

EnVIT – A Social Interface for VITians

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science

by

LIKHIT GARIMELLA

18BCE0618

APARNA M SUNIL

18BCE2331

Under the guidance of

Prof. Preetha Evangeline D.

School of Computer Science and Engineering

VIT, Vellore.



VIT[®]
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

June, 2022

DECLARATION

I hereby declare that the thesis entitled “EnVIT – A Social Interface for VITians” submitted by me, for the award of the degree of Bachelor of Technology in Computer Science to VIT is a record of bonafide work carried out by me under the supervision of Prof. Preetha Evangeline.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date : June 3rd 2022

Signature of the Candidate

CERTIFICATE

This is to certify that the thesis entitled “EnVIT – A Social Interface for VITians” submitted by **Likhit Garimella (18BCE0618), SCOPE, VIT University**, and **Aparna M Sunil (18BCE2331), SCOPE, VIT University**, for the award of the degree of Bachelor of Technology in Computer Science, is a record of bonafide work carried out by them under my supervision during the period, 01.12.2021 to 30.05.2022, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : June 3rd 2022

Signature of the Guide

Internal Examiner

External Examiner

Head of the Department
Computer Science and Engineering

ACKNOWLEDGEMENTS

We have successfully completed the “EnVIT – A Social Interface for VITians” project for our Capstone Project CSE4099. We learned various new aspects and the latest technology of Full-Stack iOS App Development, Firebase, and Analytics. Moreover, we got to enhance our practical and theoretical skills in engineering throughout the duration of this project. We would like to thank Prof. Preetha Evangeline (Project Guide) for the positive attitude she showed for our work. It has been our privilege to have a team of project guide who have assisted us from the commencement of this project. The success of this project is a result of sheer hard work, and determination put in by us with the help of our project guide. Her wisdom, knowledge, and commitment to the highest standards inspired and motivated us. Without his insight, support, and energy, this project wouldn't have kick-started and neither would have reached fruitfulness.

We express our immense pleasure and a deep sense of gratitude to Dr. Vairamuthu S (Head of Department, Department of Software Systems) for giving us this opportunity and the guidance throughout the program. The project is dedicated to all those people who helped us while doing this project.

Place : Vellore

Date : June 3rd 2022

Likhit Garimella

EXECUTIVE SUMMARY

The most common act for interaction among VIT students who belong to one class is through Microsoft Teams. Students, especially new joinees from 2020, can only contact their peers from their class through Teams and talk about class related topics. But, what's lacking is that, there must be an interface where students, irrespective of the year, irrespective of the branch, assemble on one single platform and get remotely connected to one another. Through this interface, students can make use of each other's thoughts, help each other, get guidance in subjects or curriculum, get awareness about the various domains of work, get to know about various clubs & chapters and their workshops & events, get good guidance on placements or higher studies or etc., and have a good social connection among their peers and make friends.

TABLE OF CONTENTS

S.no.	Content	Page no.
(i)	Declaration	2
(ii)	Certificate	3
(iii)	Acknowledgement	4
(vi)	Executive summary	5
(v)	Table of contents	6
(vi)	List of figures	7
(vii)	List of tables	8
1	Introduction	9
1.1	Objective	9
1.2	Motivation	10
1.3	Background	11
2	Project Description and Goals	12, 13
3	Technical Specification & Arch. Diagram	14
4	Design Approach and Details	15
4.1	Design Approach / Materials / Methods	15-25
4.2	Codes and Standards	26-50
5	Schedule, Tasks and Milestones	51, 52, 53
6	Project Demonstration	-
7	Cost Analysis / Result / Discussion	54
8	Summary	55
9	References	56, 57
10	APPENDIX A	58, 59

List of Figures

Figure No.	Page No.	Title
1	14	Architectural Diagram
2	15	Signup, Login, Home & Profile Screens Storyboarding
3	16	Tech Share - (Mentor + Mentee) Storyboarding
4	17	Project Share Storyboarding
5	18	Share on the Wall Storyboarding
6	19	Signup & Login Screens
7	20	Tech Share & Project Share Screens
8	21	Mentor & Mentee Feed Screens
9	22	Mentor & Mentee Post Screens
10	23	Personal Projects Feed & J-Component Projects Feed
11	24	Project Post & Project Request
12	25	Share on the Wall Screen & Profile Screen
13	30	Firebase Server Screenshots
14	50	Xcode Swift Codes Screenshots
15	59	Xcode project, Info plist, Google service plist, Podfile Screenshots

List of Tables

Table No.	Page No.	Title
1	51	List of Project Schedules
2	52	List of Tasks Sections
3	53	List of Project Milestones

1. Introduction:

1.1 Objective - EnVIT:

The work presented aims at developing an online social interface in the form of a fully-authentic and easy-to-use mobile application exclusively for VITians based on Swift, Firebase, Machine Learning, and many more.

The project has the following objectives which have been completed during the fulfilment of the work:

- To make juniors or beginners get the right guidance related to academic or any domain work, etc. from seniors or mentors by posting a request or a query.
- To make seniors or mentors share their domain experience or work experience, etc. which helps juniors or beginners.
- To connect seniors-juniors and peers-classmates and make friends on one social platform with a friendly user-interface.
- To make students share any of their personal projects or research or work experience which helps peers and other students.
- To enable students to find and collaborate with a team to work on a research project of a specific domain or a J-component project in a class.
- To make students get help from peers or seniors with any specific domain project or a J-component project by posting a request or a query.
- To help students get exam question papers of other batches and slots for CATs and FAT. - Essentially, to create a college atmosphere where students can share their VIT clicks/posts/snaps/memories on the ‘Wall’.
- To find peers, classmates, seniors and get connected with them and make friends online.
- To give a social touch interface by being able to like, comment, share and save posts, and stay connected with friends.

1.2 Motivation - Social Interface:

The most common act for interaction among VIT students who belong to one class is through Microsoft Teams. Students, especially new joinees from 2020, can only contact their peers from their class through Teams and talk about class related topics. But, what's lacking is that, there must be an interface where students, irrespective of the year, irrespective of the branch, assemble on one single platform and get remotely connected to one another. Through this interface, students can make use of each other's thoughts, help each other, get guidance in subjects or curriculum, get awareness about the various domains of work, get to know about various clubs & chapters and their workshops & events, get good guidance on placements or higher studies or etc., and have a good social connection among their peers and make friends.

There is a certain paradox that we must turn to in the field of technology and applications to help us not to be distracted ... well, technology. However, as many technical experts have practiced philosophy over the years, the key to getting the best out of technology is to use these tools to meet our needs, rather than letting them become our experts. There is nothing better than displaying this idea than a growing field of applications for educational purposes.

Mobile apps can help students in a variety of ways. There are learning apps that can help you learn a new language or look up information about the topic you are reading, and other learning apps that can help you track all your class notes and manage your time effectively. Naturally, it is easy to get frustrated with apps that support your online reading skills, lest you spend more time on them than actually reviewing tests or writing papers. So what you choose to do to improve your reading skills depends a lot on what you feel your weaknesses are, and what can help you to become a more productive and effective student. Access to any information from anywhere at any time makes the learning process easier and simpler. In these changing times, access to information is readily available through mobile phones.

1.3 Background - Existing Technology:

One commonly used web-based mobile application by VITians for resources and help is VITspot. It was commonly used back in a time, but over time and the pandemic situation, the features in VITspot became less feasible. The features in VITspot include – Resources, Faculty, Cab Share, Calculators, Wifi Manager and Marks. Resources like question papers for CAT & FAT – the subject pattern and exam pattern have been changed since the last 2 years. So the papers would be outdated hence we are incorporating an updated version of this feature in our project. Faculty like faculty search – it is not that feasible as we have Vtop and VITian app, and the data is getting updated every now and then too. Cab Share – as college has been closed, the ability to share a cab while travelling from one place to another wouldn't really be feasible unless the college gets reopened. Wifi Manager – again as college has been closed, making use of college wifi and hostel wifi in VIT is really not feasible at the moment. Calculators and Marks – due to the change in the credit system, exams pattern and marking evaluation since the last 2 years, this feature needs an update, which we will be incorporating in our project. Apart from the mentioned brief motto of our project, a detailed objective explaining what all we will be incorporating and building in our project is mentioned below.

2.1 Project Description:

In early 2020, the world was plagued with a fatal virus that made the people, especially students, stay at one place and be remotely connected. Students used to be connected with one another on campus when they used to attend different classes, stay in touch with the faculties, make friends and help one another. But now students couldn't continue their semesters on campus physically due to the pandemic situation. Also, new students of 2020, 2021 & 2022 batches couldn't even visit the campus or barely got a chance to meet their subject faculties or their classmates. Students find this really hard to cope with, not getting good exposure like how they do on campus by interacting with friends, classmates, seniors, clubs, chapters, workshops, tech events etc.

The smartphone is a new student editor - a repository of dates, deadlines and important information. But unlike the traditional editor, there are countless additional options in the form of mobile apps for college students. Across the digital world there seems to be an app for almost everything, whether it's ordering food, booking tickets, posting videos, identifying dog breeds or reviewing virtual flashcards for college classes. The presence of mobile applications means that the options are almost endless. Educational technologies such as apps can help students with time management, organisational skills, homework, collaboration and more.

Mobile applications that keep students on the job can be very helpful, considering that incoming college students may often be exempted, in Watson's experience. "I think the biggest challenge for many students today is that they do not have the ability to learn to control themselves and the reason for that is our education systems".

2.2 Goals:

- To make juniors and beginners receive the correct guidance related to academic or instruction Domain work by mentors by posting requests or queries. In order to make mentors, they share their domain experience and their vocational experience, and help juniors and beginners.
- Connect senior juniors and peer-classmates and find them in social friend Platform with a friendly user interface.
- Share your own personal project, research and research experience. This helps colleagues and other students.
- Students are enabled to cooperate with the team in cooperation with research projects like certain domain or j-component projects in the class which will make students get help from peers or seniors with any specific domain project or a j-component project by posting a request or a query.
- To examine slot questions for other batches and cats to help students with their study.
- In essence, students create a university atmosphere where students can share their Vit Wall click / review / snap / memory. Colleagues, classmates, elderly people, and to find friends and find friends online. Use posts to comment, share, save and add social touch interfaces.

3.1 Technical Specifications:

Above objectives are stacked upon and developed in a mobile application with the mentioned features, and the proposed technology uses the following tech stack and technical specifications:

- Xcode IDE for application development.
- Swift, Objective-C & Ruby for Frontend.
- Google's Firebase – Auth, Database, Storage, Analytics.
- Firebase & REST APIs for Backend.
- Sketch & Figma for Prototyping & Design.
- GitHub, App Store Connect and TestFlight for build collaboration.

3.2 Architectural Diagram:

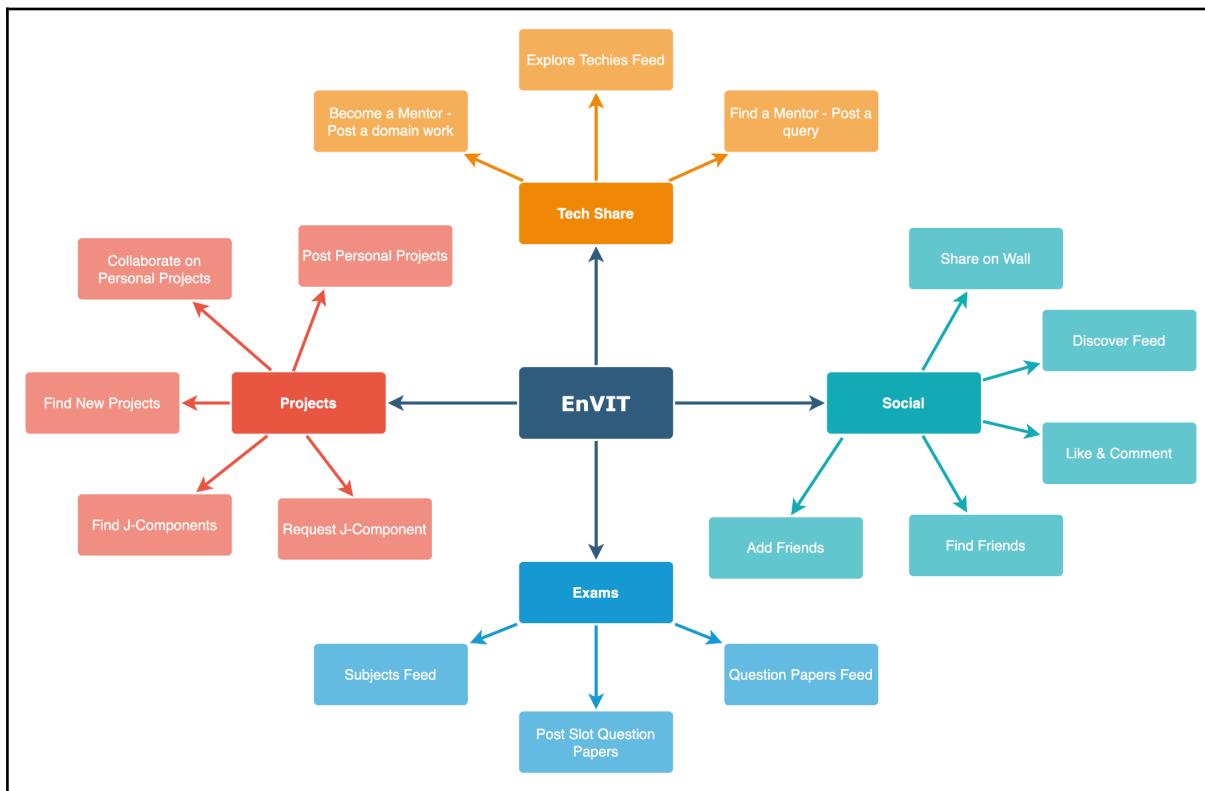


Fig 1: Architectural Diagram

4. Design Approach and Details:

4.1 Design Approach / Materials / Methods: - Xcode Storyboarding & Mobile App Screenshots:

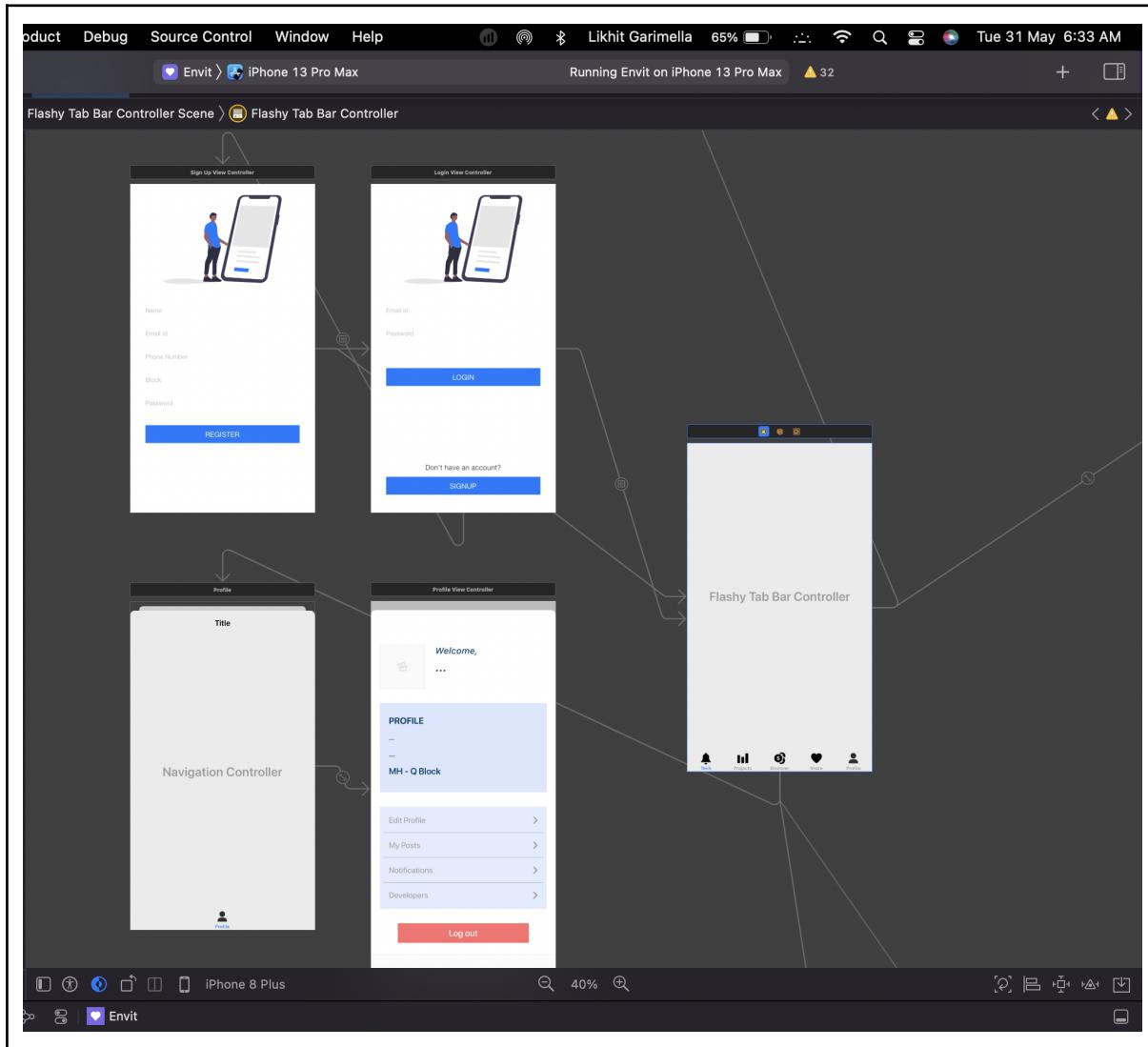


Fig 2: Signup, Login, Home & Profile Screens Storyboarding

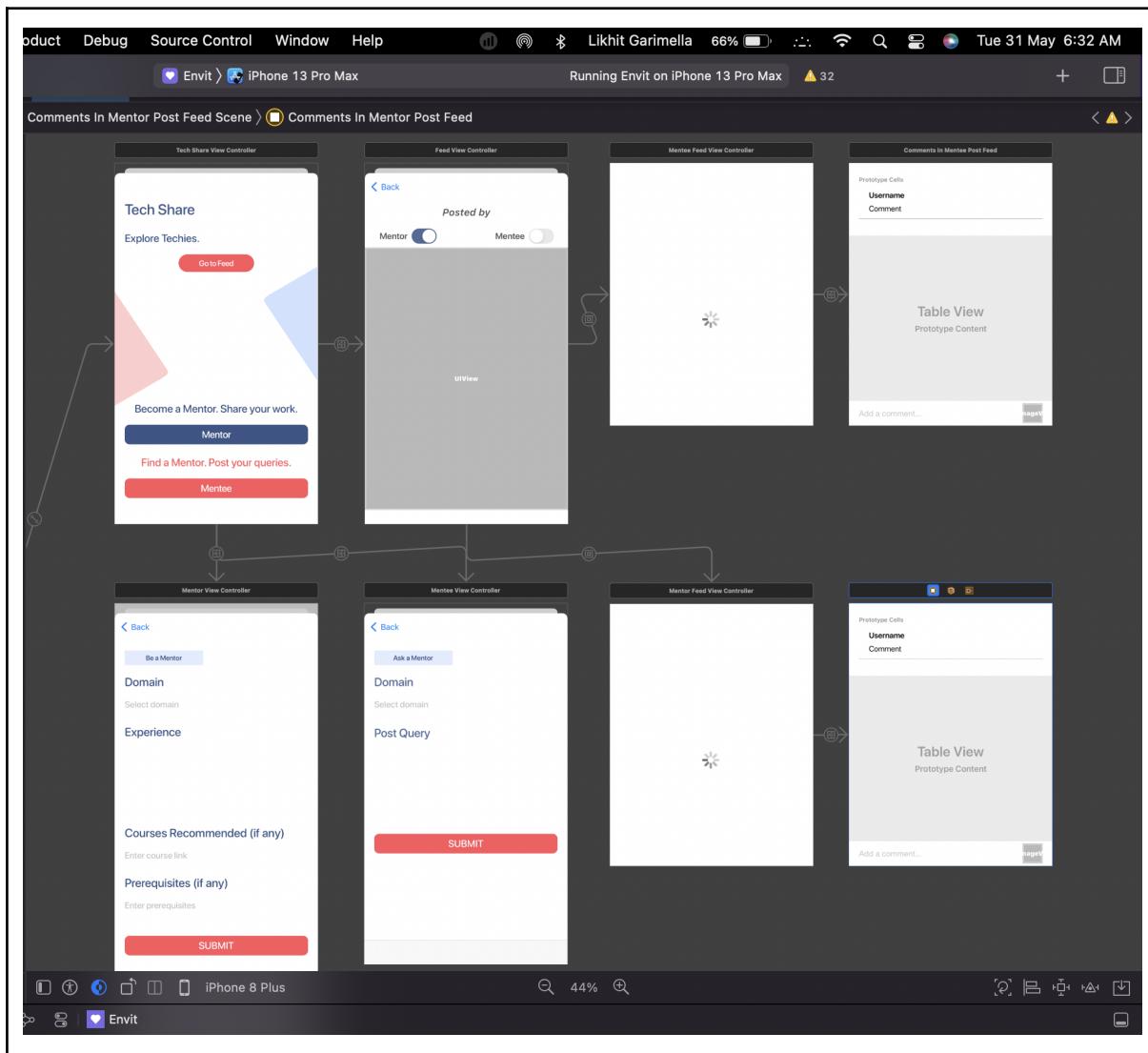


Fig 3: Tech Share - (Mentor + Mentee) Storyboarding

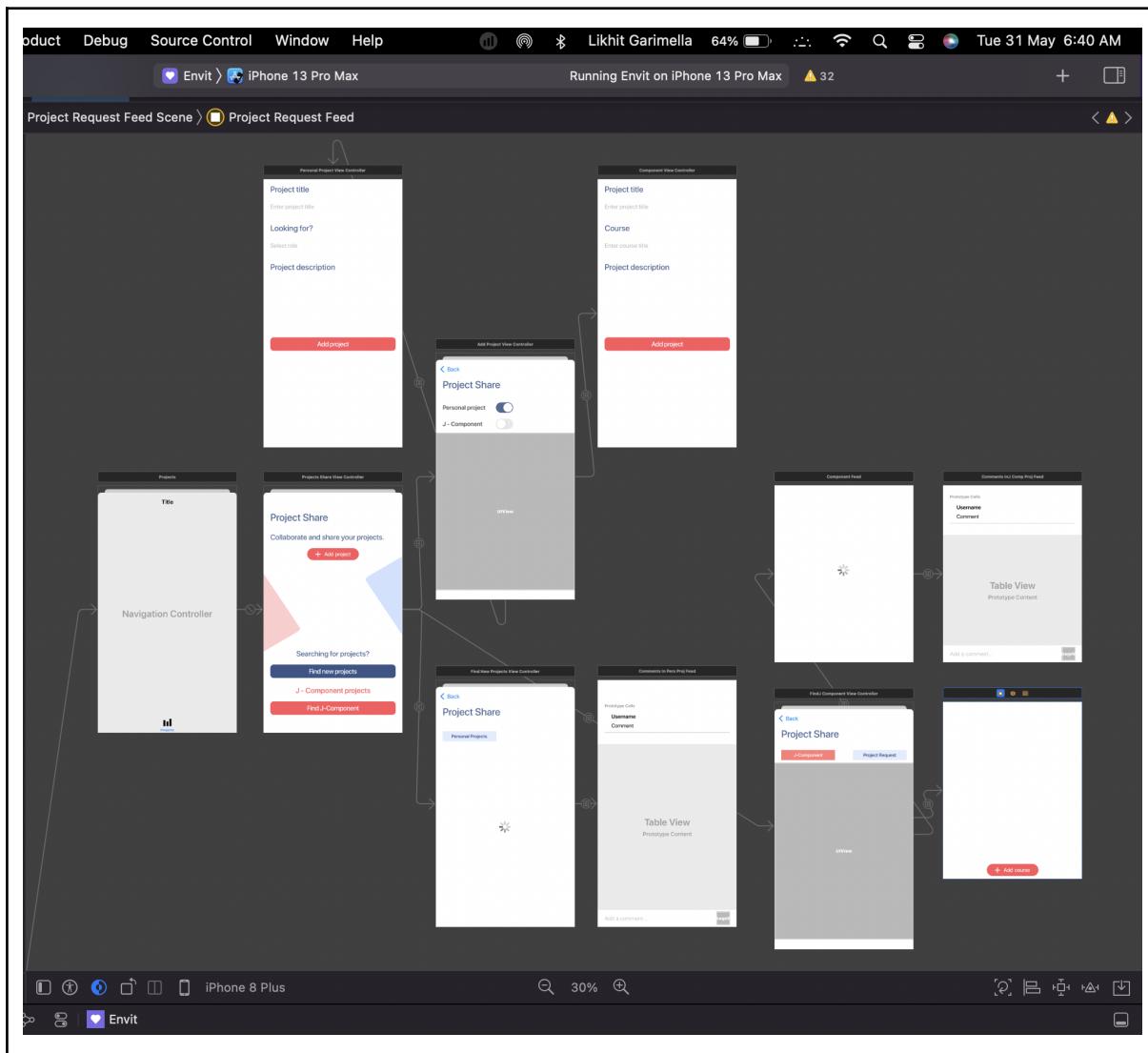


Fig 4: Project Share Storyboarding

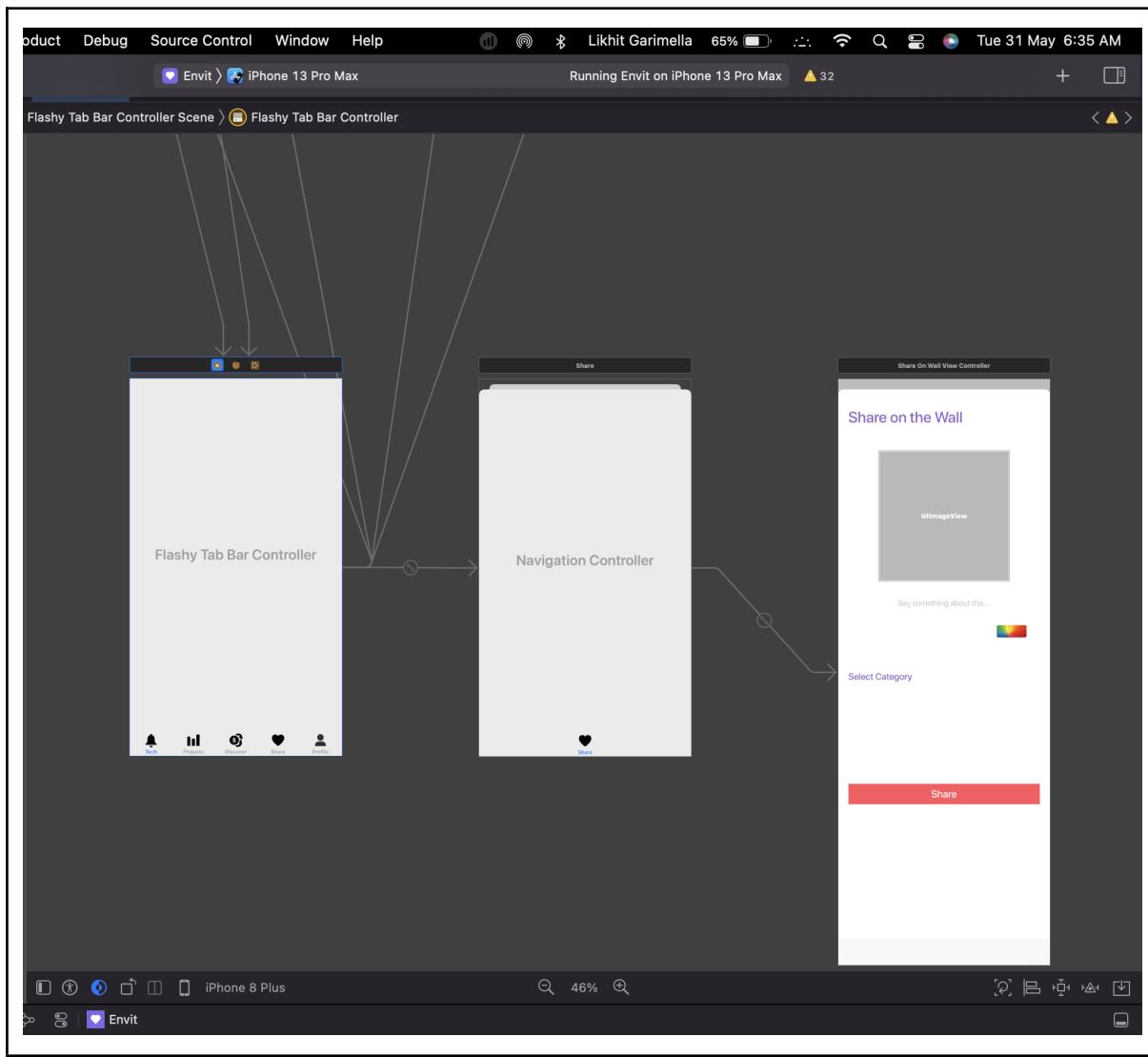


Fig 5: Share on the Wall Storyboarding

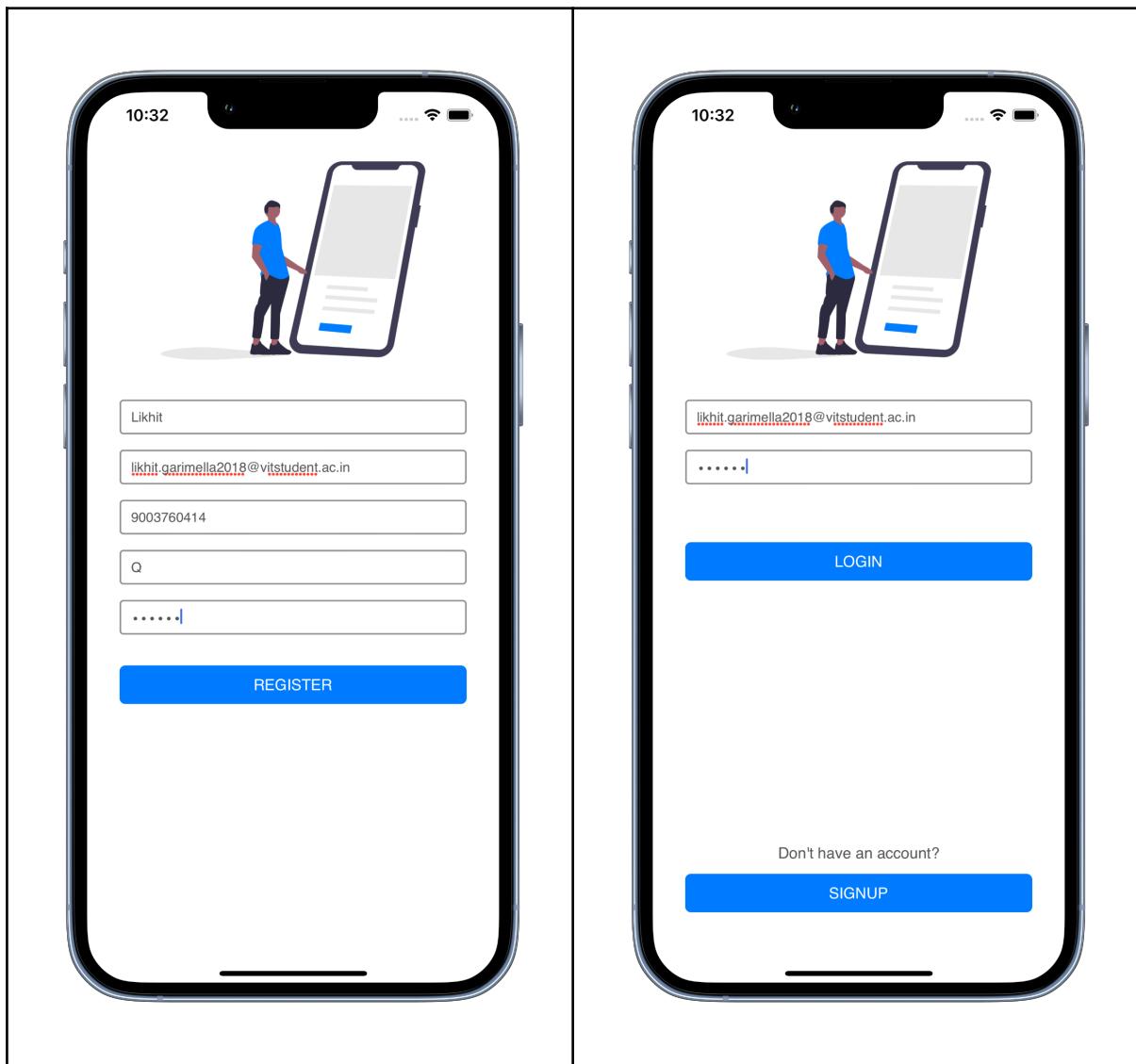


Fig 6: Signup & Login Screens

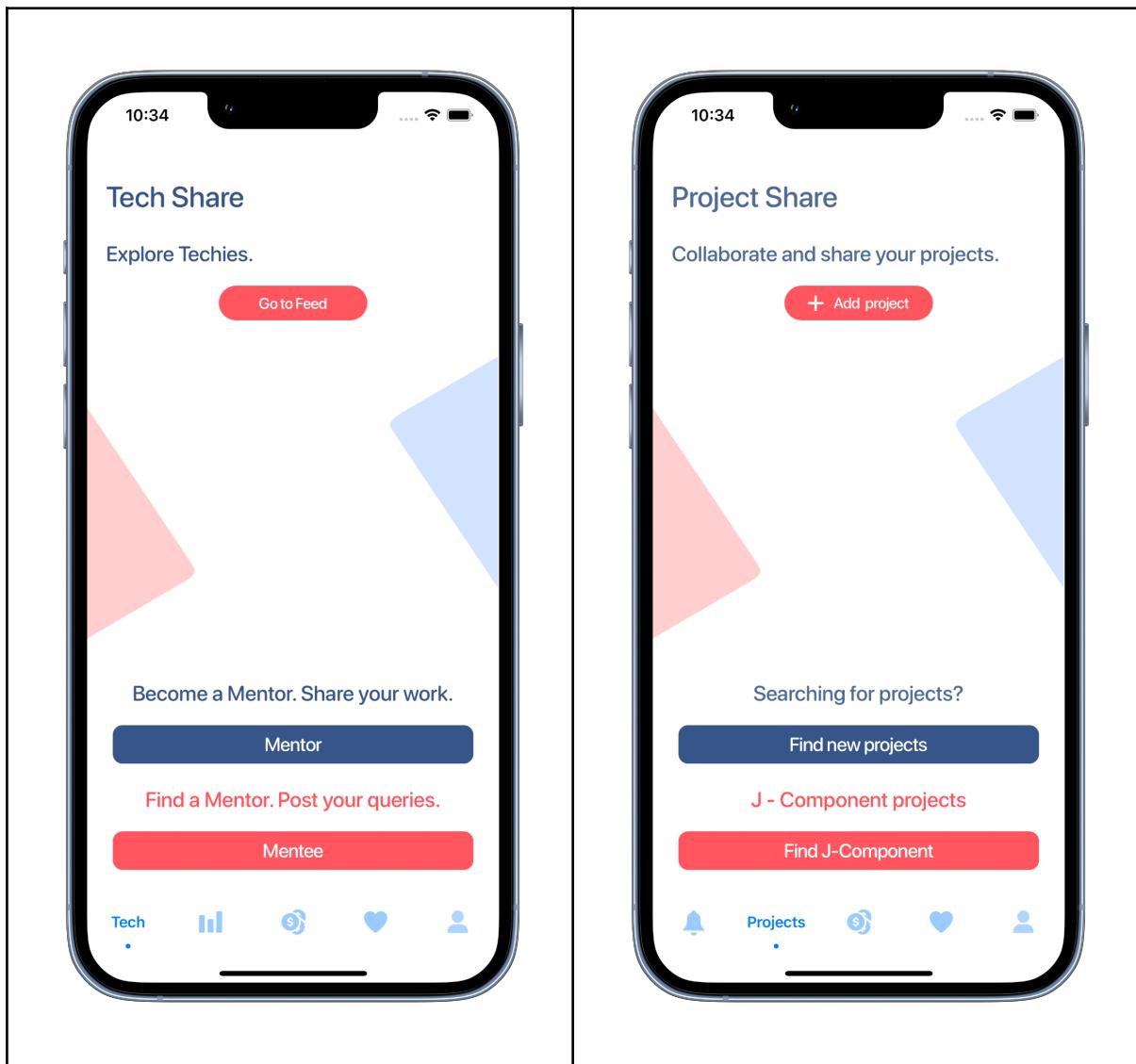


Fig 7: Tech Share & Project Share Screens

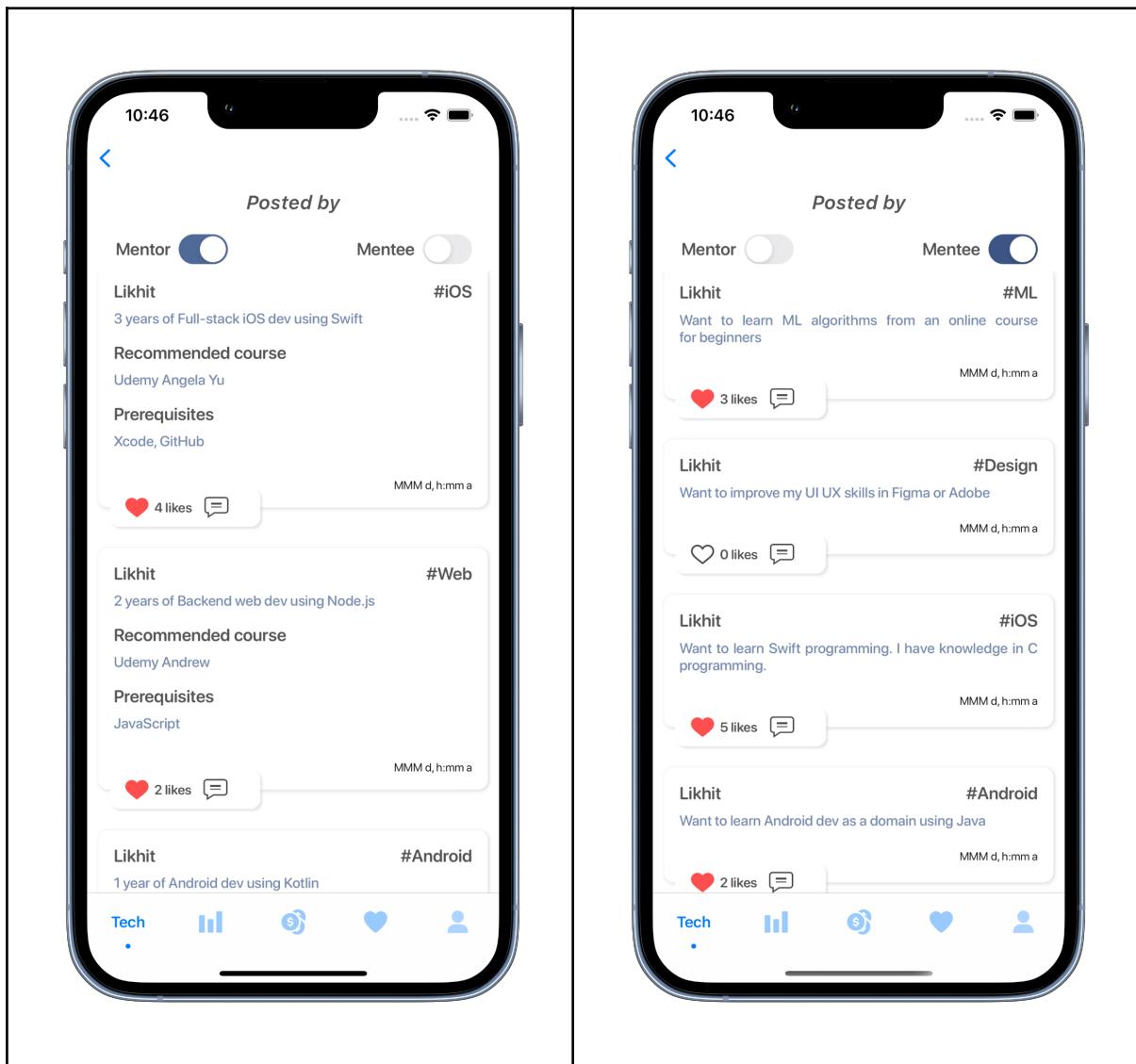


Fig 8: Mentor & Mentee Feed Screens

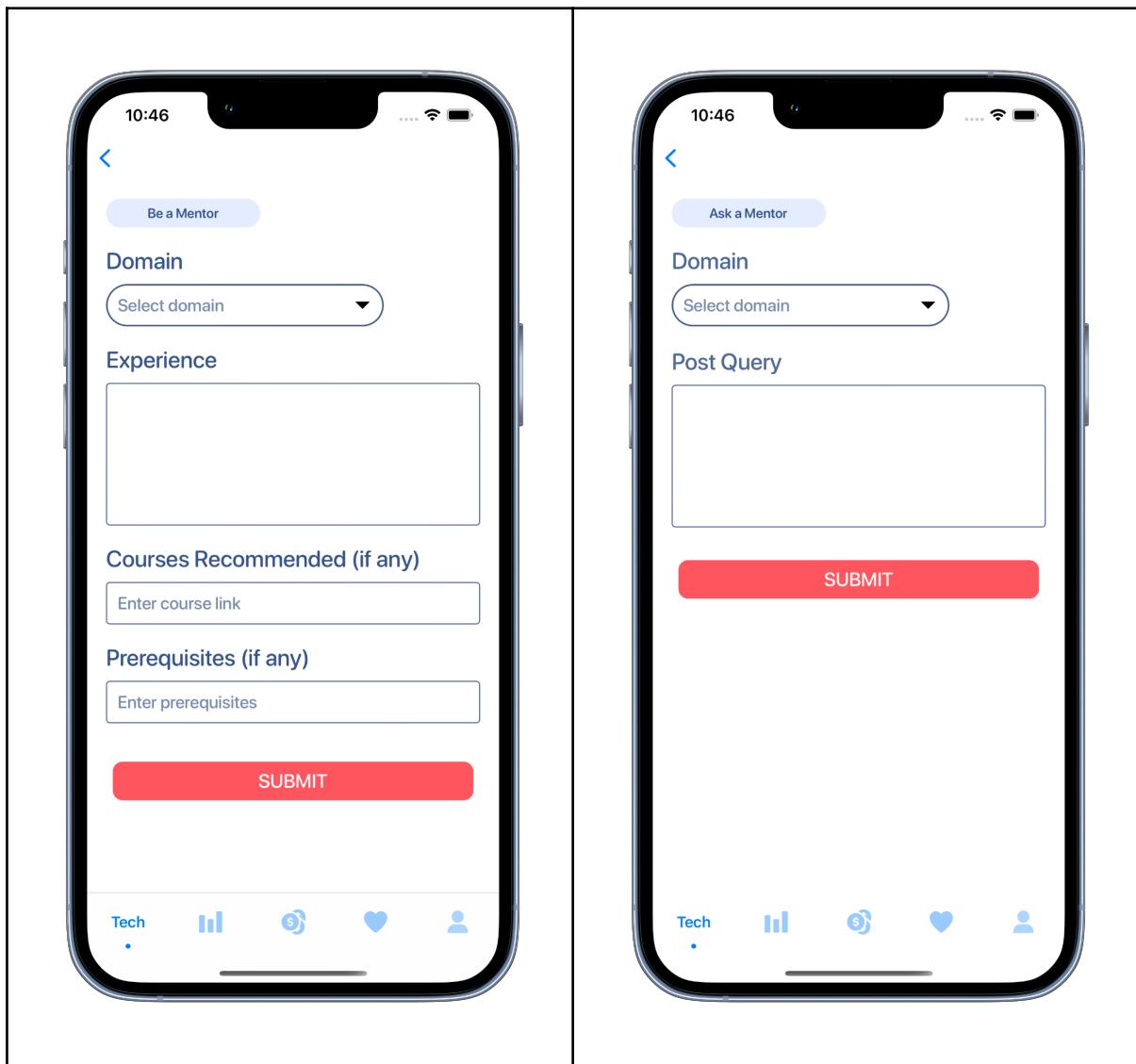


Fig 9: Mentor & Mentee Post Screens

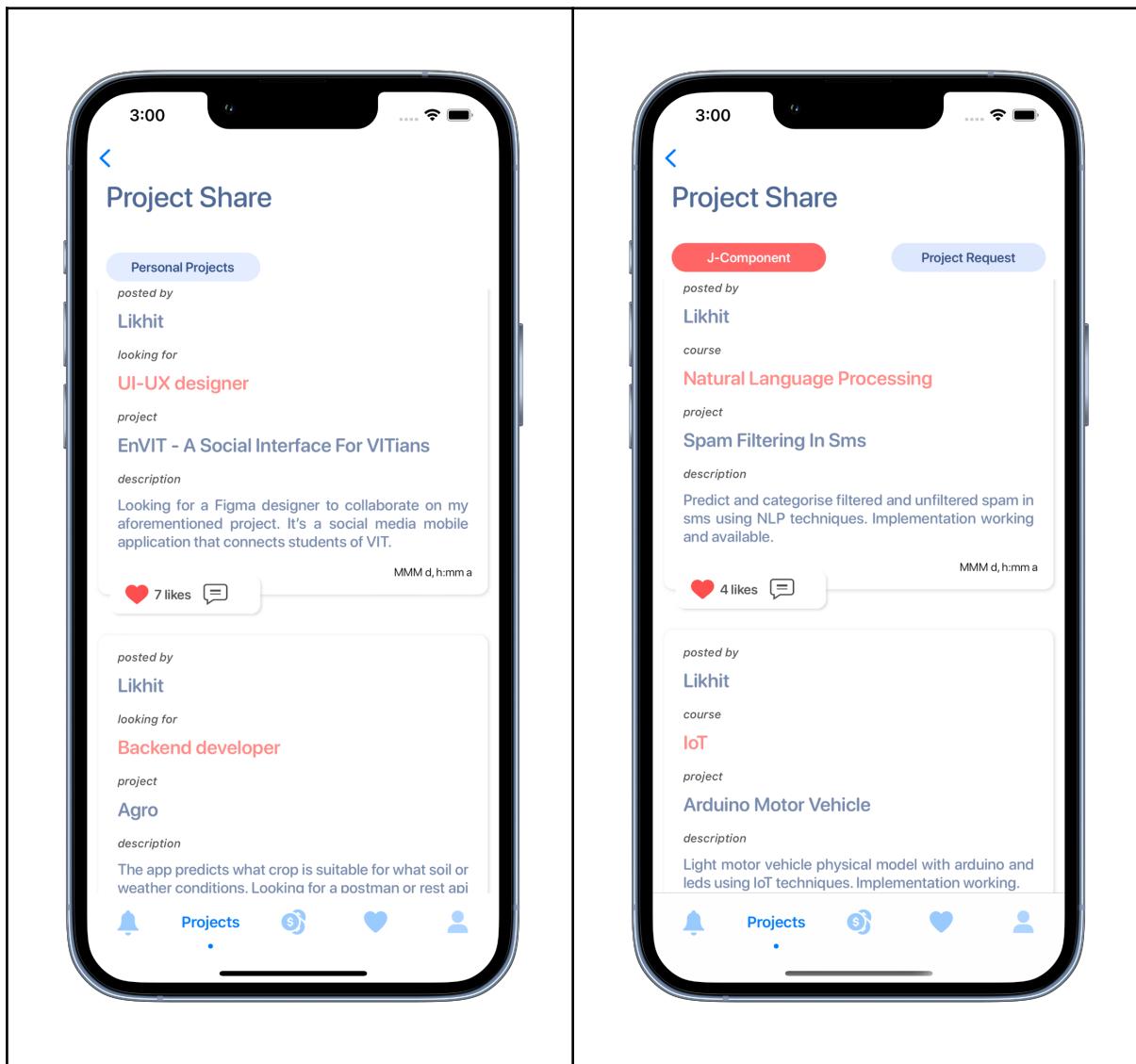


Fig 10: Personal Projects Feed & J-Component Projects Feed

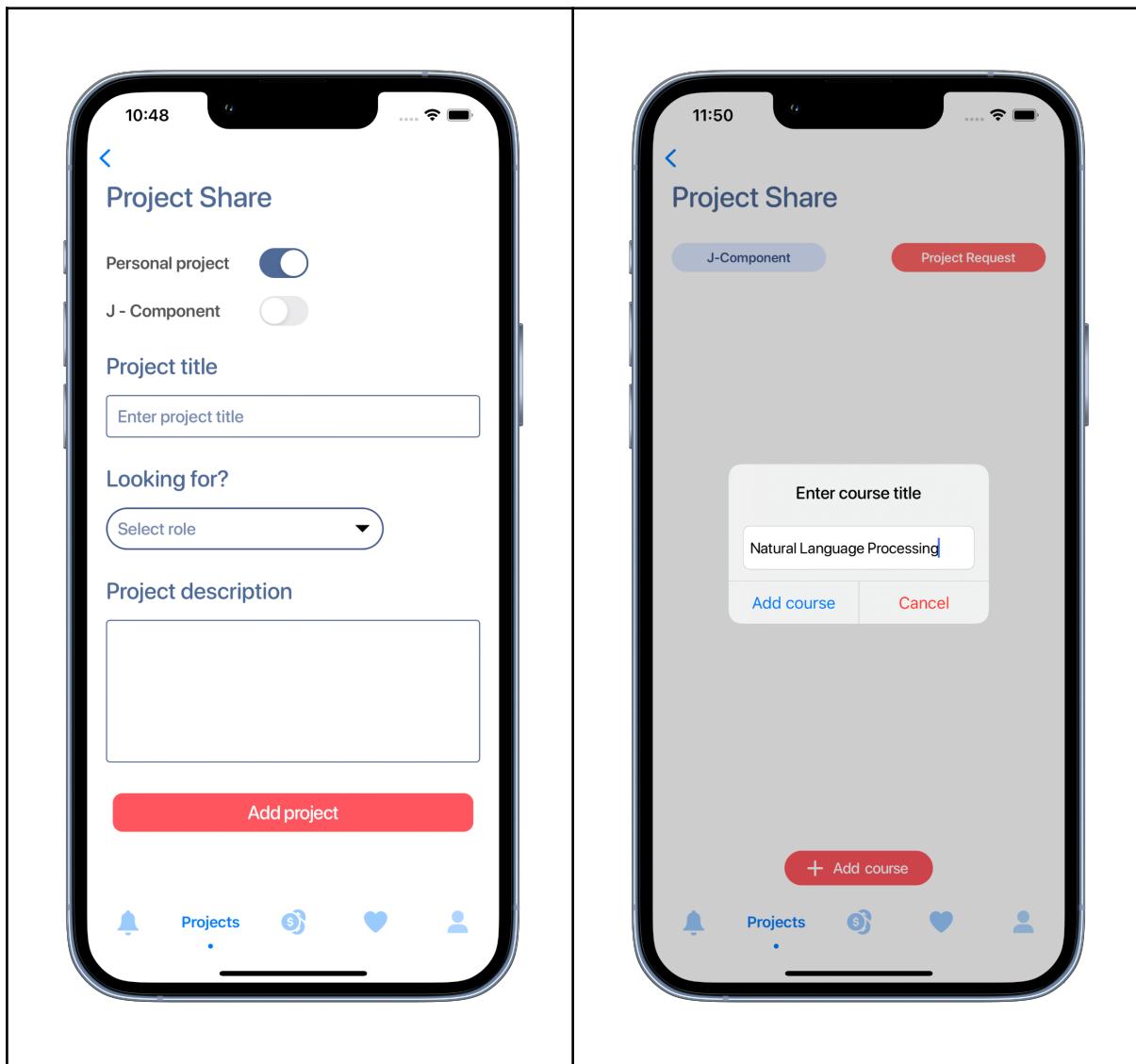


Fig 11: Project Post & Project Request

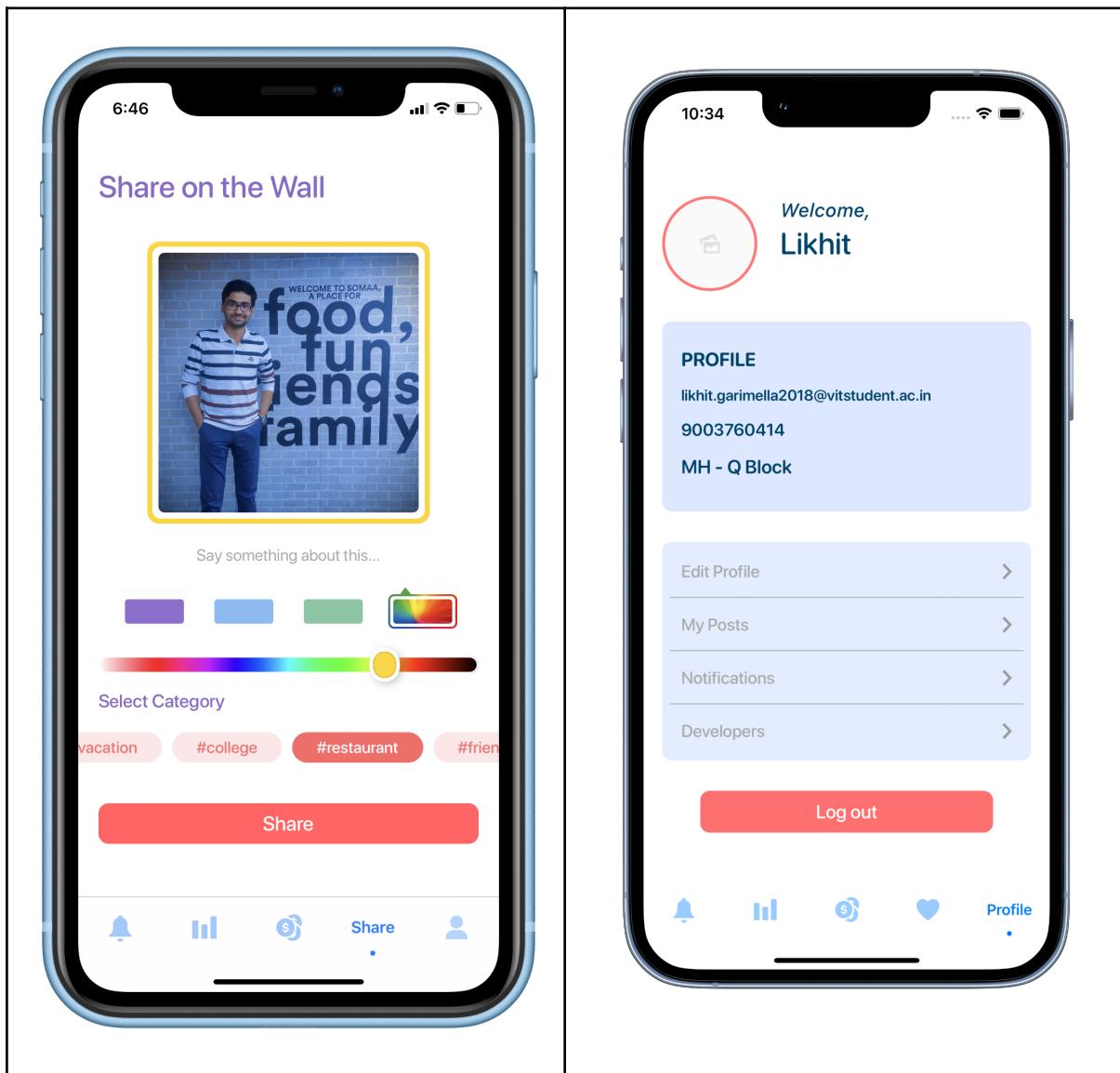
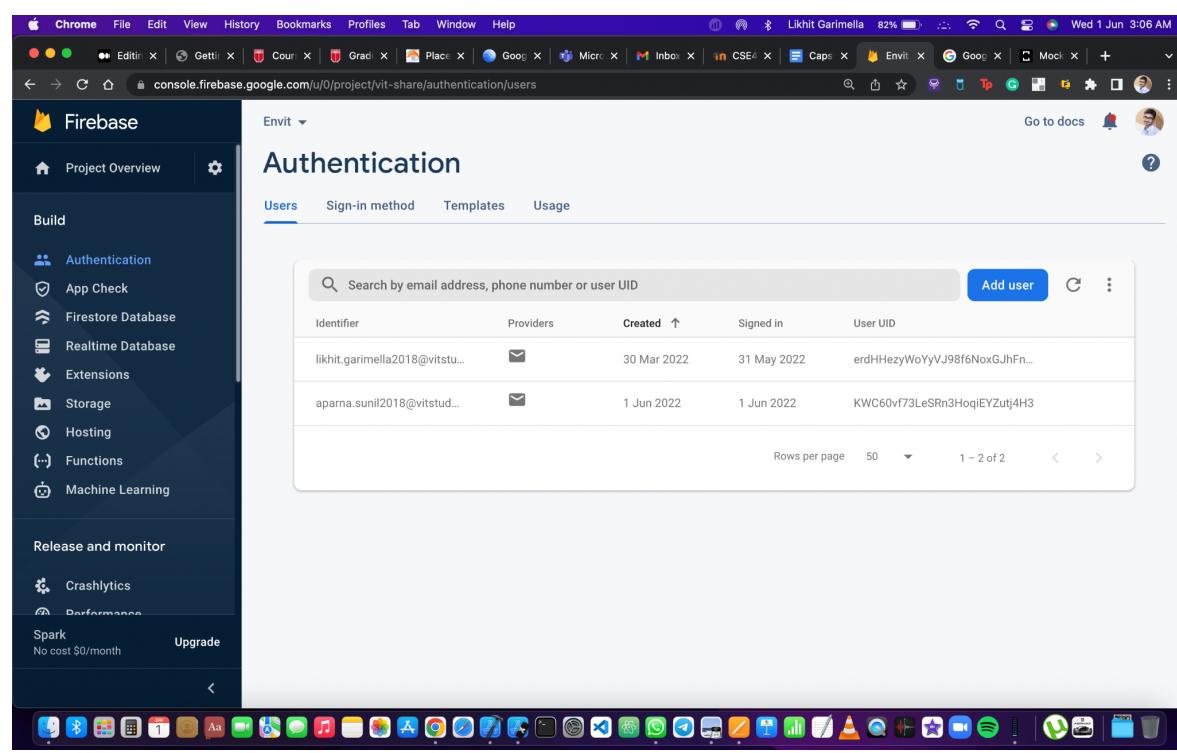


Fig 12: Share on the Wall Screen & Profile Screen

4.2 Codes and Standards: - Xcode Swift Codes & Firebase Server

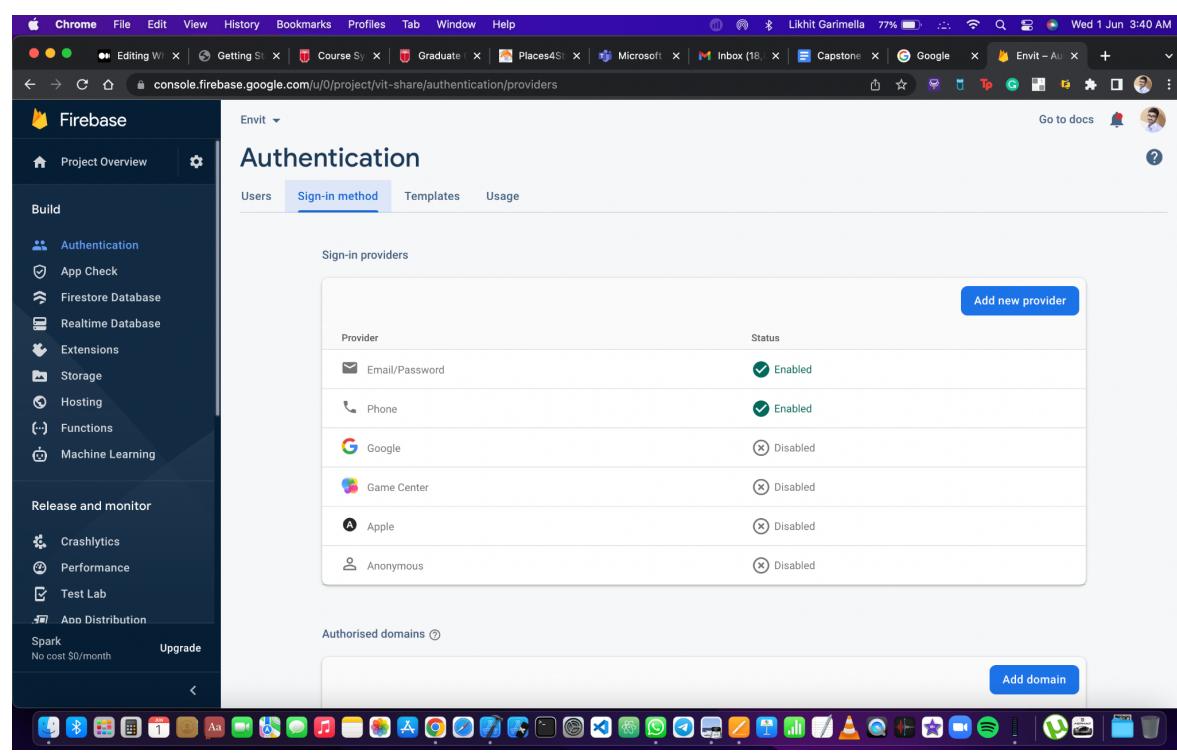
Screenshots:



The screenshot shows the Firebase Authentication console under the 'Users' tab. It displays a list of users with their email addresses, provider types, creation dates, sign-in dates, and user UIDs. A search bar at the top allows filtering by email or UID.

Identifier	Providers	Created	Signed in	User UID
likhit.garimella2018@vitstu...	✉️	30 Mar 2022	31 May 2022	erdHHezyWoYyVJ98f6NoxGJhFn...
aparna.sunil2018@vitstud...	✉️	1 Jun 2022	1 Jun 2022	KWC60vf73LeSRn3HoqjEYZutj4H3

Rows per page: 50 | 1 – 2 of 2



The screenshot shows the Firebase Authentication console under the 'Sign-in method' tab. It lists various sign-in providers: Email/Password (Enabled), Phone (Enabled), Google (Disabled), Game Center (Disabled), Apple (Disabled), and Anonymous (Disabled). An 'Add new provider' button is available. Below this, there is a section for 'Authorised domains' with a placeholder input field and an 'Add domain' button.

The screenshot shows the Firebase Realtime Database interface in a web browser. The left sidebar lists various services: Authentication, App Check, Firestore Database, Realtime Database (selected), Extensions, Storage, Hosting, Functions, and Machine Learning. The main area displays the Realtime Database structure under the URL <https://vit-share.firebaseio.com>. The tree view shows the following structure:

- https://vit-share.firebaseio.com/
 - J-Component Projects
 - Mentees
 - Mentors
 - My-Jcomp-Projects
 - My-Mentee-Posts
 - My-Mentor-Posts
 - My-Pers-Projects
 - Personal Projects
 - Users

Database location: United States (us-central1)

This screenshot shows a detailed view of a node in the Firebase Realtime Database. The URL is <https://vit-share.firebaseio.com>. The tree view shows the following structure under the J-Component Projects node:

- J-Component Projects
 - Details
 - N3R-ks95ZtSJES0QFLq
 - 1) Title: "Spam Filtering In Sms"
 - 2) Course: "Natural Language Processing"
 - 3) Description: "Predict and categorise filtered and unfiltered spam in sms using NLP techniques. Implementation working and available."
 - 4) Timestamp: 1654032369
 - 5) uid: "erdHHezyWoYyVJ98f6NoxGJhFnM2"
 - likeCount: 4
 - likes
 - erdHHezyWoYyVJ98f6NoxGJhFnM2: true
 - N3R0-xscPN5q2bX01Gw

Database location: United States (us-central1)

The screenshot shows the Firebase Realtime Database interface in a web browser. The left sidebar lists various services: Authentication, App Check, Firestore Database, **Realtime Database**, Extensions, Storage, Hosting, Functions, and Machine Learning. The main area displays the Realtime Database structure under the 'Data' tab. The URL in the address bar is `https://vit-share.firebaseio.com`. The database structure is as follows:

```
erdHHezyWoYyVJ98f6NoxGJhFnM2: true
  Mentees
    Details
      -N3NZ1P_REC_-m_nCfGd
        1) Domain: "ML"
        2) Query: "Want to learn ML algorithms from an online course for beginners"
        3) Timestamp: 1653974701
        4) uid: "erdHHezyWoYyVJ98f6NoxGJhFnM2"
        likeCount: 3
        likes
          erdHHezyWoYyVJ98f6NoxGJhFnM2: true
          -N3NZydjMzTRDdofkO1s
            1) Domain: "Design"
```

Database location: United States (us-central)

The screenshot shows the Firebase Realtime Database interface in a web browser. The left sidebar lists various services: Authentication, App Check, Firestore Database, **Realtime Database**, Extensions, Storage, Hosting, Functions, and Machine Learning. The main area displays the Realtime Database structure under the 'Data' tab. The URL in the address bar is `https://vit-share.firebaseio.com`. The database structure is as follows:

```
Mentors
  Details
    -N3NUdbk0auxMS_Vc2EU
      1) Domain: "iOS"
      2) Experience: "3 years of Full-stack iOS dev using Swift"
      3) Prerequisites: "Xcode, GitHub"
      4) Courses: "Udemy Angela Yu"
      5) Timestamp: 1653973359
      6) uid: "erdHHezyWoYyVJ98f6NoxGJhFnM2"
      likeCount: 4
      likes
        erdHHezyWoYyVJ98f6NoxGJhFnM2: true
        -N3NV7hh-kTHOhsq-DvW
          1) Domain: "Web"
```

Database location: United States (us-central)

The screenshot shows the Firebase Realtime Database interface. On the left, a sidebar lists various services: Authentication, App Check, Firestore Database, **Realtime Database**, Extensions, Storage, Hosting, Functions, and Machine Learning. The 'Realtime Database' service is selected. The main area displays the 'Realtime Database' dashboard with tabs for Data, Rules, Backups, and Usage. The Data tab shows a hierarchical database structure under 'https://vit-share.firebaseio.com'. One node, 'Personal Projects', contains a child node 'Details' which has a child node '-N3R-2T3LzeyXit48170'. This node contains several key-value pairs: 'Title' (EnVIT - A Social Interface For VITians), 'Role' (UI-UX designer), 'Description' (Looking for a Figma designer to collaborate on my aforementioned project. It's a social media mobile application that con...), 'Timestamp' (1654032185), and 'uid' (erdhHezyWoYyVJ98f6NoxGJhFnM2). Another child node 'likes' contains a single entry for user 'erdhHezyWoYyVJ98f6NoxGJhFnM2' with a value of true. A second post node '-N3R-GkIHhnutnRN1Xd' is also present with the title 'Agro'. At the bottom, a note states 'Database location: United States (us-central)'.

This screenshot shows another view of the Firebase Realtime Database. The sidebar and dashboard are identical to the first one. The Data tab shows a different database structure. A 'likes' node contains a single entry for user 'erdhHezyWoYyVJ98f6NoxGJhFnM2' with a value of true. Below it is a 'Users' node with two children: 'KWC60vf73LeSRn3HoqIEYZutj4H3' and 'erdhHezyWoYyVJ98f6NoxGJhFnM2'. The 'KWC60vf73LeSRn3HoqIEYZutj4H3' node contains four key-value pairs: 'Name' (Aparna), 'Email' (aparna.sunil2018@vitstudent.ac.in), 'Phone' (97466540732), and 'Block' (F). The 'erdhHezyWoYyVJ98f6NoxGJhFnM2' node also contains four key-value pairs: 'Name' (Likhit), 'Email' (likhit.garimella2018@vitstudent.ac.in), 'Phone' (9003760414), and 'Block' (Q). A note at the bottom indicates 'Database location: United States (us-central)'.

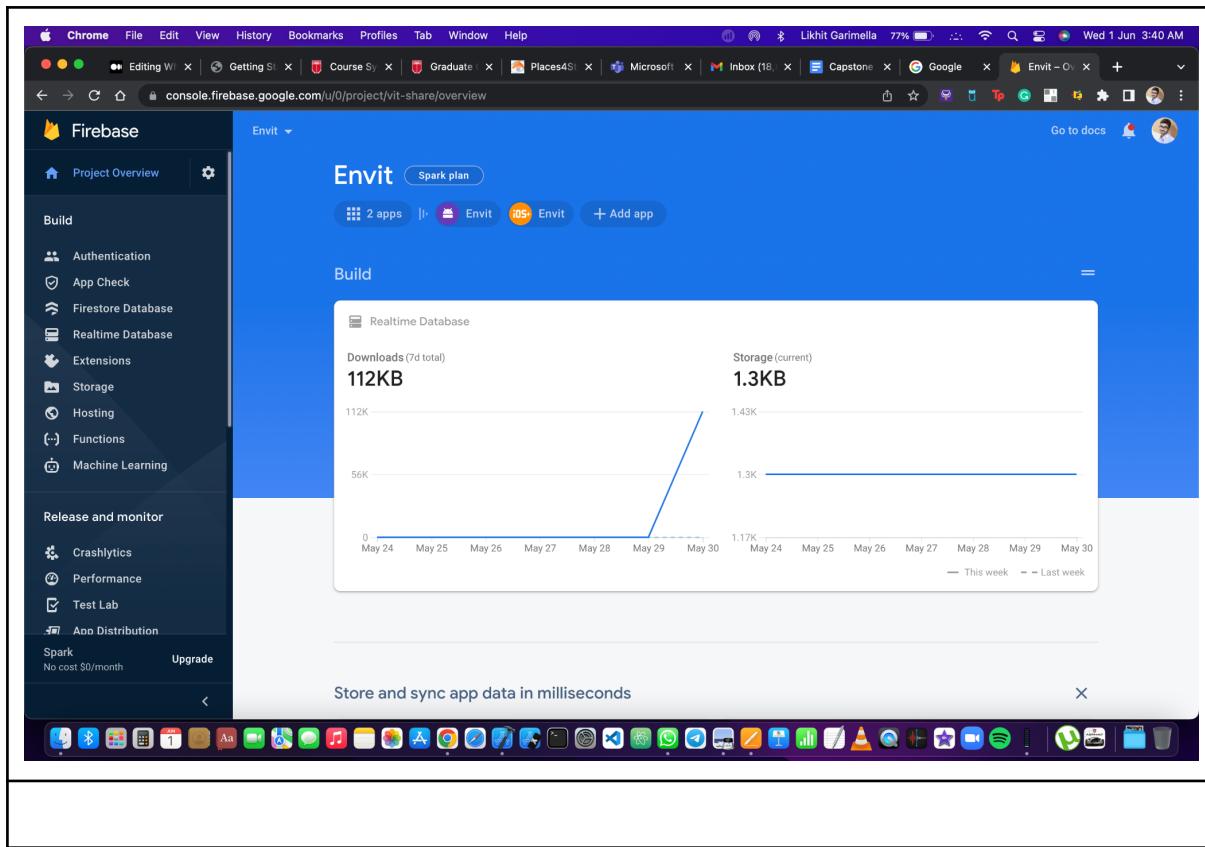


Fig 13: Firebase Server Screenshots

The screenshot shows the Xcode interface with the file `SignUpViewController.swift` open. The code handles a tap event on a button. It dismisses the keyboard, shows a progress HUD, performs validations on email and password, and then signs up the user using an AuthService. If successful, it shows a welcome message and performs a segue to the home screen. If there's an error, it prints the error string and dismisses the HUD.

```
    @IBAction func registerTapped(_ sender: UIButton) {
        // dismiss keyboard
        view.endEditing(true)

        // Progress HUD
        let hud1 = JGProgressHUD(style: .dark)
        hud1.textLabel.text = "Please Wait..."
        hud1.show(in: self.view)

        // validations
        guard let email = emailId.text, let password = pass.text, let name = name.text, let phone = phoneNo.text, let block =
            block.text else {
            print("Invalid Form Input")
            return
        }

        // Auth service sign up
        AuthService.signUp(name: name, email: email, phone: phone, block: block, password: password, onSuccess: {
            print("On Success")
            hud1.indicatorView = nil // remove indicator
            hud1.textLabel.text = "Welcome!"
            hud1.dismiss(afterDelay: 2.0, animated: true)
            // segue to tab bar VC
            self.performSegue(withIdentifier: "goToHome", sender: self)
        }) {errorString in
            // this will be the one which prints error due to auth, in console
            print(errorString!)
            hud1.indicatorView = nil // remove indicator
            hud1.textLabel.text = errorString!
            hud1.dismiss(afterDelay: 2.0, animated: true)
        }
    }
}
```

The screenshot shows the Xcode interface with the file `LoginViewController.swift` open. The code handles a tap event on a button. It dismisses the keyboard, shows a progress HUD, performs validations on email and password, and then signs in the user using an AuthService. If successful, it shows a logged-in message and performs a segue to the home screen. If there's an error, it prints the error string and dismisses the HUD.

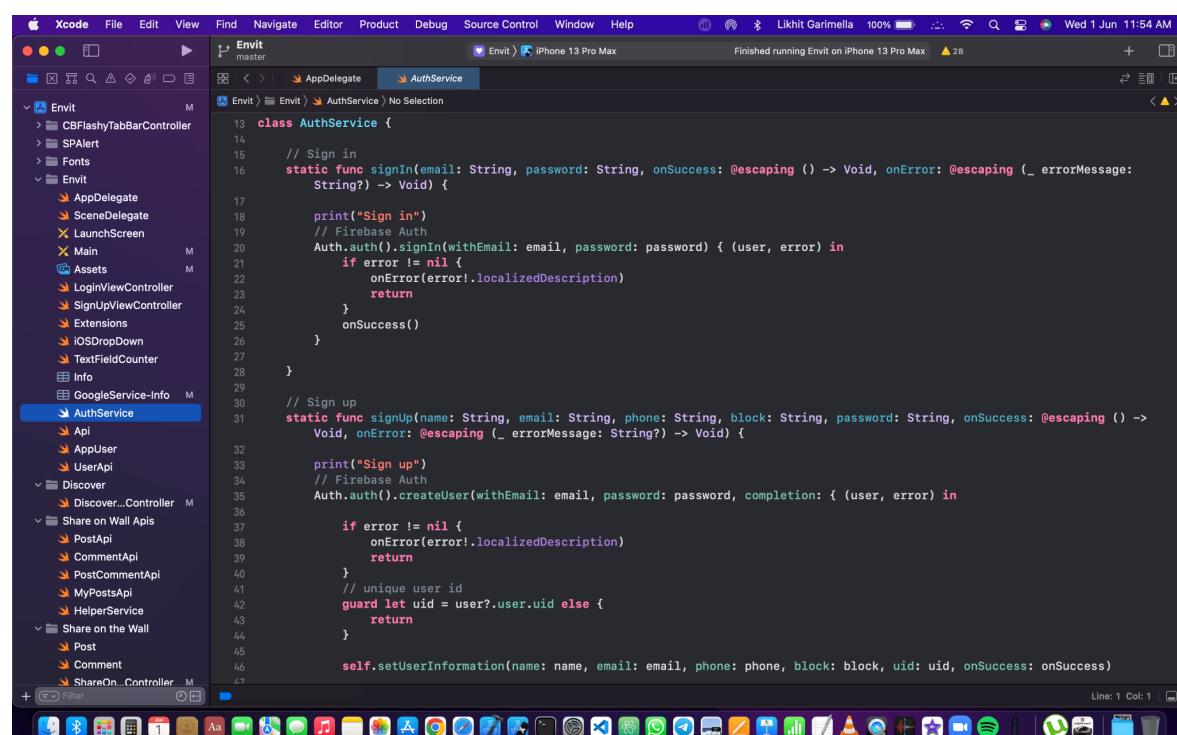
```
    @IBAction func loginTapped(_ sender: UIButton) {
        // dismiss keyboard
        view.endEditing(true)

        // Progress HUD
        let hud1 = JGProgressHUD(style: .dark)
        hud1.textLabel.text = "Please Wait..."
        hud1.show(in: self.view)

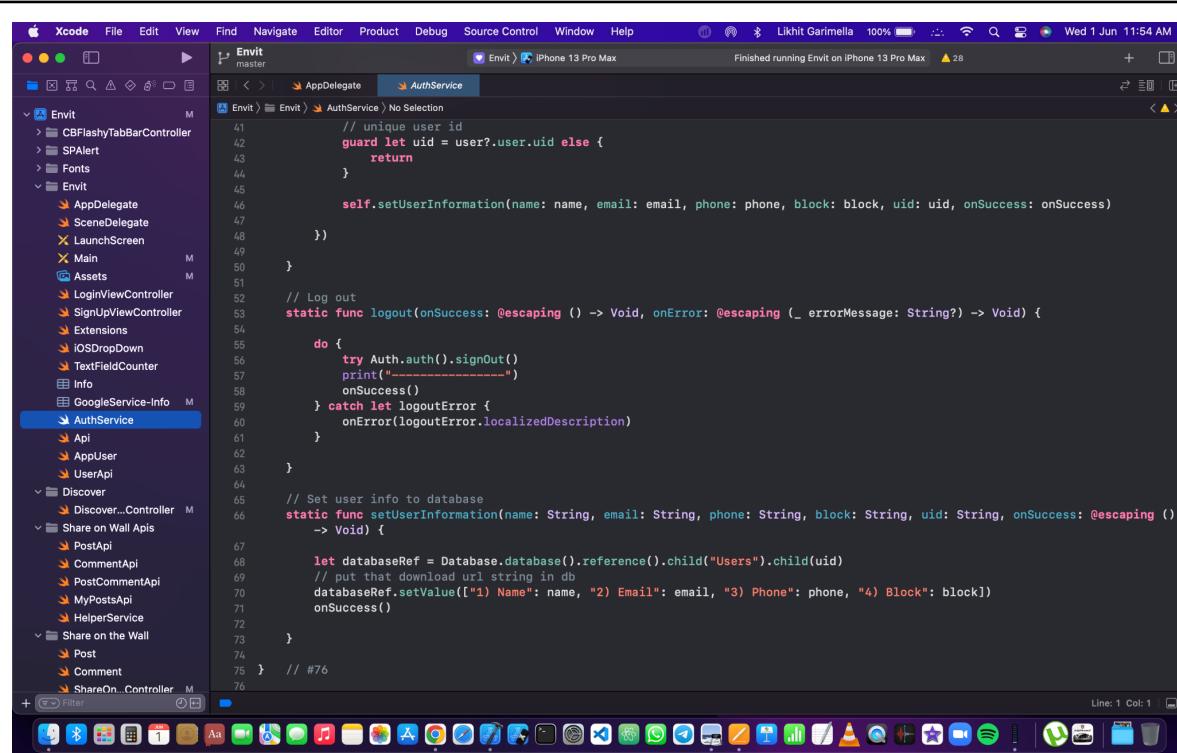
        // validations
        guard let email = emailId.text, let password = password.text else {
            print("Invalid Form Input")
            return
        }

        // Auth service sign in
        AuthService.signIn(email: email, password: password, onSuccess: {
            print("On Success")
            hud1.indicatorView = nil // remove indicator
            hud1.textLabel.text = "Logged In!"
            hud1.dismiss(afterDelay: 2.0, animated: true)
            // segue to tab bar VC
            self.performSegue(withIdentifier: "goToHome", sender: self)
        }, onError: {errorString in
            // this will be the one which prints error due to auth, in console
            print(errorString!)
            hud1.indicatorView = nil // remove indicator
            hud1.textLabel.text = errorString!
            hud1.dismiss(afterDelay: 2.0, animated: true)
        })
    }

    @IBAction func signupTapped(_ sender: UIButton) {
        self.performSegue(withIdentifier: "goToSignup", sender: self)
    }
}
```



```
13 class AuthService {
14     // Sign in
15     static func signIn(email: String, password: String, onSuccess: @escaping () -> Void, onError: @escaping (_ errorMessage: String?) -> Void) {
16         print("Sign in")
17         // Firebase Auth
18         Auth.auth().signIn(withEmail: email, password: password) { (user, error) in
19             if error != nil {
20                 onError(error!.localizedDescription)
21             } else {
22                 onSuccess()
23             }
24         }
25     }
26
27     // Sign up
28     static func signUp(name: String, email: String, phone: String, block: String, password: String, onSuccess: @escaping () -> Void, onError: @escaping (_ errorMessage: String?) -> Void) {
29         print("Sign up")
30         // Firebase Auth
31         Auth.auth().createUser(withEmail: email, password: password, completion: { (user, error) in
32             if error != nil {
33                 onError(error!.localizedDescription)
34             } else {
35                 // unique user id
36                 guard let uid = user?.user.uid else {
37                     return
38                 }
39                 // Set user info to database
40                 self.setUserInfo(name: name, email: email, phone: phone, block: block, uid: uid, onSuccess: onSuccess)
41             }
42         })
43     }
44 }
```



```
45     // unique user id
46     guard let uid = user?.user.uid else {
47         return
48     }
49
50     self.setUserInfo(name: name, email: email, phone: phone, block: block, uid: uid, onSuccess: onSuccess)
51 }
52
53 // Log out
54 static func logout(onSuccess: @escaping () -> Void, onError: @escaping (_ errorMessage: String?) -> Void) {
55     do {
56         try Auth.auth().signOut()
57         print("-----")
58         onSuccess()
59     } catch let logoutError {
60         onError(logoutError.localizedDescription)
61     }
62 }
63
64 // Set user info to database
65 static func setUserInfo(name: String, email: String, phone: String, block: String, uid: String, onSuccess: @escaping () -> Void) {
66     let databaseRef = Database.database().reference().child("Users").child(uid)
67     // put that download url string in db
68     databaseRef.setValue(["1 Name": name, "2 Email": email, "3 Phone": phone, "4 Block": block])
69     onSuccess()
70 }
71
72 }
73 }
74 }
75 } // #76
```

```

1 // 
2 //  AppUser.swift
3 //  Envit
4 //
5 //  Created by Likhit Garimella on 16/06/20.
6 //  Copyright © 2020 Likhit Garimella. All rights reserved.
7 //
8
9 import Foundation
10
11 class AppUser {
12
13     var nameString: String?
14     var emailString: String?
15     var phoneString: String?
16     var blockString: String?
17
18     var id: String?
19
20 }
21
22 extension AppUser {
23
24     static func transformUser(dict: [String: Any], key: String) -> AppUser {
25
26         let user = AppUser()
27         user.nameString = dict["1"] Name] as? String
28         user.emailString = dict["2"] Email] as? String
29         user.phoneString = dict["3"] Phone] as? String
30         user.blockString = dict["4"] Block] as? String
31
32         user.id = key
33         return user
34     }
35
36 }
37 } // #38

```

```

13 // Write your own Api, to conveniently observe database data...
14
15 class UserApi {
16
17     var REF_USERS = Database.database().reference().child("Users")
18
19     func observeUser(withId uid: String, completion: @escaping (AppUser) -> Void) {
20
21         REF_USERS.child(uid).observeSingleEvent(of: .value, with: { (snapshot) in
22
23             if let dict = snapshot.value as? [String:Any] {
24                 let user = AppUser.transformUser(dict: dict, key: snapshot.key)
25                 completion(user)
26             }
27         })
28     }
29
30
31
32     func observeCurrentUser(completion: @escaping (AppUser) -> Void) {
33
34         guard let currentUser = Auth.auth().currentUser else {
35             return
36         }
37
38         REF_USERS.child(currentUser.uid).observeSingleEvent(of: .value, with: { (snapshot) in
39
40             // transform data snapshot to user object
41             if let dict = snapshot.value as? [String:Any] {
42                 let user = AppUser.transformUser(dict: dict, key: snapshot.key)
43                 completion(user)
44             }
45         })
46     }
47
48 }
49

```

Xcode Screenshot showing the UserApi.swift file in the Envit project. The code implements a function to observe users from a database snapshot.

```

func observeUsers(completion: @escaping (AppUser) -> Void) {
    REF_USERS.observe(.childAdded, with: {
        snapshot in
        if let dict = snapshot.value as? [String:Any] {
            let user = AppUser.transformUser(dict: dict, key: snapshot.key)
            // Display list of users in 'Discover users' excluding the current user in that
            if user.id != Api.UserDet.CURRENT_USER?.uid {
                completion(user)
            }
        }
    })
}

/// This will be the search text that we get from users
// func queryUsers
// ...

var CURRENT_USER: User? {
    if let currentUser = Auth.auth().currentUser {
        return currentUser
    }
    return nil
}

var REF_CURRENT_USER: DatabaseReference? {
    guard let currentUser = Auth.auth().currentUser else {
        return nil
    }
    return REF_USERS.child(currentUser.uid)
}

```

Xcode Screenshot showing the TechShareViewController.swift file in the Envit project. The code defines outlets and implements methods for button actions.

```

// Outlets
@IBOutlet var goToFeedOutlet: UIButton!
@IBOutlet var mentorOutlet: UIButton!
@IBOutlet var menteeOutlet: UIButton!

func ButtonsProp() {
    goToFeedOutlet.layer.cornerRadius = 18
    mentorOutlet.layer.cornerRadius = 10
    menteeOutlet.layer.cornerRadius = 10
}

override func viewDidLoad() {
    super.viewDidLoad()
    ButtonsProp()
}

@IBAction func goToFeedAction(_ sender: UIButton) {
    self.performSegue(withIdentifier: "goToFeed", sender: self)
}

@IBAction func mentorAction(_ sender: UIButton) {
    self.performSegue(withIdentifier: "goToMentor", sender: self)
}

@IBAction func menteeAction(_ sender: UIButton) {
    self.performSegue(withIdentifier: "goToMentee", sender: self)
}

```

Xcode Screenshot (iPhone 13 Pro Max) showing the implementation of `MentorFeedViewController`. The code handles dequeuing a reusable cell for the mentor post collection view.

```

Envit master Envit iPhone 13 Pro Max Finished running Envit on iPhone 13 Pro Max ▲ 28
Envit < > AppDelegate MentorViewController MentorFeedViewController
Envit < > Mentor < > MentorFeedViewController No Selection
35     } /*
36
37     let mentorCell = mentorFeedCollectionView.dequeueReusableCell(withIdentifier: "MentorPostCell", for: indexPath) as! MentorPostCell
38     let post = mentorPosts[indexPath.row]
39     let user = users[indexPath.row]
40     mentorCell.mentorPost = post
41     mentorCell.user = user
42     // linking home VC & home table view cell
43     mentorCell.mentorFeedVC = self
44     return mentorCell
45
46 }
47
48 // reference to store MentorModel class info
49 var mentorPosts = [MentorModel]()
50
51 // reference to store User class info
52 var users = [AppUser]()
53
54 // DB ref
55 // ---- var refMentors: DatabaseReference!
56
57 // load mentor posts
58 func loadPosts() {
59
60     // start when loadPosts func starts
61     activityIndicatorView.startAnimating()
62
63     Api.MentorPost.observePosts { (post) in
64         guard let postId = post.uid else {
65             return
66         }
67         // fetch user data in mentor posts
68         self.fetchUserId(postId, completed: {
69             self.mentorPosts.append(post)
70             print(self.mentorPosts)
71             // stop before tableview reloads data
72             self.activityIndicatorView.stopAnimating()
73             self.activityIndicatorView.hidesWhenStopped = true
74             self.mentorFeedCollectionView.reloadData()
75         })
76     }
77 }

```

Xcode Screenshot (iPhone 13 Pro Max) showing the implementation of `HelperServiceMentee`. It includes methods for uploading data to the server and sending data to the database.

```

Envit master Envit iPhone 13 Pro Max Finished running Envit on iPhone 13 Pro Max ▲ 28
Envit < > AppDelegate MentorViewController MentorPostCell HelperServiceMentee
Envit < > Mentor < > HelperServiceMentee No Selection
11 class HelperServiceMentee {
12
13     static func uploadDataToServer(domainText: String, experienceText: String, prerequisitesText: String, coursesText: String, onSuccess: @escaping () -> Void) {
14
15         self.sendDataToDatabase(domainText: domainText, experienceText: experienceText, prerequisitesText: prerequisitesText, coursesText: coursesText, onSuccess: onSuccess)
16
17     }
18
19     static func sendDataToDatabase(domainText: String, experienceText: String, prerequisitesText: String, coursesText: String, onSuccess: @escaping () -> Void) {
20
21         let newPostId = Api.MentorPost.REF_POSTS.childByAutoId().key
22         let newPostReference = Api.MentorPost.REF_POSTS.child(newPostId)
23         guard let currentUser = Api.UserRef.CURRENT_USER else {
24             return
25         }
26         let currentUserId = currentUser.uid
27         // Creating a timestamp
28         let timestamp = NSNumber(value: Int(NSTimeIntervalSince1970))
29         // put that download url string in db
30
31         newPostReference.setValue(["1) Domain": domainText, "2) Experience": experienceText, "3) Prerequisites": prerequisitesText, "4) Courses": coursesText, "5) Timestamp": timestamp, "6) uid": currentUserId], withCompletionBlock: { (error, ref) in
32             if error != nil {
33                 print(error.localizedDescription)
34             }
35
36             // reference for my posts
37             let myPostRef = Api.MyMentorPosts.REF_MYPOSTS.child(currentUserId).child(newPostId)
38             myPostRef.setValue(true, withCompletionBlock: {
39                 (error, ref) in
40                     if error != nil {
41                         print(error.localizedDescription)
42                     }
43                 })
44             onSuccess()
45         })
46     }
47 }

```

```

33     self.id = id
34     self.domainText = domainText
35     self.experienceText = experienceText
36     self.prerequisitesText = prerequisitesText
37     self.coursesText = coursesText
38     self.timestamp = timestamp
39   }
40 }
41 }
42
43 extension MentorModel {
44
45   static func transformMentorPost(dict: [String: Any], key: String) -> MentorModel {
46
47     let mentorPost = MentorModel()
48     // Remodel Post class, bcz it currently doesn't have a post id property
49     mentorPost.id = key
50     mentorPost.domainText = dict["1") Domain"] as? String
51     mentorPost.experienceText = dict["2") Experience"] as? String
52     mentorPost.prerequisitesText = dict["3") Prerequisites"] as? String
53     mentorPost.coursesText = dict["4") Courses"] as? String
54     mentorPost.uid = dict["5") uid"] as? String
55
56     mentorPost.likeCount = dict["likeCount"] as? Int
57     mentorPost.likes = dict["likes"] as? Dictionary<String, Any>
58
59     if let currentUserID = Auth.auth().currentUser?.uid {
60       if mentorPost.likes != nil {
61         /* if post.likes[currentUserID] != nil {
62           post.isLiked = true
63         } else {
64           post.isLiked = false
65         } */
66         // Above commented snippet can be put in 1 line...as below.
67         mentorPost.isLiked = mentorPost.likes![currentUserID] != nil
68       }
69     }
70
71     return mentorPost
72
73   }
74 }

```

```

14 class MentorPostApi {
15
16   var REF_POSTS = Database.database().reference().child("Mentors").child("Details")
17
18   func observePosts(completion: @escaping (MentorModel) -> Void) {
19
20     REF_POSTS.observe(.childAdded, with: { (snapshot) in
21
22       if let dict = snapshot.value as? [String: Any] {
23         let newPost = MentorModel.transformMentorPost(dict: dict, key: snapshot.key)
24         completion(newPost)
25       }
26     })
27   }
28
29 }
30
31 func observePost(withId id: String, completion: @escaping (MentorModel) -> Void) {
32
33   REF_POSTS.child(id).observeSingleEvent(of: .value, with: {
34     snapshot in
35
36       if let dict = snapshot.value as? [String: Any] {
37         let post = MentorModel.transformMentorPost(dict: dict, key: snapshot.key)
38         completion(post)
39       }
40     })
41   }
42 }
43
44
45 func observeLikeCount(withPostId id: String, completion: @escaping (Int) -> Void) {
46
47   REF_POSTS.child(id).observe(.childChanged, with: {
48     snapshot in
49       print(snapshot)
50       if let value = snapshot.value as? Int {
51         completion(value)
52       }
53     })
54 }

```

```
func updateView() {
    domainName.text = mentorPost?.domainText
    experienceTextView.text = mentorPost?.experienceText
    courseTextView.text = mentorPost?.coursesText
    prerequisiteTextView.text = mentorPost?.prerequisitesText

    setupUserInfo()

    /// Update like
    updateLike(post: mentorPost)
    /// New
    self.updateLike(post: self.mentorPost!)
}

func updateLike(post: MentorModel) {
    /// we first checked if its true, and no one liked this post before...
    /// or if probably someone did, but the current user did not...
    /// then we display non-selected like icon...
    /// otherwise, display likeSelected icon...
    let imageName = post.likes == nil || !post.isLiked! ? "Icon1" : "likeSelected"
    mentorLikeImageView.image = UIImage(named: imageName)
    /// Below commented snippet can be put in 1 line, as above...
    /* if post.isLiked == false {
        likeImageView.image = UIImage(named: "like")
    } else {
        likeImageView.image = UIImage(named: "likeSelected")
    }*/
    // We now update like count
    /// Use optional chaining with guard
    guard let count = post.likeCount else {
        return
    }
    if count != 0 {
        likeCountButton.setTitle("\(count) likes", for: .normal)
    } else {
        likeCountButton.setTitle("0 likes", for: .normal)
    }
}
```

```
@IBAction func submitTapped(_ sender: UIButton) {
    sender.flash()

    if (domain.text!.isEmpty || experienceTextView.text!.isEmpty) {
        // Alert for empty fields
        let myAlert = UIAlertController(title: "Empty Fields", message: "", preferredStyle: UIAlertController.Style.alert)
        let okAction = UIAlertAction(title: "Ok", style: UIAlertAction.Style.default, handler: nil)
        myAlert.addAction(okAction)
        self.present(myAlert, animated: true, completion: nil)
        return
    }

    HelperServiceMentor.uploadDataToServer(domainText: domain.text!, experienceText: experienceTextView.text!.trimmingCharacters(in: .whitespacesAndNewlines), prerequisitesText: prerequisitesTextView.text!.trimmingCharacters(in: .whitespacesAndNewlines), coursesText: coursesTextView.text!.trimmingCharacters(in: .whitespacesAndNewlines), onSuccess: {
        // Alert pod - Work Added
        let alertView = SPAlertView(title: "Your work has been added", message: nil, preset: SPAlertPreset.done)
        alertView.duration = 1.2
        alertView.present()
    })

    // Clear textfields after success
    self.domain.text = ""
    self.experienceTextView.text = ""
    self.prerequisites.text = ""
    self.courses.text = ""

    // And to enable back for a new input in textfield
    self.domain.isEnabled = true
    self.experienceTextView.isEnabled = true
    self.prerequisites.isEnabled = true
    self.courses.isEnabled = true
}
```

The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure under "Envit".
- Search Bar:** Contains "Envit" and "iPhone 13 Pro Max".
- Status Bar:** Shows "Finished running Envit on iPhone 13 Pro Max" and battery level.
- Code Editor:** Displays the code for `MenteeFeedViewController`. The code handles dequeuing reusable cells for mentee posts, setting up user and mentee model references, and observing posts from the API. It also fetches user data for each post and updates the activity indicator.

```
36     let menteeCell = menteeFeedCollectionView.dequeueReusableCell(withIdentifier: "MenteePostCell", for: indexPath) as! MenteePostCell
37     let post = menteePosts[indexPath.row]
38     let user = users[indexPath.row]
39     menteeCell.mentorPost = post
40     menteeCell.user = user
41     // linking home VC & home table view cell
42     menteeCell.menteeFeedVC = self
43     return menteeCell
44
45 }
46
47 // reference to store MenteeModel class info
48 var menteePosts = [MenteeModel]()
49
50 // reference to store User class info
51 var users = [AppUser]()
52
53 // DB ref
54
55 // ---- var refMentees: DatabaseReference!
56
57 // load mentee posts
58 func loadPosts() {
59
60     // start when loadPosts func starts
61     activityIndicatorView5.startAnimating()
62
63     Api.MenteePost.observePosts { (post) in
64         guard let postId = post.uid else {
65             return
66         }
67         // fetch user data in mentee posts
68         self.fetchUser(uid: postId, completed: {
69             self.menteePosts.append(post)
70             print(self.menteePosts)
71             // stop before tableView reloads data
72             self.activityIndicatorView5.stopAnimating()
73             self.activityIndicatorView5.hidesWhenStopped = true
74             self.menteeFeedCollectionView.reloadData()
75         })
76     }
77 }
```

Bottom Bar: Includes icons for various Xcode tools and the status bar with "Line: 1 Col: 1".

The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure under "Project Share". Key components include:
 - Mentee:** Contains `HelperServiceMentee`.
 - Mentee APIs:** Contains `MenteePostApi`, `MenteeCommentApi`, `MenteePos...commentApi`, `MyMenteePostsApi`, and `HelperServiceMentee`.
 - Mentor:** Contains `MentorPostCell`, `MentorPostController`, `MentorFee...Controller`, and `MentorCommentApi`.
 - Mentor APIs:** Contains `MentorPostApi`, `MentorCommentApi`, `MentorPos...commentApi`, `MyMentorPostsApi`, and `HelperServiceMentor`.
- Code Editor:** The main window displays the `HelperServiceMentee` class with the following code:

```
class HelperServiceMentee {  
    static func uploadDataToServer(domainText: String, queryText: String, onSuccess: @escaping () -> Void) {  
        self.sendDataToDatabase(domainText: domainText, queryText: queryText, onSuccess: onSuccess)  
    }  
  
    static func sendDataToDatabase(domainText: String, queryText: String, onSuccess: @escaping () -> Void) {  
        let newPostId = Api.MenteePost.REF_POSTS.childByAutoId().key  
        let newPostReference = Api.MenteePost.REF_POSTS.child(newpostId!)  
        guard let currentUser = Api.UserData.CURRENT_USER else {  
            return  
        }  
  
        let currentUserID = currentUser.uid  
        // Creating a timestamp  
        let timestamp = NSNumber(value: Int(NSTimeInterval().timeIntervalSince1970))  
        // put that download url string in db  
  
        newPostReference.setValue(["1": Domain: domainText, "2": Query: queryText, "3": Timestamp: timestamp, "4": uid: currentUserID],  
            withCompletionBlock: { (error, ref) in  
                if error != nil {  
                    print(error!.localizedDescription)  
                    return  
                }  
  
                // reference for my posts  
                let myPostRef = Api.MyMenteePosts.REF_MYPOSTS.child(currentUserId).child(newPostId!)  
                myPostRef.setValue(true, withCompletionBlock: {  
                    (error, ref) in  
                    if error != nil {  
                        print(error!.localizedDescription)  
                        return  
                    }  
                })  
                onSuccess()  
            })  
    }  
}
```
- Top Bar:** Shows the file path "Envit master", the target "iPhone 13 Pro Max", and the status "Finished running Envit on iPhone 13 Pro Max".
- Bottom Bar:** Shows various Xcode icons and the status bar indicating "Line: 1 Col: 1".

Envit master Envit iPhone 13 Pro Max Finished running Envit on iPhone 13 Pro Max ▲ 28 Wed 1 Jun 7:53 PM

```

 29     /*
 30      init(id: String?, domainText: String?, postQueryText: String?, timestamp: Double?) {
 31          self.id = id
 32          self.domainText = domainText
 33          self.postQueryText = postQueryText
 34          self.timestamp = timestamp
 35      }
 36  */
 37
 38 extension MenteeModel {
 39
 40     static func transformMenteePost(dict: [String: Any], key: String) -> MenteeModel {
 41
 42         let menteePost = MenteeModel()
 43         /// Remodel Post class, bcz it currently doesn't have a post id property
 44         menteePost.id = key
 45         menteePost.domainText = dict["1") Domain"] as? String
 46         menteePost.postQueryText = dict["2") Query"] as? String
 47         menteePost.uid = dict["4") uid"] as? String
 48
 49         menteePost.likeCount = dict["likeCount"] as? Int
 50         menteePost.likes = dict["likes"] as? Dictionary<String, Any>
 51
 52         if let currentUserID = Auth.auth().currentUser?.uid {
 53             if menteePost.likes != nil {
 54                 /* if post.likes[currentUserID] != nil {
 55                     post.isLiked = true
 56                 } else {
 57                     post.isLiked = false
 58                 }*/
 59                 // Above commented snippet can be put in 1 line.. as below.
 60                 menteePost.isLiked = menteePost.likes![currentUserID] != nil
 61             }
 62         }
 63
 64         return menteePost
 65     }
 66 }
 67
 68 } // #70

```

Envit master Envit iPhone 13 Pro Max Finished running Envit on iPhone 13 Pro Max ▲ 28 Wed 1 Jun 7:54 PM

```

 15 var REF_POSTS = Database.database().reference().child("Mentees").child("Details")
 16
 17 func observePosts(completion: @escaping (MenteeModel) -> Void) {
 18
 19     REF_POSTS.observe(.childAdded, with: { (snapshot) in
 20
 21         if let dict = snapshot.value as? [String: Any] {
 22             let newPost = MenteeModel.transformMenteePost(dict: dict, key: snapshot.key)
 23             completion(newPost)
 24         }
 25     })
 26
 27 }
 28
 29
 30 func observePost(withId id: String, completion: @escaping (MenteeModel) -> Void) {
 31
 32     REF_POSTS.child(id).observeSingleEvent(of: .value, with: {
 33         snapshot in
 34
 35         if let dict = snapshot.value as? [String: Any] {
 36             let post = MenteeModel.transformMenteePost(dict: dict, key: snapshot.key)
 37             completion(post)
 38         }
 39     })
 40 }
 41
 42
 43 func observeLikeCount(withId postId: String, completion: @escaping (Int) -> Void) {
 44
 45     REF_POSTS.child(postId).observe(.childChanged, with: {
 46         snapshot in
 47             print(snapshot)
 48             if let value = snapshot.value as? Int {
 49                 completion(value)
 50             }
 51         })
 52     }
 53 }
 54
 55

```

This screenshot shows the Xcode interface with the file `MenteePostCell.swift` open. The code implements a `UITableViewCell` for displaying a post from a mentee. It includes methods for updating the view and handling likes. The `updateView()` method sets the domain name and query text. The `updateLike(post:)` method handles the logic for liking a post, including checking if it's already liked and updating the UI. The `Line: 96 Col: 6` status bar at the bottom indicates the current line and column.

```
func updateView() {
    domainName.text = menteePost?.domainText
    postedQueryLabel.text = menteePost?.postQueryText

    setupUserInfo()

    /// Update like
    updateLike(post: menteePost!)
    /// New
    self.updateLike(post: self.menteePost!)
}

func updateLike(post: MenteeModel) {
    /// we first checked if its true, and no one liked this post before...
    /// or if probably someone did, but the current user did not.
    /// then we display, non-selected like icon..
    /// otherwise, display likeSelected icon.
    let imageName = post.likes == nil || !post.isLiked ? "Icon1" : "likeSelected"
    menteeLikeImageView.image = UIImage(named: imageName)
    /// Below commented snippet can be put in 1 line.. as above.
    /*if post.isLiked == false {
        likeImageView.image = UIImage(named: "like")
    } else {
        likeImageView.image = UIImage(named: "likeSelected")
    }*/
    // We now update like count
    /// Use optional chaining with guard
    guard let count = post.likeCount else {
        return
    }
    if count != 0 {
        likeCountButton.setTitle("\(count) likes", for: .normal)
    } else {
        likeCountButton.setTitle("0 likes", for: .normal)
    }
}
```

This screenshot shows the Xcode interface with the file `MenteeViewController.swift` open. The code implements a controller for the mentee section. It handles a button action where the user taps a submit button. If the domain text or query text is empty, it shows an alert. Otherwise, it creates a timestamp, gets the current user ID, and uploads the data to a server using `HelperServiceMentee.uploadDataToServer`. The `Line: 125 Col: 12` status bar at the bottom indicates the current line and column.

```
// Submit button action
@IBAction func submitTapped(_ sender: UIButton) {
    sender.flash()

    if (domain.text!.isEmpty || postQueryTextView.text!.isEmpty) {
        // Alert for empty fields
        let myAlert = UIAlertController(title: "Empty Fields", message: "", preferredStyle: UIAlertController.Style.alert)
        let okAction = UIAlertAction(title: "Ok", style: UIAlertAction.Style.default, handler: nil)
        myAlert.addAction(okAction)
        self.present(myAlert, animated: true, completion: nil)
        return
    }

    /*
    // Creating a timestamp
    let timestamp = NSValue(value: Int(NSTimeIntervalSince1970))

    // Current user uid
    guard let currentUser = Auth.auth().currentUser else {
        return
    }
    let currentUserID = currentUser.uid
    */

    HelperServiceMentee.uploadDataToServer(domainText: domain.text!, queryText: postQueryTextView.text!.trimmingCharacters(in: .whitespacesAndNewlines), onSuccess: {
        // Alert pod - Query Added
        let alertView = SPAalertView(title: "Your query has been posted", message: nil, preset: SPAalertView.done)
        alertView.duration = 1.2
        alertView.present()
    })

    // Clear textfields after success
    self.domain.text = ""
    self.postQueryTextView.text = ""

    // And to enable back for a new input in textfield
    self.domain.isEnabled = true
    // self.experienceTextview.isEnabled = true
})
```

Envit master Envit iPhone 13 Pro Max Finished running Envit on iPhone 13 Pro Max ▲ 28 Wed 1 Jun 7:56 PM

```

1 // Outlets
2 @IBOutlet var addProjectOutlet: UIButton!
3 @IBOutlet var newProjectsOutlet: UIButton!
4 @IBOutlet var jComponentOutlet: UIButton!
5
6 func ButtonsProp() {
7     addProjectOutlet.layer.cornerRadius = 18
8     newProjectsOutlet.layer.cornerRadius = 10
9     jComponentOutlet.layer.cornerRadius = 10
10 }
11
12 override func viewDidLoad() {
13     super.viewDidLoad()
14
15     ButtonsProp()
16 }
17
18 @IBAction func addProjectAction(_ sender: UIButton) {
19     self.performSegue(withIdentifier: "goToAddProject", sender: self)
20 }
21
22 @IBAction func findNewProjectsAction(_ sender: UIButton) {
23     self.performSegue(withIdentifier: "goToFindNewProjects", sender: self)
24 }
25
26 @IBAction func findJComponentAction(_ sender: UIButton) {
27     self.performSegue(withIdentifier: "goToFindJComponent", sender: self)
28 }
29
30 } // #52
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51 } // #52
52

```

Envit master Envit iPhone 13 Pro Max Finished running Envit on iPhone 13 Pro Max ▲ 28 Wed 1 Jun 8:29 PM

```

1 let persProjCell = personalProjectsFeedCollectionView.dequeueReusableCell(withIdentifier: "PersProjPostCell", for: indexPath) as!
2 PersProjPostCell
3 let post = personalProjectsPosts[indexPath.row]
4 let user = users[indexPath.row]
5 persProjCell.persProjPost = post
6 persProjCell.user = user
7 // linking home VC & home table view cell
8 persProjCell.persProjFeedVC = self
9 return persProjCell
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78

```

Xcode Screenshot 1

```

11  class HelperServicePP {
12
13    static func uploadDataToServer(titleText: String, roleText: String, descText: String, onSuccess: @escaping () -> Void) {
14
15      self.sendDataToDatabase(titleText: titleText, roleText: roleText, descText: descText, onSuccess: onSuccess)
16
17  }
18
19  static func sendDataToDatabase(titleText: String, roleText: String, descText: String, onSuccess: @escaping () -> Void) {
20
21      let newPostId = Api.PersonalProjectPost.REF_POSTS.childByAutoId().key
22      let newPostReference = Api.PersonalProjectPost.REF_POSTS.child(newPostId!)
23      guard let currentUser = Api.UserData.CURRENT_USER else {
24          return
25      }
26      let currentUserID = currentUser.uid
27      // Creating a timestamp
28      let timestamp = NSNumber(value: Int(NSTimeIntervalSince1970))
29      // put that download url string in db
30
31      newPostReference.setValue(["1 Title": titleText, "2 Role": roleText, "3 Description": descText, "4 Timestamp": timestamp, "5 uid": currentUserID], withCompletionBlock: { (error, ref) in
32          if error != nil {
33              print(error.localizedDescription)
34          }
35
36          // reference for my posts
37          let myPostRef = Api.MyPosts.REF_MYPOSTS.child(currentUserID).child(newPostId!)
38          myPostRef.setValue(true, withCompletionBlock: {
39              (error, ref) in
40                  if error != nil {
41                      print(error.localizedDescription)
42                  }
43          })
44          onSuccess()
45      }
46  }
47 }
48 } // #50
49
50

```

Xcode Screenshot 2

```

19  var uid: String?
20
21  /// Remodel Post class, bcoz it currently doesn't have a post id property
22  var id: String?
23
24  ///
25  var likeCount: Int?
26  var likes: Dictionary<String, Any>?
27
28  var isLiked: Bool?
29
30 }
31
32 extension PersonalProjectModel {
33
34     static func transformPersProjPost(dict: [String: Any], key: String) -> PersonalProjectModel {
35
36         let persProjPost = PersonalProjectModel()
37         /// Remodel Post class, bcoz it currently doesn't have a post id property
38         persProjPost.id = key
39         persProjPost.projectTitle = dict["1"] Title] as? String
40         persProjPost.role = dict["2"] Role] as? String
41         persProjPost.projDesc = dict["3"] Description] as? String
42         persProjPost.uid = dict["5"] uid] as? String
43
44         persProjPost.likeCount = dict[likeCount] as? Int
45         persProjPost.likes = dict[likes] as? Dictionary<String, Any>
46
47         if let currentUserID = Auth.auth().currentUser?.uid {
48             if persProjPost.likes != nil {
49                 /* If post.likes[currentUserID] != nil {
50                     post.isLiked = true
51                 } else {
52                     post.isLiked = false
53                 }*/
54                 // Above commented snippet can be put in 1 line.. as below.
55                 persProjPost.isLiked = persProjPost.likes![currentUserID] != nil
56             }
57         }
58
59         return persProjPost
60

```

Envit master Envit iPhone 13 Pro Max Waiting to connect and unlock the device 3 28 Wed 1 Jun 8:31 PM

```

14  class PersonalProjectPostApi {
15
16    var REF_POSTS = Database.database().reference().child("Personal Projects").child("Details")
17
18    func observePosts(completion: @escaping (PersonalProjectModel) -> Void) {
19
20      REF_POSTS.observe(.childAdded, with: { [snapshot] in
21
22        if let dict = snapshot.value as? [String: Any] {
23          let newPost = PersonalProjectModel.transformPersProjPost(dict: dict, key: snapshot.key)
24          completion(newPost)
25        }
26      })
27    }
28
29  }
30
31  func observePost(withId id: String, completion: @escaping (PersonalProjectModel) -> Void) {
32
33    REF_POSTS.child(id).observeSingleEvent(of: .value, with: {
34      snapshot in
35
36        if let dict = snapshot.value as? [String: Any] {
37          let post = PersonalProjectModel.transformPersProjPost(dict: dict, key: snapshot.key)
38          completion(post)
39        }
40      }
41    })
42  }
43
44  func observeLikeCount(withPostId id: String, completion: @escaping (Int) -> Void) {
45
46    REF_POSTS.child(id).observe(.childChanged, with: {
47      snapshot in
48        print(snapshot)
49        if let value = snapshot.value as? Int {
50          completion(value)
51        }
52      })
53  }
54
55  func observeLikeCountWithPost(post: PersonalProjectModel, completion: @escaping (Int) -> Void) {
56
57    REF_POSTS.child(post.id).observe(.childChanged, with: {
58      snapshot in
59        print(snapshot)
60        if let value = snapshot.value as? Int {
61          completion(value)
62        }
63      })
64  }
65
66  func updateLike(post: PersonalProjectModel) {
67
68    // we first checked if its true, and no one liked this post before...
69    // or if probably someone did, but the current user did not...
70    // then we display, non-selected like icon...
71    // otherwise, display likeSelected icon...
72    let imageName = post.likes == nil || !post.isLiked! ? "Icon1" : "likeSelected"
73    projectLikeImageView.image = UIImage(named: imageName)
74    // Below commented snippet can be put in 1 line, as above...
75    /* if post.isLiked == false {
76      likeImageView.image = UIImage(named: "like")
77    } else {
78      likeImageView.image = UIImage(named: "likeSelected")
79    }*/
80
81    // We now update like count
82    // Use optional chaining with guard
83    guard let count = post.likeCount else {
84      return
85    }
86    if count != 0 {
87      likeCountButton.setTitle("\(count) likes", for: .normal)
88    } else {
89      likeCountButton.setTitle("@ likes", for: .normal)
90    }
91  }
92
93  func updateView() {
94
95    roleName.text = persProjPost?.role
96    projectName.text = persProjPost?.projectTitle
97    descriptionLabel.text = persProjPost?.projDesc
98    print(roleName.text)
99    print(projectName.text)
100   print(descriptionLabel.text)
101
102   setupUserInfo()
103   /// Update like
104   updateLike(post: persProjPost!)
105   /// New
106   self.updateLike(post: self.persProjPost!)
107 }
108
109 func updateUserInfo() {
110
111   // Set up user info
112   // ...
113 }
114
115 func updateLike(post: PersonalProjectModel) {
116
117   // ...
118 }
119
120 func updateView() {
121
122   // ...
123 }
124
125 func updateLike(post: PersonalProjectModel) {
126
127   // ...
128 }
129
130 func updateUserInfo() {
131
132   // ...
133 }
134
135 func updateLike(post: PersonalProjectModel) {
136
137   // ...
138 }
139
140 func updateView() {
141
142   // ...
143 }
144
145 func updateLike(post: PersonalProjectModel) {
146
147   // ...
148 }
149
150 func updateUserInfo() {
151
152   // ...
153 }
154
155 func updateLike(post: PersonalProjectModel) {
156
157   // ...
158 }
159
160 func updateView() {
161
162   // ...
163 }
164
165 func updateLike(post: PersonalProjectModel) {
166
167   // ...
168 }
169
170 func updateUserInfo() {
171
172   // ...
173 }
174
175 func updateLike(post: PersonalProjectModel) {
176
177   // ...
178 }
179
180 func updateView() {
181
182   // ...
183 }
184
185 func updateLike(post: PersonalProjectModel) {
186
187   // ...
188 }
189
190 func updateUserInfo() {
191
192   // ...
193 }
194
195 func updateLike(post: PersonalProjectModel) {
196
197   // ...
198 }
199
200 func updateView() {
201
202   // ...
203 }
204
205 func updateLike(post: PersonalProjectModel) {
206
207   // ...
208 }
209
210 func updateUserInfo() {
211
212   // ...
213 }
214
215 func updateLike(post: PersonalProjectModel) {
216
217   // ...
218 }
219
220 func updateView() {
221
222   // ...
223 }
224
225 func updateLike(post: PersonalProjectModel) {
226
227   // ...
228 }
229
230 func updateUserInfo() {
231
232   // ...
233 }
234
235 func updateLike(post: PersonalProjectModel) {
236
237   // ...
238 }
239
240 func updateView() {
241
242   // ...
243 }
244
245 func updateLike(post: PersonalProjectModel) {
246
247   // ...
248 }
249
250 func updateUserInfo() {
251
252   // ...
253 }
254
255 func updateLike(post: PersonalProjectModel) {
256
257   // ...
258 }
259
260 func updateView() {
261
262   // ...
263 }
264
265 func updateLike(post: PersonalProjectModel) {
266
267   // ...
268 }
269
270 func updateUserInfo() {
271
272   // ...
273 }
274
275 func updateLike(post: PersonalProjectModel) {
276
277   // ...
278 }
279
280 func updateView() {
281
282   // ...
283 }
284
285 func updateLike(post: PersonalProjectModel) {
286
287   // ...
288 }
289
290 func updateUserInfo() {
291
292   // ...
293 }
294
295 func updateLike(post: PersonalProjectModel) {
296
297   // ...
298 }
299
300 func updateView() {
301
302   // ...
303 }
304
305 func updateLike(post: PersonalProjectModel) {
306
307   // ...
308 }
309
310 func updateUserInfo() {
311
312   // ...
313 }
314
315 func updateLike(post: PersonalProjectModel) {
316
317   // ...
318 }
319
320 func updateView() {
321
322   // ...
323 }
324
325 func updateLike(post: PersonalProjectModel) {
326
327   // ...
328 }
329
330 func updateUserInfo() {
331
332   // ...
333 }
334
335 func updateLike(post: PersonalProjectModel) {
336
337   // ...
338 }
339
340 func updateView() {
341
342   // ...
343 }
344
345 func updateLike(post: PersonalProjectModel) {
346
347   // ...
348 }
349
350 func updateUserInfo() {
351
352   // ...
353 }
354
355 func updateLike(post: PersonalProjectModel) {
356
357   // ...
358 }
359
360 func updateView() {
361
362   // ...
363 }
364
365 func updateLike(post: PersonalProjectModel) {
366
367   // ...
368 }
369
370 func updateUserInfo() {
371
372   // ...
373 }
374
375 func updateLike(post: PersonalProjectModel) {
376
377   // ...
378 }
379
380 func updateView() {
381
382   // ...
383 }
384
385 func updateLike(post: PersonalProjectModel) {
386
387   // ...
388 }
389
390 func updateUserInfo() {
391
392   // ...
393 }
394
395 func updateLike(post: PersonalProjectModel) {
396
397   // ...
398 }
399
400 func updateView() {
401
402   // ...
403 }
404
405 func updateLike(post: PersonalProjectModel) {
406
407   // ...
408 }
409
410 func updateUserInfo() {
411
412   // ...
413 }
414
415 func updateLike(post: PersonalProjectModel) {
416
417   // ...
418 }
419
420 func updateView() {
421
422   // ...
423 }
424
425 func updateLike(post: PersonalProjectModel) {
426
427   // ...
428 }
429
430 func updateUserInfo() {
431
432   // ...
433 }
434
435 func updateLike(post: PersonalProjectModel) {
436
437   // ...
438 }
439
440 func updateView() {
441
442   // ...
443 }
444
445 func updateLike(post: PersonalProjectModel) {
446
447   // ...
448 }
449
450 func updateUserInfo() {
451
452   // ...
453 }
454
455 func updateLike(post: PersonalProjectModel) {
456
457   // ...
458 }
459
460 func updateView() {
461
462   // ...
463 }
464
465 func updateLike(post: PersonalProjectModel) {
466
467   // ...
468 }
469
470 func updateUserInfo() {
471
472   // ...
473 }
474
475 func updateLike(post: PersonalProjectModel) {
476
477   // ...
478 }
479
480 func updateView() {
481
482   // ...
483 }
484
485 func updateLike(post: PersonalProjectModel) {
486
487   // ...
488 }
489
490 func updateUserInfo() {
491
492   // ...
493 }
494
495 func updateLike(post: PersonalProjectModel) {
496
497   // ...
498 }
499
500 func updateView() {
501
502   // ...
503 }
504
505 func updateLike(post: PersonalProjectModel) {
506
507   // ...
508 }
509
510 func updateUserInfo() {
511
512   // ...
513 }
514
515 func updateLike(post: PersonalProjectModel) {
516
517   // ...
518 }
519
520 func updateView() {
521
522   // ...
523 }
524
525 func updateLike(post: PersonalProjectModel) {
526
527   // ...
528 }
529
530 func updateUserInfo() {
531
532   // ...
533 }
534
535 func updateLike(post: PersonalProjectModel) {
536
537   // ...
538 }
539
540 func updateView() {
541
542   // ...
543 }
544
545 func updateLike(post: PersonalProjectModel) {
546
547   // ...
548 }
549
550 func updateUserInfo() {
551
552   // ...
553 }
554
555 func updateLike(post: PersonalProjectModel) {
556
557   // ...
558 }
559
560 func updateView() {
561
562   // ...
563 }
564
565 func updateLike(post: PersonalProjectModel) {
566
567   // ...
568 }
569
570 func updateUserInfo() {
571
572   // ...
573 }
574
575 func updateLike(post: PersonalProjectModel) {
576
577   // ...
578 }
579
580 func updateView() {
581
582   // ...
583 }
584
585 func updateLike(post: PersonalProjectModel) {
586
587   // ...
588 }
589
590 func updateUserInfo() {
591
592   // ...
593 }
594
595 func updateLike(post: PersonalProjectModel) {
596
597   // ...
598 }
599
600 func updateView() {
601
602   // ...
603 }
604
605 func updateLike(post: PersonalProjectModel) {
606
607   // ...
608 }
609
610 func updateUserInfo() {
611
612   // ...
613 }
614
615 func updateLike(post: PersonalProjectModel) {
616
617   // ...
618 }
619
620 func updateView() {
621
622   // ...
623 }
624
625 func updateLike(post: PersonalProjectModel) {
626
627   // ...
628 }
629
630 func updateUserInfo() {
631
632   // ...
633 }
634
635 func updateLike(post: PersonalProjectModel) {
636
637   // ...
638 }
639
640 func updateView() {
641
642   // ...
643 }
644
645 func updateLike(post: PersonalProjectModel) {
646
647   // ...
648 }
649
650 func updateUserInfo() {
651
652   // ...
653 }
654
655 func updateLike(post: PersonalProjectModel) {
656
657   // ...
658 }
659
660 func updateView() {
661
662   // ...
663 }
664
665 func updateLike(post: PersonalProjectModel) {
666
667   // ...
668 }
669
670 func updateUserInfo() {
671
672   // ...
673 }
674
675 func updateLike(post: PersonalProjectModel) {
676
677   // ...
678 }
679
680 func updateView() {
681
682   // ...
683 }
684
685 func updateLike(post: PersonalProjectModel) {
686
687   // ...
688 }
689
690 func updateUserInfo() {
691
692   // ...
693 }
694
695 func updateLike(post: PersonalProjectModel) {
696
697   // ...
698 }
699
700 func updateView() {
701
702   // ...
703 }
704
705 func updateLike(post: PersonalProjectModel) {
706
707   // ...
708 }
709
710 func updateUserInfo() {
711
712   // ...
713 }
714
715 func updateLike(post: PersonalProjectModel) {
716
717   // ...
718 }
719
720 func updateView() {
721
722   // ...
723 }
724
725 func updateLike(post: PersonalProjectModel) {
726
727   // ...
728 }
729
730 func updateUserInfo() {
731
732   // ...
733 }
734
735 func updateLike(post: PersonalProjectModel) {
736
737   // ...
738 }
739
740 func updateView() {
741
742   // ...
743 }
744
745 func updateLike(post: PersonalProjectModel) {
746
747   // ...
748 }
749
750 func updateUserInfo() {
751
752   // ...
753 }
754
755 func updateLike(post: PersonalProjectModel) {
756
757   // ...
758 }
759
760 func updateView() {
761
762   // ...
763 }
764
765 func updateLike(post: PersonalProjectModel) {
766
767   // ...
768 }
769
770 func updateUserInfo() {
771
772   // ...
773 }
774
775 func updateLike(post: PersonalProjectModel) {
776
777   // ...
778 }
779
780 func updateView() {
781
782   // ...
783 }
784
785 func updateLike(post: PersonalProjectModel) {
786
787   // ...
788 }
789
790 func updateUserInfo() {
791
792   // ...
793 }
794
795 func updateLike(post: PersonalProjectModel) {
796
797   // ...
798 }
799
800 func updateView() {
801
802   // ...
803 }
804
805 func updateLike(post: PersonalProjectModel) {
806
807   // ...
808 }
809
810 func updateUserInfo() {
811
812   // ...
813 }
814
815 func updateLike(post: PersonalProjectModel) {
816
817   // ...
818 }
819
820 func updateView() {
821
822   // ...
823 }
824
825 func updateLike(post: PersonalProjectModel) {
826
827   // ...
828 }
829
830 func updateUserInfo() {
831
832   // ...
833 }
834
835 func updateLike(post: PersonalProjectModel) {
836
837   // ...
838 }
839
840 func updateView() {
841
842   // ...
843 }
844
845 func updateLike(post: PersonalProjectModel) {
846
847   // ...
848 }
849
850 func updateUserInfo() {
851
852   // ...
853 }
854
855 func updateLike(post: PersonalProjectModel) {
856
857   // ...
858 }
859
860 func updateView() {
861
862   // ...
863 }
864
865 func updateLike(post: PersonalProjectModel) {
866
867   // ...
868 }
869
870 func updateUserInfo() {
871
872   // ...
873 }
874
875 func updateLike(post: PersonalProjectModel) {
876
877   // ...
878 }
879
880 func updateView() {
881
882   // ...
883 }
884
885 func updateLike(post: PersonalProjectModel) {
886
887   // ...
888 }
889
890 func updateUserInfo() {
891
892   // ...
893 }
894
895 func updateLike(post: PersonalProjectModel) {
896
897   // ...
898 }
899
900 func updateView() {
901
902   // ...
903 }
904
905 func updateLike(post: PersonalProjectModel) {
906
907   // ...
908 }
909
910 func updateUserInfo() {
911
912   // ...
913 }
914
915 func updateLike(post: PersonalProjectModel) {
916
917   // ...
918 }
919
920 func updateView() {
921
922   // ...
923 }
924
925 func updateLike(post: PersonalProjectModel) {
926
927   // ...
928 }
929
930 func updateUserInfo() {
931
932   // ...
933 }
934
935 func updateLike(post: PersonalProjectModel) {
936
937   // ...
938 }
939
940 func updateView() {
941
942   // ...
943 }
944
945 func updateLike(post: PersonalProjectModel) {
946
947   // ...
948 }
949
950 func updateUserInfo() {
951
952   // ...
953 }
954
955 func updateLike(post: PersonalProjectModel) {
956
957   // ...
958 }
959
960 func updateView() {
961
962   // ...
963 }
964
965 func updateLike(post: PersonalProjectModel) {
966
967   // ...
968 }
969
970 func updateUserInfo() {
971
972   // ...
973 }
974
975 func updateLike(post: PersonalProjectModel) {
976
977   // ...
978 }
979
980 func updateView() {
981
982   // ...
983 }
984
985 func updateLike(post: PersonalProjectModel) {
986
987   // ...
988 }
989
990 func updateUserInfo() {
991
992   // ...
993 }
994
995 func updateLike(post: PersonalProjectModel) {
996
997   // ...
998 }
999
1000 func updateView() {
1001
1002   // ...
1003 }
1004
1005 func updateLike(post: PersonalProjectModel) {
1006
1007   // ...
1008 }
1009
1010 func updateUserInfo() {
1011
1012   // ...
1013 }
1014
1015 func updateLike(post: PersonalProjectModel) {
1016
1017   // ...
1018 }
1019
1020 func updateView() {
1021
1022   // ...
1023 }
1024
1025 func updateLike(post: PersonalProjectModel) {
1026
1027   // ...
1028 }
1029
1030 func updateUserInfo() {
1031
1032   // ...
1033 }
1034
1035 func updateLike(post: PersonalProjectModel) {
1036
1037   // ...
1038 }
1039
1040 func updateView() {
1041
1042   // ...
1043 }
1044
1045 func updateLike(post: PersonalProjectModel) {
1046
1047   // ...
1048 }
1049
1050 func updateUserInfo() {
1051
1052   // ...
1053 }
1054
1055 func updateLike(post: PersonalProjectModel) {
1056
1057   // ...
1058 }
1059
1060 func updateView() {
1061
1062   // ...
1063 }
1064
1065 func updateLike(post: PersonalProjectModel) {
1066
1067   // ...
1068 }
1069
1070 func updateUserInfo() {
1071
1072   // ...
1073 }
1074
1075 func updateLike(post: PersonalProjectModel) {
1076
1077   // ...
1078 }
1079
1080 func updateView() {
1081
1082   // ...
1083 }
1084
1085 func updateLike(post: PersonalProjectModel) {
1086
1087   // ...
1088 }
1089
1090 func updateUserInfo() {
1091
1092   // ...
1093 }
1094
1095 func updateLike(post: PersonalProjectModel) {
1096
1097   // ...
1098 }
1099
1100 func updateView() {
1101
1102   // ...
1103 }
1104
1105 func updateLike(post: PersonalProjectModel) {
1106
1107   // ...
1108 }
1109
1110 func updateUserInfo() {
1111
1112   // ...
1113 }
1114
1115 func updateLike(post: PersonalProjectModel) {
1116
1117   // ...
1118 }
1119
1120 func updateView() {
1121
1122   // ...
1123 }
1124
1125 func updateLike(post: PersonalProjectModel) {
1126
1127   // ...
1128 }
1129
1130 func updateUserInfo() {
1131
1132   // ...
1133 }
1134
1135 func updateLike(post: PersonalProjectModel) {
1136
1137   // ...
1138 }
1139
1140 func updateView() {
1141
1142   // ...
1143 }
1144
1145 func updateLike(post: PersonalProjectModel) {
1146
1147   // ...
1148 }
1149
1150 func updateUserInfo() {
1151
1152   // ...
1153 }
1154
1155 func updateLike(post: PersonalProjectModel) {
1156
1157   // ...
1158 }
1159
1160 func updateView() {
1161
1162   // ...
1163 }
1164
1165 func updateLike(post: PersonalProjectModel) {
1166
1167   // ...
1168 }
1169
1170 func updateUserInfo() {
1171
1172   // ...
1173 }
1174
1175 func updateLike(post: PersonalProjectModel) {
1176
1177   // ...
1178 }
1179
1180 func updateView() {
1181
1182   // ...
1183 }
1184
1185 func updateLike(post: PersonalProjectModel) {
1186
1187   // ...
1188 }
1189
1190 func updateUserInfo() {
1191
1192   // ...
1193 }
1194
1195 func updateLike(post: PersonalProjectModel) {
1196
1197   // ...
1198 }
1199
1200 func updateView() {
1201
1202   // ...
1203 }
1204
1205 func updateLike(post: PersonalProjectModel) {
1206
1207   // ...
1208 }
1209
1210 func updateUserInfo() {
1211
1212   // ...
1213 }
1214
1215 func updateLike(post: PersonalProjectModel) {
1216
1217   // ...
1218 }
1219
1220 func updateView() {
1221
1222   // ...
1223 }
1224
1225 func updateLike(post: PersonalProjectModel) {
1226
1227   // ...
1228 }
1229
1230 func updateUserInfo() {
1231
1232   // ...
1233 }
1234
1235 func updateLike(post: PersonalProjectModel) {
1236
1237   // ...
1238 }
1239
1240 func updateView() {
1241
1242   // ...
1243 }
1244
1245 func updateLike(post: PersonalProjectModel) {
1246
1247   // ...
1248 }
1249
1250 func updateUserInfo() {
1251
1252   // ...
1253 }
1254
1255 func updateLike(post: PersonalProjectModel) {
1256
1257   // ...
1258 }
1259
1260 func updateView() {
1261
1262   // ...
1263 }
1264
1265 func updateLike(post: PersonalProjectModel) {
1266
1267   // ...
1268 }
1269
1270 func updateUserInfo() {
1271
1272   // ...
1273 }
1274
1275 func updateLike(post: PersonalProjectModel) {
1276
1277   // ...
1278 }
1279
1280 func updateView() {
1281
1282   // ...
1283 }
1284
1285 func updateLike(post: PersonalProjectModel) {
1286
1287   // ...
1288 }
1289
1290 func updateUserInfo() {
1291
1292   // ...
1293 }
1294
1295 func updateLike(post: PersonalProjectModel) {
1296
1297   // ...
1298 }
1299
1300 func updateView() {
1301
1302   // ...
1303 }
1304
1305 func updateLike(post: PersonalProjectModel) {
1306
1307   // ...
1308 }
1309
1310 func updateUserInfo() {
1311
1312   // ...
1313 }
1314
1315 func updateLike(post: PersonalProjectModel) {
1316
1317   // ...
1318 }
1319
1320 func updateView() {
1321
1322   // ...
1323 }
1324
1325 func updateLike(post: PersonalProjectModel) {
1326
1327   // ...
1328 }
1329
1330 func updateUserInfo() {
1331
1332   // ...
1333 }
1334
1335 func updateLike(post: PersonalProjectModel) {
1336
1337   // ...
1338 }
1339
1340 func updateView() {
1341
1342   // ...
1343 }
1344
1345 func updateLike(post: PersonalProjectModel) {
1346
1347   // ...
1348 }
1349
1350 func updateUserInfo() {
1351
1352   // ...
1353 }
1354
1355 func updateLike(post: PersonalProjectModel) {
1356
1357   // ...
1358 }
1359
1360 func updateView() {
1361
1362   // ...
1363 }
1364
1365 func updateLike(post: PersonalProjectModel) {
1366
1367   // ...
1368 }
1369
1370 func updateUserInfo() {
1371
1372   // ...
1373 }
1374
1375 func updateLike(post: PersonalProjectModel) {
1376
1377   // ...
1378 }
1379
1380 func updateView() {
1381
1382   // ...
1383 }
1384
1385 func updateLike(post: PersonalProjectModel) {
1386
1387   // ...
1388 }
1389
1390 func updateUserInfo() {
1391
1392   // ...
1393 }
1394
1395 func updateLike(post: PersonalProjectModel) {
1396
1397   // ...
1398 }
1399
1400 func updateView() {
1401
1402   // ...
1403 }
1404
1405 func updateLike(post: PersonalProjectModel) {
1406
1407   // ...
1408 }
1409
1410 func updateUserInfo() {
1411
1412   // ...
1413 }
1414
1415 func updateLike(post: PersonalProjectModel) {
1416
1417   // ...
1418 }
1419
1420 func updateView() {
1421
1422   // ...
1423 }
1424
1425 func updateLike(post: PersonalProjectModel) {
1426
1427   // ...
1428 }
1429
1430 func updateUserInfo() {
1431
1432   // ...
1433 }
1434
1435 func updateLike(post: PersonalProjectModel) {
1436
1437   // ...
1438 }
1439
1440 func updateView() {
1441
1442   // ...
1443 }
1444
1445 func updateLike(post: PersonalProjectModel) {
1446
1447   // ...
1448 }
1449
1450 func updateUserInfo() {
1451
1452   // ...
1453 }
1454
1455 func updateLike(post: PersonalProjectModel) {
1456
1457   // ...
1458 }
1459
1460 func updateView() {
1461
1462   // ...
1463 }
1464
1465 func updateLike(post: PersonalProjectModel) {
1466
1467   // ...
1468 }
1469
1470 func updateUserInfo() {
1471
1472   // ...
1473 }
1474
1475 func updateLike(post: PersonalProjectModel) {
1476
1477   // ...
1478 }
1479
1480 func updateView() {
1481
1482   // ...
1483 }
1484
1485 func updateLike(post: PersonalProjectModel) {
1486
1487   // ...
1488 }
1489
1490 func updateUserInfo() {
1491
1492   // ...
1493 }
1494
1495 func updateLike(post: PersonalProjectModel) {
1496
1497   // ...
1498 }
1499
1500 func updateView() {
1501
1502   // ...
1503 }
1504
1505 func updateLike(post: PersonalProjectModel) {
1506
1507   // ...
1508 }
1509
1510 func updateUserInfo() {
1511
1512   // ...
1513 }
1514
1515 func updateLike(post: PersonalProjectModel) {
1516
1517   // ...
1518 }
1519
1520 func updateView() {
1521
1522   // ...
1523 }
1524
1525 func updateLike(post: PersonalProjectModel) {
1526
1527   // ...
1528 }
1529
1530 func updateUserInfo() {
1531
1532   // ...
1533 }
1534
1535 func updateLike(post: PersonalProjectModel) {
1536
1537   // ...
1538 }
1539
1540 func updateView() {
1541
1542   // ...
1543 }
1544
1545 func updateLike(post: PersonalProjectModel) {
1546
1547   // ...
1548 }
1549
1550 func updateUserInfo() {
1551
1552   // ...
1553 }
1554
1555 func updateLike(post: PersonalProjectModel) {
1556
1557   // ...
1558 }
1559
1560 func updateView() {
1561
1562   // ...
1563 }
1564
1565 func updateLike(post: PersonalProjectModel) {
1566
1567   // ...
1568 }
1569
1570 func updateUserInfo() {
1571
1572   // ...
1573 }
1574
1575 func updateLike(post: PersonalProjectModel) {
1576
1577   // ...
1578 }
1579
1580 func updateView() {
1581
1582   // ...
1583 }
1584
1585 func updateLike(post: PersonalProjectModel) {
1586
1587   // ...
1588 }
1589
1590 func updateUserInfo() {
1591
1592   // ...
1593 }
1594
1595 func updateLike(post: PersonalProjectModel) {
1596
1597   // ...
1598 }
1599
1600 func updateView() {
1601
1602   // ...
1603 }
1604
1605 func updateLike(post: PersonalProjectModel) {
1606
1607   // ...
1608 }
1609
1610 func updateUserInfo() {
1611
1612   // ...
1613 }
1614
1615 func updateLike(post: PersonalProjectModel) {
1616
1617   // ...
1618 }
1619
1620 func updateView() {
1621
1622   // ...
1623 }
1624
1625 func updateLike(post: PersonalProjectModel) {
1626
1627   // ...
1628 }
1629
1630 func updateUserInfo() {
1631
1632   // ...
1633 }
1634
1635 func updateLike(post: PersonalProjectModel) {
1636
1637   // ...
1638 }
1639
1640 func updateView() {
1641
1642   // ...
1643 }
1644
1645 func updateLike(post: PersonalProjectModel) {
1646
1647   // ...
1648 }
1649
1650 func updateUserInfo() {
1651
1652   // ...
1653 }
1654
1655 func updateLike(post: PersonalProjectModel) {
1656
1657   // ...
1658 }
1659
1660 func updateView() {
1661
1662   // ...
1663 }
1664
1665 func updateLike(post: PersonalProjectModel) {
1666
1667   // ...
1668 }
1669
1670 func updateUserInfo() {
1671
1672   // ...
1673 }
1674
1675 func updateLike(post: PersonalProjectModel) {
1676
1677   // ...
1678 }
1679
1680 func updateView() {
1681
1682   // ...
1683 }
1684
1685 func updateLike(post: PersonalProjectModel) {
1686
1687   // ...
1688 }
1689
1690 func updateUserInfo() {
1691
1692   // ...
1693 }
1694
1695 func updateLike(post: PersonalProjectModel) {
1696
1697   // ...
1698 }
1699
1700 func updateView() {
1701
1702   // ...
1703 }
1704
1705 func updateLike(post: PersonalProjectModel) {
1706
1707   // ...
1708 }
1709
1710 func updateUserInfo() {
1711
1712   // ...
1713 }
1714
1715 func updateLike(post: PersonalProjectModel) {
1716
1717   // ...
1718 }
1719
1720 func updateView() {
1721
1722   // ...
1723 }
1724
1725 func updateLike(post: PersonalProjectModel) {
1726
1727   // ...
1728 }
1729
1730 func updateUserInfo() {
1731
1732   // ...
1733 }
1734
1735 func updateLike(post: PersonalProjectModel) {
1736
1737   // ...
1738 }
1739
1740 func updateView() {
1741
1742   // ...
1743 }
1744
1745 func updateLike(post: PersonalProjectModel) {
1746
1747   // ...
1748 }
1749
1750 func updateUserInfo() {
1751
1752   // ...
1753 }
1754
1755 func updateLike(post: PersonalProjectModel) {
1756
1757   // ...
1758 }
1759
1760 func updateView() {
1761
1762   // ...
1763 }
1764
1765 func updateLike(post: PersonalProjectModel) {
1766
1767  
```

Envit master Envit iPhone 13 Pro Max Finished running Envit on iPhone 13 Pro Max ▲ 28 Wed 1 Jun 8:29 PM

```

J-comp Project Apis
  JCompon...jectPostApi
  JComProjCommentApi
  JComProj...mentApi
  MyJCompProjApi
  HelperServiceJC

J-Component
  JCompone...wController
  FindJComp...Controller
  JComponentFeed
  JCompProjPostCell
  JCompProjPostCell
  Comments...eViewCell
  JCompone...rojectModel
  JCompProj...Comments
  ProjectRequestFeed

Pers Project Apis
  PersonalProjectPostApi
  PersProjCommentApi
  PersProj...commentApi
  MyPersProjApi
  HelperServicePP

Personal Project
  PersonalPr...wController
  FindNewPr...wController
  PersProjPostCell
  PersProjPostCell
  Comments...sProjFeed
  Comments...eViewCell
  PersonalProjectModel
  PersProjectComments

Project Share

```

```

76     BorderProp()
77     CornerRadius()
78     LeftPadding()
79     DropDownOptions()
80     TextViewProperties()
81   }
82
83   // Submit button action
84   @IBAction func addProjectTapped(_ sender: UIButton) {
85     sender.flash()
86
87     if (projectTitle.text!.isEmpty || roleOption.text!.isEmpty || projDescTextView.text!.isEmpty) {
88       // Alert for empty fields
89       let myAlert = UIAlertController(title: "Empty Fields", message: "", preferredStyle: UIAlertController.Style.alert)
90       let okAction = UIAlertAction(title: "Ok", style: UIAlertAction.Style.default, handler: nil)
91       myAlert.addAction(okAction)
92       self.present(myAlert, animated: true, completion: nil)
93     }
94
95     HelperServicePP.uploadDataToServer(titleText: projectTitle.text!, roleText: roleOption.text!, descText:
96       projDescTextView.text!.trimmingCharacters(in: .whitespacesAndNewlines), onSuccess: {
97       // Alert pod - Project Added
98       let alertView = SPAalertView(title: "Your project has been added", message: nil, preset: SPAAlertPreset.done)
99       alertView.duration = 1.2
100      alertView.present()
101
102      // Clear textfields after success
103      self.projectTitle.text = ""
104      self.roleOption.text = ""
105      self.projDescTextView.text = ""
106
107      // And to enable back for a new input in textfield
108      self.projectTitle.isEnabled = true
109      self.roleOption.isEnabled = true
110      self.projDescTextView.isEnabled = true
111    })
112  }
113
114 }

```

Line: 117 Col: 12

Envit master Envit iPhone 13 Pro Max Finished running Envit on iPhone 13 Pro Max ▲ 28 Wed 1 Jun 8:50 PM

```

J-comp Project Apis
  JCompon...jectPostApi
  JComProjCommentApi
  JComProj...mentApi
  MyJCompProjApi
  HelperServiceJC

J-Component
  JCompone...wController
  FindJComp...Controller
  JComponentFeed
  JCompProjPostCell
  JCompProjPostCell
  Comments...eViewCell
  JCompone...rojectModel
  JCompProj...Comments
  ProjectRequestFeed

Pers Project Apis
  PersonalProjectPostApi
  PersProjCommentApi
  PersProj...commentApi
  MyPersProjApi
  HelperServicePP

Personal Project
  PersonalPr...wController
  FindNewPr...wController
  PersProjPostCell
  PersProjPostCell
  Comments...sProjFeed
  Comments...eViewCell
  PersonalProjectModel
  PersProjectComments

Project Share

```

```

37   let jCompProjCell = jCompProjectsFeedCollectionView.dequeueReusableCell(withIdentifier: "JCompProjPostCell", for: indexPath) as!
38   JCompProjPostCell
39   let post = jCompProjectsPosts[indexPath.row]
40   let user = users[indexPath.row]
41   jCompProjCell.jCompProjPost = post
42   jCompProjCell.user = user
43   // linking home VC & home table view cell
44   jCompProjCell.jCompProjFeedVC = self
45   return jCompProjCell
46 }

47 // reference to store JComponentProjectModel class info
48 var jCompProjectsPosts = [JComponentProjectModel]()
49
50 // reference to store User class info
51 var users = [AppUser]()
52
53 // DB ref
54 // --- var refJCompProjects: DatabaseReference!
55
56 // load jcomp project posts
57 func loadPosts() {
58
59   // start when loadPosts func starts
60   activityIndicatorView8.startAnimating()
61
62   Api.jComponentProjectPost.observePosts { (post) in
63     guard let postId = post.uid else {
64       return
65     }
66     // fetch user data in mentor posts
67     self.fetchUser(uid: postId, completed: {
68       self.jCompProjectsPosts.append(post)
69       // print(self.posts)
70       // stop before tableview reloads data
71       self.activityIndicatorView8.stopAnimating()
72       self.activityIndicatorView8.hidesWhenStopped = true
73       self.jCompProjectsFeedCollectionView.reloadData()
74     })
75   }
76 }

```

Line: 120 Col: 1

Xcode Screenshot showing the code for `HelperServiceJC` class:

```

Envit master Envit iPhone 13 Pro Max Finished running Envit on iPhone 13 Pro Max ▲ 28
Envit > J-comp Project Apis HelperServiceJC No Selection
Envit > J-comp Project Apis HelperServiceJC
11  class HelperServiceJC {
12
13     static func uploadDataToServer(titleText: String, courseText: String, descText: String, onSuccess: @escaping () -> Void) {
14
15         self.sendDataToDatabase(titleText: titleText, courseText: courseText, descText: descText, onSuccess: onSuccess)
16
17     }
18
19     static func sendDataToDatabase(titleText: String, courseText: String, descText: String, onSuccess: @escaping () -> Void) {
20
21         let newPostId = Api.JComponentProjectPost.REF_POSTS.childByAutoId().key
22         let newPostReference = Api.JComponentProjectPost.REF_POSTS.child(newPostId!)
23         guard let currentUser = Api.UserRef.CURRENT_USER else {
24             return
25         }
26         let currentUserID = currentUser.uid
27         // Creating a timestamp
28         let timestamp = NSNumber(value: Int(NSTimeIntervalSince1970))
29         // put that download url string in db
30
31         newPostReference.setValue(["1 Title": titleText, "2 Course": courseText, "3 Description": descText, "4 Timestamp": timestamp, "5 uid": currentUserID], withCompletionBlock: { (error, ref) in
32             if error != nil {
33                 print(error!.localizedDescription)
34                 return
35             }
36             // reference for my posts
37             let myPostRef = Api.MyJcompProjPosts.REF_MYPOSTS.child(currentUserID).child(newPostId!)
38             myPostRef.setValue(true, withCompletionBlock: {
39                 (error, ref) in
40                     if error != nil {
41                         print(error!.localizedDescription)
42                         return
43                     }
44                 })
45             onSuccess()
46         }
47     }
48 }
49 } // #50
50

```

Xcode Screenshot showing the code for `JComponentProjectModel` extension:

```

Envit master Envit iPhone 13 Pro Max Finished running Envit on iPhone 13 Pro Max ▲ 28
Envit > J-comp Project Apis JComponentProjectModel No Selection
Envit > J-comp Project Apis JComponentProjectModel
21     /// Remodel Post class, bcz it currently doesn't have a post id property
22     var id: String?
23
24     /**
25      var likeCount: Int?
26      var likes: Dictionary<String, Any>?
27
28      var isLiked: Bool?
29
30  }
31
32 extension JComponentProjectModel {
33
34     static func transformJCompProjPost(dict: [String: Any], key: String) -> JComponentProjectModel {
35
36         let jcompProjPost = JComponentProjectModel()
37         /// Remodel Post class, bcz it currently doesn't have a post id property
38         jcompProjPost.id = key
39         jcompProjPost.projectTitle = dict["1 Title"] as? String
40         jcompProjPost.courseTitle = dict["2 Course"] as? String
41         jcompProjPost.projDesc = dict["3 Description"] as? String
42         jcompProjPost.uid = dict["5 uid"] as? String
43
44         jcompProjPost.likeCount = dict["likeCount"] as? Int
45         jcompProjPost.likes = dict["likes"] as? Dictionary<String, Any>
46
47         if let currentUserID = Auth.auth().currentUser?.uid {
48             if jcompProjPost.likes != nil {
49                 /* if post.likes[currentUserID] != nil {
50                     post.isLiked = true
51                 } else {
52                     post.isLiked = false
53                 }*/
54                 // Above commented snippet can be put in 1 line.. as below.
55                 jcompProjPost.isLiked = jcompProjPost.likes![currentUserID] != nil
56             }
57         }
58
59         return jcompProjPost
60     }
61 }

```

Xcode Screenshot (Wednesday, June 1, 8:51 PM)

```

Envit master Envit iPhone 13 Pro Max Finished running Envit on iPhone 13 Pro Max ▲ 28
J-comp Project Apis
  JCompon...jectPostApi
  JComProjCommentApi
  JComProj...ommentApi
  MyJComProjApi
  HelperServiceJC
  J-Component
    JCompone...wController
    FindJComp...Controller
    JComProjFeed
    JComProjPostCell
    JComProjPostCell M
    JComProj...pProjFeed
    Comments...leViewCell
    JCompone...rojectModel
    JComProj...Comments
    ProjectRequestFeed
  Pers Project Apis
    PersonalProjectPostApi
    PersProjCommentApi
    PersProj...ommentApi
    MyPersProjApi
    HelperServicePP
  Personal Project
    PersonalPr...wController
    FindNew...Controller M
    PersProjPostCell
    PersProjPostCell M
    Comments...rsProjFeed
    Comments...leViewCell
    PersonalProjectModel
    PersProjectComments
  Project Share

```

```

14  class JComponentProjectPostApi {
15
16    var REF_POSTS = Database.database().reference().child("J-Component Projects").child("Details")
17
18    func observePosts(completion: @escaping (JComponentProjectModel) -> Void) {
19
20      REF_POSTS.observe(.childAdded, with: { [snapshot] in
21
22        if let dict = snapshot.value as? [String: Any] {
23          let newPost = JComponentProjectModel.transformJComProjPost(dict: dict, key: snapshot.key)
24          completion(newPost)
25        }
26      })
27    }
28
29  }
30
31  func observePost(withId id: String, completion: @escaping (JComponentProjectModel) -> Void) {
32
33    REF_POSTS.child(id).observeSingleEvent(of: .value, with: {
34      snapshot in
35
36        if let dict = snapshot.value as? [String: Any] {
37          let post = JComponentProjectModel.transformJComProjPost(dict: dict, key: snapshot.key)
38          completion(post)
39        }
40      })
41    }
42
43  }
44
45  func observeLikeCount(withPostId id: String, completion: @escaping (Int) -> Void) {
46
47    REF_POSTS.child(id).observe(.childChanged, with: {
48      snapshot in
49      print(snapshot)
50      if let value = snapshot.value as? Int {
51        completion(value)
52      }
53    })
54  }

```

Line: 1 Col: 1

Xcode Screenshot (Wednesday, June 1, 8:50 PM)

```

Envit master Envit iPhone 13 Pro Max Finished running Envit on iPhone 13 Pro Max ▲ 28
J-comp Project Apis
  JCompon...jectPostApi
  JComProjCommentApi
  JComProj...ommentApi
  MyJComProjApi
  HelperServiceJC
  J-Component
    JCompone...wController
    FindJComp...Controller
    JComProjFeed
    JComProjPostCell
    JComProjPostCell M
    Comments...pProjFeed
    Comments...leViewCell
    JCompone...rojectModel
    JComProj...Comments
    ProjectRequestFeed
  Pers Project Apis
    PersonalProjectPostApi
    PersProjCommentApi
    PersProj...ommentApi
    MyPersProjApi
    HelperServicePP
  Personal Project
    PersonalPr...wController
    FindNew...Controller M
    PersProjPostCell
    PersProjPostCell M
    Comments...rsProjFeed
    Comments...leViewCell
    PersonalProjectModel
    PersProjectComments
  Project Share

```

```

48  func updateView() {
49
50    courseName.text = jComProjPost?.courseTitle
51    projectName.text = jComProjPost?.projectTitle
52    descriptionLabel.text = jComProjPost?.projDesc
53    print(courseName.text)
54    print(projectName.text)
55    print(descriptionLabel.text)
56
57    setupUserInfo()
58    /// Update like
59    updateLike(post: jComProjPost!)
60    /// New
61    self.updateLike(post: self.jComProjPost!)
62
63  }
64
65  func updateLike(post: JComponentProjectModel) {
66
67    /// we first checked if its true, and no one liked this post before.
68    /// or if probably someone did, but the current user did not.
69    /// then we display, non-selected like icon.
70    /// otherwise, display likeSelected icon.
71    let imageName = post.likes == nil || !post.isLiked! ? "Icon1" : "likeSelected"
72    projectLikeImageView.image = UIImage(named: imageName)
73    // Below commented snippet can be put in line. as above.
74    /* if post.isLiked == false {
75      likeImageView.image = UIImage(named: "like")
76    } else {
77      likeImageView.image = UIImage(named: "likeSelected")
78    }*/
79
80    // We now update like count
81    /// Use optional chaining with guard
82    guard let count = post.likeCount else {
83      return
84    }
85    if count != 0 {
86      likeCountButton.setTitle("\(count) likes", for: .normal)
87    } else {
88      likeCountButton.setTitle("@ likes", for: .normal)
89    }
90
91  }

```

Line: 93 Col: 33

Xcode Screenshot (iPhone 13 Pro Max) showing code for `ProjectRequestFeed` in Envit master:

```

28     @IBAction func addCourseAction(_ sender: UIButton) {
29
30         /// Alert
31         let alertController = UIAlertController(title: "Enter course title", message: "", preferredStyle: .alert)
32         /// Add textfield
33         alertController.addTextField(configurationHandler: courseFunc(textField:))
34
35         /// Ok action
36         let okAction = UIAlertAction(title: "Add course", style: .default) { (_) in
37     }
38
39         /// Cancel action
40         let cancelAction = UIAlertAction(title: "Cancel", style: .destructive) { (_) in
41     }
42
43
44         /// AlertView font
45         let titleFont = NSAttributedString.Key.font = UIFont(name: "SFProRounded-Medium", size: 18.0)!)
46         let titleAttrString = NSMutableAttributedString(string: "Enter course title", attributes: titleFont)
47         alertController.setValue(titleAttrString, forKey: "attributedTitle")
48
49         /// Adding Ok action
50         alertController.addAction(okAction)
51         /// Adding Cancel action
52         alertController.addAction(cancelAction)
53         /// Present alert controller
54         present(alertController, animated: true, completion: nil)
55
56     }
57
58     // Textfield prop
59     func courseFunc(textField: UITextField) {
60         let heightConstraint = NSLayoutConstraint(item: textField!, attribute: .height, relatedBy: .equal, toItem: nil, attribute: .notAnAttribute,
61             multiplier: 1, constant: 32)
62         textField.addConstraint(heightConstraint)
63         textField.minimumFontSize = 18
64         textField.font = UIFont(name: "SFProRounded-Regular", size: 16.0)
65     }
66 } // #68

```

Xcode Screenshot (iPhone 13 Pro Max) showing code for `JComponentViewController` in Envit master:

```

67     BorderProp()
68     CornerRadius()
69     LeftPadding()
70     TextViewProperties()
71
72 }
73
74 // Submit button action
75 @IBAction func addProjectTapped(_ sender: UIButton) {
76
77     sender.flash()
78
79     if (projTitle.text!.isEmpty || courseTitle.text!.isEmpty || projDescTextView.text!.isEmpty) {
80         // Alert for empty fields
81         let myAlert = UIAlertController(title: "Empty Fields", message: "", preferredStyle: UIAlertController.Style.alert)
82         let okAction = UIAlertAction(title: "Ok", style: UIAlertAction.Style.default, handler: nil)
83         myAlert.addAction(okAction)
84         self.present(myAlert, animated: true, completion: nil)
85     }
86
87
88     HelperServiceJC.uploadDataToServer(titleText: projTitle.text!, courseText: courseTitle.text!, descText:
89         projDescTextView.text!.trimmingCharacters(in: .whitespacesAndNewlines), onSuccess: {
90         // Alert pod - Project Added
91         let alertView = SPAalertView(title: "Your project has been added", message: nil, preset: SPAAlertPreset.done)
92         alertView.duration = 1.2
93         alertView.present()
94
95         // Clear textfields after success
96         self.projTitle.text = ""
97         self.courseTitle.text = ""
98         self.projDescTextView.text = ""
99
100        // And to enable back for a new input in textfield
101        self.projTitle.isEnabled = true
102        self.courseTitle.isEnabled = true
103        self.projDescTextView.isEnabled = true
104    })
105

```

The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure under "Envit". Key files include AppDelegate.swift, HelperService.swift, and various ViewControllers like LoginViewController, SignUpViewController, etc.
- Assistant Editor:** Displays the code for **HelperService.swift**. The code handles file upload to Firebase Storage and database, and sends data to the database.
- Top Bar:** Shows the current target as "Envit" for "iPhone 13 Pro Max". Status bar indicates "Finished running Envit on iPhone 13 Pro Max" and battery level at 100%.
- Bottom Bar:** Shows the Dock with various application icons.

```
12 class HelperService {
13
14     static func uploadDataToServer(data: Data, caption: String, frame: String, category: String, onSuccess: @escaping () -> Void) {
15         // NSUUID()
16         let photoIdString = NSUUID().uuidString
17         print("Photo Id String: \(photoIdString)")
18         let storageRef = Storage.storage().reference(forURL: "gs://vit-share.appspot.com").child("Posts").child(photoIdString)
19         let metadata = StorageMetadata()
20         metadata.contentType = "image/jpg"
21         // put image data
22         storageRef.putData(data, metadata: nil) { (metadata, error) in
23             if error != nil {
24                 print(error!.localizedDescription)
25                 return
26             }
27             // get download url for image from Firebase Storage
28             storageRef.downloadURL { (url, error) in
29                 // convert that download url to string
30                 if let metaImageUrl = url?.absoluteString {
31                     print(metaImageUrl)
32                     self.sendDataToDatabase(photoUrl: metaImageUrl, caption: caption, frame: frame, category: category, onSuccess: onSuccess)
33                 }
34             }
35         }
36     }
37
38     static func sendDataToDatabase(photoUrl: String, caption: String, frame: String, category: String, onSuccess: @escaping () -> Void) {
39         let newPostId = Api.Post.REF_POSTS.childByAutoId().key
40         let newPostReference = Api.Post.REF_POSTS.child(newPostId)
41         guard let currentUser = Api.UserDb.CURRENT_USER else {
42             return
43         }
44         let currentUserID = currentUser.uid
45         // put that download url string in db
46         newPostReference.setValue(["uid": currentUserID, "photoUrl": photoUrl, "caption": caption, "frameColor": frame, "categoryLabel": category], withCompletionBlock: { (error, ref) in
47             if error != nil {
48                 print(error!.localizedDescription)
49             }
50         })
51     }
52 }
```

```
12 class Post {
13
14     var caption: String?
15     var photoUrl: String?
16     var frameColor: String?
17     var categoryLabel: String?
18
19     var uid: String?
20
21     /// Remodel Post class, bcuz it currently doesn't have a post id property
22     var id: String?
23
24     var likeCount: Int?
25     var likes: Dictionary<String, Any>?
26
27     var isLiked: Bool?
28
29 }
30
31 extension Post {
32
33     // Photo
34     static func transformPostPhoto(dict: [String: Any], key: String) -> Post {
35
36         let post = Post()
37         /// Remodel Post class, bcuz it currently doesn't have a post id property
38         post.id = key
39         post.frameColor = dict["frameColor"] as? String
40         post.categoryLabel = dict["categoryLabel"] as? String
41         post.caption = dict["caption"] as? String
42         post.photoUrl = dict["photoUrl"] as? String
43         post.uid = dict["uid"] as? String
44
45         post.likeCount = dict["likeCount"] as? Int
46         post.likes = dict["likes"] as? Dictionary<String, Any>
47
48         if let currentUserID = Auth.auth().currentUser?.uid {
49             if post.likes != nil {
50                 /* if post.likes[currentUserID] != nil {
```

The screenshot shows the Xcode interface with the project 'Envit' open. The left sidebar displays the file structure, and the main editor area shows the implementation of the `PostApi` class. The code handles observing posts and post likes.

```
14  class PostApi {
15
16    var REF_POSTS = Database.database().reference().child("Wall-Posts").child("Details")
17
18    func observePosts(completion: @escaping (Post) -> Void) {
19
20      REF_POSTS.observe(.childAdded, with: { [snapshot] in
21
22        if let dict = snapshot.value as? [String: Any] {
23          let newPost = Post.transformPostPhoto(dict: dict, key: snapshot.key)
24          completion(newPost)
25        }
26      })
27    }
28
29  }
30
31  func observePost(withId id: String, completion: @escaping (Post) -> Void) {
32
33    REF_POSTS.child(id).observeSingleEvent(of: .value, with: {
34      snapshot in
35
36      if let dict = snapshot.value as? [String: Any] {
37        let post = Post.transformPostPhoto(dict: dict, key: snapshot.key)
38        completion(post)
39      }
40    })
41  }
42
43  }
44
45  func observeLikeCount(withPostId id: String, completion: @escaping (Int) -> Void) {
46
47    REF_POSTS.child(id).observe(.childChanged, with: {
48      snapshot in
49      print(snapshot)
50      if let value = snapshot.value as? Int {
51        completion(value)
52      }
53    })
54  }
55
56
57  func incrementLikes(postId: String, onSuccess: @escaping (Post) -> Void, onError: @escaping (_ errorMessage: String?) -> Void ) {
58
59    let postRef = Api.Post.REF_POSTS.child(postId)
60    postRef.runTransactionBlock ({ (currentData: MutableData) -> TransactionResult in
61
62      if var post = currentData.value as? [String: AnyObject], let uid = Api.UserDet.CURRENT_USER?.uid {
63
64        // print("Value 1: \(currentData.value)")
65        var likes: Dictionary<String, Bool>
66        likes = post["likes"] as? [String: Bool] ?? [:]
67        var likeCount = post["likeCount"] as? Int ?? 0
68
69        if let _ = likes[uid] {
70          likeCount -= 1
71          likes.removeValue(forKey: uid)
72        } else {
73          likeCount += 1
74          likes[uid] = true
75        }
76
77        post["likeCount"] = likeCount as AnyObject
78        post["likes"] = likes as AnyObject
79
80        currentData.value = post
81
82        return TransactionResult.success(withValue: currentData)
83      }
84
85      return TransactionResult.success(withValue: currentData)
86    }) { (error, committed, snapshot) in
87
88      if let error = error {
89        onError(error.localizedDescription)
90      }
91
92    }
93  }
94
95
96  // print("Value 2: \(snapshot?.value)")
97  if let dict = snapshot?.value as? [String: Any] {
98    let post = Post.transformPostPhoto(dict: dict, key: snapshot!.key)
99    onSuccess(post)
100  }
101
102 }
```

The screenshot shows the Xcode interface with the project 'Envit' open. The left sidebar displays the file structure, and the main editor area shows the continuation of the `PostApi` class implementation, specifically the `incrementLikes` function.

```
55
56
57  func incrementLikes(postId: String, onSuccess: @escaping (Post) -> Void, onError: @escaping (_ errorMessage: String?) -> Void ) {
58
59    let postRef = Api.Post.REF_POSTS.child(postId)
60    postRef.runTransactionBlock ({ (currentData: MutableData) -> TransactionResult in
61
62      if var post = currentData.value as? [String: AnyObject], let uid = Api.UserDet.CURRENT_USER?.uid {
63
64        // print("Value 1: \(currentData.value)")
65        var likes: Dictionary<String, Bool>
66        likes = post["likes"] as? [String: Bool] ?? [:]
67        var likeCount = post["likeCount"] as? Int ?? 0
68
69        if let _ = likes[uid] {
70          likeCount -= 1
71          likes.removeValue(forKey: uid)
72        } else {
73          likeCount += 1
74          likes[uid] = true
75        }
76
77        post["likeCount"] = likeCount as AnyObject
78        post["likes"] = likes as AnyObject
79
80        currentData.value = post
81
82        return TransactionResult.success(withValue: currentData)
83      }
84
85      return TransactionResult.success(withValue: currentData)
86    }) { (error, committed, snapshot) in
87
88      if let error = error {
89        onError(error.localizedDescription)
90      }
91
92    }
93  }
94
95
96  // print("Value 2: \(snapshot?.value)")
97  if let dict = snapshot?.value as? [String: Any] {
98    let post = Post.transformPostPhoto(dict: dict, key: snapshot!.key)
99    onSuccess(post)
100  }
101
102 }
```

The screenshot shows the Xcode interface with the following details:

- Project Navigator:** Shows the project structure with files like Main.storyboard, Assets.xcassets, Extensions, iOSDropDown, TextFieldCounter, Info.plist, GoogleService-Info.plist, AuthService, Api, AppUser, UserApi, Discover, Discover..Controller, Share on Wall Apis, PostApi, CommentApi, PostCommentApi, MyPostApi, HelperService, Share on the Wall, Post, Comment, ShareOn..Controller, J-comp Project Apis, JCompone..jectPostApi, JCompProjCommentApi, MyJCompProjCommentApi, JCompProjPostApi, HelperServiceJC, J-Component, and JCompone..wController.
- Editor:** Displays the Swift code for `ShareOnWallViewController`. The code handles sharing functionality, including image selection, upload to a server, and UI updates. It includes guard statements for selected images and network requests, and a clean() function to reset the state.
- Toolbar:** Shows various Xcode tools and icons.
- Status Bar:** Displays the date (Wed 1 Jun 12:06 PM), battery level (100%), signal strength, and a message indicating the app has finished running on an iPhone 13 Pro Max.

Fig 14: Xcode Swift Codes Screenshots

5.1 Schedule:

1.	EnVIT Project prototype design
2.	Authentication - Frontend + Backend
2.	Mentor feature implementation - Frontend + Backend
3.	Mentee feature implementation - Frontend + Backend
4.	Personal Projects feature implementation - Frontend + Backend
5.	J-component Projects feature implementation - Frontend + Backend
6.	Share on the Wall feature implementation - Frontend + Backend
7.	Profile feature implementation - Frontend + Backend
8.	Overall Frontend + Backend + Design + Testing

Table 1: List of Project Schedules

5.2 Tasks:

Task No.	Task Section Name
1	Design
2	Tech Share
3	Project Share
4	Share on the Wall
5	Profile + Authentication
6	Database + Backend

Table 2: List of Tasks Sections

5.3 Milestones:

December 2021	EnVIT Project prototype design
January 2022	Authentication - Frontend + Backend Mentor feature implementation - Frontend + Backend
February 2022	Mentee feature implementation - Frontend + Backend Personal Projects feature implementation - Frontend + Backend
March 2022	J-component Projects feature implementation - Frontend + Backend Profile feature implementation - Frontend + Backend
April 2022	Share on the Wall feature implementation - Frontend + Backend
May 2022	Overall Frontend + Backend + Design + Testing

Table 3: List of Project Milestones

7. Cost Analysis / Result / Discussion:

1. App functionality and purpose – what this app will be capable of doing for its users.
2. Integration points – this app will be integrated with third-party apps like Firebase that will be the source of its content.
3. Use of visual objects – complexity of visual objects inside of this app will significantly influence the cost.
4. Maintenance plan – once this project is over, certainly it will require technical support from its developer to provide updates that patch bugs or introduce new features.
5. Free of cost app.
6. Ability to connect all VITians irrespective of year, branch, and course.

The mobile application that will be created is easy and friendly to use. It can be used by all the VITians irrespective of the year and branch and stay connected at all times. Furthermore, due to the unique modularised structure of the software, it will become easy to make changes to the thresholds and the classified steps independently and with ease which will make the application flexible.

8. Summary:

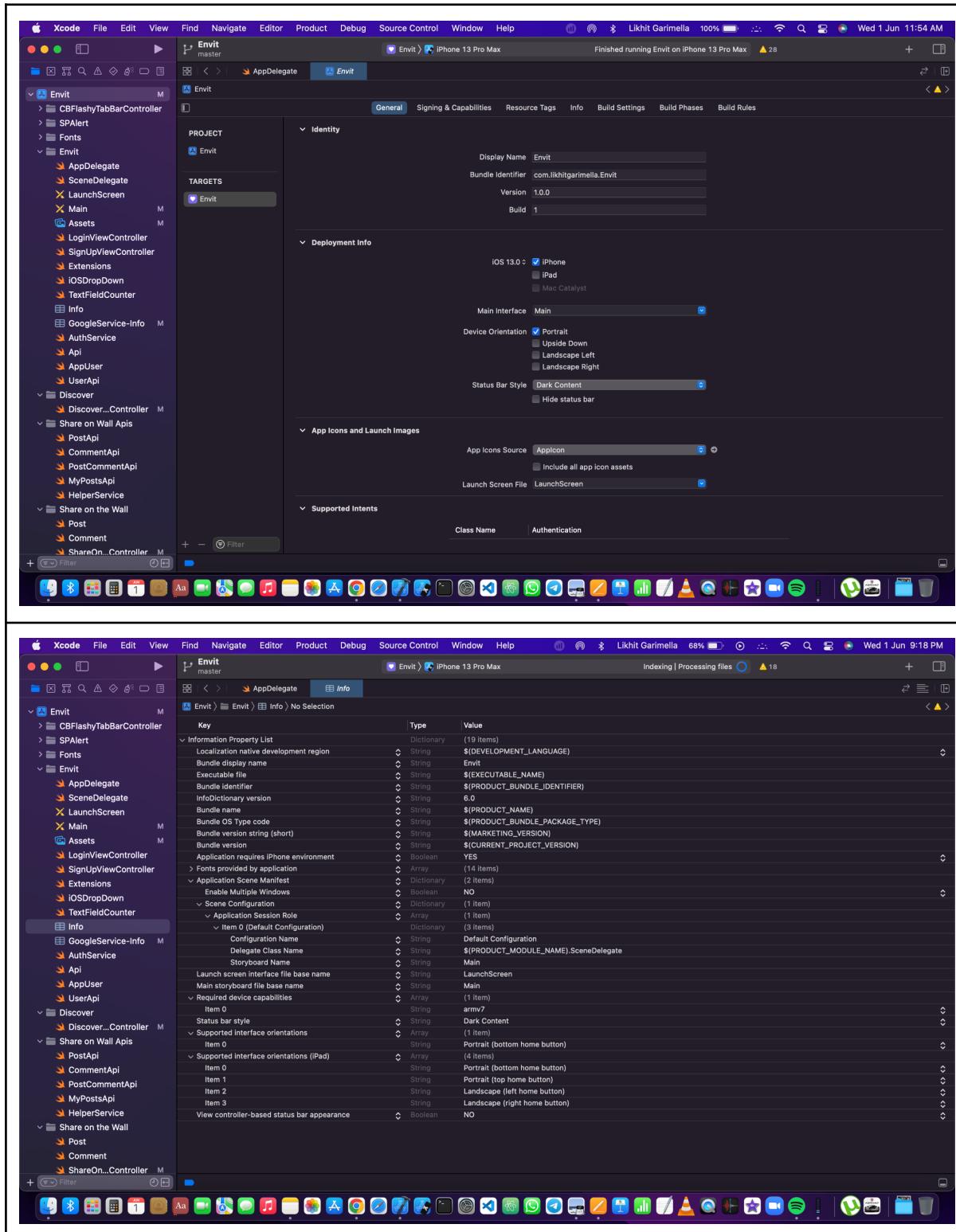
The most common act for interaction among VIT students who belong to one class is through Students, especially new joinees from 2020, can only contact their peers there must be an interface where students, irrespective of the year, irrespective of the branch, interface, students can make use of each other's thoughts, help each other, get guidance in higher studies or etc., and have a good social connection among their peers and make friends. There is a certain paradox that we must turn to in the field of technology and applications to made a philosophy over the years, the key to getting the best out of technology is to use these better than displaying this , there is a growing range of applications for mobile applications that can help students in a variety of ways to help you learn a new language or look up information about the topic you are studying, as well as practical learning apps that can help you track all your class notes and manage your reading time.

9. References:

- [1] Brian Thoms , “A Dynamic Social Feedback System to Support Learning and Social Interaction in Higher Education” , IEEE Transactions on Learning Technologies, Volume: 4, Issue: 4, Oct.-Dec. 2011.
- [2] Amy L. Soller , “Supporting Social Interaction in an Intelligent Collaborative Learning System” , International Journal of Artificial Intelligence in Education , hal version-1, 14 Dec 2007.
- [3] Robert W. Root , “Design of a multi-media vehicle for social browsing” , CSCW '88: Proceedings of the 1988 ACM conference on Computer-supported cooperative work , 01 January 1988.
- [4] J. Lynn McBrien , Rui Cheng , Phyllis Jones, “Virtual Spaces: Employing a Synchronous Online Classroom to Facilitate Student Engagement in Online Learning” , International Review of Research in Open and Distributed Learning , Volume 10, Number 3, June 2009.
- [5] Eman Javed, Naeem Ahmed Mahoto, Vijdan Khalique ,Mohsin Ali, “Exploring Role of Social Media Usage Towards Academic Activities of Students” , International Multi-Topic ICT Conference (IMTIC) , 25-27 April 2018.
- [6] Kirsi Silius, Thumas Miilumäki, Jukka Huhtamäki, Teemo Tebest, Joonas Meriläinen , “Social media enhanced studying and learning in higher education” , IEEE EDUCON 2010 Conference , 14-16 April 2010.
- [7] Marta Sánchez-Saus Laserna ,Mario Crespo Miguel , “Social media as a teaching innovation tool for the promotion of interest and motivation in higher education” , International Symposium on Computers in Education (SIIE) , 19-21 Sept. 2018.
- [8] Magda Chelly , Hana Mataillet , “Social Media and the impact on education: Social media and home education” ,2012 International Conference on E-Learning and E-Technologies in Education (ICEEE) , 24-26 Sept. 2012.
- [9] Asaf Varol , Naveed Ahmed , “Social networks' role in online education” , International Conference on Application of Information and Communication Technologies (AICT) , 23-25 Oct. 2013.
- [10] Antonio Chella,Rosario Sorbello,Giovanni Pilato, Giorgio Vassallo,Marcello Giardina , “A New Humanoid Architecture for Social Interaction between Human and a Robot Expressing Human-Like Emotions Using an Android Mobile Device as Interface”, Advances in Intelligent Systems and Computing book series (AISC, volume 196).

- [11] Agnès Blaye, Paul Light & Vitaly Rubtsov , “Collaborative learning at the computer; How social processes ‘interface’ with human-computer interaction” , European Journal of Psychology of Education volume 7, 257–267 (1992).
- [12] Venkata N. Inukollu, Divya D. Keshamoni, Taeghyun Kang, Manikanta Inukollu, “Factors Influencing Quality of Mobile Apps:Role of Mobile App Development Life Cycle”, International journal of Software Engineering & Applications (IJSEA), 2014.
- [13] José Miguel Mota,Iván Ruiz-Rube,Juan Manuel Dodero, “Augmented reality mobile app development for all”, Department of Computer Engineering, University of Cádiz, Puerto Real (Cádiz), Spain.
- [14] Rita Francese, Carmine Gravino,Michele Risi, Giuseppe Scanniello, Genoveffa Tortora, “Mobile App Development and Management: Results from a Qualitative Investigation” , Buenos Aires, Argentina,22-23 May 2017.
- [15] Ronald Jabangwe,Henry Edison, Anh NguyenDuc , “Software engineering process models for mobile app development: A systematic literature review” , University of Southern Denmark, The Maersk Mc-Kinney Moller Institute, SDU Software Engineering, Campusvej 55, Odense M DK-5230, Denmark.
- [16] Melvyn Zhang , Enquan Cheow , Cyrus SH Ho , Beng Yeong Ng , “Application of Low-Cost Methodologies for Mobile Phone App Development” , Published on 9.12.2014 in Vol 2, No 4 (2014): Oct-Dec.
- [17] Michiel Willocx, Jan Vossaert, Vincent Naessens , “ A Quantitative Assessment of Performance in Mobile App Development Tools” , 27 June 2015 - 02 July 2015.
- [18] Scott Barnett, Rajesh Vasa, John Grundy , “ Bootstrapping Mobile App Development”, 17 August 2015.
- [19] Mona Erfani Joorabchi ,Ali Mesbah, Philippe Kruchten , “Real Challenges in Mobile App Development” ,10-11 October 2013.
- [20] Michiel Willocx , Jan Vossaert ,Vincent Naessens , “ Comparing performance parameters of mobile app development strategies” , 14 May 2016.

10. Appendix A:



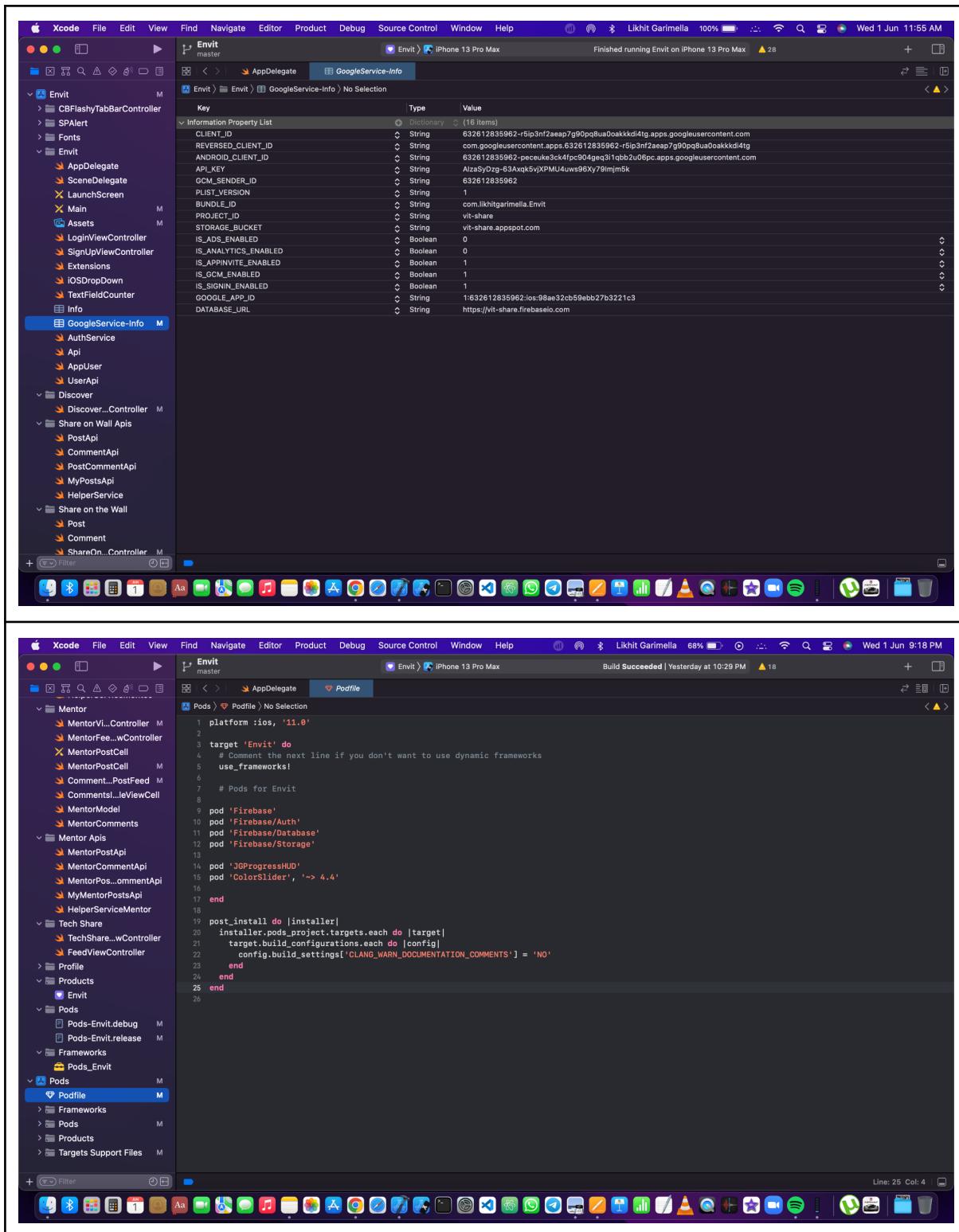


Fig 15: Xcode project, Info plist, Google service plist, Podfile Screenshots