A Design Specification Document for the course CS5617 (Software Engineering)

Design Specification Document

by

Jasir K

UX Team

(Neel Kabra)

(Parichita Das)

(Jasir K)

Under the supervision of

Ramaswamy Krishnan Chittur

INDIAN INSTITUTE
OF TECHNOLOGY
PALAKKAD

Department of Computer Science and Engineering

Kerala, India - 678557

HomeScreen Spec Document

Jasir K - 111901025

---

1. Overview

The UX team for this project is responsible for joining all the UX components of all other teams. This document is dedicated to the home screen of the application. After starting the application, there will be a splash screen which will just open up for a second showing our application logo and some quotes. Which is followed by the Authentication using Google sign-in and checking if the user is a student of IIT Palakkad. After the Authentication is done we will be redirected to the **Home Screen,** and all the data (Name, profile picture, Email ID, Roll No) will be passed from the authentication page . This screen will have the following features :-

A. Host a New Meeting
   a. Makes the current system as server
   b. Provides a button to host a meeting
   c. Generates sessionID - Unique for each session
   d. Generates UserID - We will always maintain the servers/hosts UserID as 1.
   e. Will pass on the name, profile picture, userID, sessionID to the dashboard module.

B. Join a meeting
   a. Takes the server's IP address(and the port number), and the sessionID (for validation if students are in the correct meeting or not)
   b. Get the Name, EmailID from the Authentication screen.
   c. Provides a button to join the meeting

    d. Send all the relevant details to the dashboard module.

C. Retrieve session (for Cloud Module)
    a. A button would be provided named as “sessions” which will redirect to a new screen from where we will be able to access the previous sessions and their details, which will be implemented by the cloud module. And in that page we will be having a button to return to the home screen.

D. Profile Edits
    a. User should be able to upload or change his profile picture which would be passed on to the dashboard.
    b. Take input from the user for Server’s IP, Server’s Port Number and SessionID.
    c. User should not be able to change his email details but it should be shown in the profile details.

E. Light/Dark Mode
    a. There will be two themes for the application - Light theme and a dark theme.
    b. There will be a toggle button at the bottom right to change between these themes and this mode will also be passed to the dashboard team.

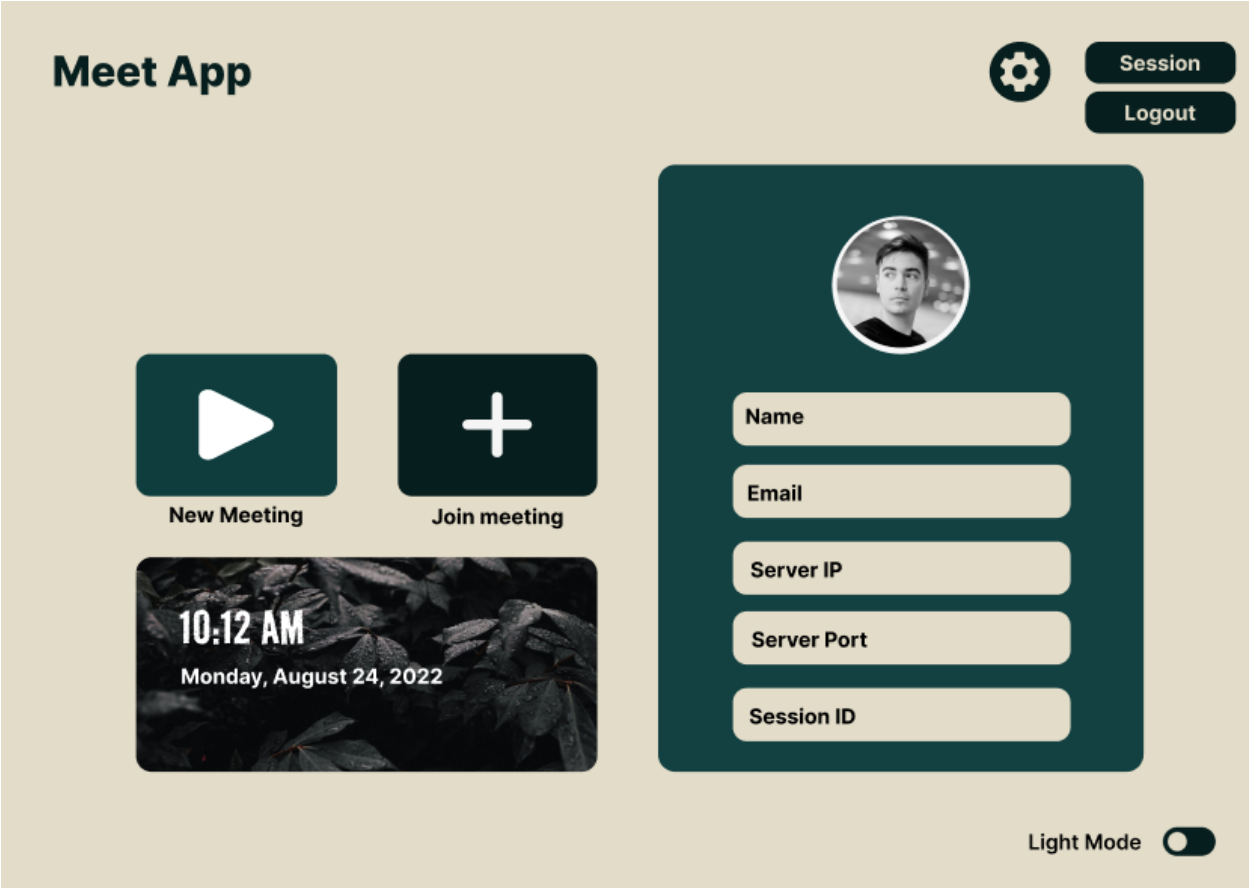
F. Date and Time section
    a. A section showing the current date and time in an elegant manner which would be far more convenient to the user to when to join the meet.

G. Logout Button
    a. A button which should be able to logout from the google authentication and redirect login page.

H. Settings - If required then only.

a. We will be supporting shortcut keys such as “Ctrl+tab” and “Ctrl+Shift+tab”. If any user wants to change these shortcuts then he can use the settings button.
b. An option to change the date and time format.

The Layout of the HomeScreen is shown below -



Prototype design for the Application has been made. Link is given below :-
https://www.figma.com/proto/6OcWQH82ImzvPBnYOXRogk/MeetApp?node-id=22%3A82&scaling=scale-down&page-id=0%3A1&starting-point-node-id=22%3A82

2. Objective

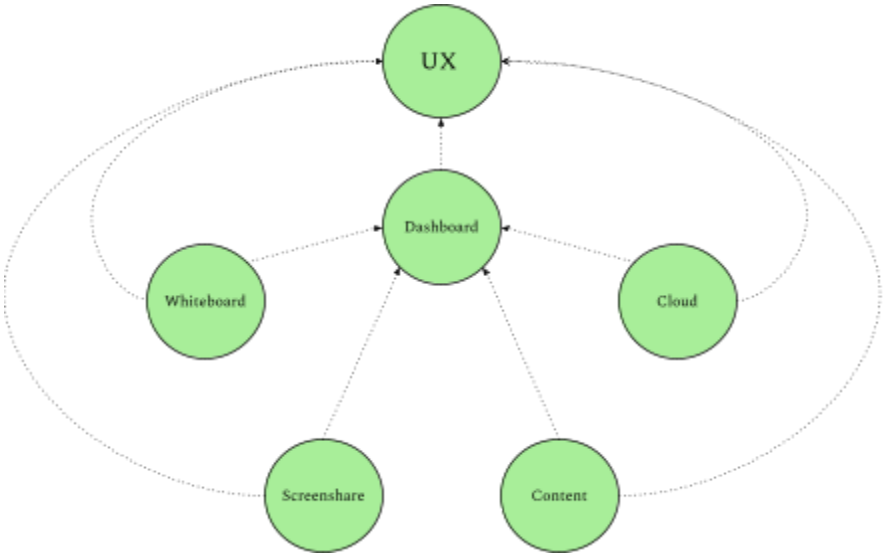
The main objectives of the Home screen is :-

- After validation of the user who belongs to our organization, we will be getting input from students/clients to enter the host IP and port number and the sessionID and even update the profile picture, so it can be passed to the dashboard team for further validation of meeting credentials.
- A user friendly UI/UX, with keyboard shortcuts and option to change the theme. Implementing HomeScreenView and HomeScreenViewModel which will be used for binding the click buttons to redirect to dashboard or cloud session details page.

3. UML

In this section we will discuss the different types of interactions between different sections of the entire codebase.

3.1 Module Diagram

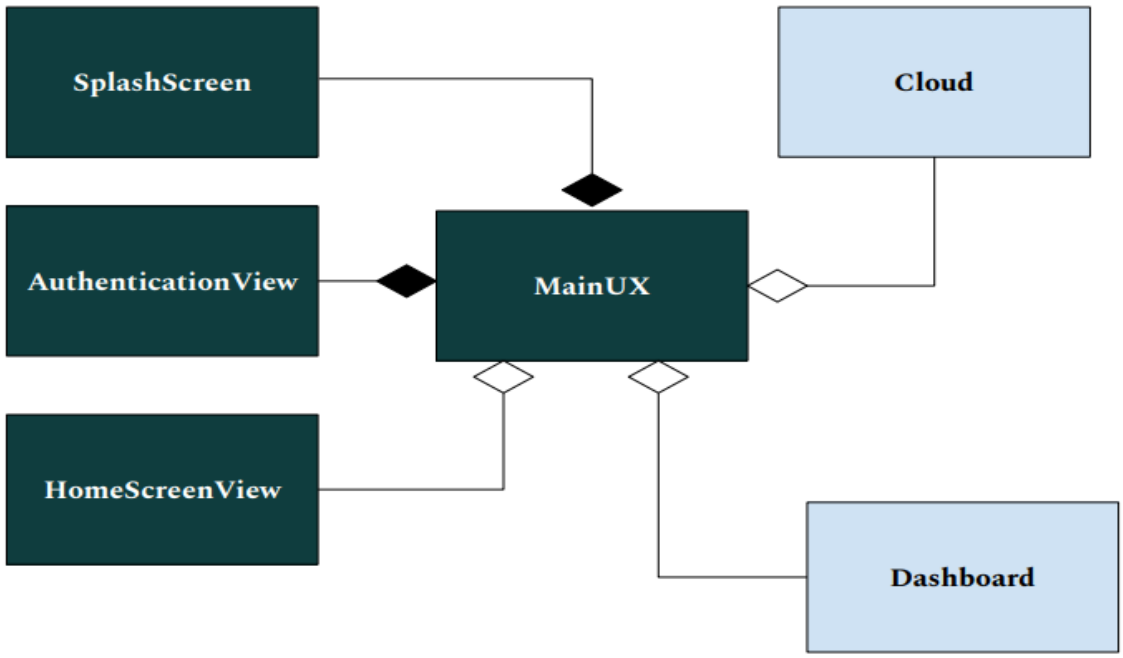


As we can see in the picture above, all of the modules other than Networking are dependent on the UX module. Once it is instantiated, the dashboard is in charge of constructing the other modules. The various modules then send us each of their unique UIs, which we then show in a Navigation Tab Layout.

3.2 Class Diagram

We will be having two different class diagrams. Out of which one will be for overall UX, and other one be specific to the HomeScreen part.

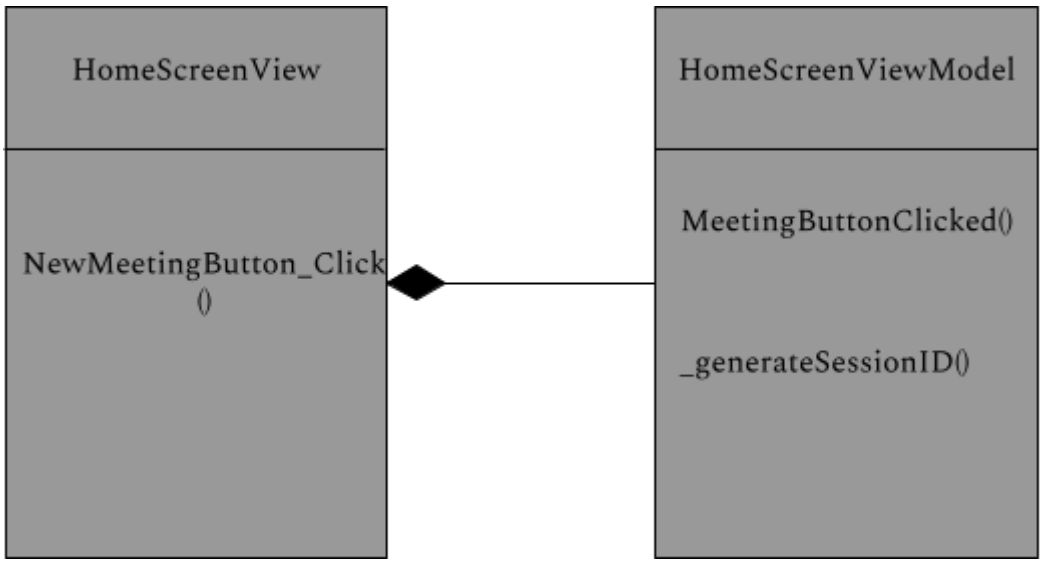
3.2.1 Main UX Class Diagram



MainUX is composed of the SplashScreen and AuthenticationView as both of them must be constructed if MainUX gets instantiated. The user information struct or object is returned to the MainUX once authentication is complete. The user might not successfully authenticate, in which case the MainUX will not continue. In the alternative scenario, the MainUX will now build an object called HomeScreenView and send user data to it. In the home screen, if the user joins or

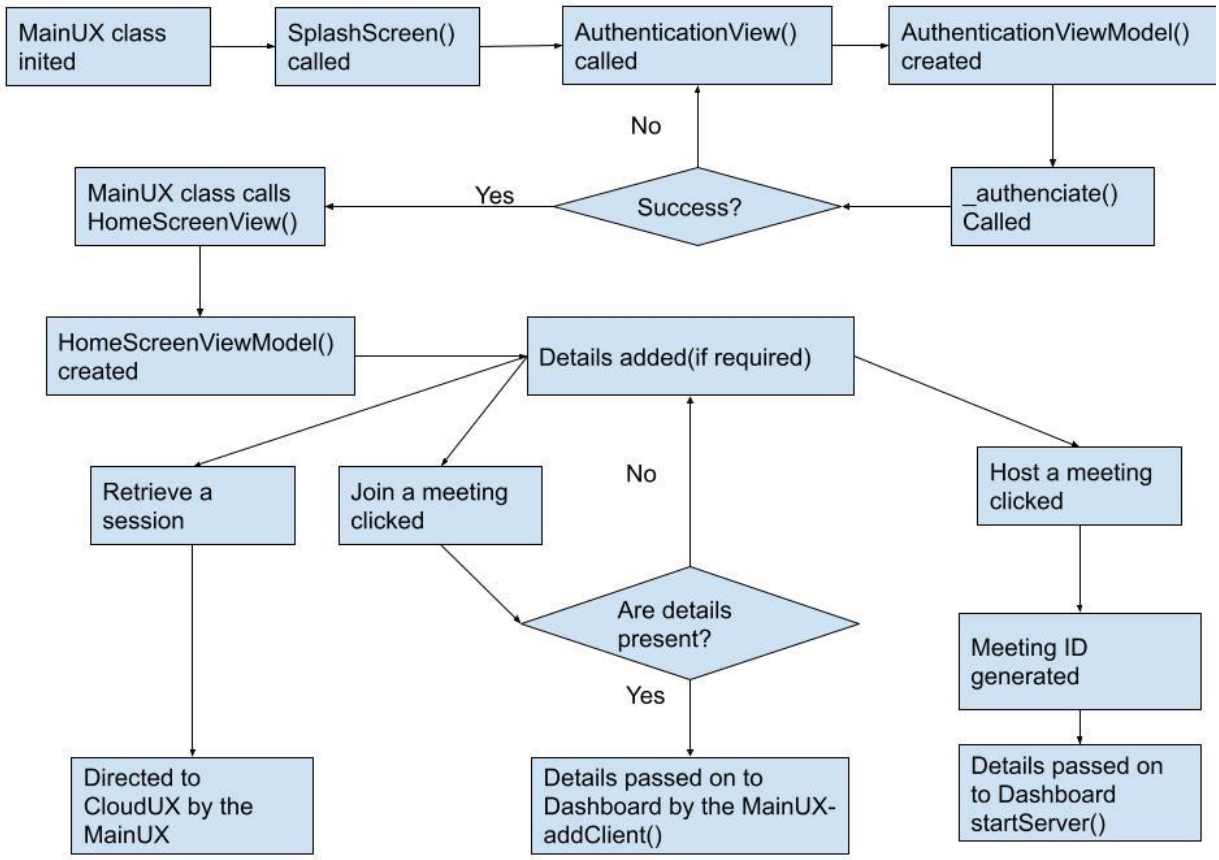
hosts a new meeting, the Home Screen module sends the necessary information back to the MainUX and requests that it build a Dashboard object. The user can even retrieve his old session details using the session button, which will be redirected to the cloud UI page. In this way the MainUX aggregates HomeScreen, Dashboard and Cloud Modules.

3.2.2 HomeScreen Module Class Diagram



We are using a Model-View-Viewmodel Design Pattern, for the HomeScreen part, we only have a view and a view model. The view must compose the View Model, and likewise the View Model must bind to the View. And further all these details of the user would be passed on to the Dashboard team via MainUX.

3.2.2 Activity Diagram for UX module



Activity Diagram

4. Design

We are using an MVVM design pattern as if it gives a clean and structured code. We can easily extend our project and test our code or debug. And more on like we will have a clear understanding of how they are binded.

4.1 View and View Model

We have the XAML file containing the button for *NewMeeting, JoinMeeting, SessionDetails, Logout and Toggle between light and dark mode* which will be in the

HomeScreenView. In this class we will be binding the *HomeScreenViewModel* class object to itself.

```
public HomeScreenView() {
      HomeScreenViewModel = new HomeScreenViewModel();
      this.DataContext = viewModel;
}
```

Home Screen won't be instantiating any model, we will be simply sending the data to the MainUX which will pass on to the Dashboard.

We will be having a *SessionButton_Click* which will pass on the user details as a list to the MainUX and which will create a CloudSession object and pass on a **list of details.**

Similarly we will be having a *JoinMeetingButton_Click* which will pass on the user details as a list and it will make sure the users has taken **IP address, port number and SessionID** as input in the text space. If not it will notify the user that those fields are empty and they are mandatory to fill.
In this we will be having a NewMeetingButton_Click which should notify the view model that we need to generate a SessionID.

```
private void NewMeetingButton_Click(object sender, RoutedEventArgs e)
{

  HomeScreenViewModel viewModel = this.DataContext as HomeScreenViewModel;
  viewModel.MeetingButtonCLicked = true;
}
```

In the view model we have a *_generateSessionID* function, which would generate a 6 letters random code.

```
public string HomeScreenViewModel
{
      return _generateSessionID();
}
```

We will be having a structure for users which should have the following variable and this structure should be passed on to the MainUX.

```
struct User
{
     public string Name;
     public string Email;
     public string UUID;
     public string Server_IP;
     public string Server_port_number;
     public string SessionID;
     public string Picture_Address;
     public string theme;
}
```

5. Interfaces

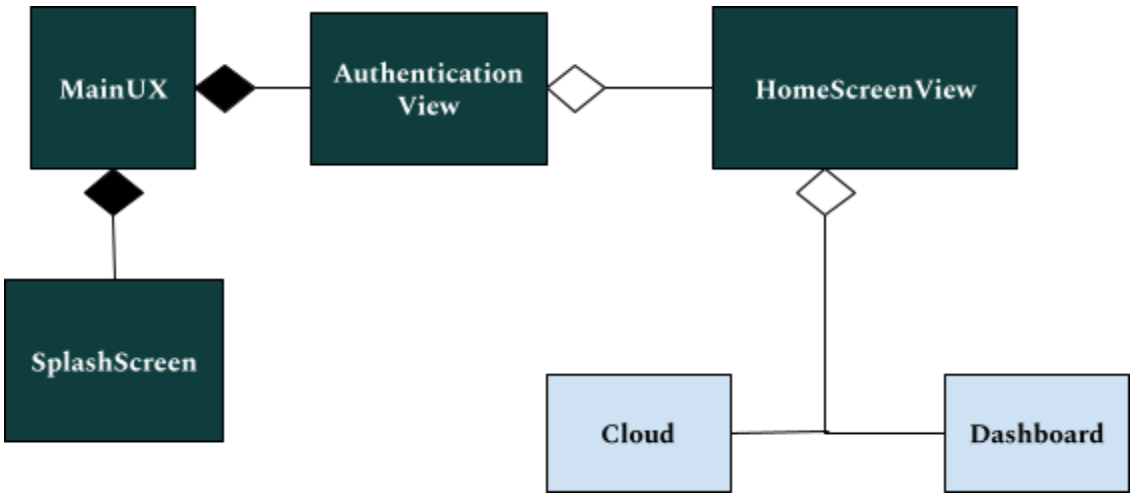
We do not use any *Interfaces* for HomeScreen.

6. Analysis

We did analysis for different design patterns like MVC, MVP and MVVM for UX. Out of these we chose the MVVM model since it was extensible and testable. The code modules we write makes it easy to debug. That is why we are preferring it over other design patterns.

6.1 Class Diagram

In addition to the previously described method, we had another idea on how to handle endpoints in our module. It goes like this:



In this case, the MainUX builds the Authentication Module, which then calls the Home Screen Module directly, which calls the Cloud or the Dashboard directly in turn.

However, we employ the previously indicated approach, wherein the MainUX serves as the endpoint for all other modules. In this manner, the MainUX has complete control and serves as a distinct module for the other modules.

8. Conclusion

From the HomeScreen Module, either the control can be passed down to the cloud module or the dashboard module through the MainUX. If it is passed on to the cloud module then they will have the option to view all their previous session details and select and explore more about the specific session. And they will be having a button to redirect to the homescreen from their UI page. If the control is shifted to the dashboard then it will validate the IP, port Number and Session ID of the meeting and if it fails it will redirect to the home screen. Home screen is an important part of the application.