.module.ts* file in *Figure 4.2*. Initializing Firebase provided access to firebase commands, functions, and services.



```
import firebase from 'firebase/app';

firebase.initializeApp(environment.firebase)
```

Figure 4.2: app.module.ts file

4.3 Implementation Stages

This section focuses on the breakdown of the project and the development process at each stage.

4.3.1 Registration and Login

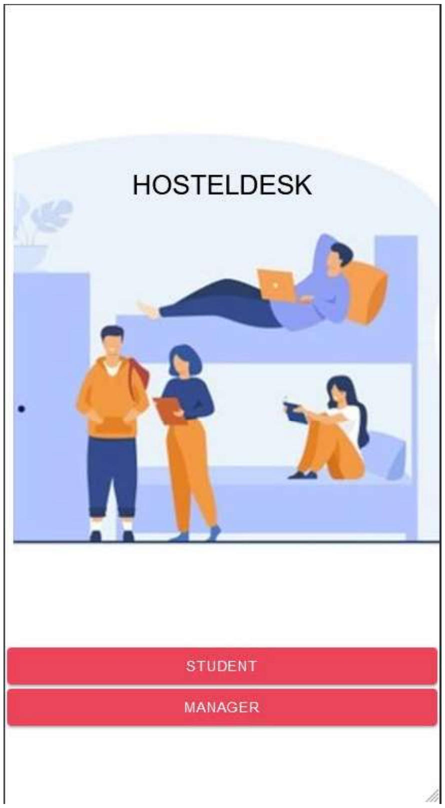
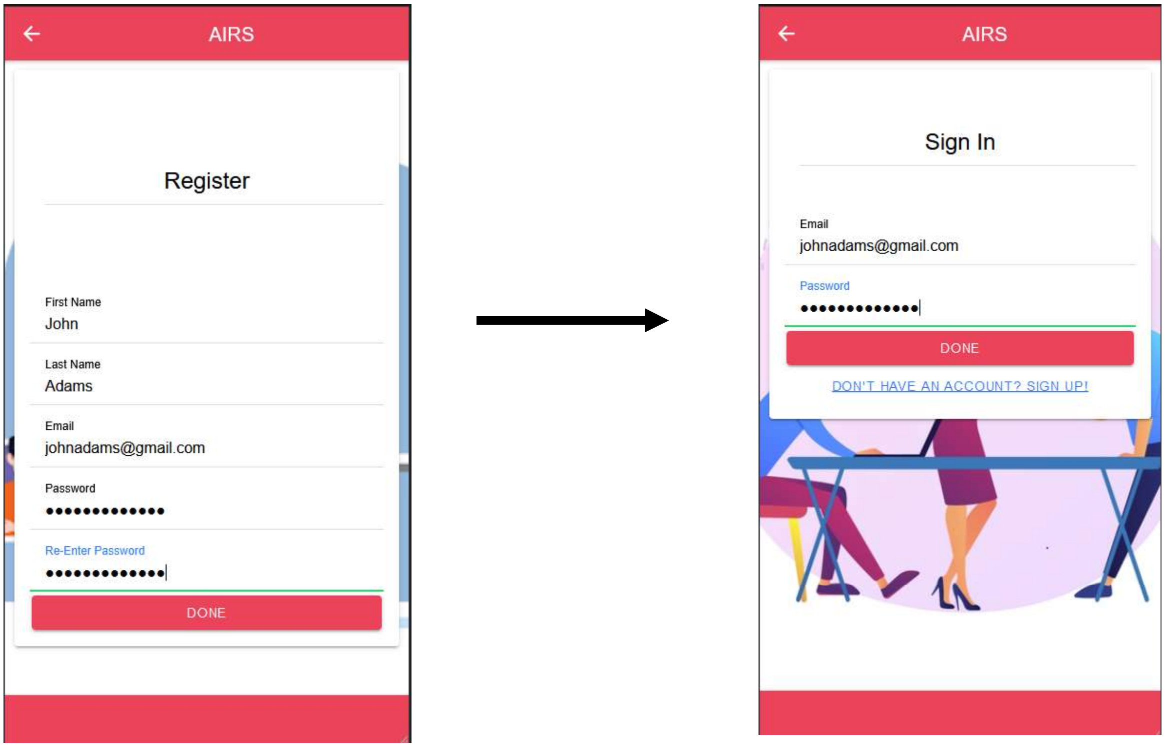


Figure 4.3: Landing page

4.3.1.1 Student Process

Firebase has properties where users can be authenticated through different methods. However, for this application, the user is able to access the account using only the email and password. A method is created with parameters to register the user. Email is the option chosen for both registration and sign-up. Angular Fire Authentication is a class that is imported and is called to access functions and properties to assign and log the user with the email and password. The user's email, password, and other information are created for the registration using the try and catch method, and once done, the user is redirected to the login page in *Figure 4.4.1*. A function was called to create the user's login credentials with the email and password provided when registering on the sign-up page. When the user signs in, the email and password are assigned as the login details and stored on the database allowing the user to access the account.

Once the user provides the correct details, the user can access the account. The catch method is used to get any errors when registering or logging in. When the user provides the wrong information, the error is captured and displayed to the user. The HTML template is created for the user to view and input data. The method was connected to the template and once the user has provided the necessary information, the user's credentials are stored on the database. A loading controller was created to allow a smooth transition and routing into the user's account page. The catch method is implemented to display any errors and return them to the user. These errors prompt the user and alert them on any mistakes made while logging in. *Figure 4.4.2* and *Figure 4.4.3* displays the back-end implementation to store registered details and allow the user to sign in.



First Name
John

Last Name
Adams

Email
johnadams@gmail.com

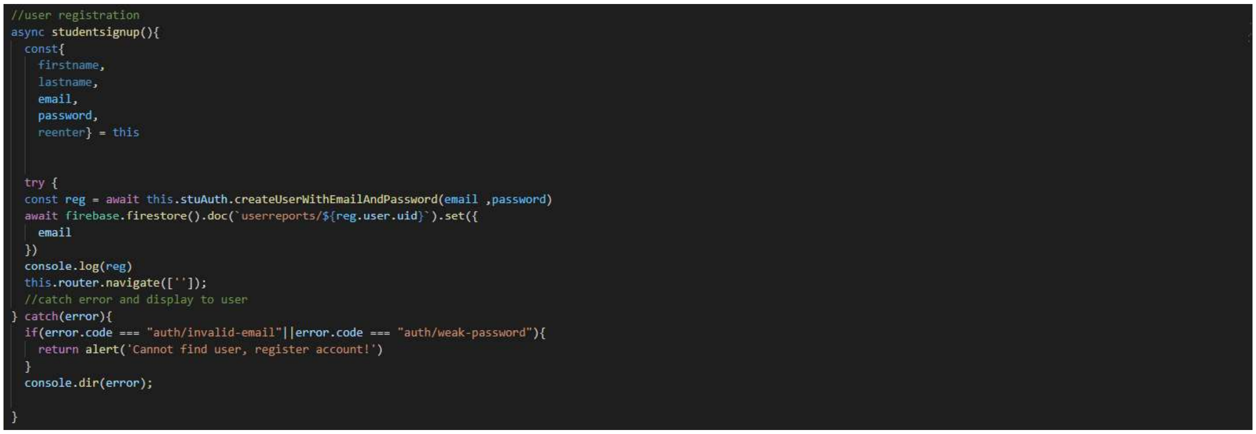
Password
•••••••••••••

Re-Enter Password
•••••••••••••

Email
johnadams@gmail.com

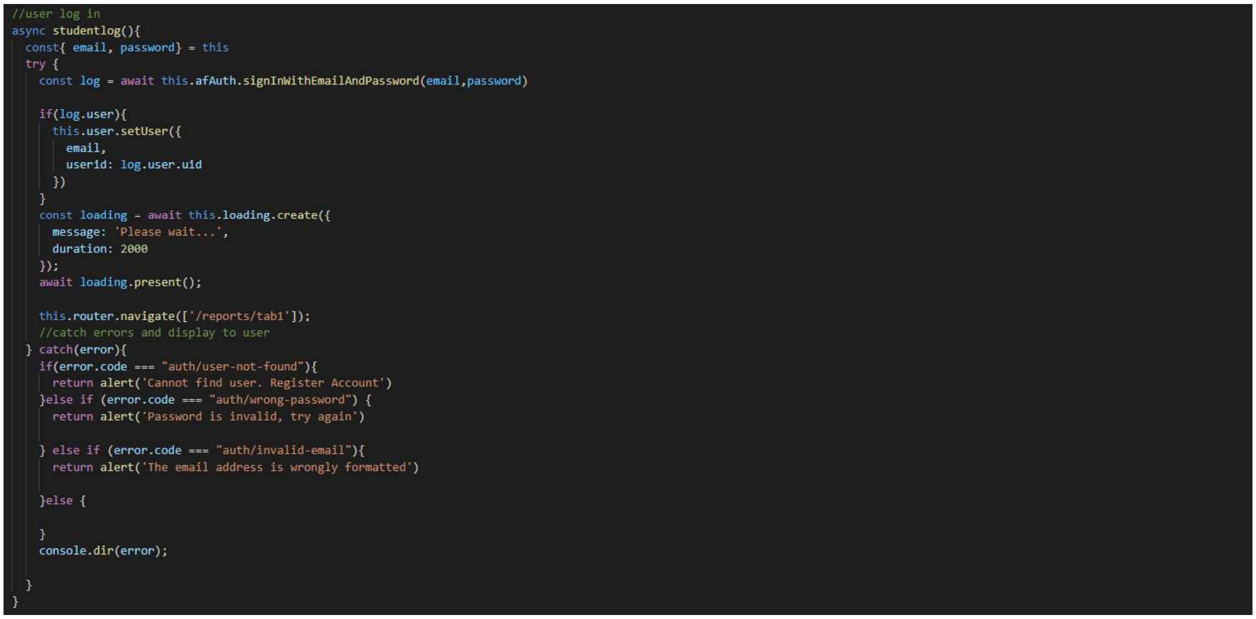
Password
•••••••••••••

Figure 4.4.1: Student registration and log in



```
//user registration
async studentsignup(){
  const{
    firstname,
    lastname,
    email,
    password,
    reenter} = this


  try {
  const reg = await this.stuAuth.createUserWithEmailAndPassword(email ,password)
  await firebase.firestore().doc(`userreports/${reg.user.uid}`).set({
    email
  })
  console.log(reg)
  this.router.navigate(['']);
  //catch error and display to user
} catch(error){
  if(error.code === "auth/invalid-email"||error.code === "auth/weak-password"){
    return alert('Cannot find user, register account!')
  }
  console.dir(error);

}
```

Figure 4.4.2: Back-end for Student Registration

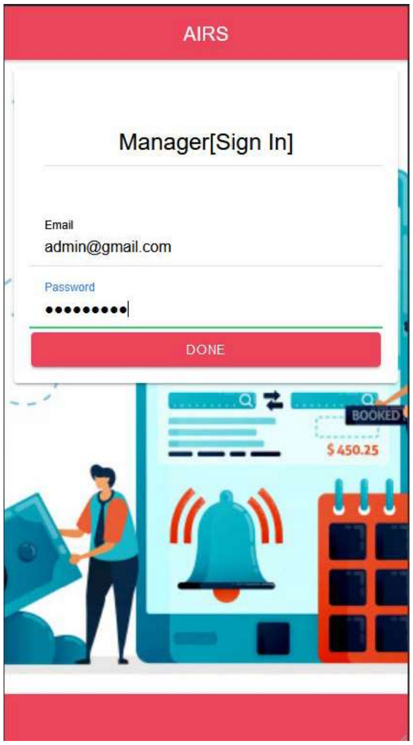


```
//user log in
async studentlog(){
  const{ email, password} = this
  try {
    const log = await this.afAuth.signInWithEmailAndPassword(email,password)

    if(log.user){
      this.user.setUser({
        email,
        userid: log.user.uid
      })
    }
    const loading = await this.loading.create({
      message: 'Please wait...',
      duration: 2000
    });
    await loading.present();

    this.router.navigate(['/reports/tab1']);
    //catch errors and display to user
  } catch(error){
    if(error.code === "auth/user-not-found"){
      return alert('Cannot find user. Register Account')
    }else if (error.code === "auth/wrong-password") {
      return alert('Password is invalid, try again')

    } else if (error.code === "auth/invalid-email"){
      return alert('The email address is wrongly formatted')

    }else {

    }
    console.dir(error);

  }
}
```

Figure 4.4.3: Back-end for Student Login

4.3.1.2 Manager Process

The manager, on the other hand, does not register. These users' details are already taken since the manager handles the reports. The manager only logs in to view the reports made by the students. The back end uses the same procedure as the student login as the manager's details are already stored on the database. This is shown in *Figure 4.4.*



AIRS

Manager[Sign In]

Email
admin@gmail.com

Password
•••••••••

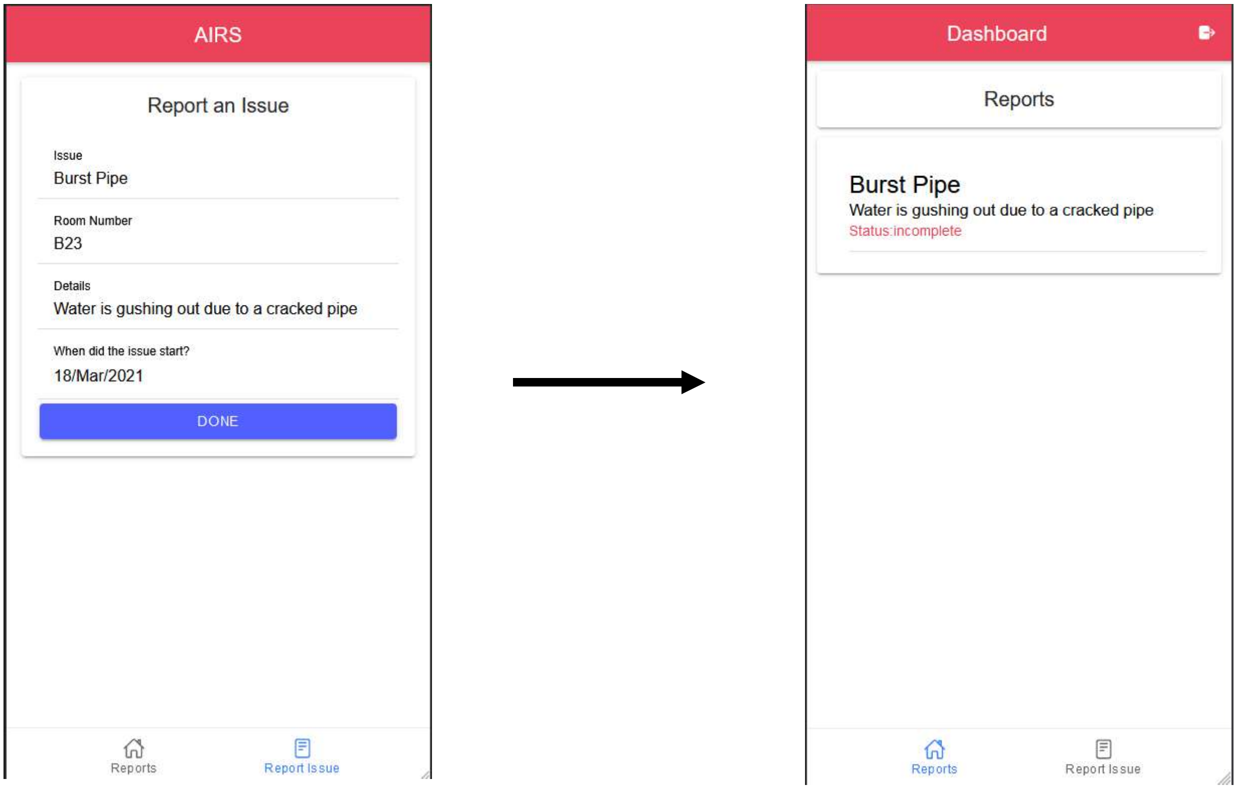
Figure 4.4: Manager Sign In

4.3.2 User Report and List

In developing the user’s account, a tab page is created to make a report and view the list of reports. In creating the issue report page, a form is created in the HTML section. The page displays the issue name, the student room number, the details, and the date the issue began. In the *ts* file, firebase firestore is imported. Parameters are created for the form containing the problem, details, room number, date, and status. These parameters are also created on the firebase cloud store database to be stored as fields in a document. Using the firestore function in Firebase, a function called *collection* holds the database name created on the firebase console. Also, the add function is implemented and adds and stores the report on the database.

The method is connected with HTML to add the report to the database. A toast notification is sent to the user once the report has been sent to the database successfully. A catch method is created to alert the user of any problem when storing the information.

The home tab shows the list of reports made by the user. A method is created in the *ts* file to using the get method from Firebase to retrieve the documents stored on the database and take a snapshot of the user's report. In the HTML file, **ngFor* is used to iterate over an array holding the data to display the list of reports made by the user, taking the information from the database. *Figure 4.5.1* are the front-end diagrams for adding and displaying the report and *Figure 4.5.2* and *Figure 4.5.3* shows the back-end implementation for adding the report to the database and displaying the information.



Issue
Burst Pipe

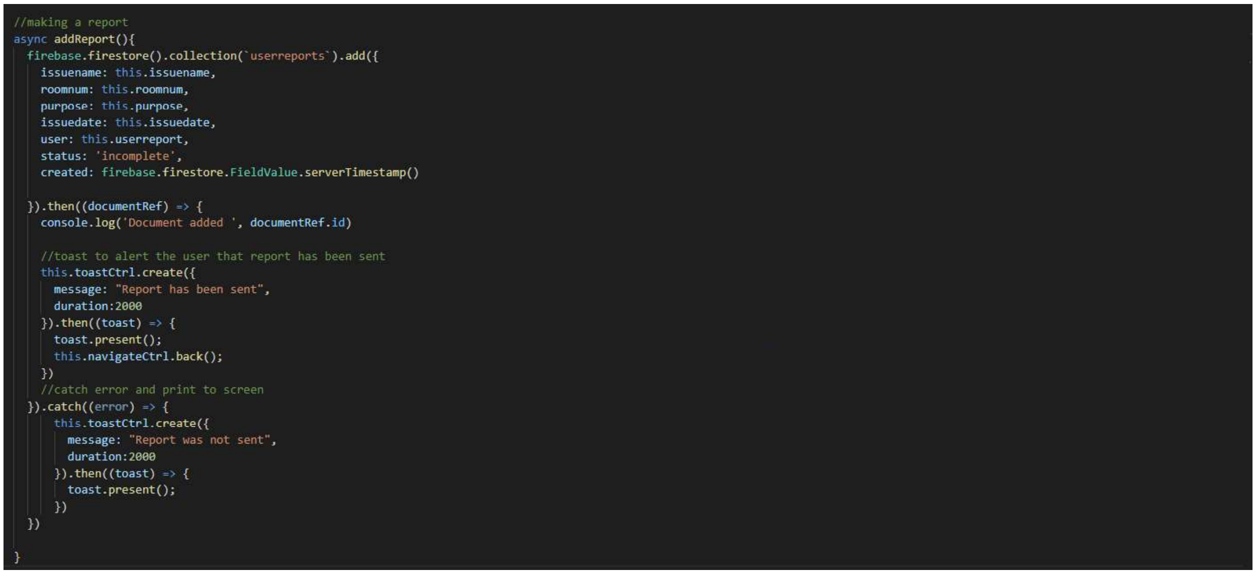
Room Number
B23

Details
Water is gushing out due to a cracked pipe

When did the issue start?
18/Mar/2021

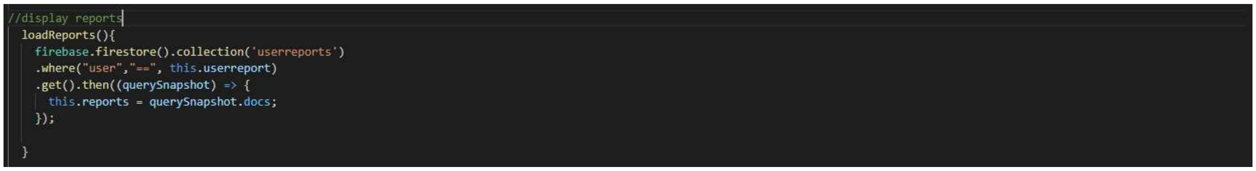
Burst Pipe
Water is gushing out due to a cracked pipe
Status:incomplete

Figure 4.5.1: Issuing and report



```
//making a report
async addReport(){
  firebase.firestore().collection(`userreports`).add({
    issuename: this.issuename,
    roomnum: this.roomnum,
    purpose: this.purpose,
    issuedate: this.issuedate,
    user: this.userreport,
    status: 'incomplete',
    created: firebase.firestore.FieldValue.serverTimestamp()

  }).then((documentRef) => {
    console.log('Document added ', documentRef.id)

    //toast to alert the user that report has been sent
    this.toastCtrl.create({
      message: "Report has been sent",
      duration:2000
    }).then((toast) => {
      toast.present();
      this.navigateCtrl.back();
    })
    //catch error and print to screen
  }).catch((error) => {
      this.toastCtrl.create({
        message: "Report was not sent",
        duration:2000
      }).then((toast) => {
        toast.present();
      })
  })

}
```

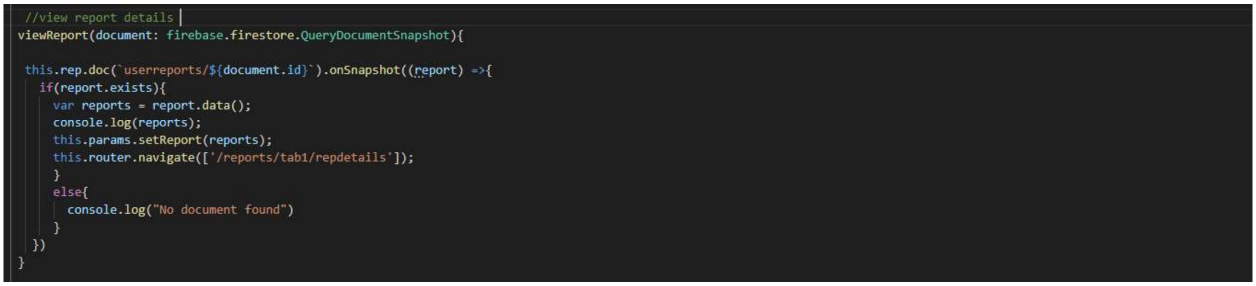
Figure 4.5.2: Back-end to add report



```
//display reports
  loadReports(){
    firebase.firestore().collection('userreports')
    .where("user","==", this.userreport)
    .get().then((querySnapshot) => {
      this.reports = querySnapshot.docs;
    });

  }
```

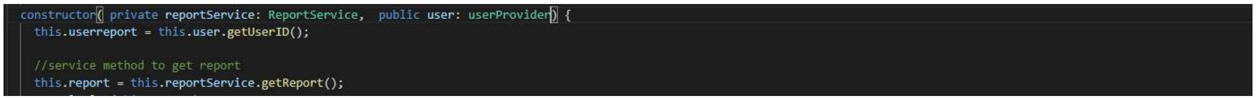
Figure 4.5.3: Back-end to view report list

To view the details of a specific report, the user clicks on the report from the list. The document is retrieved from the database, and a snapshot of the details is stored as an object. A service file is created to hold the object from the database using a set function and get the document the get function is called. The service file is imported, and a method is created to access the database. An *if* statement is used to ensure the document exists, and if valid, the details are retrieved from the database. The data is routed to the next page to display the details by importing the service file and using the get method from the file. If the document does not exist, the error is printed to the browser console stating that no document exists. *Figure 4.5.5* represents the back-end for retrieving the report stored in the database and placing it in a set method while *Figure 4.5.6* uses a get method to get the report stored to display the details of the report.



```
 //view report details
viewReport(document: firebase.firestore.QueryDocumentSnapshot){

 this.rep.doc(`userreports/${document.id}`).onSnapshot((report) =>{
   if(report.exists){
     var reports = report.data();
     console.log(reports);
     this.params.setReport(reports);
     this.router.navigate(['/reports/tab1/repdetails']);
     }
     else{
       console.log("No document found")
     }
  })
}
```

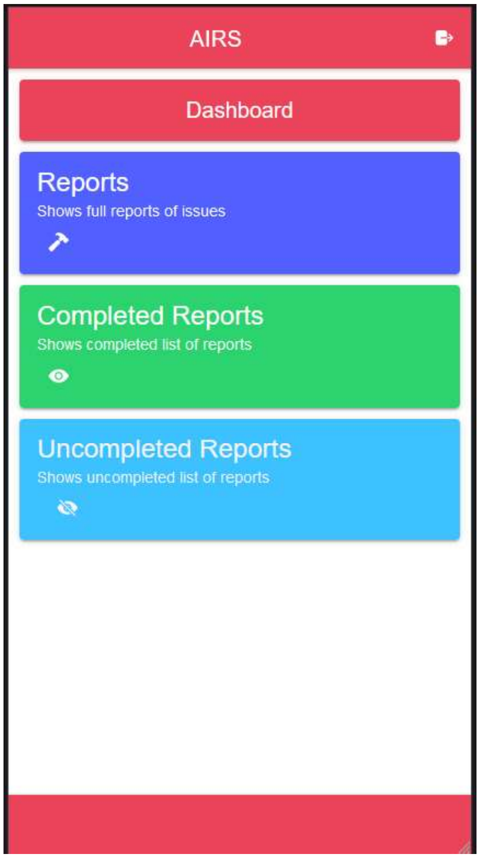
Figure 4.5.5: Back-end to retrieve report details



```
 constructor( private reportService: ReportService,  public user: userProvider) {
   this.userreport = this.user.getUserID();

   //service method to get report
   this.report = this.reportService.getReport();
```

Figure 4.5.6: Back-end to display report details

The manager’s account has a dashboard displaying the list of reports in sections to keep track of the reports made by students. The reports are loaded in each section showing the reports are complete and ones are not completed. The manager handles the changes once the issues have been resolved. This is shown in *Figure 4.6.*



AIRS

Dashboard

Reports

Shows full reports of issues

Completed Reports

Shows completed list of reports

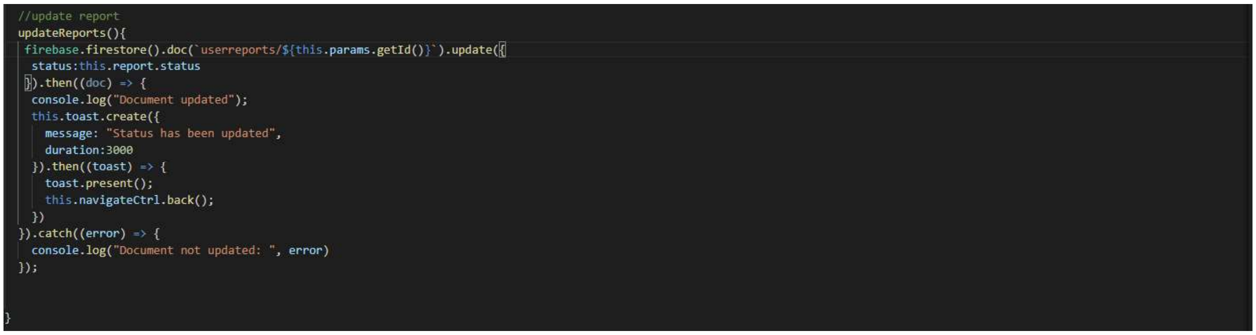
Uncompleted Reports

Shows uncompleted list of reports

Figure 4.6: Manager Dashboard

4.3.3 Update Report

To update students on the issue, the manager views the details and can change the status of the problem reported. The document id is retrieved, and the update function is called and updates only the report's status. The information on the document remains the same, except for the status. Once the manager updates the document, a toast notification alerts the user that the change was successful. The changes are then reflected on the database. Also, the completed report is removed from the list of issues that are not complete. *Figure 4.7* shows the back-end code for updating the status of the report.



```
//update report
updateReports(){
 firebase.firestore().doc(`userreports/${this.params.getId()}`).update({
  status:this.report.status
 }).then((doc) => {
  console.log("Document updated");
  this.toast.create({
    message: "Status has been updated",
    duration:3000
  }).then((toast) => {
    toast.present();
    this.navigateCtrl.back();
  })
}).catch((error) => {
  console.log("Document not updated: ", error)
});


}
```

Figure 4.7: Back-end to update status

4.3.4 Media Upload

The media page allows the user to access the camera and store it on firebase storage in the firebase database console. Developing the media page involves accessing the camera, storing and displaying the media file to the user. Native functions like the camera, image picker, streaming media, photo viewer, and file are imported. These native functions can run once the application is deployed and launched on a native operating system. An action sheet is created and allows the user to either take a photo or video of the issue. Any media captured is stored as files in a folder assigned.

The file method has a parameter that is used in creating a file path. The file contains the name and the extension of the media used. Methods are created to display and delete the media file with a parameter to display and delete individual files. If there is a media file name and extension, the media can be viewed or streamed. The file stored is then uploaded to firebase cloud storage which handles large media files. A blob containing a buffer of the media file data is created to upload any image or video to the cloud. Angular Firebase Storage is imported, the function *upload* allows the media file data to be uploaded and show the upload progress. A toast notification is displayed to the user once the file has been uploaded. The file is stored on Firebase, where the manager is able to access the file. Figure 4.8 portrays the media page where images are captured and the media button to access the media page. The back-end in *Figure 4.8.2* and *Figure 4.8.3* shows the method for capturing images and videos and uploading the image or video file to firebase cloud storage.

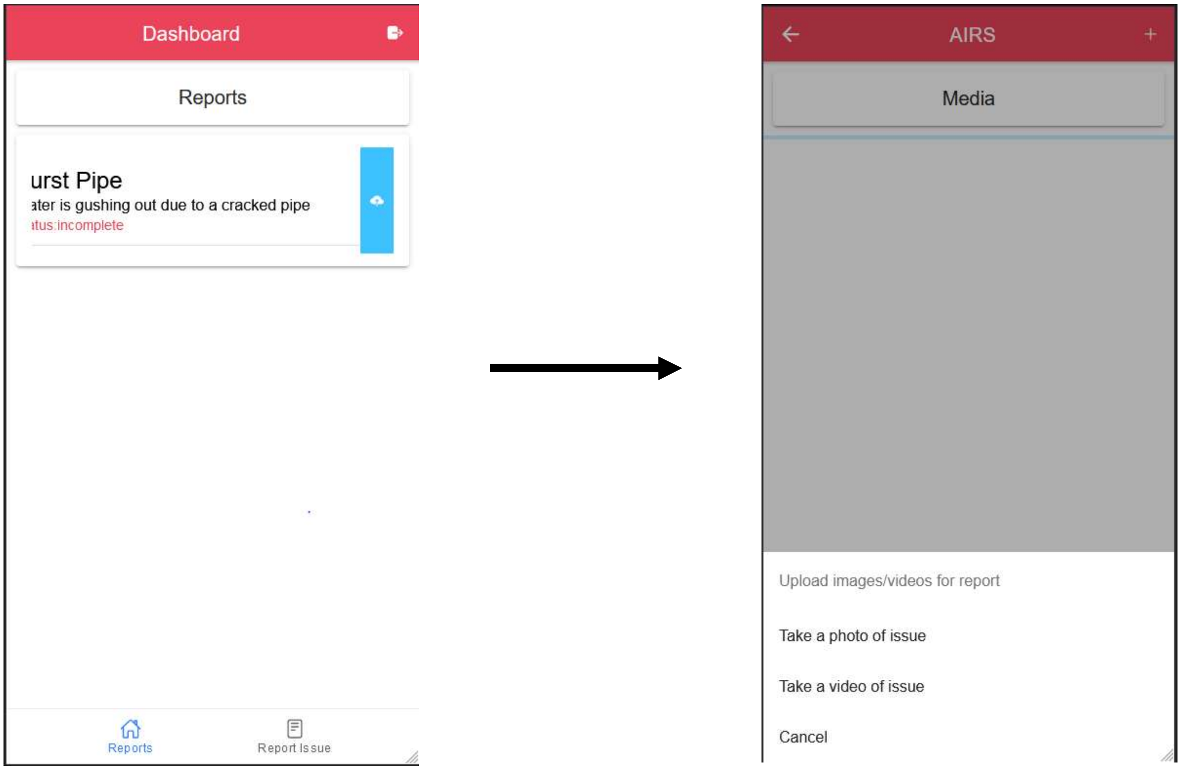
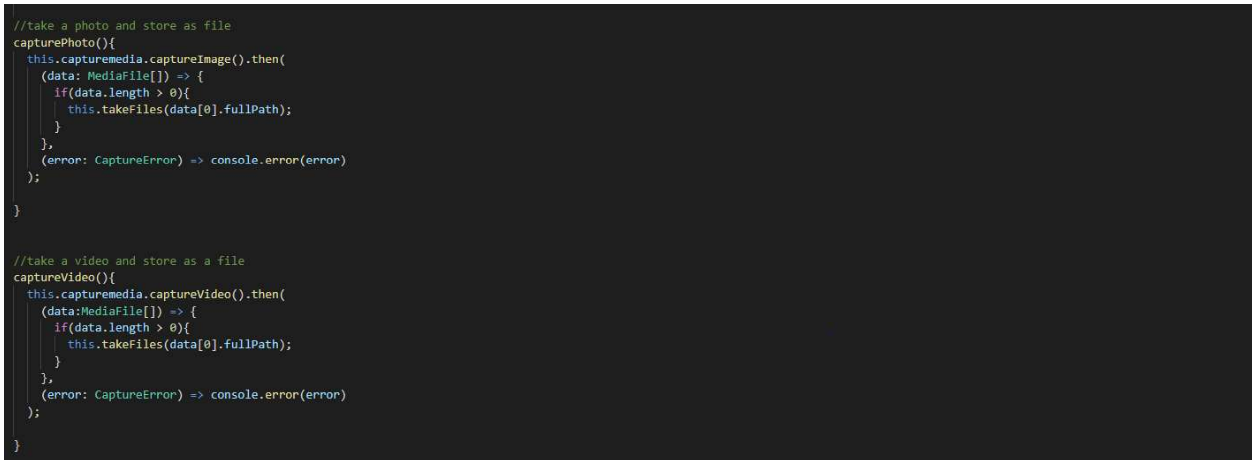
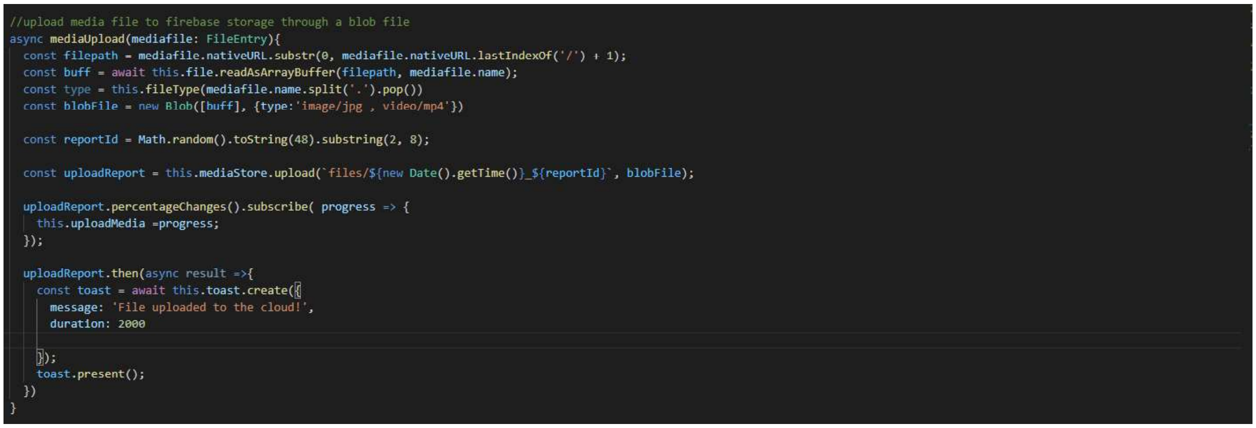


Figure 4.8.1: Media page



```
//take a photo and store as file
capturePhoto(){
  this.capturemedia.captureImage().then(
    (data: MediaFile[]) => {
      if(data.length > 0){
        this.takeFiles(data[0].fullPath);
      }
    },
    (error: CaptureError) => console.error(error)
  );

}


//take a video and store as a file
captureVideo(){
  this.capturemedia.captureVideo().then(
    (data:MediaFile[]) => {
      if(data.length > 0){
        this.takeFiles(data[0].fullPath);
      }
    },
    (error: CaptureError) => console.error(error)
  );

}
```

Figure 4.8.2: Back-end to capture Media

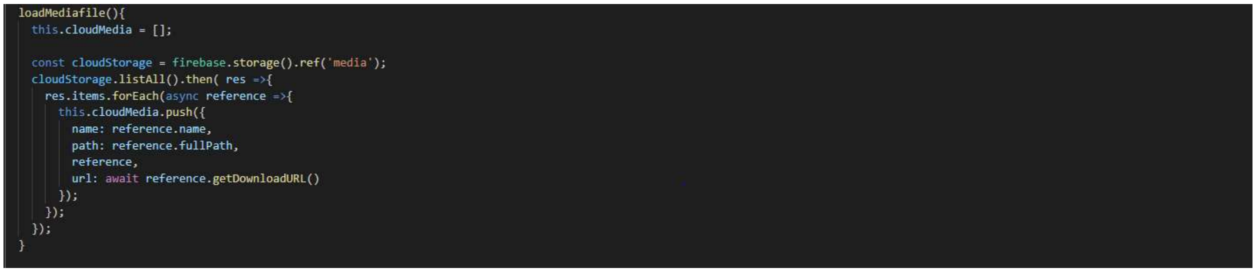


```
//upload media file to firebase storage through a blob file
async mediaUpload(mediafile: FileEntry){
  const filepath = mediafile.nativeURL.substr(0, mediafile.nativeURL.lastIndexOf('/') + 1);
  const buff = await this.file.readAsArrayBuffer(filepath, mediafile.name);
  const type = this.fileType(mediafile.name.split('.').pop())
  const blobFile = new Blob([buff], {type:'image/jpg , video/mp4'})

  const reportId = Math.random().toString(48).substring(2, 8);

  const uploadReport = this.mediaStore.upload(`files/${new Date().getTime()}_${reportId}`, blobFile);

  uploadReport.percentageChanges().subscribe( progress => {
    this.uploadMedia =progress;
  });

  uploadReport.then(async result =>{
    const toast = await this.toast.create({
      message: 'File uploaded to the cloud!',
      duration: 2000

    });
    toast.present();
  })
}
```

Figure 4.8.3: Back-end for Cloud Upload

4.3.5 Media View

Once the media file is uploaded to firebase storage, a method is created in the manager section to load the image files from the database and display them. Firebase is imported to get access to the database. The firebase storage function is called to reference the name given to the firebase storage folder. The name, path, reference, and URL are pushed using an async pipe. The image is viewed through a URL on the browser by importing InAppBrowser and generating a URL. A function is also created to delete the media file as an option for the manager. *Figure 4.9* shows the back-end for viewing any image or video sent to firebase cloud storage.



```
loadMediafile(){
  this.cloudMedia = [];

  const cloudStorage = firebase.storage().ref('media');
  cloudStorage.listAll().then( res =>{
    res.items.forEach(async reference =>{
      this.cloudMedia.push({
        name: reference.name,
        path: reference.fullPath,
        reference,
        url: await reference.getDownloadURL()
      });
    });
  });
}
```

Figure 4.9: Back-end to display media List

4.4 Implementation Challenges

Though the development of most components of the application went smoothly, there were a few challenges that delayed the programming. Storing and displaying any media captured was a challenge since firebase firestore could not store large files like images and videos. This challenge made it difficult to store both the report and any photo or video captured under a single document on the database. However, Firebase also has an additional service known as firebase storage to handle large files so the user can store and view them.

Moreover, routing the stored document and displaying its details was also a challenge. The details show as an object but can be seen on the browser console. Though the document details are captured, they do not display when routed to the details page. However, by creating a service file and using the get and set method, the information is held in a set method and retrieve it using a get method. However, the document was transferred and displayed.

Chapter 5: Testing and Outcome

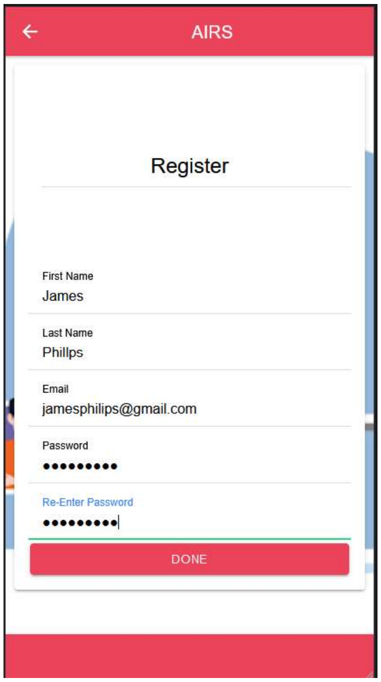
5.1 Functionality Testing

Components of the application are tested to guarantee its functionality and responsiveness when running the application. Testing the major features ensures that the application works well in terms of specifications.

5.1.1 Components

User Registration and Login

User registration is required to add the user's data to the database to log in to the account. The user's email and password are assigned as login details. The user's information has been registered and stored to test this component. The assigned login details should be displayed on the firebase database console. *Figure 5.1* shows the registration details of a new user.



First Name
James

Last Name
Phillps

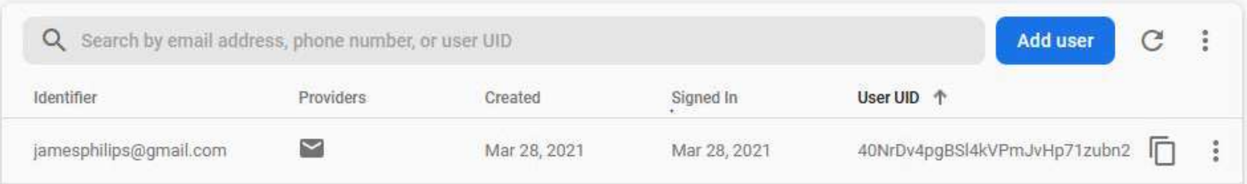
Email
jamesphilips@gmail.com

Password
•••••••••

Re-Enter Password
•••••••••

Figure 5.1: Registration Page

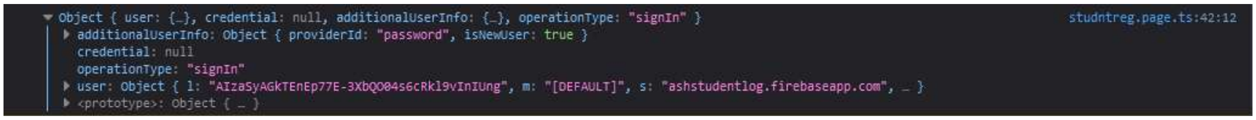
Once the form is sent, the user's email and password are assigned as the login parameters for the user to access the account. The user's email is stored on the Firebase console since the provider option for logging in was email and password. The user is provided with an automatically generated id as well. *Figure 5.2* displays the user's authentication details on the console.



Identifier	Providers	Created	Signed In	User UID ↑
jamesphilips@gmail.com	✉	Mar 28, 2021	Mar 28, 2021	40NrDv4pgBSl4kVPmJvHp71zubn2

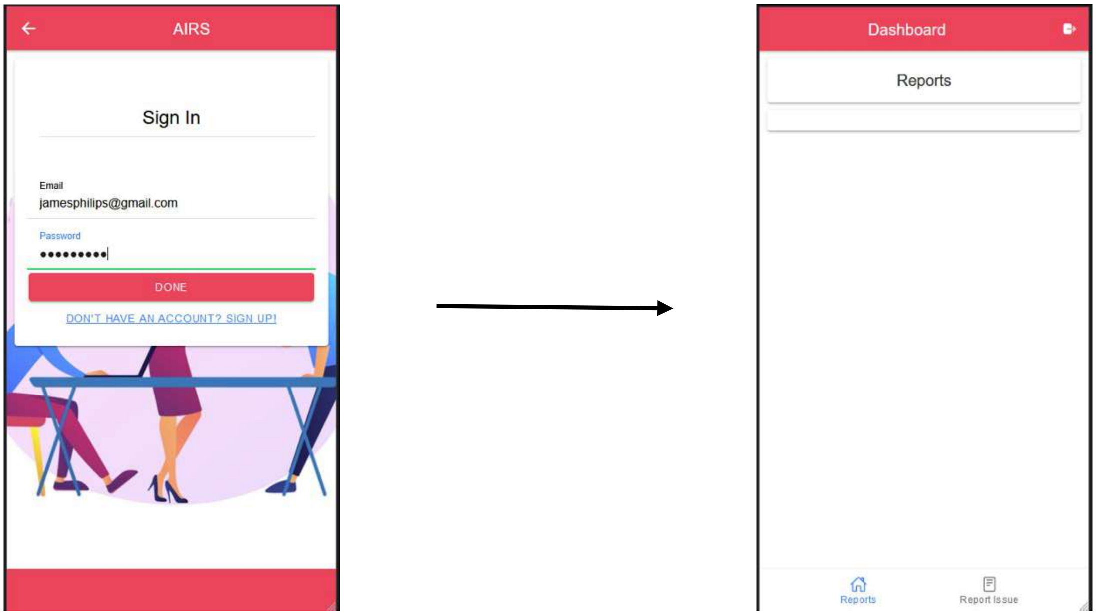
Figure 5.2: Authentication Details on Firebase

Since the user's email and password are stored on the console, the user is granted access to the account with correct details to match the ones stored on the database. *Figure 5.4* shows the user logging in and accessing the account. The browser console shown in *Figure 5.3* displays the object which contains the user's details and has been retrieved from the firebase console. It also shows the provider id being used and a boolean to indicate the user has been recognized and verified.



```
▼ Object { user: {…}, credential: null, additionalUserInfo: {…}, operationType: "signIn" }        studntreg.page.ts:42:12
  ▶ additionalUserInfo: Object { providerId: "password", isNewUser: true }
    credential: null
    operationType: "signIn"
  ▶ user: Object { l: "AIzaSyAGkTEnEp77E-3XbQO84s6cRkl9vInIUng", m: "[DEFAULT]", s: "ashstudentlog.firebaseapp.com", … }
  ▶ <prototype>: Object { … }
```

Figure 5.3: User Object in the browser console

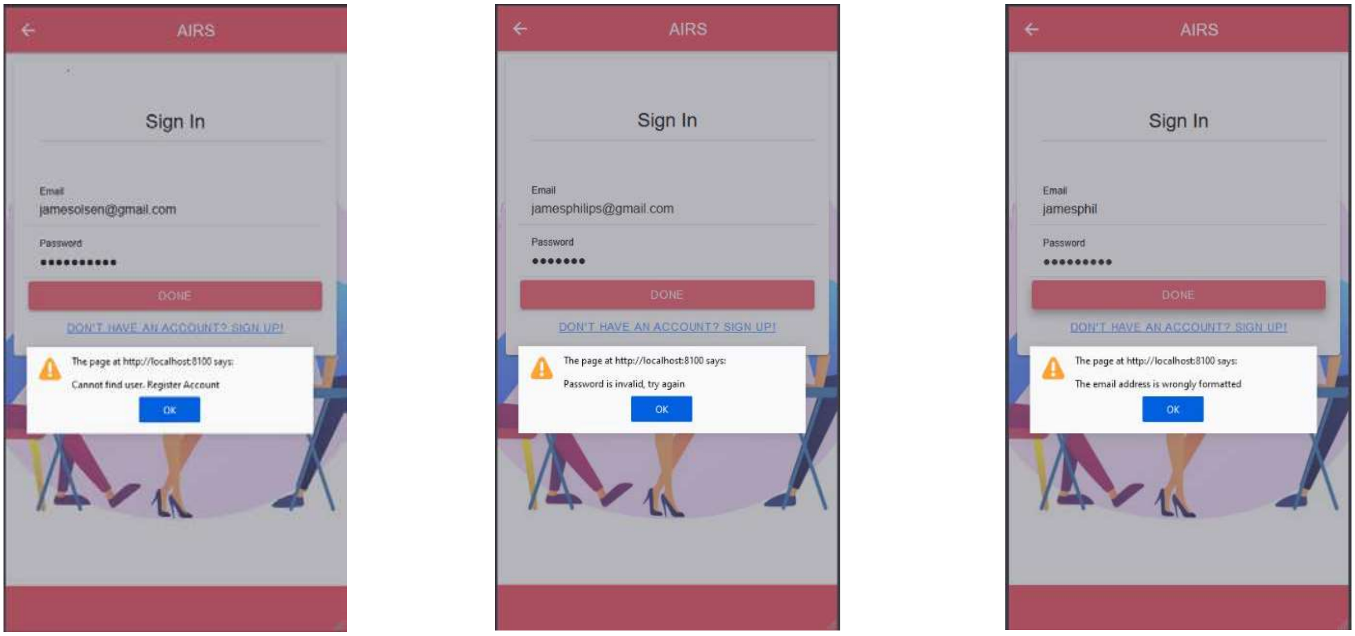


Email
jamesphilips@gmail.com

Password
•••••••••

Figure 5.4: Test to access user account

Also, the user should be alerted if the password or email used is incorrect. If the user's details are wrong, the issue should be displayed to the user. The error should prompt the user to know the exact problem and provide the correct information. If the user also tries to sign in with unregistered details, the user should be prompted, and the error should be displayed to the user. These results in *Figure 5.5* show that the user registration and authentication function correctly and show the critical errors.



Email
jamesolsen@gmail.com
Password
••••••••••

The page at http://localhost:8100 says:
Cannot find user. Register Account

Email
jamesphilips@gmail.com
Password
•••••••

The page at http://localhost:8100 says:
Password is invalid, try again

Email
jamesphil
Password
•••••••••

The page at http://localhost:8100 says:
The email address is wrongly formatted

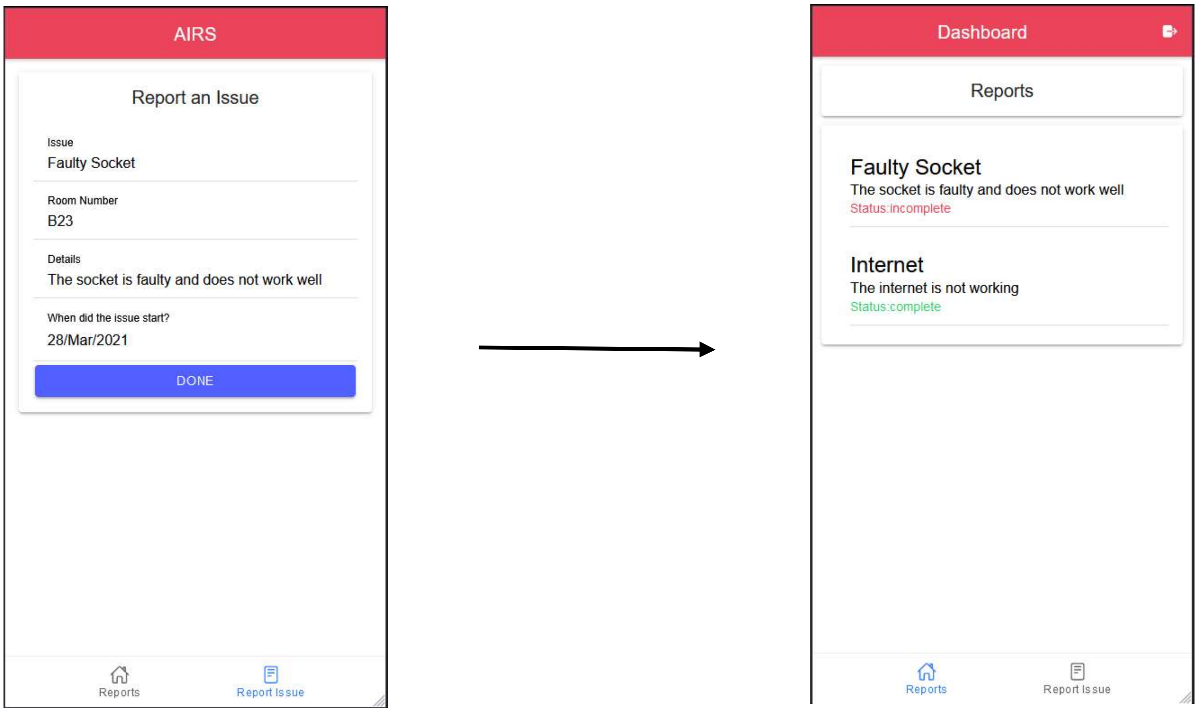
Figure 5.5: User Prompts for incorrect details

Adding and Displaying Reports

This component allows the user to make reports. These reports are stored on the firebase database, and a list of reports are displayed to the user. Since there are two stakeholders, the students can only view the reports they made, while the manager can view all the reports made by students.

Student Report

A report is made, and once the report is done, the information is sent to the database. The report is then displayed on the account, providing the user with a live update of the reports made. The report should also appear on the firebase database with all the information that was added.



Report an Issue

Issue
Faulty Socket

Room Number
B23

Details
The socket is faulty and does not work well

When did the issue start?
28/Mar/2021

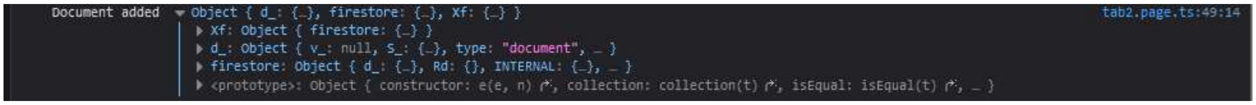
DONE

Reports

Faulty Socket
The socket is faulty and does not work well
Status:incomplete

Internet
The internet is not working
Status:complete

Figure 5.6: Test of student report



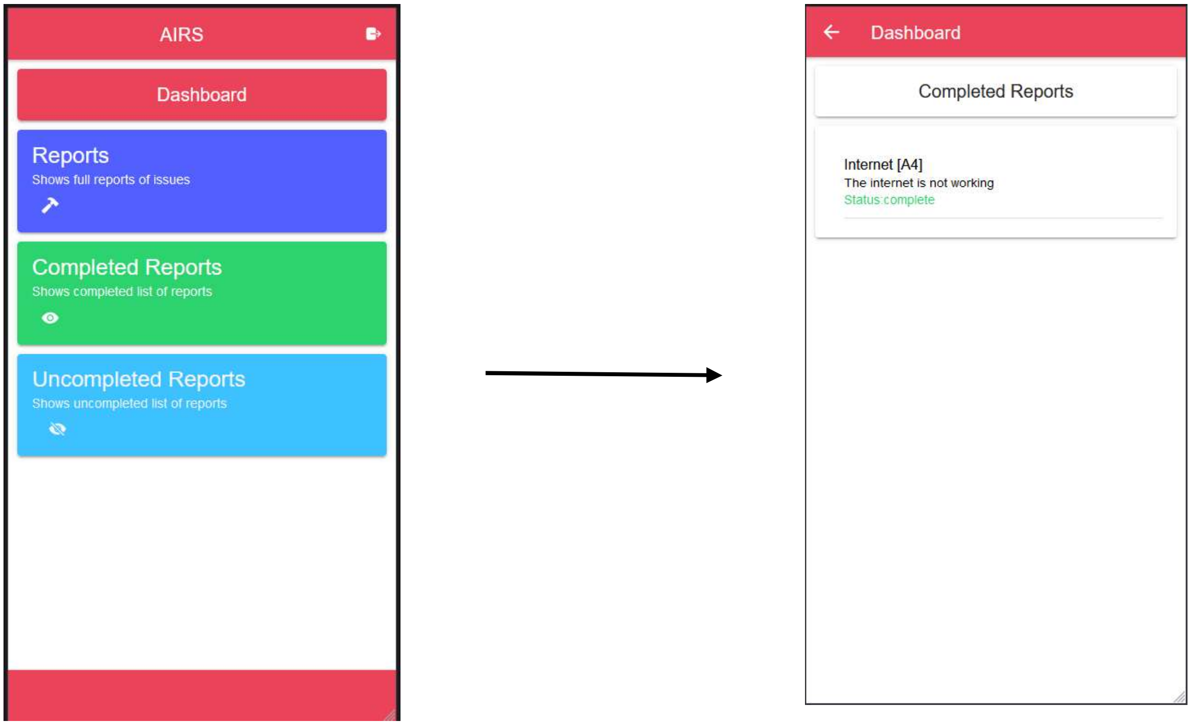
```
Document added  ▾ Object { d_: {…}, firestore: {…}, Xf: {…} }                                    tab2.page.ts:49:14
                  ▸ Xf: Object { firestore: {…} }
                  ▸ d_: Object { v_: null, S_: {…}, type: "document", … }
                  ▸ firestore: Object { d_: {…}, Rd: {}, INTERNAL: {…}, … }
                  ▸ <prototype>: Object { constructor: e(e, n) ↱, collection: collection(t) ↱, isEqual: isEqual(t) ↱, … }
```

Figure 5.7: Browser console output of added report

The result in *Figure 5.6* shows the report being made as well as the report being displayed in the list of reports. Also, the console output in the browser in *Figure 5.7* shows that the document has been added to the database as an object.

Manager Report

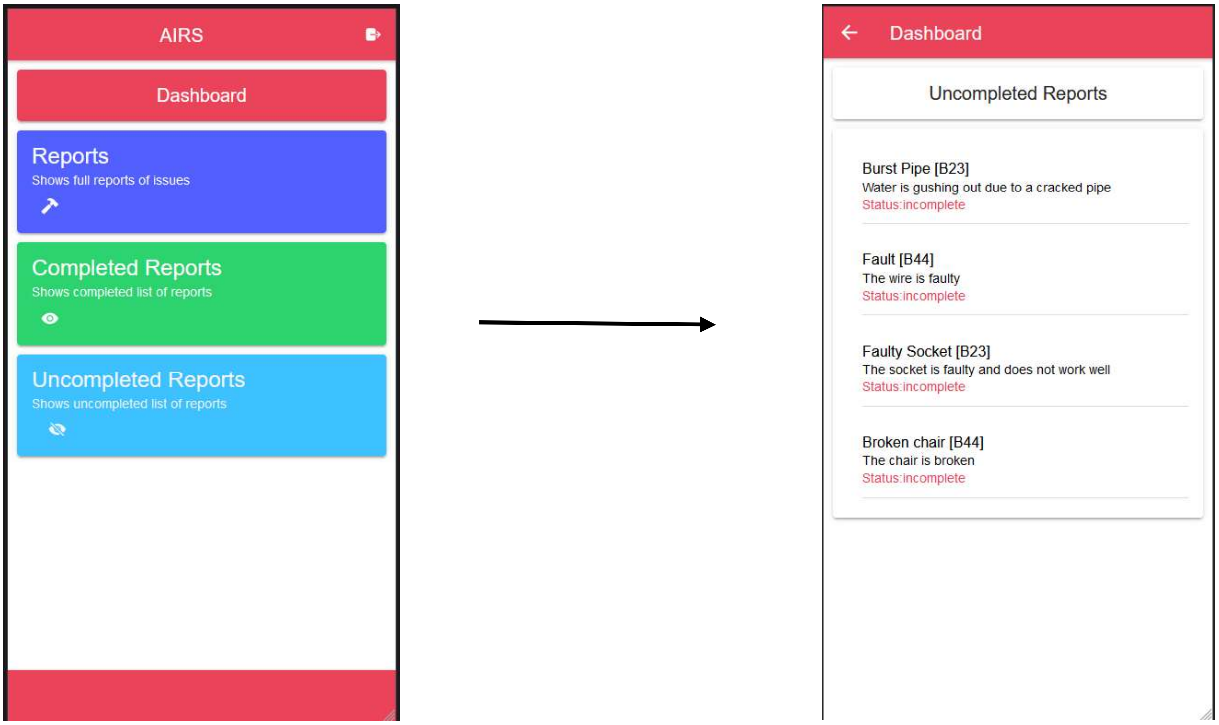
The manager's account displays all the reports made by the student. The manager has a dashboard to help facilitate and handle reports. The manager has a list of the incomplete report and, once the issue is resolved, the report moves to the list of complete reports. This component is tested, and the results displayed showing the management of reports. *Figure 5.8* below shows the list of completed files when the icon for completed reports is clicked.


AIRS
Dashboard
Reports
Shows full reports of issues
Completed Reports
Shows completed list of reports
Uncompleted Reports
Shows uncompleted list of reports
Dashboard
Completed Reports

Internet [A4]
The internet is not working
Status:complete

Figure 5.8: Test to display complete reports

Next, the manager clicks the icon for uncompleted reports. The manager is then routed to the next page, displaying a list of incomplete issues that need to be resolved. Each report shows the issue, room number, details, and the status of the report. *Figure 5.9* below displays the list of incomplete reports when the icon is clicked navigating the manager to the list.



AIRS

Dashboard

Reports
Shows full reports of issues

Completed Reports
Shows completed list of reports

Uncompleted Reports
Shows uncompleted list of reports

Dashboard

Uncompleted Reports

Burst Pipe [B23]
Water is gushing out due to a cracked pipe
Status:incomplete

Fault [B44]
The wire is faulty
Status:incomplete

Faulty Socket [B23]
The socket is faulty and does not work well
Status:incomplete

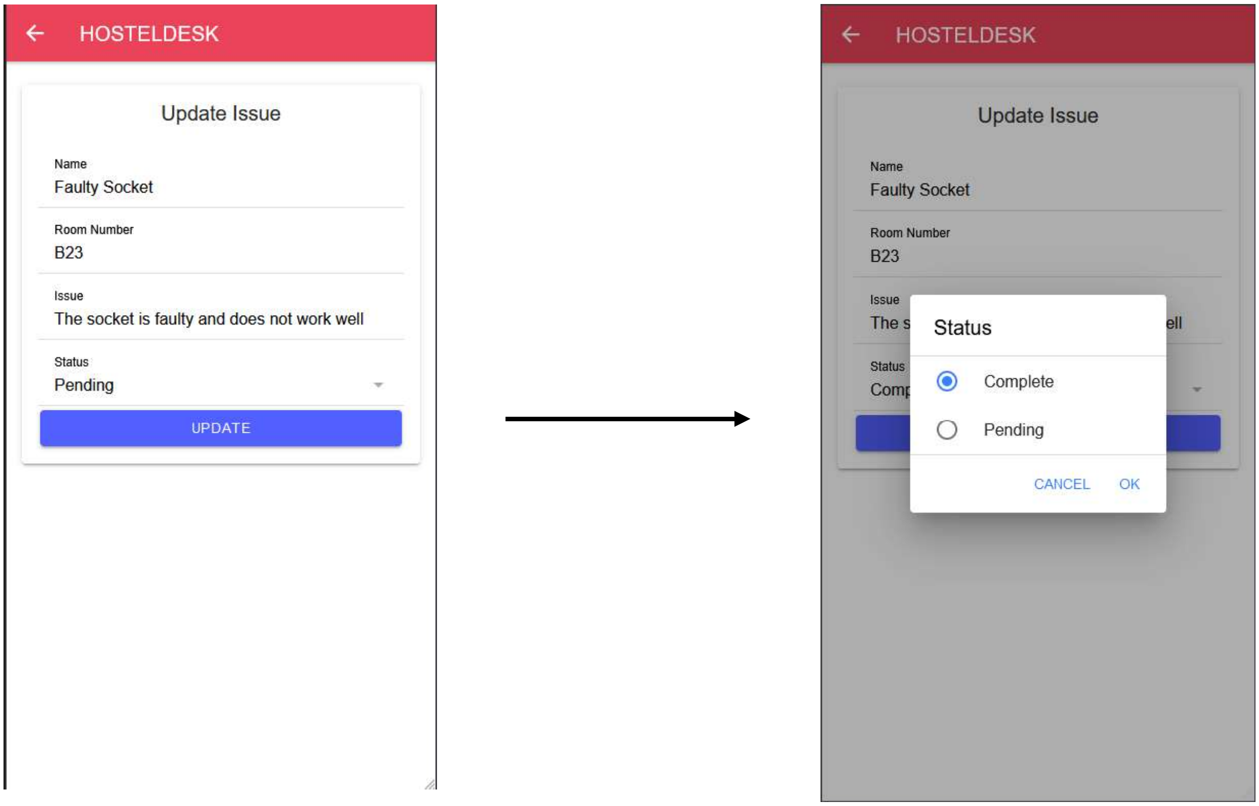
Broken chair [B44]
The chair is broken
Status:incomplete

Figure 5.9: Test to display incomplete reports

Update Report

The manager needs to update students when the issue is resolved or still ongoing. In testing this component, the report is selected, and the status is updated. The updated information is stored in the database and displayed to the manager in the completed reports list. Once the updated report has been stored, the updated details of the document can be printed on the console.

The images in *Figure 5.10* and *Figure 5.11* below shows the flow of the report being updated and routed to the complete list of reports. Once the report is updated, the document's details are printed on the console shown in *Figure 5.12*. The exact details are shown on the console except for the status is now stored as 'complete' in the database. *Figure 5.13* shows the updated information on the firebase database.



Name
Faulty Socket

Room Number
B23

Issue
The socket is faulty and does not work well

Status
Pending

Name
Faulty Socket

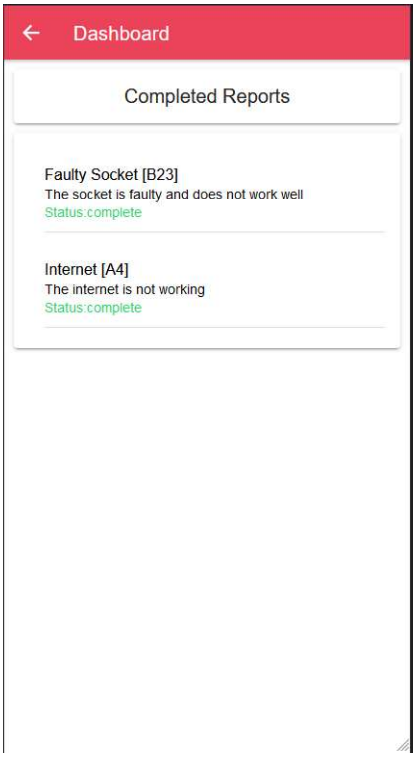
Room Number
B23

Status

● Complete

○ Pending

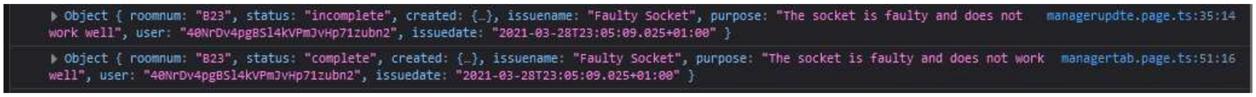
Figure 5.10: Test of updated status



Faulty Socket [B23]
The socket is faulty and does not work well
Status:complete

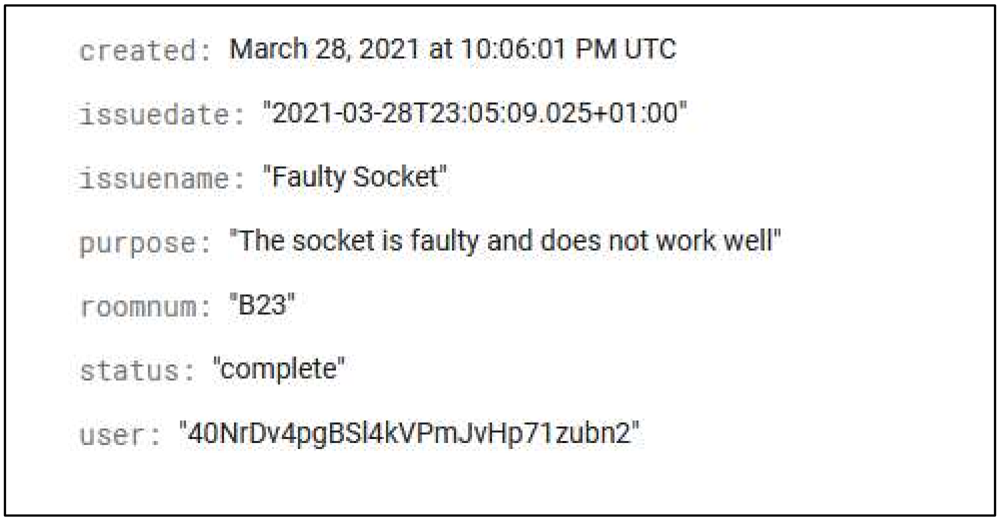
Internet [A4]
The internet is not working
Status:complete

Figure 5.11: Completed Reports List



```
▶ Object { roomnum: "B23", status: "incomplete", created: {…}, issuename: "Faulty Socket", purpose: "The socket is faulty and does not   managerupdte.page.ts:35:14
work well", user: "40NrDv4pgBSl4kVPmJvHp71zubn2", issuedate: "2021-03-28T23:05:09.025+01:00" }
▶ Object { roomnum: "B23", status: "complete", created: {…}, issuename: "Faulty Socket", purpose: "The socket is faulty and does not work  managertab.page.ts:51:16
well", user: "40NrDv4pgBSl4kVPmJvHp71zubn2", issuedate: "2021-03-28T23:05:09.025+01:00" }
```

Figure 5.12: Browser console Output (Status update)



```
created: March 28, 2021 at 10:06:01 PM UTC
issuedate: "2021-03-28T23:05:09.025+01:00"
issuename: "Faulty Socket"
purpose: "The socket is faulty and does not work well"
roomnum: "B23"
status: "complete"
user: "40NrDv4pgBSl4kVPmJvHp71zubn2"
```

Figure 5.13: Database Document

Media View

Once the photo or video has been uploaded to firebase storage, the manager can click on the file to view the image or video. In testing this component, an image is uploaded to firebase storage. When the image is clicked, the generated URL is displayed in the web console. The image file is displayed in the web browser. *Figure 5.14* displays the flow of the image when clicked, displaying the images on the browser from a URL. Also, in *Figure 5.15*, the output shows the URL for the image.

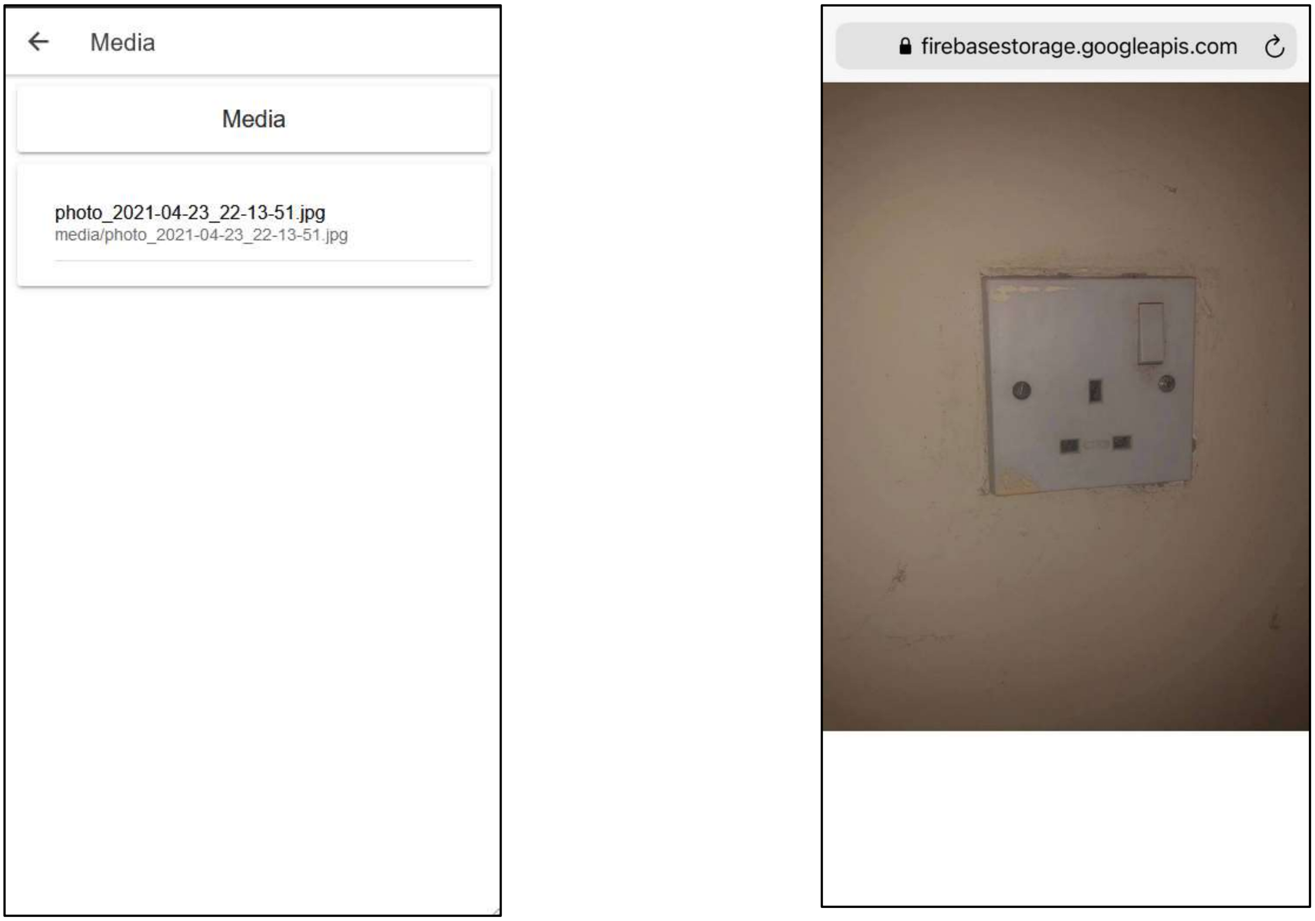
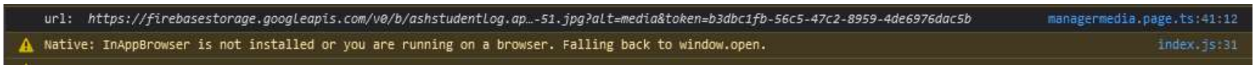


photo_2021-04-23_22-13-51.jpg
media/photo_2021-04-23_22-13-51.jpg

Figure 5.14: Test to View Media



```
   url:  https://firebasestorage.googleapis.com/v0/b/ashstudentlog.ap…-51.jpg?alt=media&token=b3dbc1fb-56c5-47c2-8959-4de6976dac5b          managermedia.page.ts:41:12
⚠  Native: InAppBrowser is not installed or you are running on a browser. Falling back to window.open.                                           index.js:31
```

Figure 5.15: Browser console Output (URL)

5.2 Compatibility Testing

Ionic is a hybrid application used on the two leading mobile operating systems, Android and IOS. Developing the application using ionic makes it compatible with both platforms when deployed. However, in deploying the application to the IOS platform, there are measures taken to ensure the safety of the application before it is available on its store. On the other hand, the Android platform is open source and allows developers to deploy and test applications with minimal measures. Testlab is one of the services provided by Firebase. This cloud-based testing infrastructure enables developers to run applications on a wide range of devices. Using this testing software provides insight into any issues encountered in the real world when downloaded on any mobile device. Robotest is a testing tool on the Firebase TestLab to simulate user operation on a virtual device to assess the application's compatibility when installed on an android device. The application is tested on devices with API levels 23, 26, and 27. The test from the crawl statistics shows that five actions were taken when the application was simulated. The test result shows that the application runs on the three APIs successfully without any accessibility issues.

Chapter 6: Conclusion and Recommendations

6.1 Conclusion

In conclusion, the incident reporting application, HostelDesk is developed to solve communication between students and managers in managing hostel problems. Through research, communication between hostel managers and students was a problem, creating bad reviews for hostels. Also, the manager, a single individual, is responsible for handling the hostel and its facilities, making it difficult to attend to every issue. The proposed solution, HostelDesk, aims to create a simple way for students to report problems in the hostel, reducing the stress on the manager responsible for handling the hostel. The application also makes it possible for the user to send and view any photos or videos of the issue, allowing the manager to see the problem reported effectively. Developing and implementing the application marks the end of the project.

6.2 Recommendations

Though the project is complete, many other additional features can make the application better, helping both the student and manager. The application can only be used by one hostel. In newer releases, the application can allow multiple hostels to register on the application, receiving reports from students living in their respective hostels. Other features like facial recognition and fingerprint recognition can be implemented to improve authentication and security.

The application can also be integrated into universities and hostels to make it easier for students to log in with their student emails and automatically receive emails regarding their problems. Also, using student emails, the manager can send general announcements to students on any changes in the hostel. A review page can be implemented to allow students to rate hostel services. An analysis chart can enable the manager to assess the number of reports made within a fixed period, helping the manager know which persistent problems students encounter. Having this adds feature will provide managers with added information to improve hostel facilities and student satisfaction. Finally, Google maps can be integrated into the application to allow the manager to find shops and businesses tailored to fix or repair any issue. These integrations and implementation will improve the overall use of the application and assist managers in maintaining hostel facilities.

References

[1] A. Danso and S. Hammond. 2017. Level of Satisfaction with Private Hostels around Knust Campus. *International Journal of Science and Technology* 6, 3 (2017), 719–727. https://www.researchgate.net/profile/Sammy_Hammond/ publication/318866986_Level_of_Satisfaction_with_Private_Hostels_around_ Knust_Campus/links/59826c3faca272a370f58265/Level-of-Satisfaction-withPrivate-Hostels-around-Knust-Campus.pdf

[2] Roel P. Masongsong and Maria Amelia E Damian. 2016. Help desk management system. *In Proceedings of the World Congress on Engineering and Computer Science*, Vol. 1. 1–6. http://www.iaeng.org/publication/WCECS2016/WCECS2016_pp269- 274.pdf

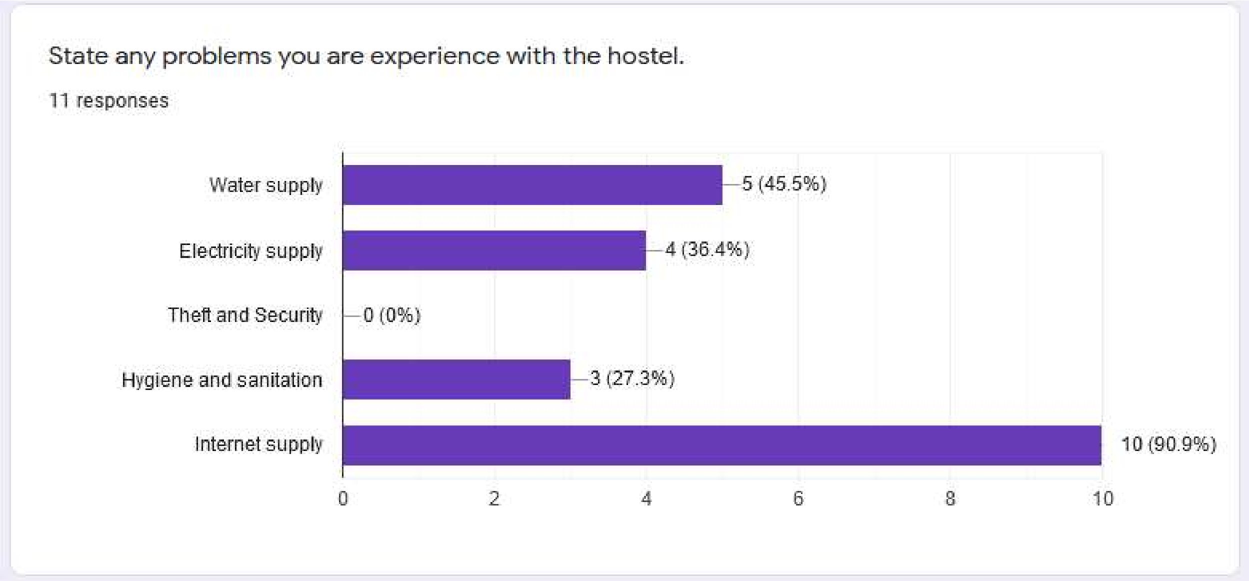
[3] O.J. Oladiran. 2013. A POST OCCUPANCY EVALUATION OF STUDENTS' HOSTELS ACCOMMODATION. *Journal of Building Performance* 4, 1 (2013), 1–11. http://spaj.ukm.my/jsb/index.php/jbp/article/view/78

[4] Nur Afifah Sanusi. 2019. A STUDY ON THE BUILDING MAINTENANCE PRACTICES IN STUDENTS'HOSTELS AT PUBLIC UNIVERSITIES. *e-Bangi* 16, 3 (2019), 1–17. http://ejournals.ukm.my/ebangi/article/view/31409/9094

[5] Clarkson M. Wanie, Emmanuel E.E. Oben, Jeff Mbella Molombe, and Ivo T. Tassah. 2017. Youth advocacy for efficient hostel management and affordable university students' housing in Buea, Cameroon. *International Journal of Housing Markets and Analysis* (2017), 1–3.

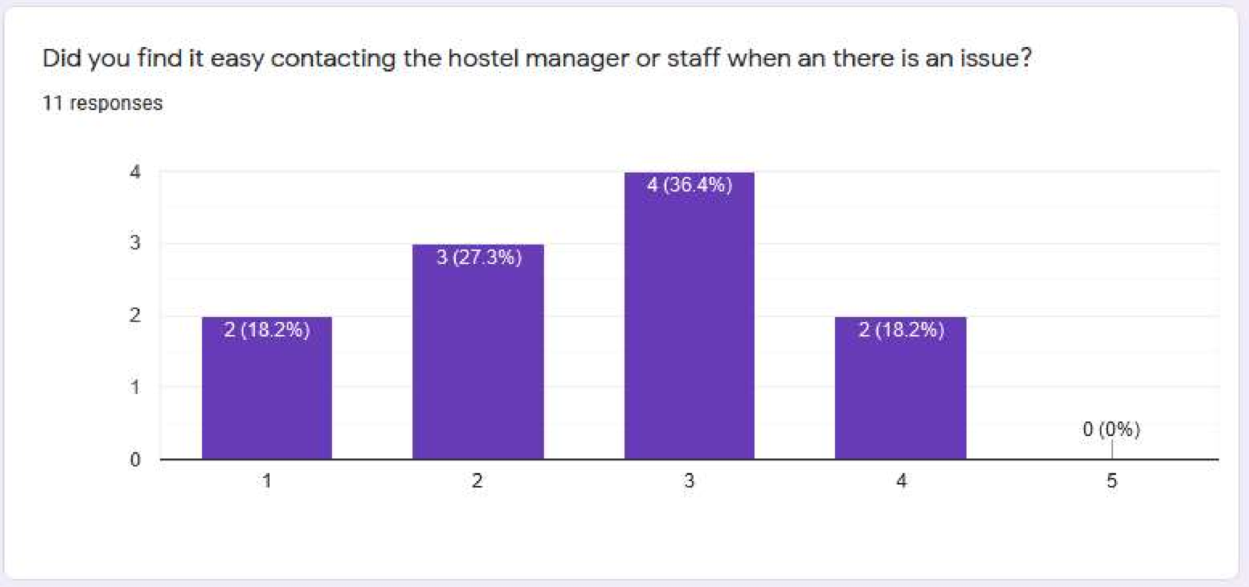
Appendix

Appendix A: Requirement analysis results



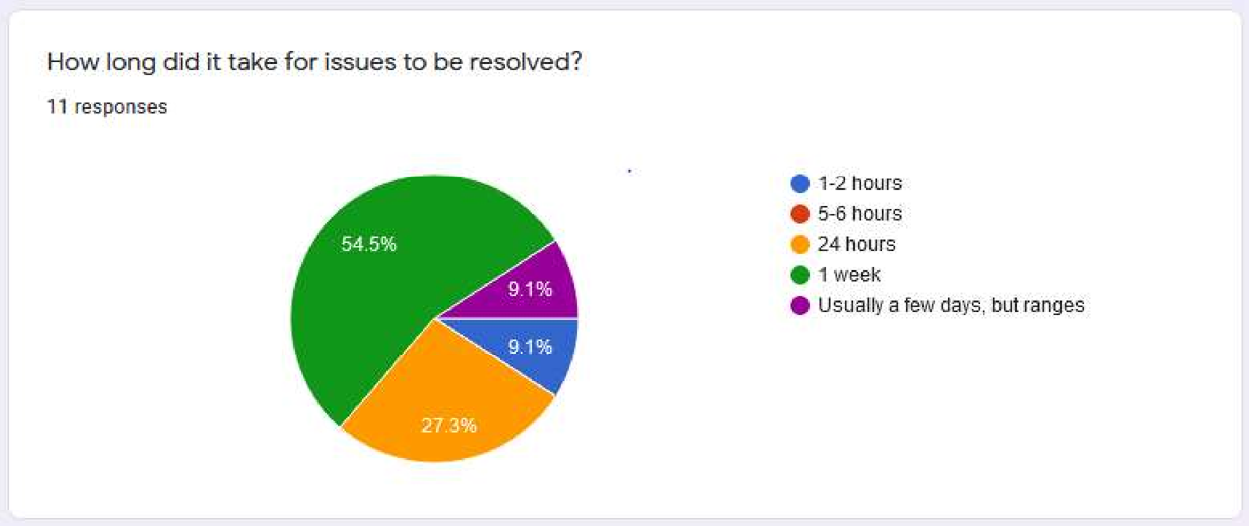
State any problems you are experience with the hostel.

11 responses



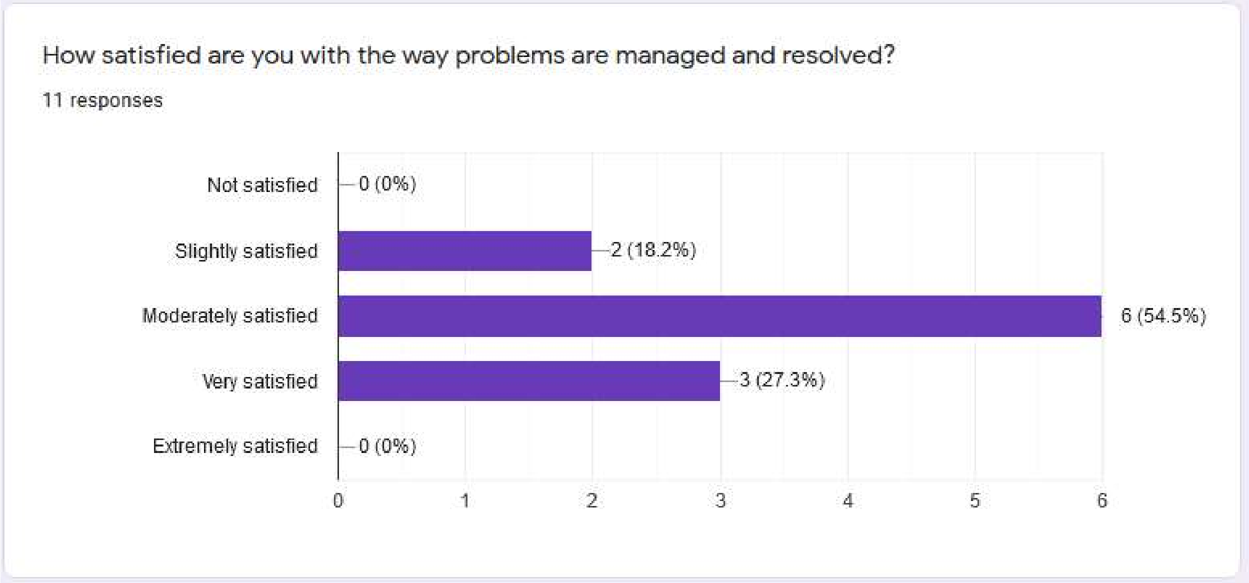
Did you find it easy contacting the hostel manager or staff when an there is an issue?

11 responses



How long did it take for issues to be resolved?

11 responses



How satisfied are you with the way problems are managed and resolved?

11 responses

Appendix B: Hostel Incident Management Survey

Hostel incident management

This survey will go towards the development of an application for hostel incidents off-campus. For Ashesi students only.

* Required

1. What year group do you belong to? *

2020

2021

2022

2023

2024

2. Do you live on-campus or off-campus during the academic year? *

On-campus

Off-campus

Any reason for your choice?

Your answer

3. If you live off-campus during the academic year, which hostel is your preferred choice ? *

Hosanna

Dufie

Columbiana

Masere

Tanko

Ceewus

Charlotte

Queenstar

Other:

4. What type of accommodation do you living in? (2 in a room, 3 in a room etc.). *

1 in a room

2 in a room

3 in a room

4 in a room

5. Were you satisfied with the services provided by the hostel. i.e Electricity, water, management. *

Not satisfied 1 2 3 4 5 Very Satisfied

6. State any problems you are experience with the hostel. *

Water supply

Electricity supply

Theft and Security

Hygiene and sanitation

Internet supply

7. Give a rating for each of the following services in your hostel. *

	Not a problem	Minor problem	Moderate problem	Serious Problem
Water Supply				
Electricity Supply				
Theft and Security				
Hygiene and Sanitation				
Internet Supply				

8. To whom do you report to when are any problems of issues *

Caretaker

Hostel owner

9. Did you find it easy contacting the hostel manager or staff when an there is an issue? *

Very easy 1 2 3 4 5 Difficult

10. How long did it take for issues to be resolved? *

1-2 hours

5-6 hours

24 hours

1 week

Other:

11. How satisfied are you with the way problems are managed and resolved? *

Not satisfied

Slightly satisfied

Moderately satisfied

Very satisfied

Extremely satisfied

12. Would you be interested in using an application that allows you to report issues to the manager easily?

Yes

No

13. What functionalities would you want to see on the app?