IOS APP DEVELOPMENT INTERNSHIP

FROM

An Internship Report

Submitted by:

Mirva Dudhagara

(190630107024)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

MADHUBEN & BHANUBHAI PATEL

INSTITUTE OF TECHNOLOGY (CE)

At

Gujarat Technological University,

Ahmedabad, Gujarat

April, 202

Madhuben and Bhanubhai Patel Institute of Technology

New Vallabh Vidyanagar, 388121

CERTIFICATE

This is to certify that the project report submitted along with the project entitled **iOS App Development Internship** has been carried out by **Mirva D. Dudhagara** under my guidance in partial fulfillment for the degree of Bachelor of Engineering in **Computer engineering**, 8th Semester of Gujarat Technological University, Ahmedabad during the academic year 2023- 24.

Dr. Ashwini Kumar Jha

Internal Guide

Dr. Shital Gondaliya

Head of the Department

Swayam Infotech

Date: 1st February 2023

To Whom It May Concern

This letter is to confirm that **Miss. Mirva Dineshbhai Dudhagara** is enrolled in the **Advanced iOS Apps Development** Training Programme at the **Swayam Infotech** since 1st February 2023. This program consists of practical study and works on industrial projects along with the final year academic project.

Project Name: swiftPocket - UPI App

Note:
This letter has been issued for the purpose of academic project requirements. An internship is not completed yet.

For Swayam Infotech
Jitesh Vadodariya

2 Jalaram Plot, Street - 4, Janki Marg, University Road, Rajkot 360005, Gujarat - India
www.swayaminfotech.com

Madhuben and Bhanubhai Patel Institute of Technology

New Vallabh Vidyanagar, 388121

DECLARATION

I hereby declare that the Internship report submitted along with the entitled iOS App Development Intership submitted in partial fulfillment for the degree of Bachelor of Engineering in computer engineering to Gujarat Technological University, Ahmedabad, is a bonafide record of original project work carried out by me at Swayam Infotech under the supervision of Rohit Tank and that no part of this report has been directly copied from any students' reports or taken from any other source, without providing due reference.

Name of the Student **Sign of Student**

Mirva D. Dudhagara

ACKNOWLEDGMENT

I would like to thank the Department of Computer Engineering, Madhuben and Bhanubhai Institute of Technology, Gujarat Technological University, and Swayam Infotech, Rajkot for giving me an opportunity to work practically in the industry under a major project.

I would like to thank my internal member **Dr. Ashwini Kumar Jha** for providing the guidance and support throughout this journey. I would like to thank **Prof. Sunit Parmar** and **Prof. Nirav Patel** for guide me about internship and helped me during internship. I would also like to thank Head of Department **Dr. Shital Gondaliya** for being a great support during internship. I also like to thank **StackOverflow** and the **Github** community for providing support for my implementation problems and thanks to all the creators of this technology.

I would like to thank my external mentors **Mr. Rohit Tank** and **Mr. Jitesh Vadodariya** and entire **Swayam iOS** team for providing me the opportunity to work on different iOS technology and providing his assistance throughout my projects.

Mirva Dudhagara

ABSTRACT

This abstract provides an overview of a robust UPI payment app designed to simplify and optimize the payment experience for users. The app enables individuals and businesses to effortlessly transfer funds, pay bills, and make purchases using their smartphones, leveraging the Unified Payments Interface (UPI) system for real-time transactions between banks and payment service providers. The UPI payment app focuses on user convenience and security. It offers a user-friendly interface for easy navigation and operation. To protect sensitive data, the app implements strong security measures including advanced encryption and multi-factor authentication, such as one-time passwords (OTP). Supporting various transaction types, the app facilitates person-to-person transfers, peer-to-merchant payments, and seamless integration with popular digital wallets. Businesses can also benefit from dedicated merchant accounts, enabling direct payments into their bank accounts for streamlined financial operations.

It prioritizes speed, accuracy, and reliability to deliver a seamless user experience. In conclusion, the UPI payment app represents a significant advancement in the digital payment landscape, combining convenience, security, and efficiency. By simplifying financial transactions, facilitating interbank transfers, and focusing on user-friendly features, this app serves as a comprehensive platform for individuals and businesses alike, enhancing the overall payment ecosystem.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1: OVERVIEW OF THE COMPANY

Swayam Infotech is a web and mobile app development company providing web and mobile app solutions of any complexity worldwide. It was established on 7th March, 2007 and it recently completed 16 years of being in the IT business. Swayam Infotech is an established company in India designs, develops, integrates, and maintains website application development that enables enterprises to solve complex and critical business problems.

Fig 1.1 Company Logo

1.1 HISTORY

Swayam Infotech is an established company in India designs, develops, integrates, and maintains website application development that enables enterprises to solve complex and critical business problems. Its basic competencies include Outsource Web Development, Application Development, E- Commerce Solutions, Business Process Outsourcing, and other IT Services. Over the 16 long years, Swayam Infotech has contributed to various industries by providing quality IT services. Besides, it has also recently embarked upon developing some projects of its own. The industries that have been benefitted by Swayam Infotech include the following

- Real estate
- Online auctioning
- Healthcare
- Music and entertainment
- Social networking
- Education and e-learning
- E-commerce

- Restaurants and bars
- Personal safety
- Blogs
- Finance
- Travel
- Hotel bookings
- Market place
- Job portals.

1.2 DIFFERENT SERVICES

Swayam Infotech mainly works on web and mobile app development. It also provides services for testing, quality assurance, search engine optimization and graphic designing.

- **Mobile App Development**
  - iOS – iPhone and iPad application development
  - Android – Mobile and tab application development
  - React Native – Cross platform application development
  - Ionic – for mobile and web application development
- **Web Development**
  - Node Js – front-end and back-end development
  - React Js – front-end development
  - Laravel – front-end and back-end development'
  - Shopify – e-commerce web development
- **Others**
  - UI/UX graphic designing
  - SEO/SEM
  - Quality Assurance

1.3 PRODUCTS

Product	Purpose
30 secs	On demand video sharing app
Property Hunter	Property rental and sell app
Medix	Video learning platform
Swayam Taxi app	Cab booking
Your wardrobe	E-commerce
Monkey Jobs	Mobile wallet app
Muser	Video player app

Table 1: Company products

1.4 ADMINISTRATION AND LOCATION

Swayam Infotech is founded and lead by its CEO Mr. Anand Patel with a team of around 15 employees including iOS app developers, Android app developers, Digital marketer and UI/UX Designer. Swayam Infotech has two firms in total.

- Jalaram 2, Street 4, Janki Marg, University Road, Rajkot, Gujarat (India)
- 89 Windale Crescent, Kitchener, on N2E 3H4, Canada

CHAPTER 2: INTRODUCTION TO INTERNSHIP

This chapter presents the process of the training, the overview of the technology, scope, objective and purpose of the internship.

2.1 INTERNSHIP SUMMARY

During my 12-week internship focused on iOS app development, I gained valuable hands-on experience and expanded my skillset in creating robust and user-friendly applications for Apple devices. Working closely with a talented team of developers at Swayam Infotech, I had the opportunity to contribute to the development of an iOS app and learn from experienced mentors in the field.

Throughout the internship, I was involved in the entire app development lifecycle. This included requirements gathering, UI/UX design and coding. I primarily utilized Swift, Apple's programming language for iOS app development, and worked with various frameworks and tools such as UIKit, CoreData, and Xcode. I implemented new features, resolved bugs, and optimized performance, ensuring seamless navigation and responsiveness.

Additionally, I had the opportunity to work on a greenfield project, where I designed and developed a new iOS app from scratch. This involved creating intuitive user interfaces, integrating with backend APIs. I also stayed updated with the latest trends and best practices in iOS app development and self-directed learning to enhance my skills.

Overall, my 12-week internship in iOS app development was a transformative experience that allowed me to deepen my knowledge, hone my technical skills, and gain real-world exposure to the app development process. I am confident that the experience and expertise I gained will serve as a strong foundation for my future career in mobile app development.

The title of the greenfield project that I developed from scratch is swiftPocket – UPI App. This app allows UPI payments and more.

2.2 PROBLEM DEFINITION

SwiftPocket, an iOS UPI (Unified Payments Interface) app, aims to address the problem of inefficient and cumbersome financial transactions on phones. Traditional methods of making payments and managing finances often involve multiple steps, manual input of details, and limited accessibility. SwiftPocket seeks to streamline this process by providing a user-friendly platform that enables swift and secure transactions. It addresses the need for a seamless and intuitive interface, extensive UPI bank integration, real-time notifications, and advanced security measures. By solving these pain points, SwiftPocket aims to enhance the overall user experience and revolutionize the way users handle their financial activities on their iPhones.

2.3 PURPOSE OF THE PROJECT

The purpose of SwiftPocket is to provide iOS users with a user-friendly, comprehensive, and secure financial management solution. The app aims to simplify financial transactions on iPhones, addressing the pain points associated with traditional financial management methods, such as complexity, inefficiency, and limited accessibility.

SwiftPocket's purpose is to offer users a seamless and secure platform for managing their finances, making payments, and transferring funds. The app's extensive integration with UPI-enabled banks provides users with a broad range of options to link their bank accounts and manage their finances effortlessly. Additionally, SwiftPocket offers various features such as bill payments, QR code scanning for payments, and integration with popular digital wallets, making it a comprehensive financial management solution.

Another critical purpose of SwiftPocket is to provide advanced security measures to safeguard sensitive information. The app offers features such as biometric authentication, transaction limits, and real-time notifications to ensure secure financial transactions.

Overall, the purpose of SwiftPocket is to enhance the user experience and revolutionize the way users handle their financial activities on their iPhones.

2.4 OBJECTIVE OF THE PROJECT

The objective of SwiftPocket is to provide iOS users with a comprehensive, secure, and user-friendly financial management solution. The app's primary objective is to simplify financial transactions on iPhones, enabling users to manage their finances effortlessly and efficiently.

The app's objectives include offering a seamless and secure platform for managing financial transactions, providing extensive integration with UPI-enabled banks, and supporting various financial activities such as bill payments, QR code scanning for payments, and integration with popular digital wallets.

Another objective of SwiftPocket is to enhance the user experience by providing a user-friendly interface, advanced security measures, and real-time notifications. The app aims to offer users a hassle-free financial management solution that meets their needs and exceeds their expectations.

SwiftPocket's objectives also include catering to the increasing demand for mobile payments, which is expected to grow in the coming years. With its comprehensive financial management features and intuitive interface, SwiftPocket aims to provide users with a reliable and user-friendly financial management solution.

Overall, the objective of SwiftPocket is to provide iOS users with a comprehensive and user-friendly financial management solution that enables them to manage their financial activities with ease, security, and convenience. The app's objectives align with the needs of iOS users seeking a reliable, secure, and intuitive financial management solution.

2.5 SCOPE OF THE PROJECT

The scope of swiftPocket app encompasses a wide range of features and functionalities to provide users with a comprehensive digital payment solution. Here are the key aspects of its scope:

1. User Registration and Account Linking: SwiftPocket allows users to register and create accounts using their mobile numbers, email addresses, and other required details. It enables users to securely link their bank accounts to the app, supporting multiple bank accounts for a single user.

2. Fund Transfers: Users can initiate secure and instant fund transfers from their linked bank accounts to other UPI users. The app provides various options for transferring funds, including mobile numbers, UPI IDs, and QR code scanning. It implements necessary validation and authorization processes to ensure the authenticity and security of transactions.

3. Transaction History and Statements: SwiftPocket maintains a detailed record of all transactions made through the app. Users can view their transaction history, including transaction amounts, recipient details, timestamps, and transaction status. It also provides downloadable transaction statements for easy record-keeping.

4. Payment Requests: Users can create payment requests and send them to other SwiftPocket users. Payment requests can include optional due dates and reminders, making it convenient for users to request and receive payments.

5. Bill Payments: SwiftPocket integrates with bill payment services, enabling users to conveniently pay utility bills, mobile recharges, and other expenses. It supports a wide range of billers and services, allowing users to make timely and hassle-free payments.

The scope of SwiftPocket UPI app encompasses these features and functionalities to deliver a comprehensive and user-centric digital payment solution, facilitating convenient, secure, and efficient financial transactions for its users.

CHAPTER 3: LITERATURE SURVEY

The primary focus of the system analysis is to assess user requirements and expectations. It involves gathering feedback from potential users, studying market trends, and understanding the target audience's demographics, preferences, and behaviors. By understanding the users' needs, the UPI app can be tailored to provide a seamless and user-friendly experience.

3.1 STUDY OF CURRENT SYSTEM

UPI apps have significantly transformed the digital payment landscape in India, providing users with a seamless and efficient payment experience. However, they face several challenges that can impact user confidence and limit their adoption.

One of the most significant challenges is the occasional transaction failures that users may encounter. While UPI has a high success rate, occasional transaction failures can occur due to network issues or server downtime, causing inconvenience to users. Additionally, interoperability issues between different apps and banks can further complicate the payment process. These challenges make it essential to improve the technical infrastructure, ensuring robust servers that can handle increased transaction volumes and interoperability between different apps and banks.

Another challenge is the lack of standardized user interfaces leading to inconsistencies in user experience. This is because each UPI app has a unique interface, making it challenging for users to switch between apps seamlessly. Additionally, occasional system downtimes can affect availability, making it challenging for users to complete transactions. To address these challenges, developers should adopt standardized user interfaces to enhance user experience and provide a consistent look and feel across all apps. Additionally, system downtimes should be minimized through regular maintenance and updates.

Concerns regarding transaction security and fraud prevention are another challenge that can impact user confidence. Users are wary of security breaches and fraudulent transactions, leading to hesitancy in adopting UPI apps. To address this, UPI apps must adopt advanced security measures,

including robust encryption and real-time fraud prevention. Additionally, developers should offer comprehensive customer support to assist users with account linking, dispute resolution, and prompt issue resolution, ensuring a hassle-free payment experience.

In conclusion, the challenges facing UPI apps highlight the need for continuous improvements in app performance, security measures, user interfaces, and customer support to ensure a seamless and reliable digital payment experience for all users.

3.2 PROPOSED SYSTEM

Introducing "SwiftPocket," an innovative UPI app designed to address the challenges faced by users. SwiftPocket offers a seamless and secure digital payment experience with a robust infrastructure that minimizes transaction failures and ensures interoperability between different apps and banks.

SwiftPocket's advanced server infrastructure ensures 24/7 availability, reducing system downtimes and enabling users to complete transactions smoothly. SwiftPocket also employs advanced security measures, including robust encryption and real-time fraud prevention, instilling confidence in users.

SwiftPocket features a standardized user interface, providing a consistent and intuitive experience across all interactions. This helps users switch between apps seamlessly, eliminating the need to learn new interfaces. Additionally, SwiftPocket offers personalized payment options, including the ability to save payment preferences and frequently used payment recipients.

SwiftPocket offers a range of customer support options to assist users with account linking, dispute resolution, and prompt issue resolution. Users can reach customer support via chat, phone, or email, enabling prompt and effective communication.

SwiftPocket also offers a loyalty program that rewards users for their transactions. The loyalty program offers cashback, discounts, and other incentives to users, encouraging them to make more transactions and further increasing adoption.

To ensure interoperability with different banks and apps, SwiftPocket offers an open API that allows developers to integrate with the app seamlessly. This encourages app developers to create a more seamless user experience across different apps, improving the overall payment experience for users.

In conclusion, SwiftPocket is an innovative UPI app designed to address the challenges faced by users, including transaction failures, interoperability issues, inconsistent user interfaces, occasional system downtimes, and concerns regarding transaction security and fraud prevention. SwiftPocket offers a robust infrastructure, advanced security measures, standardized user interfaces, personalized payment options, comprehensive customer support, a loyalty program, and an open API, ensuring a seamless and reliable digital payment experience for all users.

CHAPTER 4: SYSTEM ANALYSES

4.1 NON FUNCTIONAL REQUIREMENTS

Here are some common non-functional requirements that are generally applicable to UPI apps:

1. Security: UPI apps handle financial transactions and sensitive user data, so security is of utmost importance.

2. Performance: UPI apps should provide a seamless and responsive user experience. Non-functional requirements related to performance may include fast response times, low latency, efficient handling of concurrent users and transactions, and optimized data storage and retrieval.

3. Scalability: UPI apps should be able to handle increasing user loads and transaction volumes as the user base grows. Non-functional requirements related to scalability may include the ability to handle a large number of concurrent users, support for peak loads, horizontal and vertical scalability, and load balancing mechanisms.

4. Usability: UPI apps should be user-friendly and intuitive, catering to users with varying levels of technical expertise. Non-functional requirements related to usability may include a simple and intuitive user interface, clear instructions and guidance, support for multiple languages, accessibility features, and responsiveness across different devices and screen sizes.

5. Compatibility: UPI apps should be compatible with a wide range of devices and platforms to ensure accessibility for a broader user base. Non-functional requirements related to compatibility may include support for different operating systems (e.g., Android, iOS), compatibility with different screen resolutions, support for various device capabilities (e.g., camera for scanning QR codes), and integration with other systems and services (e.g., banking APIs, payment gateways).

6. Maintenance and Support: UPI apps require ongoing maintenance and support to address issues, apply updates, and introduce new features. Non-functional requirements related to maintenance

and support may include easy maintenance and upgradability, effective monitoring and error logging mechanisms, timely bug fixes, and responsive customer support.

4.2 FUNCTIONAL REQUIREMENTS

1. User Registration: The app should allow users to create an account by providing necessary information such as mobile number, email address, and personal details. It should also support authentication mechanisms like OTP (One-Time Password) verification or biometric authentication.

2. Linking Bank Accounts: Users should be able to link their bank accounts to the UPI app. The app should support multiple banks and allow users to add, remove, or switch between linked bank accounts.

3. Fund Transfer: The app should facilitate seamless transfer of funds between users. It should support various transaction types like person-to-person (P2P) transfers, person-to-merchant (P2M) transfers, and even person-to-account (P2A) transfers. Users should be able to specify the recipient's UPI ID, mobile number, or account details.

4. Payment Requests: Users should be able to send payment requests to other users, allowing them to initiate fund transfers easily. The app should generate payment links or request codes that can be shared via messaging or other communication channels.

5. Transaction History: The app should maintain a transaction history, displaying details such as transaction amount, date and time, sender/receiver information, and transaction status. Users should be able to search and filter transaction records.

6. Bill Payments: The app should allow users to pay bills for utilities, services, or subscriptions. It should support the integration of billers and service providers, enabling users to view and pay bills directly within the app.

7. UPI Collect: Users should be able to generate UPI Collect requests, allowing others to make payments to them. This feature is particularly useful for merchants or individuals receiving payments for goods, services, or donations.

8. Customer Support: The app should provide a customer support mechanism, such as in-app chat or helpline numbers, to assist users in case of any issues or inquiries related to transactions, account management, or app usage.

4.3 HARDWARE REQUIREMENTS

1. Device: The UPI app should be compatible with iPhones and iPads running iOS. The specific iOS version compatibility may vary based on the minimum deployment target set by the app developer.

2. Operating System: The UPI app should support the minimum required iOS version specified by the developer. For example, iOS 12 or later.

3. Processor: The device should have a compatible processor that meets the requirements specified by Apple for the supported iOS version. Apple uses different processors in their devices, such as A14 Bionic, A13 Bionic, etc. The UPI app should work smoothly on devices with the supported processors.

4. Memory (RAM): The device should have sufficient RAM to handle the app's requirements and ensure smooth performance. The amount of RAM required can vary depending on the complexity of the app, but typically, devices with at least 2 GB of RAM should be able to run the app efficiently.

5. Storage: The device should have enough storage space to install the UPI app and store any necessary data. The specific storage requirement will depend on the size of the app and any additional data it may need to store. It's recommended to have a device with a minimum of 16 GB of storage.

6. Display: The UPI app should be designed to work well with different screen sizes and resolutions supported by iOS devices. This includes devices with different aspect ratios and resolutions, such as iPhone SE, iPhone XR, iPhone 12 Pro Max, and various iPad models. The app's user interface should be responsive and adapt to different screen sizes without any issues.

7. Connectivity: The device should have the necessary connectivity features to support UPI transactions. This includes Wi-Fi and/or cellular data capabilities to connect to the internet and establish secure connections for financial transactions.

4.4 SOFTWARE REQUIREMENTS

1. iOS Version: The UPI app should be compatible with a specific minimum version of the iOS operating system. The minimum iOS version should be determined based on the target audience and market analysis. For example, the app may require iOS 12 or later.

2. Xcode: Xcode is the integrated development environment (IDE) used for iOS app development. The app developers should use the latest stable version of Xcode to develop and build the UPI app.

3. Programming Language: The UPI app for iOS is typically developed using Swift or Objective-C programming languages. The specific language used depends on the developer's preference and the existing codebase.

4. UPI SDK or API: The UPI app may require integration with UPI-specific software development kits (SDKs) or APIs provided by the UPI platform or the respective banks. These SDKs or APIs facilitate secure transaction processing and communication with the UPI infrastructure.

5. Networking Frameworks: iOS UPI apps often rely on networking frameworks such as URLSession or Alamofire to establish secure connections with servers and handle data transmission for transactions.

6. Database Frameworks: Depending on the app's requirements, it may use database frameworks such as Core Data or Realm to store and manage user and transaction data locally on the device.

7. Security Frameworks: To ensure the security of UPI transactions and user data, the app may utilize iOS security frameworks such as Keychain Services, Secure Enclave, and Touch ID/Face ID for authentication and data encryption.

8. Payment Gateway Integration: If the UPI app allows users to link credit/debit cards or bank accounts, it may require integration with a payment gateway provider's SDK or API for secure payment processing.

CHAPTER 5: SYSTEM DESIGN

This chapter describes the data flow and the Interface design of the project. It includes the overall journey of a user upon opening the application till making a payment.

5.1 SYSTEM DESIGN & METHODOLOGY

Journey Map: This journey map shows all the steps or processes that a user will encounter upon opening the swiftPocket application.

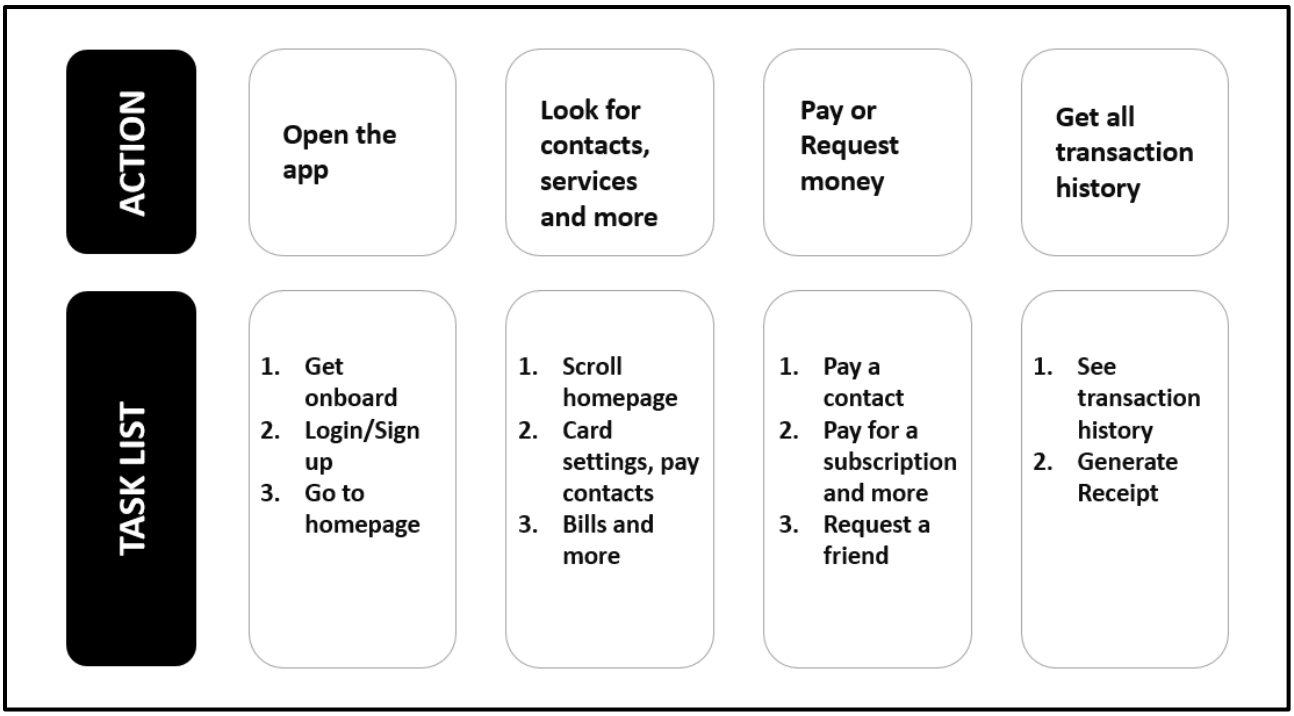


Fig. 5.1 Journey Map (swiftPocket)

UPI Architecture: This is a standard diagram of ho UPI works in the back-end and it shows all the bodies that are a part of thus architecture. This architecture is proposed by NPCI and all the UPI apps follow the same.

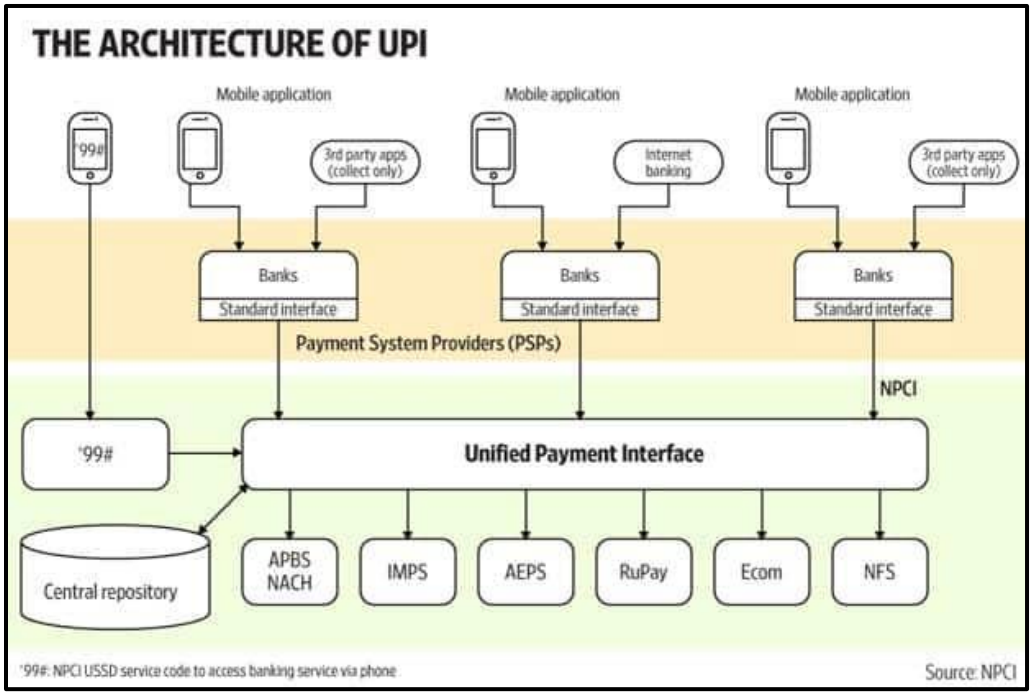


Fig. 5.2 UPI Architecture by NPCI

Application Flow: This diagram shows a set of activities that will take place according to the user's status, meaning if the user is registered or not.

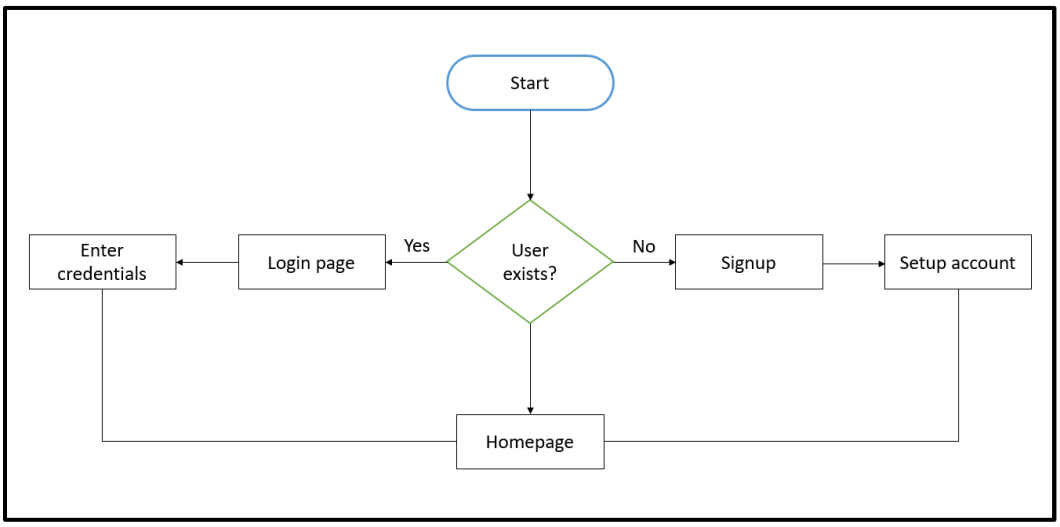


Fig. 5.3 Application Flow

Main Activity: This diagram shows a set of activities that will take place upon reaching homepage of the application.

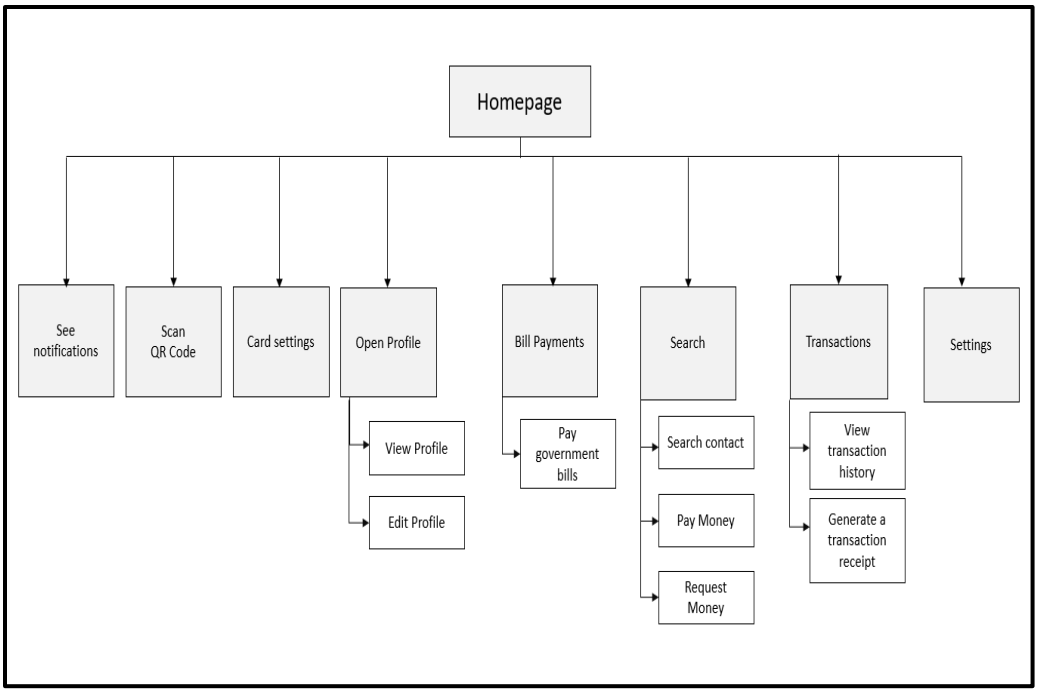


Fig. 5.4 Main Activity Flow

Application Architecture: This diagram shows the MVC (Model-View-Controller) architecture that is followed to design an iOS application.

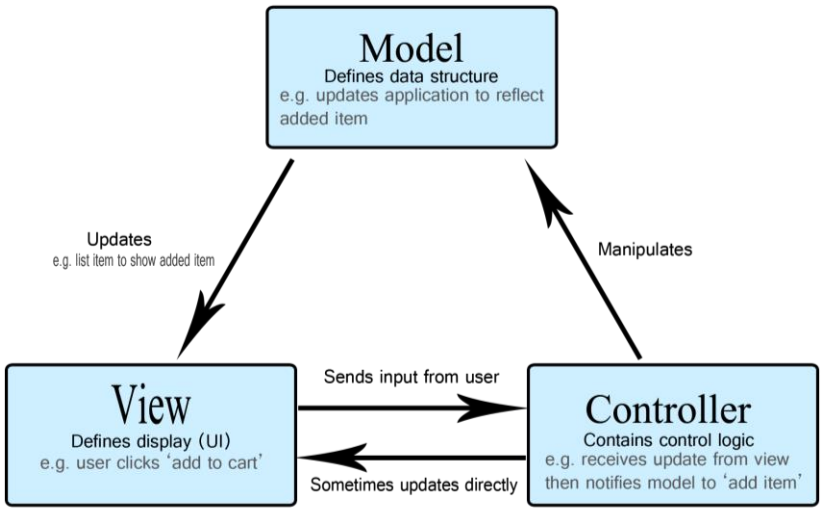


Fig. 5.5 MVC Architecture

CHAPTER 6: IMPLEMENTATION

User Interface:

Onboarding:

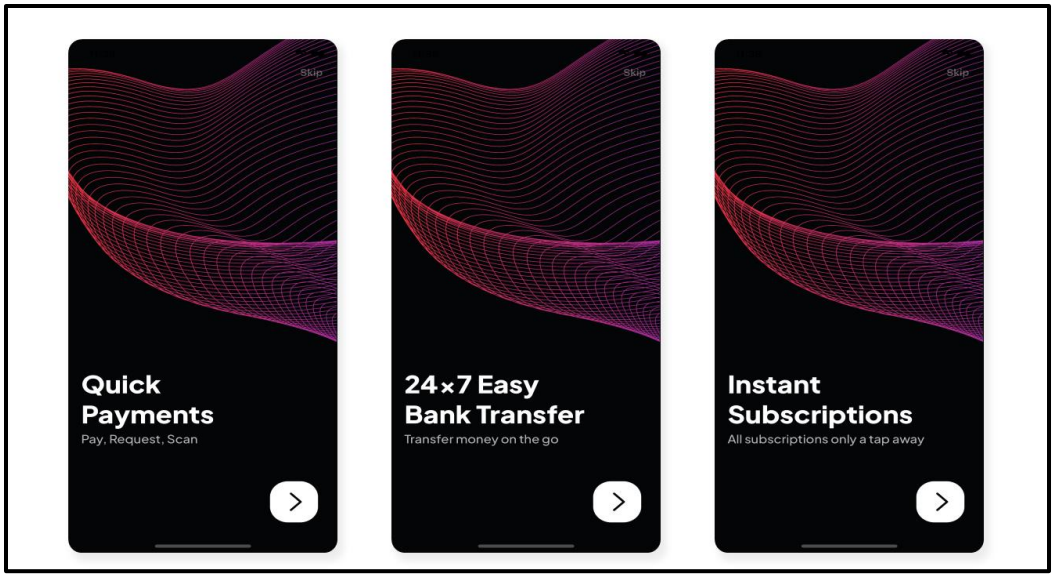


Fig. 6.1 Onboarding Screens

Sign Up:

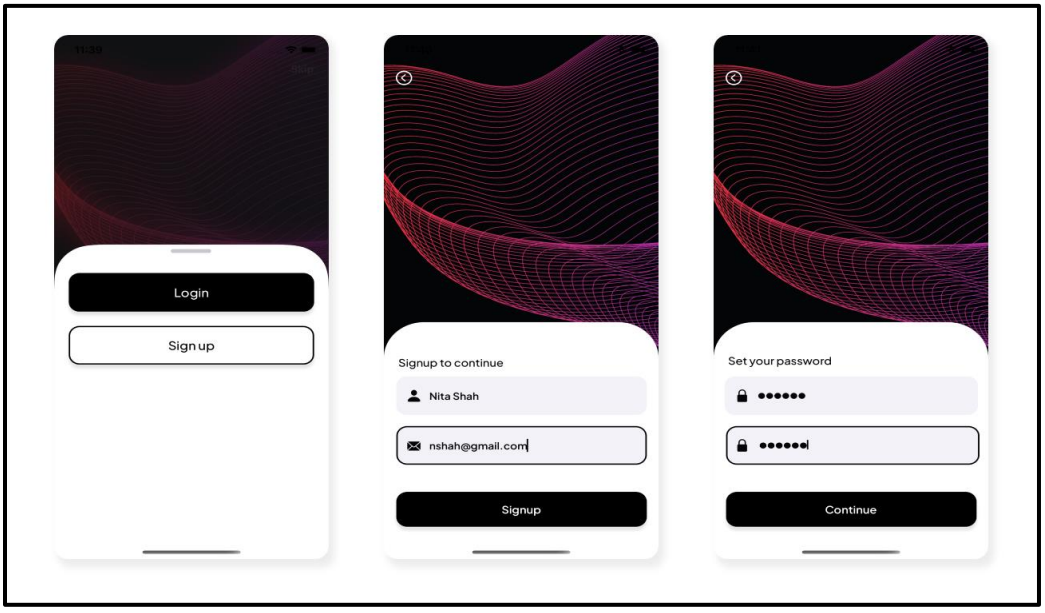


Fig. 6.2 Sign up

Setup Account:

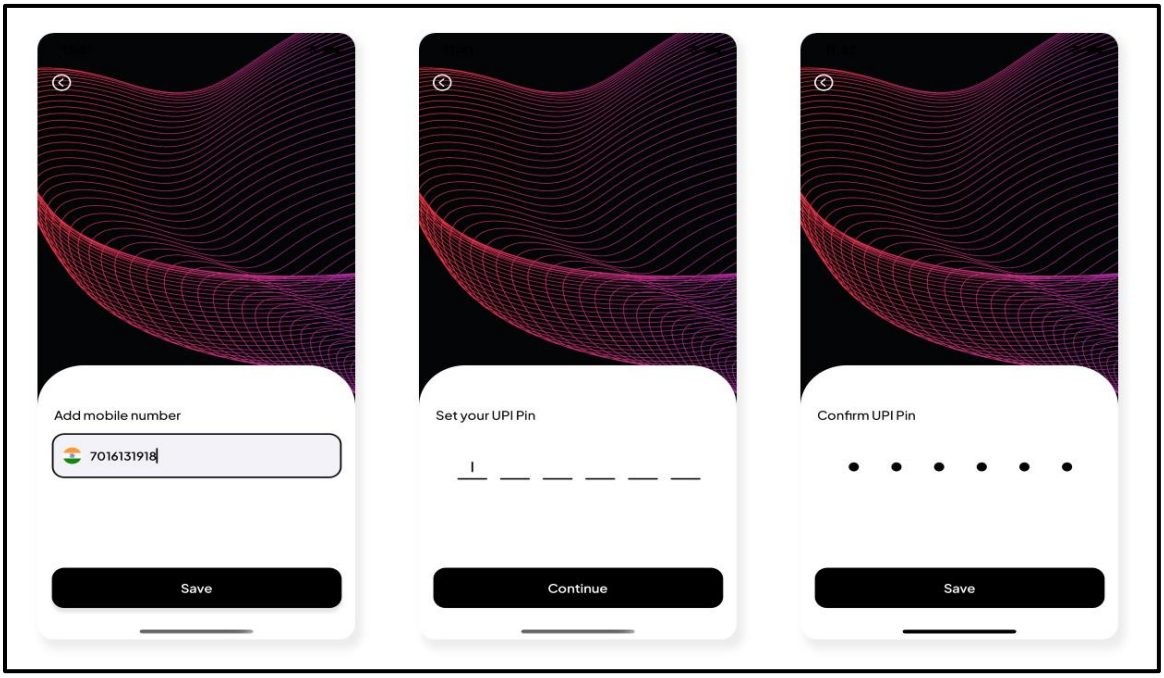


Fig. 6.3 Setup Account Screens

Add Cards:

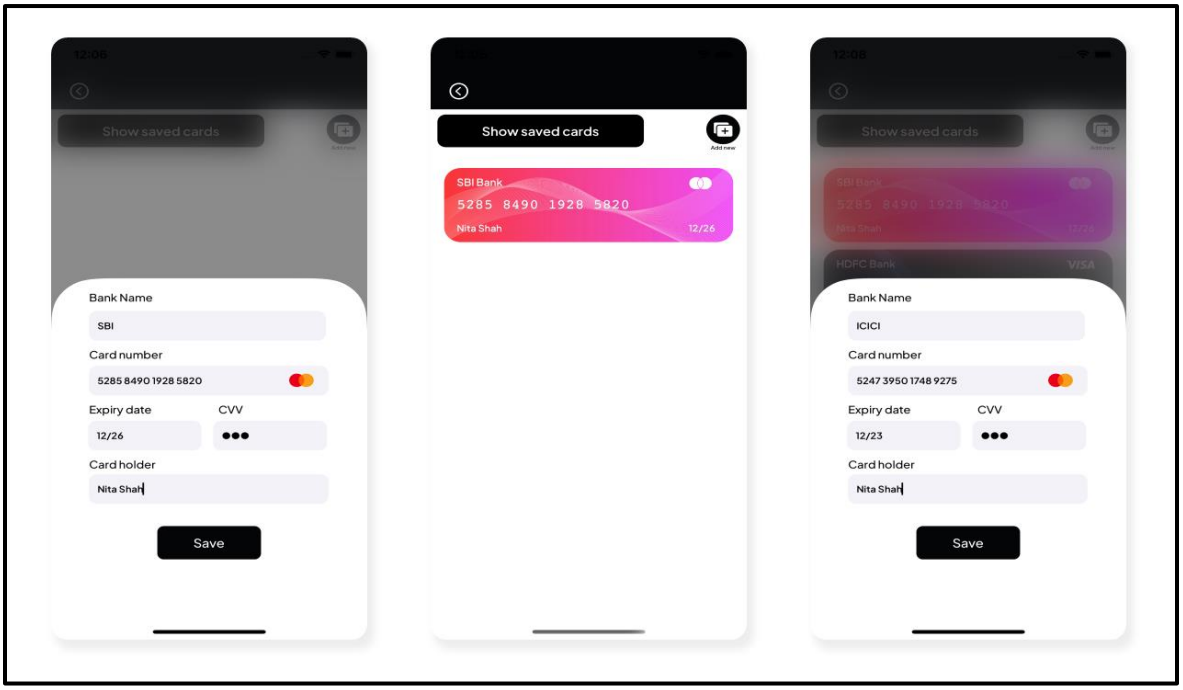


Fig. 6.4 Add card by filling details

Editing profile image:

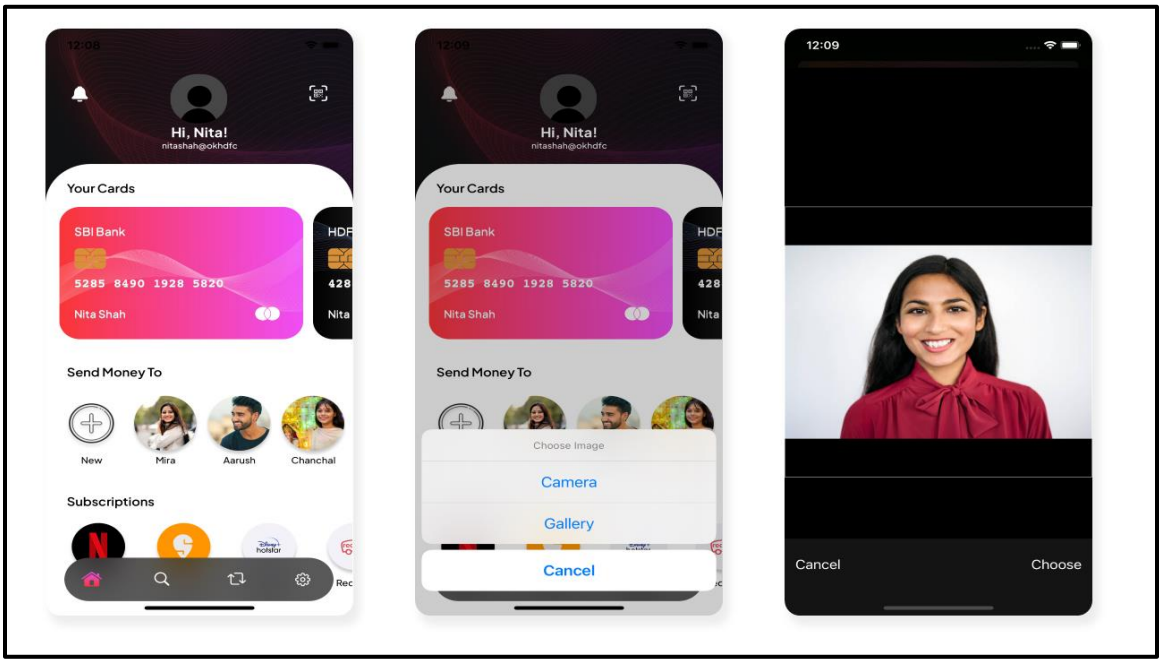


Fig. 6.5 Editing Profile Image

Profile and Edit Profile:

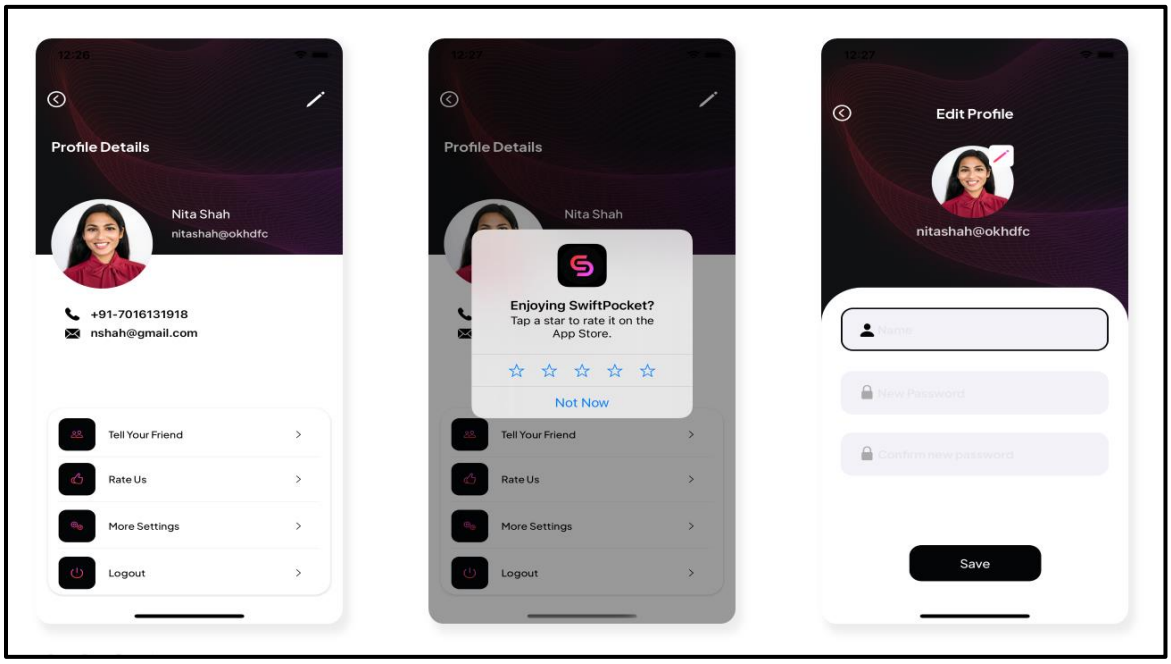


Fig. 6.6 Profile and edit profile page

Application Homepage:

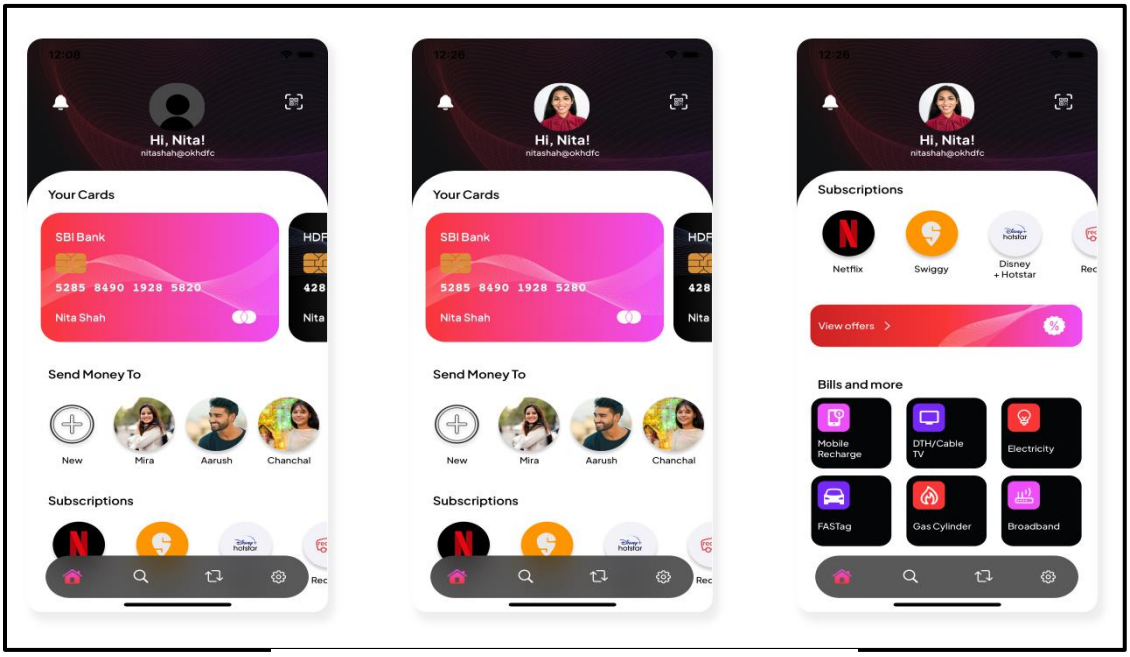


Fig. 6.7 Homepage

Card settings and bill payment screens:

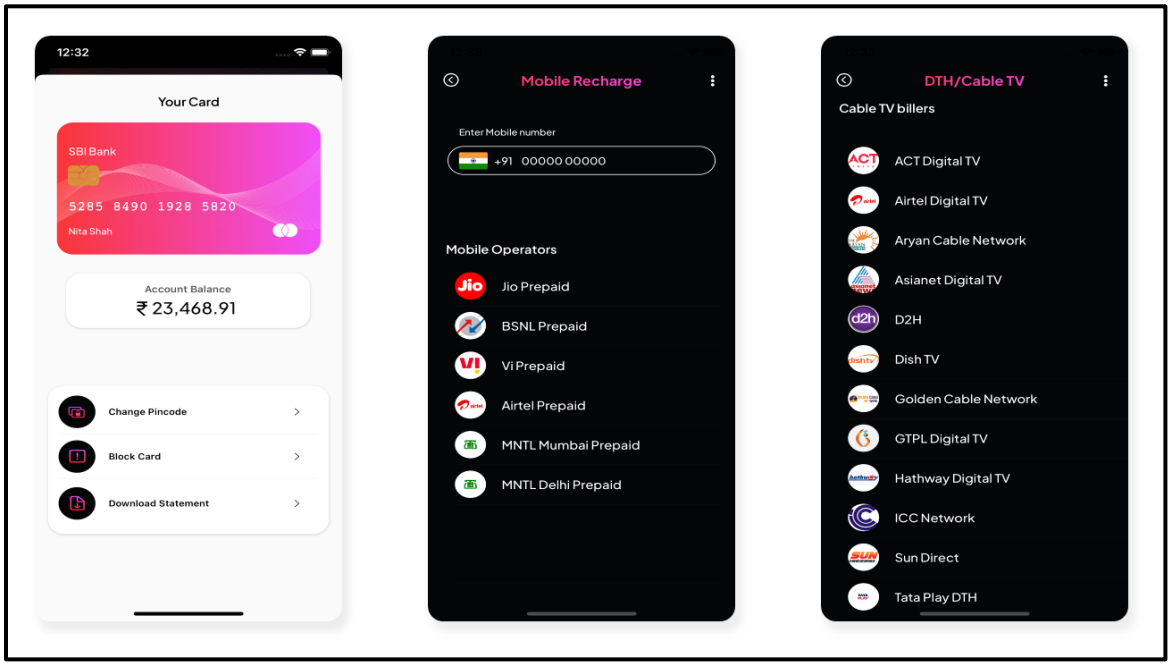


Fig. 6.8 Card settings and bill payment

Settings:

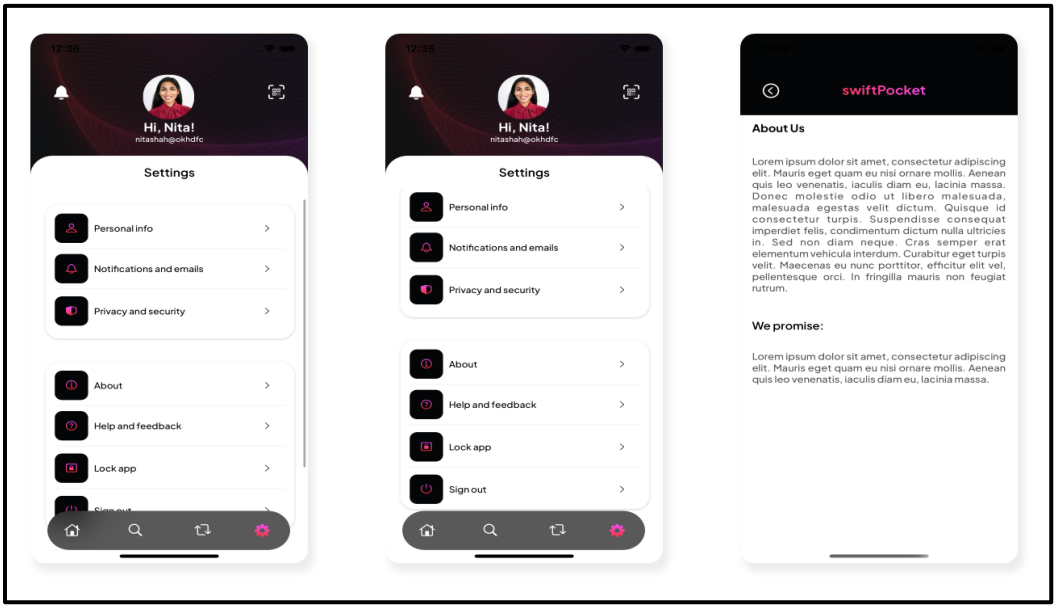


Fig. 6.9 Settings

Transactions and Receipt:

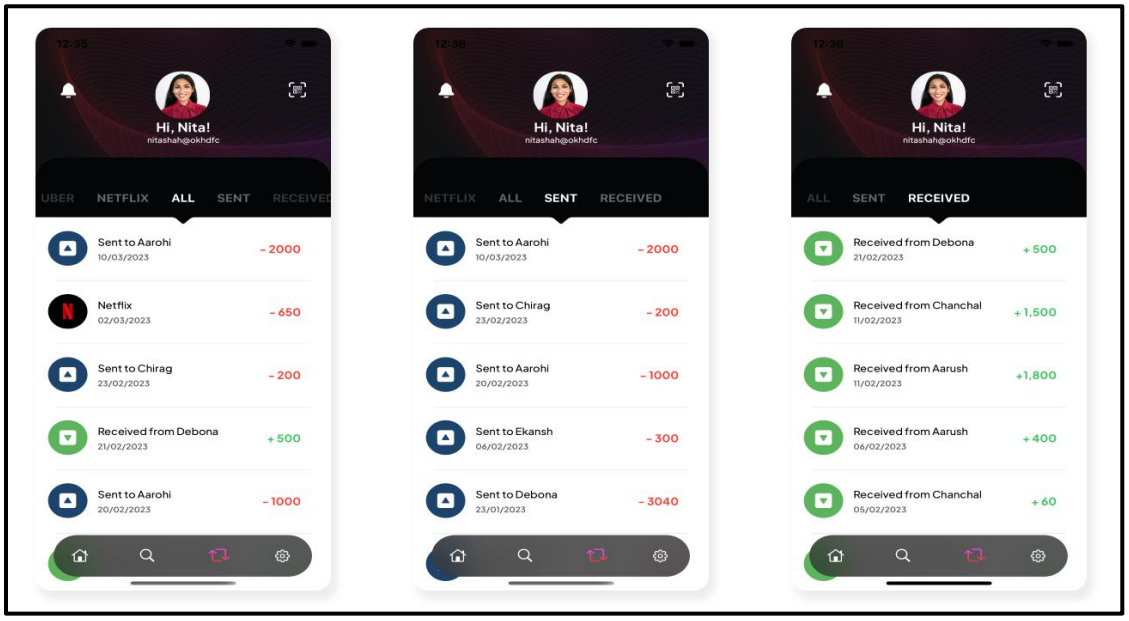


Fig. 6.10 Transaction history

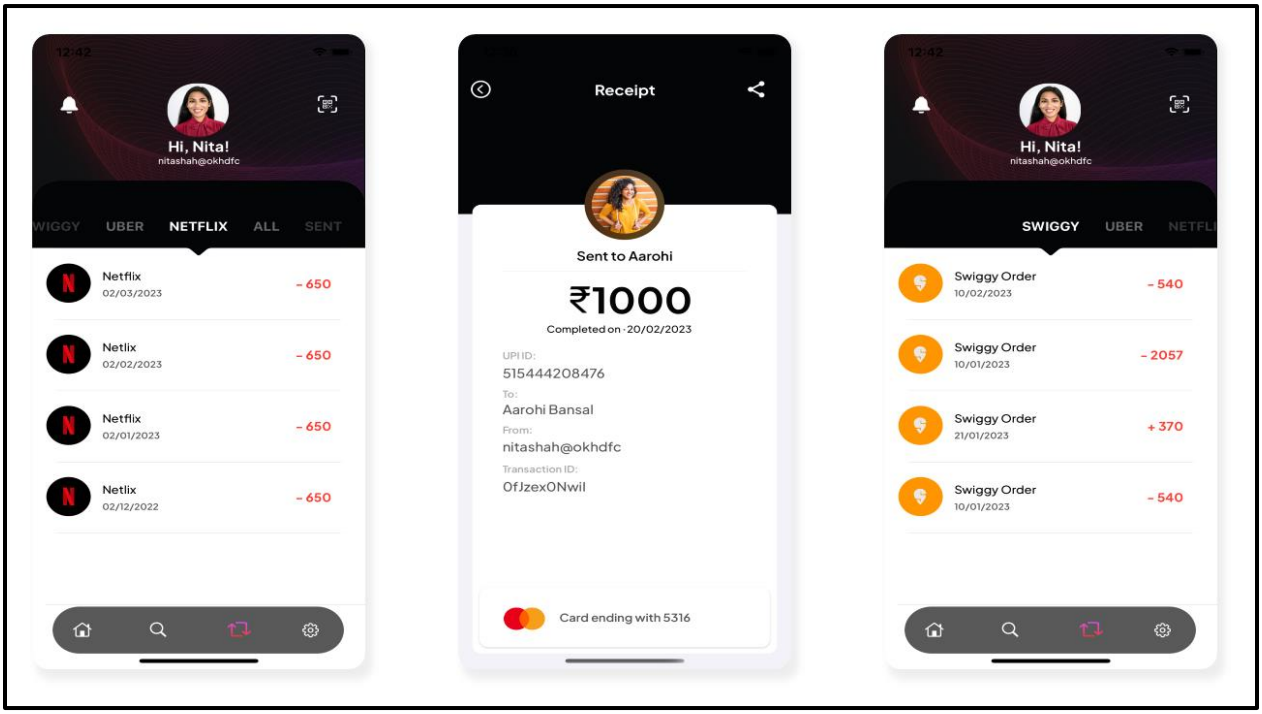


Fig. 6.11 Transaction receipt

Search Contacts:

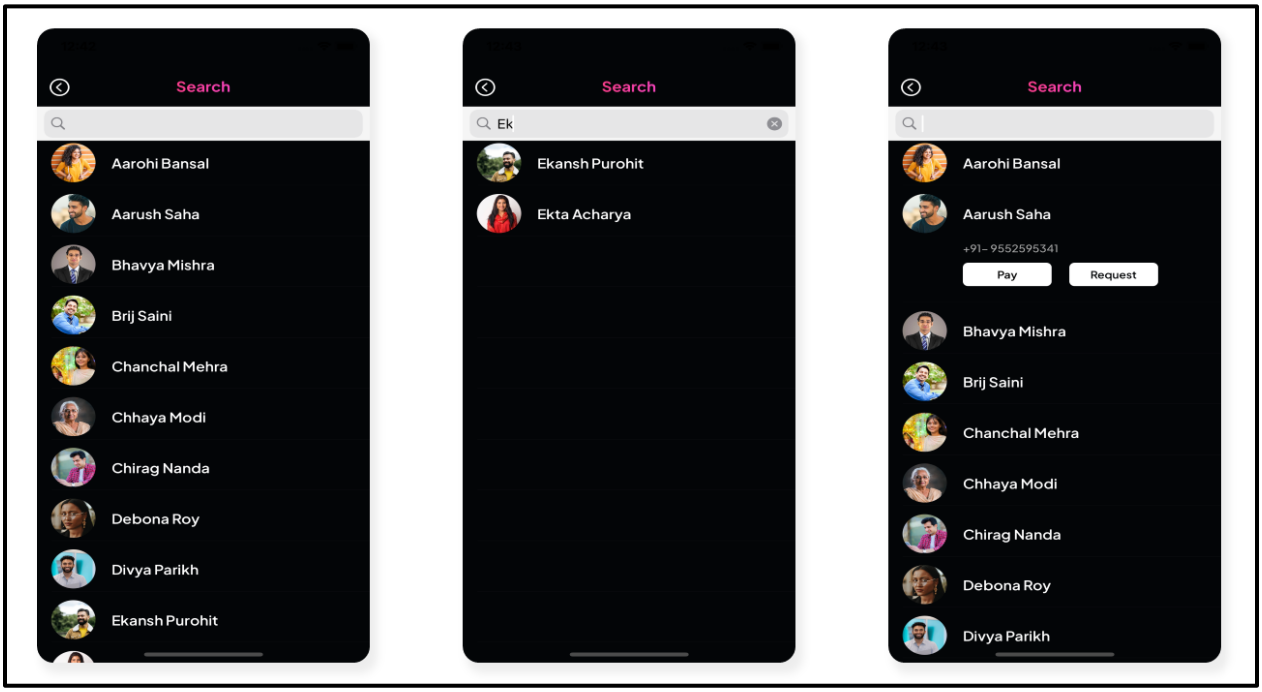


Fig. 6.12 Search Contacts

Making a payment:

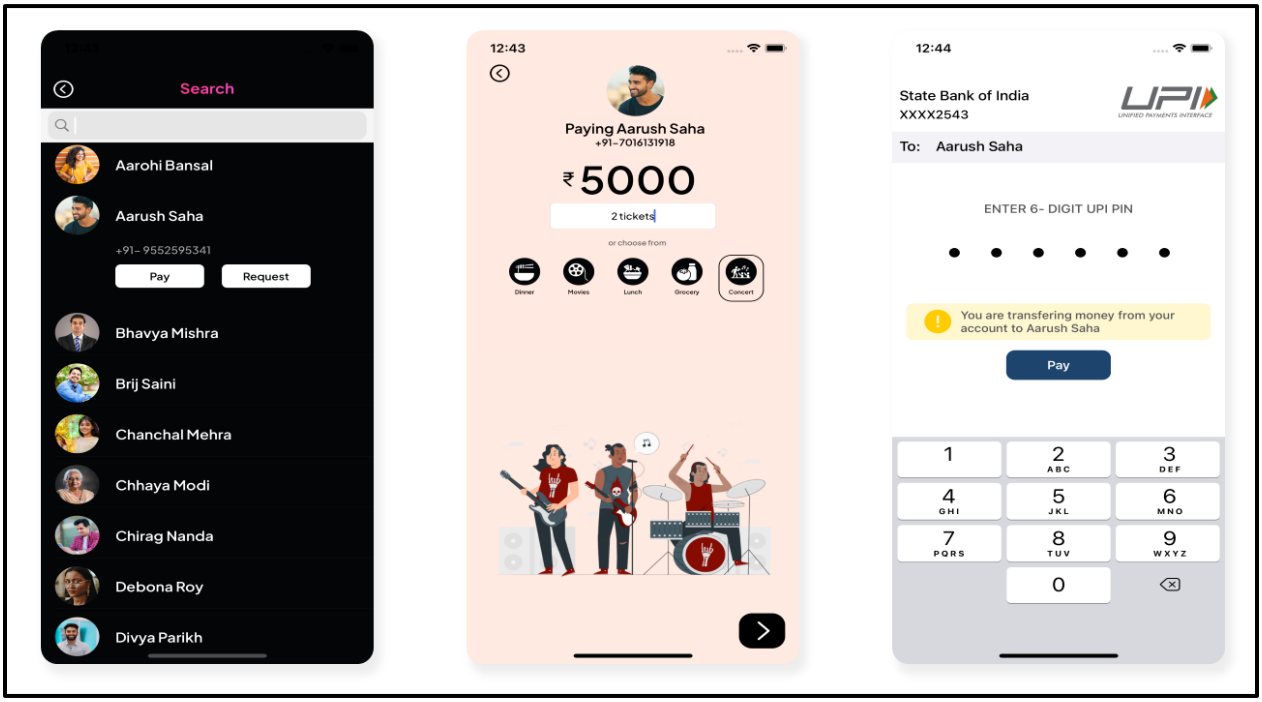


Fig. 6.13 Paying to a peer

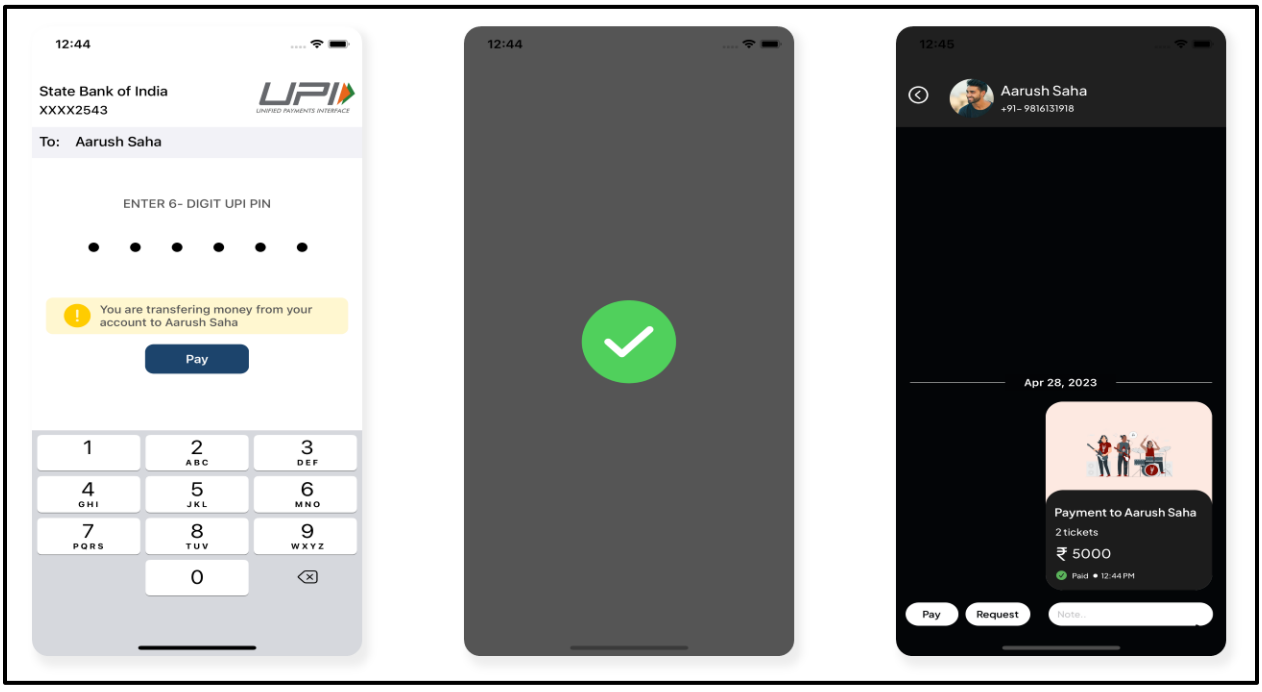


Fig. 6.14 Successful payment

App architecture:

MVC Files: An iOS application uses MVC architecture i.e. Model-View-Controller. Models are files that us throughout the project like some constants or global functions. Views are objects that are used commonly throughout the project like a table cell or a card from. Controller or more specifically view controller for iOS refer to the entire screen that a user sees on the phone. It declares how one page will navigate to another and more.

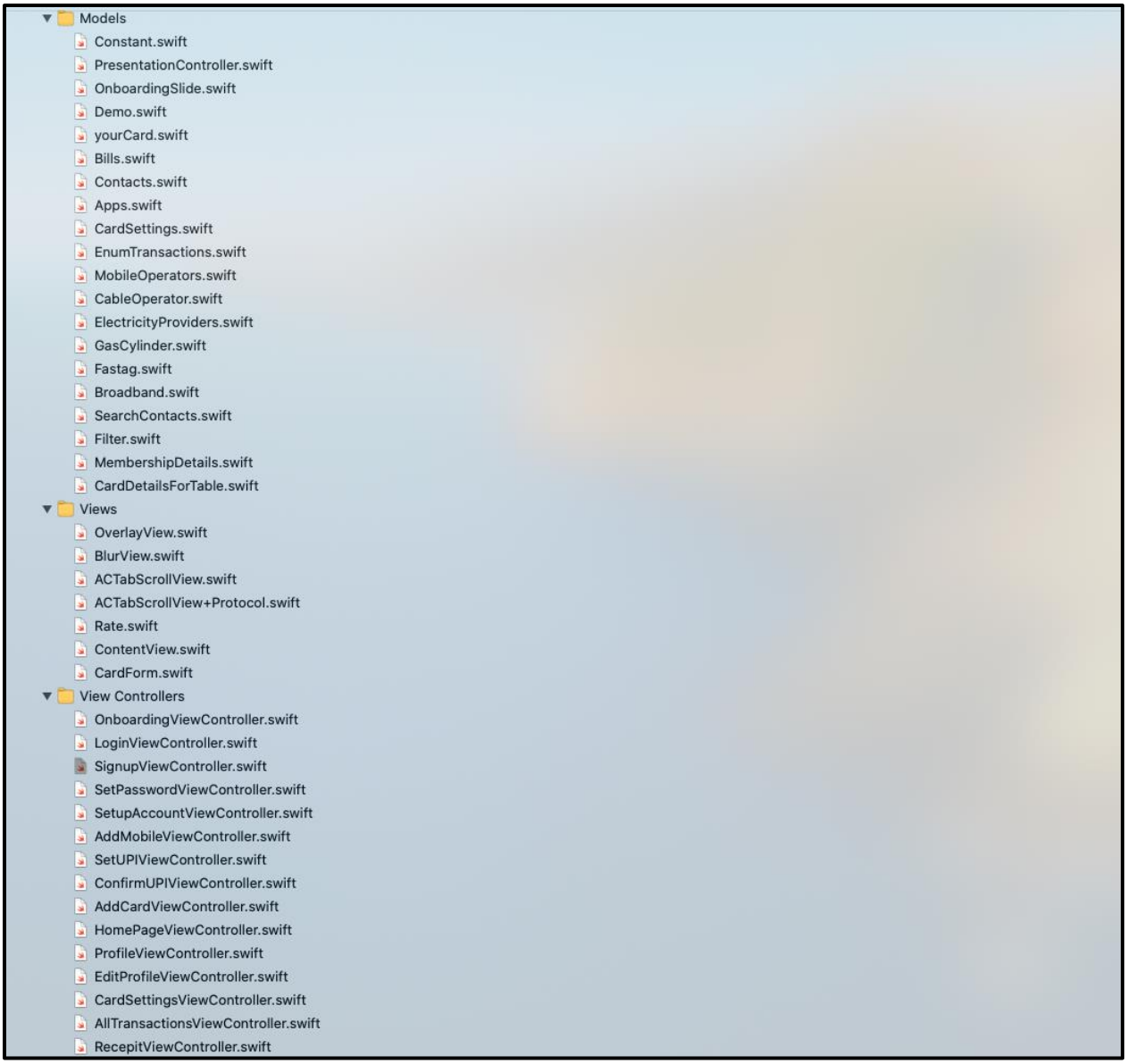


Fig. 6.15 MVC files

Model files for validation:

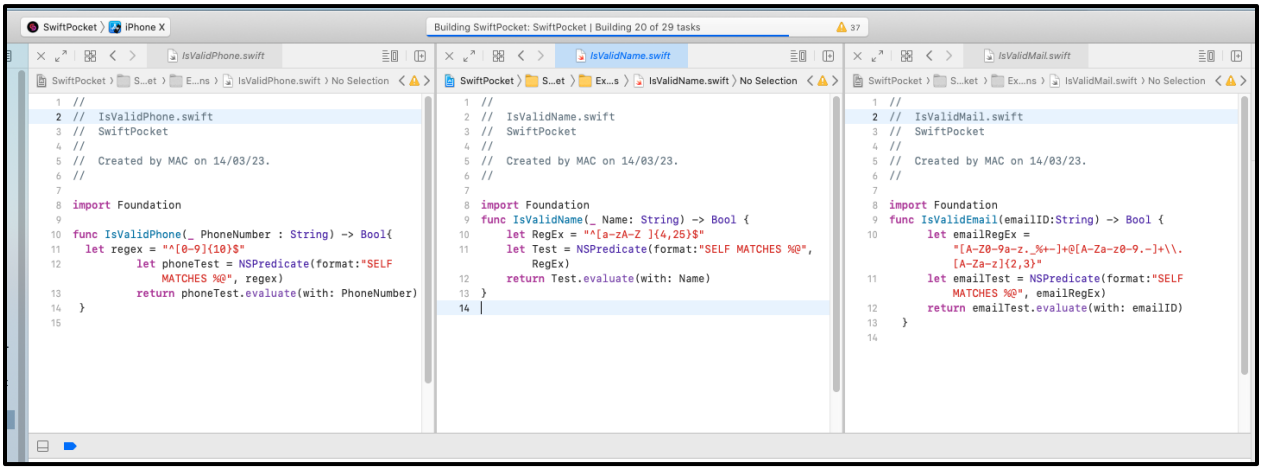


Fig. 6.16 Models

View files:

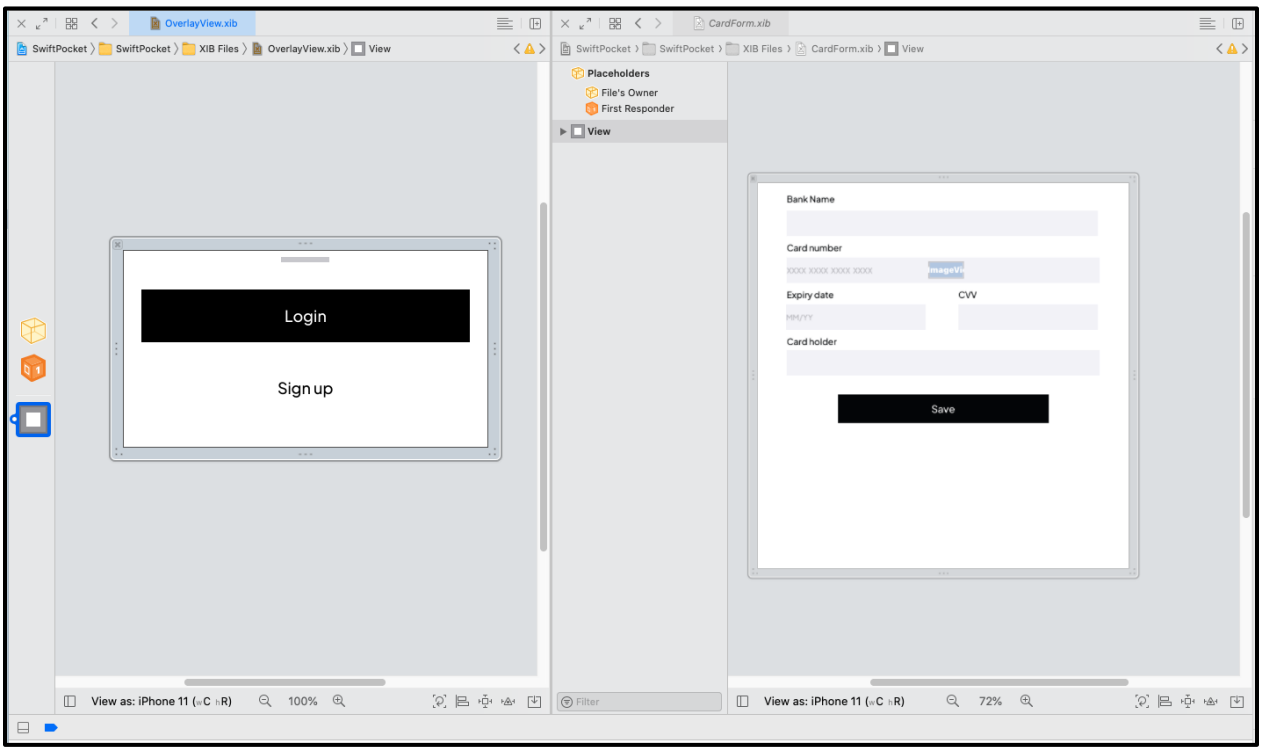


Fig. 6.17 Views

Controllers in storyboard:

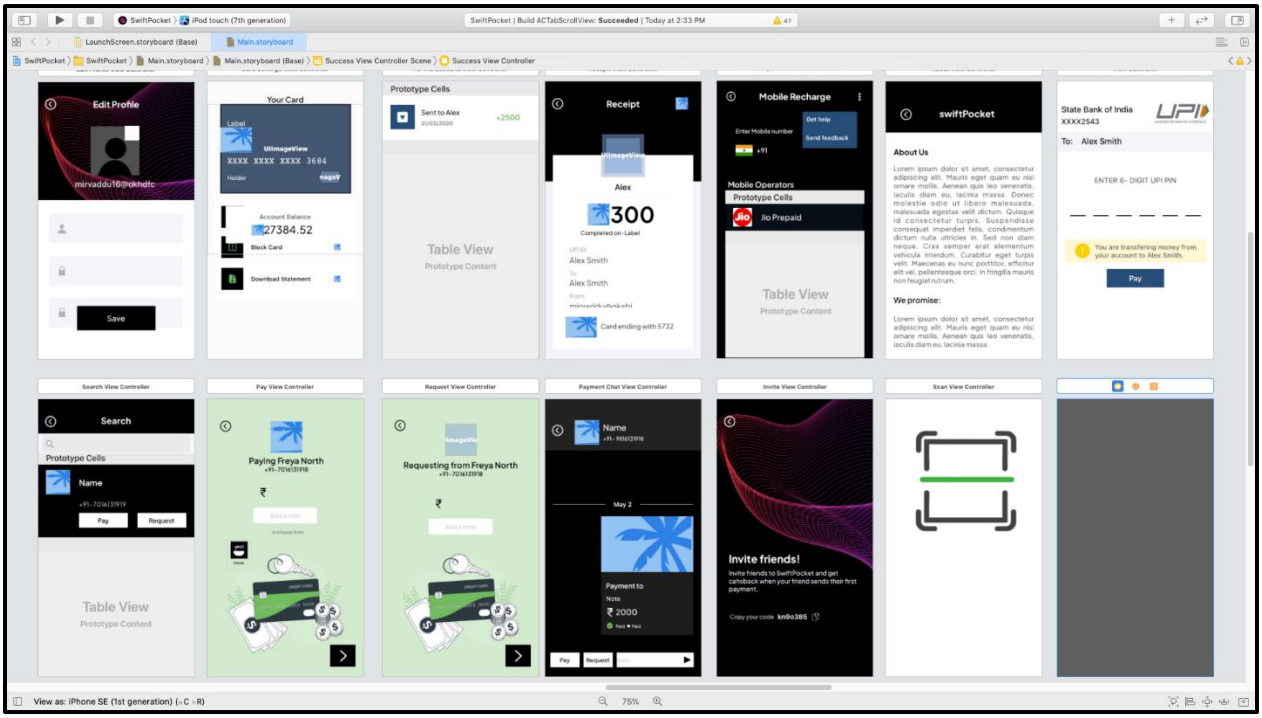
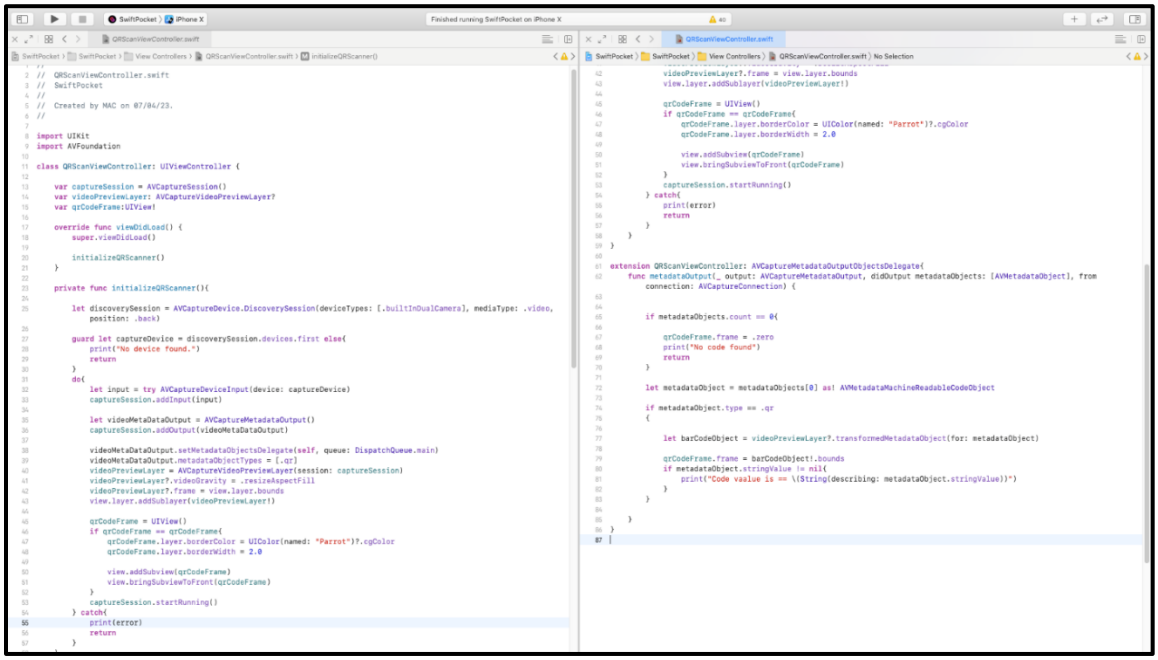


Fig. 6.18 Storyboard

Additional Code for Scanning QR:



```
//
//  QRScanViewController.swift
//  SwiftPocket
//
//  Created by MAC on 07/06/23.
//

import UIKit
import AVFoundation

class QRScanViewController: UIViewController {

    var captureSession = AVCaptureSession()
    var videoPreviewLayer: AVCaptureVideoPreviewLayer?
    var qrCodeFrame:UIView!

    override func viewDidLoad() {
        super.viewDidLoad()

        initializeQRScanner()
    }

    private func initializeQRScanner(){

        let discoverySession = AVCaptureDevice.DiscoverySession(deviceTypes: [.builtInDualCamera], mediaType: .video,
            position: .back)

        guard let captureDevice = discoverySession.devices.first else{
            print("No device found.")
            return
        }
        do{
            let input = try AVCaptureDeviceInput(device: captureDevice)
            captureSession.addInput(input)

            let videoMetaDataOutput = AVCaptureMetadataOutput()
            captureSession.addOutput(videoMetaDataOutput)

            videoMetaDataOutput.setMetadataObjectsDelegate(self, queue: DispatchQueue.main)
            videoMetaDataOutput.metadataObjectTypes = [.qr]
            videoPreviewLayer = AVCaptureVideoPreviewLayer(session: captureSession)
            videoPreviewLayer?.videoGravity = .resizeAspectFill
            videoPreviewLayer?.frame = view.layer.bounds
            view.layer.addSublayer(videoPreviewLayer!)

            qrCodeFrame = UIView()
            if qrCodeFrame == qrCodeFrame{
                qrCodeFrame.layer.borderColor = UIColor(named: "Parrot")?.cgColor
                qrCodeFrame.layer.borderWidth = 2.0

                view.addSubview(qrCodeFrame)
                view.bringSubviewToFront(qrCodeFrame)
            }
            captureSession.startRunning()
        } catch{
            print(error)
            return
        }
    }
}

extension QRScanViewController: AVCaptureMetadataOutputObjectsDelegate{
    func metadataOutput(_ output: AVCaptureMetadataOutput, didOutput metadataObjects: [AVMetadataObject], from
        connection: AVCaptureConnection) {


        if metadataObjects.count == 0{

            qrCodeFrame.frame = .zero
            print("No code found")
            return
        }

        let metadataObject = metadataObjects[0] as! AVMetadataMachineReadableCodeObject

        if metadataObject.type == .qr
        {

            let barCodeObject = videoPreviewLayer?.transformedMetadataObject(for: metadataObject)

            qrCodeFrame.frame = barCodeObject!.bounds
            if metadataObject.stringValue != nil{
                print("Code value is == \($String(describing: metadataObject.stringValue))")
            }
        }

    }
}
```

Fig. 6.19 QR Scan Code

CHAPTER 7: PROJECT REQUIREMENTS

This is an iOS UPI payment application and it needs a lot of software and hardware requirements to be met in order to launch and use it. This chapter discusses the requirements for both developers and users separately. Despite this, the application is user-friendly and super simple to use.

1. Figma

Figma is a cloud-based design tool that allows designers to create and collaborate on UI/UX designs for iOS apps. With its intuitive interface and powerful features, Figma streamlines the design process and makes it easy for teams to work together in real-time. Figma offers a range of tools for designing iOS app interfaces, including vector editing, layout grids, and responsive design features. Designers can also easily create interactive prototypes and animations to test and refine their designs. One of the key benefits of Figma for iOS app design is its ability to create and manage design systems. This allows designers to create a library of reusable components, styles, and assets that can be easily accessed and updated across multiple designs. Figma also integrates with other design tools and services, such as Sketch, Adobe Creative Cloud, and Zeplin, making it easy to work with existing design workflows and share designs with developers and stakeholders. Overall, Figma is a powerful and versatile tool for designing iOS app interfaces that can help designers streamline their workflow and collaborate effectively with their team.

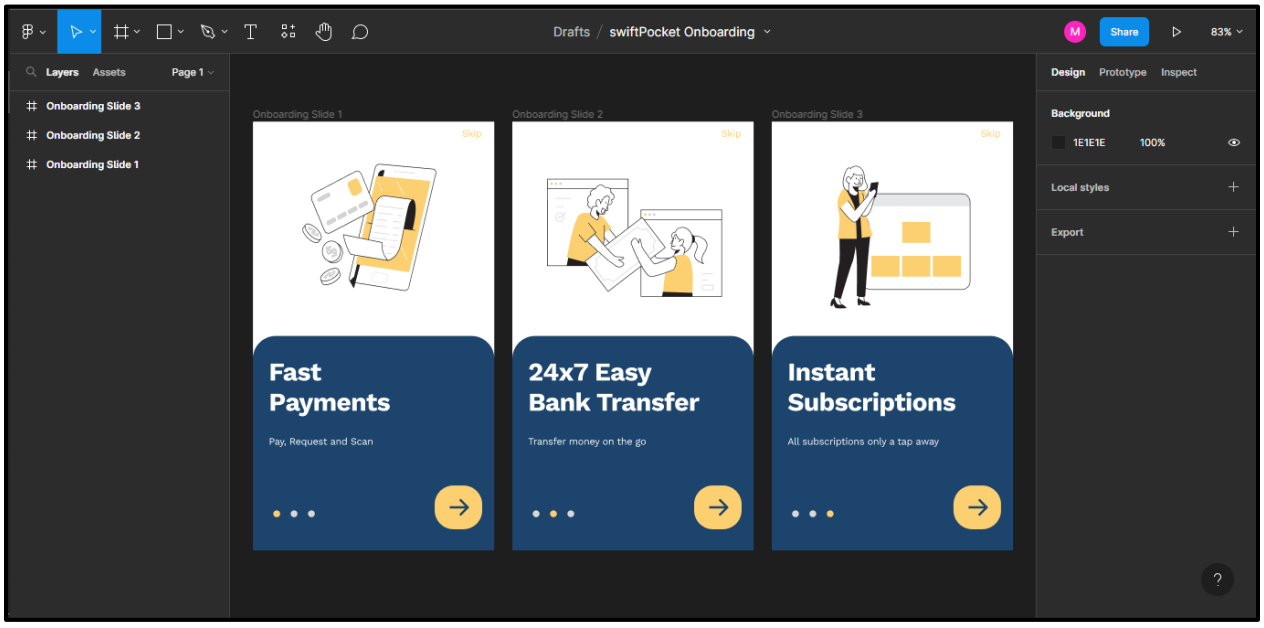


Fig. 7.1 Figma Prototype

2. XCode

Xcode is an integrated development environment (IDE) for building software for macOS, iOS, watchOS, and tvOS. It includes a suite of tools and resources that developers use to create, test, and distribute their applications for Apple's platforms.

Xcode provides a graphical interface and a set of tools for writing, debugging, and optimizing code. It includes a code editor, a graphical user interface (GUI) editor, a simulator for testing apps on various devices, a debugger, and a profiler. Xcode also includes frameworks, libraries, and templates to help developers create high-quality, efficient, and scalable applications. One of the key features of Xcode is its support for multiple programming languages, including Swift, Objective-C, C++, and AppleScript. This allows developers to choose the language that best suits their needs and preferences.

In addition to its development tools, Xcode also provides tools for managing the distribution of applications, including app signing, provisioning profiles, and App Store submission. Overall, Xcode is a comprehensive and powerful development environment that enables developers to create high-quality applications for Apple's platforms.

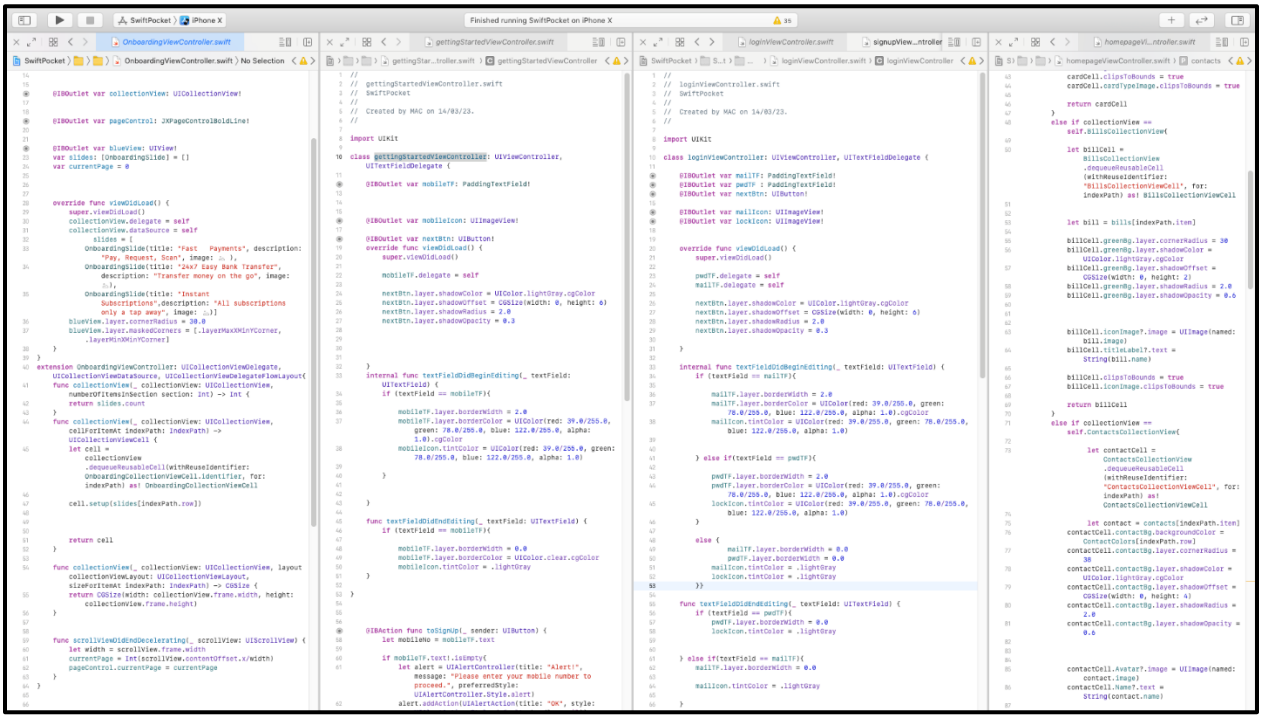


Fig. 7.2 XCode

3. CocoaTouch Framework

CocoaTouch is a framework for developing iOS applications. It provides a set of APIs and tools for building user interfaces, managing user interactions, and accessing device-specific features such as the camera, accelerometer, and GPS. CocoaTouch is built on top of the Core Foundation and Foundation frameworks, which provide low-level functionality such as data types, collections, and memory management. The framework includes classes for creating views, handling touch events, managing navigation, and displaying media content. It also includes support for network communication, data storage, and localization. CocoaTouch is a key component of iOS development, enabling developers to create powerful and engaging applications that take full advantage of the capabilities of the iPhone, iPad, and iPod touch.

4. Swift Language

Swift is a programming language developed by Apple for building software applications for its platforms, including iOS, macOS, watchOS, and tvOS. It was first introduced in 2014 as a replacement for Objective-C.Swift is designed to be fast, safe, and easy to use, with a syntax that is concise and expressive. It includes features such as type inference, optionals, and closures that make it easier to write code that is both efficient and readable.

Swift is also designed to work seamlessly with existing Objective-C code, allowing developers to easily integrate Swift into their existing projects. It is an open-source language, with a growing community of developers contributing to its ongoing development and improvement. Overall, Swift is a powerful and versatile programming language that enables developers to create high-quality applications for Apple's platforms with greater ease and efficiency.

CHAPTER 8: CONCLUSION

In conclusion, SwiftPocket is a state-of-the-art iOS UPI app that provides a comprehensive solution for managing finances, making payments, and transferring funds on iPhones. It addresses the pain points associated with traditional financial management methods, such as complexity, inefficiency, and limited accessibility. With its user-friendly interface, advanced security measures, and extensive UPI bank integration, SwiftPocket offers a seamless and secure platform for managing financial transactions.

The app's features, such as real-time notifications, QR code scanning for payments, and integration with digital wallets, make it a comprehensive solution for users seeking a hassle-free financial management experience. SwiftPocket is a valuable addition to the financial technology space, empowering users with a single application that enables them to stay in control of their finances.

Overall, SwiftPocket is a game-changer in the financial management industry, providing a seamless and secure platform for iOS users to manage their financial activities with ease. As the demand for mobile payments continues to grow, SwiftPocket is well-positioned to meet the needs of users seeking a reliable and intuitive financial management solution.

8.1 LEARNING OUTCOMES

Working as an iOS app developer at Swayam Infotech can offer several learning outcomes that can help enhance one's technical and professional skills. Firstly, the job can provide an opportunity to work on a diverse range of projects, from simple utility apps to complex enterprise-level applications. This variety of work can help expand one's technical knowledge and expertise, and improve problem-solving and critical thinking skills.

Secondly, working as an iOS app developer can help individuals develop their proficiency in programming languages such as Swift, Objective-C, and Xcode. Through hands-on experience, developers can deepen their understanding of these languages and their application in building high-quality iOS apps.

Thirdly, the role can provide an opportunity to work in a collaborative environment and gain experience working in a team. Developers can learn how to work with project managers, designers, and other team members, improving their communication skills and ability to work in a team. Developers can stay up-to-date with the latest technologies, design patterns, and methodologies, which can help enhance their skills and knowledge and make them more competitive in the job market.

8.2 RESULTS

SwiftPocket is a financial management app for iOS devices that offers a range of features to simplify financial transactions and enhance the overall user experience. Some of the key features of SwiftPocket include:

1. Integration with UPI-enabled banks: SwiftPocket is integrated with a wide range of UPI-enabled banks, making it easy for users to link their bank accounts and perform financial transactions such as money transfers and bill payments.

2. User-friendly interface: The app has a simple and intuitive user interface that allows users to manage their finances effortlessly. The app's design and navigation are optimized for ease of use, making it a user-friendly financial management solution.

3. Advanced security measures: SwiftPocket offers advanced security measures to ensure secure financial transactions. The app supports biometric authentication, transaction limits, and real-time notifications, providing users with peace of mind when using the app for financial transactions.

4. Bill payments: The app allows users to pay their bills directly from their bank accounts, eliminating the need to visit bill payment centers or use other payment methods.

CHAPTER 9: REFERENCES

1. Apple Developer Community: https://developer.apple.com/
2. Stack Overflow Community: https://stackoverflow.com/
3. Swift Programming: https://www.programiz.com/swift-programming
4. iOS Dev Weekly Newsletter: https://iosdevweekly.com/
5. AppCoda Tutorials: https://www.appcoda.com/tutorials/ios/
6. Dribbble iOS App Design Inspiration: https://dribbble.com/search/ios%20app
7. . Figma Resources for iOS UI Design: https://www.figmaresources.com/resources/tag/ios