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.

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:

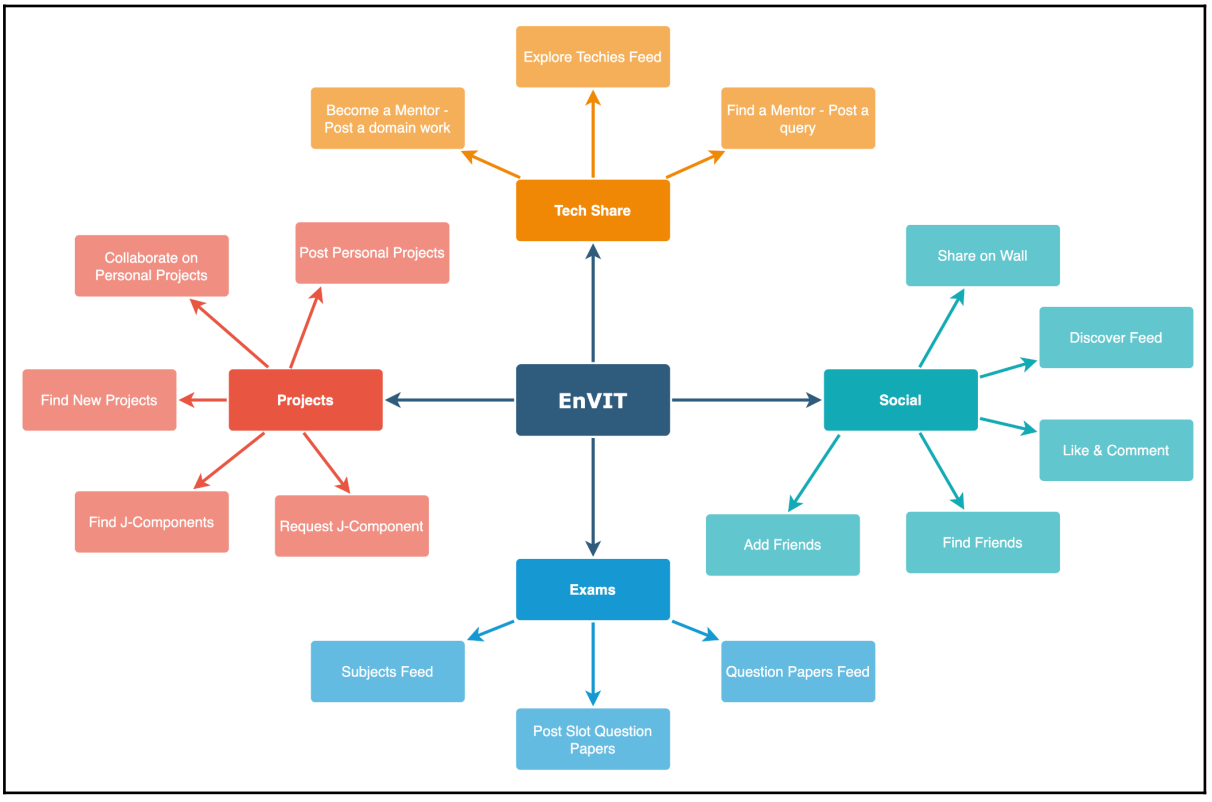


Fig 1: Architectural Diagram

4. Design Approach and Details:

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

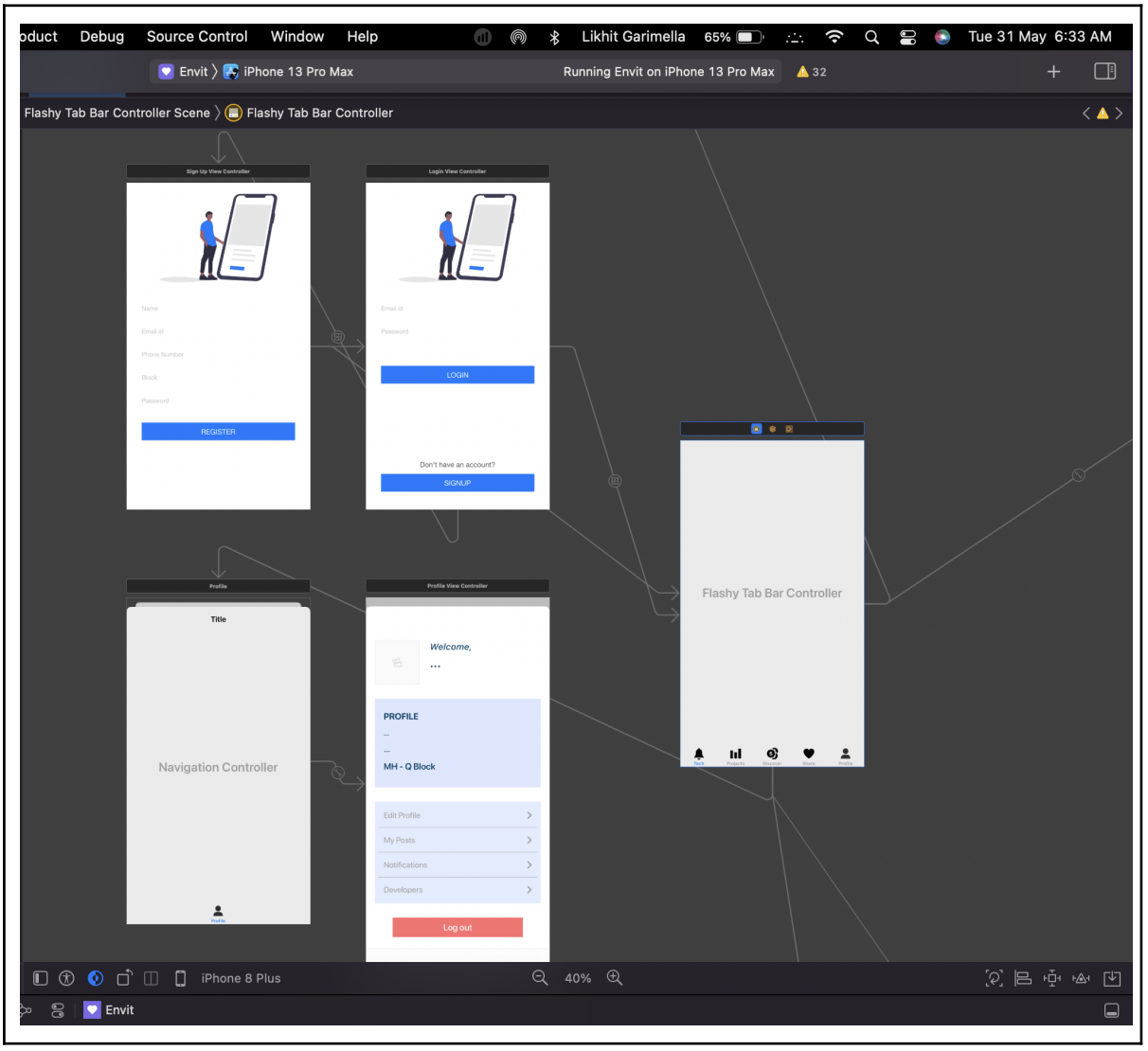


Fig 2: Signup, Login, Home & Profile Screens Storyboarding

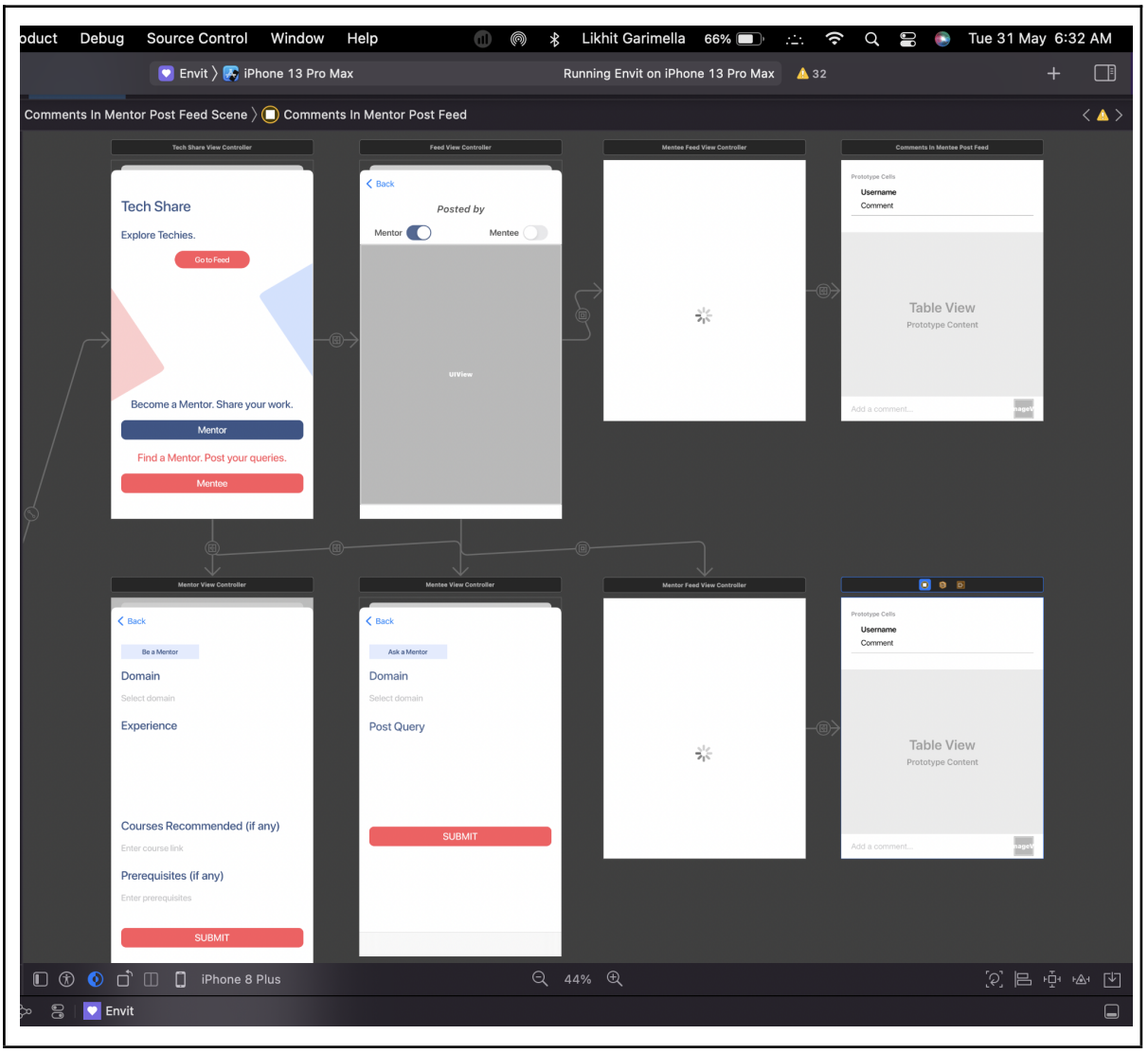


Fig 3: Tech Share - (Mentor + Mentee) Storyboarding

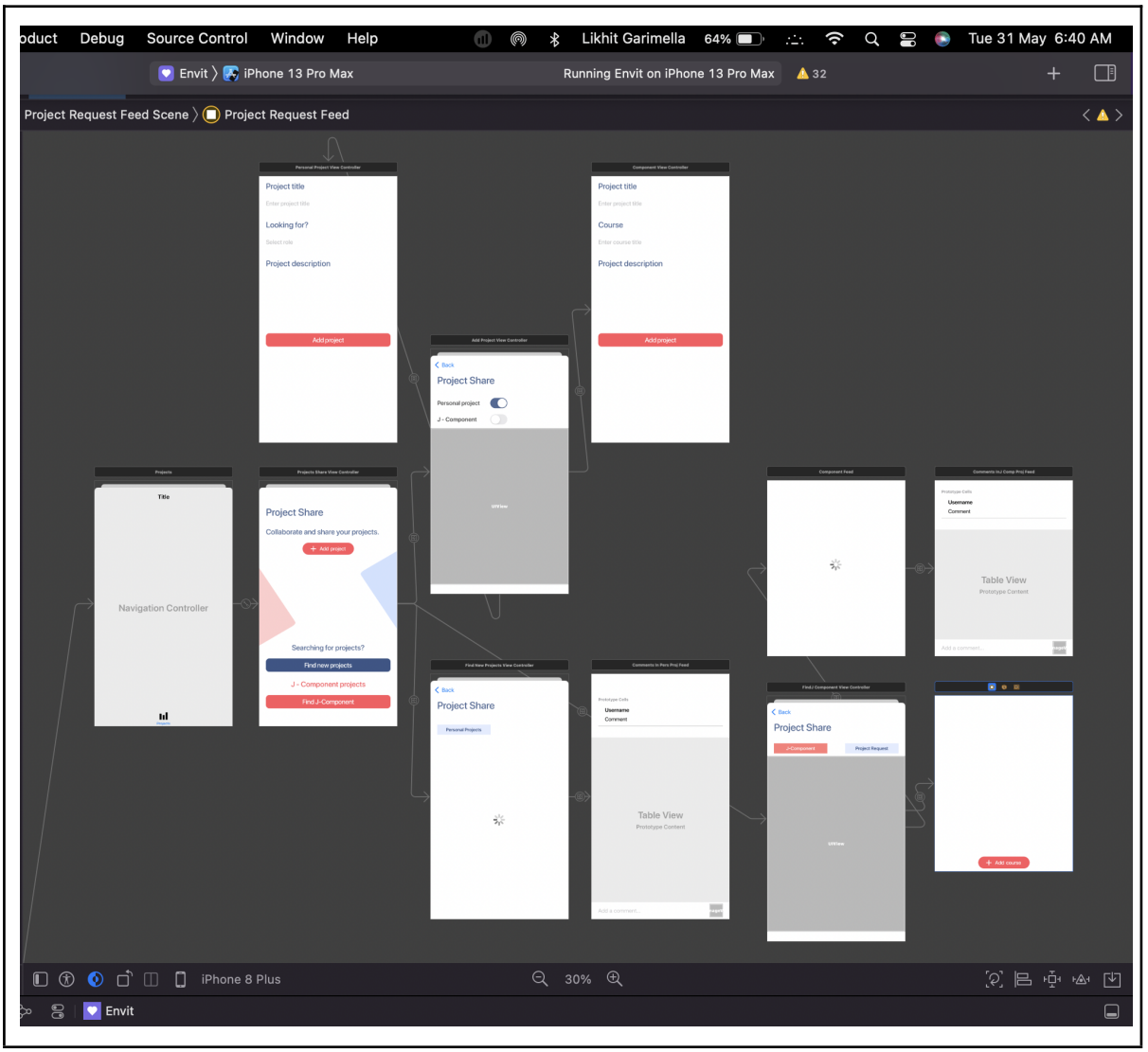


Fig 4: Project Share Storyboarding

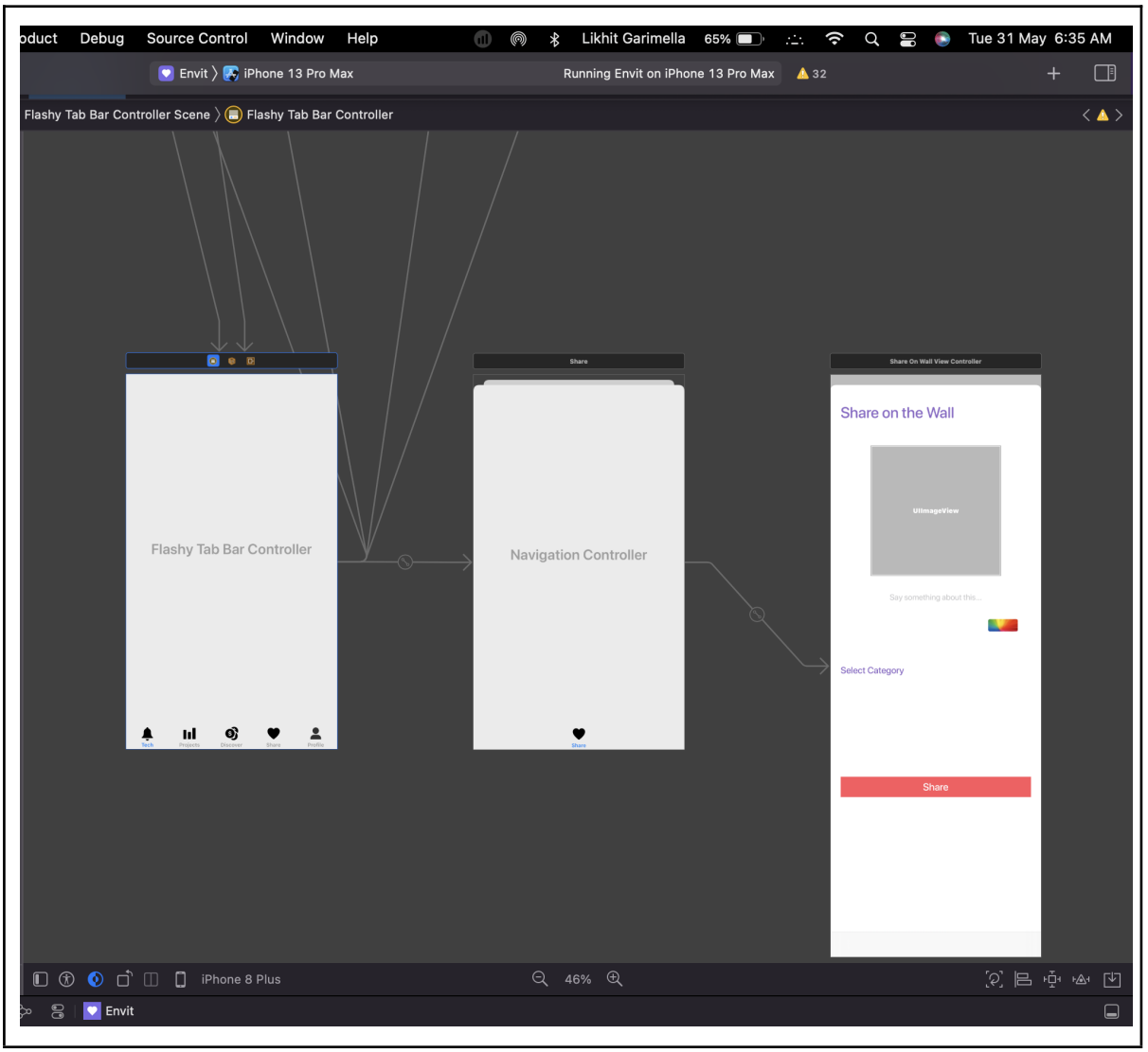


Fig 5: Share on the Wall Storyboarding

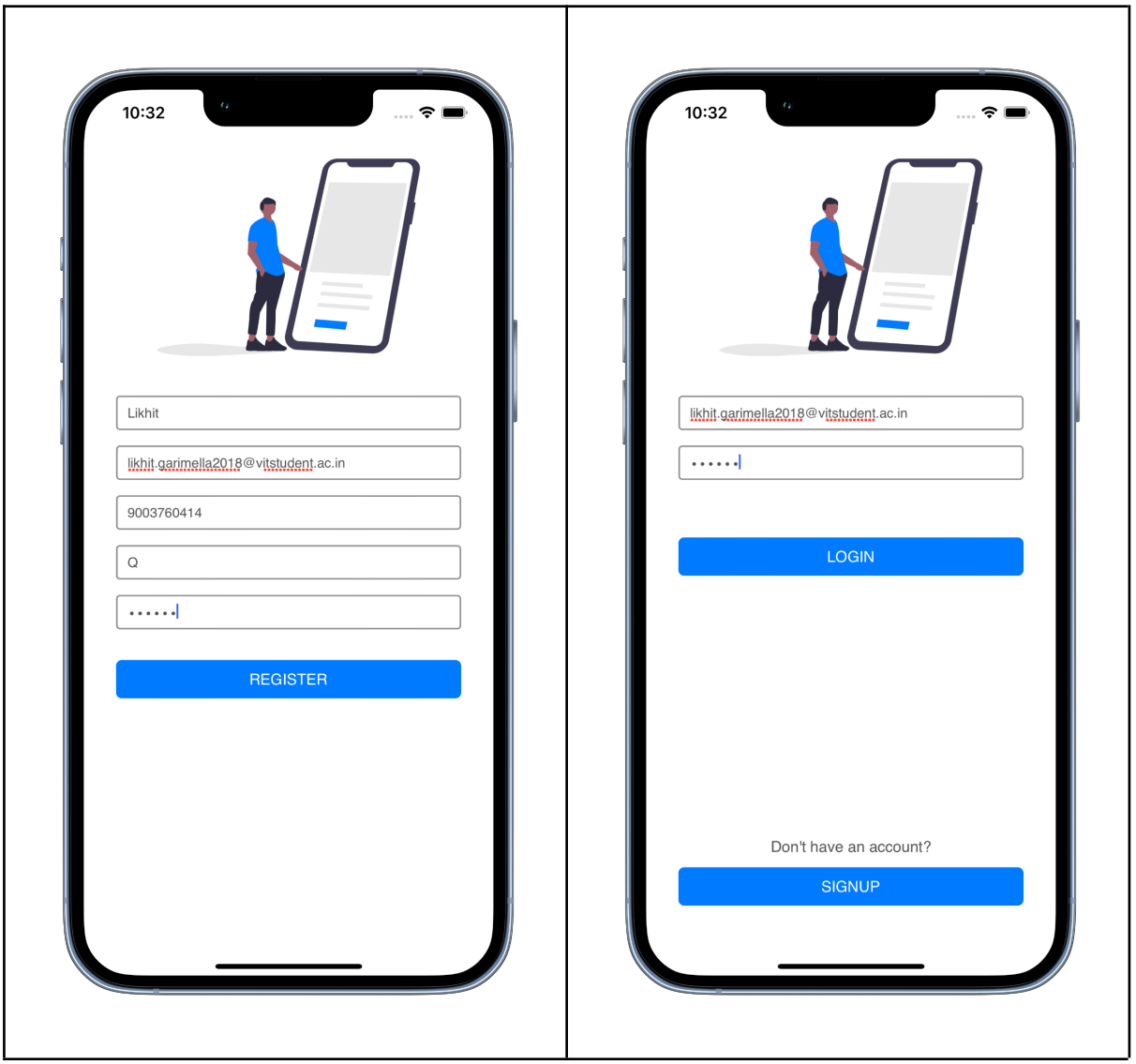


Fig 6: Signup & Login Screens

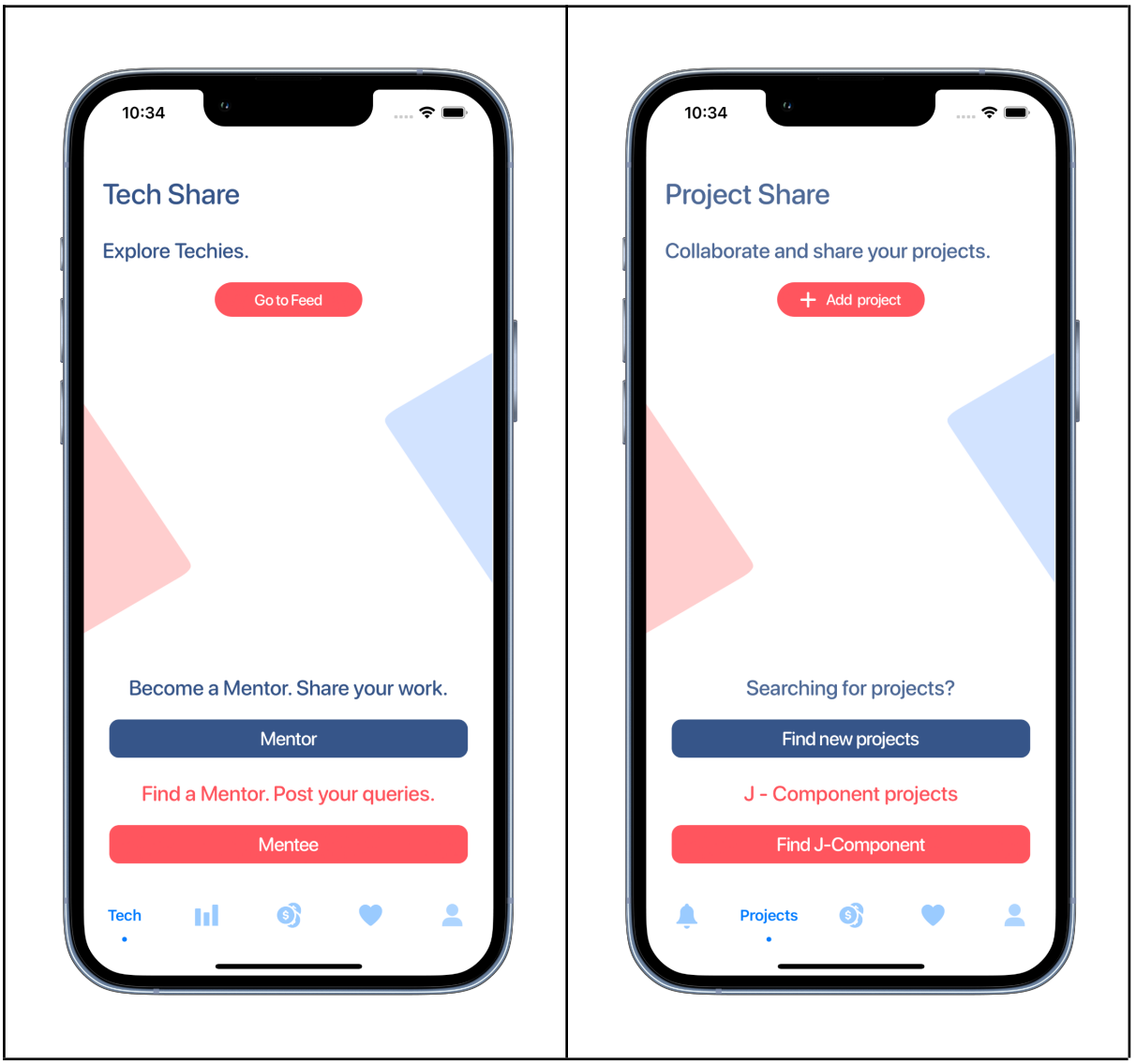


Fig 7: Tech Share & Project Share Screens

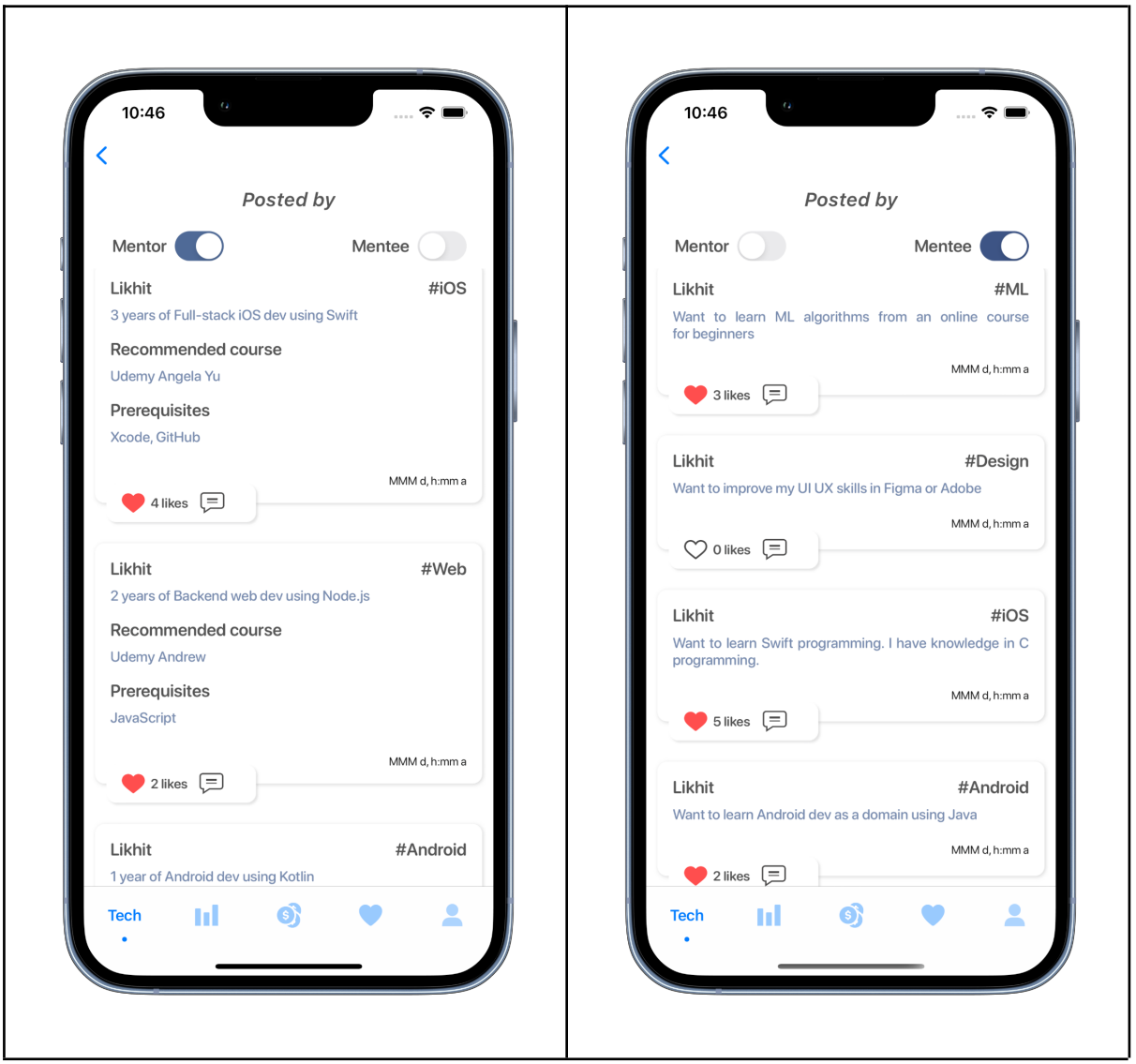


Fig 8: Mentor & Mentee Feed Screens

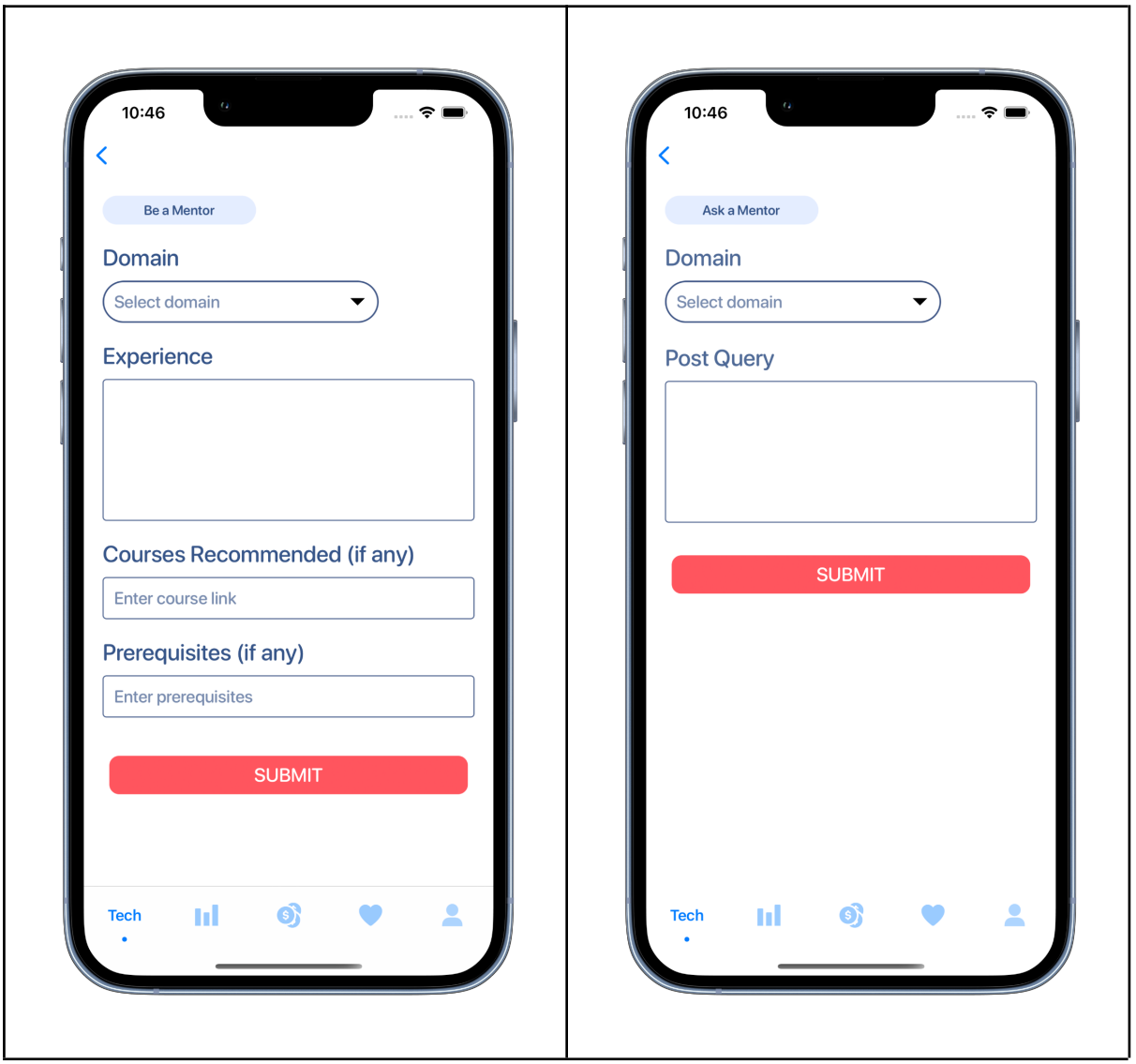


Fig 9: Mentor & Mentee Post Screens

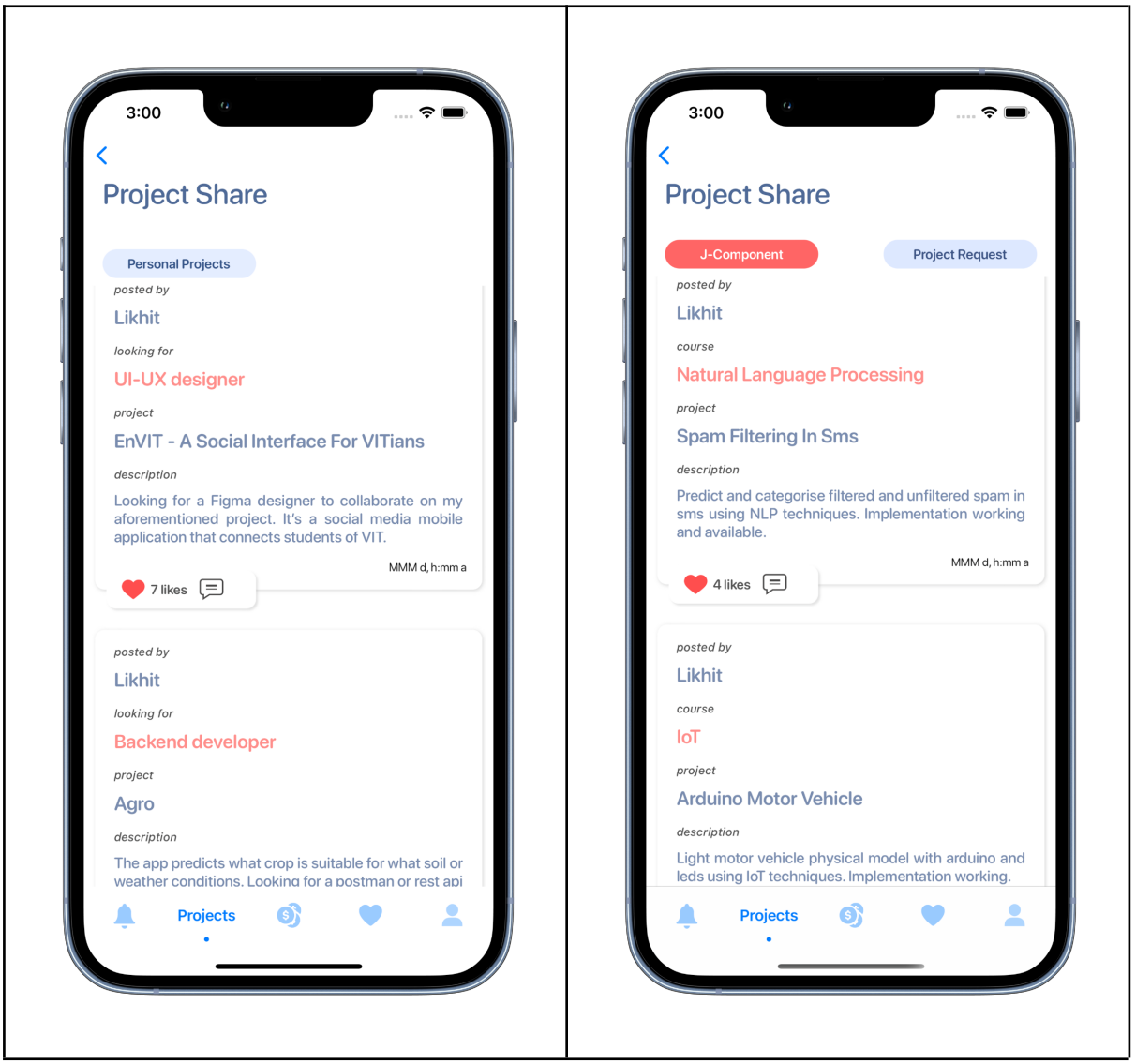


Fig 10: Personal Projects Feed & J-Component Projects Feed

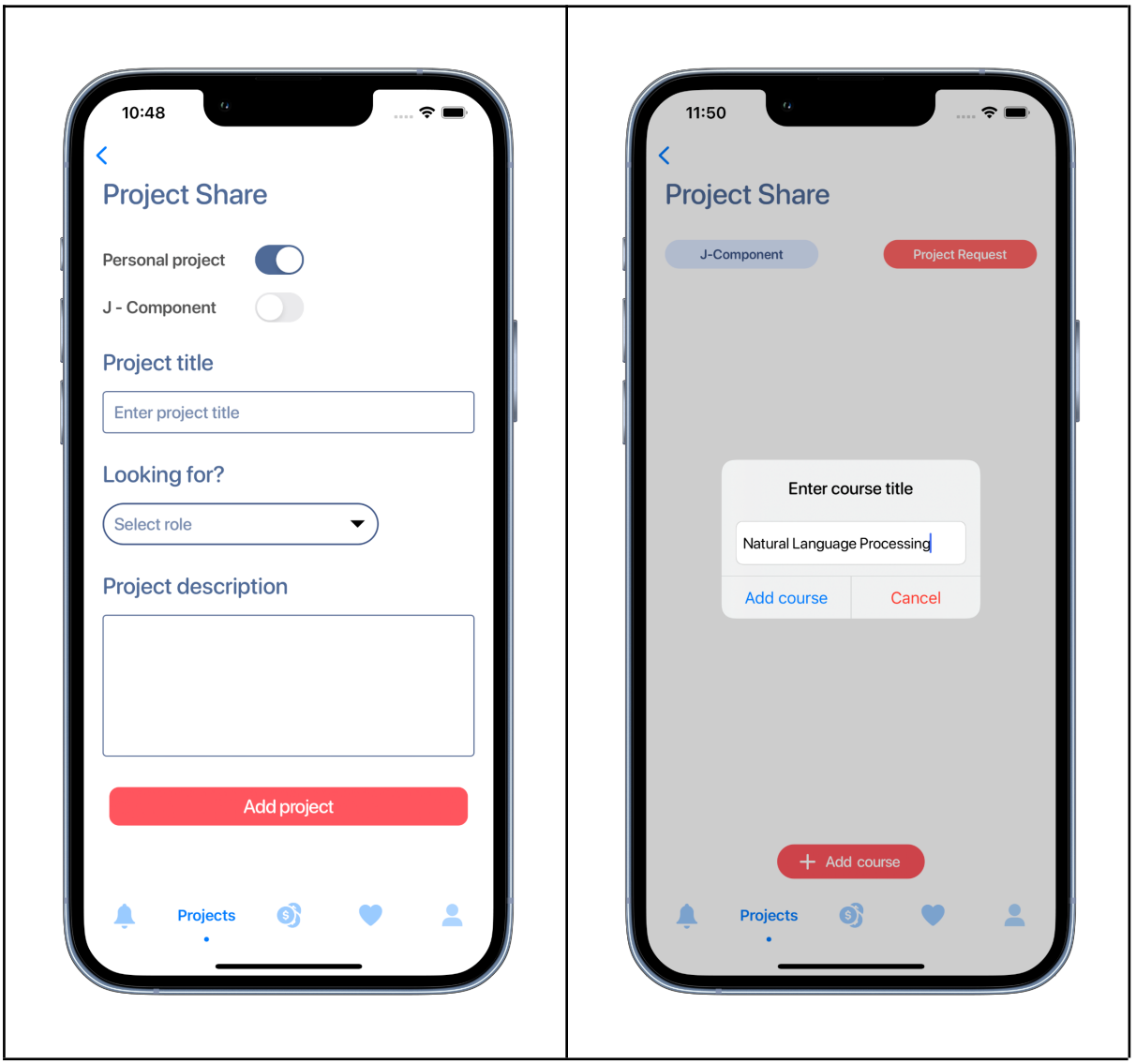


Fig 11: Project Post & Project Request

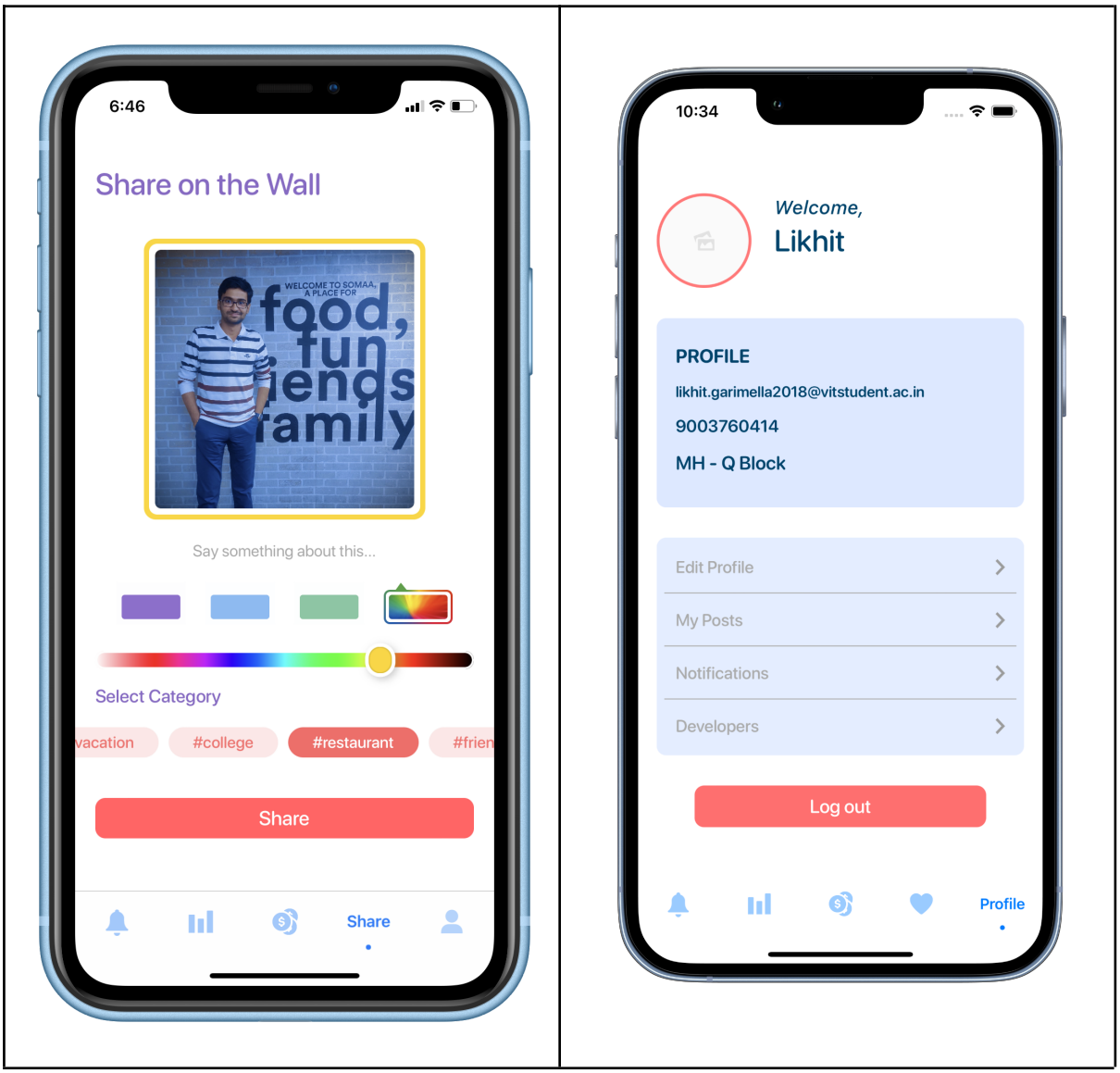
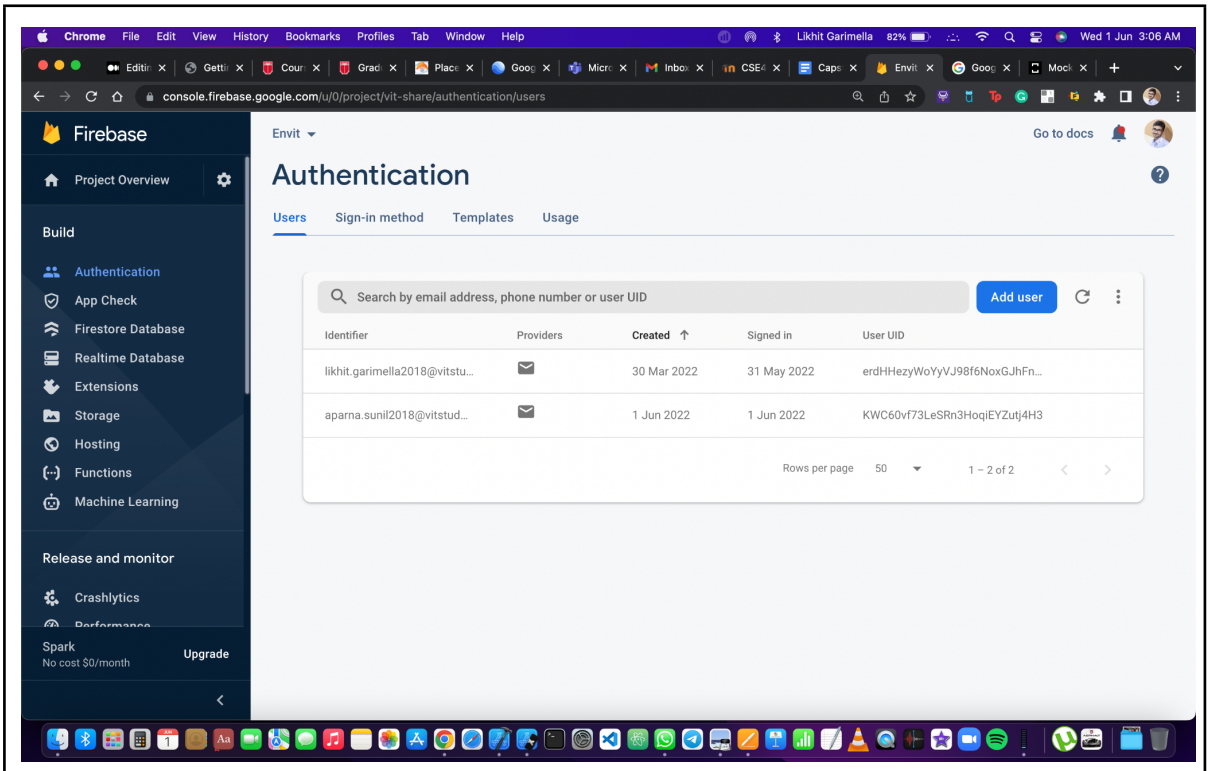
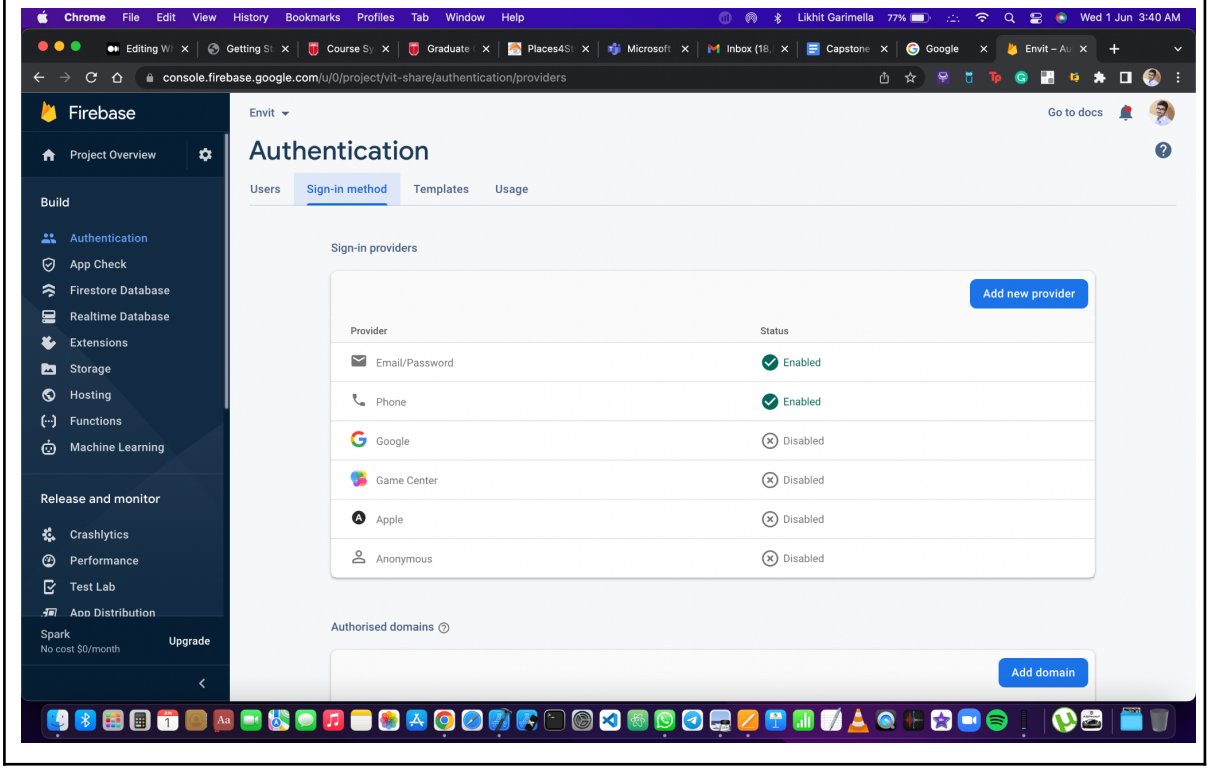
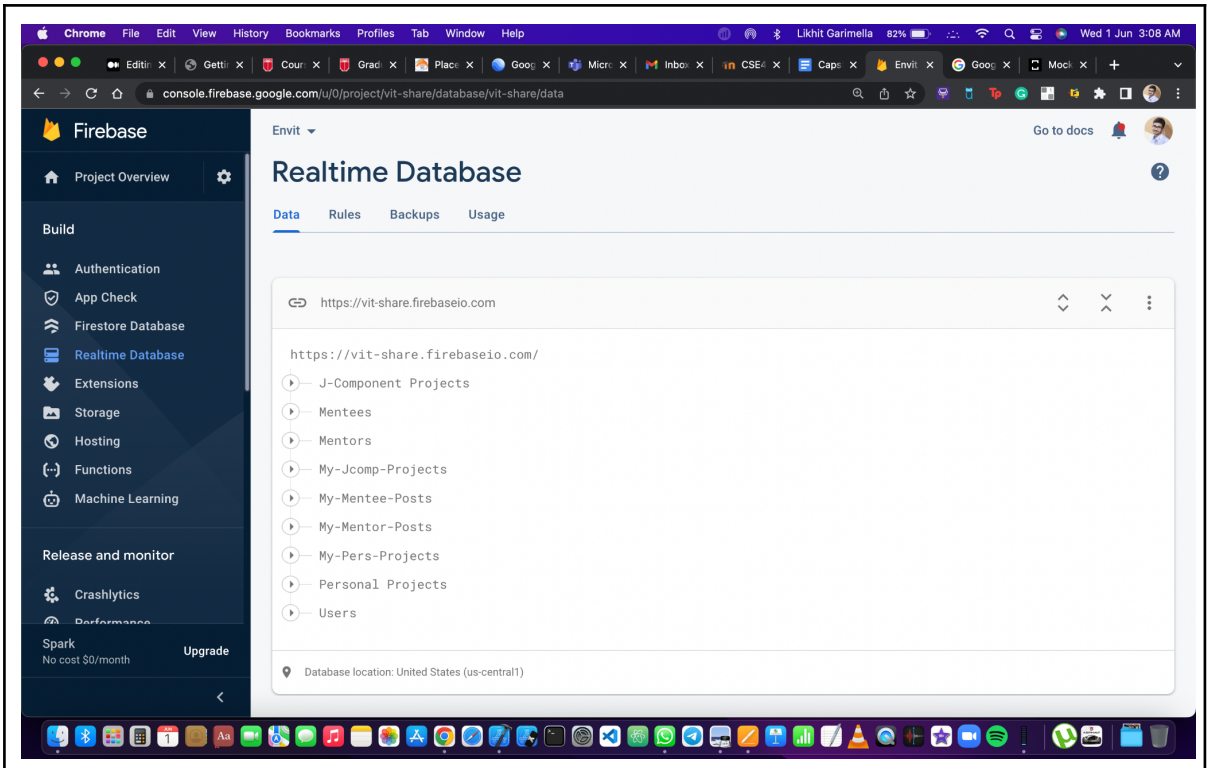


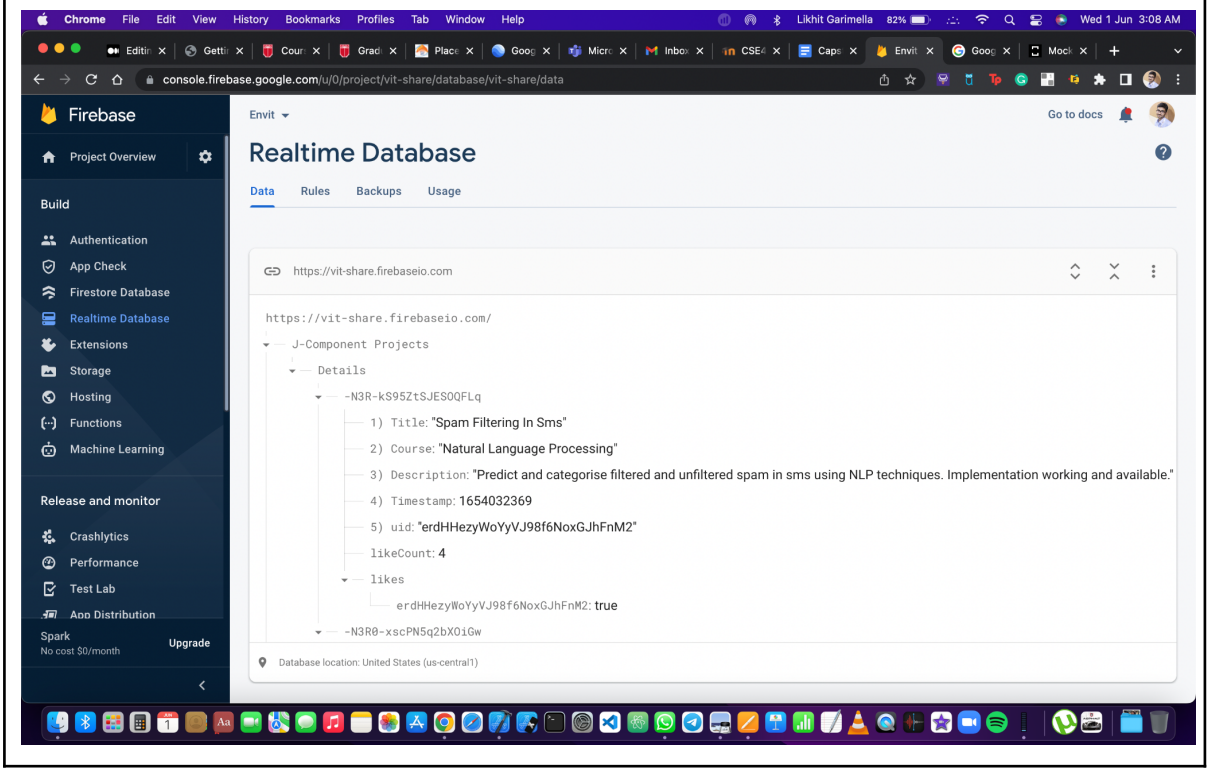
Fig 12: Share on the Wall Screen & Profile Screen

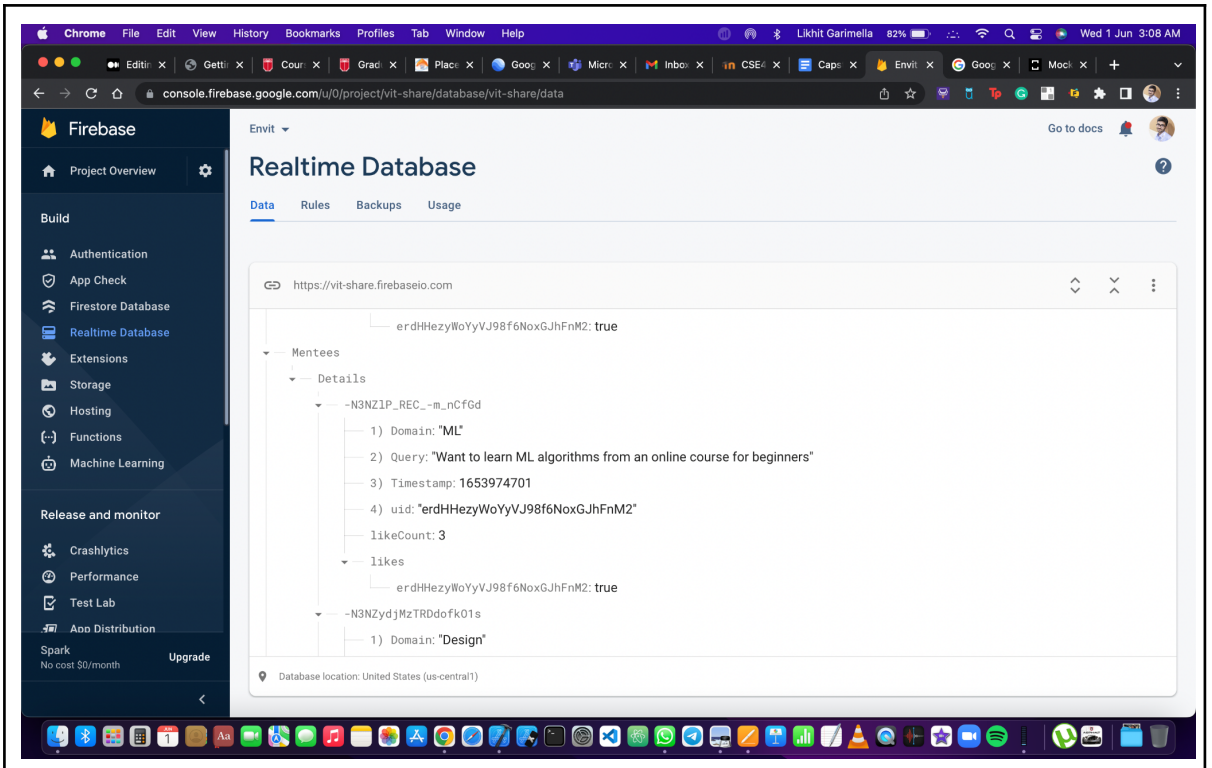
4.2 Codes and Standards: - Xcode Swift Codes & Firebase Server Screenshots:

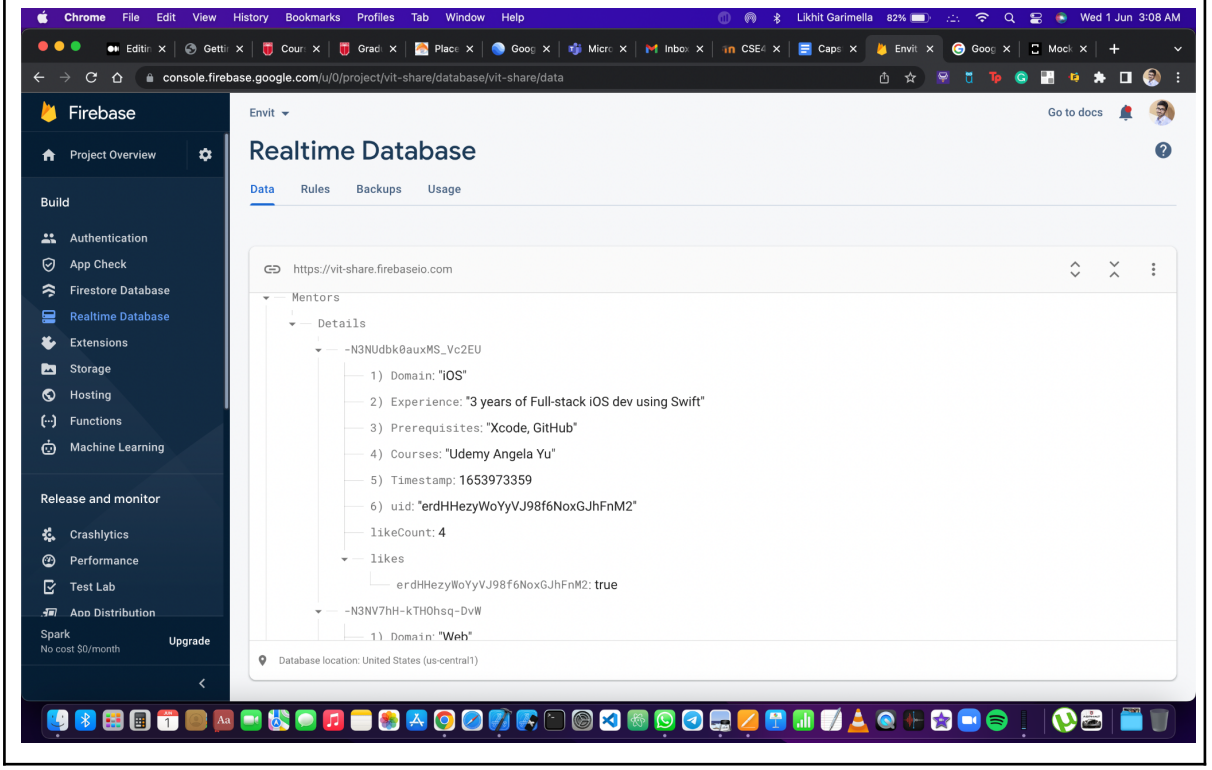


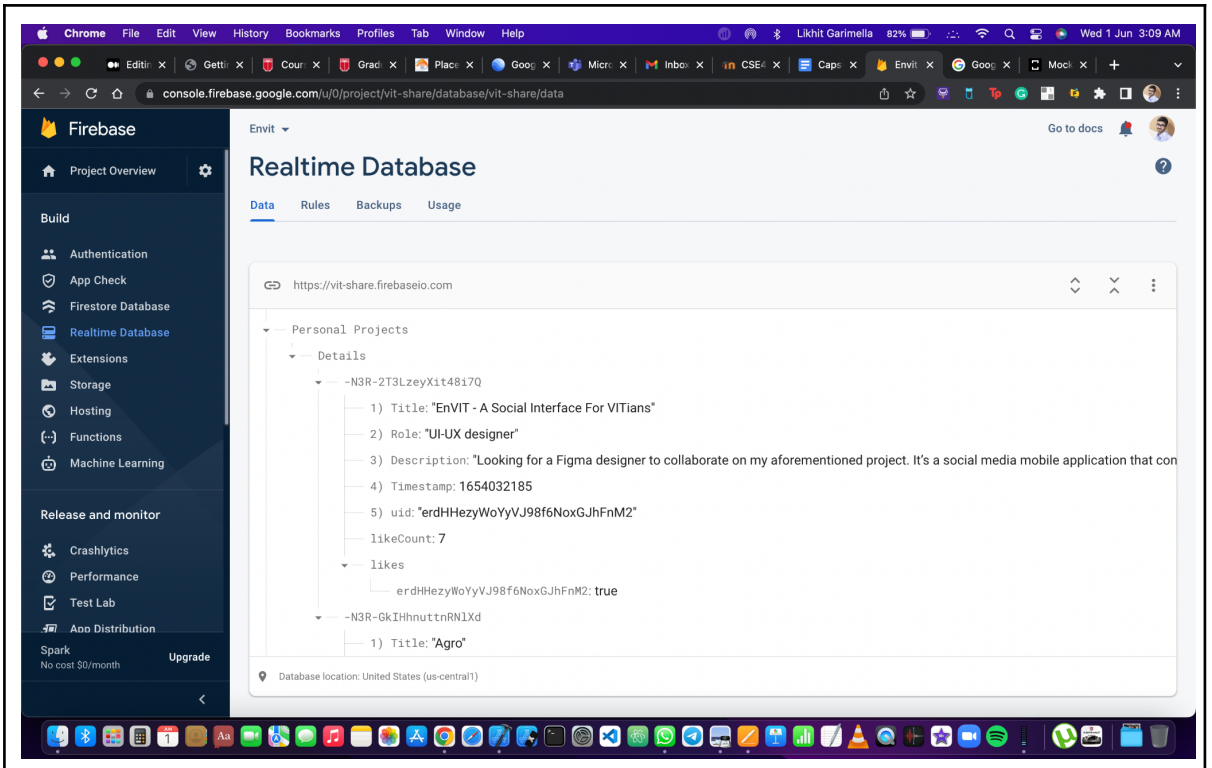


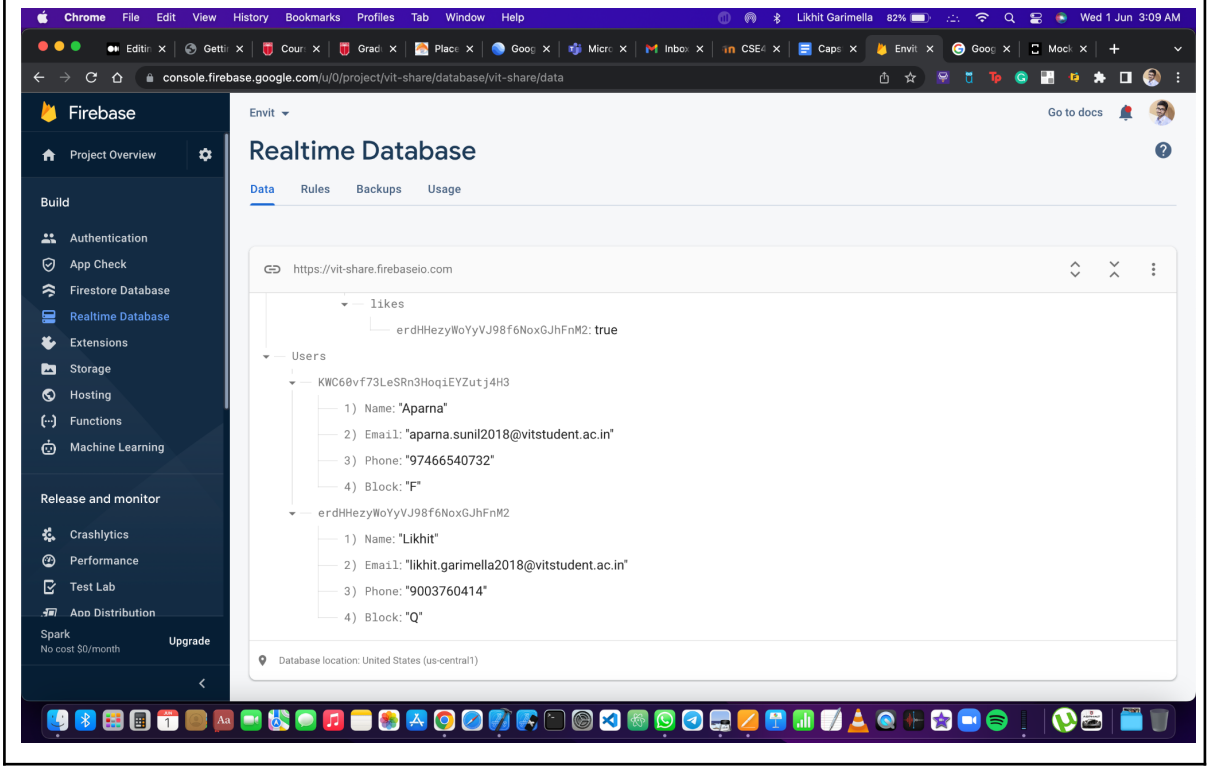

Firebase
Project Overview
Build
Authentication
App Check
Firestore Database
Realtime Database
Extensions
Storage
Hosting
Functions
Machine Learning
Release and monitor
Crashlytics
Spark
No cost $0/month
Upgrade
Envit
Go to docs
Realtime Database
Data
Rules
Backups
Usage
https://vit-share.firebaseio.com
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)


Firebase
Project Overview
Build
Authentication
App Check
Firestore Database
Realtime Database
Extensions
Storage
Hosting
Functions
Machine Learning
Release and monitor
Crashlytics
Performance
Test Lab
Spark
No cost $0/month
Upgrade
Envit
Go to docs
Realtime Database
Data
Rules
Backups
Usage
https://vit-share.firebaseio.com
https://vit-share.firebaseio.com/
J-Component Projects
Details
-N3R-kS95ZtSJESOQFLq
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-xscPN5q2bXOiGw
Database location: United States (us-central1)


Firebase
Project Overview
Build
Authentication
App Check
Firestore Database
Realtime Database
Extensions
Storage
Hosting
Functions
Machine Learning
Release and monitor
Crashlytics
Performance
Test Lab
App Distribution
Spark
No cost $0/month
Upgrade
Realtime Database
Data
Rules
Backups
Usage
https://vit-share.firebaseio.com
erdHHezyWoYyVJ98f6NoxGJhFnM2: true
Mentees
Details
-N3NZlP_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-central1)


Firebase
Project Overview
Build
Authentication
App Check
Firestore Database
Realtime Database
Extensions
Storage
Hosting
Functions
Machine Learning
Release and monitor
Crashlytics
Performance
Test Lab
App Distribution
Spark
No cost $0/month
Upgrade
Realtime Database
Data
Rules
Backups
Usage
https://vit-share.firebaseio.com
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
Database location: United States (us-central1)


console.firebase.google.com/u/0/project/vit-share/database/vit-share/data
Firebase
Project Overview
Build
Authentication
App Check
Firestore Database
Realtime Database
Extensions
Storage
Hosting
Functions
Machine Learning
Release and monitor
Crashlytics
Performance
Test Lab
Spark
No cost $0/month
Upgrade
Envit
Go to docs
Realtime Database
Data
Rules
Backups
Usage
https://vit-share.firebaseio.com
Personal Projects
Details
-N3R-2T3LzeyXit48i7Q
1) Title: "EnVIT - A Social Interface For VITians"
2) Role: "UI-UX designer"
3) Description: "Looking for a Figma designer to collaborate on my aforementioned project. It's a social media mobile application that con
4) Timestamp: 1654032185
5) uid: "erdHHezyWoYyVJ98f6NoxGJhFnM2"
likeCount: 7
likes
erdHHezyWoYyVJ98f6NoxGJhFnM2: true
-N3R-GkIHhnuttnRN1Xd
1) Title: "Agro"
Database location: United States (us-central1)


console.firebase.google.com/u/0/project/vit-share/database/vit-share/data
Firebase
Project Overview
Build
Authentication
App Check
Firestore Database
Realtime Database
Extensions
Storage
Hosting
Functions
Machine Learning
Release and monitor
Crashlytics
Performance
Test Lab
Spark
No cost $0/month
Upgrade
Envit
Go to docs
Realtime Database
Data
Rules
Backups
Usage
https://vit-share.firebaseio.com
likes
erdHHezyWoYyVJ98f6NoxGJhFnM2: true
Users
KWC60vf73LeSRn3HoqiEYZutj4H3
1) Name: "Aparna"
2) Email: "aparna.sunil2018@vitstudent.ac.in"
3) Phone: "97466540732"
4) Block: "F"
erdHHezyWoYyVJ98f6NoxGJhFnM2
1) Name: "Likhit"
2) Email: "likhit.garimella2018@vitstudent.ac.in"
3) Phone: "9003760414"
4) Block: "Q"
Database location: United States (us-central1)

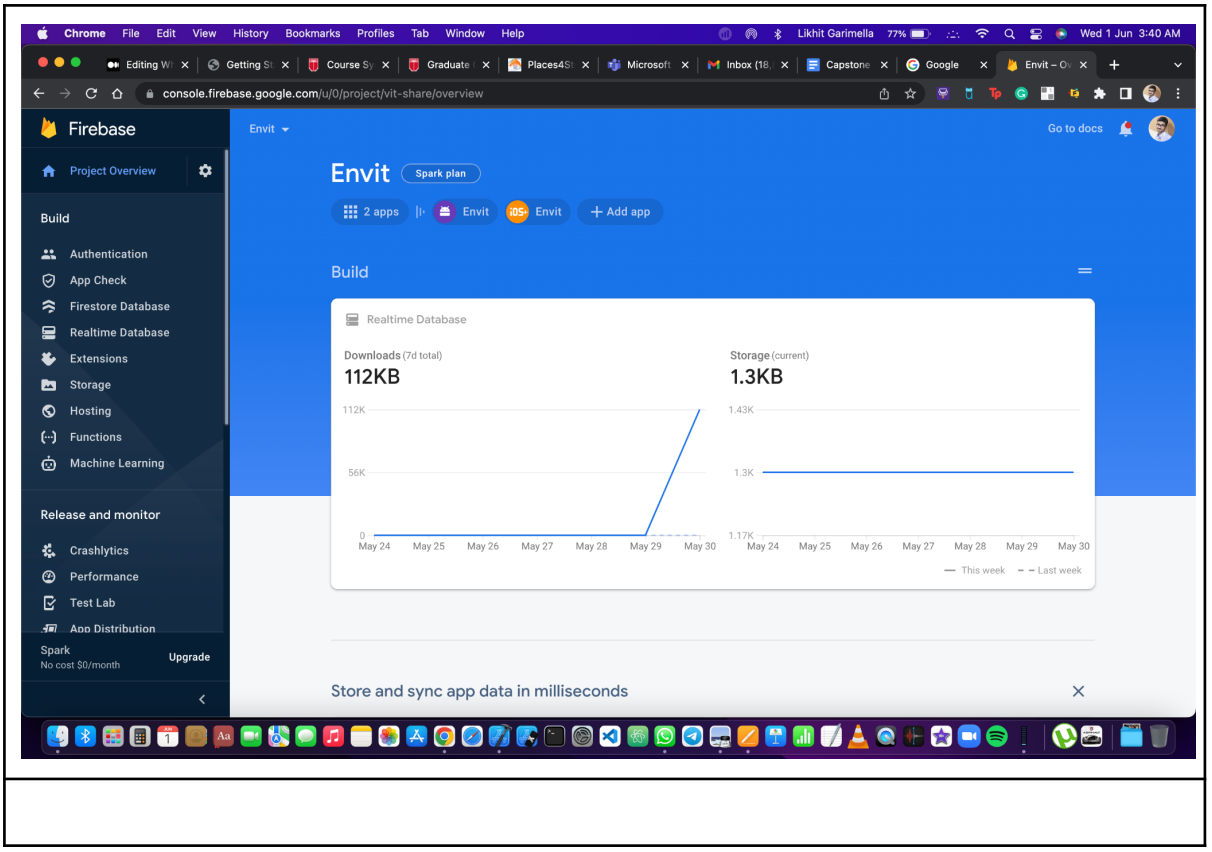
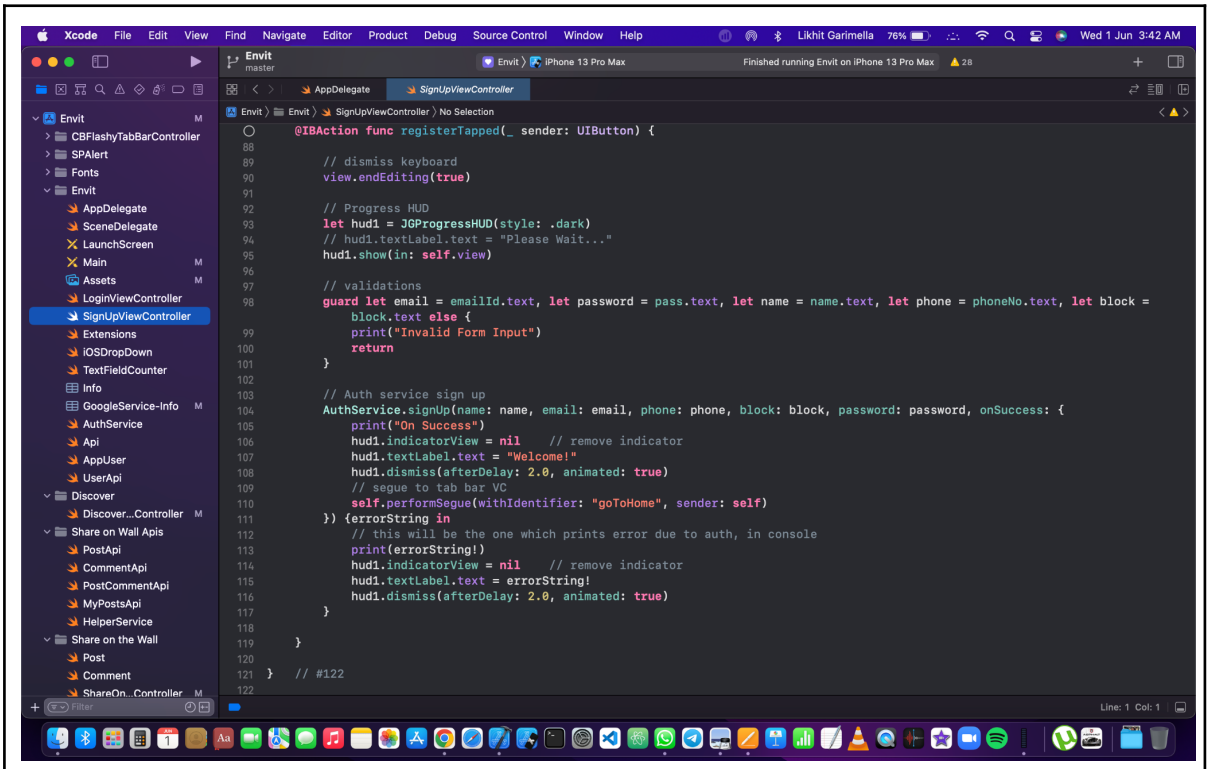
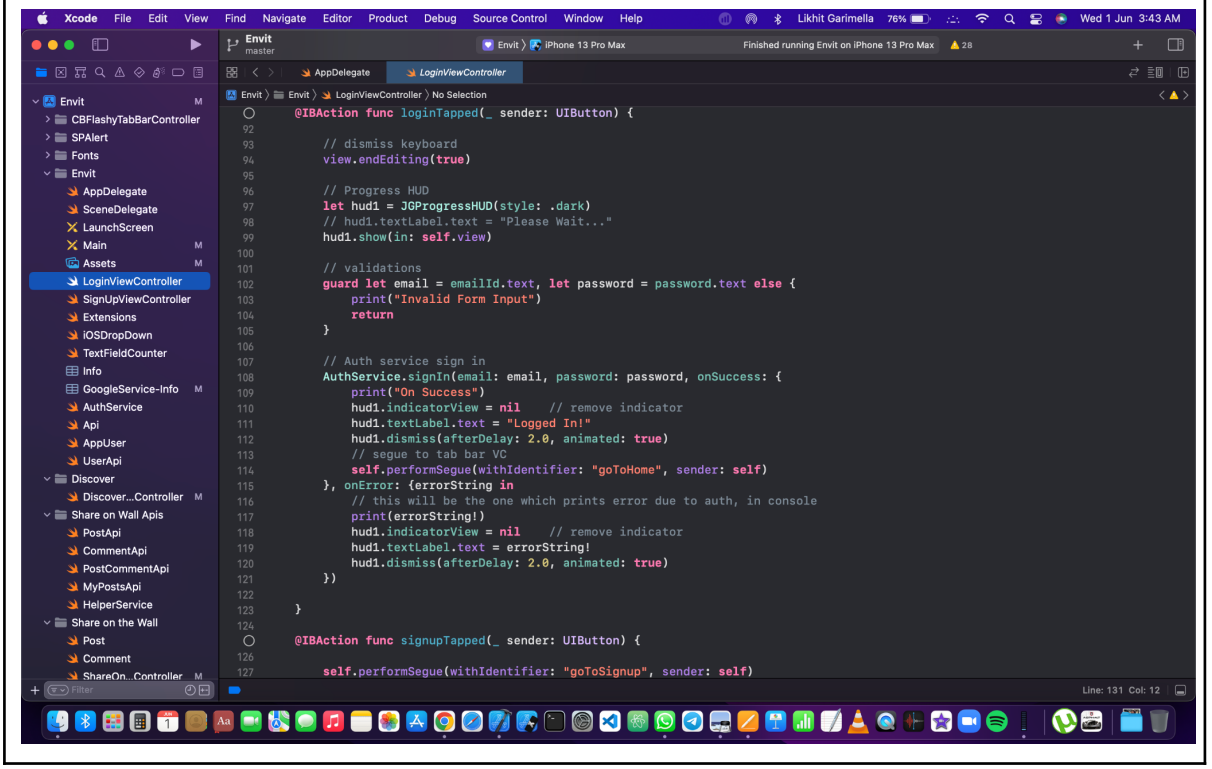


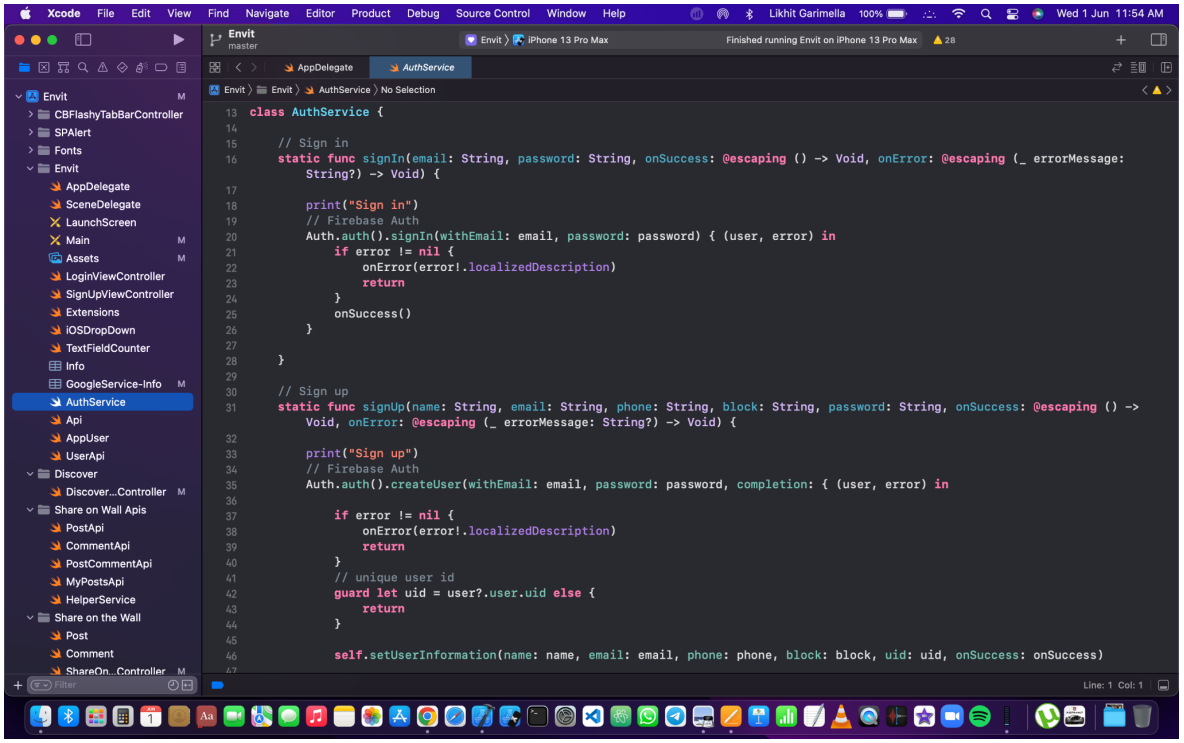
Fig 13: Firebase Server Screenshots



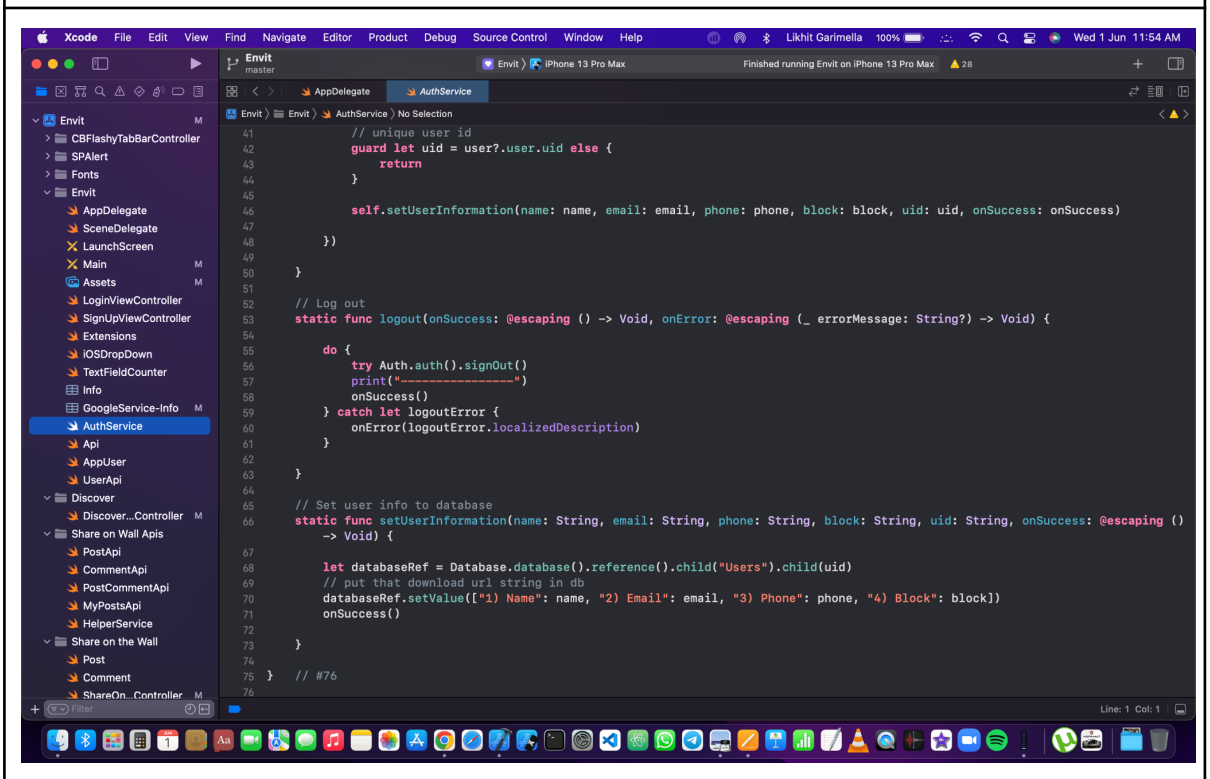
```
    @IBAction func registerTapped(_ sender: UIButton) {

        // 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)
        }

    }

}   // #122
```



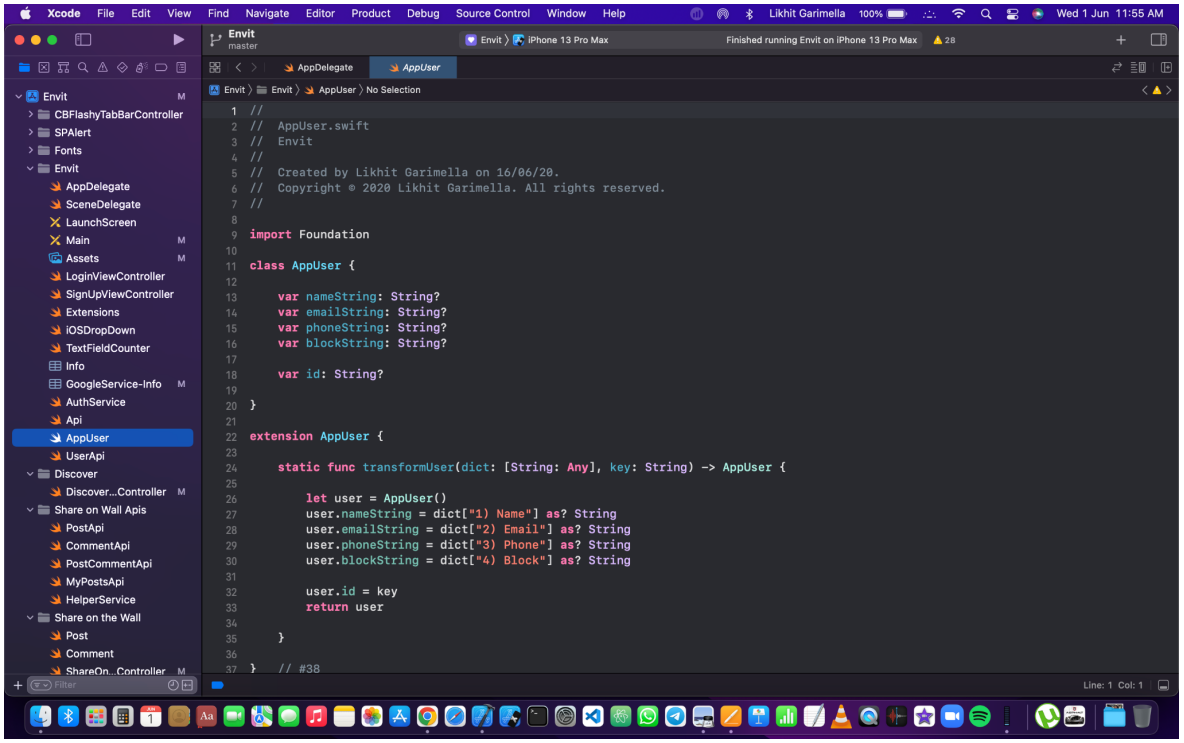
```
    @IBAction func loginTapped(_ sender: UIButton) {

        // 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)
```



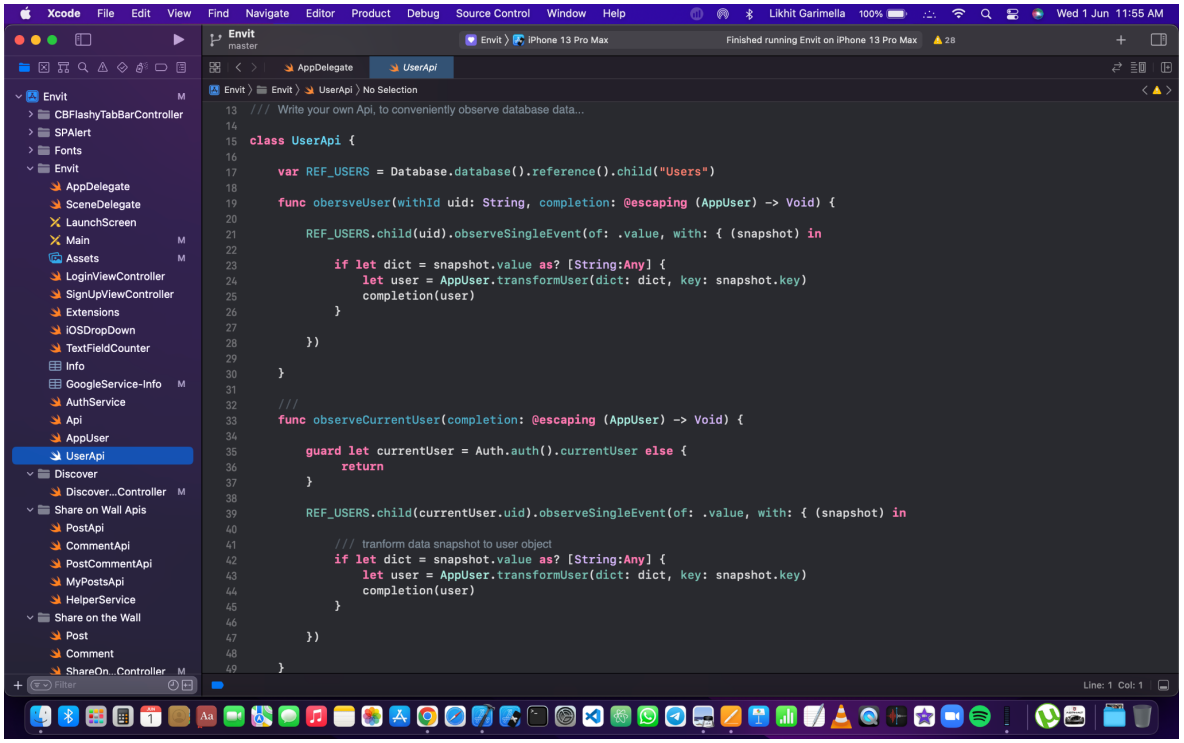
```
class AuthService {

    // Sign in
    static func signIn(email: String, password: String, onSuccess: @escaping () -> Void, onError: @escaping (_ errorMessage:
        String?) -> Void) {

        print("Sign in")
        // Firebase Auth
        Auth.auth().signIn(withEmail: email, password: password) { (user, error) in
            if error != nil {
                onError(error!.localizedDescription)
                return
            }
            onSuccess()
        }

    }

    // Sign up
    static func signUp(name: String, email: String, phone: String, block: String, password: String, onSuccess: @escaping () ->
        Void, onError: @escaping (_ errorMessage: String?) -> Void) {

        print("Sign up")
        // Firebase Auth
        Auth.auth().createUser(withEmail: email, password: password, completion: { (user, error) in

            if error != nil {
                onError(error!.localizedDescription)
                return
            }
            // unique user id
            guard let uid = user?.user.uid else {
                return
            }

            self.setUserInformation(name: name, email: email, phone: phone, block: block, uid: uid, onSuccess: onSuccess)
```



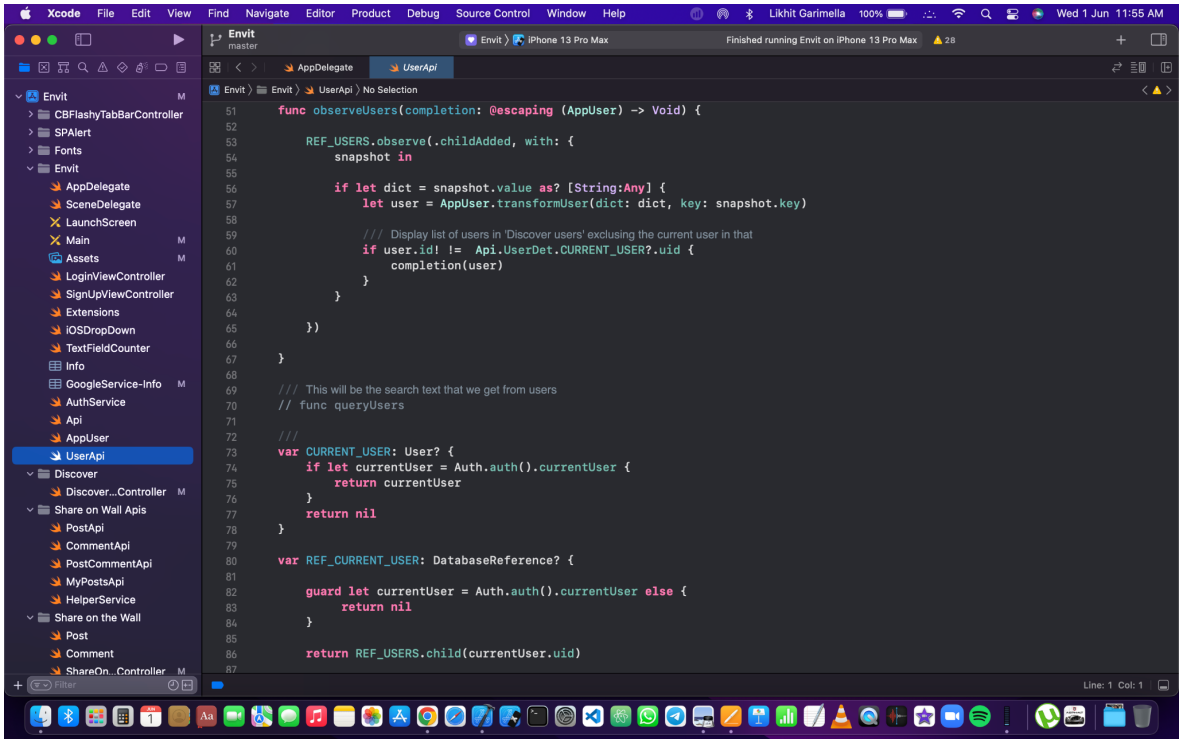
```
            // unique user id
            guard let uid = user?.user.uid else {
                return
            }

            self.setUserInformation(name: name, email: email, phone: phone, block: block, uid: uid, onSuccess: onSuccess)

        })

    }

    // Log out
    static func logout(onSuccess: @escaping () -> Void, onError: @escaping (_ errorMessage: String?) -> Void) {

        do {
            try Auth.auth().signOut()
            print("----------------")
            onSuccess()
        } catch let logoutError {
            onError(logoutError.localizedDescription)
        }

    }

    // Set user info to database
    static func setUserInformation(name: String, email: String, phone: String, block: String, uid: String, onSuccess: @escaping ()
        -> Void) {

        let databaseRef = Database.database().reference().child("Users").child(uid)
        // put that download url string in db
        databaseRef.setValue(["1) Name": name, "2) Email": email, "3) Phone": phone, "4) Block": block])
        onSuccess()

    }

}   // #76
```



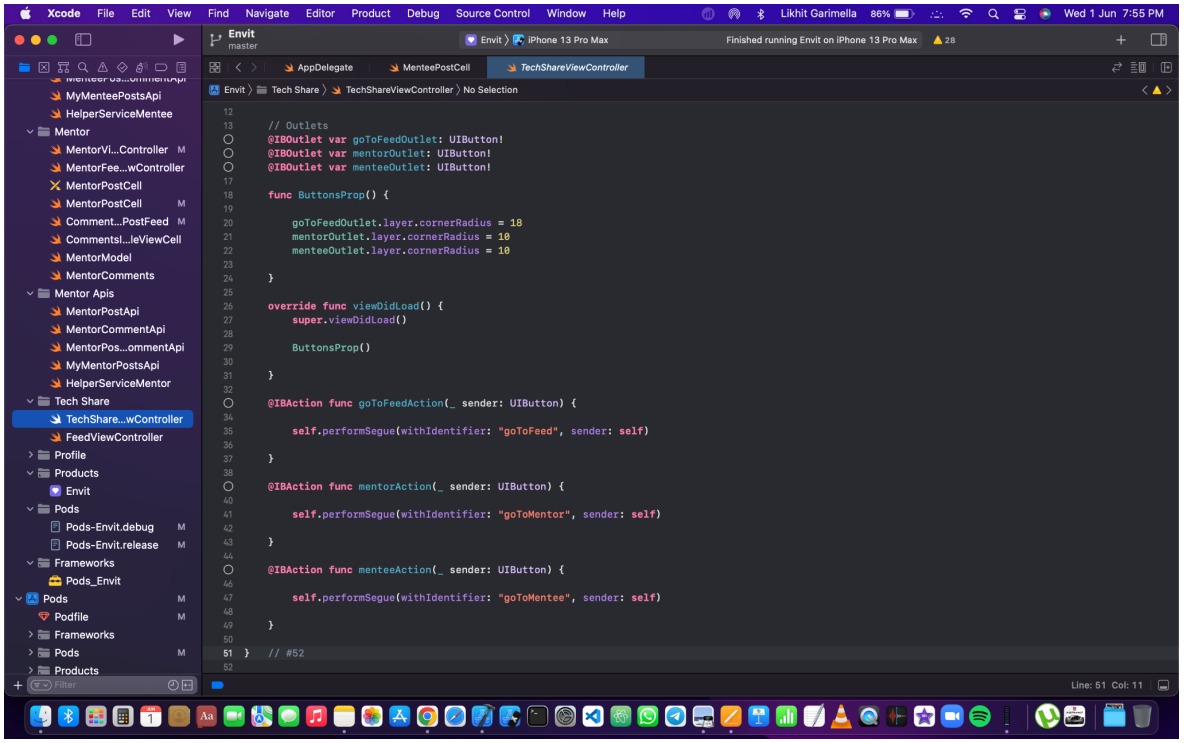
```
//
//  AppUser.swift
//  Envit
//
//  Created by Likhit Garimella on 16/06/20.
//  Copyright © 2020 Likhit Garimella. All rights reserved.
//

import Foundation

class AppUser {

    var nameString: String?
    var emailString: String?
    var phoneString: String?
    var blockString: String?

    var id: String?

}

extension AppUser {

    static func transformUser(dict: [String: Any], key: String) -> AppUser {

        let user = AppUser()
        user.nameString = dict["1) Name"] as? String
        user.emailString = dict["2) Email"] as? String
        user.phoneString = dict["3) Phone"] as? String
        user.blockString = dict["4) Block"] as? String

        user.id = key
        return user

    }

}  // #38
```



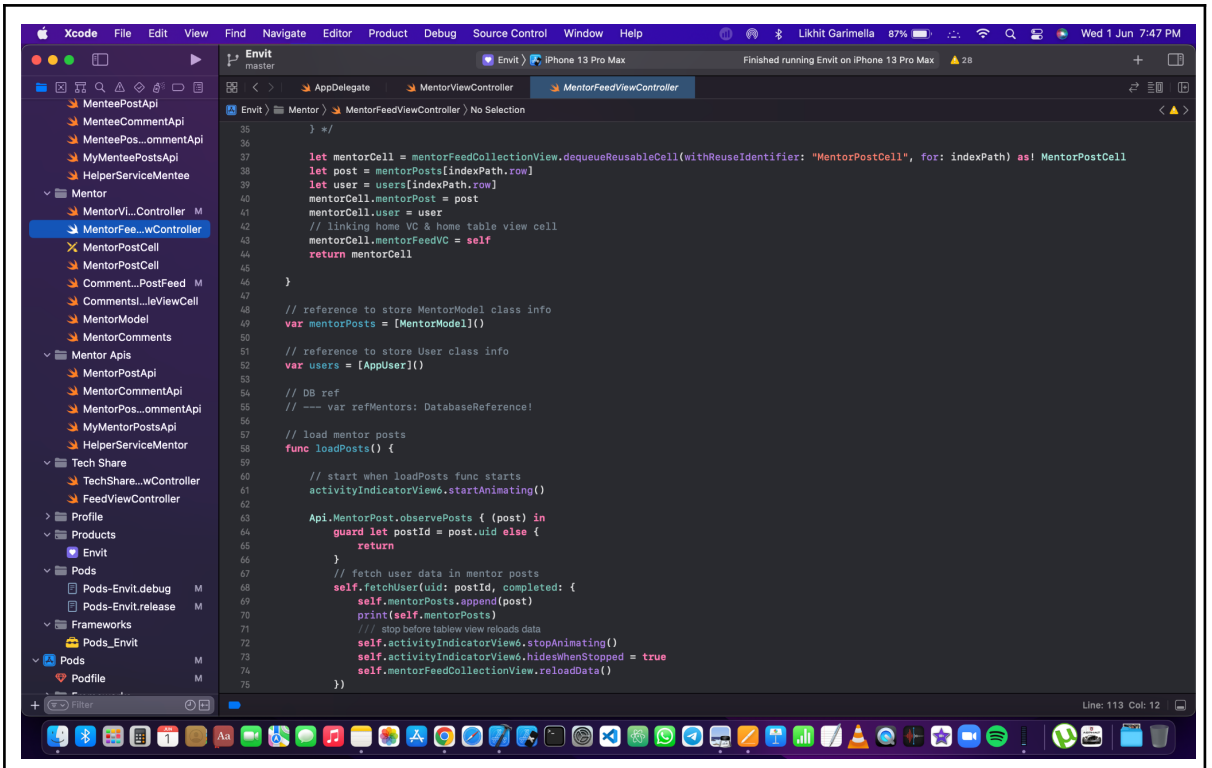
```
/// Write your own Api, to conveniently observe database data...

class UserApi {

    var REF_USERS = Database.database().reference().child("Users")

    func obersveUser(withId uid: String, completion: @escaping (AppUser) -> Void) {

        REF_USERS.child(uid).observeSingleEvent(of: .value, with: { (snapshot) in

            if let dict = snapshot.value as? [String:Any] {
                let user = AppUser.transformUser(dict: dict, key: snapshot.key)
                completion(user)
            }

        })

    }

    ///
    func observeCurrentUser(completion: @escaping (AppUser) -> Void) {

        guard let currentUser = Auth.auth().currentUser else {
             return
        }

        REF_USERS.child(currentUser.uid).observeSingleEvent(of: .value, with: { (snapshot) in

            /// tranform data snapshot to user object
            if let dict = snapshot.value as? [String:Any] {
                let user = AppUser.transformUser(dict: dict, key: snapshot.key)
                completion(user)
            }

        })

    }
```

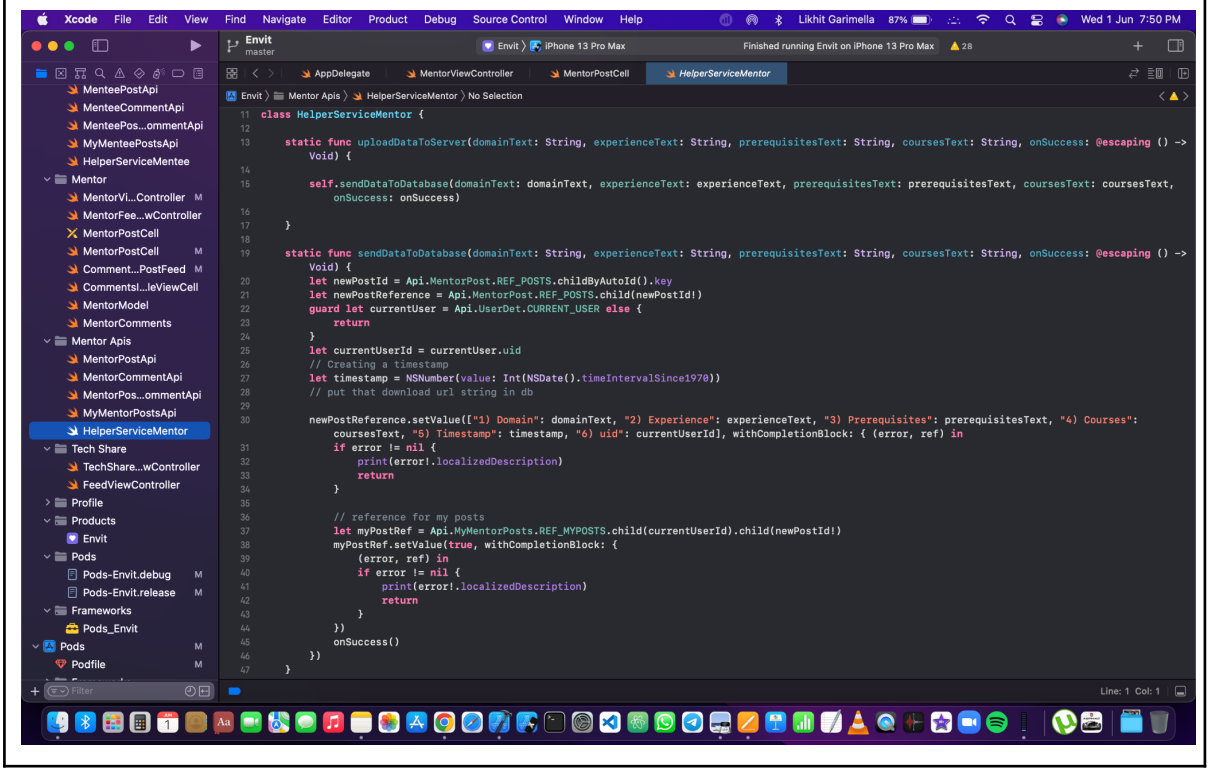


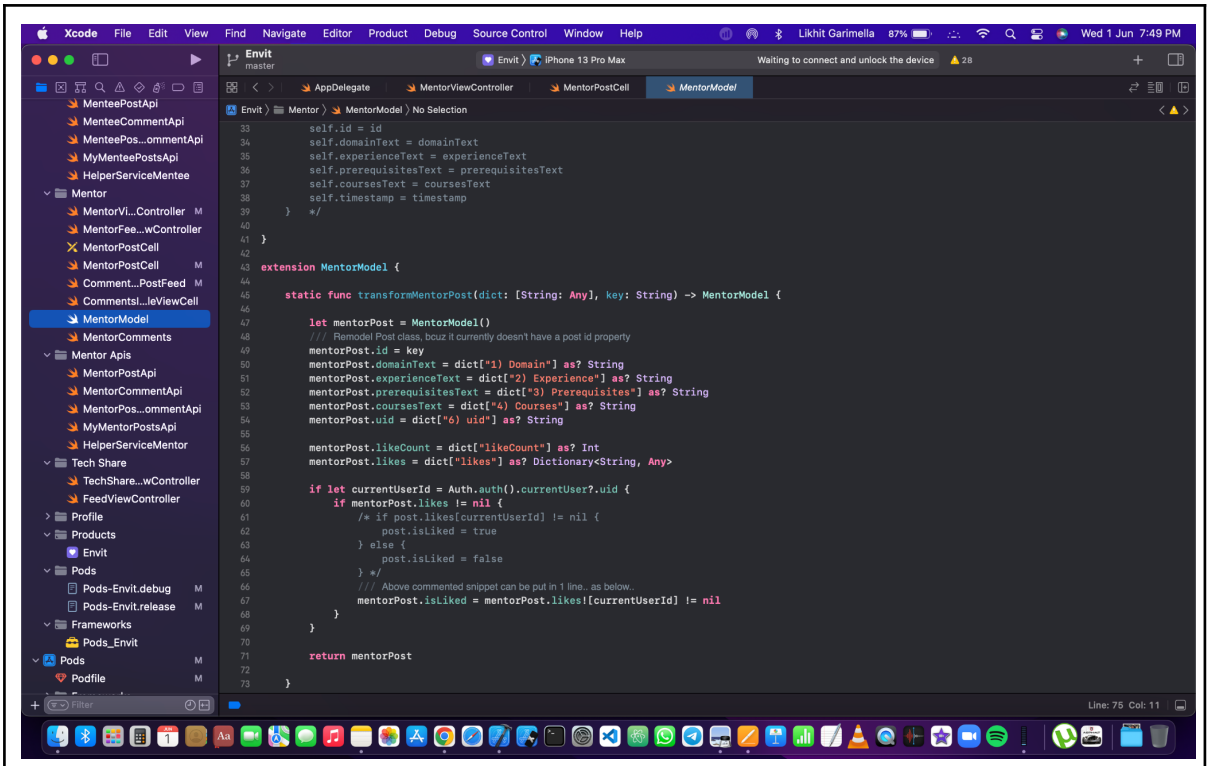
```
func observeUsers(completion: @escaping (AppUser) -> Void) {

    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' exclusing 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)
```

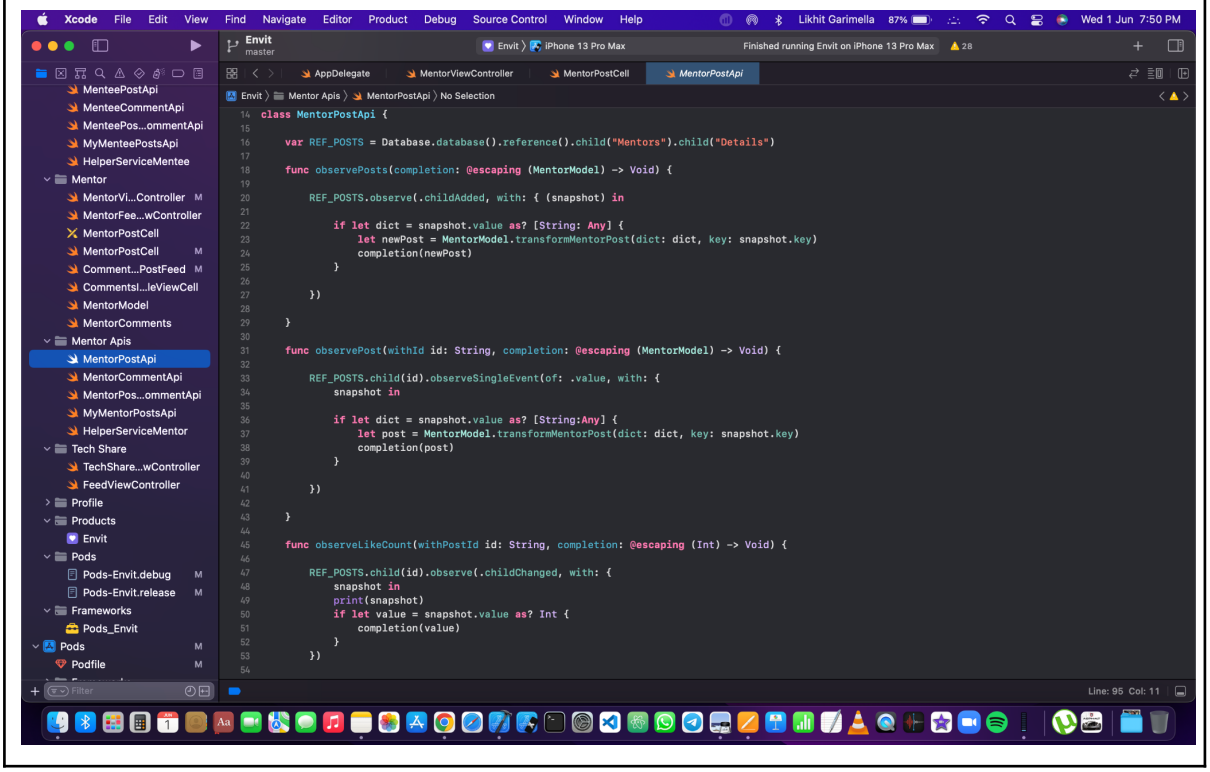


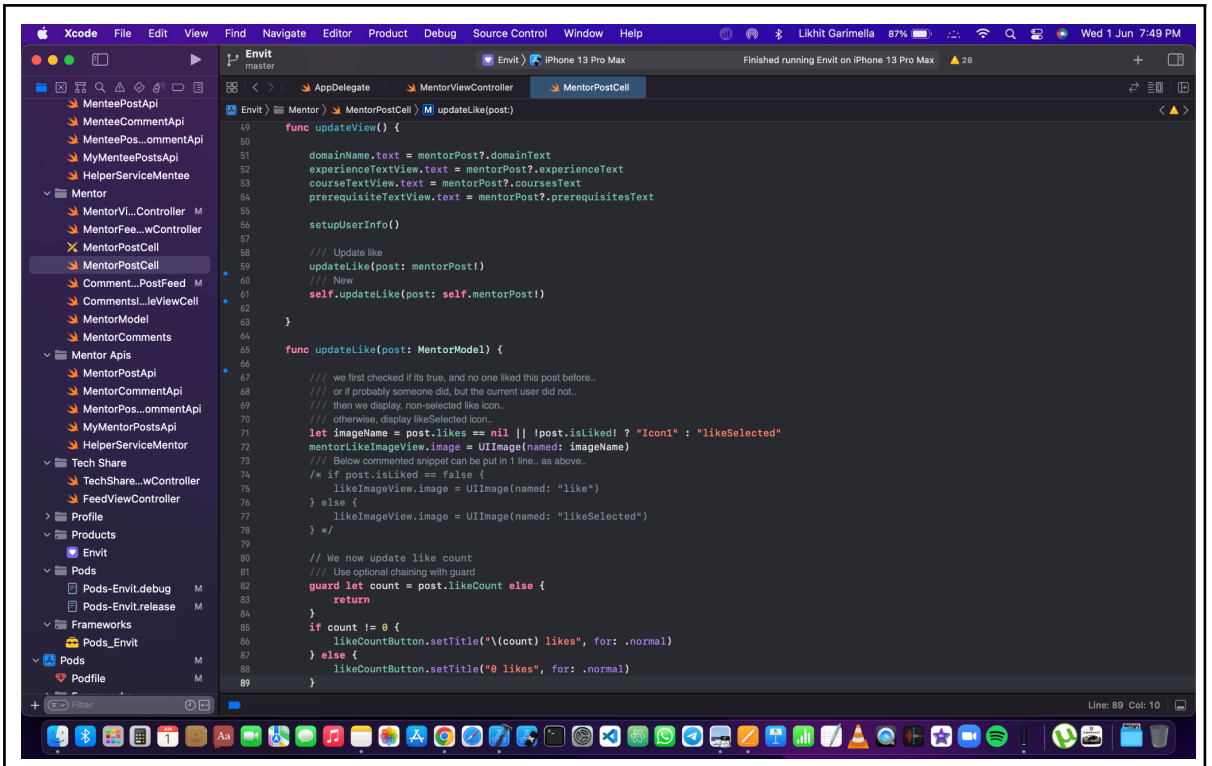
```
// Outlets
@IBOutlet var goToFeedOutlet: UIButton!
@IBOutlet var mentorOutlet: UIButton!
@IBOutlet var menteeOutlet: UIButton!

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)

}
// #52
```

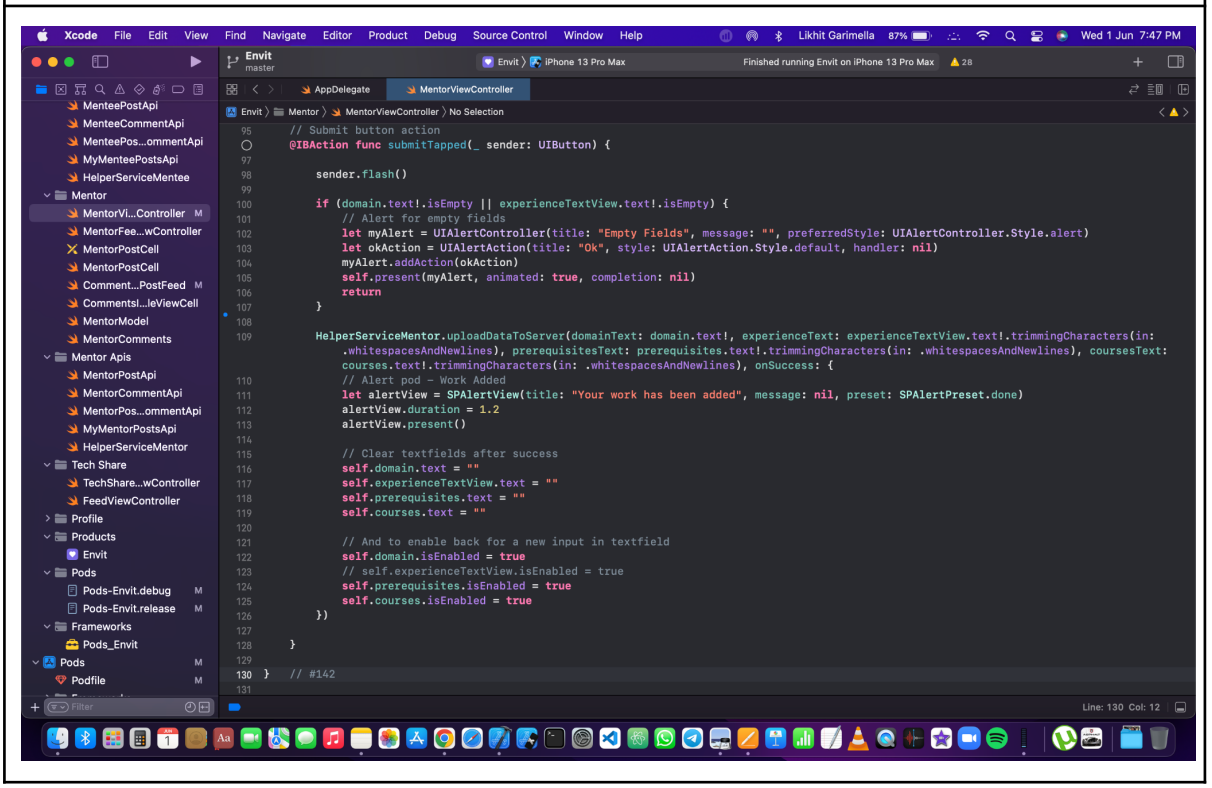

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Likhit Garimella 87% Wed 1 Jun 7:47 PM
Envit master
Envit 〉 iPhone 13 Pro Max
Finished running Envit on iPhone 13 Pro Max
AppDelegate
MentorViewController
MentorFeedViewController
Envit 〉 Mentor 〉 MentorFeedViewController 〉 No Selection
MenteePostApi
MenteeCommentApi
MenteePos...ommentApi
MyMenteePostsApi
HelperServiceMentee
Mentor
MentorVi...Controller M
MentorFee...wController
MentorPostCell
MentorPostCell
Comment...PostFeed M
CommentsI...leViewCell
MentorModel
MentorComments
Mentor Apis
MentorPostApi
MentorCommentApi
MentorPos...ommentApi
MyMentorPostsApi
HelperServiceMentor
Tech Share
TechShare...wController
FeedViewController
Profile
Products
Envit
Pods
Pods-Envit.debug M
Pods-Envit.release M
Frameworks
Pods_Envit
Pods M
Podfile M
Line: 113 Col: 12

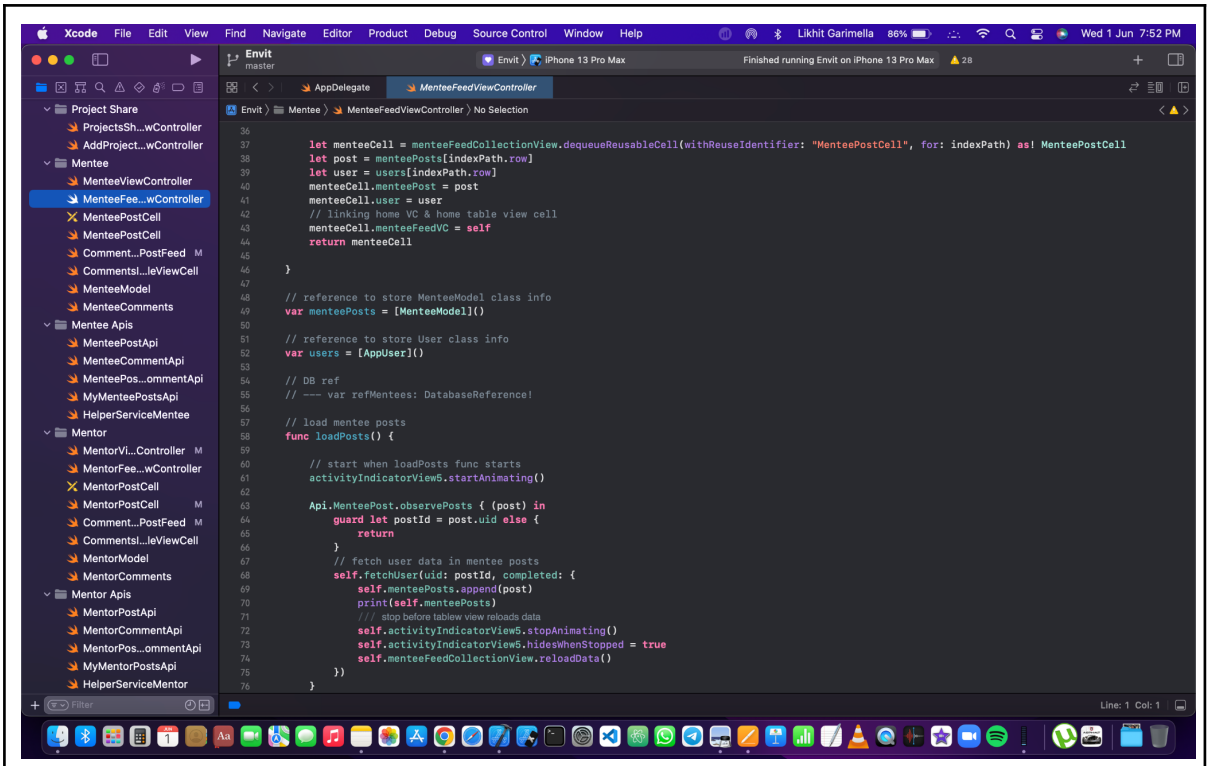

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Likhit Garimella 87% Wed 1 Jun 7:50 PM
Envit master
Envit 〉 iPhone 13 Pro Max
Finished running Envit on iPhone 13 Pro Max
AppDelegate
MentorViewController
MentorPostCell
HelperServiceMentor
Envit 〉 Mentor Apis 〉 HelperServiceMentor 〉 No Selection
MenteePostApi
MenteeCommentApi
MenteePos...ommentApi
MyMenteePostsApi
HelperServiceMentee
Mentor
MentorVi...Controller M
MentorFee...wController
MentorPostCell
MentorPostCell M
Comment...PostFeed M
CommentsI...leViewCell
MentorModel
MentorComments
Mentor Apis
MentorPostApi
MentorCommentApi
MentorPos...ommentApi
MyMentorPostsApi
HelperServiceMentor
Tech Share
TechShare...wController
FeedViewController
Profile
Products
Envit
Pods
Pods-Envit.debug M
Pods-Envit.release M
Frameworks
Pods_Envit
Pods M
Podfile M
Line: 1 Col: 1


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Likhit Garimella 87% Wed 1 Jun 7:49 PM
Envit
master
Envit 〉 iPhone 13 Pro Max
Waiting to connect and unlock the device
AppDelegate
MentorViewController
MentorPostCell
MentorModel
Envit 〉 Mentor 〉 MentorModel 〉 No Selection
MenteePostApi
MenteeCommentApi
MenteePos...ommentApi
MyMenteePostsApi
HelperServiceMentee
Mentor
MentorVi...Controller
MentorFee...wController
MentorPostCell
MentorPostCell
Comment...PostFeed
CommentsI...leViewCell
MentorModel
MentorComments
Mentor Apis
MentorPostApi
MentorCommentApi
MentorPos...ommentApi
MyMentorPostsApi
HelperServiceMentor
Tech Share
TechShare...wController
FeedViewController
Profile
Products
Envit
Pods
Pods-Envit.debug
Pods-Envit.release
Frameworks
Pods_Envit
Pods
Podfile
Line: 75 Col: 11

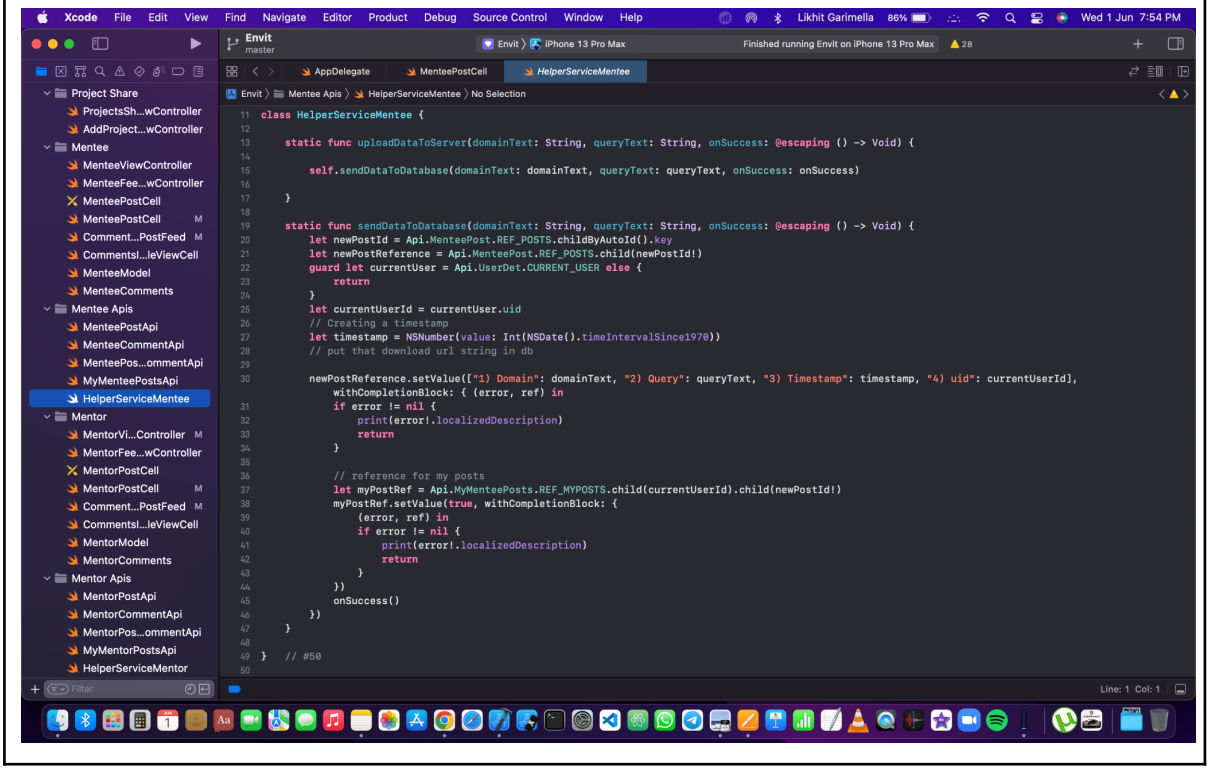

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Likhit Garimella 87% Wed 1 Jun 7:50 PM
Envit
master
Envit 〉 iPhone 13 Pro Max
Finished running Envit on iPhone 13 Pro Max
AppDelegate
MentorViewController
MentorPostCell
MentorPostApi
Envit 〉 Mentor Apis 〉 MentorPostApi 〉 No Selection
MenteePostApi
MenteeCommentApi
MenteePos...ommentApi
MyMenteePostsApi
HelperServiceMentee
Mentor
MentorVi...Controller
MentorFee...wController
MentorPostCell
MentorPostCell
Comment...PostFeed
CommentsI...leViewCell
MentorModel
MentorComments
Mentor Apis
MentorPostApi
MentorCommentApi
MentorPos...ommentApi
MyMentorPostsApi
HelperServiceMentor
Tech Share
TechShare...wController
FeedViewController
Profile
Products
Envit
Pods
Pods-Envit.debug
Pods-Envit.release
Frameworks
Pods_Envit
Pods
Podfile
Line: 95 Col: 11


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)
}

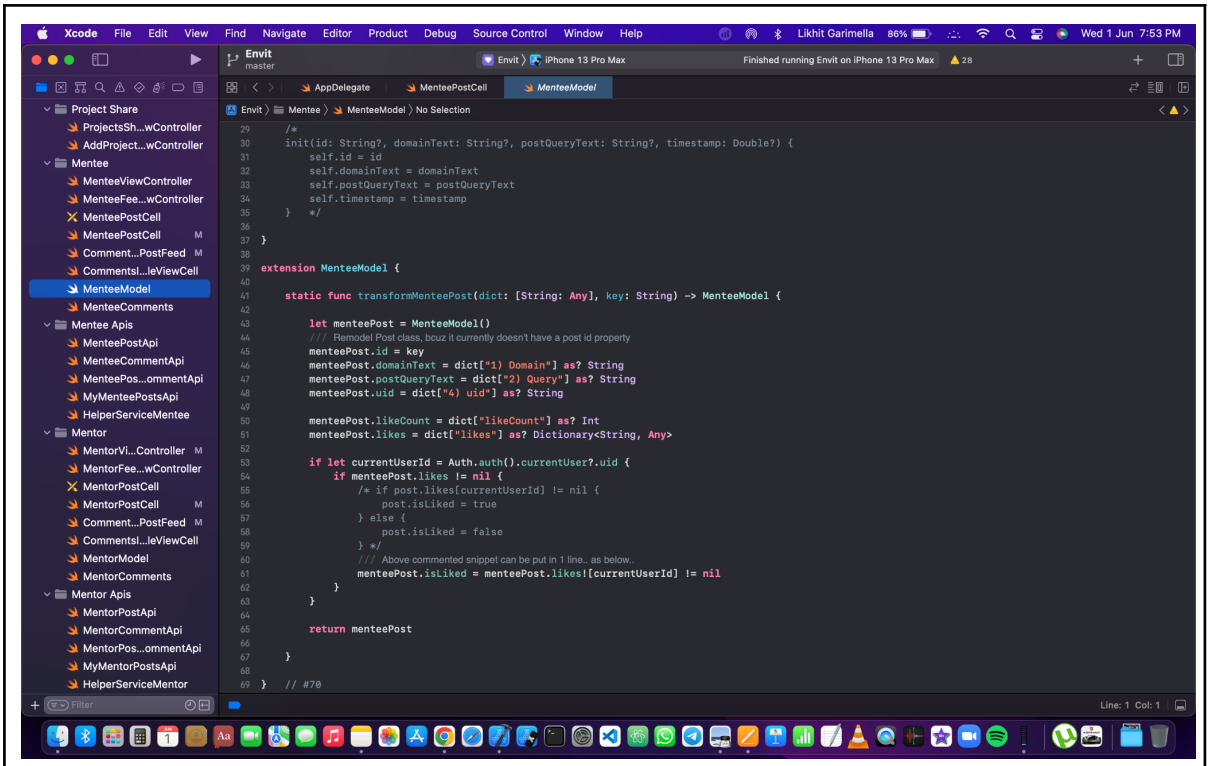

// Submit button action
@IBAction func submitTapped(_ sender: UIButton) {
sender.flash()
if (domain.text!.isEmpty || experienceTextView.text!.isEmpty) {
// Alert for empty fields
let myAlert = UIAlertController(title: "Empty Fields", message: "", preferredStyle: UIAlertController.Style.alert)
let okAction = 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: prerequisites.text!.trimmingCharacters(in: .whitespacesAndNewlines), coursesText:
courses.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
})
}
} // #142

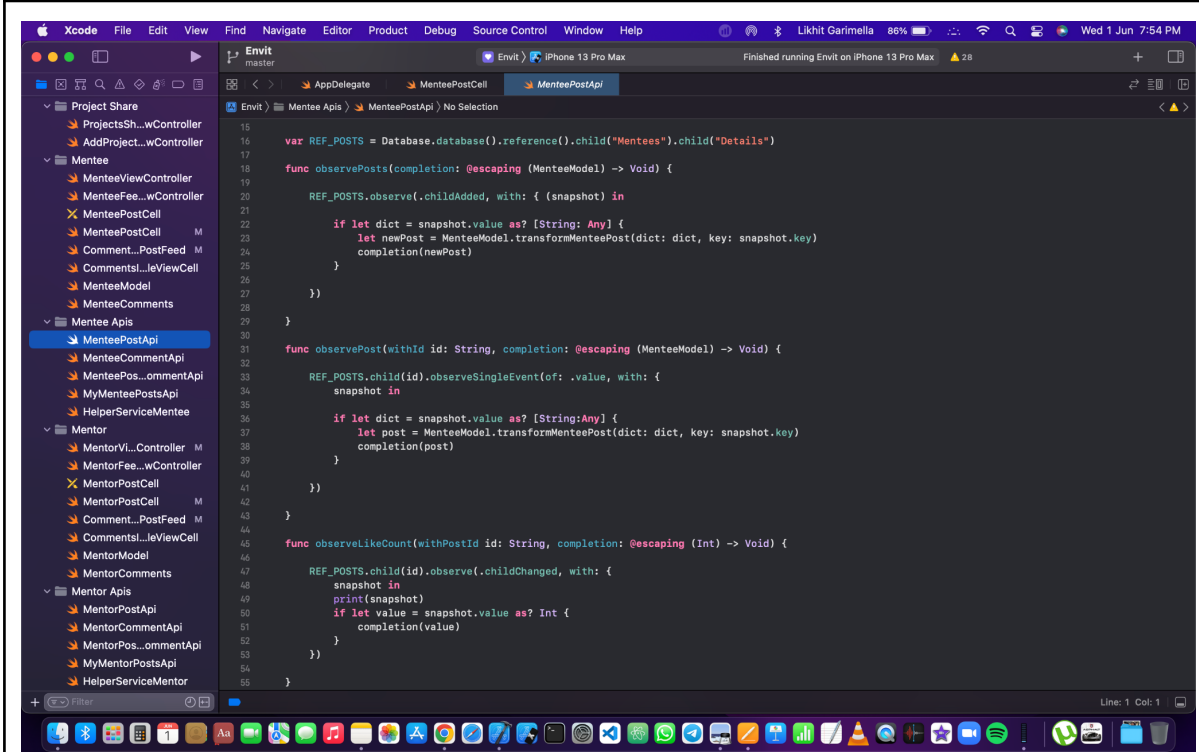


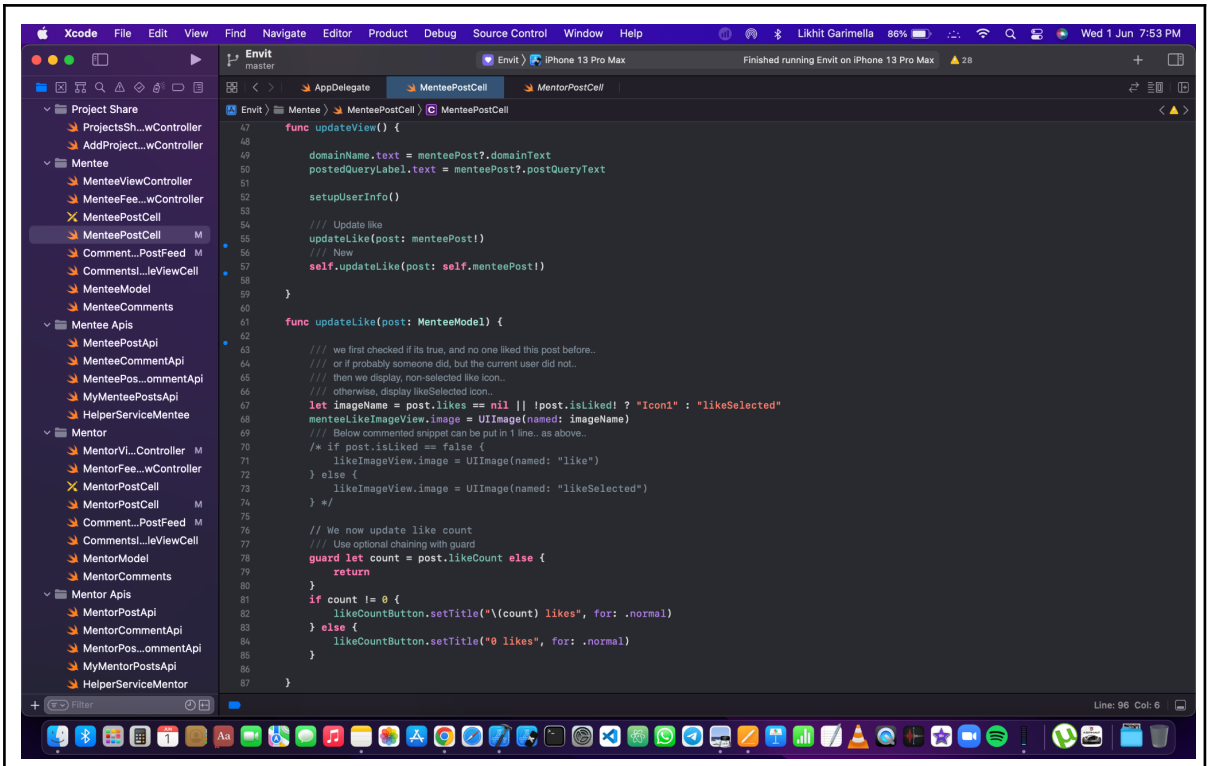

```
36
37          let menteeCell = menteeFeedCollectionView.dequeueReusableCell(withReuseIdentifier: "MenteePostCell", for: indexPath) as! MenteePostCell
38          let post = menteePosts[indexPath.row]
39          let user = users[indexPath.row]
40          menteeCell.menteePost = post
41          menteeCell.user = user
42          // linking home VC & home table view cell
43          menteeCell.menteeFeedVC = self
44          return menteeCell
45
46      }
47
48      // reference to store MenteeModel class info
49      var menteePosts = [MenteeModel]()
50
51      // reference to store User class info
52      var users = [AppUser]()
53
54      // DB ref
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 tablew view reloads data
72                  self.activityIndicatorView5.stopAnimating()
73                  self.activityIndicatorView5.hidesWhenStopped = true
74                  self.menteeFeedCollectionView.reloadData()
75              })
76          }
```



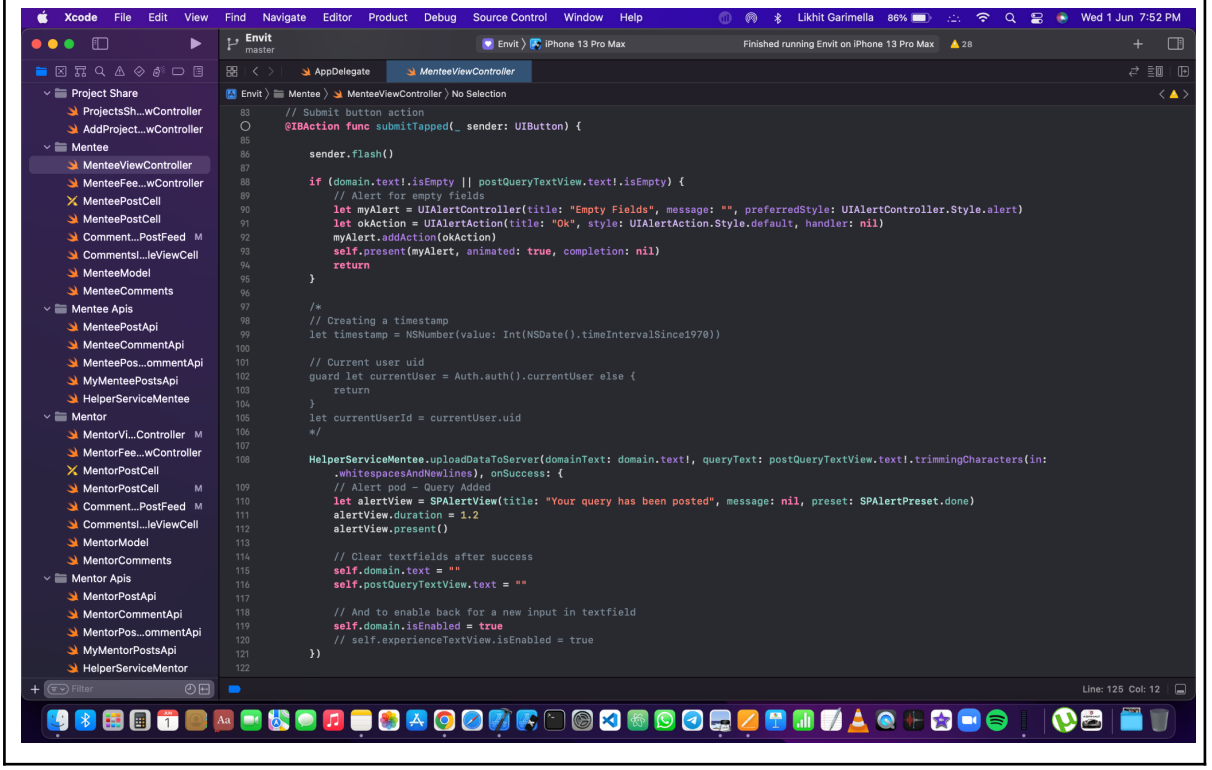

```
11  class HelperServiceMentee {
12
13      static func uploadDataToServer(domainText: String, queryText: String, onSuccess: @escaping () -> Void) {
14
15          self.sendDataToDatabase(domainText: domainText, queryText: queryText, onSuccess: onSuccess)
16
17      }
18
19      static func sendDataToDatabase(domainText: String, queryText: String, onSuccess: @escaping () -> Void) {
20          let newPostId = Api.MenteePost.REF_POSTS.childByAutoId().key
21          let newPostReference = Api.MenteePost.REF_POSTS.child(newPostId!)
22          guard let currentUser = Api.UserDet.CURRENT_USER else {
23              return
24          }
25          let currentUserId = currentUser.uid
26          // Creating a timestamp
27          let timestamp = NSNumber(value: Int(NSDate().timeIntervalSince1970))
28          // put that download url string in db
29
30          newPostReference.setValue(["1) Domain": domainText, "2) Query": queryText, "3) Timestamp": timestamp, "4) uid": currentUserId],
                withCompletionBlock: { (error, ref) in
31              if error != nil {
32                  print(error!.localizedDescription)
33                  return
34              }
35
36              // reference for my posts
37              let myPostRef = Api.MyMenteePosts.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 File Edit View Find Navigate Editor Product Debug Source Control Window Help
Likhit Garimella 86% Wed 1 Jun 7:53 PM
Envit
master
Envit iPhone 13 Pro Max
Finished running Envit on iPhone 13 Pro Max
AppDelegate
MenteePostCell
MenteeModel
Envit Mentee MenteeModel No Selection
Project Share
ProjectsSh...wController
AddProject...wController
Mentee
MenteeViewController
MenteeFee...wController
MenteePostCell
MenteePostCell
Comment...PostFeed
CommentsI...leViewCell
MenteeModel
MenteeComments
Mentee Apis
MenteePostApi
MenteeCommentApi
MenteePos...ommentApi
MyMenteePostsApi
HelperServiceMentee
Mentor
MentorVi...Controller
MentorFee...wController
MentorPostCell
MentorPostCell
Comment...PostFeed
CommentsI...leViewCell
MentorModel
MentorComments
Mentor Apis
MentorPostApi
MentorCommentApi
MentorPos...ommentApi
MyMentorPostsApi
HelperServiceMentor
/*
init(id: String?, domainText: String?, postQueryText: String?, timestamp: Double?) {
self.id = id
self.domainText = domainText
self.postQueryText = postQueryText
self.timestamp = timestamp
} */
}
extension MenteeModel {
static func transformMenteePost(dict: [String: Any], key: String) -> MenteeModel {
let menteePost = MenteeModel()
/// Remodel Post class, bcuz it currently doesn't have a post id property
menteePost.id = key
menteePost.domainText = dict["1) Domain"] as? String
menteePost.postQueryText = dict["2) Query"] as? String
menteePost.uid = dict["4) uid"] as? String
menteePost.likeCount = dict["likeCount"] as? Int
menteePost.likes = dict["likes"] as? Dictionary<String, Any>
if let currentUserId = Auth.auth().currentUser?.uid {
if menteePost.likes != nil {
/* if post.likes[currentUserId] != nil {
post.isLiked = true
} else {
post.isLiked = false
} */
/// Above commented snippet can be put in 1 line.. as below..
menteePost.isLiked = menteePost.likes![currentUserId] != nil
}
}
return menteePost
}
} // #70
Line: 1 Col: 1

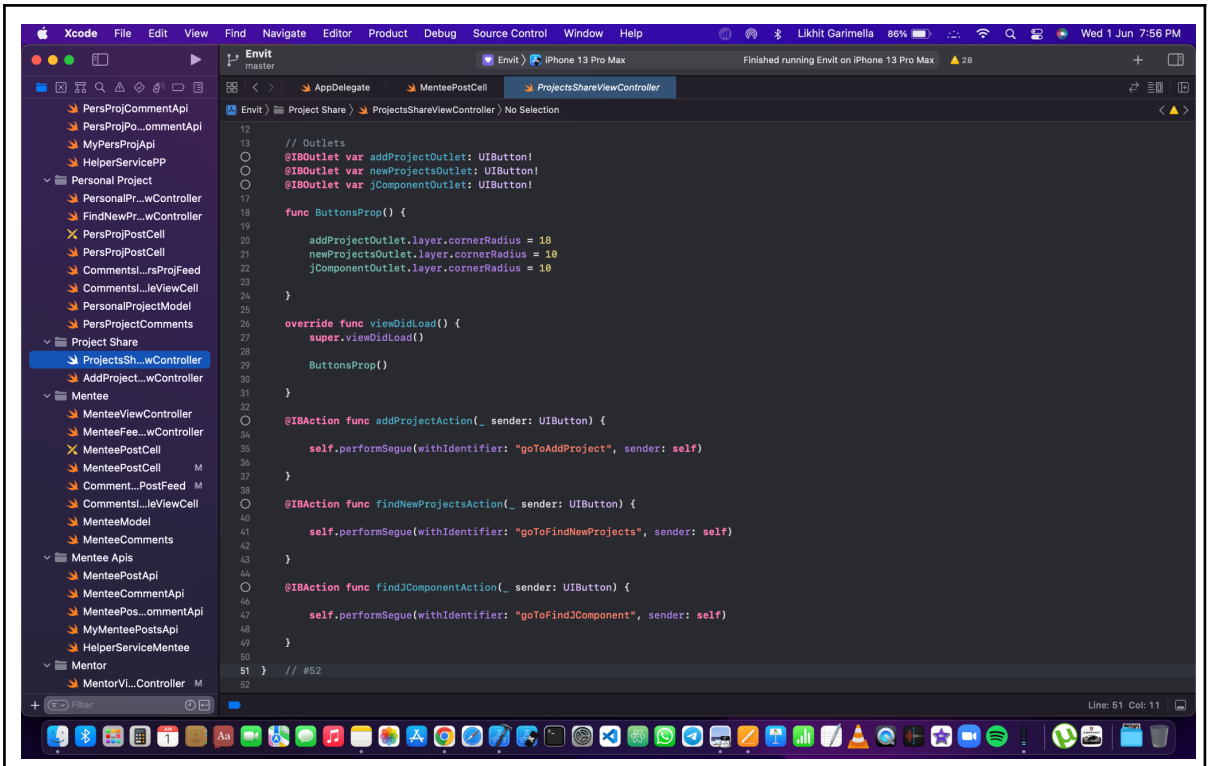

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Likhit Garimella 86% Wed 1 Jun 7:54 PM
Envit
master
Envit iPhone 13 Pro Max
Finished running Envit on iPhone 13 Pro Max
AppDelegate
MenteePostCell
MenteePostApi
Envit Mentee Apis MenteePostApi No Selection
Project Share
ProjectsSh...wController
AddProject...wController
Mentee
MenteeViewController
MenteeFee...wController
MenteePostCell
MenteePostCell
Comment...PostFeed
CommentsI...leViewCell
MenteeModel
MenteeComments
Mentee Apis
MenteePostApi
MenteeCommentApi
MenteePos...ommentApi
MyMenteePostsApi
HelperServiceMentee
Mentor
MentorVi...Controller
MentorFee...wController
MentorPostCell
MentorPostCell
Comment...PostFeed
CommentsI...leViewCell
MentorModel
MentorComments
Mentor Apis
MentorPostApi
MentorCommentApi
MentorPos...ommentApi
MyMentorPostsApi
HelperServiceMentor
var REF_POSTS = Database.database().reference().child("Mentees").child("Details")
func observePosts(completion: @escaping (MenteeModel) -> Void) {
REF_POSTS.observe(.childAdded, with: { (snapshot) in
if let dict = snapshot.value as? [String: Any] {
let newPost = MenteeModel.transformMenteePost(dict: dict, key: snapshot.key)
completion(newPost)
}
})
}
func observePost(withId id: String, completion: @escaping (MenteeModel) -> Void) {
REF_POSTS.child(id).observeSingleEvent(of: .value, with: {
snapshot in
if let dict = snapshot.value as? [String:Any] {
let post = MenteeModel.transformMenteePost(dict: dict, key: snapshot.key)
completion(post)
}
})
}
func observeLikeCount(withPostId id: String, completion: @escaping (Int) -> Void) {
REF_POSTS.child(id).observe(.childChanged, with: {
snapshot in
print(snapshot)
if let value = snapshot.value as? Int {
completion(value)
}
})
}
Line: 1 Col: 1

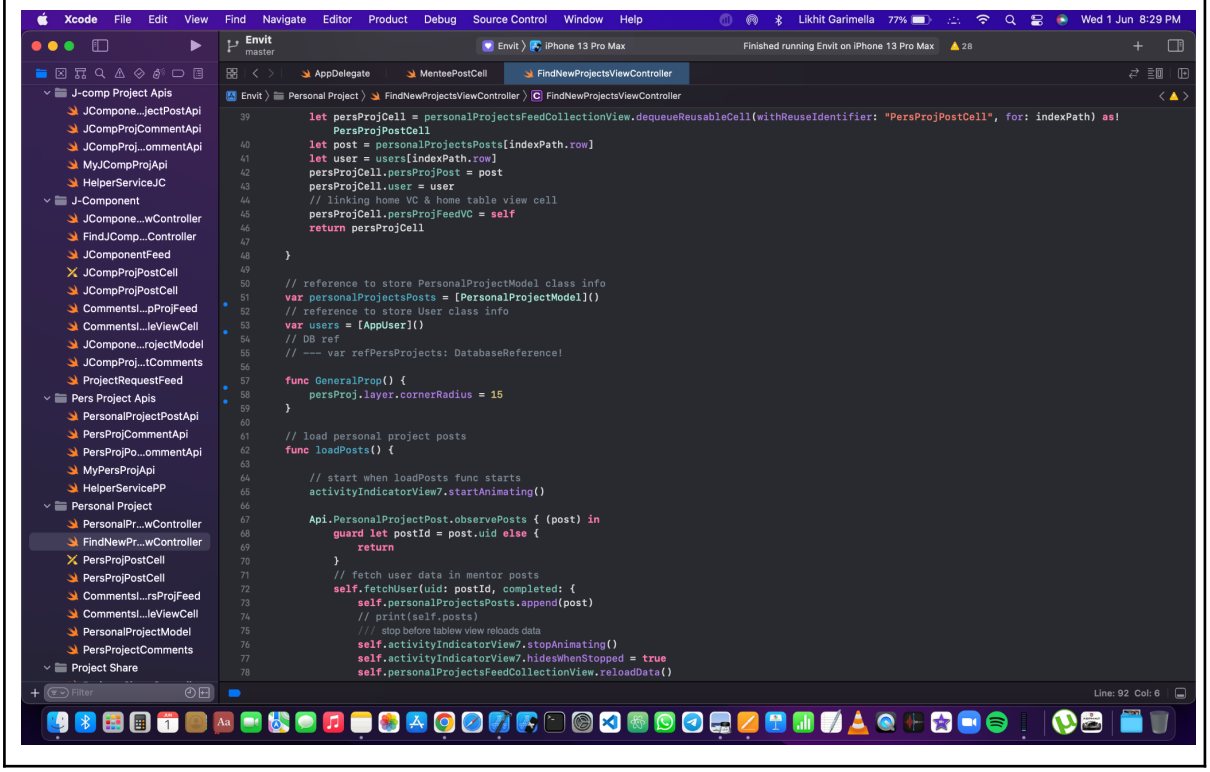


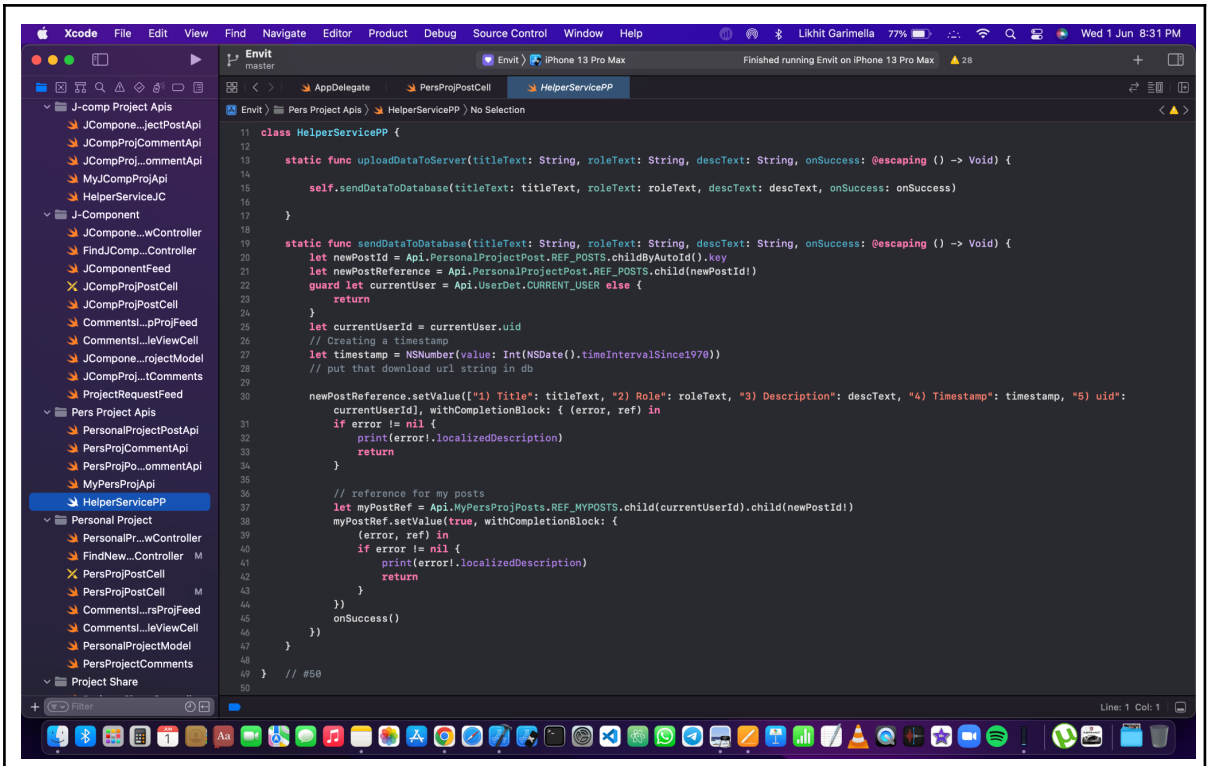
```
func updateView() {

    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)
    }

}
```



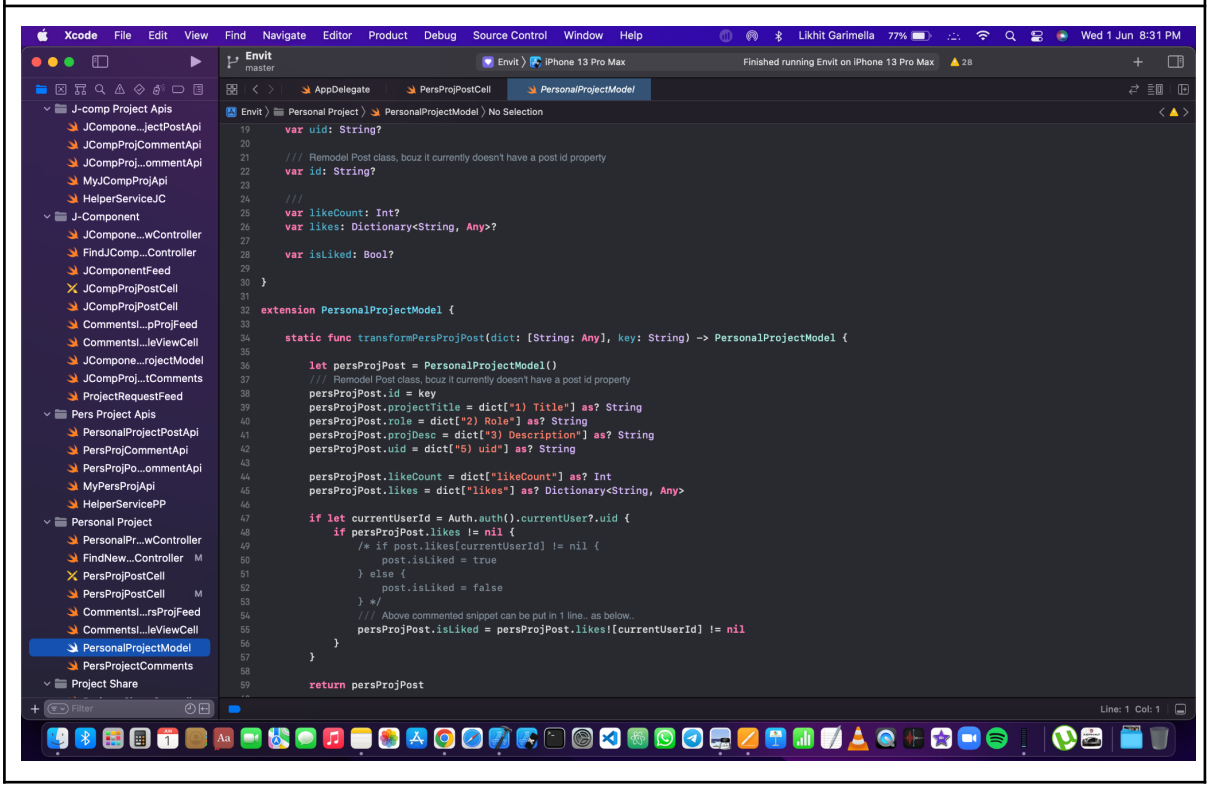
```
// Submit button action
@IBAction func submitTapped(_ sender: UIButton) {

    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 = NSNumber(value: Int(NSDate().timeIntervalSince1970))

    // 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 = SPAlertView(title: "Your query has been posted", message: nil, preset: SPAlertPreset.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
    })
```


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Likhit Garimella 86% Wed 1 Jun 7:56 PM
Envit master
Envit iPhone 13 Pro Max Finished running Envit on iPhone 13 Pro Max
AppDelegate MenteePostCell ProjectsShareViewController
Envit Project Share ProjectsShareViewController No Selection
PersProjCommentApi
PersProjPo...ommentApi
MyPersProjApi
HelperServicePP
Personal Project
PersonalPr...wController
FindNewPr...wController
PersProjPostCell
PersProjPostCell
CommentsI...rsProjFeed
CommentsI...leViewCell
PersonalProjectModel
PersProjectComments
Project Share
ProjectsSh...wController
AddProject...wController
Mentee
MenteeViewController
MenteeFee...wController
MenteePostCell
MenteePostCell M
Comment...PostFeed M
CommentsI...leViewCell
MenteeModel
MenteeComments
Mentee Apis
MenteePostApi
MenteeCommentApi
MenteePos...ommentApi
MyMenteePostsApi
HelperServiceMentee
Mentor
MentorVi...Controller M
// Outlets
@IBOutlet var addProjectOutlet: UIButton!
@IBOutlet var newProjectsOutlet: UIButton!
@IBOutlet var jComponentOutlet: UIButton!
func ButtonsProp() {
addProjectOutlet.layer.cornerRadius = 18
newProjectsOutlet.layer.cornerRadius = 10
jComponentOutlet.layer.cornerRadius = 10
}
override func viewDidLoad() {
super.viewDidLoad()
ButtonsProp()
}
@IBAction func addProjectAction(_ sender: UIButton) {
self.performSegue(withIdentifier: "goToAddProject", sender: self)
}
@IBAction func findNewProjectsAction(_ sender: UIButton) {
self.performSegue(withIdentifier: "goToFindNewProjects", sender: self)
}
@IBAction func findJComponentAction(_ sender: UIButton) {
self.performSegue(withIdentifier: "goToFindJComponent", sender: self)
}
} // #52
Line: 51 Col: 11

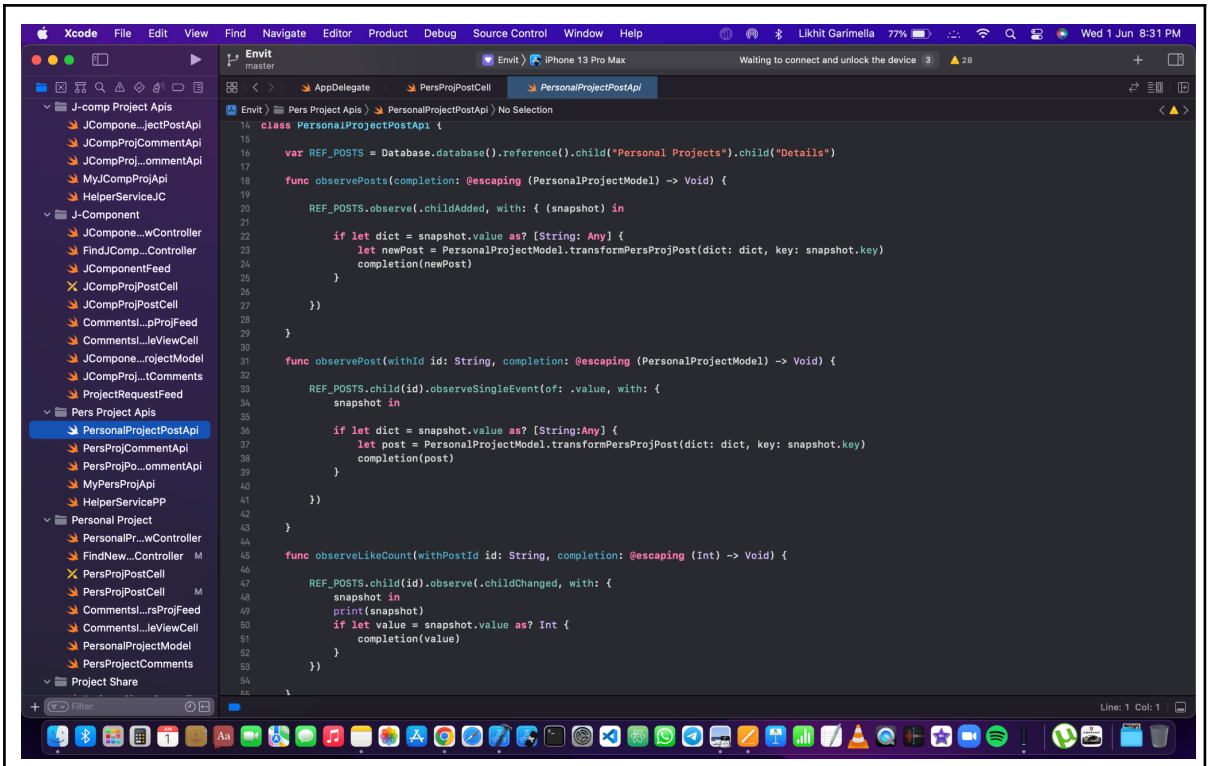

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Likhit Garimella 77% Wed 1 Jun 8:29 PM
Envit master
Envit iPhone 13 Pro Max Finished running Envit on iPhone 13 Pro Max
AppDelegate MenteePostCell FindNewProjectsViewController
Envit Personal Project FindNewProjectsViewController FindNewProjectsViewController
J-comp Project Apis
JCompone...jectPostApi
JCompProjCommentApi
JCompProj...ommentApi
MyJCompProjApi
HelperServiceJC
J-Component
JCompone...wController
FindJComp...Controller
JComponentFeed
JCompProjPostCell
JCompProjPostCell
CommentsI...pProjFeed
CommentsI...leViewCell
JCompone...rojectModel
JCompProj...tComments
ProjectRequestFeed
Pers Project Apis
PersonalProjectPostApi
PersProjCommentApi
PersProjPo...ommentApi
MyPersProjApi
HelperServicePP
Personal Project
PersonalPr...wController
FindNewPr...wController
PersProjPostCell
PersProjPostCell
CommentsI...rsProjFeed
CommentsI...leViewCell
PersonalProjectModel
PersProjectComments
Project Share
let persProjCell = personalProjectsFeedCollectionView.dequeueReusableCell(withReuseIdentifier: "PersProjPostCell", for: indexPath) as!
PersProjPostCell
let post = personalProjectsPosts[indexPath.row]
let user = users[indexPath.row]
persProjCell.persProjPost = post
persProjCell.user = user
// linking home VC & home table view cell
persProjCell.persProjFeedVC = self
return persProjCell
}
// reference to store PersonalProjectModel class info
var personalProjectsPosts = [PersonalProjectModel]()
// reference to store User class info
var users = [AppUser]()
// DB ref
// --- var refPersProjects: DatabaseReference!
func GeneralProp() {
persProj.layer.cornerRadius = 15
}
// load personal project posts
func loadPosts() {
// start when loadPosts func starts
activityIndicatorView7.startAnimating()
Api.PersonalProjectPost.observePosts { (post) in
guard let postId = post.uid else {
return
}
// fetch user data in mentor posts
self.fetchUser(uid: postId, completed: {
self.personalProjectsPosts.append(post)
// print(self.posts)
/// stop before tablew view reloads data
self.activityIndicatorView7.stopAnimating()
self.activityIndicatorView7.hidesWhenStopped = true
self.personalProjectsFeedCollectionView.reloadData()
Line: 92 Col: 6



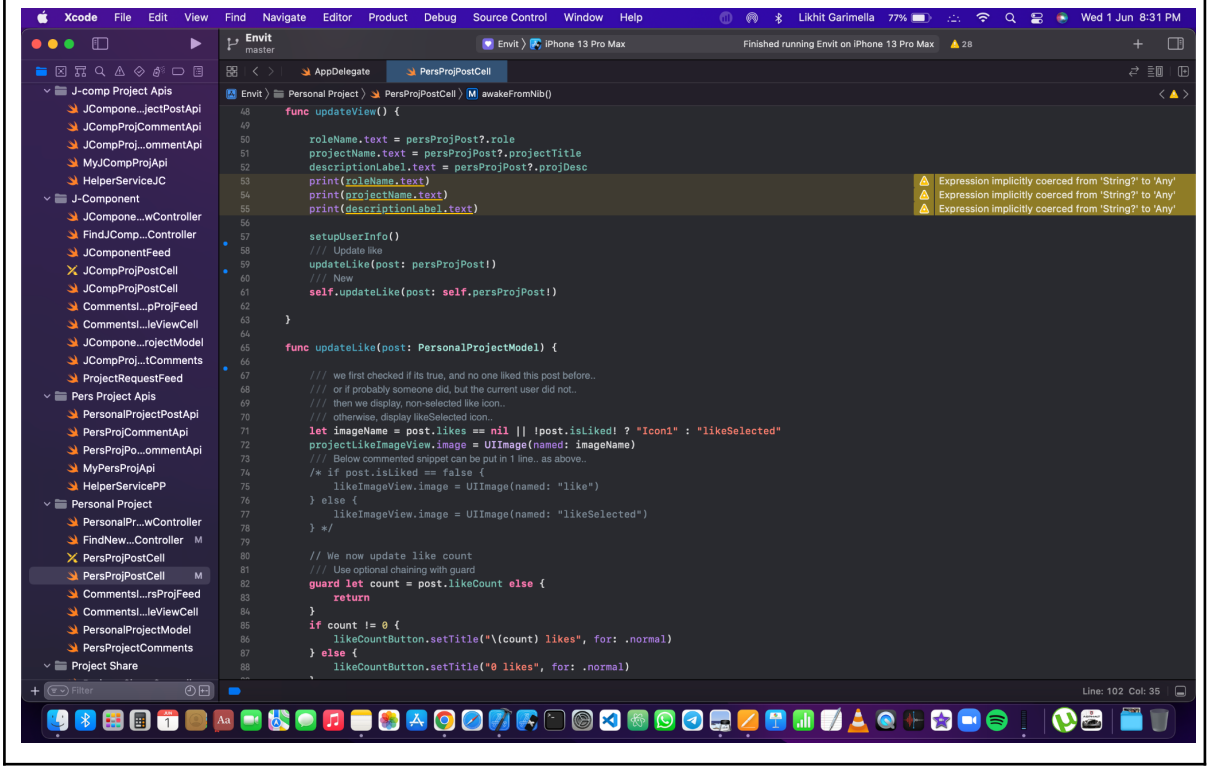
```
class HelperServicePP {

    static func uploadDataToServer(titleText: String, roleText: String, descText: String, onSuccess: @escaping () -> Void) {

        self.sendDataToDatabase(titleText: titleText, roleText: roleText, descText: descText, onSuccess: onSuccess)

    }

    static func sendDataToDatabase(titleText: String, roleText: String, descText: String, onSuccess: @escaping () -> Void) {
        let newPostId = Api.PersonalProjectPost.REF_POSTS.childByAutoId().key
        let newPostReference = Api.PersonalProjectPost.REF_POSTS.child(newPostId!)
        guard let currentUser = Api.UserDet.CURRENT_USER else {
            return
        }
        let currentUserId = currentUser.uid
        // Creating a timestamp
        let timestamp = NSNumber(value: Int(NSDate().timeIntervalSince1970))
        // put that download url string in db

        newPostReference.setValue(["1) Title": titleText, "2) Role": roleText, "3) Description": descText, "4) Timestamp": timestamp, "5) uid":
            currentUserId], withCompletionBlock: { (error, ref) in
            if error != nil {
                print(error!.localizedDescription)
                return
            }

            // reference for my posts
            let myPostRef = Api.MyPersProjPosts.REF_MYPOSTS.child(currentUserId).child(newPostId!)
            myPostRef.setValue(true, withCompletionBlock: {
                (error, ref) in
                if error != nil {
                    print(error!.localizedDescription)
                    return
                }
            })
            onSuccess()
        })
    }

}   // #50
```



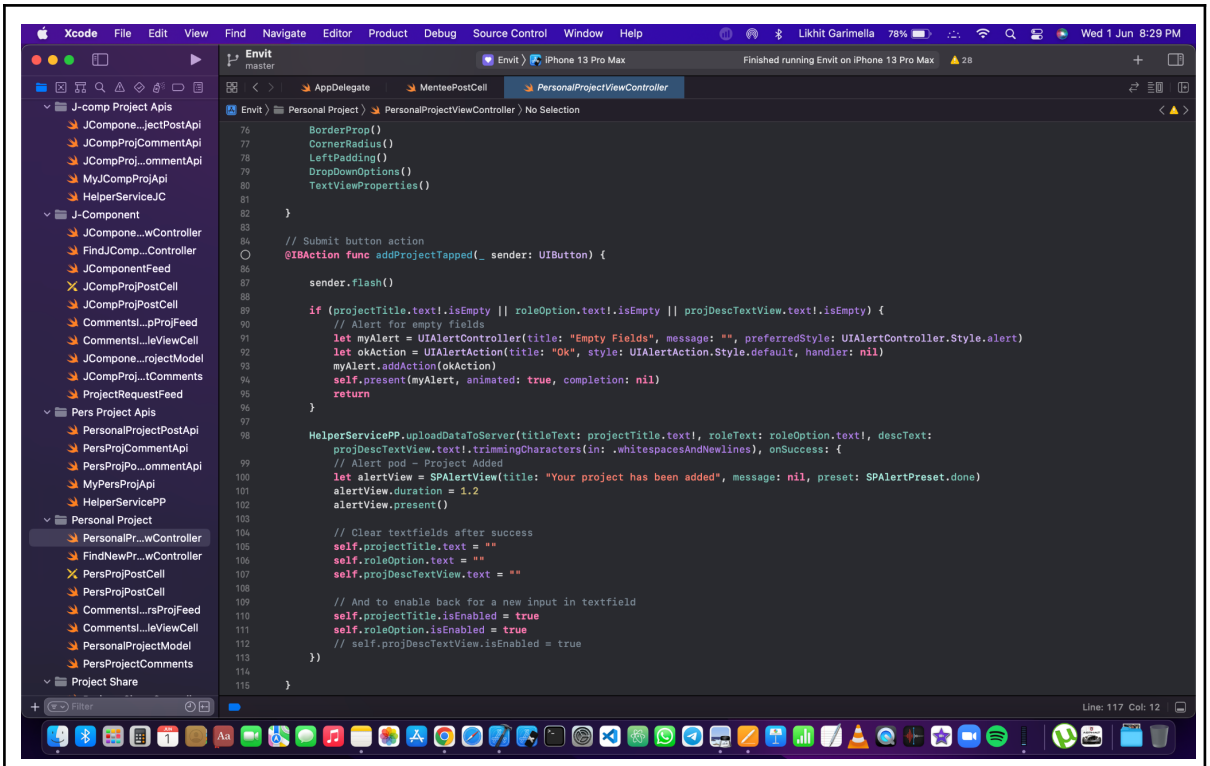
```
    var uid: String?

    /// Remodel Post class, bcuz it currently doesn't have a post id property
    var id: String?

    ///
    var likeCount: Int?
    var likes: Dictionary<String, Any>?

    var isLiked: Bool?

}

extension PersonalProjectModel {

    static func transformPersProjPost(dict: [String: Any], key: String) -> PersonalProjectModel {

        let persProjPost = PersonalProjectModel()
        /// Remodel Post class, bcuz it currently doesn't have a post id property
        persProjPost.id = key
        persProjPost.projectTitle = dict["1) Title"] as? String
        persProjPost.role = dict["2) Role"] as? String
        persProjPost.projDesc = dict["3) Description"] as? String
        persProjPost.uid = dict["5) uid"] as? String

        persProjPost.likeCount = dict["likeCount"] as? Int
        persProjPost.likes = dict["likes"] as? Dictionary<String, Any>

        if let currentUserId = Auth.auth().currentUser?.uid {
            if persProjPost.likes != nil {
                /* if post.likes[currentUserId] != nil {
                    post.isLiked = true
                } else {
                    post.isLiked = false
                } */
                /// Above commented snippet can be put in 1 line.. as below..
                persProjPost.isLiked = persProjPost.likes![currentUserId] != nil
            }
        }

        return persProjPost
```

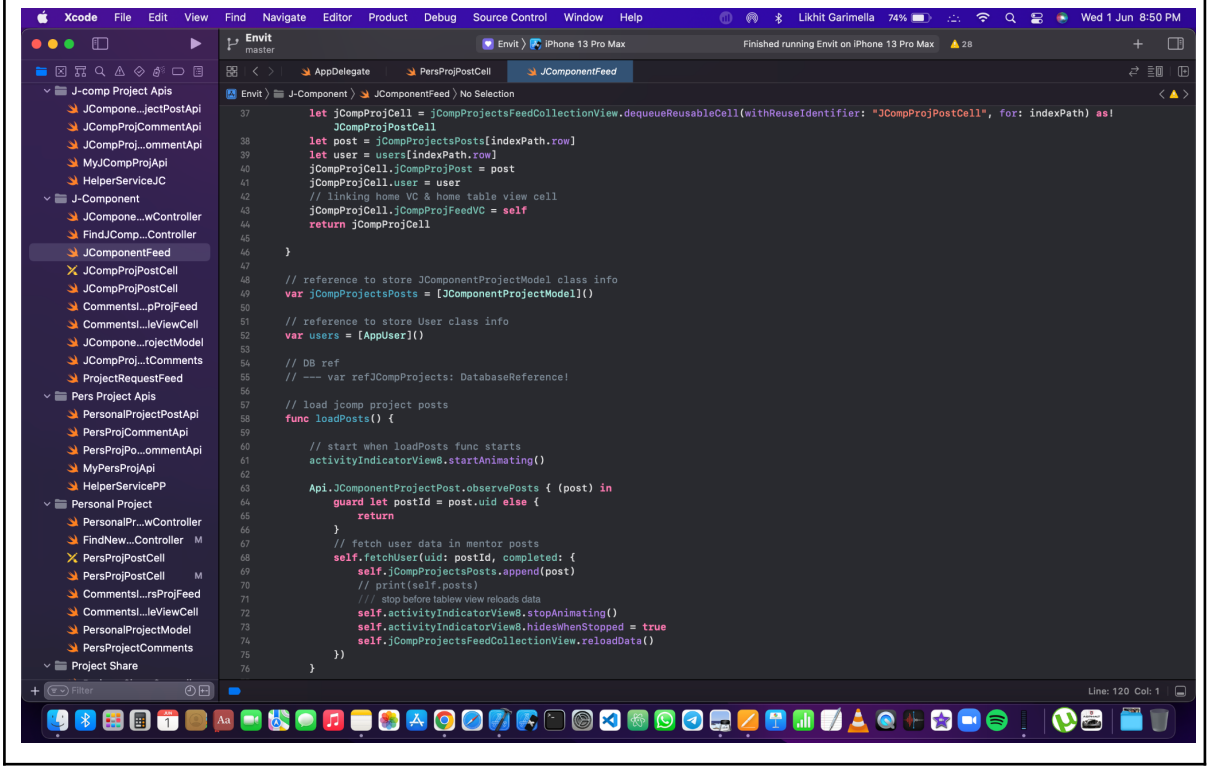


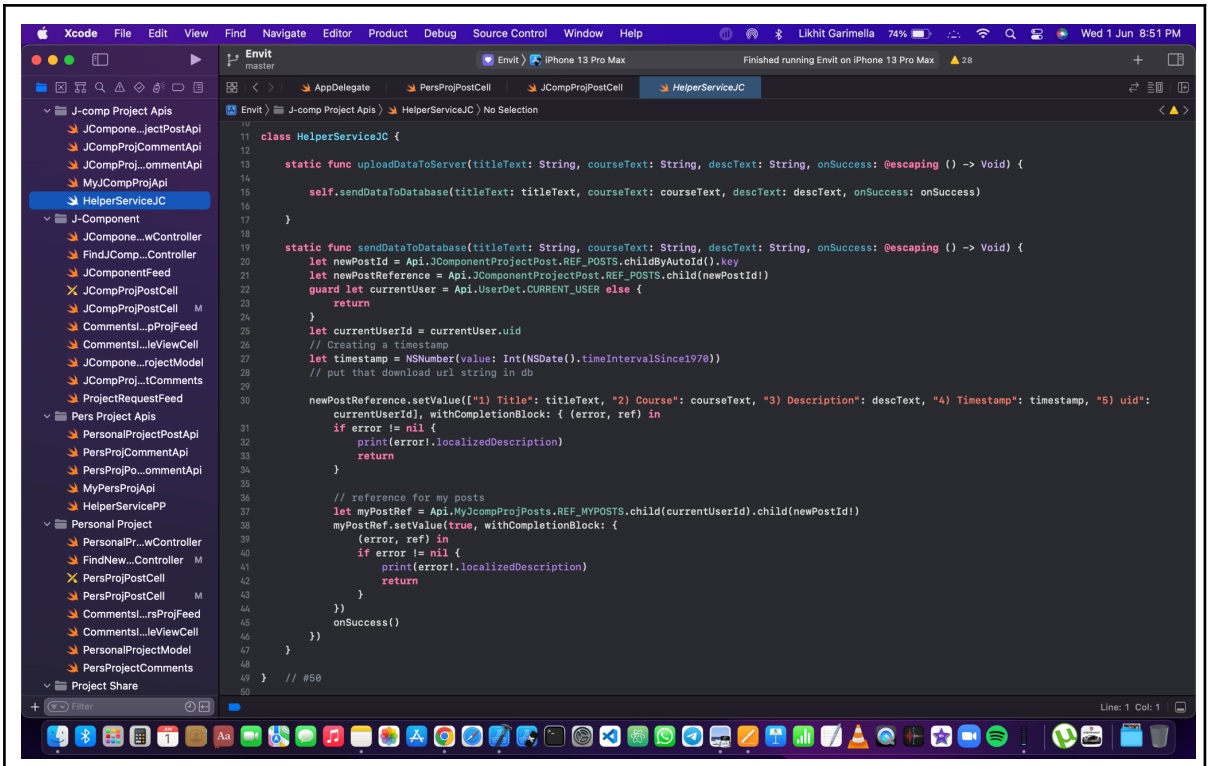
```
class PersonalProjectPostApi {

    var REF_POSTS = Database.database().reference().child("Personal Projects").child("Details")

    func observePosts(completion: @escaping (PersonalProjectModel) -> Void) {

        REF_POSTS.observe(.childAdded, with: { (snapshot) in

            if let dict = snapshot.value as? [String: Any] {
                let newPost = PersonalProjectModel.transformPersProjPost(dict: dict, key: snapshot.key)
                completion(newPost)
            }

        })

    }

    func observePost(withId id: String, completion: @escaping (PersonalProjectModel) -> Void) {

        REF_POSTS.child(id).observeSingleEvent(of: .value, with: {
            snapshot in

            if let dict = snapshot.value as? [String:Any] {
                let post = PersonalProjectModel.transformPersProjPost(dict: dict, key: snapshot.key)
                completion(post)
            }

        })

    }

    func observeLikeCount(withPostId id: String, completion: @escaping (Int) -> Void) {

        REF_POSTS.child(id).observe(.childChanged, with: {
            snapshot in
            print(snapshot)
            if let value = snapshot.value as? Int {
                completion(value)
            }
        })
    }
```



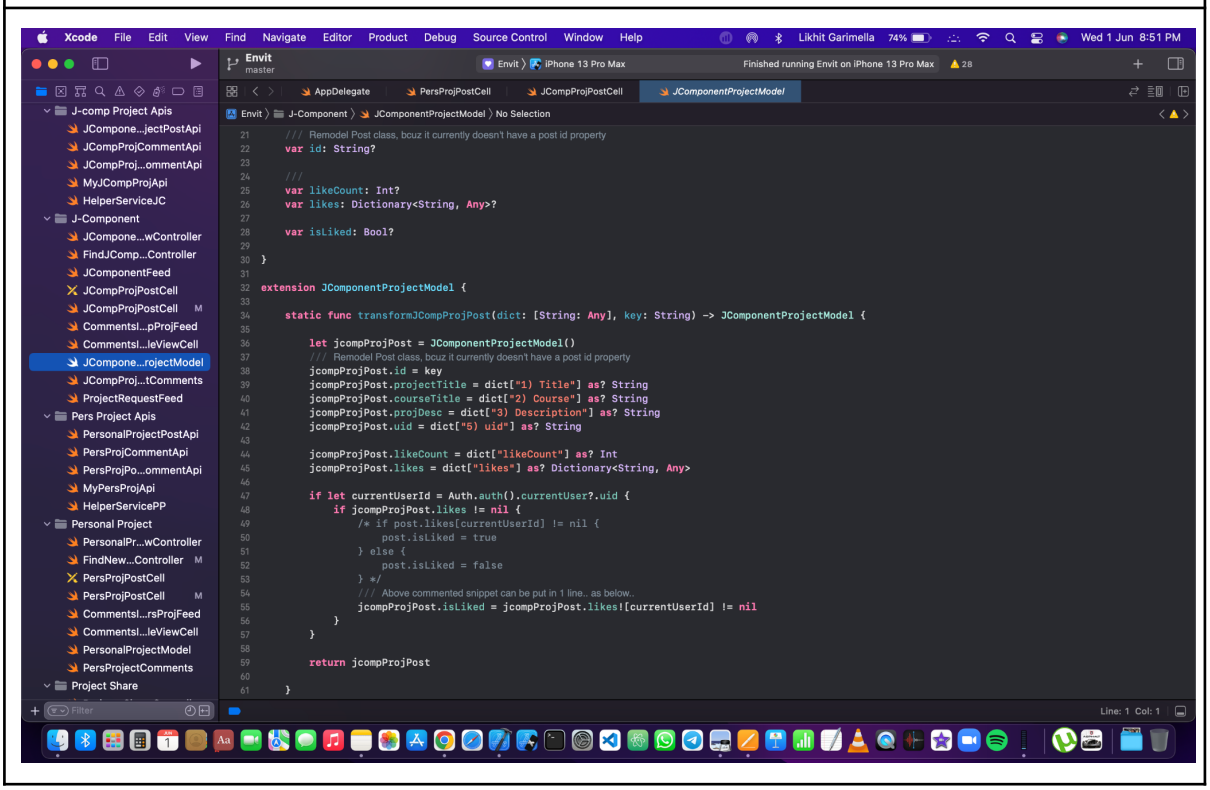
```
    func updateView() {

        roleName.text = persProjPost?.role
        projectName.text = persProjPost?.projectTitle
        descriptionLabel.text = persProjPost?.projDesc
        print(roleName.text)
        print(projectName.text)
        print(descriptionLabel.text)

        setupUserInfo()
        /// Update like
        updateLike(post: persProjPost!)
        /// New
        self.updateLike(post: self.persProjPost!)

    }

    func updateLike(post: PersonalProjectModel) {

        /// 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"
        projectLikeImageView.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)
```


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Likhit Garimella 78% Wed 1 Jun 8:29 PM
Envit
master
Envit iPhone 13 Pro Max
Finished running Envit on iPhone 13 Pro Max
AppDelegate
MenteePostCell
PersonalProjectViewController
Envit Personal Project PersonalProjectViewController No Selection
J-comp Project Apis
JCompone...jectPostApi
JCompProjCommentApi
JCompProj...ommentApi
MyJCompProjApi
HelperServiceJC
J-Component
JCompone...wController
FindJComp...Controller
JComponentFeed
JCompProjPostCell
JCompProjPostCell
CommentsI...pProjFeed
CommentsI...leViewCell
JCompone...rojectModel
JCompProj...tComments
ProjectRequestFeed
Pers Project Apis
PersonalProjectPostApi
PersProjCommentApi
PersProjPo...ommentApi
MyPersProjApi
HelperServicePP
Personal Project
PersonalPr...wController
FindNewPr...wController
PersProjPostCell
PersProjPostCell
CommentsI...rsProjFeed
CommentsI...leViewCell
PersonalProjectModel
PersProjectComments
Project Share
BorderProp()
CornerRadius()
LeftPadding()
DropDownOptions()
TextViewProperties()
}
// Submit button action
@IBAction func addProjectTapped(_ sender: UIButton) {
sender.flash()
if (projectTitle.text!.isEmpty || roleOption.text!.isEmpty || projDescTextView.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
}
HelperServicePP.uploadDataToServer(titleText: projectTitle.text!, roleText: roleOption.text!, descText:
projDescTextView.text!.trimmingCharacters(in: .whitespacesAndNewlines), onSuccess: {
// Alert pod - Project Added
let alertView = SPAlertView(title: "Your project has been added", message: nil, preset: SPAlertPreset.done)
alertView.duration = 1.2
alertView.present()
// Clear textfields after success
self.projectTitle.text = ""
self.roleOption.text = ""
self.projDescTextView.text = ""
// And to enable back for a new input in textfield
self.projectTitle.isEnabled = true
self.roleOption.isEnabled = true
// self.projDescTextView.isEnabled = true
})
}
Line: 117 Col: 12

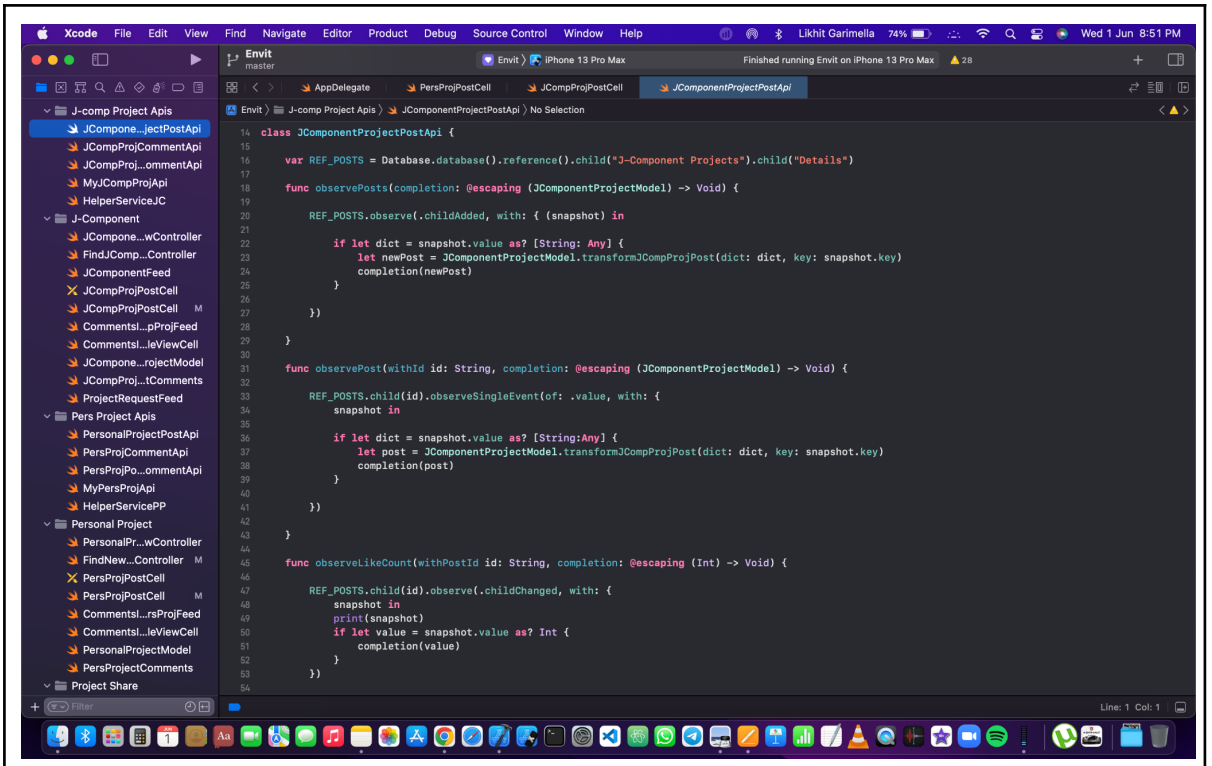

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Likhit Garimella 74% Wed 1 Jun 8:50 PM
Envit
master
Envit iPhone 13 Pro Max
Finished running Envit on iPhone 13 Pro Max
AppDelegate
PersProjPostCell
JComponentFeed
Envit J-Component JComponentFeed No Selection
J-comp Project Apis
JCompone...jectPostApi
JCompProjCommentApi
JCompProj...ommentApi
MyJCompProjApi
HelperServiceJC
J-Component
JCompone...wController
FindJComp...Controller
JComponentFeed
JCompProjPostCell
JCompProjPostCell
CommentsI...pProjFeed
CommentsI...leViewCell
JCompone...rojectModel
JCompProj...tComments
ProjectRequestFeed
Pers Project Apis
PersonalProjectPostApi
PersProjCommentApi
PersProjPo...ommentApi
MyPersProjApi
HelperServicePP
Personal Project
PersonalPr...wController
FindNew...Controller M
PersProjPostCell
PersProjPostCell M
CommentsI...rsProjFeed
CommentsI...leViewCell
PersonalProjectModel
PersProjectComments
Project Share
let jCompProjCell = jCompProjectsFeedCollectionView.dequeueReusableCell(withReuseIdentifier: "JCompProjPostCell", for: indexPath) as!
JCompProjPostCell
let post = jCompProjectsPosts[indexPath.row]
let user = users[indexPath.row]
jCompProjCell.jCompProjPost = post
jCompProjCell.user = user
// linking home VC & home table view cell
jCompProjCell.jCompProjFeedVC = self
return jCompProjCell
}
// reference to store JComponentProjectModel class info
var jCompProjectsPosts = [JComponentProjectModel]()
// reference to store User class info
var users = [AppUser]()
// DB ref
// --- var refJCompProjects: DatabaseReference!
// load jcomp project posts
func loadPosts() {
// start when loadPosts func starts
activityIndicatorView8.startAnimating()
Api.JComponentProjectPost.observePosts { (post) in
guard let postId = post.uid else {
return
}
// fetch user data in mentor posts
self.fetchUser(uid: postId, completed: {
self.jCompProjectsPosts.append(post)
// print(self.posts)
/// stop before tablew view reloads data
self.activityIndicatorView8.stopAnimating()
self.activityIndicatorView8.hidesWhenStopped = true
self.jCompProjectsFeedCollectionView.reloadData()
})
}
Line: 120 Col: 1



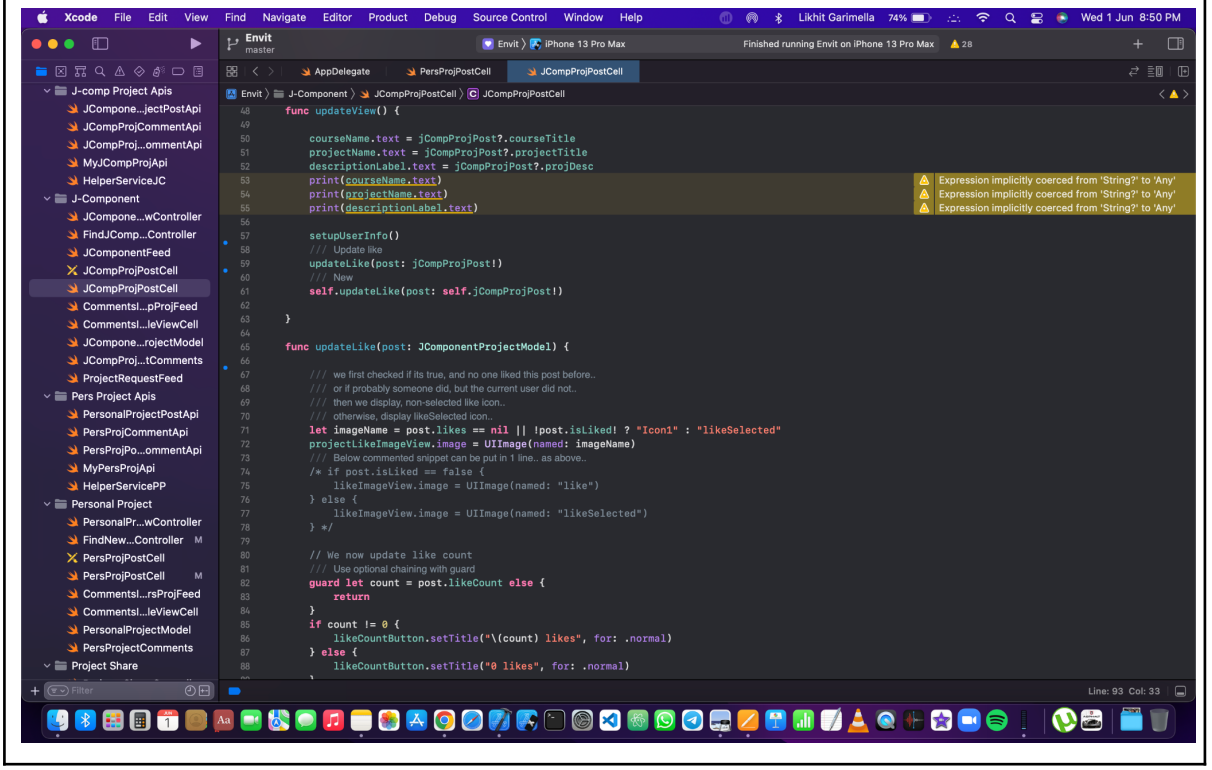
```
class HelperServiceJC {

    static func uploadDataToServer(titleText: String, courseText: String, descText: String, onSuccess: @escaping () -> Void) {

        self.sendDataToDatabase(titleText: titleText, courseText: courseText, descText: descText, onSuccess: onSuccess)

    }

    static func sendDataToDatabase(titleText: String, courseText: String, descText: String, onSuccess: @escaping () -> Void) {
        let newPostId = Api.JComponentProjectPost.REF_POSTS.childByAutoId().key
        let newPostReference = Api.JComponentProjectPost.REF_POSTS.child(newPostId!)
        guard let currentUser = Api.UserDet.CURRENT_USER else {
            return
        }
        let currentUserId = currentUser.uid
        // Creating a timestamp
        let timestamp = NSNumber(value: Int(NSDate().timeIntervalSince1970))
        // put that download url string in db

        newPostReference.setValue(["1) Title": titleText, "2) Course": courseText, "3) Description": descText, "4) Timestamp": timestamp, "5) uid":
            currentUserId], withCompletionBlock: { (error, ref) in
            if error != nil {
                print(error!.localizedDescription)
                return
            }

            // reference for my posts
            let myPostRef = Api.MyJcompProjPosts.REF_MYPOSTS.child(currentUserId).child(newPostId!)
            myPostRef.setValue(true, withCompletionBlock: {
                (error, ref) in
                if error != nil {
                    print(error!.localizedDescription)
                    return
                }
            })
            onSuccess()
        })
    }

}  // #50
```



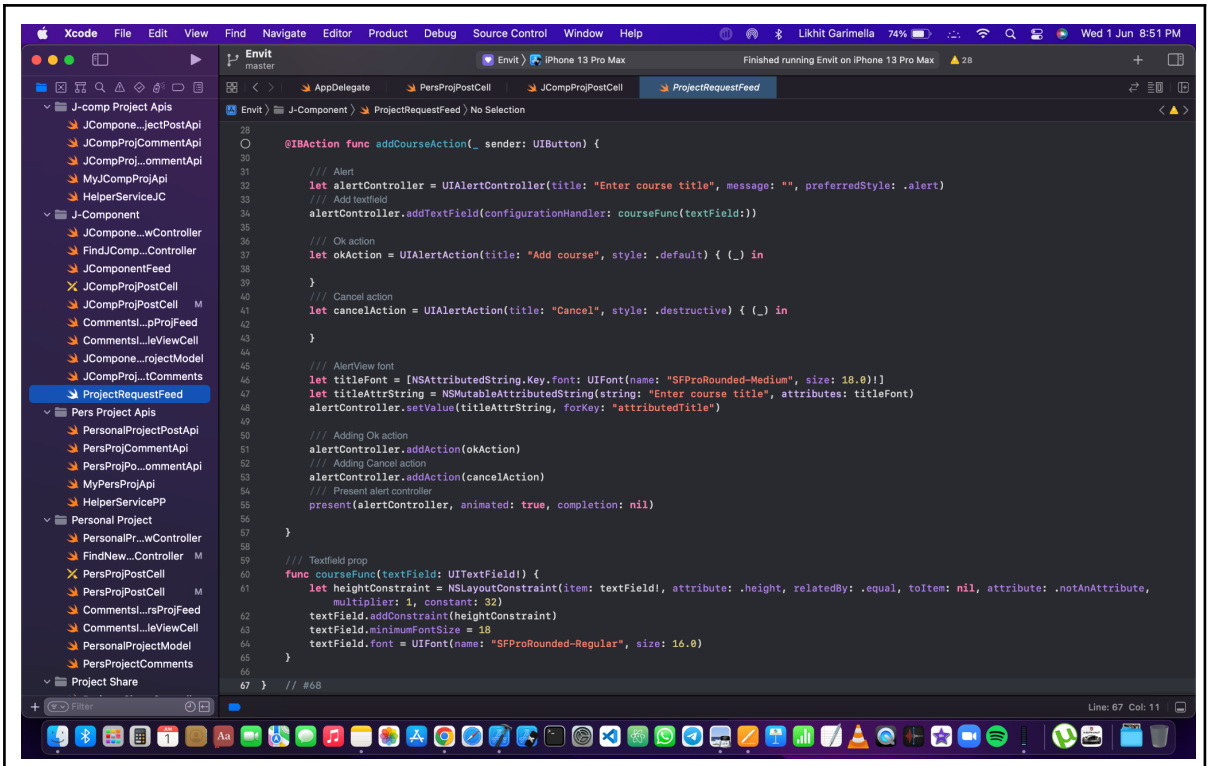
```
    /// Remodel Post class, bcuz it currently doesn't have a post id property
    var id: String?

    ///
    var likeCount: Int?
    var likes: Dictionary<String, Any>?

    var isLiked: Bool?

}

extension JComponentProjectModel {

    static func transformJCompProjPost(dict: [String: Any], key: String) -> JComponentProjectModel {

        let jcompProjPost = JComponentProjectModel()
        /// Remodel Post class, bcuz it currently doesn't have a post id property
        jcompProjPost.id = key
        jcompProjPost.projectTitle = dict["1) Title"] as? String
        jcompProjPost.courseTitle = dict["2) Course"] as? String
        jcompProjPost.projDesc = dict["3) Description"] as? String
        jcompProjPost.uid = dict["5) uid"] as? String

        jcompProjPost.likeCount = dict["likeCount"] as? Int
        jcompProjPost.likes = dict["likes"] as? Dictionary<String, Any>

        if let currentUserId = Auth.auth().currentUser?.uid {
            if jcompProjPost.likes != nil {
                /* if post.likes[currentUserId] != nil {
                    post.isLiked = true
                } else {
                    post.isLiked = false
                } */
                /// Above commented snippet can be put in 1 line.. as below..
                jcompProjPost.isLiked = jcompProjPost.likes![currentUserId] != nil
            }
        }

        return jcompProjPost

    }
```

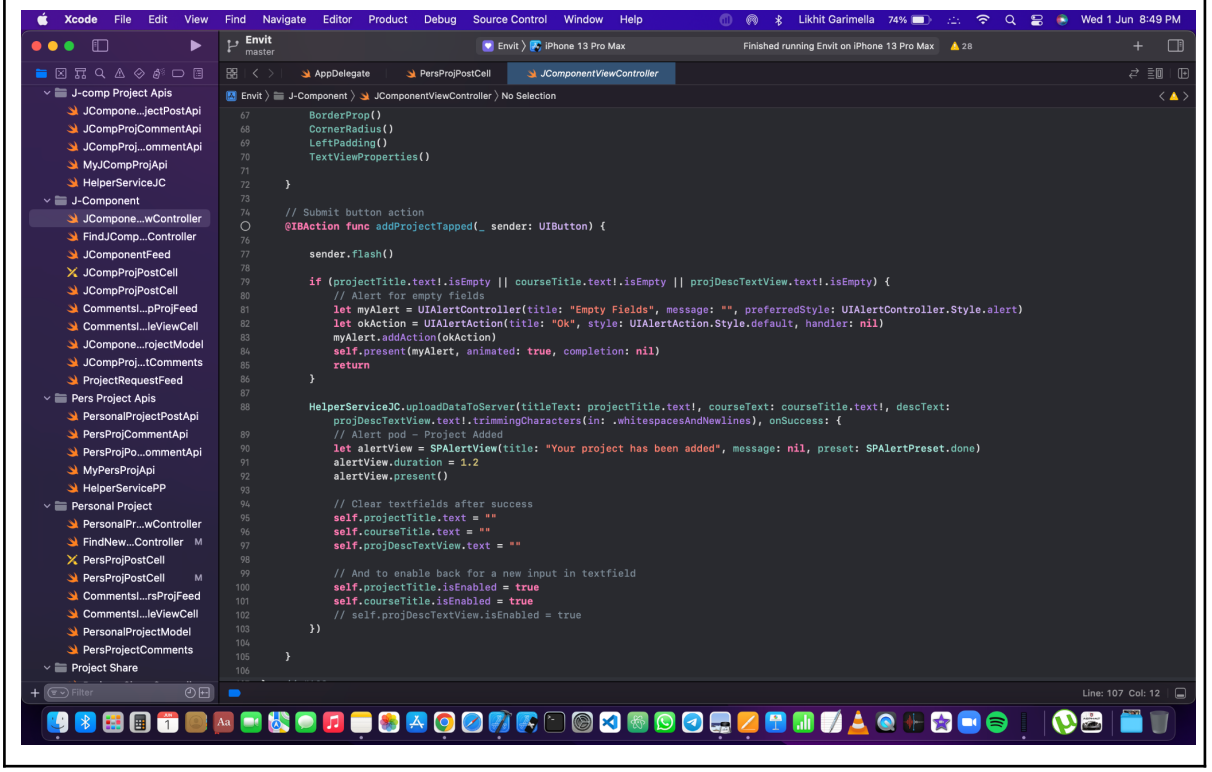


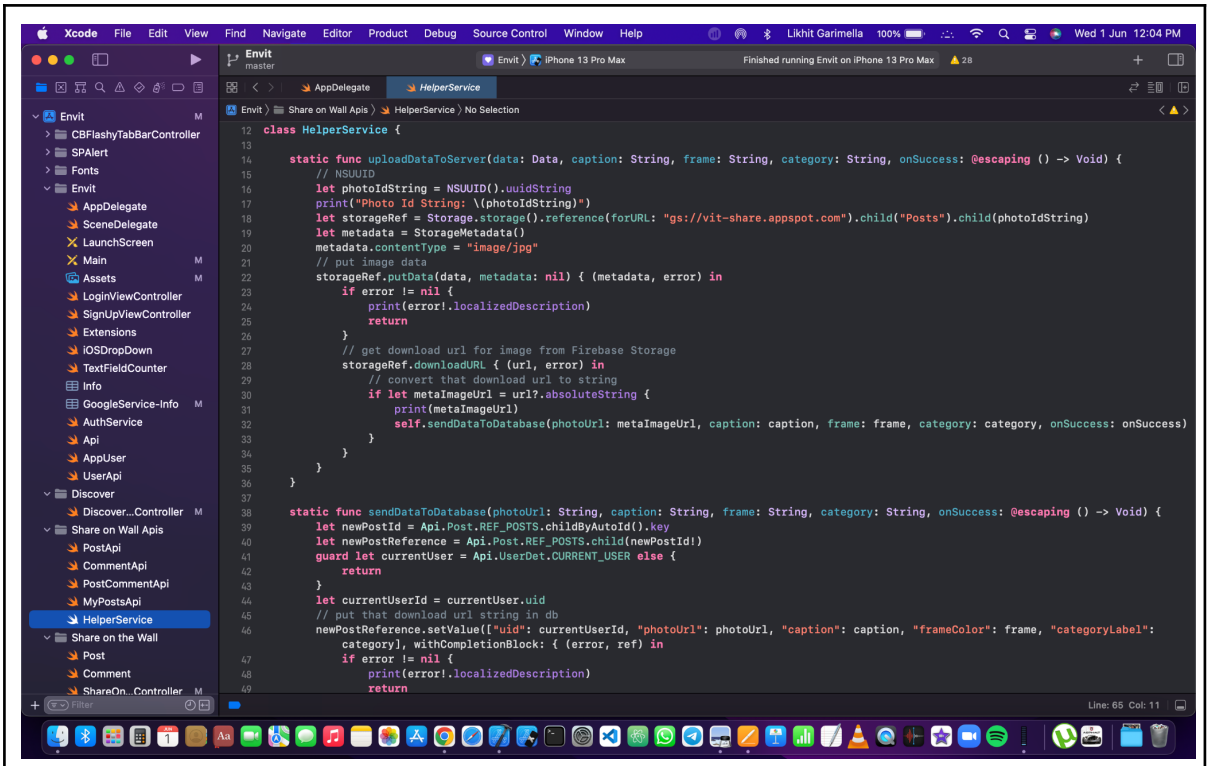
```
class JComponentProjectPostApi {

    var REF_POSTS = Database.database().reference().child("J-Component Projects").child("Details")

    func observePosts(completion: @escaping (JComponentProjectModel) -> Void) {

        REF_POSTS.observe(.childAdded, with: { (snapshot) in

            if let dict = snapshot.value as? [String: Any] {
                let newPost = JComponentProjectModel.transformJCompProjPost(dict: dict, key: snapshot.key)
                completion(newPost)
            }

        })

    }

    func observePost(withId id: String, completion: @escaping (JComponentProjectModel) -> Void) {

        REF_POSTS.child(id).observeSingleEvent(of: .value, with: {
            snapshot in

            if let dict = snapshot.value as? [String:Any] {
                let post = JComponentProjectModel.transformJCompProjPost(dict: dict, key: snapshot.key)
                completion(post)
            }

        })

    }

    func observeLikeCount(withPostId id: String, completion: @escaping (Int) -> Void) {

        REF_POSTS.child(id).observe(.childChanged, with: {
            snapshot in
            print(snapshot)
            if let value = snapshot.value as? Int {
                completion(value)
            }
        })
```



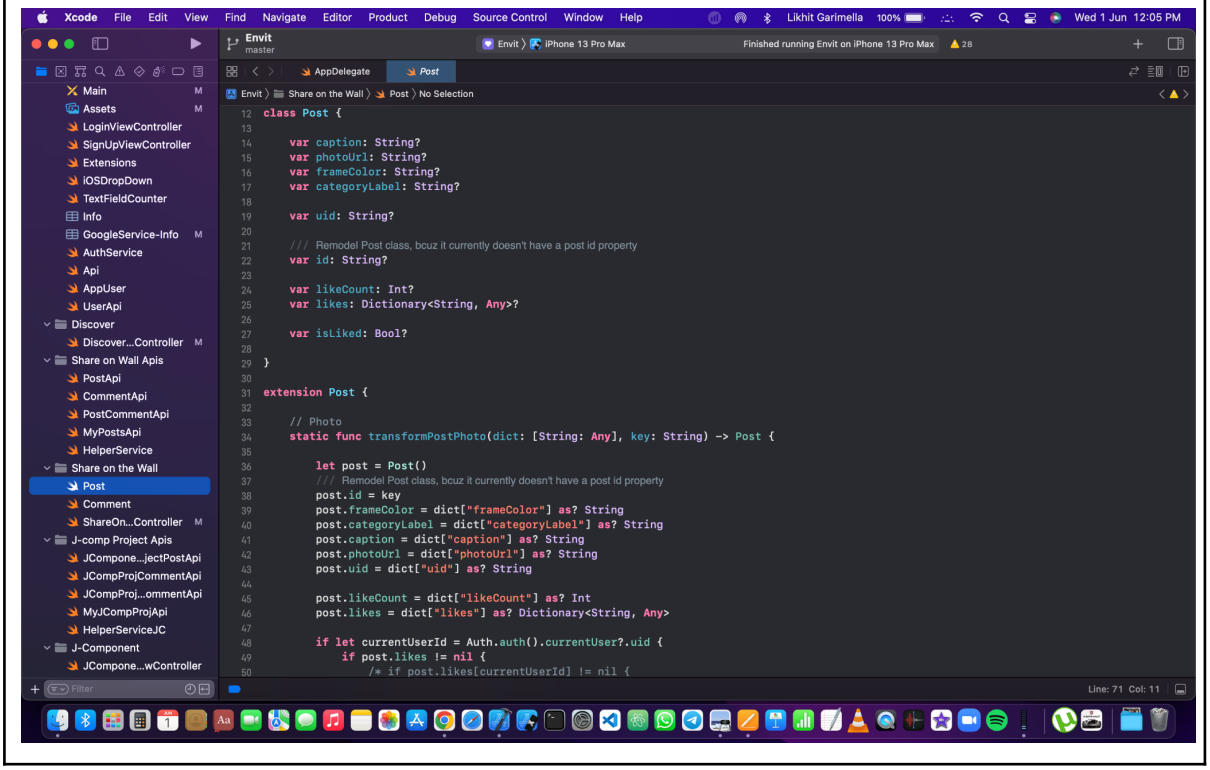
```
    func updateView() {

        courseName.text = jCompProjPost?.courseTitle
        projectName.text = jCompProjPost?.projectTitle
        descriptionLabel.text = jCompProjPost?.projDesc
        print(courseName.text)
        print(projectName.text)
        print(descriptionLabel.text)

        setupUserInfo()
        /// Update like
        updateLike(post: jCompProjPost!)
        /// New
        self.updateLike(post: self.jCompProjPost!)

    }

    func updateLike(post: JComponentProjectModel) {

        /// 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"
        projectLikeImageView.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)
```


Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Likhit Garimella 74% Wed 1 Jun 8:51 PM
Envit
master
Envit 〉 iPhone 13 Pro Max
Finished running Envit on iPhone 13 Pro Max
AppDelegate
PersProjPostCell
JCompProjPostCell
ProjectRequestFeed
Envit 〉 J-Component 〉 ProjectRequestFeed 〉 No Selection
J-comp Project Apis
JCompone...jectPostApi
JCompProjCommentApi
JCompProj...ommentApi
MyJCompProjApi
HelperServiceJC
J-Component
JCompone...wController
FindJComp...Controller
JComponentFeed
JCompProjPostCell
JCompProjPostCell M
CommentsI...pProjFeed
CommentsI...leViewCell
JCompone...rojectModel
JCompProj...tComments
ProjectRequestFeed
Pers Project Apis
PersonalProjectPostApi
PersProjCommentApi
PersProjPo...ommentApi
MyPersProjApi
HelperServicePP
Personal Project
PersonalPr...wController
FindNew...Controller M
PersProjPostCell
PersProjPostCell M
CommentsI...rsProjFeed
CommentsI...leViewCell
PersonalProjectModel
PersProjectComments
Project Share
Filter
Line: 67 Col: 11

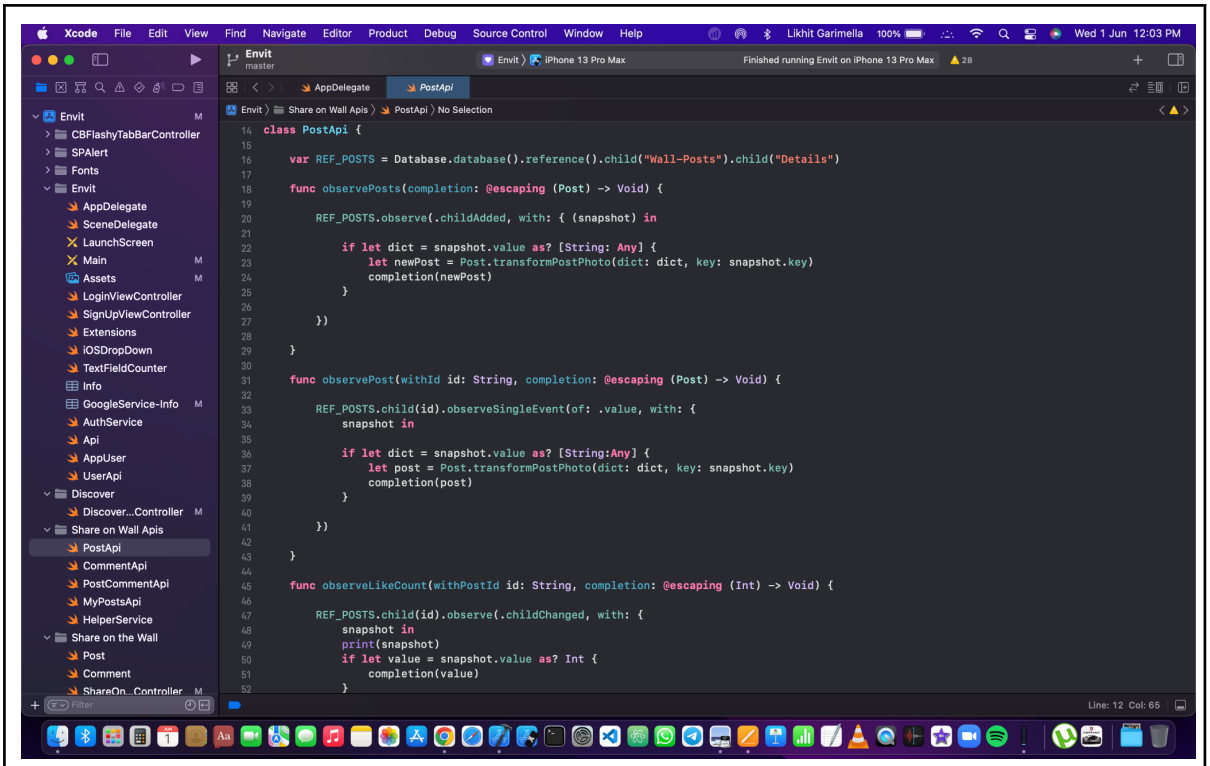

Xcode File Edit View Find Navigate Editor Product Debug Source Control Window Help
Likhit Garimella 74% Wed 1 Jun 8:49 PM
Envit
master
Envit 〉 iPhone 13 Pro Max
Finished running Envit on iPhone 13 Pro Max
AppDelegate
PersProjPostCell
JComponentViewController
Envit 〉 J-Component 〉 JComponentViewController 〉 No Selection
J-comp Project Apis
JCompone...jectPostApi
JCompProjCommentApi
JCompProj...ommentApi
MyJCompProjApi
HelperServiceJC
J-Component
JCompone...wController
FindJComp...Controller
JComponentFeed
JCompProjPostCell
JCompProjPostCell
CommentsI...pProjFeed
CommentsI...leViewCell
JCompone...rojectModel
JCompProj...tComments
ProjectRequestFeed
Pers Project Apis
PersonalProjectPostApi
PersProjCommentApi
PersProjPo...ommentApi
MyPersProjApi
HelperServicePP
Personal Project
PersonalPr...wController
FindNew...Controller M
PersProjPostCell
PersProjPostCell M
CommentsI...rsProjFeed
CommentsI...leViewCell
PersonalProjectModel
PersProjectComments
Project Share
Filter
Line: 107 Col: 12



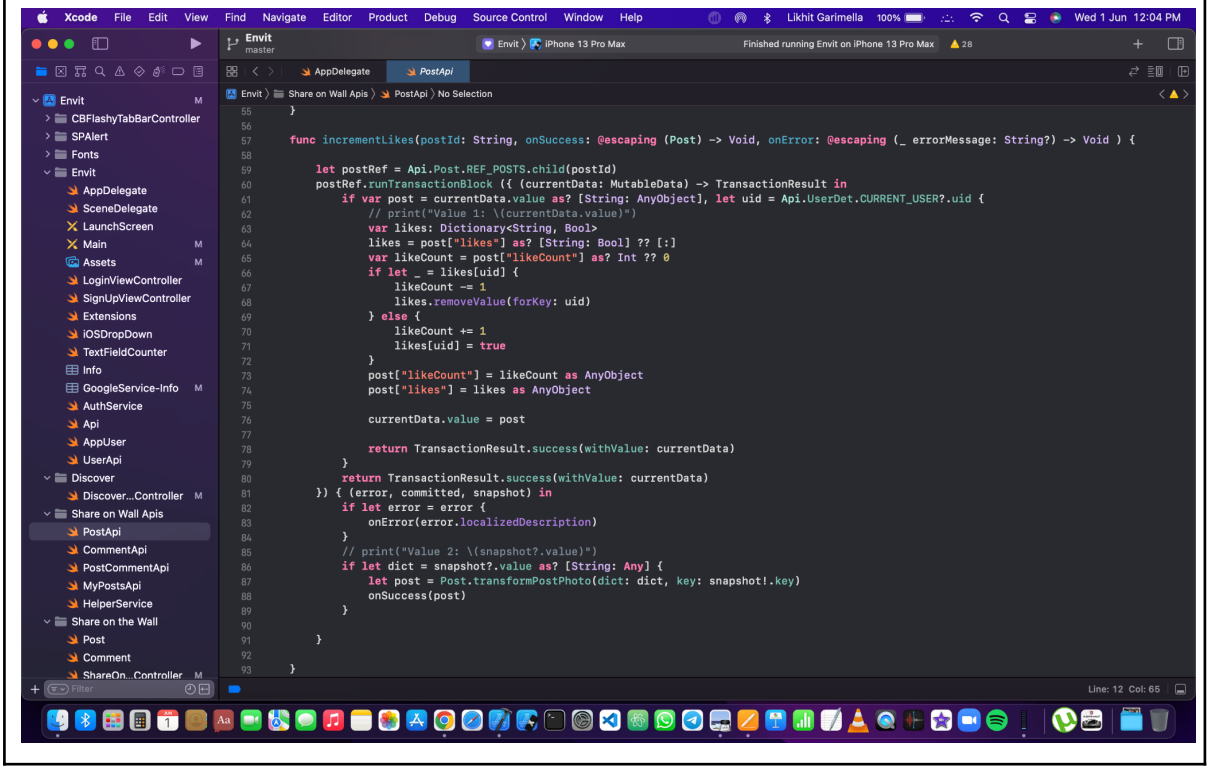
```
class HelperService {

    static func uploadDataToServer(data: Data, caption: String, frame: String, category: String, onSuccess: @escaping () -> Void) {
        // NSUUID
        let photoIdString = NSUUID().uuidString
        print("Photo Id String: \(photoIdString)")
        let storageRef = Storage.storage().reference(forURL: "gs://vit-share.appspot.com").child("Posts").child(photoIdString)
        let metadata = StorageMetadata()
        metadata.contentType = "image/jpg"
        // put image data
        storageRef.putData(data, metadata: nil) { (metadata, error) in
            if error != nil {
                print(error!.localizedDescription)
                return
            }
            // get download url for image from Firebase Storage
            storageRef.downloadURL { (url, error) in
                // convert that download url to string
                if let metaImageUrl = url?.absoluteString {
                    print(metaImageUrl)
                    self.sendDataToDatabase(photoUrl: metaImageUrl, caption: caption, frame: frame, category: category, onSuccess: onSuccess)
                }
            }
        }
    }

    static func sendDataToDatabase(photoUrl: String, caption: String, frame: String, category: String, onSuccess: @escaping () -> Void) {
        let newPostId = Api.Post.REF_POSTS.childByAutoId().key
        let newPostReference = Api.Post.REF_POSTS.child(newPostId!)
        guard let currentUser = Api.UserDet.CURRENT_USER else {
            return
        }
        let currentUserId = currentUser.uid
        // put that download url string in db
        newPostReference.setValue(["uid": currentUserId, "photoUrl": photoUrl, "caption": caption, "frameColor": frame, "categoryLabel":
            category], withCompletionBlock: { (error, ref) in
            if error != nil {
                print(error!.localizedDescription)
                return
```



```
class Post {

    var caption: String?
    var photoUrl: String?
    var frameColor: String?
    var categoryLabel: String?

    var uid: String?

    /// Remodel Post class, bcuz it currently doesn't have a post id property
    var id: String?

    var likeCount: Int?
    var likes: Dictionary<String, Any>?

    var isLiked: Bool?

}

extension Post {

    // Photo
    static func transformPostPhoto(dict: [String: Any], key: String) -> Post {

        let post = Post()
        /// Remodel Post class, bcuz it currently doesn't have a post id property
        post.id = key
        post.frameColor = dict["frameColor"] as? String
        post.categoryLabel = dict["categoryLabel"] as? String
        post.caption = dict["caption"] as? String
        post.photoUrl = dict["photoUrl"] as? String
        post.uid = dict["uid"] as? String

        post.likeCount = dict["likeCount"] as? Int
        post.likes = dict["likes"] as? Dictionary<String, Any>

        if let currentUserId = Auth.auth().currentUser?.uid {
            if post.likes != nil {
                /* if post.likes[currentUserId] != nil {
```



```
class PostApi {

    var REF_POSTS = Database.database().reference().child("Wall-Posts").child("Details")

    func observePosts(completion: @escaping (Post) -> Void) {

        REF_POSTS.observe(.childAdded, with: { (snapshot) in

            if let dict = snapshot.value as? [String: Any] {
                let newPost = Post.transformPostPhoto(dict: dict, key: snapshot.key)
                completion(newPost)
            }

        })

    }

    func observePost(withId id: String, completion: @escaping (Post) -> Void) {

        REF_POSTS.child(id).observeSingleEvent(of: .value, with: {
            snapshot in

            if let dict = snapshot.value as? [String:Any] {
                let post = Post.transformPostPhoto(dict: dict, key: snapshot.key)
                completion(post)
            }

        })

    }

    func observeLikeCount(withPostId id: String, completion: @escaping (Int) -> Void) {

        REF_POSTS.child(id).observe(.childChanged, with: {
            snapshot in
            print(snapshot)
            if let value = snapshot.value as? Int {
                completion(value)
            }
```



```
    }

    func incrementLikes(postId: String, onSuccess: @escaping (Post) -> Void, onError: @escaping (_ errorMessage: String?) -> Void ) {

        let postRef = Api.Post.REF_POSTS.child(postId)
        postRef.runTransactionBlock ({ (currentData: MutableData) -> TransactionResult in
            if var post = currentData.value as? [String: AnyObject], let uid = Api.UserDet.CURRENT_USER?.uid {
                // print("Value 1: \(currentData.value)")
                var likes: Dictionary<String, Bool>
                likes = post["likes"] as? [String: Bool] ?? [:]
                var likeCount = post["likeCount"] as? Int ?? 0
                if let _ = likes[uid] {
                    likeCount -= 1
                    likes.removeValue(forKey: uid)
                } else {
                    likeCount += 1
                    likes[uid] = true
                }
                post["likeCount"] = likeCount as AnyObject
                post["likes"] = likes as AnyObject

                currentData.value = post

                return TransactionResult.success(withValue: currentData)
            }
            return TransactionResult.success(withValue: currentData)
        }) { (error, committed, snapshot) in
            if let error = error {
                onError(error.localizedDescription)
            }
            // print("Value 2: \(snapshot?.value)")
            if let dict = snapshot?.value as? [String: Any] {
                let post = Post.transformPostPhoto(dict: dict, key: snapshot!.key)
                onSuccess(post)
            }

        }

    }
```

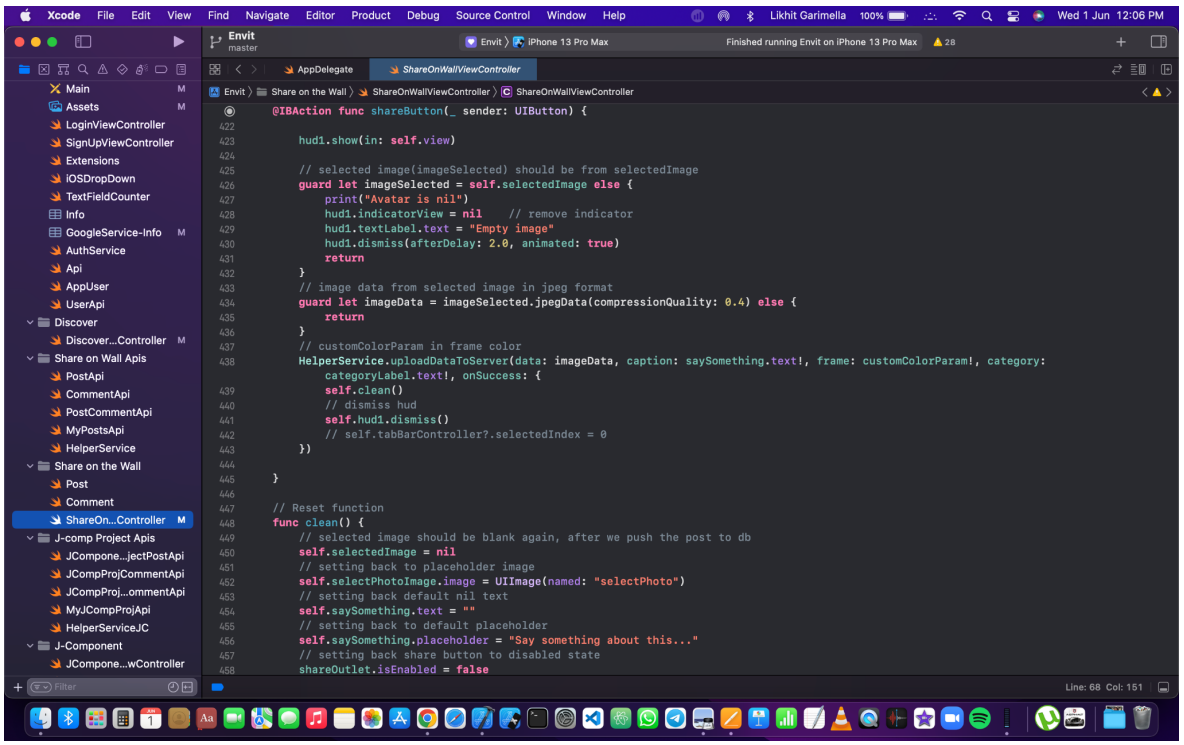


Fig 14: Xcode Swift Codes Screenshots

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.

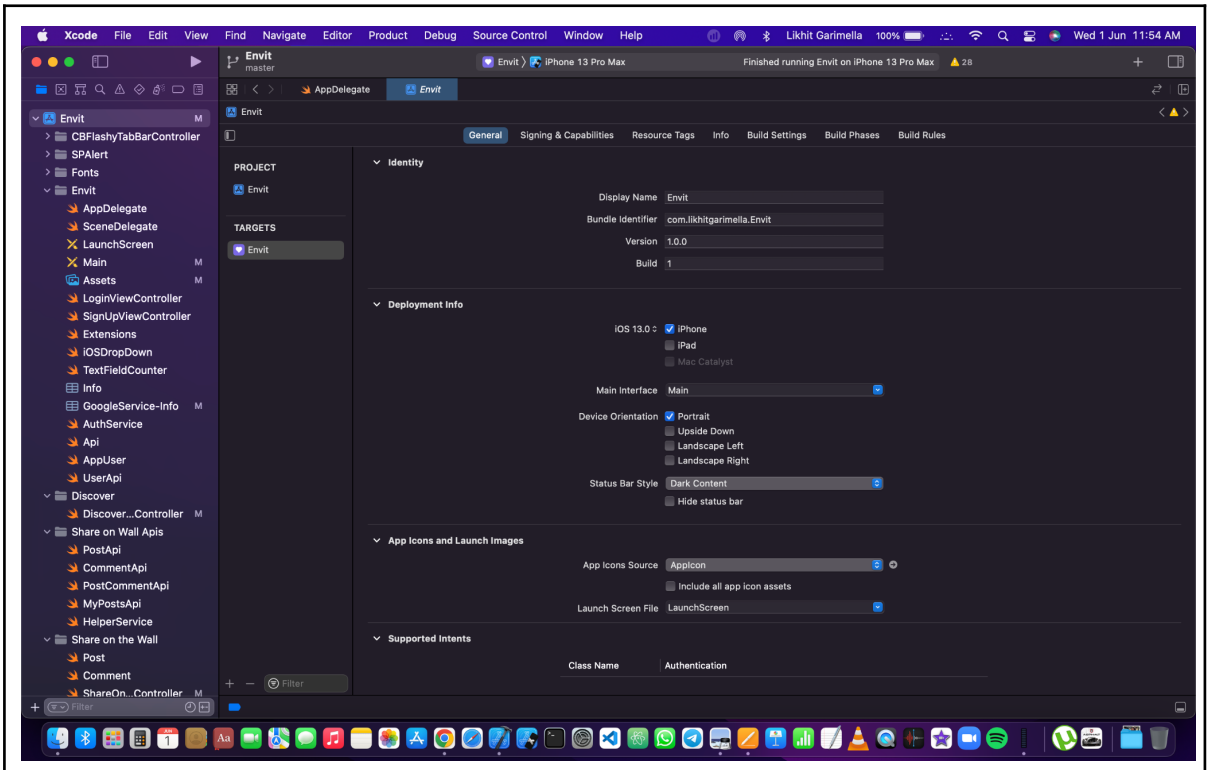
[8] Magda Chelly , Hana Mataillet , “Social Media and the impact on education: Social media and home education” ,2012 International Conference on E-Learning and E-Technologies in Education (ICEEE) , 24-26 Sept. 2012.

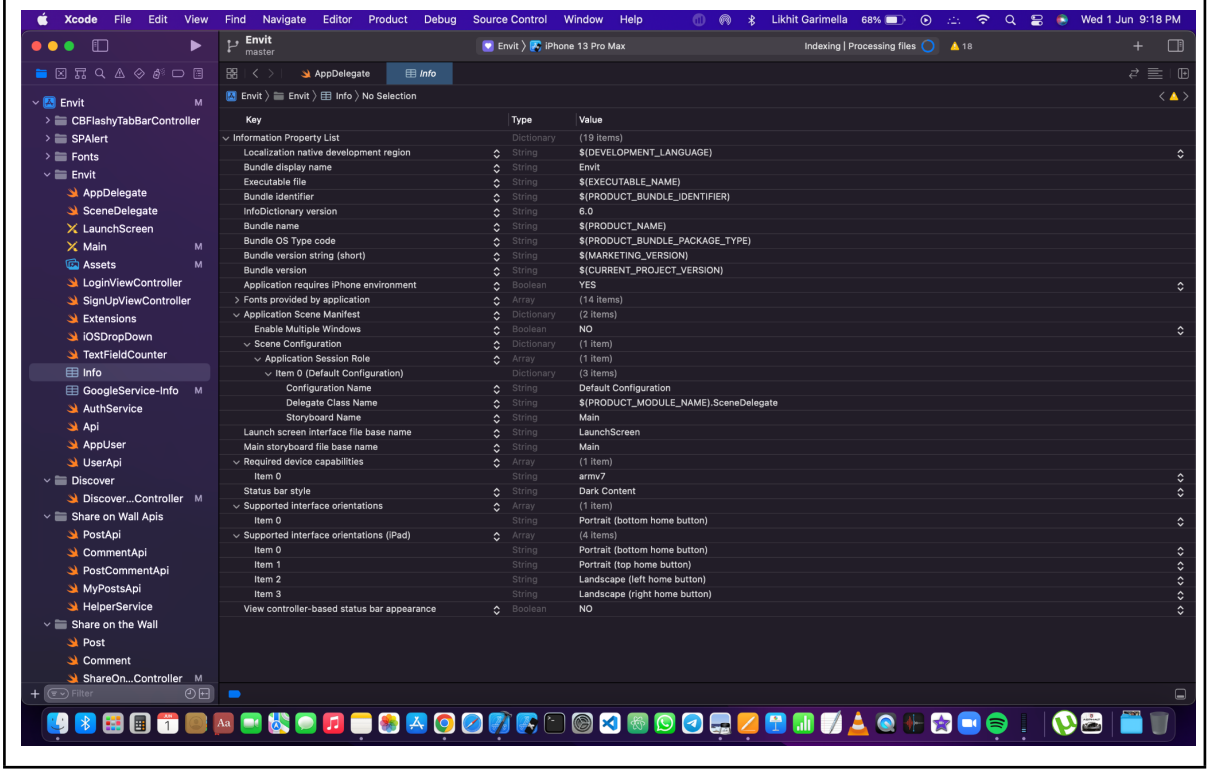
[9] Asaf Varol , Naveed Ahmed , “Social networks' role in online education” , International Conference on Application of Information and Communication Technologies (AICT) , 23-25 Oct. 2013.

[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:





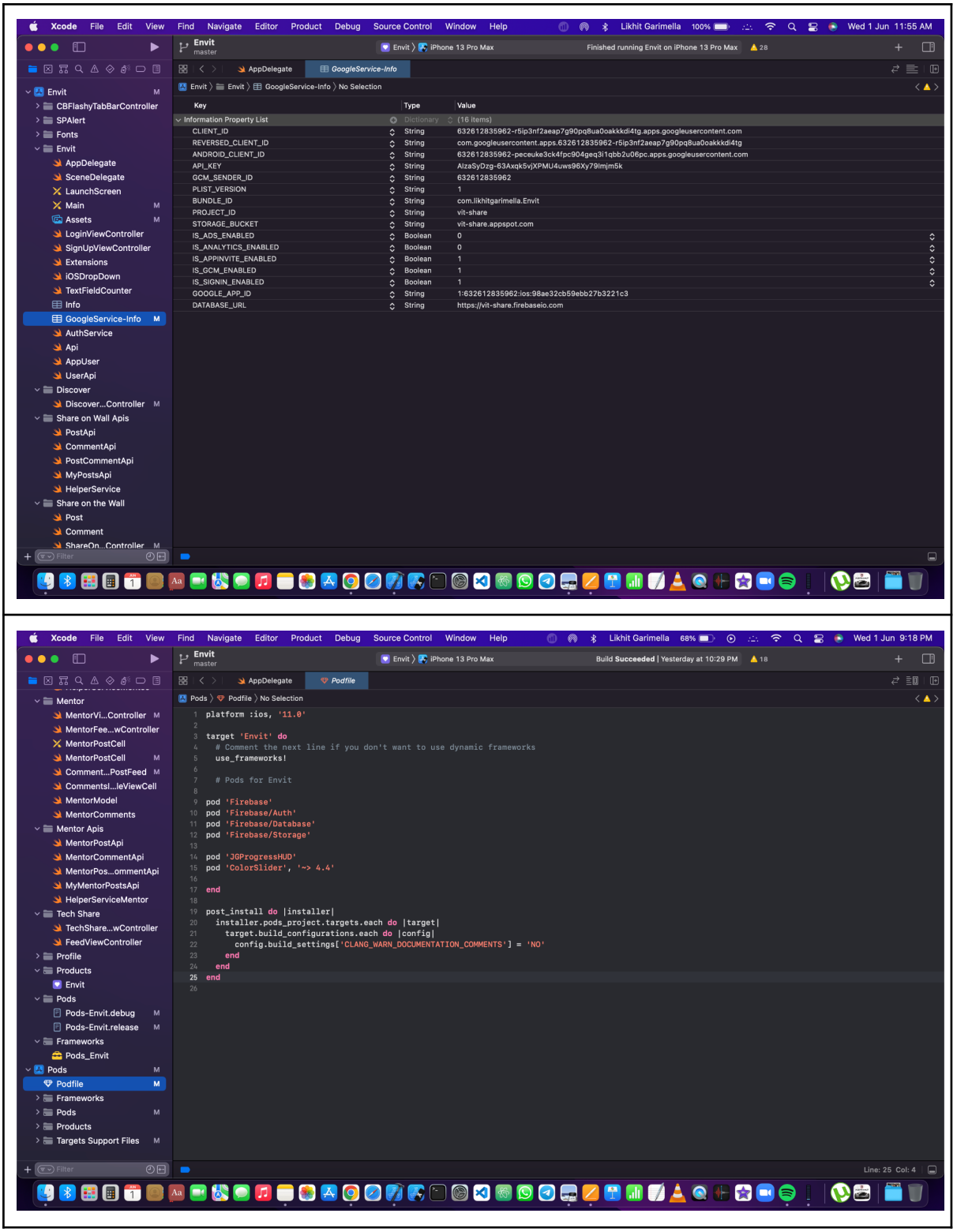


Fig 15: Xcode project, Info plist, Google service plist, Podfile Screenshots