**Software Design Specification**

1. **Introduction**

The Software Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. Within the Software Design Document are narrative and graphical documentation of the software design for the project including use case models, sequence diagrams, collaboration models, object behaviour models, and other supporting requirement information.

1.1**Purpose of this document**

This document will define the design of the one runway simulator. It contains specific information about the expected input, output, classes, and functions. The interaction between the classes to meet the desired requirements are outlined in detailed figures at the end of the document.

1.2 **Scope of the development project**

We describe what features are in the scope of the software and what are not in the scope of the software to be developed.

In Scope:

a.    Music player works in Back ground while recorder is in use.

Out of Scope:

a.    This application does not have sharing option.

**1.3 Definitions, acronyms, and abbreviations**

**OS-Operating System**

**1.4 References**

**1.4.1** IEEE SDS template

**1.5 Overview of Document**

This SDS is divided into seven sections with various sub-sections. The sections of the Software Design Document are:

1. **Introduction**: describes about the document, purpose, scope of development project definitions and abbreviations used in the document.

2.  **Conceptual Architecture/Architecture Diagram:** describes the overview of components, modules, structure and relationships and user interface issues.

3. **Logical Architecture:** describes Logical Architecture Description and Components.

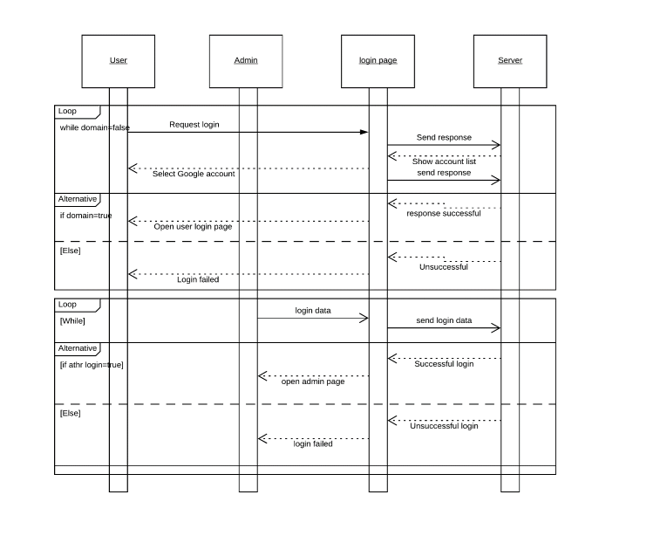
4. **Execution Architecture:** defines the runtime environment, processes, deployment view.

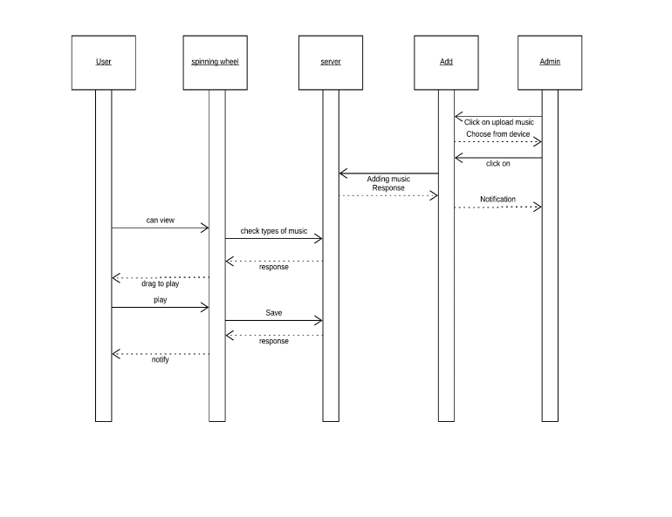
5. **Design Decisions and Trade-offs**: describes the decisions taken along with the reason as to why they were chosen over other alternatives.

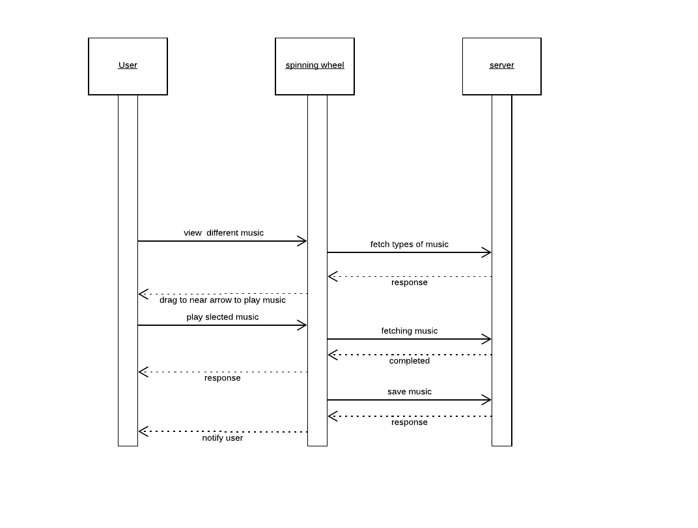
6. **Pseudocode for components:** describes pseudocode, as the name indicates. 7. Appendices: describes subsidiary matter if any.

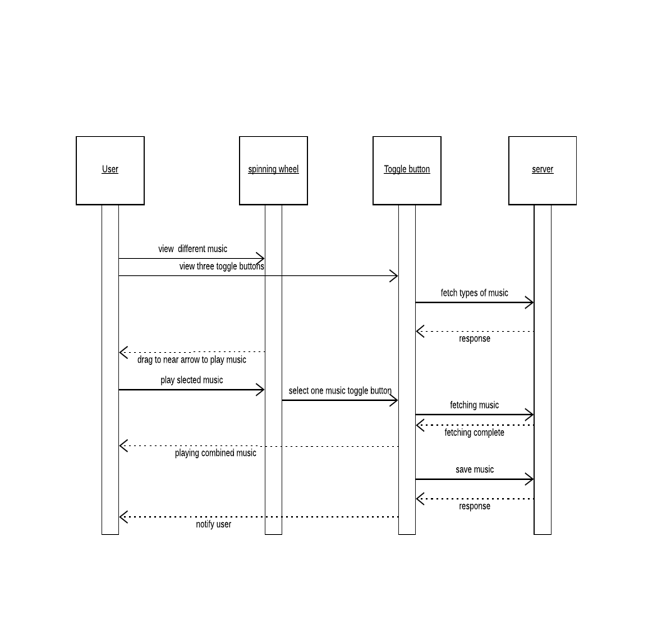
**3.Logical Architecture (Class Diagram, Sequence Diagram, State Diagram)**

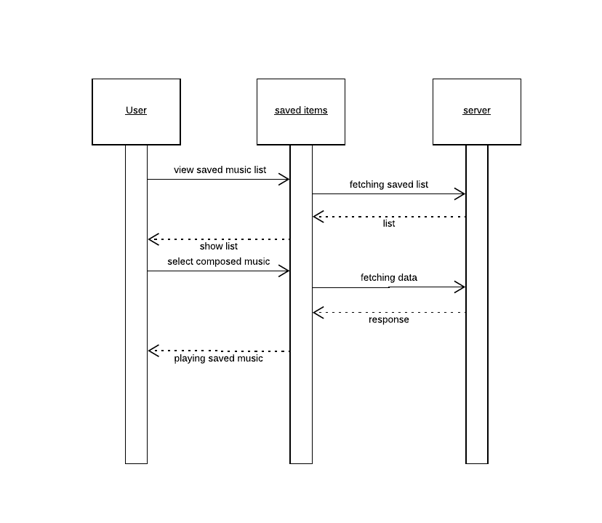
**Sequence Diagrams**:

****

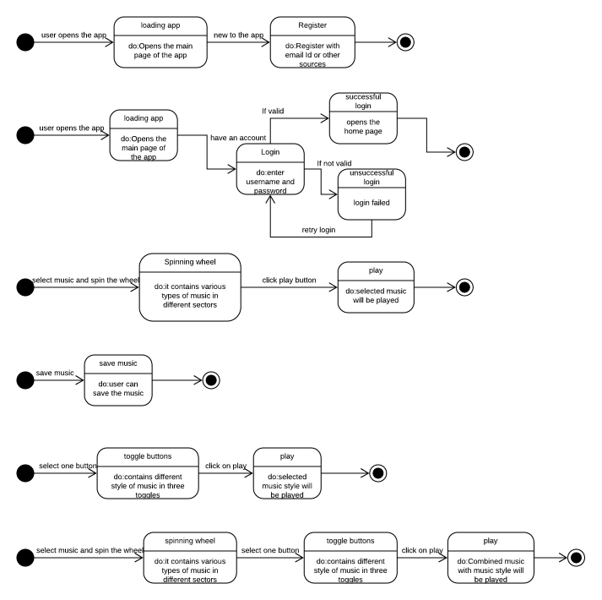
****

****

****

****

**State Diagrams:**

****

3.2 **Class name:**

**Login Description:** This class allows the user to enter the system by authenticating the entered credentials.

3.2.1 **Method 1**:

**OnStart()**

 Input  : mAuthListener

 Output : handle authentication state changes

Method Description: When activity start getting visible to user then onStart() will is called. Inside this method parent onStart constructor and addAuthSateListener is invoked. Since it is a Login activity so several state may occur like

 • Right after the listener has been registered

• When a user is signed in

 • When the current user is signed out

• When the current user changes Hence this method listen to the listener and handle authentication state changes.

3.2.2 **Method 2**:

**onCreate()**

Input : savedInstanceState , Email, Password

 Output: Launch the Activity, SignIn

Method Description: When an Activity first call or launched then onCreate(Bundle savedInstanceState) method is responsible to create the activity. When ever orientation(i.e. from horizontal to vertical or vertical to horizontal) of activity gets changed the object of Bundle class will save the state of an Activity. Basically Bundle class is used to stored the data of activity whenever above condition occur in app. SetContentView is used to fill the window with the UI provided from layout file. This method takes input as email id and password and as a result opens the landing page if login successful.

3.2.3 **Method 3:**

**onClick(View view)**

 Input: view – object of View class

 Output: userLogin is initiated

 Method Description: A View occupies a rectangular area on the screen and is responsible for drawing and event handling. View is the base class for widgets, which are used to create interactive UI components (buttons, text fields, etc.) . This method take parameter as object of the view class. When the view is equal to button sign in then userLogin method is called.

3.2.4 **Method 4**:

**UserSignIn()**

 Input: SignIn through Google

Output: User landing page of login successful

 Method Description: Intents are asynchronous messages which allow application components to request functionality from other Android components. This method allows student to login through their Google account. SignIn Intent stores the resultant value by checking whether the sign from Google is authorized login or not . If the user is not a Google client then the login will fail and if it is a authorized login then it will lead to user landing page.

3.2.5 **Method 5:**

**onAcitivityResult(int requestCode, int resultCode, Intent data)**

Input: requestCode, resultCode, object of Intent

Output: Landing to the Main Activity page

Method Description: Intent is used in order to communicate with the other components. Hence this method is used to get back to the original page after fetching the results. This method takes an input as requestCode, resultCode, intent data. It get result from GoogleAPI , if the result is successful then login in completed else it will display the message of authentication failure.

3.2.6 **Method 6:**

**firebaseAuthWithGoogle(GoogleSignInAccount account)**

Input: account

Output: Authentication with firebase

Method Description: After a user successfully signs in, get an ID token from the ‘GoogleSignInAccount’ object, exchange it for a Firebase credential, and authenticate with Firebase using the Firebase credential. This method allows authentication with firebase. It takes the input as account and register to the firebase.

3.3Class Name:   MainActivity-

This class enables the functions of the front end main page .

3.3.1 **Method 1:**

**OnCreate(Bundle savedinstancestate)**

 Input: bundle object

 Output: opens a layout

Description: Calls method ‘onCreate (saveInstanceState)’ of ‘super’ class and method ‘setContentView (layout\_name)’.

3.3.2 **Method 2:**

**initViews();**

   Input  :object of Cursor Wheel Layout

  Output :User defined objects

Description: Casting 'findViewById(R.id.Wheel\_text)' to 'CursorWheelLayout' .

**3.3.3 Method 3:**

**loadData():**

Input  : Image objects from object bundle

Output : Storing into a Array List.

   Description:This method create a memory to store all the images into the list and displayed after begin called by other method.

**3.3.4 Method 4:**

**onItemSelected(CursorWheelLayout parent, View view, int pos)**

Input  : parent,view,pos (Variables)

   Output : toast Message

Description : This method gives a toast message for every input from the user.

**3.4 Class Name :WheelImageAdapter**

**3.4.1 Method 1:**

**View getView(View parent, int position)**

Input  : parent,position

Output : returns to MainActivity and ImageData.

Description:Casting 'root.findViewById(R.id.wheel\_menu\_item\_iv)' to 'ImageView'  .

**3.4.2 Method 2:**

**ImageData getItem(int position)**

Input  :position

Output :returns to MainActivity and ImageData.

Description: Type Casting position with programmer defined position.

**3.4.3 Method 3:**

**getCount()**

     Input  :count

     Output : size of the count

Description : Maintains the count of images of the wheel.

**3.5 Class Name:  ImageData**

**3.5.1 Method 1: ImageData(int imageResource, String imageDescription)**

     Input  : imageResource,imageDescription

     Output : returns to other classes

Description: stores the above inputs in respective variables for future use by the other classes.

**4.0 Execution Architecture**

Runtime environment required is any device supporting Android Operating System with the minimum version of lollipop, Android Studio as a deployment platform.

**4.1 Reuse and relationships to other products**

NIL

**5.0 Design decisions and tradeoffs**

The basic decision structured to use spinning wheel as it basic feet rather than radio buttons or any other options. A toggle button allows the user to change a setting between two states. Tradeoffs for this particular decisions are spinning wheel is time saving becomes more user friendly and also improves appearance of the home page. It removes the ambiguity between choosing the style of music.

6.0 Pseudocode for components

6.0.1 **Class Name: Login**

Method 1: **OnStart()**

Pseudo-code:

Input: mAuthListener

Output: handle authentication state changes

1. super.onStart();

2. mAuth.addAuthStateListener(mAuthListener);

Method 2: **onCreate()**

Pseudo-code:

Input: savedInstanceState , Email, Password

Output: Launch the Activity, SignIn

1. super.onCreate(savedInstanceState);

2. setContentView(R.layout.activity\_main);

3. button = stores the ID of Google sign-in button

4. mAuth = get FirebaseAuth instance

5. editTextEmail = stores the ID of Admin editTextEmail

6. editTextPassword = stores the ID of Admin editTextPassword

7. buttonSignIn = stores the ID of Admin SignInButton

8. progressDialog = create new progressDialog object

9. button.setOnClickListener(new View.OnClickListener()

10. public void onClick(View v)

11. signIn() ) // Admin sign in

12. button.setOnClickListener(new View.OnClickListener()) // Google sign in

13. mAuthListener = new FirebaseAuth.AuthStateListener()

14. public void onAuthStateChanged( FirebaseAuth firebaseAuth )

15. if firebaseAuth.getCurrentUser() != null then

16. start another Activity 15. end if 12. if firebaseAuth.getCurrentUser() != null then 13. finish();

17. start another Activity

18. end if

Method 3: **onClick(View view)**

Pseudo-code:

Input: view – object of View class

Output: userLogin is initiated

1. if view=buttonSignIn then

2. userLogin()

3. end if

Method 5: **UserSignIn()**

Pseudo-code:

Input: SignIn through Google

Output: User landing page of login successful

1. Intent signInIntent = get signIn Intent from Google for the given Google client

2. if signInIntent==Google client then

3. start another activity leading to User landing page

4.  else

5. print message login unsuccessful

6. end if

Method 6: **onAcitivityResult(int requestCode, int resultCode, Intent data)**

Pseudo-code:

Input: requestCode, resultCode, object of Intent

Output: Landing to the Main Activity page

1. super.onActivityResult(requestCode, resultCode, data);

2. if requestCode == RC\_SIGN\_IN then

3. GoogleSignInResult result = get result from Google API about the SignIn from intent

4. end if

5. if result= Successful then

6. GoogleSignInAccount account = get SignIn account from GoogleAPI firebaseAuthWithGoogle(account)

7. end if

8. else

9. print authentication went wrong try again

Method 7: **firebaseAuthWithGoogle(GoogleSignInAccount account)**

Pseudo-code:

Input: account

Output: Authentication with firebase

1. AuthCredential credential = GoogleAuthProvider.getCredential(Id Token, null);

2. mAuth.signInWithCredential(credential)

3.  .addOnCompleteListener(this, new

OnCompleteListener() {

4. public void onComplete(@NonNull Task task)

{

5. if task is Successful then

6. Sign in success, update UI with the signed-in user's information

7.  FirebaseUser user = mAuth.getCurrentUser();

8. updateUI(user);

9. end if

10. else

11. If sign in fails, display a message to the user

6.0.2 Class Name:   MainActivity-

**Method 1:**

**OnCreate(Bundle savedinstancestate)**

Pseudo-code:

Input: bundle object

Output: opens a layout

1.protected void onCreate(Bundle savedInstanceState) {

2.       super.onCreate(savedInstanceState);

3.       setContentView(R.layout.activity\_main);

4.       initViews();

5.       loadData();

6.       wheel\_text.setOnMenuSelectedListener(this);

7.       wheel\_image.setOnMenuSelectedListener(this);

**Method 2:**

**initViews();**

Pseudo-code:

   Input  :object of Cursor Wheel Layout

  Output :User defined objects

1.private void initViews() {

2.       wheel\_image = 3.(CursorWheelLayout)findViewById(R.id.Wheel\_image);

4.        wheel\_text = 5.(CursorWheelLayout)findViewById(R.id.Wheel\_text);

**Method 3:**

**loadData():**

Pseudo-code:

Input  : Image objects from object bundle

Output : Storing into a Array List.

  lstText = new ArrayList<>();

      lstImage = new ArrayList<>();

      for (int i=0;i<9;i++)

         lstText.add(new MenuItemData("0" + i));

      lstText.add(new MenuItemData("OFF"));

      WheelTextAdapter adapter = new WheelTextAdapter(getBaseContext(),lstText);

      wheel\_text.setAdapter(adapter);

        lstImage = new ArrayList<>();

        lstImage.add(new ImageData(R.drawable.home,"bulb"));

        lstImage.add(new ImageData(R.drawable.wifi,"down"));

        lstImage.add(new ImageData(R.drawable.home,"home"));

        lstImage.add(new ImageData(R.drawable.home,"laugh"));

        lstImage.add(new ImageData(R.drawable.wifi,"wifi"));

        WheelImageAdapter imgAdapter = new WheelImageAdapter(getBaseContext(),lstImage);

        wheel\_image.setAdapter(imgAdapter);

}

**Method 4:**

**onItemSelected(CursorWheelLayout parent, View view, int pos)**

Pseudo-code:

Input  : parent,view,pos (Variables)

   Output : toast Message

public void onItemSelected(CursorWheelLayout parent, View view, int pos) {

if (parent.getId() == R.id.Wheel\_text)

Toast.makeText(getBaseContext(),"Selected : "+lstText.get(pos).mTitle,Toast.LENGTH\_SHORT).show();

else if(parent.getId() == R.id.Wheel\_image)

Toast.makeText(getBaseContext(),"Selected : "+lstImage.get(pos).imageDescription,Toast.LENGTH\_SHORT).show();

}

6.0.3 **Class Name: WheelImageAdapter**

**Method 1:**

**View getView(View parent, int position)**

Pseudo-code:

Input  : parent,position

Output : returns to MainActivity and ImageData.

public View getView(View parent, int position) {

ImageData data = getItem(position);

View root = inflater.inflate(R.layout.wheel\_image\_layout,null,false);

ImageView imageView = (ImageView)root.findViewById(R.id.wheel\_menu\_item\_iv);

imageView.setImageResource(data.imageResource);

return root;

}

**Method 2:**

**ImageData getItem(int position)**

Pseudo-code:

Input  :position

Output :returns to MainActivity and ImageData.

public ImageData getItem(int position) {

return menuItems.get(position);

}

**Method 3:**

**getCount()**

Pseudo-code:

     Input  :count

     Output : size of the count

public int getCount() {

return menuItems.size();

}

6.0.4  **Class Name:  ImageData**

**Method 1: ImageData(int imageResource, String imageDescription)**

Pseudo-code:

     Input  : imageResource,imageDescription

     Output : returns to other classes

public ImageData(int imageResource, String imageDescription) {

this.imageResource = imageResource;

this.imageDescription = imageDescription;

}