**Software Design Specification**

**The Software Design Specification Outline**

**1. Introduction**

A document that provides an overview about the process and development of the software. This document provides narrative and graphical records of the software design for the project that includes class diagrams, sequences diagrams, collaboration models, object behavior models and other supporting requirement information.

**1.1 Purpose of this document**

To a degree, the Design Document could be seen a part of the plan of what the outcome would be while providing a particular input, containing an amount of information about the kinds of inputs, outputs, classes and functions.

**1.2 Scope of the development project**

This document provides a characteristic description of the software architecture of Hostel Tour System, Specifying the structure and design of some of the modules discussed in the SRS. It also displays some of the use cases that had transformed into sequential and activity diagrams. The class diagrams show how the programming team would implement specific module.

**1.3 Definitions, acronyms, and abbreviations**

IEEE: Institute of Electrical and Electronics Engineers

SDS: Software Design Specification

Document Interaction Class, XML Document Interaction Engine: These are the two terms that will be used to refer to the main software class described within this document.

Required Field: A critical field is a field in a data set for a document that is required for successful document generation. For example, missing parties in a case, missing county location of court, or other data elements that are required to create a valid legal document.

Workflow: The movement of documents through a work process that is structured into tasks with designated persons or systems to perform them and the definition of the order or pathway from start to finish for the work process.

Structured Data Format: A structured data format is data assembled into a discernible structure, such as when data is placed into an XML instance which is validated through the use of an XML schema which defines the structure of the XML document.

**1.4 References**

IEEE SDS template.

Lucidchart tutorials

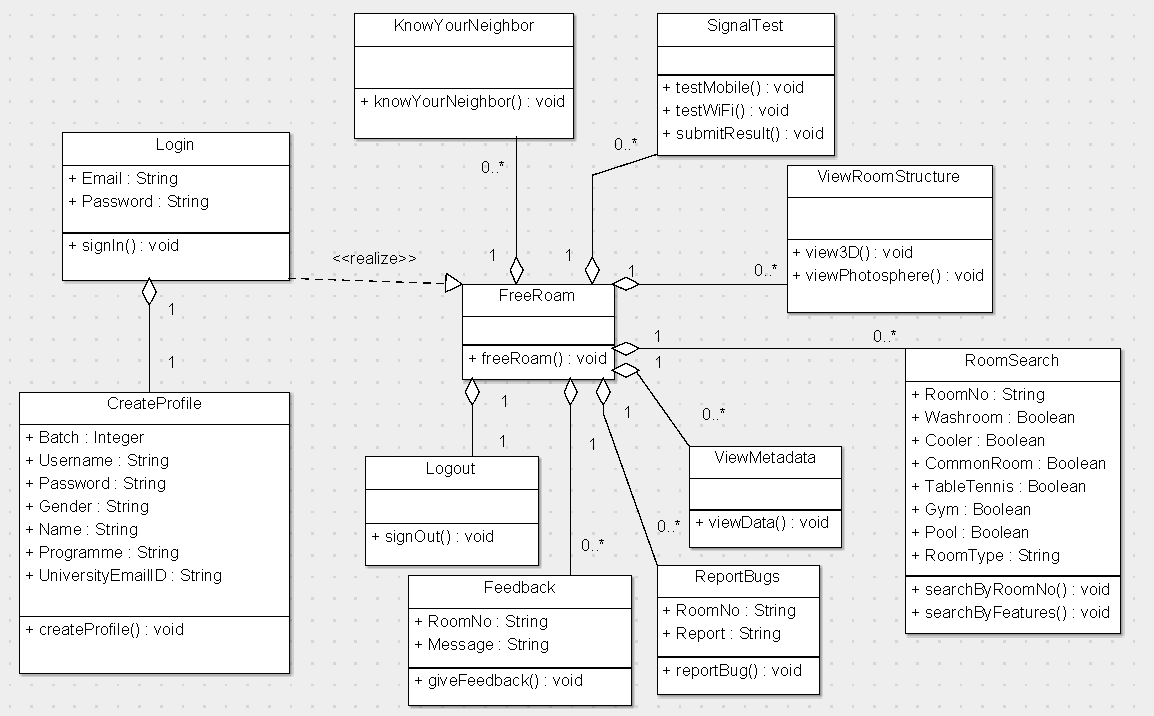
Last years SDS document.

**1.5 Overview of document**

The Software Design Specification is divided into 5 sections with various subsections. The sections of the Software Design Specification Document are :

1. Introduction
2. Logical Architecture
3. Execution Architecture
4. Design decisions and tradeoffs
5. Pseudocode

**2. Logical Architecture (Class Diagram, Sequence Diagram, State Diagram)**



**2.1 Class Diagram**

**2.1.1 Class name : CreateProfile**

**Input :** Batch, Username, Password, Gender, Name, Programme, UniversityEmaild.

**Output :**  Notification is generated displaying account is created if the account is created, else a notification displaying account is not created.

**Description :** This class allows the user to create a new profile to login into the system.

**Method :** createProfile()

**Method Description :** As the user enters the data or fills the form and accepts the terms and conditions. This method then creates a new account by adding the email and password to the Firebase Authentication.

**2.1.2 Class name : Login**

**Input :** Email, Password.

**Output :** Free Roam Scene initiated.

**Description :** This class allows the user to enter the system by authenticating the email and password entered.

**Method :** signIn()

**Method Description :** This method is responsible for using the entered credentials to authenticate with the Firebase Database Authentication. If the credentials are true, it would enter into the system, if false then a notification would pop from from the right saying wrong credentials(wrong password/ user does not exists).

**2.1.3 Class name : Logout**

**Input :** none

**Output :** Notification displaying Logged out successfully is displayed and the login screen is initiated.

**Description :** This class allows the user to logout from the application.

**Method :** signOut()

**Method Description :** As the user clicks on the Logout button the signOut() method gets called, invalidating the signed in user and redirecting the application to the login screen.

**2.1.4 Class name : RoomSearch**

**Description :** This class allows the user filter out the desired room that they were searching for based on the speciality or features or only based on the room number.

**Method1 :** searchByFeatures()

**Input :** boolean RoomType, Washroom, WaterCooler, CommonRoom, TT, Gym, Pool, WiFiStrength, MobileStrength.

**Output :** A list of room numbers is displayed according to the search results.

**Method Description :** As the user clicks on the checkboxes making the values true, and when the search button is clicked on then the filter then sorts out all the rooms based on the users choices. And displays a list of room numbers.

**Method2 :** searchByRoomNo()

**Input :** RoomNo

**Output :** The character present in free roam is teleported outside of the room number entered(Unity Scene)

.

**Method Description :** As the user enters the room number and clicks on search button, the character present in Free Roam Scene is then teleported to the area outside the room number entered by the user.

**2.1.5 Class name : BugReport**

**Description :** This class allows the user report an issue that the application has in its current version.

**Method :** reportBug()

**Input :** Message.

**Output :** A notification from the right popping out saying that the bug has been reported.

**Method Description :** As the user enters the message in the message box, the message is then inserted into the Firebase Database so that the admins can rectify the issues in our application.

**2.1.6 Class name : Feedback**

**Description :** This class allows the user to give a feedback about the users experience and ways of improving it.

**Method :** giveFeedback()

**Input :** Message.

**Output :** A notification from the right popping out saying that the feedback has been submitted.

**Method Description :** As the user enters the message in the message box, the message is then inserted into the Firebase Database so that the admins can apply their suggestions if appreciated.

**2.1.7 Class Name : KnowYourNeighbor**

**Description :** This class allows the user to check his/her neighbor’s details.

**Method :** knowYourNeighbor()

**Input :** None

**Output :** List of nearby students

**Method Description:** This Method will show the resident of the particular room brief details of the students residing nearby his room. It will open a new unity scene which will have Name, Batch and programme of the students residing in the nearby rooms in a defined radius.

**2.1.8 Class Name : FreeRoam**

**Description:** Free Roam class allows the user to freely roam inside the hostel corridors and check the internal structure of rooms and hostels.

**Method :** freeroam()

**Input :** None

**Output :** Unity Scene

**Method Description:** When user click on Free roam Button the user will be redirected to a unity scene which will enable the user to roam freely in the hostel and explore it. The user will be able to check new rooms.

**21.9 Class Name : Metadata**

**Description:** Metadata allows the user to access the room information.

**Method :** viewMetadata(String RoomNo)

**Input :** Click on the door of the room user want to see the metadata of

**Output :** Unity Scene

**Method Description:** When the user stands in from of a room in the FreeRoam unity scene and clicks on the door, a new unity scene will popup showing the Metadata of the room which will include :-

* WiFi Strength
* Mobile Strength
* WiFi Internet Speed
* Mobile Internet Speed
* Vicinity to pool room, washroom, watercooler, gym, Table tennis room, Common Room

**2.1.10 Class Name : ViewStructure**

**Description:** This class enables the user to view the structure of a room as per his/her wish be it a 3D model, a photosphere or only a photo.

**Method 1 :** viewPhotosphere()

**Input :** When the user clicks on the Photosphere option.

**Output :** A photosphere preview of a room.

**Method Description :** When the user clicks on the Photosphere button, an image of that particular room is displayed.

**Method 2 :** view3D()

**Input :** When the user clicks on the 3D option.

**Output :** A character is allowed to roam in the new room environment.

**Method Description :** When the user clicks on the 3D button, the Unity scene changes according to the particular room type.

**2.1.11 Class Name : SignalTest**

**Description:** This class enables the user to perform a test to find out the strength and speed of either Mobile or the WiFi.

**Method :** testWifi()

**Input :** Click on the start button to start the test.

**Output :** Test results in a tabular form.

**Method Description:** This method is initiated when the start test button is pressed. And returns the results of test conducted for the WiFi back to the user in a tabular form.

**Method :** testMobile()

**Input :** Click on the start button to start the test.

**Output :** Test results in a tabular form.

**Method Description:** This method is initiated when the start test button is pressed. And returns the results of test conducted for the Mobile back to the user in a tabular form.

**Method :** submitResult()

**Input :** Click on the submit button.

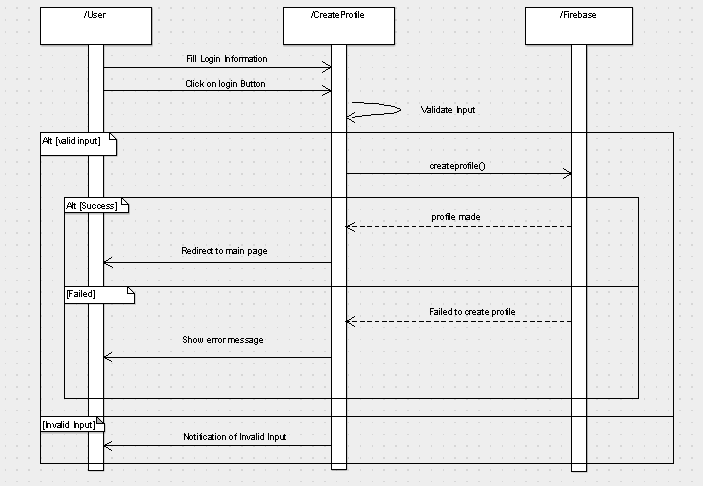
**Output :** A notification displaying that the results have been submitted.

**Method Description:** This method is used to send the test results that were generated to the admins to the Firebase database and then the admins would take note of it to modify the results for the metadata after verifying.

**2.2 Sequence Diagrams**

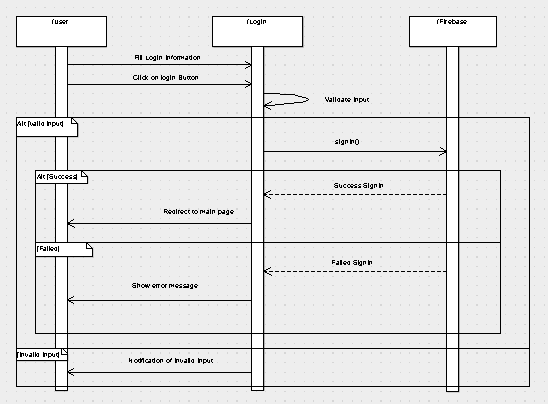
Arrow line signifies there is a send message taken place. Response is being shown by dotted arrows.

**2.2.1 Sequence Diagrams : Creating a Profile**

****

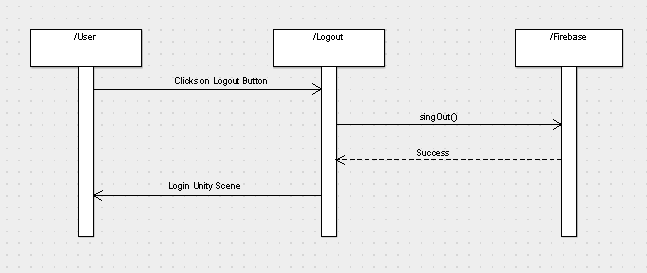
It allows the users to create a new user profile by entering the necessary information and clicking the register button and after verification of the profile a new user is created in Firebase.

**2.2.2 Sequence Diagrams : Login**



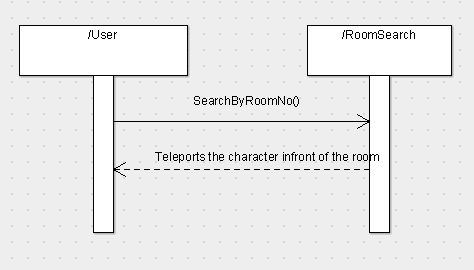
It allows students to login with their NIIT University mail domain and password which are registered in the database already.

**2.2.3 Sequence Diagrams : Logout**

****

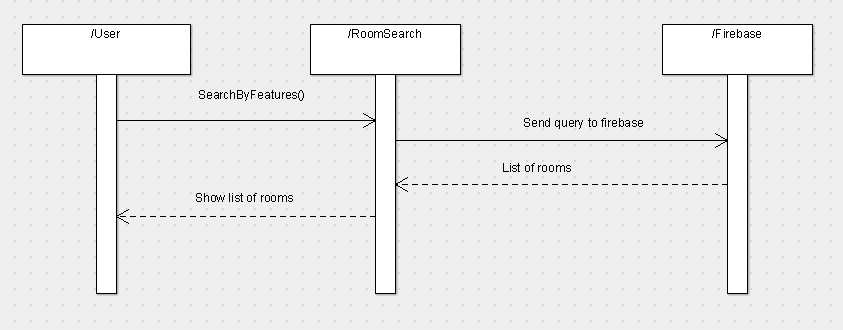
It will sign out the user from the application

**2.2.4 Sequence Diagrams : Search room by room number**

****

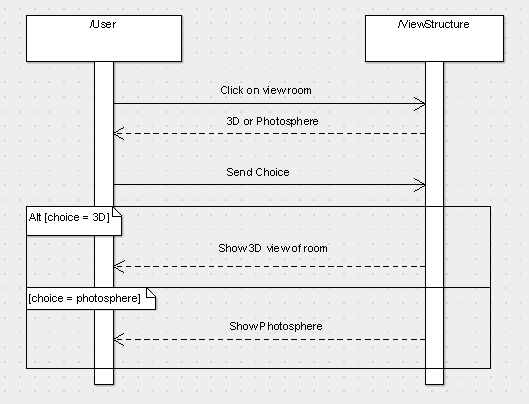
It allows the user to teleport the character present in the Unity Scene in front of the room number entered.

**2.2.5 Sequence Diagrams : Search rooms based on features**

****

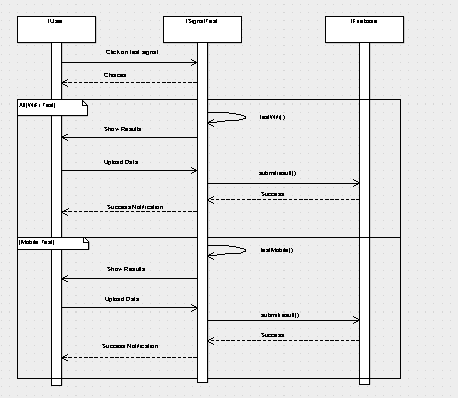
User will have the ability to search rooms based on various parameters which will be shown in the form displayed on the screen. Once the user clicks on the submit a list of rooms will appear having the qualities based on the input by the user in the previous form.

**2.2.6 Sequence Diagrams : View room structure**



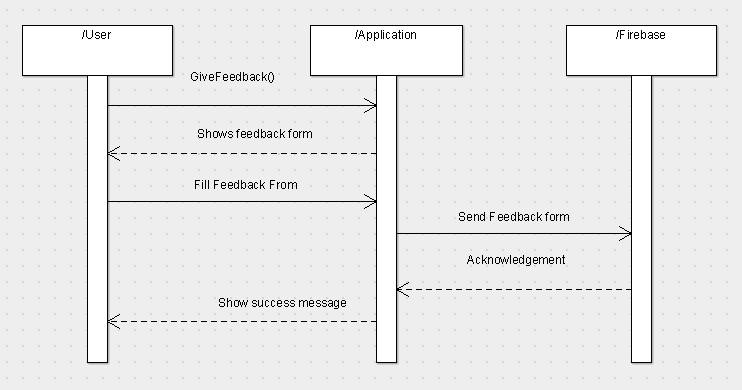
It allows the user to preview the room in the form of a photosphere or a 3D model, where the photosphere is static and the 3D is like another environment where the character can freely roam.

**2.2.7 Sequence Diagrams : Signal test**

****

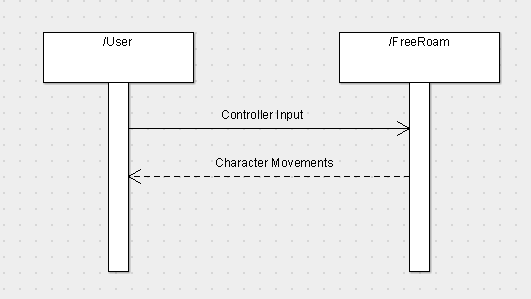
User can test their Wifi signal and strength, Mobile signal and strength and then an option will appear for the user to submit the test results which were conducted before.

**2.2.8 Sequence Diagrams : Giving feedback for a room**

****

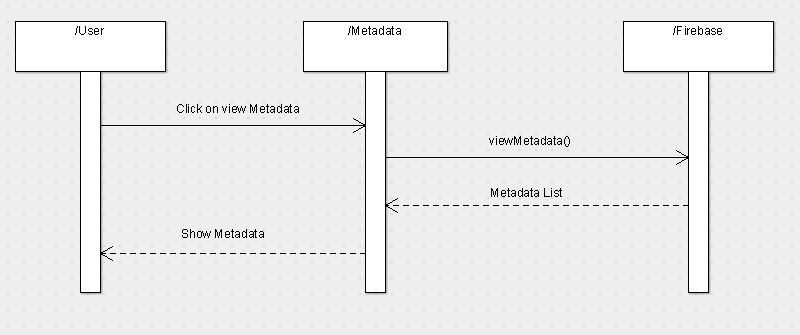
Providing the user a dialog box having a message box where the user enters the feedback (his/her experience with the application) or some suggestions for improving. And a submit button to send this message to the Firebase Database.

**2.2.9 Sequence Diagrams : Free roam**

****

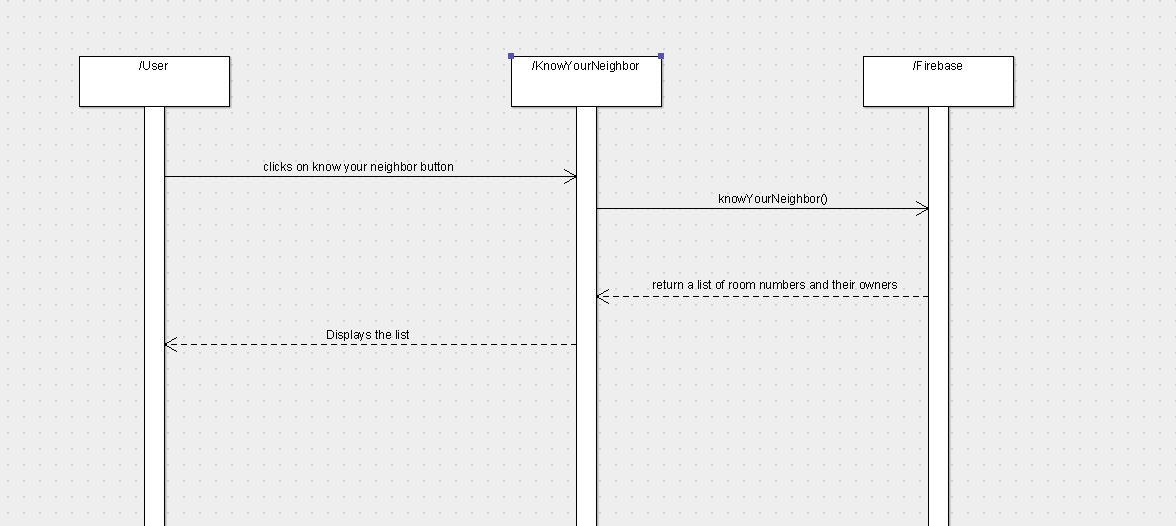
It allows the user to freely control the character using the dual third person controller to control the movements of the character and the characters perspective.

**2.2.10 Sequence Diagrams : View metadata rooms**



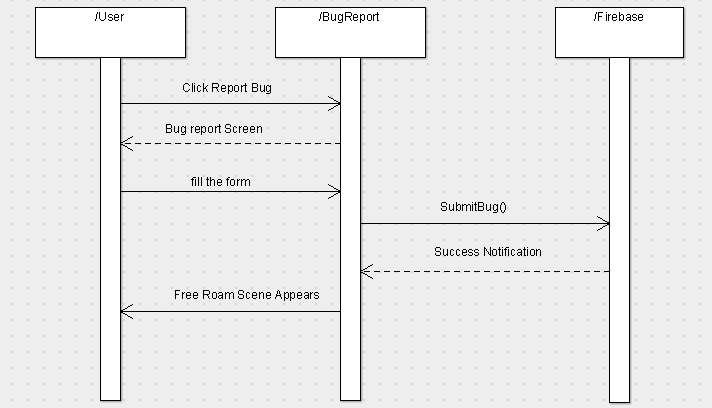
When the user is in the free roam scene and clicks on the door a list will appear which will show the Metadata of that particular room.

**2.2.11 Sequence Diagrams : Know your Neighbor**



User will see the students residing in the nearby rooms.

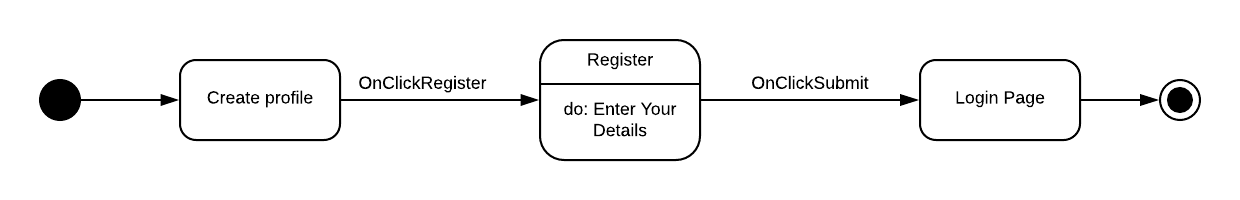
**2.2.12 Sequence Diagrams : Bug report**

****

User will be able to report bugs to developers of the application.

**2.3 State Diagrams**

**2.3.1 State Diagram : Creating a Profile**



**Description**

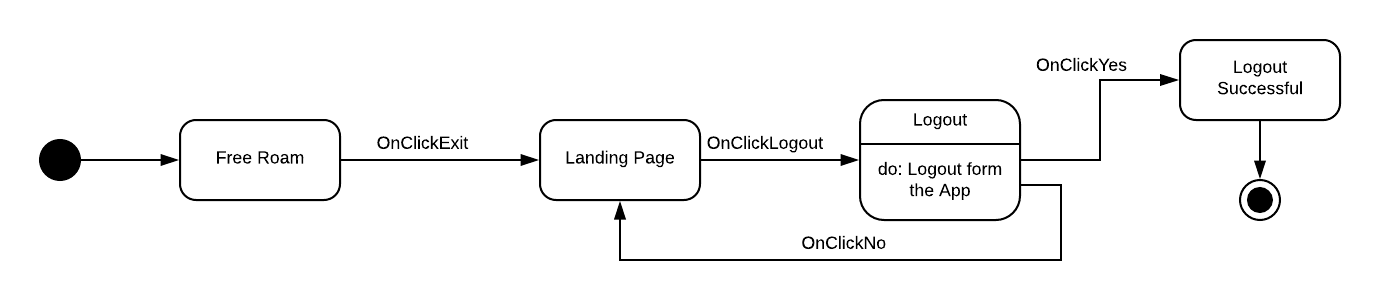
In this the user will register for the app if he/she is new to the app. On clicking on the OnClickRegister Button, this will direct the user to Register functionality, which will ultimately take the user to Login Page just after successful completion of details.

**2.3.2 State Diagram : Login**

**Description**

On the landing page of the app, you will be having a Login Page option from where you can Login into the app (if you have already registered for the app). If details are invalid then it’ll take back you to the same Login Page and if Login is successful then user will be directed to Free Roam option.

**2.3.3 State Diagram : Logout**

****

**Description**

If the user clicks the Exit Button then it’ll direct the user to Landing Page of the app. From there is Logout button is clicked then on clicking YES, user will be logged out of the app and if clicked NO then it returns to Landing Page.

**2.3.4 State Diagram : Search room by room number**

****

**Description**

While searching the rooms, if your search is based on room number then on clicking SearchByRoomNo. Button you can do so. It will direct you to the Room No. box and then to Free Roam functionality. If details are wrong then Error Log dialog box will pop up.

**2.3.5 State Diagram : Search rooms based on features**

****

**Description**

This will provide you the Feature based search facility. If you click on the OnClickSearchByFeature button, you will be directed to a form which on clicking Search button will take you to the desired room. On wrong filling of details Error Log box will pop up.

**2.3.6 State Diagram : View room structure**



**Description**

From Free Roam if you click on the ViewRoomStructure Button then you will land at the room’s internal structure page. Here you can view the Photosphere or 3D structure of room. On clicking Back Button you will again land up at Free Roam page.

**2.3.7 State Diagram : Signal test**

****

**Description**

Signal Test is the feature where you can view Wifi and phone network signal strengths. When you click on the Mobile Signal button, you can search mobile signal strength and same goes with the Wifi Speed signal. On clicking Start, the Signal Test process starts and after that on clicking Submit button you can submit the data collected. You can also test the signal again by clicking on SearchAgain button.

**2.3.8 State Diagram : Giving feedback for a room**

****

**Description**

Feedback option allows you to write your experience of the rooms if you have visited it or lived in it. On clicking the EnterFeedbackDetails you are allowed to write details and then Submit button allows you to submit the details. After clicking the OK button you will end up at the Landing Page.

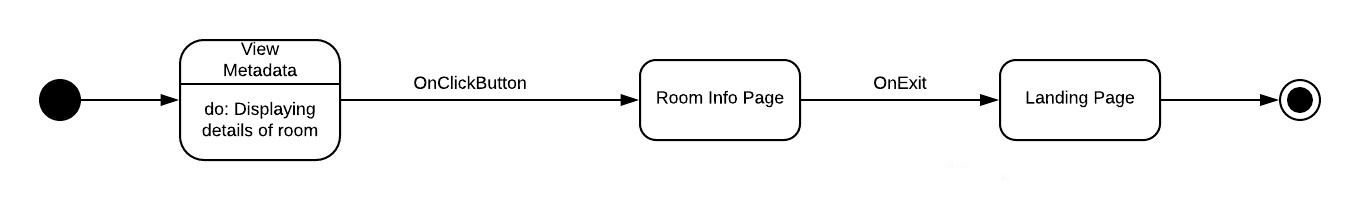
**2.3.9 State Diagram : Free roam**

****

**Description**

Free Roam allows you to freely roam in the hostels. The View Room Structure button allows you to check the internal structure of room. On clicking the Exit button you will be directed to the Landing Page of the app.

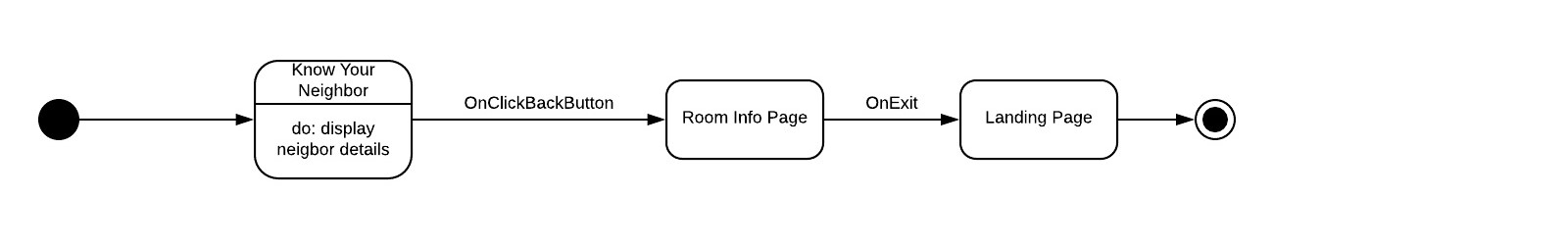
**2.3.10 State Diagram : View metadata rooms**



**Description**

View Metadata displays the internal information of the room. The exit button will take you to the Landing Page.

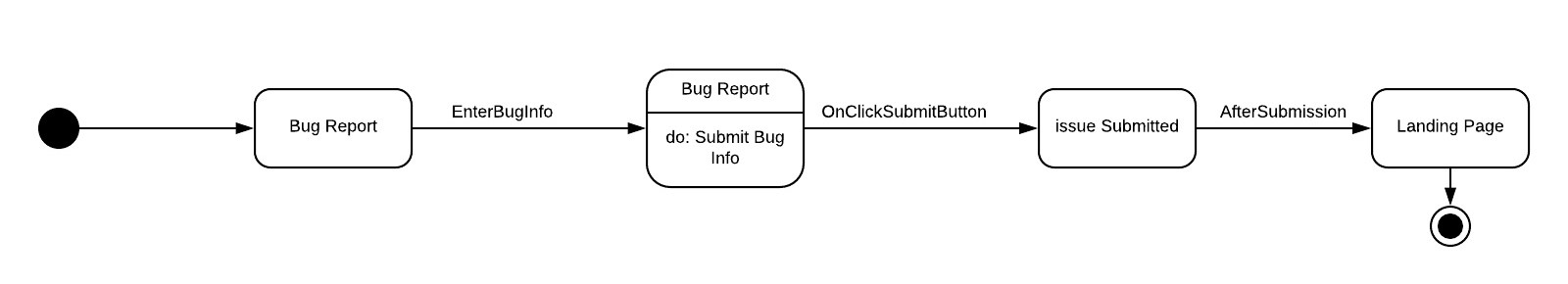
**2.3.11 State Diagrams : Know your Neighbor**



**Description**

Know Your Neighbor feature displays your neighbor’s information. The Back button will take you to the Room Info Page.

**2.3.12 State Diagrams : Bug report**



**Description**

If the user encounters any kind of Bug or Error, he or she can report it to the developers via this feature. The Enter Bug Info button allows you to enter your bug details and the Submit button submits the details to developer. After successful submission the user is directed to Landing Page.

**3.0 Execution Architecture**

Runtime environment required is any device supporting Android Operating System with the minimum version of Android Marshmallow, minimum 4 GB of ram, Unity as a deployment platform.

**4.0 Design decisions and tradeoffs**

Database memory vs Algorithm complexity for Search room by features

To implement a basic filter section for the Search room based on features, had two options. Either we go with a long key and value in database to help filter the search based on rooms, making the algorithmic process way too easy to implement. But storing such a long key for each room making the database a huge mess. And the other way is to compare many keys and their values together making the algorithmic process too much complex as Firebase only allows to one query at a particular time. So we had to store a bit of data on the client side to make our search easier.

We decided to keep the database simple and make the code a bit complex to implement the filtering process faster and easier.

**5.0 Pseudocode**

**5.1 Class Name: Login**

Method 1 : Start()

Input : None

Output: None

1. auth = Firebase Authorizer using the Default Instance

Method 2 : SignIn()

Input : email, password

Output : Free Roam Scene

1. email = stores the email from user input.
2. password = stores the password from user input.
3. auth.SignInWithEmailAndPasswordAsync(email, password).ContinueWith(task =>
4. if task.IsCanceled is true {
5. Console displays "SignInWithEmailAndPasswordAsync was canceled.";
6. Return;
7. End if
8. if task.IsFaulted is true

7. Console displays "SignInWithEmailAndPasswordAsync encountered an error: " + task.Exception;

8. return;

9. End id

10. Firebase.Auth.FirebaseUser newUser = task.Result;

11. Console displays "User signed in successfully:user display name, user id

12. Unity SceneManager switches to first scene

**5.2 Class Name: CreateProfile**

Method 1 : Start()

Input : None

Output : None

1. auth = Firebase Authorizer using the Default Instance;
2. FirebaseApp.DefaultInstance.SetEditorDatabaseUrl<<<database--url>>>

Method 2 : createProfile()

Input : email, password, confirmpassword, gender, programme, batch

Output : Login Scene

1. email = Iemail.text + "@st.niituniversity.in";
2. password = Ipassword.text;
3. confirmpassword = Iconfirmpassword.text;
4. gender = Igender.text;
5. programme = Iprogramme.text;
6. batch = Ibatch.text;
7. if password is equal to “confirmpassword”

8. auth.CreateUserWithEmailAndPasswordAsync(email, password).ContinueWith(task =>

9. if task.IsCanceled is true

10. Console displays "CreateUserWithEmailAndPasswordAsync was canceled.";

11. return;

12. End if;

13. If task.IsFaulted is true

14. Console displays "CreateUserWithEmailAndPasswordAsync encountered an error: " + task.Exception;

15. return;

16. End if;

17. Firebase.Auth.FirebaseUser newUser = task.Result;

18. UserID = newUser.UserId;

19. Console displays “Firebase user created successfully: display name, id;

20. User user = new User(email, password, confirmpassword, gender, batch, programme);

21. string json = JsonUtility.ToJson(user);

22. DatabaseReference mDatabaseRef = FirebaseDatabase.DefaultInstance.RootReference;

23. mDatabaseRef.Child("User").Child(UserID).SetRawJsonValueAsync(json);

24. End if;

25. Else

26. Notification displays "Try Again";

27. End else;

**5.3 Class Name: SearchRoom**

Method 1 : Start()

Input : None

Output : None

1. FirebaseApp.DefaultInstance.SetEditorDatabaseUrl”<<database-url>>”;
2. DatabaseReference reference = FirebaseDatabase.DefaultInstance.RootReference;

Method 2 : searchByRoomNumber()

Input : roomNumber

Output : FreeRoam Scene

1. String roomNumber = takes the room number as input;
2. Button teleport = SceneManager changes the scene as per the room number;

Method 3 : searchByFeature()

Method 3.1 : public void getroomtype()

1. if TsingleRoom.isOn is true
2. FirebaseDatabase.DefaultInstance

.GetReference("rooms").OrderByChild("type").EqualTo("single")

.ValueChanged += HandleValueChanged;

3. End if

4. else if TtripleRoom.isOn is true

5. FirebaseDatabase.DefaultInstance

.GetReference("rooms").OrderByChild("type").EqualTo("triple")

.ValueChanged += HandleValueChanged;

6. End if

7. else if TsingleRoom.isOn is true && TtripleRoom.isOn is true

8. FirebaseDatabase.DefaultInstance

.GetReference("rooms").OrderByChild("type")

.ValueChanged += HandleValueChanged;

9. End if

Method 3.2 : void HandleValueChanged(object sender, ValueChangedEventArgs args)

1. if args.DatabaseError not equal to null
2. Console displays args.DatabaseError.Message;
3. return;
4. End if;
5. int index = 0;
6. foreach DataSnapshot snappy in args.Snapshot.Children
7. droom[index] = snappy.Key;
8. index++;
9. End of foreach
10. snapshot = args.Snapshot;
11. Call getroomwashroom(args.Snapshot) method

Method 3.3 : public void getroomwashroom(DataSnapshot snapshota)

1. int index = 0;
2. Console displays "Came to washroom";
3. if Twashroom.isOn is true {
4. foreach DataSnapshot snappy in snapshota.Children
5. if snappy.Child("washroom").Value.ToString().Equals("true")
6. droom[index] = snappy.Key;
7. "Washroom : " + droom[index]);
8. Index++;
9. End if;
10. End foreach
11. End if
12. Calling getwatercooler() method

Method 3.4 : public void getwatercooler()

1. int i = 0, j = 0;
2. Console displays "Inside Water Cooler before IF!";
3. if TwaterCooler.isOn is true // toggle of water cooler [on or off]
4. foreach DataSnapshot snappy in snapshot.Children
5. foreach string l in droom
6. if snappy.Key.ToString().Equals(l) is true
7. if snappy.Child("waterCooler").Value.ToString().Equals("true") is true

8. troom[j] = snappy.Key;

9. Console displays "Water Cooler: " + troom[j];

10. j++;

11. End if

12. End if

13. End foreach

14. End foreach

15. End if

16. else

17. troom = new string[droom.Length];

18. droom.CopyTo(troom, 0);

19. End else.

20. Calling getcommonroom() method

Method 3.5 : public void getcommonroom()

1. int i = 0, j = 0;
2. Array.Clear(droom, 0, droom.Length);
3. Console displays "Inside common room before IF";
4. if TcommonRoom.isOn is true
5. foreach DataSnapshot snappy in snapshot.Children
6. foreach string l in troom
7. if snappy.Key.ToString().Equals(l) is true
8. if snappy.Child("commonRoom").Value.ToString().Equals("true") is true
9. droom[j] = snappy.Key;
10. Console displays "Common Room: " + droom[j];
11. j++;
12. End if
13. End if
14. End foreach
15. End foreach
16. End if
17. else
18. droom = new string[troom.Length];
19. troom.CopyTo(droom, 0);
20. End else.
21. Calling getpoolroom() method

Method 3.6 : public void getpoolroom()

1. int i = 0, j = 0;
2. Array.Clear(troom, 0, troom.Length);
3. if Tpool.isOn is true
4. foreach DataSnapshot snappy in snapshot.Children
5. foreach string l in droom
6. if snappy.Key.ToString() is equal to (l)
7. if snappy.Child("poolRoom").Value.ToString() is equal to ("true")
8. troom[j] = snappy.Key;
9. Console displays "Pool Room: " + troom[j];
10. j++;
11. End if
12. End if
13. End foreach
14. End foreach
15. End if
16. else
17. troom = new string[droom.Length];
18. droom.CopyTo(troom, 0);
19. End else
20. Calling getgym() method

Method 3.7 : public void getgym()

1. int i = 0, j = 0;
2. Array.Clear(droom, 0, droom.Length);
3. if Tgym.isOn is true
4. foreach DataSnapshot snappy in snapshot.Children
5. foreach string l in troom
6. if snappy.Key.ToString() is equal to (l)
7. if snappy.Child("gym").Value.ToString() is equal to ("true")
8. droom[j] = snappy.Key;
9. Console displays "Gym Room: " + droom[j];
10. j++;
11. End if
12. End if
13. End foreach
14. End foreach
15. End if
16. else
17. droom = new string[troom.Length];
18. troom.CopyTo(droom, 0);
19. End else
20. Calling getTTroom() method

Method 3.8 : public void getTTroom()

1. int i = 0, j = 0;
2. Array.Clear(troom, 0, troom.Length);
3. if TTT.isOn is true
4. foreach DataSnapshot snappy in snapshot.Children
5. foreach string l in droom
6. if snappy.Key.ToString() is equal to (l)
7. if snappy.Child("TTRoom").Value.ToString() is equal to ("true")
8. troom[j] = snappy.Key;
9. Console displays "Table tennis Room: " + troom[j];
10. j++;
11. End if
12. End if
13. End foreach
14. End foreach
15. End if
16. else
17. troom = new string[droom.Length];
18. droom.CopyTo(troom, 0);
19. End else

Method 3.9 : void filter(DataSnapshot snapshot)

1. foreach DataSnapshot snappy in snapshot.Children
2. if snappy.Child("network\_strength\_category").Value.ToString() is equals to "good"
3. Console displays "Test : " + snappy.Key;
4. End if
5. End foreach

**5.4 Class Name: Logout**

Method 1 : start()

Input : None.

Output : None.

1. Firebase.Auth.FirebaseAuth auth = SignIn.auth;

Method 2 : signout()

Input : None.

Output : Login Scene.

1. string l = auth.CurrentUser.Email;
2. auth.SignOut();
3. Unity SceneManager loads the login screen.

**5.5 Class Name: FreeRoam**

Method 1 : start()

1. Calls startFreeRoam() method.

Method 2 : startFreeRoam()

Input : Controller

Output : Movements

1. Unity SceneManager loads the Free Roam scene.

**5.6 Class Name : ViewMetadata**

Method 1 : start()

Input : None

Output : None

1. FirebaseApp.DefaultInstance.SetEditorDatabaseUrl(“<<<database-url>>>”);
2. DatabaseReference reference = FirebaseDatabase.DefaultInstance.RootReference;

Method 2 : viewMetadata()

Input : None

Output : Metadata Panel

1. FirebaseDatabase.DefaultInstance

.GetReference("rooms").OrderByChild(<<room\_no>>)

.ValueChanged += HandleValueChanged.

Method 3 : HandleValueChanged()

1. if args.DatabaseError not equal to null
2. Console displays the args.DatabaseError.Message;
3. return;
4. End if;
5. Console displays the args.Snapshot.GetRawJsonValue().ToString();
6. mtext.text = args.Snapshot.GetRawJsonValue().ToString();

**5.7 Class Name : ReportBug**

Method 1 : start()

Input : None

Output : None

1. FirebaseApp.DefaultInstance.SetEditorDatabaseUrl<<database-url>>;

Method 2 : reportbug()

Input : report.

Output : Submitted panel and back to Free roam Scene

1. report = mField.text;
2. Bugreport b = new Bugreport(report);
3. string json = JsonUtility.ToJson(b);
4. Console displays the report;
5. DatabaseReference mDatabaseRef = FirebaseDatabase.DefaultInstance.RootReference;
6. mDatabaseRef.Child(BugReports).Child("eafvd").SetRawJsonValueAsync(json);

**5.8 Class Name : Feedback**

Method 1 : start()

Input : None

Output : None

1. FirebaseApp.DefaultInstance.SetEditorDatabaseUrl"<<<database-url>>";

Method 2 : feedback()

Input : note

Output : Submitted Panel and back to Free roam Scene

1. note = mField.text;
2. Bugreport b = new Bugreport(note);
3. string json = JsonUtility.ToJson(b);
4. Console displays the note;
5. DatabaseReference mDatabaseRef = FirebaseDatabase.DefaultInstance.RootReference;
6. mDatabaseRef.Child("Feedbacks").Child("eafvd").SetRawJsonValueAsync(json)

**5.9 Class Name : ViewRoomStructure**

Method 1 : view3D()

Input : None

Output : Loads the room view according to the objects present in the free roam

1. SceneManager.LoadScene("3D");

Method 2 : viewPhotosphere()

Input : None

Output : Loads the room view according to the objects present in the free roam

1. SceneManager.LoadScene("360Image");

Method 3 : Update()

Input : MouseInput

Output: Change in view

1. if (Input.GetMouseButton(0))
2. float rotateAboutX = Input.GetAxis("Mouse Y") \* Time.deltaTime \* rotateSpeed;
3. float rotateAboutY = -Input.GetAxis("Mouse X") \* Time.deltaTime \* rotateSpeed;
4. gameObject.transform.Rotate(rotateAboutX, rotateAboutY, 0.0f);
5. var newRotation = gameObject.transform.rotation.eulerAngles;
6. newRotation.z = 0;
7. gameObject.transform.rotation = Quaternion.Euler(newRotation);
8. end if