Collaborative University Android App

WAYNE STATE
UNIVERSITY

Product Design Specifications Document | Version 1.2
February 25, 2020

Team: Adeel Asghar, Tyler Gross, Palak Patel, and Hala Ali
Clients: Thomas Anter and Jahnu Best
GTA: Son Dang

VERSION HISTORY

Version	Author	Date	Description
1.0	Hala Ali	2/10/20	Initial Design Definition draft
1.1	Adeel Asghar	2/21/20	Formatting
1.2	Adeel, Hala, Tyler, Palak	2/24/20	Added diagrams and finished document. Completion of Design Document Final Draft

TABLE OF CONTENTS

1. INTRODUCTION

The introduction section of the Product Design Specification Document will cover the purpose of developing such a document and the significance the document holds for all parties involved. This includes the development team, the client, and other stakeholders.

1.1 Purpose of the Product Design Specification Document

The Product Design Specification document documents and tracks the necessary information required to effectively define architecture and system design. It provides the development team with guidance on the architecture of the system to be developed. The Product Design Specification document is created during the Planning Phase of the project. Its intended audience is the project manager, project team, and development team. Some portions of this document such as the user interface (UI) may on occasion be shared with the client/user, and other stakeholder whose input/approval into the UI is needed.

2. GENERAL OVERVIEW & DESIGN GUIDELINES/APPROACH

This section describes the principles and strategies to be used as guidelines when designing and implementing the system.

2.1 Assumptions / Constraints / Standards

2.1.1 Assumptions

1. The users have an android mobile device which runs an operating system of at least Android 5.0 or an emulator with an API level of 21. In the case of the user not having access to either of the mentioned technologies, our product will not run.
2. The user must be in ownership of a google email address and have knowledge of their username and password. The system currently supports users resetting their passwords by sending users a link via email. If the user does not have access to their email, they will not be able to recover their account.

2.1.2 Technical Constraints

1. The app is targeted for Android devices.
2. MVVM is a technical constraint.
3. Firebase limits nonpaying users to 1Gb of stored data, 50,000 document reads, 20,000 document writes, and 20,000 document deletes.

2.1.3 Business Constraints

1. The Firebase Realtime Database server must always be available.
2. Due to the self-funded nature of the project, there will be no funds allocated to buying the firebase Realtime database spark plan. In recognition of this the application must have under 100 simultaneous connections, less than 1GB of data stored, and under 10 GBs downloaded per month. The application must also only have one Firebase database
3. The requirements for this project must be achieved by 4/14/2020.

2.1.4 Design Constraints

1. The application design should adhere to the MVVM architectural design pattern. Data from the database will flow through models, repositories, view models, and into the views.
2. In the case of dependent classes, the application should adhere to the dependency inversion principle. All class dependencies should be passed from the outside of

functions using dependency injection. To implement dependency injection the team will be using the Dagger dependency injection framework

3. Data from the database that is displayed to the user as well as data taken in by the user interface will be bound between the view models and the XML layout files. The activities in the application will not contain any data related functions.

2.1.3 Standards

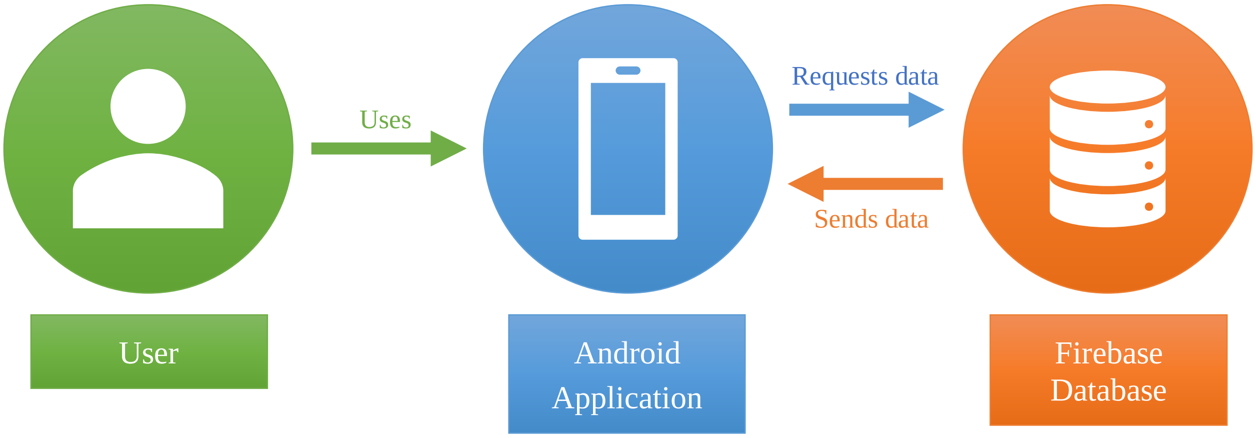
1. All features of the application must be thoroughly tested before release.
2. The UI of the application must be easily usable by the products target demographic.

3. ARCHITECTURE DESIGN

This section outlines the University Collaborative Forum system and hardware architecture design of the system that is being built.

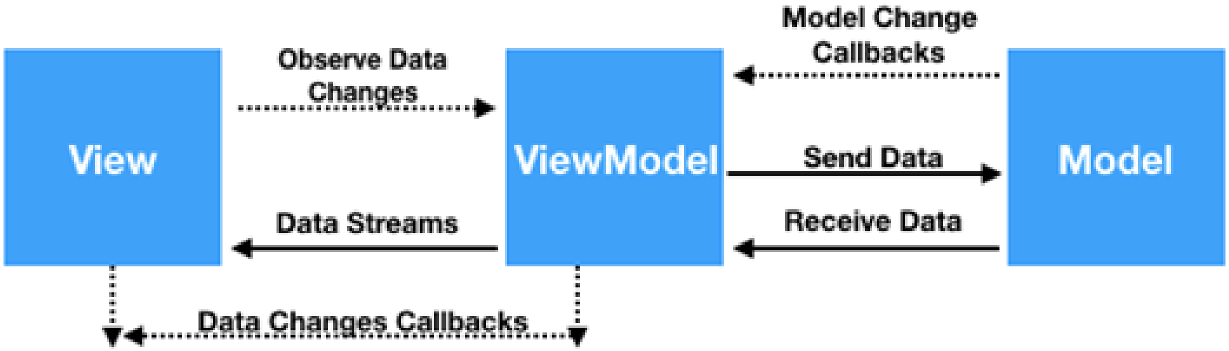
3.1 Hardware Architecture

The application will have 3 main components: the user, the Android device, and Firebase Realtime Database. The user will use the Android Device to open the Collaborative University Application. The application will request authentication or data from the Firebase Realtime Database which will send back that authentication or data to display in the application. The following diagram displays the University Collaborative Application interactions within a larger system.



3.2 Software Architecture

The following diagram displays the architectural design of the University Collaborative Application. The software architectural pattern used is MVVM (Model-View-View Model). The three components of the MVVM are models, views and view models. The View is displayed to the user. The view model passes data to and from the view to the model. The model contains data containers that are populated with data from the database.

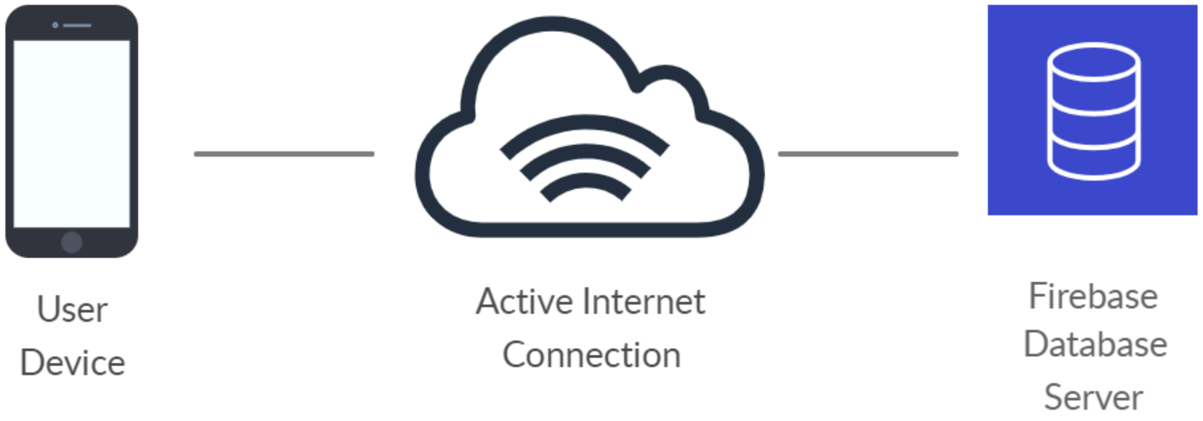


3.3 Security Architecture

The security architecture for this application is done through Firebase Authentication which supports registration and login for valid users. Registration is done using an email, username, and password. These passwords are hashed through a Firebase script. Login can then be done using a username and password which are passed to the Firebase Authentication SDK. The SDK verifies those credentials using its backend services and returns a response to our client. Users can then read and write to the database but only delete posts that they have made themselves.

3.4 Communication Architecture

The communication architecture of this application is dependent on the connection to the internet of the cellar device being used to run the application. The application must be connected to the internet to access the Firebase Realtime Database. The application depends on Firebase for both user authentication and storage.



3.5 Performance

The time it takes for the app to open once a user has clicked on it will be 200 milliseconds.

After a user registers their information correctly and hits register button, it will take anywhere between 200 to 500 milliseconds to redirect to the login page. Once the user has typed in their email and password and hits login button, it should take anywhere between 500 milliseconds to 1 second to direct user to main forum page.

4. SYSTEM DESIGN

The following section goes over the use cases, sequence diagrams, dataflow diagrams, database design, class diagram, and application program interface.

4.1 Use-Cases

4.1.1 Use Case 1: User Registration	
Created by	Palak Patel
Date Created	2/18/2020
Last Updated by	Palak Patel
Date of last revision	2/24/2020
Actors	Application User
Description	This use case will describe a user registering for an account.
Trigger	The user wants to register for an account.
Preconditions	The user does not already have an account.
Post conditions	The username, email and password will be evaluated by the application through Firebase User Authentication. If the username, email or password is not correct formatted then they user will be prompted to re-enter their information. (UC-1)
Normal Flow	1.The user selects the I do not have an account button. 2.The user enters their username, email and password. 3.The user selects the register button. 4.the user is redirected to the login page.
Alternative Flows	Alt Flow: username, password or email formatting parameters are not met. 1.The user enters invalid information. 2.The user is prompted with an error message, returns to UC-1. Alt Flow: User already has an account with that email address. 1.The user enter an email that is already in use 2. The user is prompted with an error message, returns to UC-1.
Frequency of Use	Once per account.
Assumptions	

4.1.2 Use Case 2: User Login

Created by	Palak Patel
Date Created	2/18/2020
Last Updated by	Palak Patel
Date of last revision	2/24/2020
Actors	Application User
Description	This use case will describe a user logging into the application.
Trigger	The user wants to login to their account.
Preconditions	The user has not already logged in.
Post conditions	The email and password will be evaluated by the application through Firebase User Authentication. If the email or password is not recognized the user will be prompted to re-enter their information. If the email and password are both correct the user is redirected.
Normal Flow	1.The user enters their username and password. 2.The user selects the login button. 3.The user is redirected to the main forum page.
Alternative Flows	ALT FLOW: The login information cannot be found by Firebase Authentication. 1.The user enters invalid information, returns to UC-2.
Frequency of Use	Every time a user logs out of the application.
Assumptions	There is an active internet connection and Firebase Authentication is Available

4.1.3 Use Case 3: Subscribe to a Class Forum	
Created by	Palak Patel
Date Created	2/18/2020
Last Updated by	Palak Patel
Date of last revision	2/24/2020
Actors	Application User
Description	This use case will describe a user subscribing to a class forum.
Trigger	The user wants to subscribe to a class forum.
Preconditions	The user is logged in.
Post conditions	The posts from the forum that the user subscribed to will now show on their main forum page.
Normal Flow	1.The user navigates to the all classes screen. 2.The user selects the subscribe button for their chosen class.
Alternative Flows	
Frequency of Use	This will be used whenever a user would like to subscribe to a forum.
Assumptions	There is an active internet connection and the firebase Realtime Database is available.

4.1.4 Use Case 4: Unsubscribe from Class	
Created by	Palak Patel
Date Created	2/18/2020
Last Updated by	Palak Patel
Date of last revision	2/24/2020
Actors	Application User
Description	This use case will describe a user unsubscribing for a class.
Trigger	This user wants to unsubscribe from a class forum.
Preconditions	The user is logged in and subscribed to the class that they wish to unsubscribe from.
Post conditions	The posts from the unsubscribed forum will disappear from that user’s main forum page.
Normal Flow	1.The user navigates to the all classes screen. 2.The user selects the un-subscribe button for their chosen class.
Alternative Flows	
Frequency of Use	Every time the user wishes to no longer view posts from a specific forum.
Assumptions	There is an active internet connection and the firebase Realtime Database is available.

4.1.5 Use Case 5: Create Post	
Created by	Palak Patel
Date Created	2/18/2020
Last Updated by	Palak Patel
Date of last revision	2/24/2020
Actors	Application User
Description	This use case will describe a user creating a post.
Trigger	The user wants to create a post.
Preconditions	The user is logged in and has navigated to the main forum page.
Post conditions	The post will show up on the user's main forum page, their profile page as well as in the forum that they posted to.
Normal Flow	1.The user navigates to the create post screen. 2.The user chooses the forum that the wish to create their post in. 3.The user enters the title of their post. 4.The user enters the content of their post. 5.The user selects the add post button.
Alternative Flows	
Frequency of Use	Every time the user wishes to make a post.
Assumptions	There is an active internet connection and the firebase Realtime Database is available.

4.1.6 Use Case 6: Delete Post	
Created by	Palak Patel
Date Created	2/18/2020
Last Updated by	Palak Patel
Date of last revision	2/24/2020
Actors	Application User
Description	This use case will describe a user deleting a post.
Trigger	The user wants to delete a post.
Preconditions	The user must be logged as well as successfully created the post that they wished to delete.
Post conditions	The post will no longer appear in the forum it was posted to, the users main forum screen or their user profile screen.
Normal Flow	1.The user navigates to their profile screen. 2.The user located the post they wish to delete by scrolling through their posts on their profile. 3.The user selected the delete post button. 4.The post is no longer visible because it has been deleted.
Alternative Flows	
Frequency of Use	Every time the user decides to delete a post.
Assumptions	There is an active internet connection and the firebase Realtime Database is available.

4.1.7 Use Case 7: Create Comment	
Created by	Palak Patel
Date Created	2/19/2020
Last Updated by	Palak Patel
Date of last revision	2/24/2020
Actors	Application User
Description	This use case will describe a user adding a comment to a post.
Trigger	The user wants to add a comment.
Preconditions	The user must be logged in.
Post conditions	A new comment will appear under the post that the user decided to comment on.
Normal Flow	1.The user navigates to their main forum screen or to a specific forum. 2.The user locates the post by scroll through the list on any of the previously described screens. 3.The user selects add comment. 4.The user is prompted with a text field and they enter their comment. 5.The user selects post comment.
Alternative Flows	
Frequency of Use	Every time the user wishes to comment on their own post or another student's post.
Assumptions	There is an active internet connection and the firebase Realtime Database is available.

4.1.8 Use Case 8: Delete Comment	
Created by	Palak Patel
Date Created	2/19/2020
Last Updated by	Palak Patel
Date of last revision	2/24/2020
Actors	Application User
Description	This use case will describe a user deleting a post comment.
Trigger	The user wants to delete a comment.
Preconditions	The user must be logged in and have successfully made the post that they wish to delete.
Post conditions	A comment that the user intended to delete will be removed from all screens displaying that comment.
Normal Flow	1.The user navigates to their profile screen, main forum screen or to a specific forum. 2.The user locates a comment that they made wish to delete by scrolling through the list. 3.The user selects delete comment. 4.The comment is no longer visible anywhere in the application
Alternative Flows	
Frequency of Use	This will be used every time the user wishes to delete their own comment on a post.
Assumptions	There is an active internet connection and the firebase Realtime Database is available.

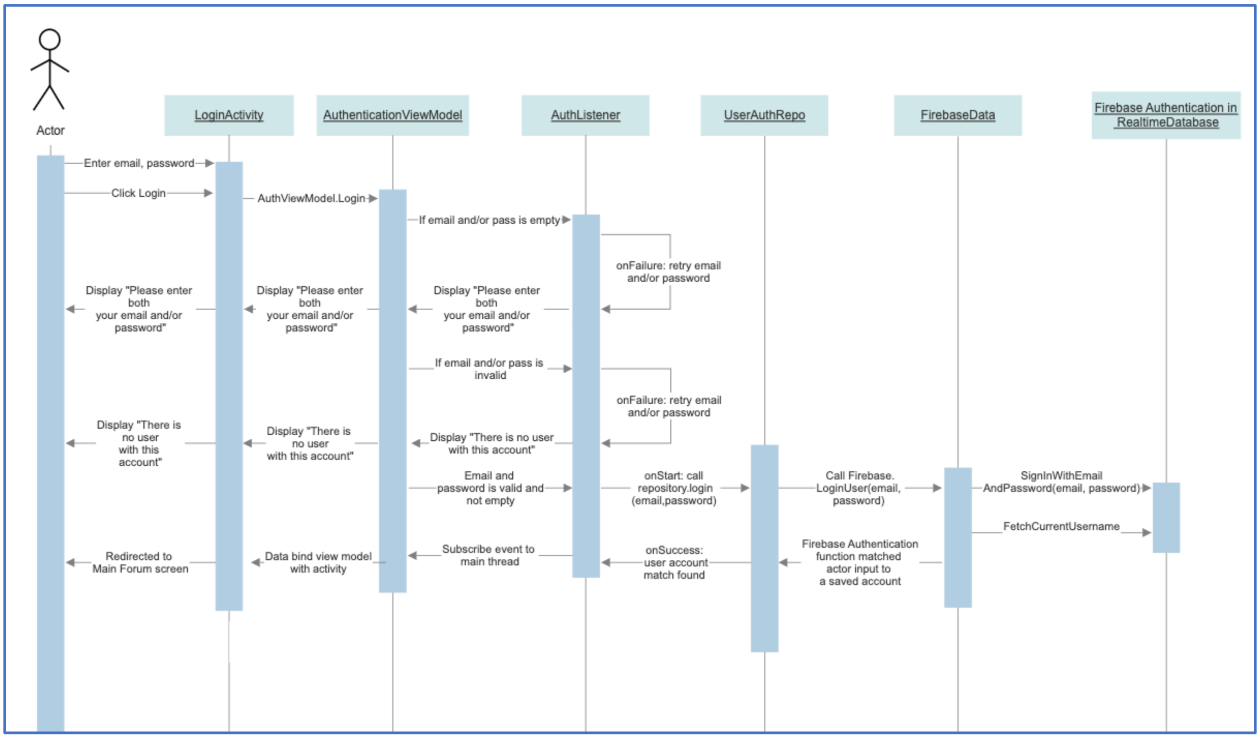
4.1.9 Use Case 9: Searching for a Forum by Subject and Class Title	
Created by	Palak Patel
Date Created	2/19/2020
Last Updated by	Palak Patel
Date of last revision	2/24/2020
Actors	Application User
Description	This use case will describe a user searching for a forum.
Trigger	The user wants to search for a specific forum.
Preconditions	The user must be logged in.
Post conditions	The user will be able to view all posts from the forum they were searching for.
Normal Flow	1.The user navigates to the all classes page. 2.The begins the search by entering the subject name. 3.The user enters class title name. 4.The user selects the enter button. 5.The user is redirected to that forums screen.
Alternative Flows	1.The user navigates to the all classes page. 2.The begins the search by entering the subject name. 3.The user enters class title name. 4.The user selects the enter button. 5.If the user entered the username or class title incorrectly, they will be prompted with an error message. Returns to UC-9.
Frequency of Use	This will be used every time the user wishes to search for a class.
Assumptions	There is an active internet connection and the firebase Realtime Database is available.

4.1.10 Use Case 10: Reset Password	
Created by	Palak Patel
Date Created	2/19/2020
Last Updated by	Palak Patel
Date of last revision	2/24/2020
Actors	Application User
Description	This use case will describe a user resetting their password.
Trigger	The user wants to reset their password.
Preconditions	The user must already have an account registered with an email address that they can log on to.
Post conditions	The user will be able to log in with their new password.
Normal Flow	1.The user navigates to reset password screen. 2.The use enters their email address. 3.The user logs into the email address they just entered and selects the email from Firebase. 4.The user clicks on the link in the email and is navigated to a reset password page. 5.The user enters their new password.
Alternative Flows	1.The user navigates to the reset password screen. 2.The user enter an invalid email. 3.The user is prompted with an error message to re-enter their email. Return to UC-10.
Frequency of Use	This will be used every time the user forgets their password.
Assumptions	There is an active internet connection and the firebase Realtime Database is available.

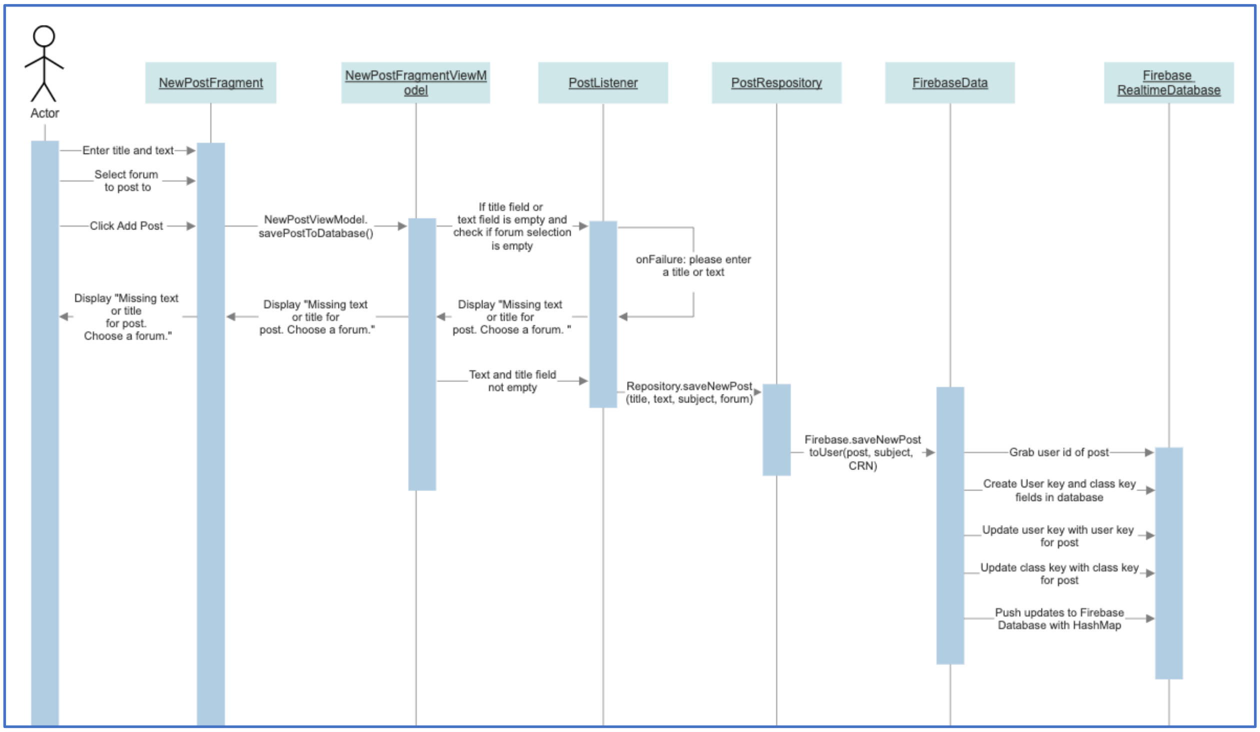
4.2 Sequence Diagram

The Sequence Diagram section of the Product Design Specification document will consist of three sequence diagrams that model significant functionalities found in the software product that are of high-level priority. These diagrams will model how the software product interacts with the user and updates itself accordingly from the following UI changes made. The three diagrams that are in this section include the “Login” sequence diagram, the “Add Post” sequence diagram, and the “Subscribe to Class Forum” sequence diagram.

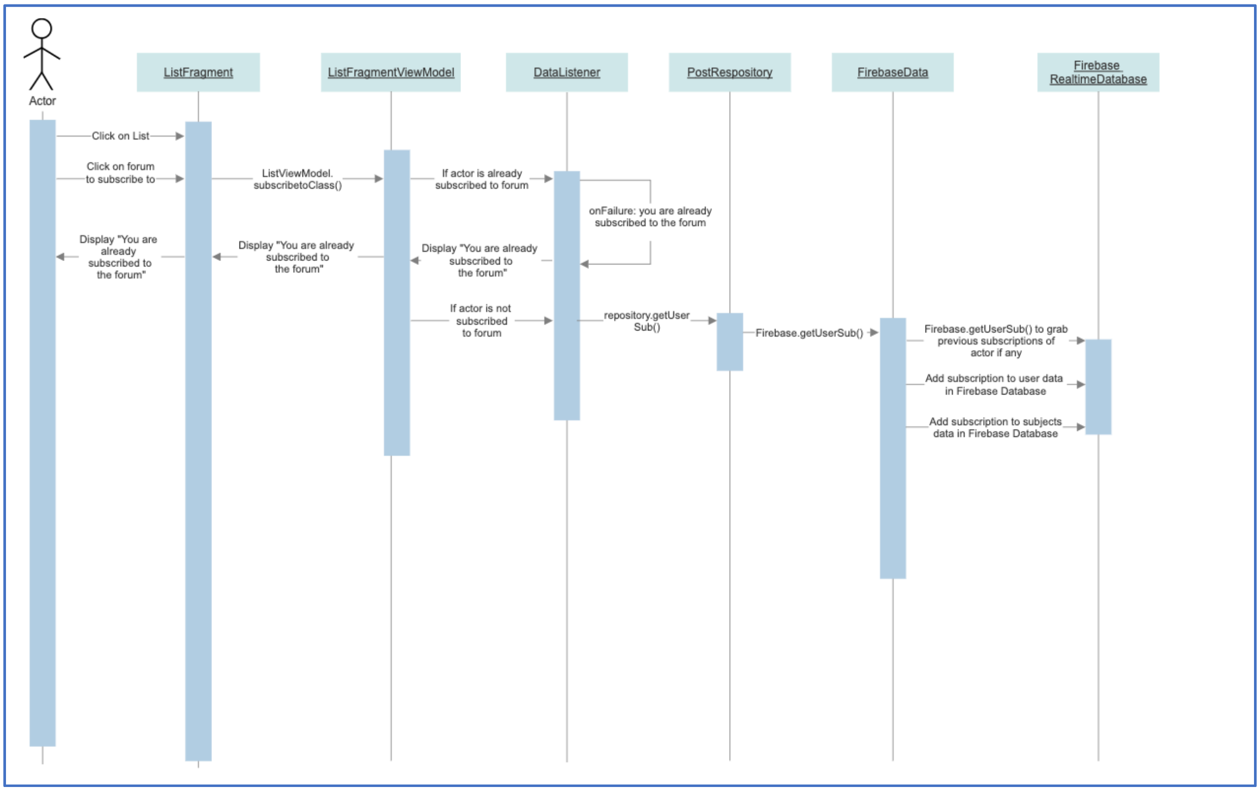
4.2.1 Login Sequence Diagram


Actor
LoginActivity
AuthenticationViewModel
AuthListener
UserAuthRepo
FirebaseData
Firebase Authentication in RealtimeDatabase
Enter email, password
Click Login
AuthViewModel.Login
If email and/or pass is empty
onFailure: retry email and/or password
Display "Please enter both your email and/or password"
Display "Please enter both your email and/or password"
Display "Please enter both your email and/or password"
If email and/or pass is invalid
onFailure: retry email and/or password
Display "There is no user with this account"
Display "There is no user with this account"
Display "There is no user with this account"
Email and password is valid and not empty
onStart: call repository.login (email,password)
Call Firebase. LoginUser(email, password)
SignInWithEmail AndPassword(email, password)
FetchCurrentUsername
Firebase Authentication function matched actor input to a saved account
onSuccess: user account match found
Subscribe event to main thread
Data bind view model with activity
Redirected to Main Forum screen

4.2.2 Add Post Sequence Diagram


Actor
NewPostFragment
NewPostFragmentViewModel
PostListener
PostRespository
FirebaseData
Firebase RealtimeDatabase
Enter title and text
Select forum to post to
Click Add Post
NewPostViewModel. savePostToDatabase()
If title field or text field is empty and check if forum selection is empty
onFailure: please enter a title or text
Display "Missing text or title for post. Choose a forum. "
Display "Missing text or title for post. Choose a forum."
Display "Missing text or title for post. Choose a forum."
Text and title field not empty
Repository.saveNewPost (title, text, subject, forum)
Firebase.saveNewPost toUser(post, subject, CRN)
Grab user id of post
Create User key and class key fields in database
Update user key with user key for post
Update class key with class key for post
Push updates to Firebase Database with HashMap

4.2.3 Subscribe to Class Forum Sequence Diagram

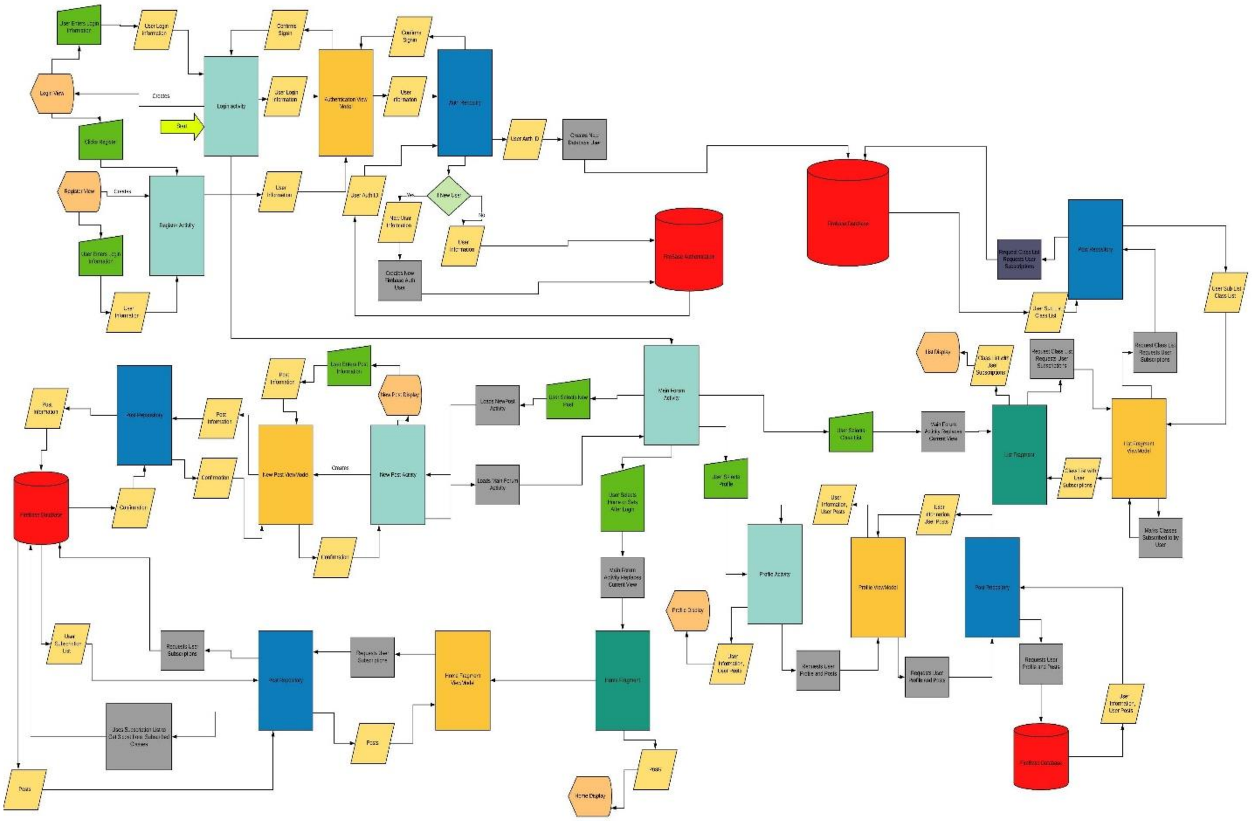

Actor
ListFragment
ListFragmentViewModel
DataListener
PostRespository
FirebaseData
Firebase RealtimeDatabase
Click on List
Click on forum to subscribe to
ListViewModel. subscribetoClass()
If actor is already subscribed to forum
onFailure: you are already subscribed to the forum
Display "You are already subscribed to the forum"
Display "You are already subscribed to the forum"
Display "You are already subscribed to the forum"
If actor is not subscribed to forum
repository.getUser Sub()
Firebase.getUserSub()
Firebase.getUserSub() to grab previous subscriptions of actor if any
Add subscription to user data in Firebase Database
Add subscription to subjects data in Firebase Database

4.3 Data Flow Diagram

The Data Flow Diagram section will consist of 5 data flow diagrams that will explain each of the complex requirements of the software product: overarching application, user registration, user login, loading posts to the class forum, and creating new posts.

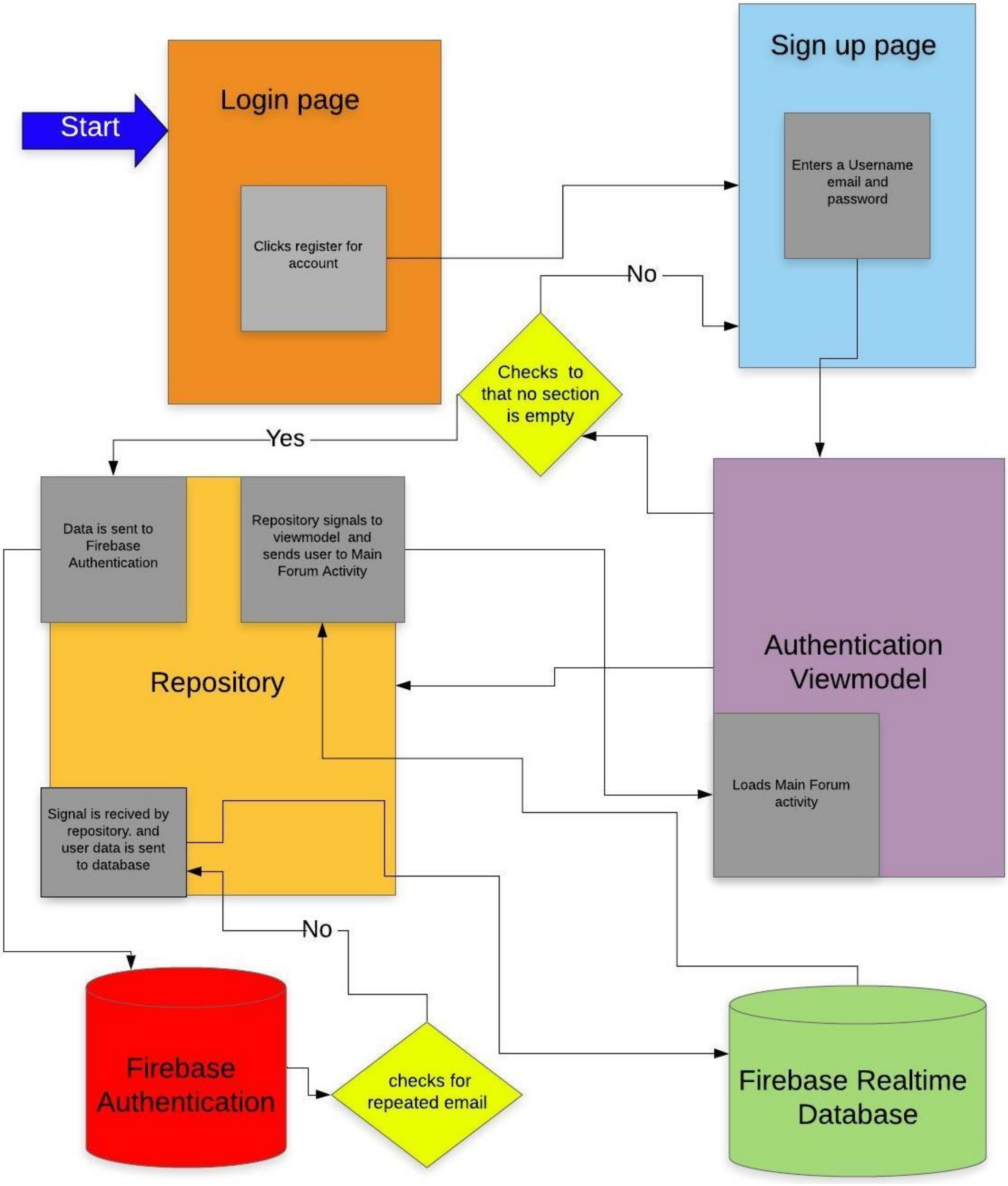
4.3.1 Application Data Flow Diagram (DFD)

The DFD shows the data flow of the entire application. The start of the data flow begins with the login activity screen and shows how the data is updated into the Firebase Realtime Database.



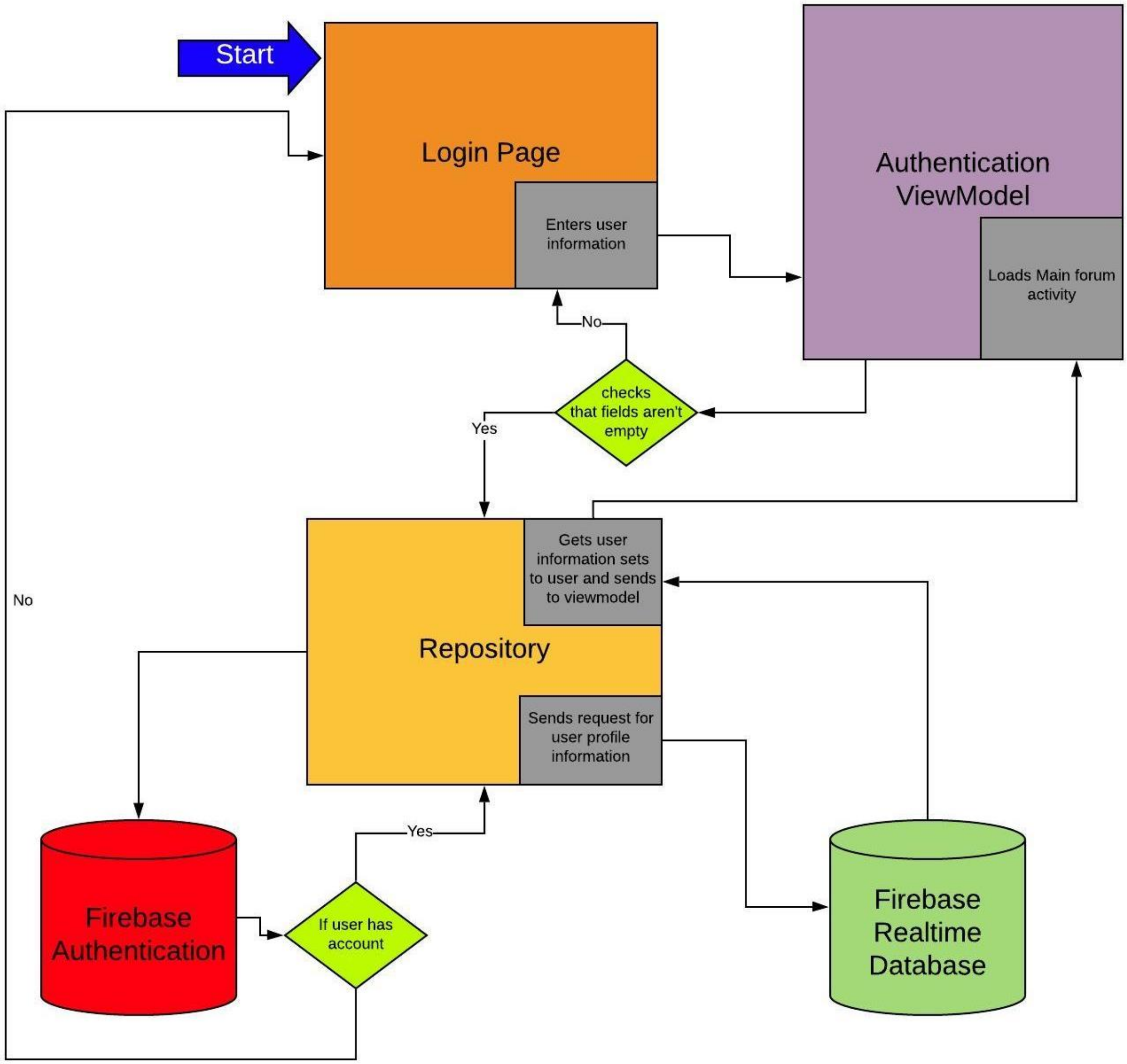
4.3.2 Registration Data Flow Diagram (DFD)

The DFD shows how the system processes the user registration. The registration process is shown interacting with the registration activity, the authentication view model, the repository class, the Firebase Authentication, and the Firebase Realtime Database.


Start
Login page
Clicks register for account
Sign up page
Enters a Username email and password
No
Checks to that no section is empty
Yes
Data is sent to Firebase Authentication
Repository signals to viewmodel and sends user to Main Forum Activity
Repository
Authentication Viewmodel
Loads Main Forum activity
Signal is recived by repository. and user data is sent to database
No
Firebase Authentication
checks for repeated email
Firebase Realtime Database

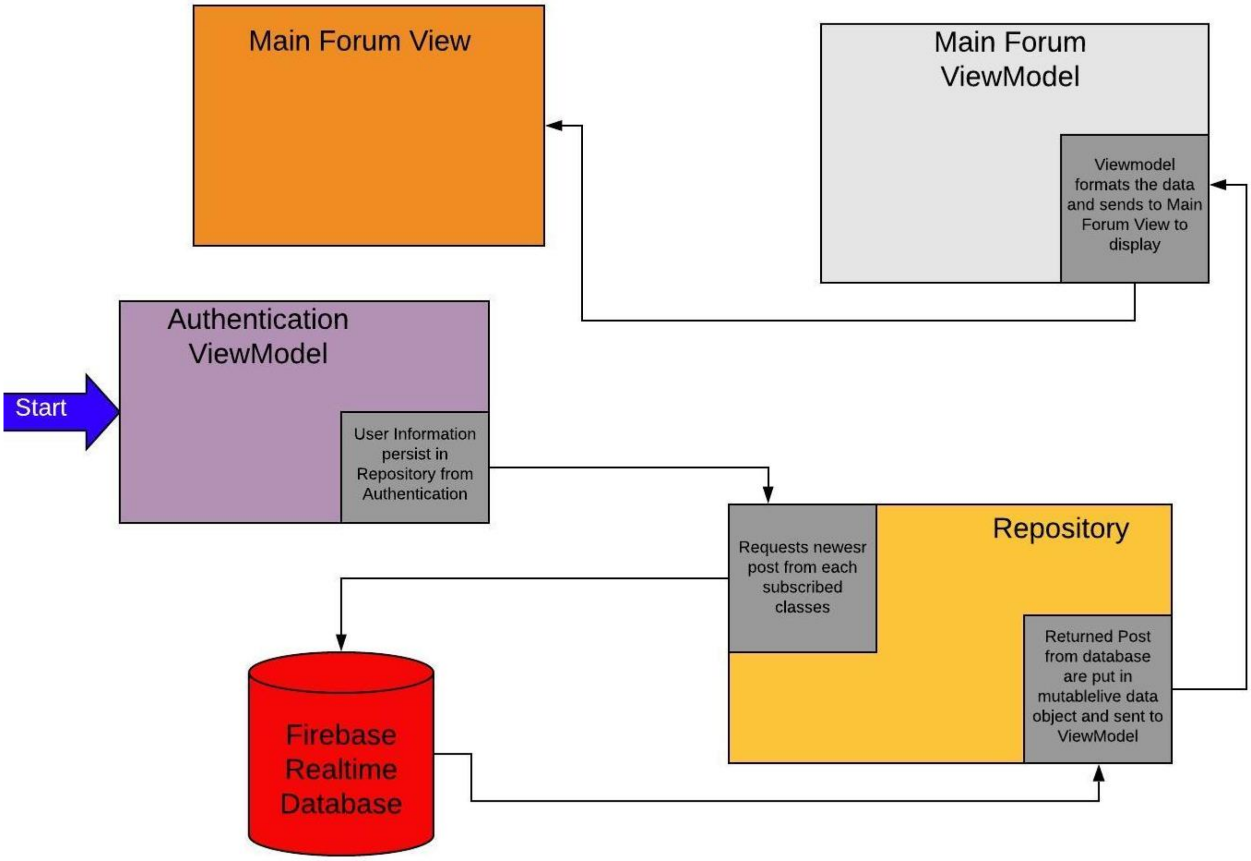
4.3.3 Log in Data Flow Diagram (DFD)

The DFD shows how the system processes the log in. The login process is shown interacting with the log in activity, the authentication view model, the repository class, the Firebase Authentication, and the Firebase Realtime Database.


Start
Login Page
Enters user information
Authentication ViewModel
Loads Main forum activity
No
checks that fields aren't empty
Yes
No
Repository
Gets user information sets to user and sends to viewmodel
Sends request for user profile information
Yes
Firebase Authentication
If user has account
Firebase Realtime Database

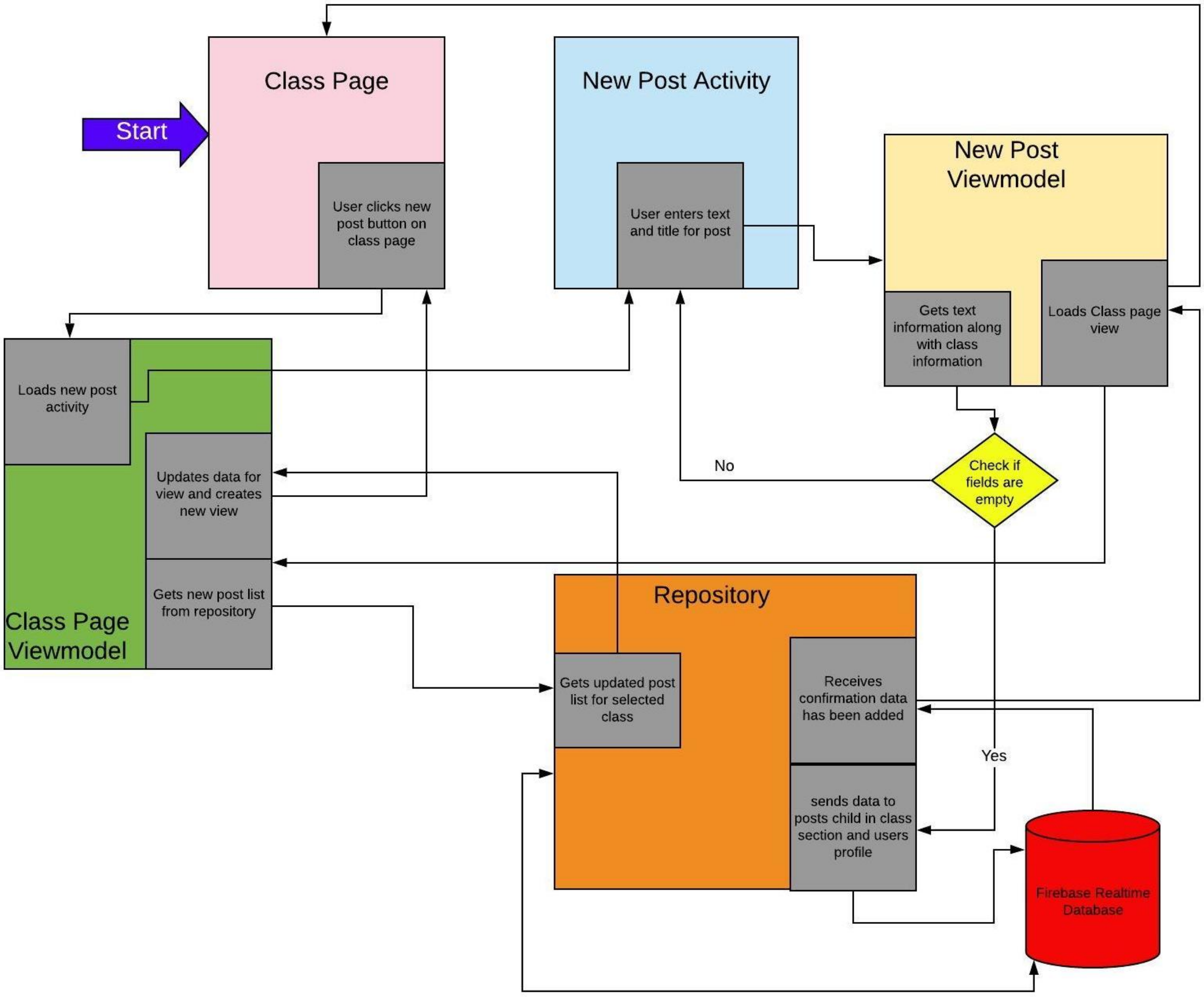
4.3.4 Load Posts Flow Diagram (DFD)

The DFD shows how the system processes loading posts for the user to see on the main forum. The load posts process is shown interacting with the new posts activity, the view model, the repository class, and the Firebase Realtime Database.

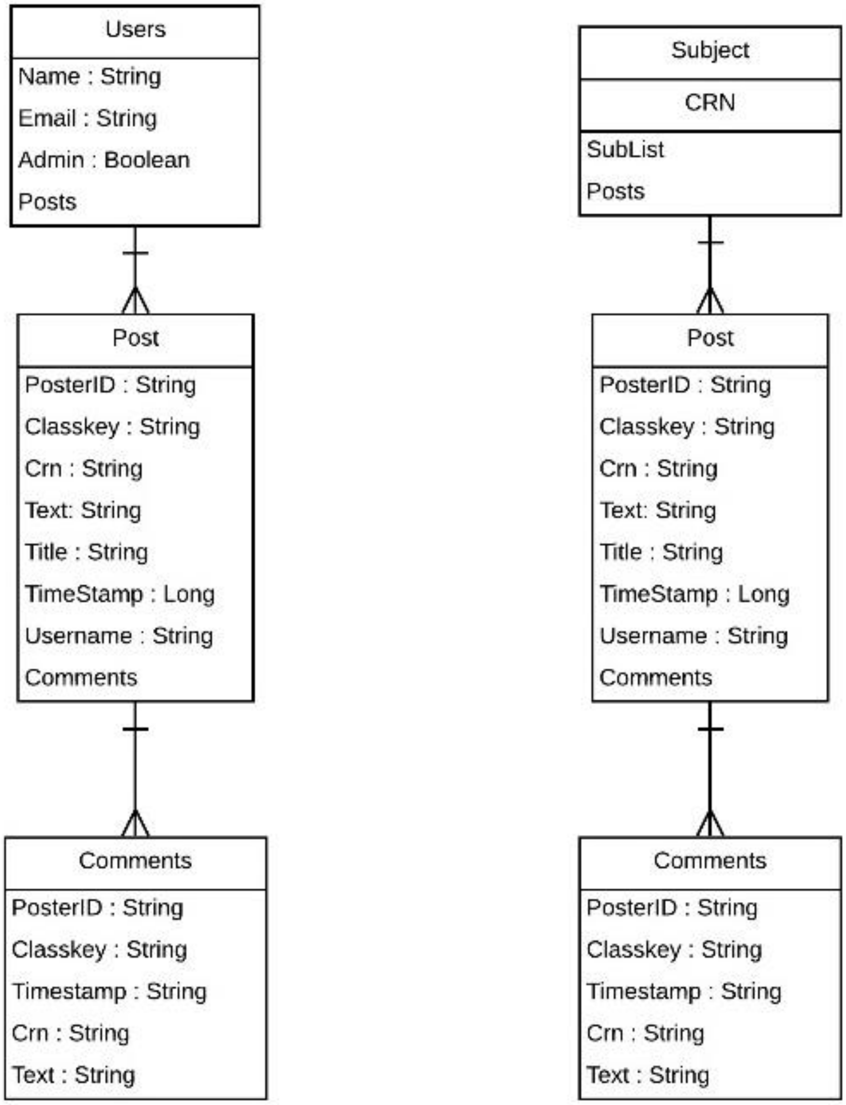

Main Forum View
Main Forum ViewModel
Viewmodel formats the data and sends to Main Forum View to display
Authentication ViewModel
Start
User Information persist in Repository from Authentication
Repository
Requests newesr post from each subscribed classes
Returned Post from database are put in mutablelive data object and sent to ViewModel
Firebase Realtime Database

4.3.5 New Post Data Flow Diagram (DFD)

The DFD shows how the system processes the user creating a new post. The load posts process is shown interacting with the new posts activity, the view model, the repository class, and the Firebase Realtime Database.


Class Page
Start
User clicks new post button on class page
New Post Activity
User enters text and title for post
New Post Viewmodel
Gets text information along with class information
Loads Class page view
Loads new post activity
Updates data for view and creates new view
Gets new post list from repository
Class Page Viewmodel
No
Check if fields are empty
Yes
Repository
Gets updated post list for selected class
Receives confirmation data has been added
sends data to posts child in class section and users profile
Firebase Realtime Database

4.4 Database Design

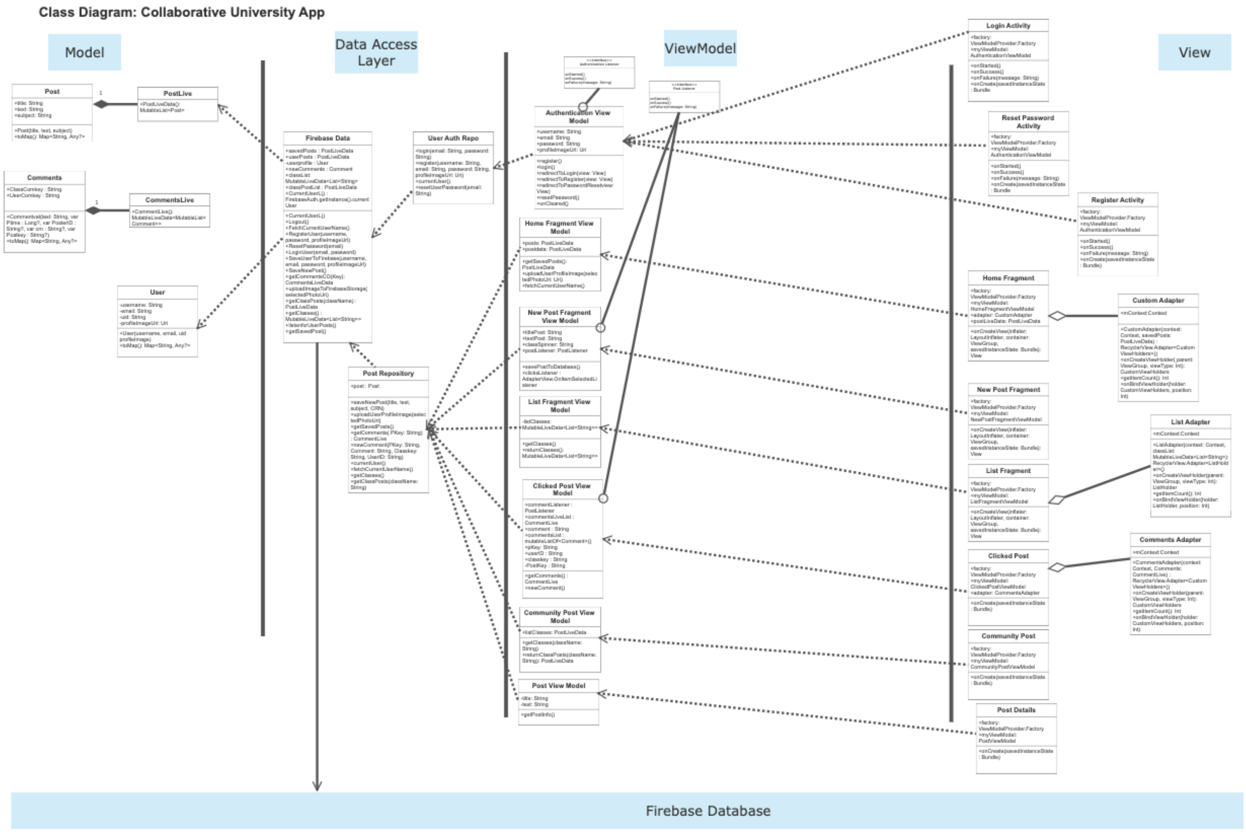


4.5 Class Diagram

The Class diagram includes two diagrams. The application class diagram covers the general structure of the entire system. Another one is a smaller class diagram that explains how the binding done with Dagger dependency injection framework works and how the view models interact with view model factory etc.

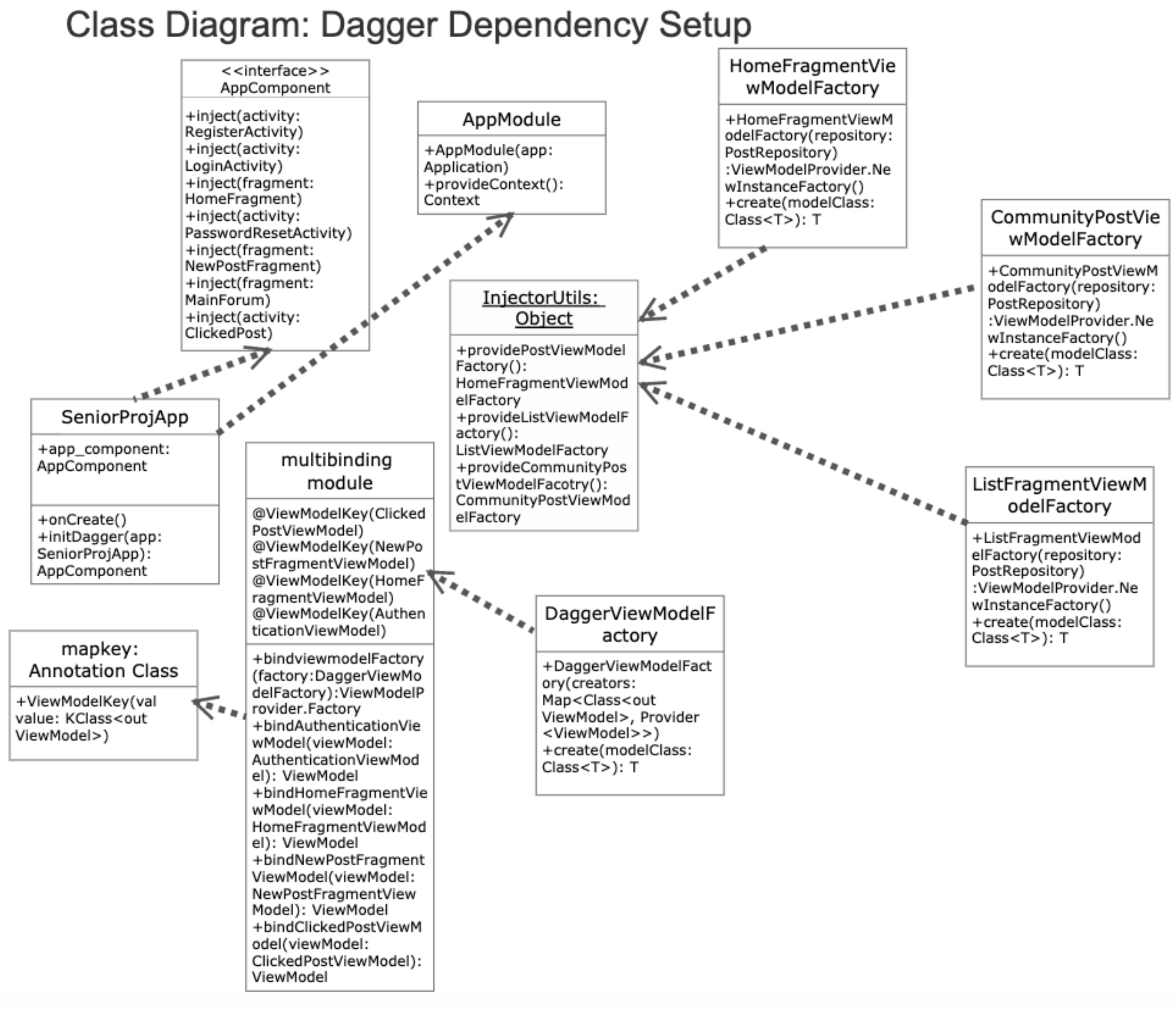
4.5.1 Application Class Diagram

The over-arching class diagram shows the entire system design of the software product using the MVVM design pattern.



4.5.2 Dagger Dependency Class Diagram

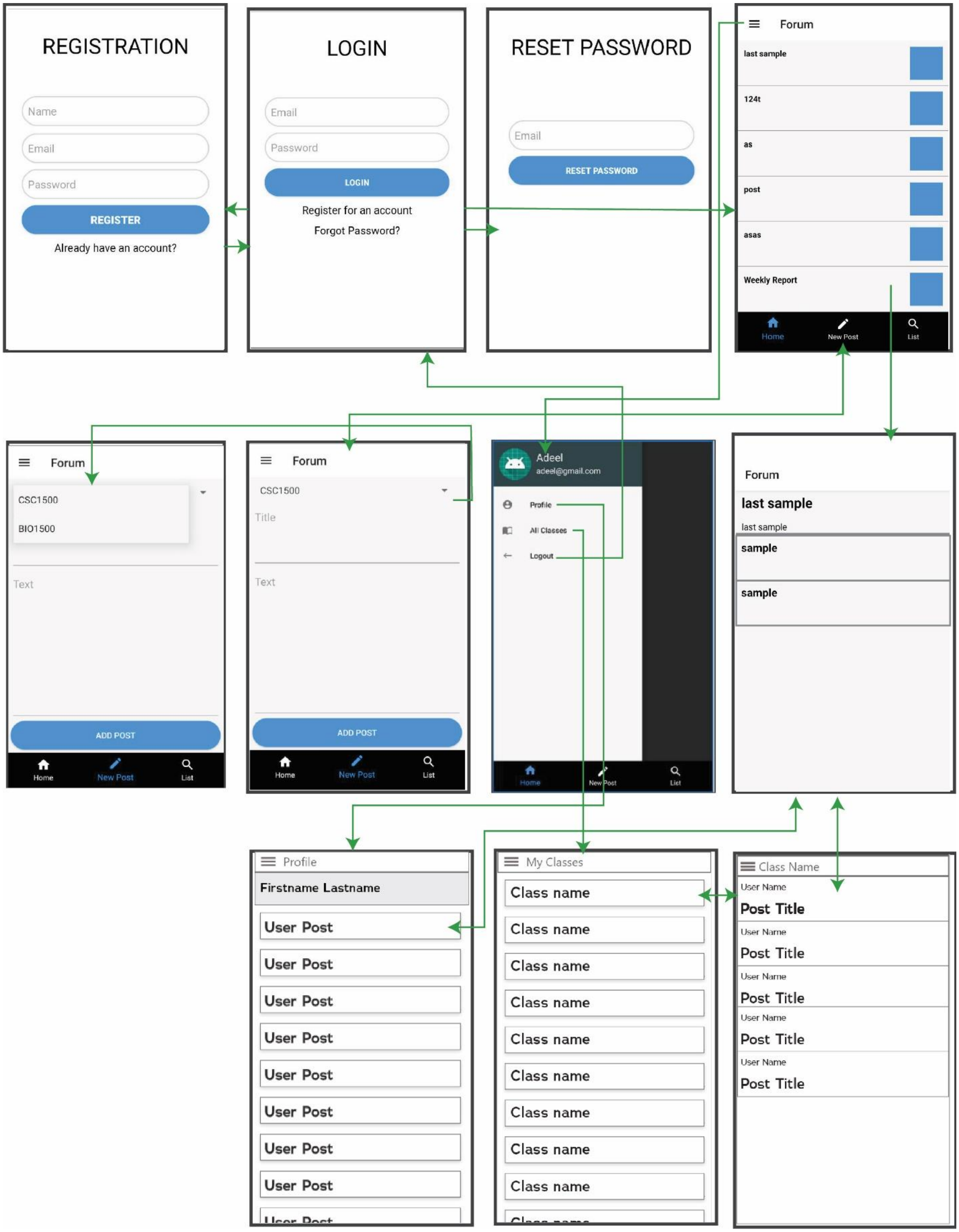
The dagger dependency class diagram shows how the dagger dependency framework is implemented in the app. The authentication view model, clicked post view model, and new post fragment view model are all bound to the generic dagger view model factory while the home fragment view model, community post view model, and list fragment view model are all bound to their respective factories. Dagger is initialized in the Senior Project Application class with the help of Application Module and Application Component.


Class Diagram: Dagger Dependency Setup
<<interface>>
AppComponent
+inject(activity: RegisterActivity)
+inject(activity: LoginActivity)
+inject(fragment: HomeFragment)
+inject(activity: PasswordResetActivity)
+inject(fragment: NewPostFragment)
+inject(fragment: MainForum)
+inject(activity: ClickedPost)
AppModule
+AppModule(app: Application)
+provideContext(): Context
HomeFragmentViewModelFactory
+HomeFragmentViewModelFactory(repository: PostRepository) :ViewModelProvider.NewInstanceFactory()
+create(modelClass: Class<T>): T
CommunityPostViewModelFactory
+CommunityPostViewModelFactory(repository: PostRepository) :ViewModelProvider.NewInstanceFactory()
+create(modelClass: Class<T>): T
InjectorUtils: Object
+providePostViewModelFactory(): HomeFragmentViewModelFactory
+provideListViewModelFactory(): ListViewModelFactory
+provideCommunityPostViewModelFacotry(): CommunityPostViewModelFactory
SeniorProjApp
+app_component: AppComponent
+onCreate()
+initDagger(app: SeniorProjApp): AppComponent
multibinding module
@ViewModelKey(ClickedPostViewModel)
@ViewModelKey(NewPostFragmentViewModel)
@ViewModelKey(HomeFragmentViewModel)
@ViewModelKey(AuthenticationViewModel)
+bindviewmodelFactory (factory:DaggerViewModelFactory):ViewModelProvider.Factory
+bindAuthenticationViewModel(viewModel: AuthenticationViewModel): ViewModel
+bindHomeFragmentViewModel(viewModel: HomeFragmentViewModel): ViewModel
+bindNewPostFragmentViewModel(viewModel: NewPostFragmentViewModel): ViewModel
+bindClickedPostViewModel(viewModel: ClickedPostViewModel): ViewModel
ListFragmentViewModelFactory
+ListFragmentViewModelFactory(repository: PostRepository) :ViewModelProvider.NewInstanceFactory()
+create(modelClass: Class<T>): T
mapkey: Annotation Class
+ViewModelKey(val value: KClass<out ViewModel>)
DaggerViewModelFactory
+DaggerViewModelFactory(creators: Map<Class<out ViewModel>, Provider <ViewModel>>)
+create(modelClass: Class<T>): T

4.6 Application Program Interfaces

We are using the Firebase API which provides us with Authentication and Realtime Database. The Firebase Authentication includes email verification, password reset functionality, and email and password formatting. The Authentication Realtime Database allows us to store users' information, new posts, communities, and comments. These are then displayed in the application using text views and recycler views.

4.7 User Interface Design



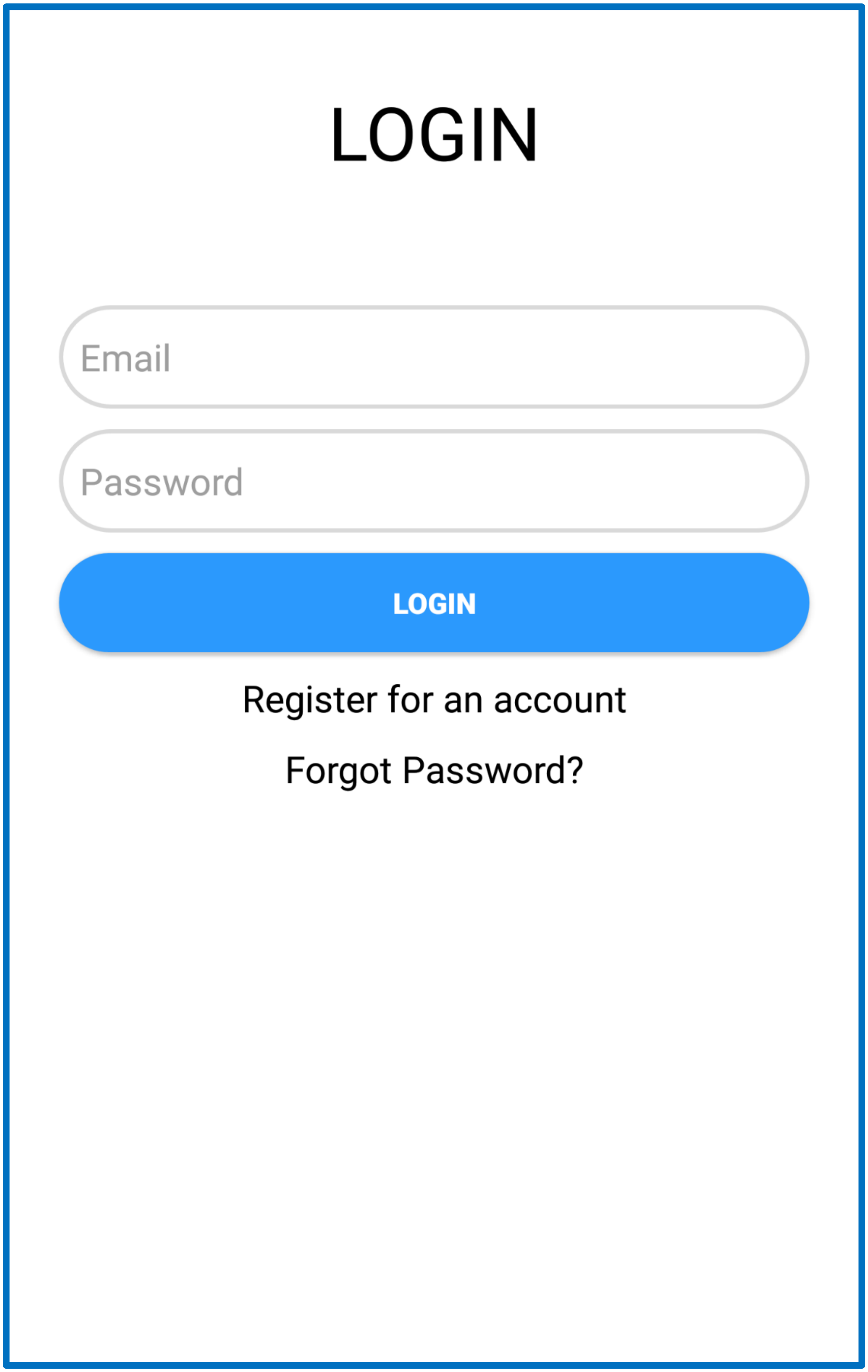
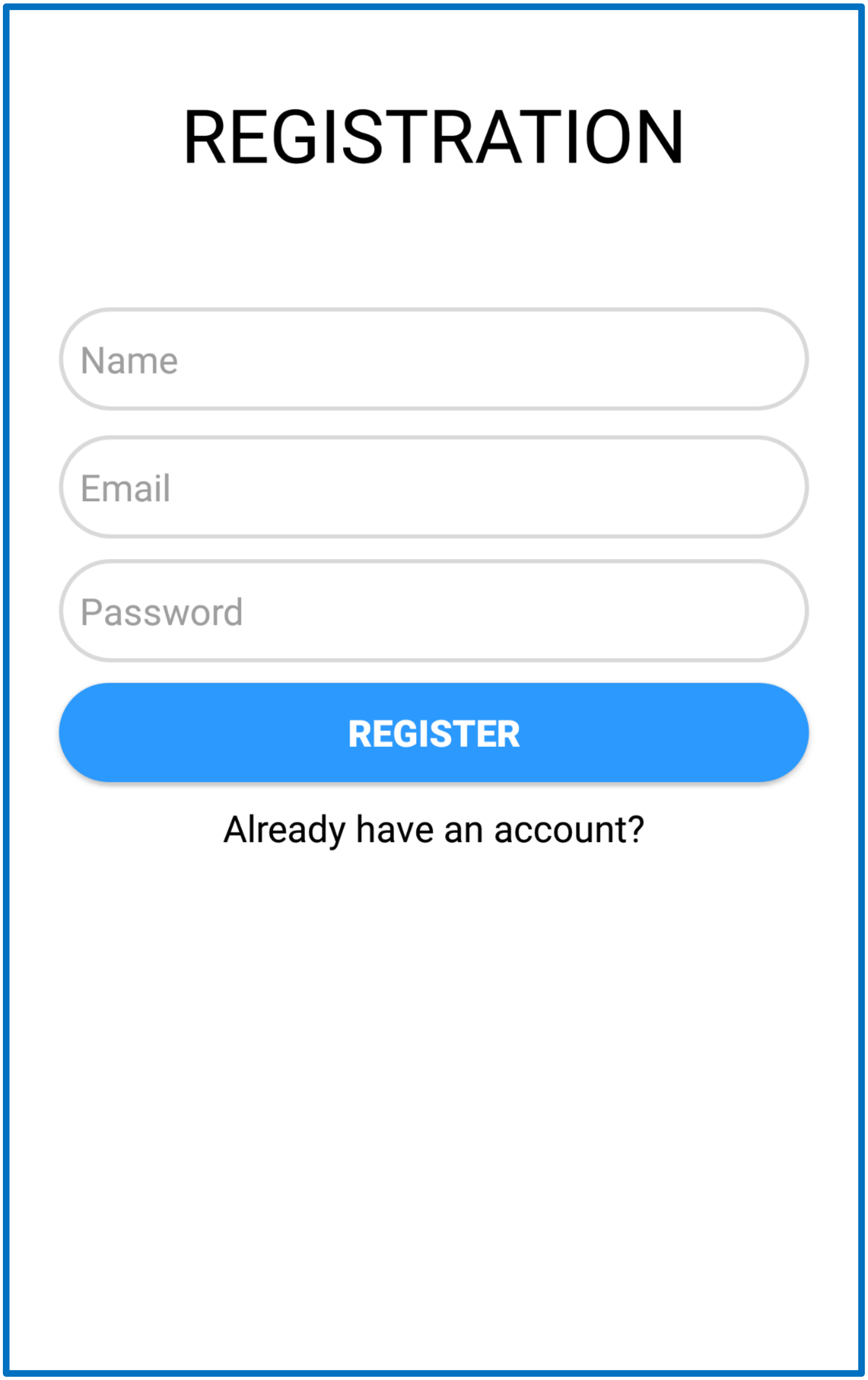


Figure 1: Login Screen



REGISTRATION

Figure 2: Registration Screen

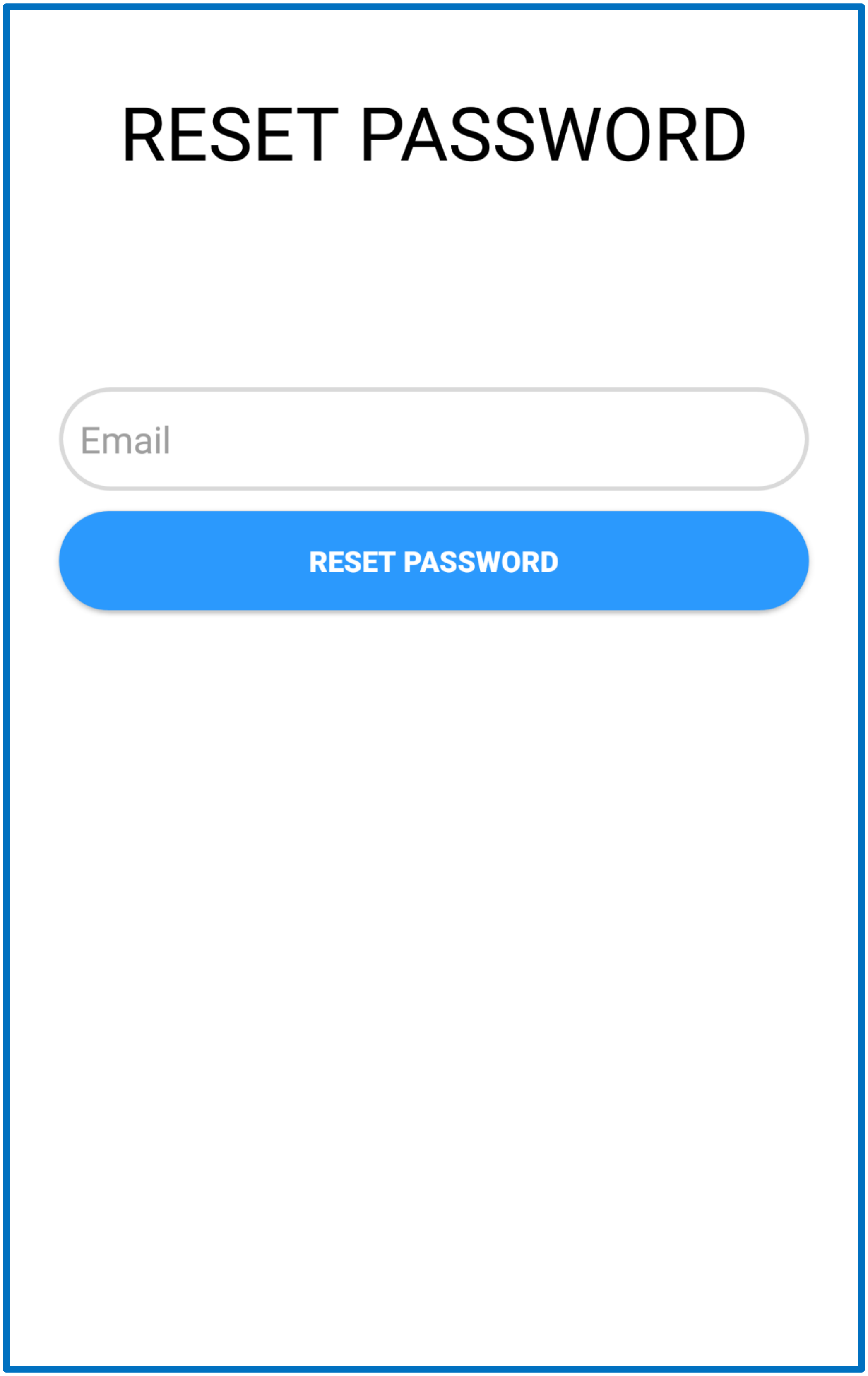
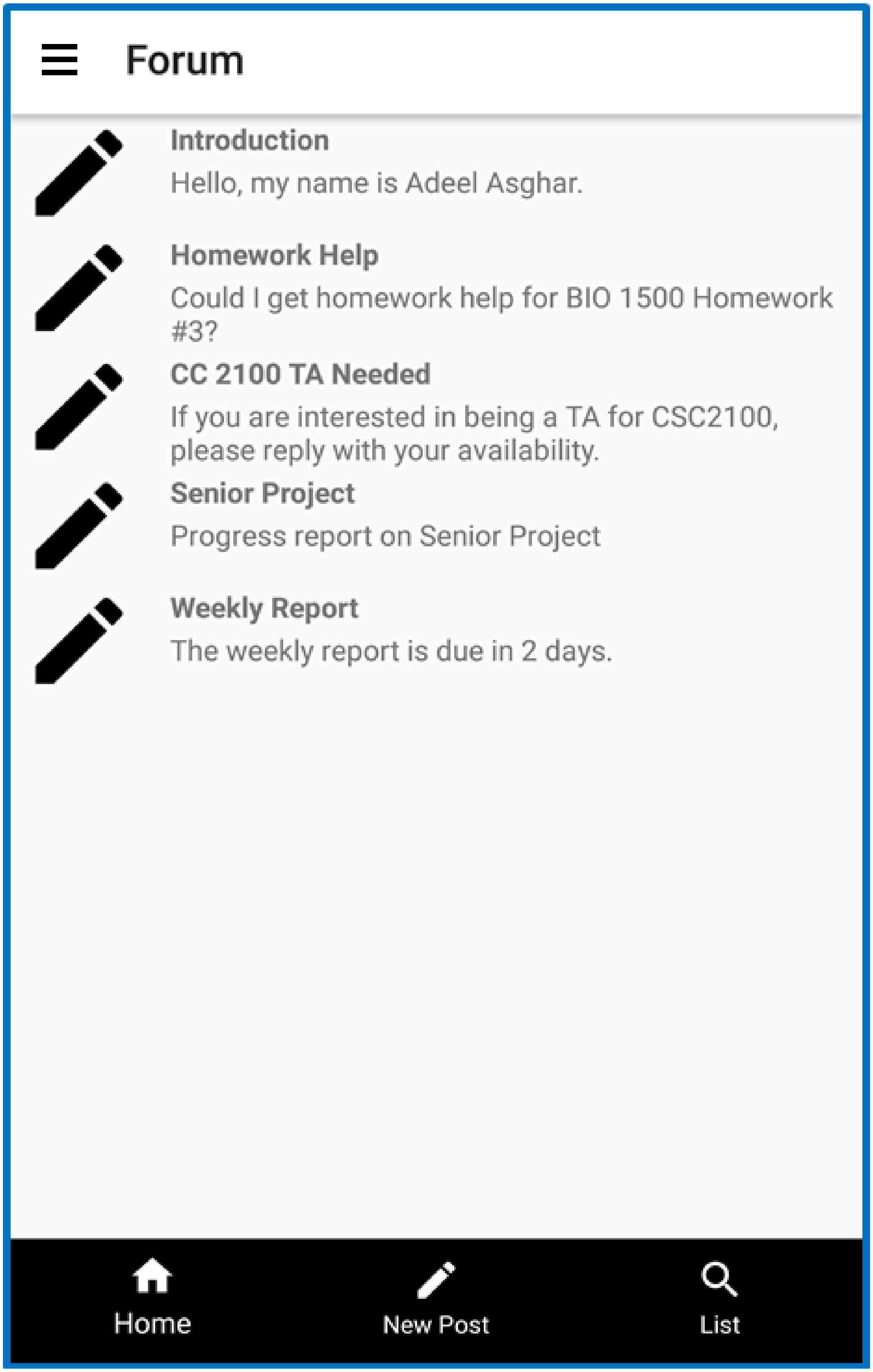


Figure 3: Password Reset Screen



Introduction

Hello, my name is Adeel Asghar.

Homework Help

Could I get homework help for BIO 1500 Homework #3?

CC 2100 TA Needed

If you are interested in being a TA for CSC2100, please reply with your availability.

Senior Project

Progress report on Senior Project

Weekly Report

The weekly report is due in 2 days.

Figure 4: Main forum screen

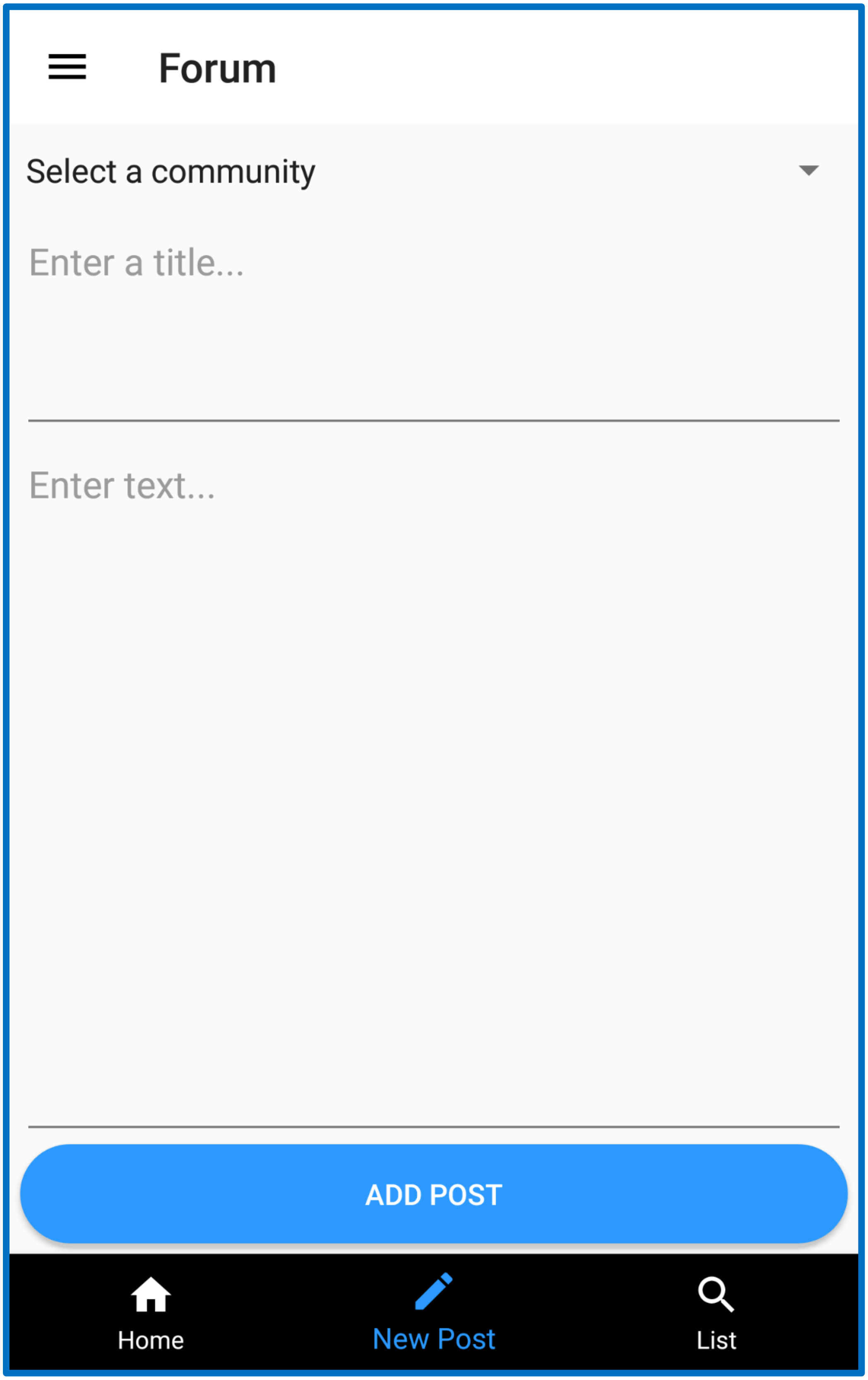
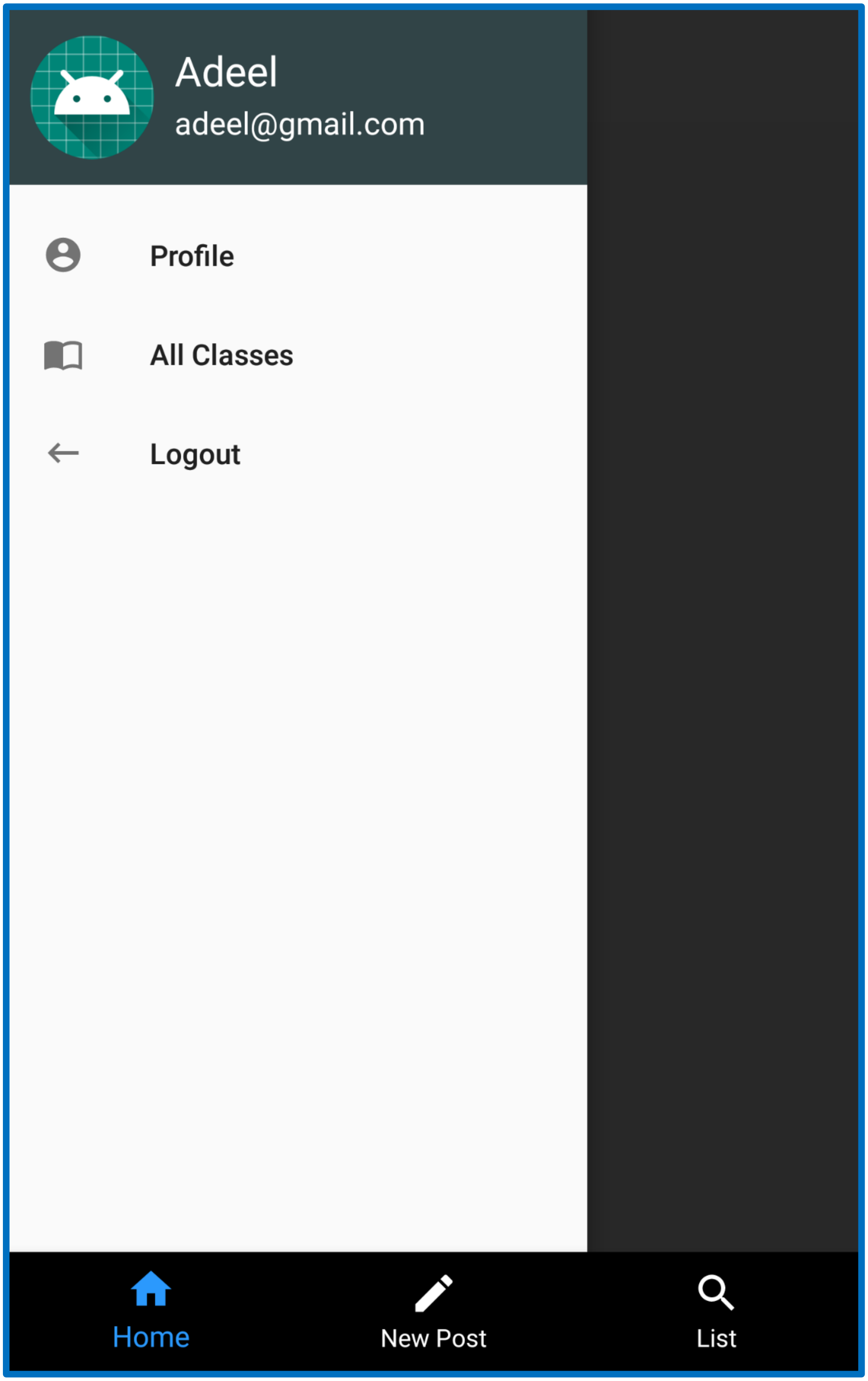


Figure 5: New Post Screen



Adeel

adeel@gmail.com

Figure 5: Navigation Drawer Screen

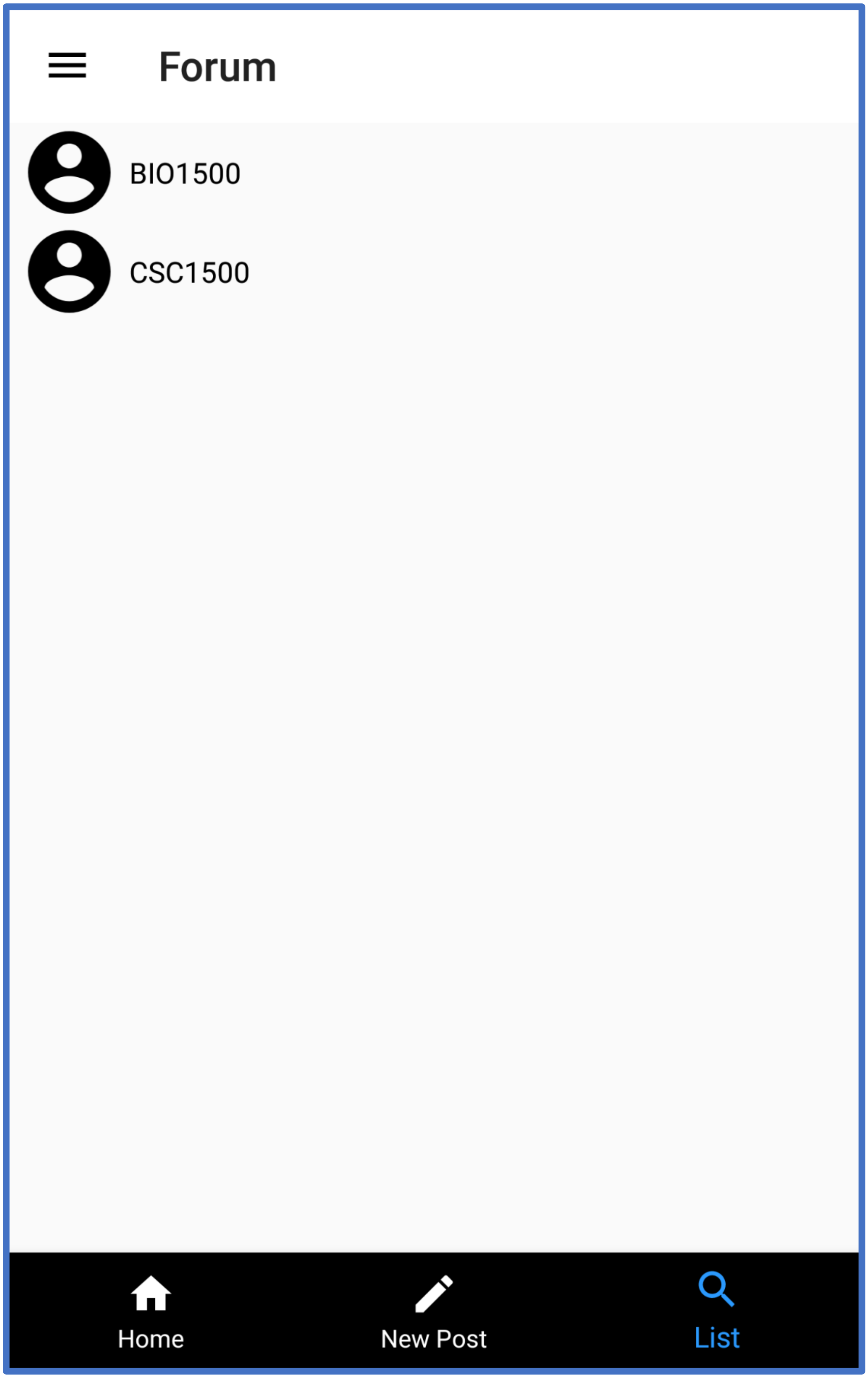
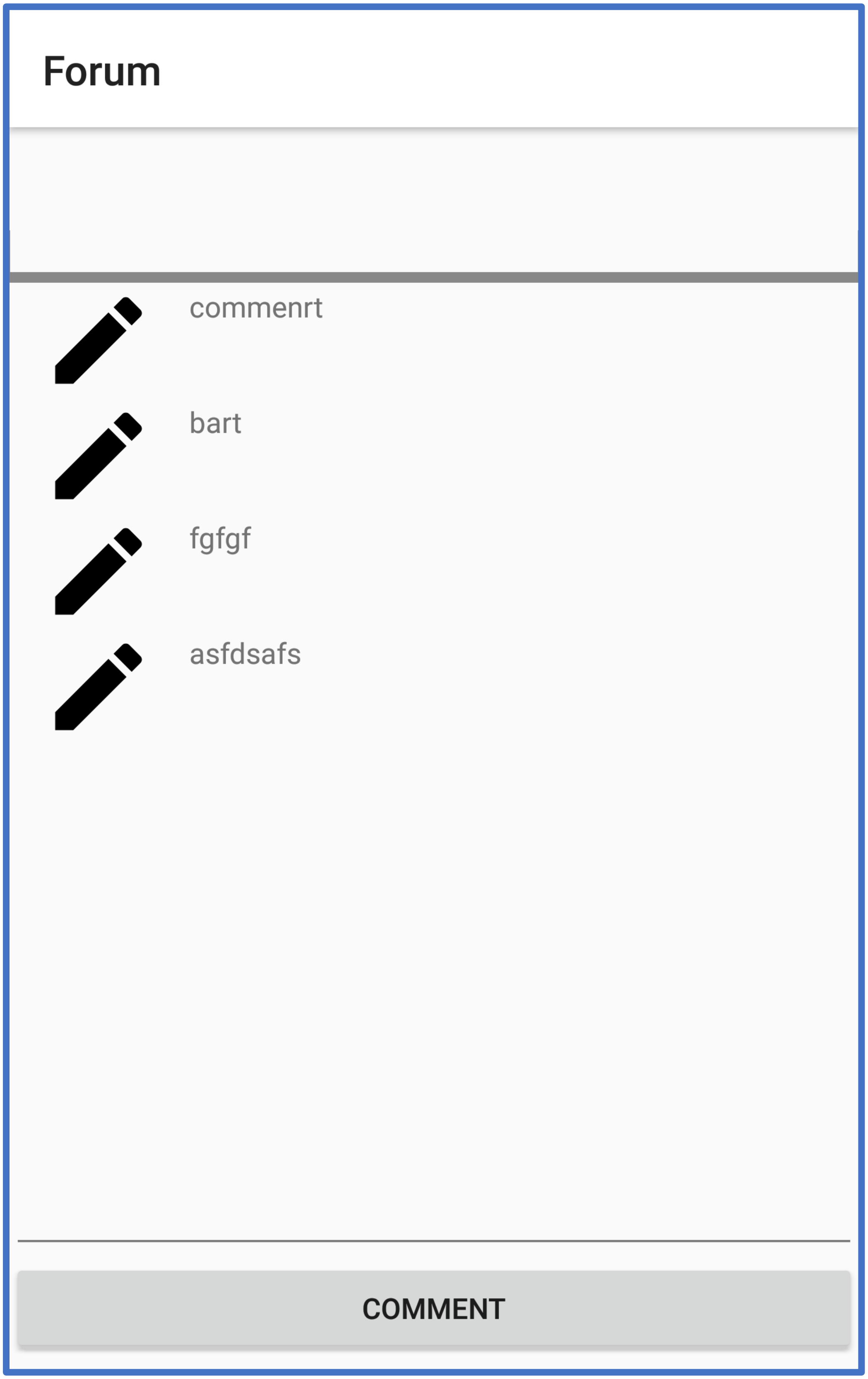


Figure 6: List of Communities Fragment Screen



Forum

commenrt

bart

fgfgf

asfdsafs

Figure 7: Post Comments Screen

5. PRODUCT DESIGN SPECIFICATION APPROVAL

The undersigned acknowledge they have reviewed the Collaborative University Application **Product Design Specification** document and agree with the approach it presents. Any changes to this Requirements Definition will be coordinated with and approved by the undersigned or their designated representatives.

Signature: ______________________________

Print Name: ______________________________

Title: ______________________________

Role: ______________________________

Date: ______________________________

Signature: ______________________________

Print Name: ______________________________

Title: ______________________________

Role: ______________________________

Date: ______________________________

Signature: ______________________________

Print Name: ______________________________

Title: ______________________________

Role: ______________________________

Date: ______________________________

APPENDIX A: KEY TERMS

The following table provides definitions for terms relevant to this document.

Term	Definition
MVVM	Model-View-View Model is an architectural software design pattern for designing a software product.
API	Application Program Interface
UI	User Interface