

# **ABFL-EMICard in iOS Application Development**

Project report submitted in partial fulfillment of the requirement  
for the degree of Bachelor of Technology

in

**Computer Science and Engineering**

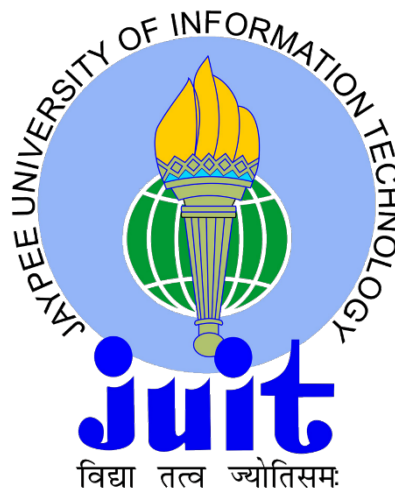
By

Haider Ali (141324)

Under the supervision of

Purushottam Naveen Chandra

To



Department of Computer Science & Engineering and Information  
Technology

**Jaypee University of Information Technology, Wagnaghat,  
Solan-173234, Himachal Pradesh**

## CANDIDATES' DECLARATION

I hereby declare that the work presented in this report entitled “**ABFL-EMICard**” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** submitted in the department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat, is an authentic record of my own work carried out over a period from February 2018 to May 2018 under the supervision of **Mr. Purushottam Naveen Chandra**, Developer , iOS Department.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Haider Ali (141324)

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Mr. Purushottam Naveen Chandra

Developer

iOS Department

Dated:

## **ACKNOWLEDGEMENT**

I would like to thank and convey my deep sense of gratitude to my project guide Mr. Purushottam Naveen Chandra and my mentor Mr. Ansu Jain for their huge assistance and valuable counselling without which my final year project would lack all the necessary components.

I would also like to thank my parents and friends for their huge support and faith bestowed upon me. I consider it as an important duty to put on record my sincere and honest gratitude to my project guide for his foresightedness, knowledge and assistance which helped me in tackling complex components of my project in a very logical and optimized way.

# TABLE OF CONTENTS

## TABLE OF CONTENTS

<b>Candidate's Declaration .....</b>	<b>(i)</b>
<b>Acknowledgement .....</b>	<b>(ii)</b>
<b>Table of Contents .....</b>	<b>(iii)</b>
<b>List of Figures.....</b>	<b>(iv)</b>
<b>Abstract .....</b>	<b>(vi)</b>
<b>1. Introduction .....</b>	<b>(1)</b>
<b>2. Literature Survey.....</b>	<b>(3)</b>
<b>3. System Development .....</b>	<b>(11)</b>
<b>4. Performance Analysis.....</b>	<b>(45)</b>
<b>5. Conclusions.....</b>	<b>(46)</b>
<b>References.....</b>	<b>(48)</b>

## LIST OF FIGURES

1.1 Layers of iOS .....	(1)
3.1 Login Screen .....	(11)
3.2 OTP Screen .....	(12)
3.3 Home Screen .....	(13)
3.4 Dashboard Screen.....	(14)
3.5 Service Page Screen .....	(15)
3.6 Transaction Screen .....	(16)
3.7 Transaction PIN Screen .....	(17)
3.8 Forget Transaction PIN Screen.....	(18)
3.9 Forget Transaction PIN Screen-II.....	(19)
3.10 Update Transaction PIN Screen .....	(20)
3.11 Dispute Inquiry.....	(21)
3.12 Hotlisting Screen.....	(22)
3.13 Activate/Deactivate Screen.....	(23)
3.14 Side Menu Screen.....	(24)
3.15 My Profile Screen .....	(25)
3.16 Change Login PIN Screen.....	(26)
3.17 Apply Home Loan Screen .....	(27)
3.18 Line Utilization Screen .....	(28)
3.19 Email Transaction Report Screen .....	(29)
3.20 Forget Login PIN Screen .....	(30)
3.21 Withdraw Screen .....	(31)
3.22 Repayment Screen .....	(32)

3.23 Find IFSC Code Screen .....	(33)
3.24 Completion Screen .....	(34)
3.25 Apply Home Loan Screen .....	(37)
3.26 Network Error Screen .....	(38)
3.27 Transaction Failure Screen .....	(39)
3.28 Transaction Details Screen .....	(40)
3.29 Saved Bank Transaction Screen .....	(41)
3.30 Transaction Confirmation Screen .....	(42)
3.31 Report Sent Screen .....	(43)
3.32 Launch Screen .....	(44)
4.1 Instruments Screen .....	(45)

## **ABSTRACT**

In the present computerized world, having an advanced mobile has turned into a piece of individuals' way of life. Now- a-days, advanced cell phones are not simply utilized for stimulation or calling, but rather they are additionally used for ordering items, deal with bank, documenting tax declaration, following individuals and so on.

In India, android is very overarching yet now-a-days numerous individuals do have iPhone. What's more, in the vast majority of the nations, individuals do utilize i-Phone only. Every app should be created for iPhone and also android. In this way, the task of my internship was to learn advancement of iPhone development improvement utilizing most recent language, Swift 4. I will quickly expound on iOS and its significant segment and fundamental ideas which are used in developing an application.

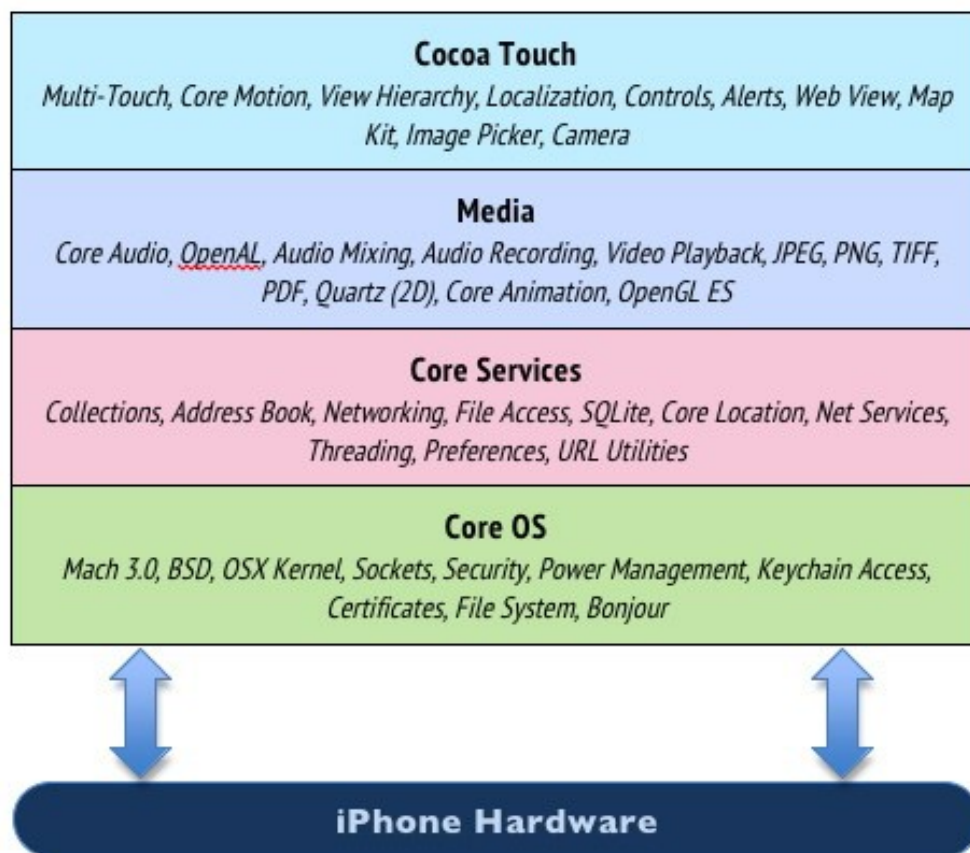
My Internship at Kuliza Technologies began with a learning time. I completed a demo task to get comfortable with the principles and practice of the organization and in addition to learn iOS improvement and then I worked on the live project ABFL-EMICard. This report contains my work of the live venture.

# CHAPTER 1

## INTRODUCTION

### 1.1 Introduction

iOS or iPhone Operating System is a mobile OS planned and created by Apple. furthermore, circulated just for Apple equipment. iOS represents numerous gadgets like i-Phone, i-Pad, i-Pod. I took a shot at improvement of use for i-Phone gadgets. There is some basic system in iOS and OSX like CoreFoundation; be that as it may, it's UI toolbox is CocoaTouch as opposed to OSX's Cocoa. The most important distinction amongst Cocoa and CocoaTouch is of UI-layer. In Macintosh, to create apps, AppleKit is utilized while for i-Phone, to create apps, UI Kit is utilized which contains all the important UI segments which are utilized to build up an application. iOS isn't completely Unix-perfect due to this distinction as it were. Unix look alike shell get to is limited for i-Phone applications.



**Figure 1.1:** Layers of iOS



## **1.2 Problem Statement**

To start with assignment is to comprehend the improvement procedure and accumulate the prerequisites of the task. There are a few inquiries that should be addressed like what design example to take after, what innovation to use to manufacture app and so forth. After everything is set up, the undertaking of outlining the easy to understand screen presents.

The screen comprises of numerous components. To create screen, basic undertaking is to comprehend the components and to build up every single one of them productively. Keeping in mind the end goal to build up a component with a appropriate usefulness, there might be a scenario that we need to utilize outer framework. That framework ought to be productive and occupy limited memory. Everything relies upon the need of venture. While building up a screen, one ought to think about its deployment on changed gadget screen. Creating UI isn't a major assignment yet deployment of the same screen in various is tedious. Along these lines, as per distinctive screen, we have to adjust the limitations in design.

## CHAPTER 2

### LITERATURE SURVEY

#### 2.1 What's in iOS

iOS contains 4 layers - CoreOS, CoreServices, Media, CocoaTouch. CoreOS is close to equipment and the greater part of that to CocoaTouch that is close to client. Essentially in iPhone OS, there is a rendition on UNIX executing on it. Because of this, it has completely multi-tasking UNIX document framework. CoreOS layer has control administration, KeyChain to ensure that individuals are not getting to the information of telephone that they shouldn't. Because we need to write in OOPS way, there is CoreServices layer that is composed in OO Way over Unix. It has no UI at its layer. Then comes MediaLayer that has a wide range of video, sound structures. The most imperative layer that necessities consideration is CocoaTouch, a OO API for building UIs. These UIs dislike catch, slider on screen instead rather it is used to execute when user shakes his telephone or fundamentally when orientations change, animation.

#### 2.2 Platform Component

##### 2.2.1 Tool

I utilized XCode programming to build up the iPhone app. XCode is an Integrated Development Environment with editors, compilers, and other programming devices that cooperate to help compose programming, incorporate it, stack it onto a gadget, investigate it, and eventually deploy it to the application-store. It is a wise programming which gives run-time issues to alarm memory spills. Essential documents in an undertaking incorporate AppDelegate, Storyboard and ViewControllers.

**Application Delegate:** For each app, AppDelegate file is made as a matter of course by Xcode. It is essential record. At the point when an application begins, it initially communicates with AppDelegate. Delegation is an imperative idea in iPhone OS. It is discussed later.

**Storyboard:** StoryBoard is an arrangement of screen in our application. StoryBoard portrays a stream in application. Naturally, two StoryBoards are created when an application is made:

LaunchStoryboard and MainStoryboard. LaunchStoryboard is kept as it is. On the off chance that there are more streams in app, so it is great habit to create various storyboards.

Essentially, for each stream a StoryBoard is made to keep designing great and justifiable.

**View Controller:** ViewControllers are the establishment of any application's interior working. Each application has no less than one ViewController. It deals with an arrangement of perspectives and generates applications UI. It organizes with display items and other ViewControllers. Essentially, it assumes a join part for the two view items and controller-objects. Each ViewController demonstrates its own perspectives to show the application content. The perspectives are consequently stacked when the properties of ViewController is gotten to in the application. iOS gives a few created ViewController classes for creating basic UIs. These are UI-TableViewController, UI-NavigationController, UI-TabBarController and so on.

### 2.2.2 Language

iPhone apps can be produced in excess of one language like in the beginning code was written in Objective-C. A large portion of the frameworks are composed in Objective-C. At that point in 2013, Swift appeared. Consistently another form of Swift is discharged. I am taking a shot at Swift-4 to create applications. Swift is protocol oriented and is extremely easy to understand dialect. Swift has a ton of nice things, for example, easy memory administration, generics and optionals, basic yet tight inheritance protocol. Swift is clean and more meaningful than ObjectiveC.

### 2.2.3 Framework

Framework are packs that contain a importable library and related assets and header for coding. Structure is utilized with the goal that code is reusable, code is typified and adjusted. I utilized for the most part UIKit structure, AVFoundation system. There are numerous more structures are accessible like quartz-outline work for cutting edge liveliness, MapKit system for tweak guide's substance and appearance and so on.

UI-Kit structure is composed in Objective-C. A User Interface unit contains UI components with the end goal of UI outline. These incorporate segments, for example, UIButton, checkbox, ProgressBar, TextView, TextFields, sliders, NavigationBar, switch, and some more.

AVFoundation system benefits an arrangement of Objective C class for playing, recording, and overseeing sound and video items. In the event that application requests for media capacities in UI like camera get to, playing sound and so forth, at that point this casing work is utilized.

#### **2.2.4 Design Strategy**

There are numerous design system that can be used to build up an app like ModelViewController (MVC), ModelViewViewModel (MVVM), ModelViewPresentor (MVP) and so on. It relies upon the need of utilization as well. I utilized MVVM (ModelViewViewModel) in ABFL-EMICard.

MVVM is a structural example which isolates the portrayal and client communication. It has three noteworthy parts - Model, View Controller and View Model.

View is the substance of application. It is in charge of the layout and behaviour. Class of a view is commonly reusable in light of the fact that they don't contain any space particular rationale. For instance, a UI-TextField is a view that exhibits a text-field onto the screen, and its effortlessly reusable. Model and View don't connect with each other. Along these lines, thats why we use controller.

Controller is in charge of taking the end client demand and stacking the proper View and Model. It intervenes between model and view, commonly by means of the assignment design. All the client collaboration rationales are characterized here. In the perfect situation, the controller substance wont know the solid view its managing. Rather, it will speak with a reflection by means of a convention. A great illustration is the way a UI-CollectionView speaks with its information origin by means of the UI-CollectionView-DataSource convention.

## **2.3 Useful Concepts**

There are couple of essential ideas that I have to depict before I present my undertaking assignment. It isn't conceivable to build up an effective app without information of these.

### **2.3.1 Auto Layout**

AutoLayout is a constraint established design framework. In light of the imperatives connected on see, it dynamically makes position and size of all perspectives in see chain of command. View takes characteristic substance estimate in light of the substance under the view.

For instance, a label is focused on a level vertically and horizontally on the face of iPhone 7. At that point because of AutoLayout, on different screens additionally, that label will show up in the focal point of the screen.

### **2.3.2 CocoaPod**

A Cocoa Pod, or Pod in short, is used for a framework that is included to our venture by utilizing the CocoaPod's apparatus. It is fundamentally a reliance chief who settle conditions between frameworks, bring the subsequent code of them, at that point connect it together in a XCode-Workspace to fabricate our task. The frameworks that we need to use in our task, are composed in pod file (to make this we need to express "case init" summon in our terminal where our undertaking is found). Subsequent to composing all conditions, run "unit introduce" order in terminal. Furthermore, by this, every one of those frameworks will be consequently gotten and connected to the undertaking.

### **2.3.3 App Delegate Life Cycle**

Application designate adjusts to UIApplicationDelegate. Be that as it may, all strategies are discretionary. In this way, it isn't important to actualize all. In any case, There is a life-cycle of AppDelegate.

1. application:didFinishLaunchingWithOptions: Before showing substance of application to client, if application needs to initialise something, those work are done under this strategy

2. `application:willFinishLaunchingWithOptions:` At dispatch time, when application is executed first time, this strategy is summoned. This strategy returns genuine if application can be propelled.
3. `applicationWillResignActive:` When application is changing far from being the frontal area application, this technique is conjured. This strategy is utilized to place application into a dormant state.
4. `applicationDidBecomeActive:` When application is going to wind up closer view this technique is called This strategy is utilized for any very late arrangement.
5. `applicationWillTerminate:` When application is being ended, it is called. This is just approached end not suspension of application
6. `applicationDidEnterBackground:` Sometimes we don't know however application still keeps running in the back-ground. At the point when application is running out of sight, this strategy is called.
7. `applicationWillEnterForeground:` When application is going to move from foundation to frontal area, this is called. Application isn't at present in dynamic mode now.

#### **2.3.4 Delegate and Protocol**

It is a standout amongst the most essential idea in iOS improvement.

Protocols characterizes a set of strategies, rules, and different necessities that is required for a specific undertaking or a bit of usefulness. It is similar to JAVA's interface class. For instance, one is enlisting a man to create site. In this way, he creates a protocol which makes sure individual have knowledge of HTML. Each individual who gives meet for that same thing, must need to fulfil the convention. The protocols would then be able to be embraced by a classes, structs, or identification. Any write that fulfils the necessities of a convention is said to comply with that convention and must execute the majority of the strategies and rules of that protocols. protocols can have a discretionary technique or properties.

For instance, UI-TableView-Delegate, UI-TableView-DataSource are conventions of TableView in Swift.

In Swift, to impart between two class, assign idea is utilized. Delegation is basically only a methods for correspondence in objects of iPhone app. delegate is a basic method for associating objects and speak to others. As it were, appoint enables objects implement techniques in light of occasions that happen in another question.

For instance, TableView has,delegate and data-source. The ViewController, having TableView-outlet, acknowledges as delegate of TableView. All the important strategies that are characterized in convention are actualized in that ViewController since designate complies with the convention inside.

### **2.3.5 Life cycle of View Controller**

Each view controller has its own particular life cycle. View controller utilizes these following techniques to oversee sees.

1. loadView:- This strategy is not called by a programmer himself. It is consequently called when ViewController's properties is gotten to. On the off chance that view controller's default see is made physically, at that point this technique is superseded. In the event that we utilize interface developer to squeak sees, at that point it isn't abrogated..

2. viewDidLoad: This is called naturally when ViewController is stacked in the memory. This technique is abrogated in each ViewController so as to play out extra introduction on see that is stacked from nib documents or different errands. For instance, to instate factors, organize ask for, database get to and so on. On the off chance that the view is available in memory, at that point this strategy isn't called naturally. For instance, if ViewController B is pushed on a ViewController A that is as of now exhibited in the memory and when B is popped, at that point A's viewDidLoad isn't called around then since it was at that point introduce in the memory.A is expelled from the view chain of command when B is pushed over A.

3. **viewWillAppear:** This is called when ViewControllers view will be stacked to view-hierarchy. Override this function to use custom errands related with showing the view like refreshing screen information.

4. **viewWillDisappear:** This is called when the ViewControllers view is going to be expelled from the view-hierarchy. This technique is utilized to conceal the console, conferring altering changed, return the progressions made in ViewWillAppear and so forth.

5. **viewDidAppear:** This is called when the ViewControllers view is stacked to the view-hierarchy. This strategy is abrogated to play out extra undertakings related with showing the view. For instance, begin UI activity.

6. **viewDidDisappear:** This is called when the ViewControllers view is expelled from the view-hierarchy. This strategy is utilized to evacuate store information, stop administrations identified with see like sound.

## 2.4 UI Elements

UI-Elements are those UI components which we find in our app. In XCode, there is office of intuitive of a component to interface-builder. There is no compelling reason to compose any code for it. Every UI component can be altered in light of necessity. I utilized after segments to build up the app:

**Button** is utilized to deal with some activity. For instance, to present a frame, catch is utilized.

**Label** is utilized to uproot static content to client. For instance, to indicate portrayal about a specific text, we can utilize labels.

**TableView** is utilized to dislodge a scrollinf information in numerous columns and segments. For TableView, controller needs to acclimate two conventions - UI-TableView-Delegate and UI-TableView-DataSource.



**TextField** is utilized to take enter from client. In view of various info, similar to content information or number info, we can tweak the console compose in TextField.

**TextView** is additionally used to take textual input from client. For E.g. name, remark

**PickerView** is flexible. It uses its own data-source and delegation to fill information. E.g. DatePickerView

**NavigationBar** contains the navigation button of a UINavigationController. UINavigationController is a heap of ViewControllers which are pushed and popped like a stack.

**CollectionView** resembles TableView and is more bland than TableView. You can alter the view of information by different layout. Basic format is FlowLayout.

**ScrollView** is utilized when information size develops more than screen estimate. In this way, we have to move down to view the information.

**ImageView** is utilized to dislodge picture

## CHAPTER 3

### SYSTEM DEVELOPMENT

#### 3.1 Login Screen

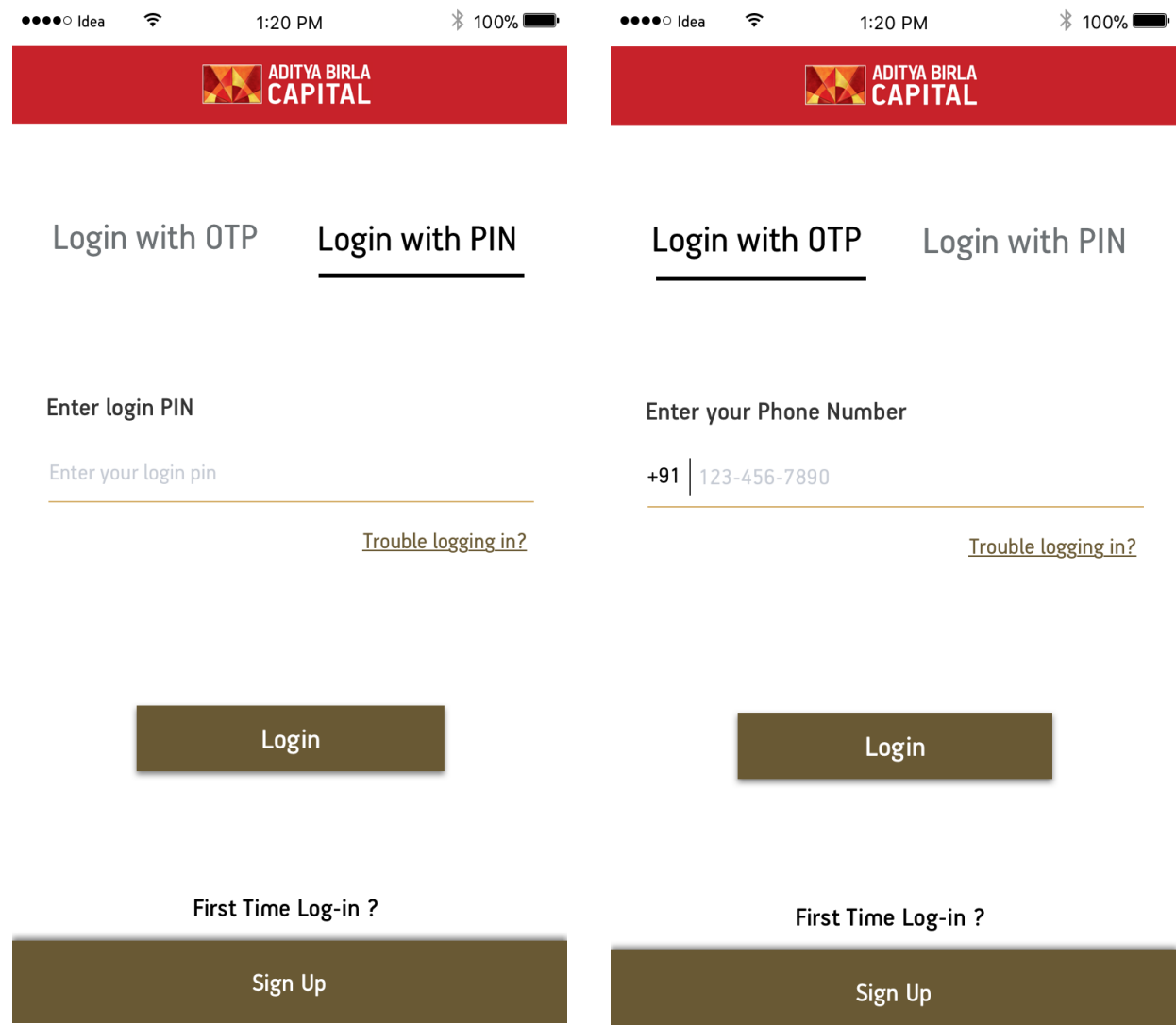
UI Element: Textfield, Label, TableView, Button

Delegation utilized: UITextFieldDelegate

Pods utilized: SkyFloatingLabelTextField - It is used for floating placeholder text and to update error on validation.

Validation Conditions: Mobile Number should be of 10 digits and PIN of 4 digits.

Definition: User can use either login with pin or login with mobile. Login with mobile moves to otp screen on validation



**Figure 3.1:** Login Screen

### 3.2 OTP Screen

UI Element: Textfield, Label, TableView, Button

Delegation utilized: UITextFieldDelegate

Pods utilized: SkyFloatingLabelTextField - It is used for floating placeholder text and to update error on validation.

Validation Conditions: OTP should be of six digits

Definition: User can enter the OTP which is validated by API call. Moreover entered OTP is secured with sha256 encryption.

The image shows a mobile app interface for verifying an OTP. At the top, there's a status bar with 'Idea' as the carrier, signal strength, Wi-Fi, time '1:20 PM', Bluetooth, and 100% battery. Below the status bar is a back arrow. The main title is 'Verify OTP'. Underneath is a floating label 'Enter OTP PIN' with a text input field. To the right of the input field is a link 'Resend OTP Pin'. Below this is a numeric keypad titled 'Confirm OTP'. The keypad has a 3x3 grid of buttons for digits 1-9, a row for '+ \* #', '0', and a backspace button (arrow with 'x'). Each digit button also has its corresponding letters (e.g., 1 has no letters, 2 has ABC, 3 has DEF, etc.).

Confirm OTP		
1	2 ABC	3 DEF
4 GHI	5 JKL	6 MNO
7 PQRS	8 TUV	9 WXYZ
+ * #	0	⬅ x

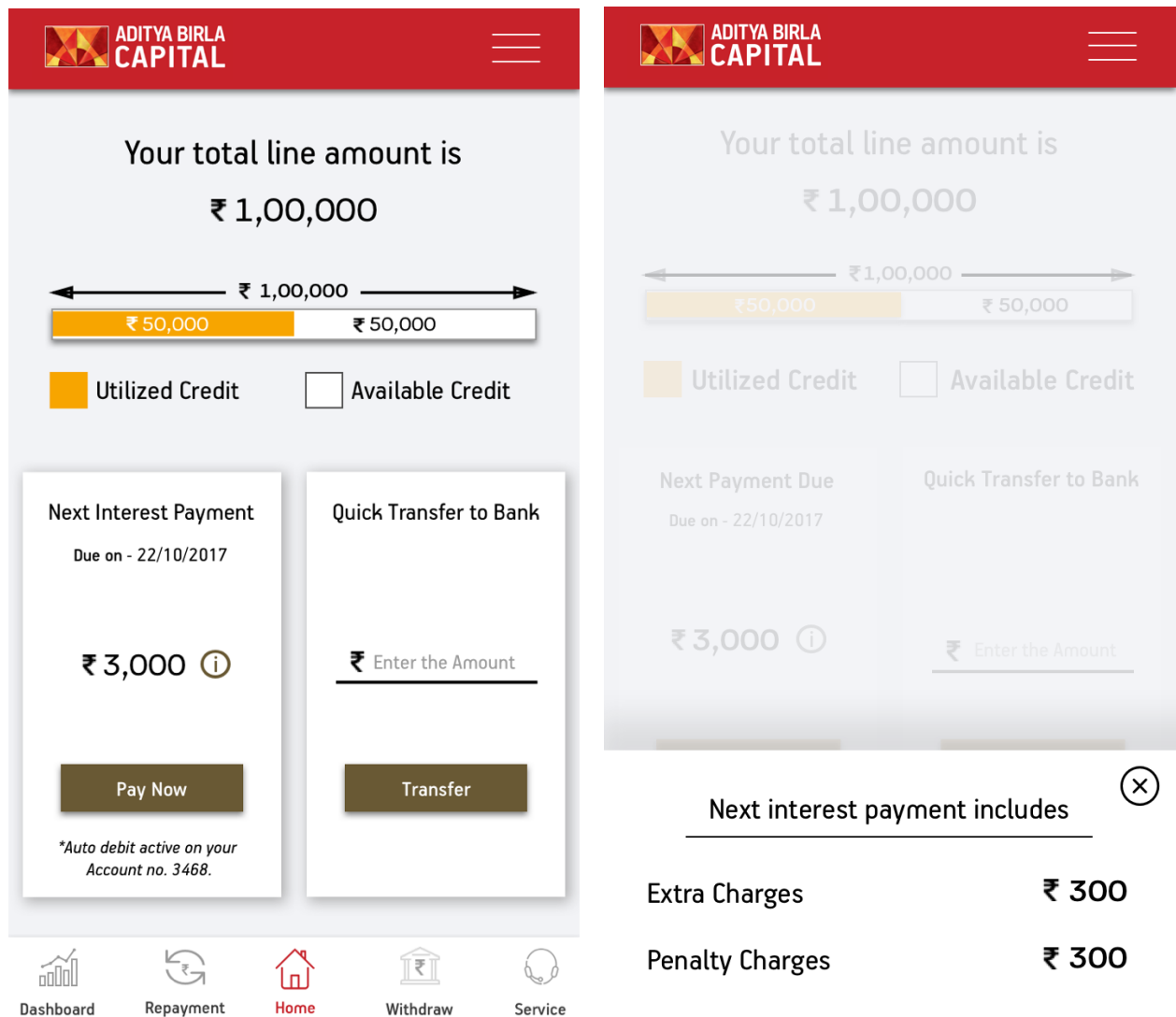
**Figure 3.2:** OTP Screen

### 3.3 Home Screen

UI Element: Textfield, Label, Custom View, Scroll View, Button

Pods utilized: SkyFloatingLabelTextField - It is used for floating placeholder text and to update error on validation.

Definition: User can click on “i” button and get the info in scroll view and can also transfer money to hi other bank accounts

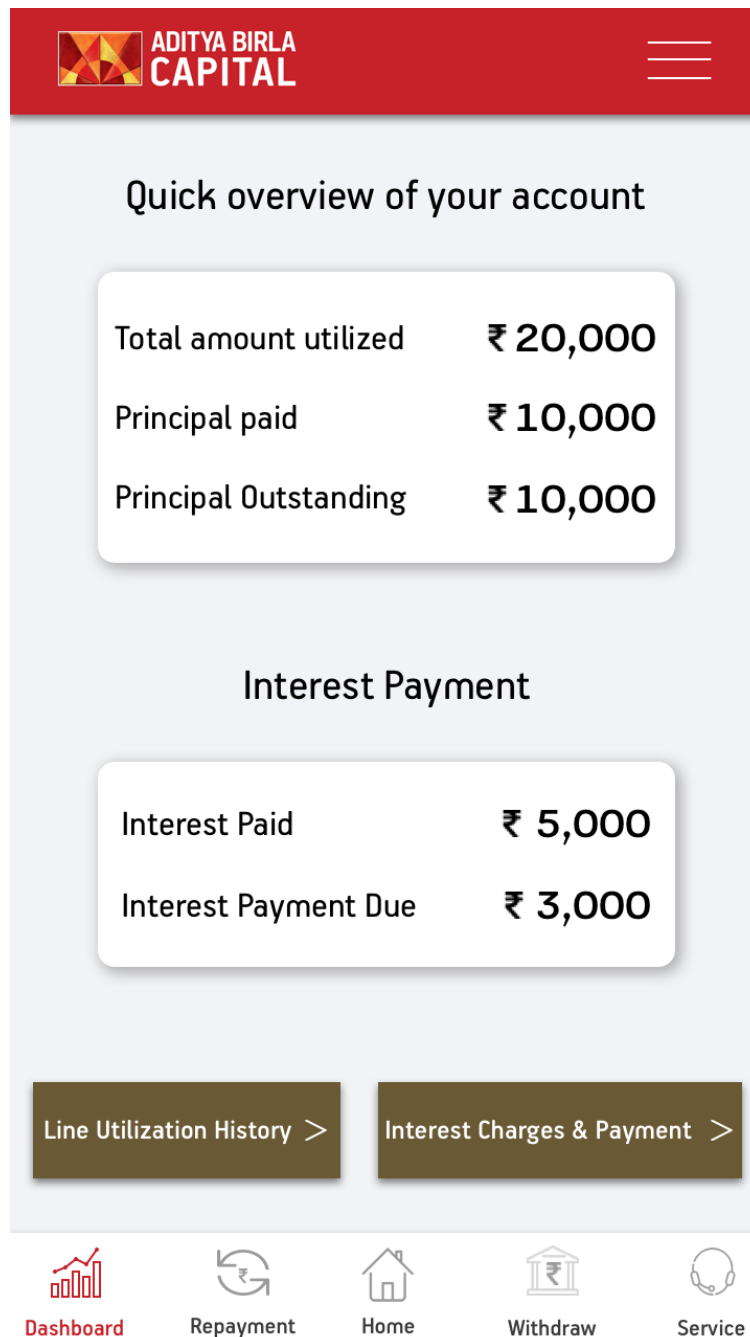


**Figure 3.3:** Home Screen

### 3.4 Dashboard Screen

UI Element: Label, Stack View, Button

Definition: User can see both of his history line utilization and interest charges and payment history.

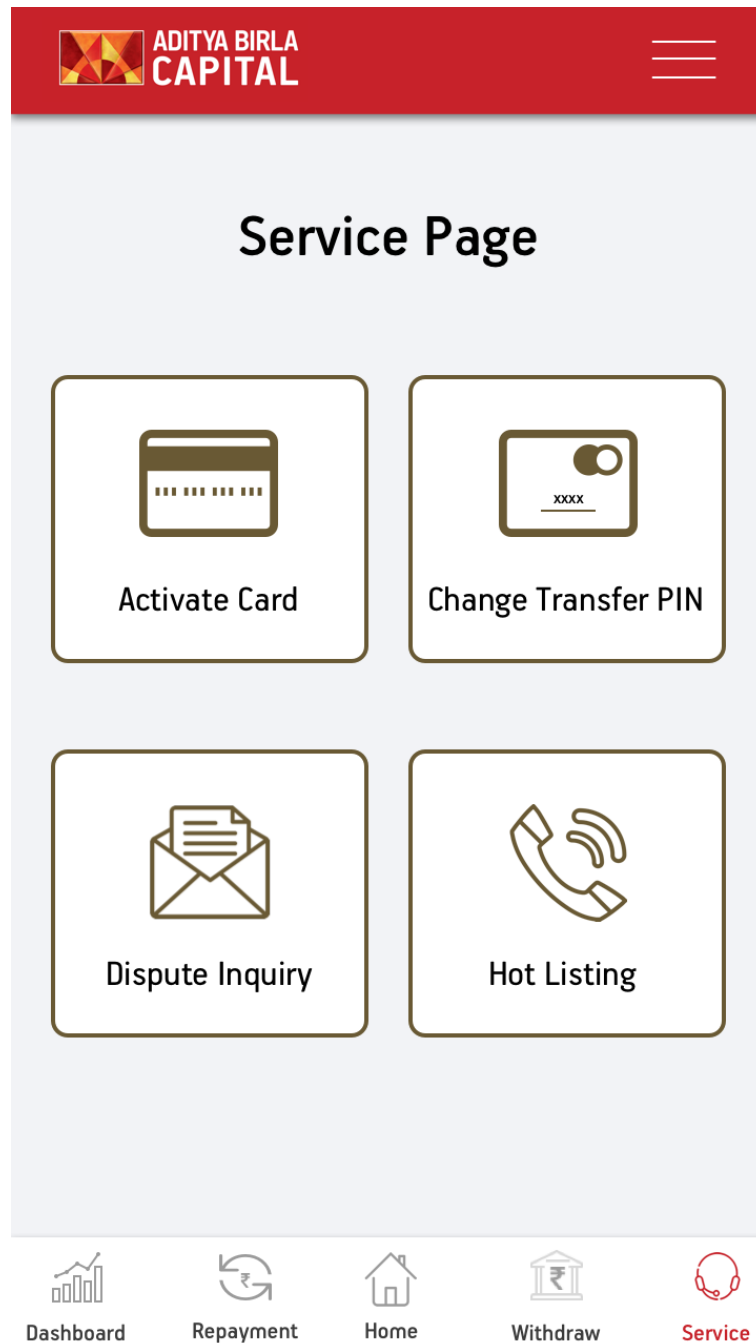


**Figure 3.4:** Dashboard Screen

### 3.5 Service Page Screen

UI Element: Label, Custom View, StackView

Definition: User can use any dispute inquiry if they have



**Figure 3.5:** Service Page Screen

### 3.6 Transaction Screen

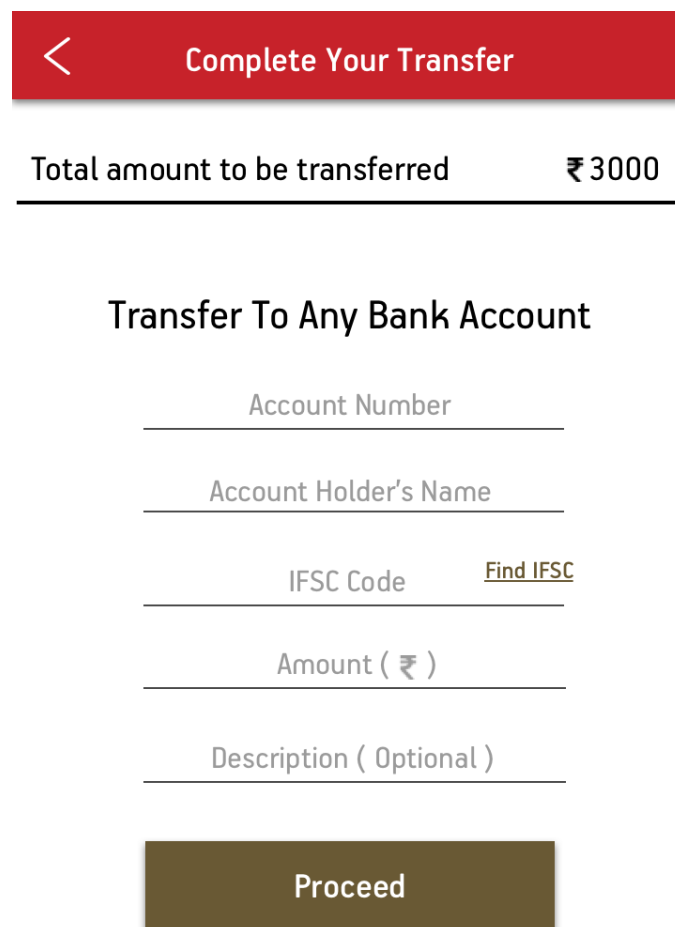
UI Element: Textfield, Label, TableView, Button

Delegation utilized: UITextFieldDelegate

Pods utilized: SkyFloatingLabelTextField - It is used for floating placeholder text and to update error on validation.

Validation Conditions: No of digits in account number should be between 9 to 18.

Definition: User has to enter all the information to proceed.



The image shows a mobile app screen for completing a bank transfer. At the top is a red header bar with a white back arrow and the text 'Complete Your Transfer'. Below this is a summary line: 'Total amount to be transferred ₹ 3000'. The main section is titled 'Transfer To Any Bank Account' and contains five input fields with floating labels: 'Account Number', 'Account Holder's Name', 'IFSC Code' (with a 'Find IFSC' link), 'Amount ( ₹ )', and 'Description ( Optional )'. At the bottom is a large brown button labeled 'Proceed'.

*\*As you add a new bank account we will be first verifying it, only then the transaction of funds will take place.*

**Figure 3.6:** Transaction Screen

### 3.7 Transaction PIN Screen

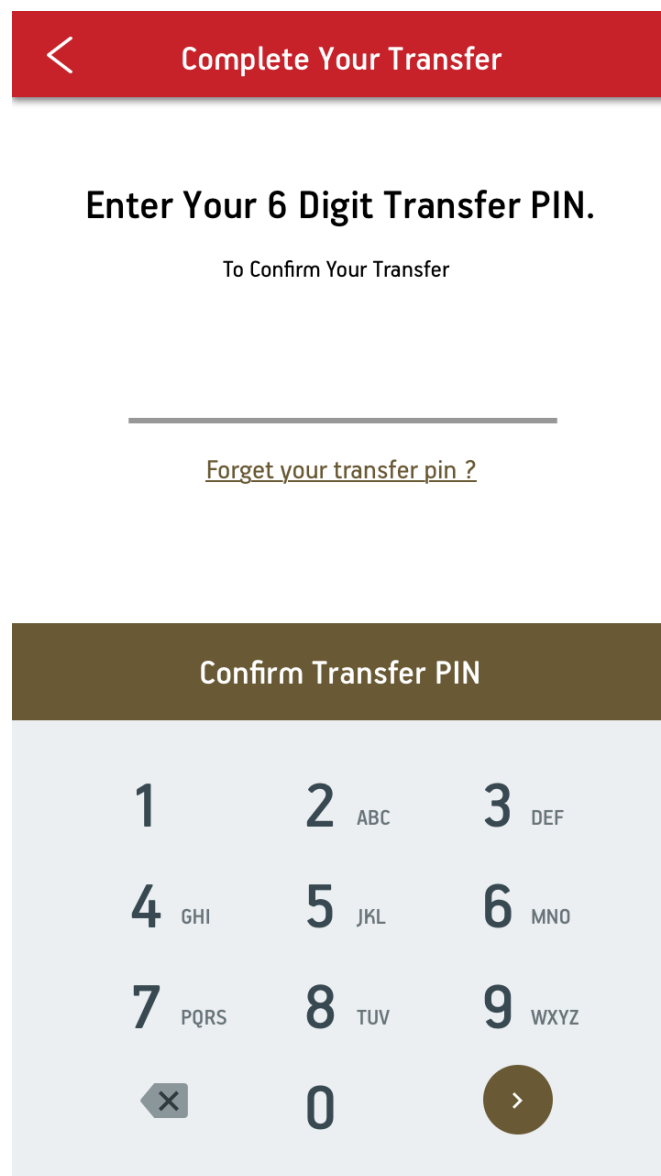
UI Element: Textfield, Label, Button

Delegation utilized: UITextFieldDelegate

Pods utilized: SkyFloatingLabelTextField - It is used for floating placeholder text and to pop up error on validation.

Validation Conditions: No of digits in transaction number should be between exactly 6

Definition: User has to enter all the information to proceed.



The image shows a mobile app screen for completing a transaction. At the top is a red header bar with a white back arrow and the text "Complete Your Transfer". Below this is the main heading "Enter Your 6 Digit Transfer PIN." followed by the subtitle "To Confirm Your Transfer". A horizontal line separates the heading from a link that says "Forget your transfer pin ?". The bottom half of the screen features a dark brown header bar with the text "Confirm Transfer PIN". Below this is a light gray area containing a numeric keypad. The keypad has digits 1 through 9, a backspace button (a gray diamond with an 'x'), a zero button, and a next button (a dark brown circle with a white right arrow). Each digit is accompanied by its corresponding letters in a smaller font.

**Figure 3.7:** Transaction PIN Screen



### 3.8 Forget Transaction PIN Screen

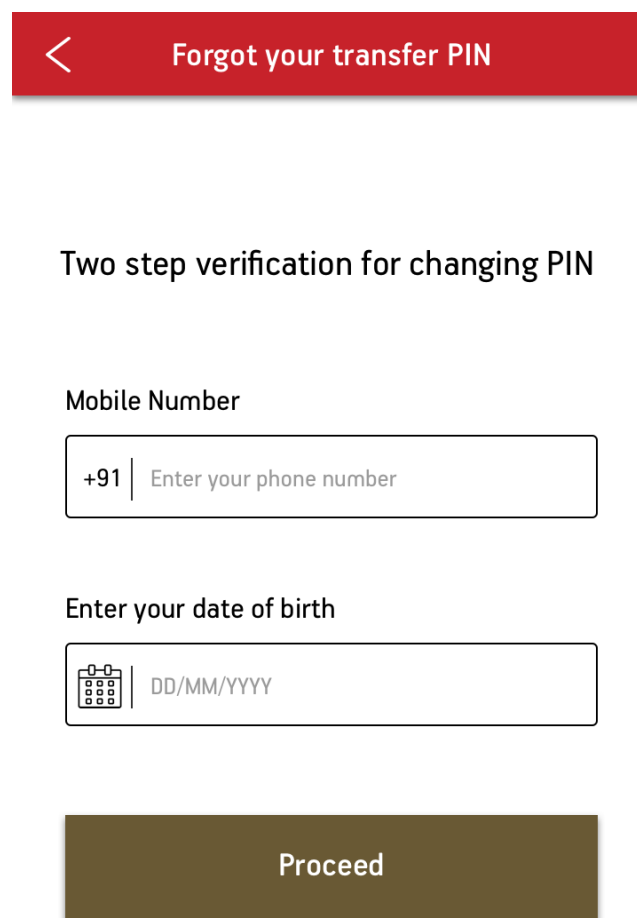
UI Element: Textfield, Label, Button, ImageView

Delegation utilized: UITextFieldDelegate

Pods utilized: SkyFloatingLabelTextField - It is used for floating placeholder text and to pop up error if validation fails.

Validation Conditions: Mobile Number should be of 10 digits.

Definition: Both the information is verified via API Call before proceeding



< Forgot your transfer PIN

Two step verification for changing PIN

Mobile Number

+91 | Enter your phone number

Enter your date of birth

DD/MM/YYYY

Proceed

**Figure 3.8:** Forget Transfer PIN Screen

### 3.9 Forget Transaction PIN Screen - II

UI Element: Textfield, Label, Button

Delegation utilized: UITextFieldDelegate

Pods utilized: SkyFloatingLabelTextField - It is used for floating placeholder text and to pop up error if validation fails.

Validation Conditions: OTP should be of 6 digits.

Definition: OTP is encrypted using sha256 and then verified via API Call.

< Forgot your transfer PIN

Two step verification for changing PIN

Enter OTP

Enter OTP

[Resend OTP Pin](#)

Reset Transfer PIN

**Figure 3.9:** Forget Transfer PIN Screen-II

### 3.10 Update Transaction PIN Screen

UI Element: Textfield, Label, Button

Delegation utilized: UITextFieldDelegate

Pods utilized: SkyFloatingLabelTextField - It is used for floating placeholder text and to pop up error if validation fails.

Validation Conditions: PIN should be of 6 digits

Definition: PIN is updated via PUT API Call

< Forgot your transfer PIN

## Change Your Transfer Pin

Enter your new PIN

XXXX

Re-type PIN again

XXXX

Confirm PIN

**Figure 3.10:** Update Transaction PIN Screen

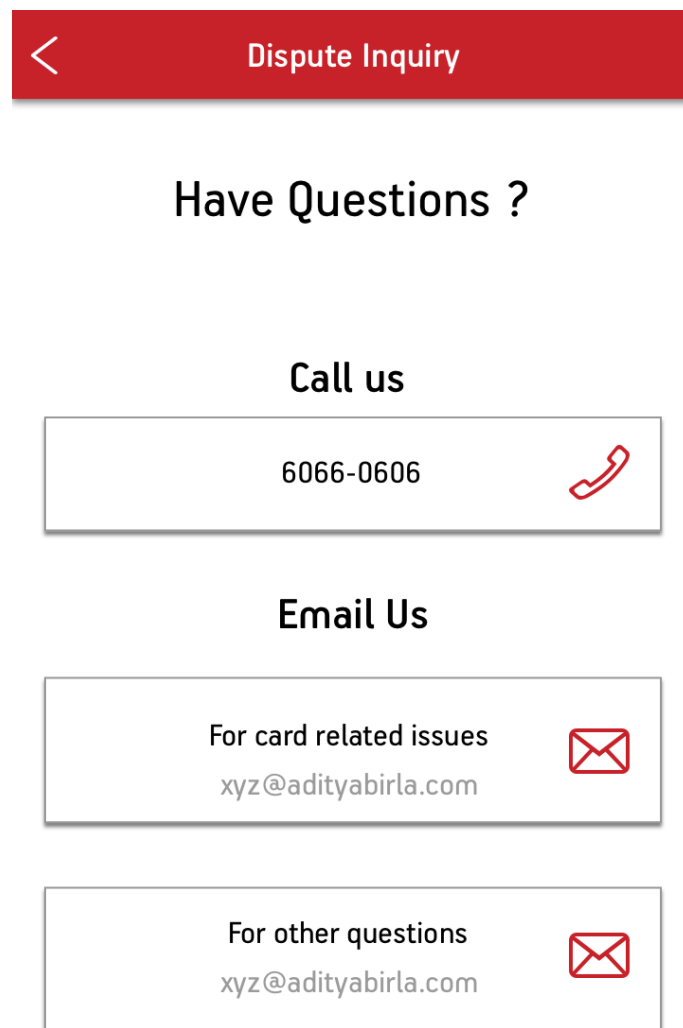
### 3.11 Dispute Inquiry Screen

UI Element: UIImageView, Label, Button

Delegation utilized: MFMailComposeViewControllerDelegate

Validation Conditions: Mail App should be downloaded

Definition: Call the helpline number and mail through app



**Figure 3.11:** Dispute Inquiry Screen

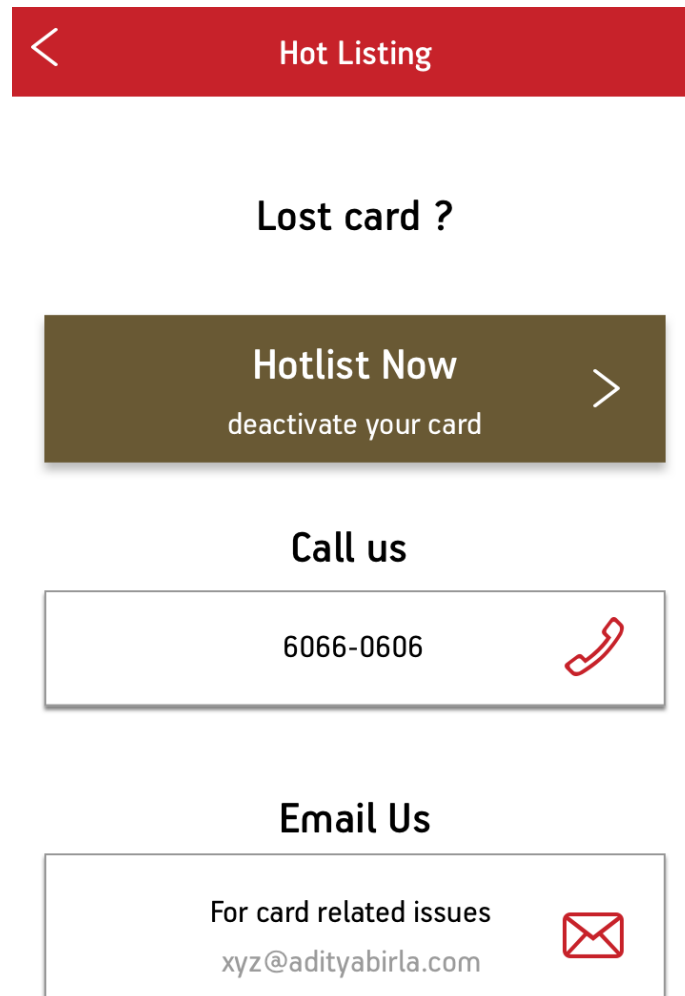
### 3.12 Hotlisting Screen

UI Element: ImageView, Label, Button

Delegation utilized: MFMailComposeViewControllerDelegate

Validation Conditions: Mail App should be downloaded

Definition: Call the helpline number and mail through app and also activate card



**Figure 3.12:** Hotlisting Screen

### 3.13 Activate/Deactivate Screen


UI Element: ImageView, Label, TextField Button

Delegation utilized: UITextFieldDelegate.

Pods utilized: RxSwift – It is used to update the text on card image via text field


Validation Conditions: Digits entered should be of 4 length


Definition: User enters the digits which calls the API to activate or deactivate card of the user.


 Hot Listing

## Deactivate Your Card

Enter Card Details To Deactivate Your Card

**Kotak**  
Kotak Mahindra Bank

**ADITYA BIRLA  
CAPITAL**



First 4 Digit


xxxx-xxxx

Last 4 Digit

VALID FROM

VALID FROM


Card Holder Name

  
mastercard

Enter your first 4 digit of card


Enter your last 4 digit of card


Deactivate Card


 Activate Card

## Activate Your Card

Enter Card Details To Activate Your Card

**Kotak**  
Kotak Mahindra Bank

**ADITYA BIRLA  
CAPITAL**



First 4 Digit


xxxx-xxxx

Last 4 Digit

VALID FROM

VALID FROM

Card Holder Name

  
mastercard

Enter your first 4 digit of card

Enter your last 4 digit of card

Activate Card

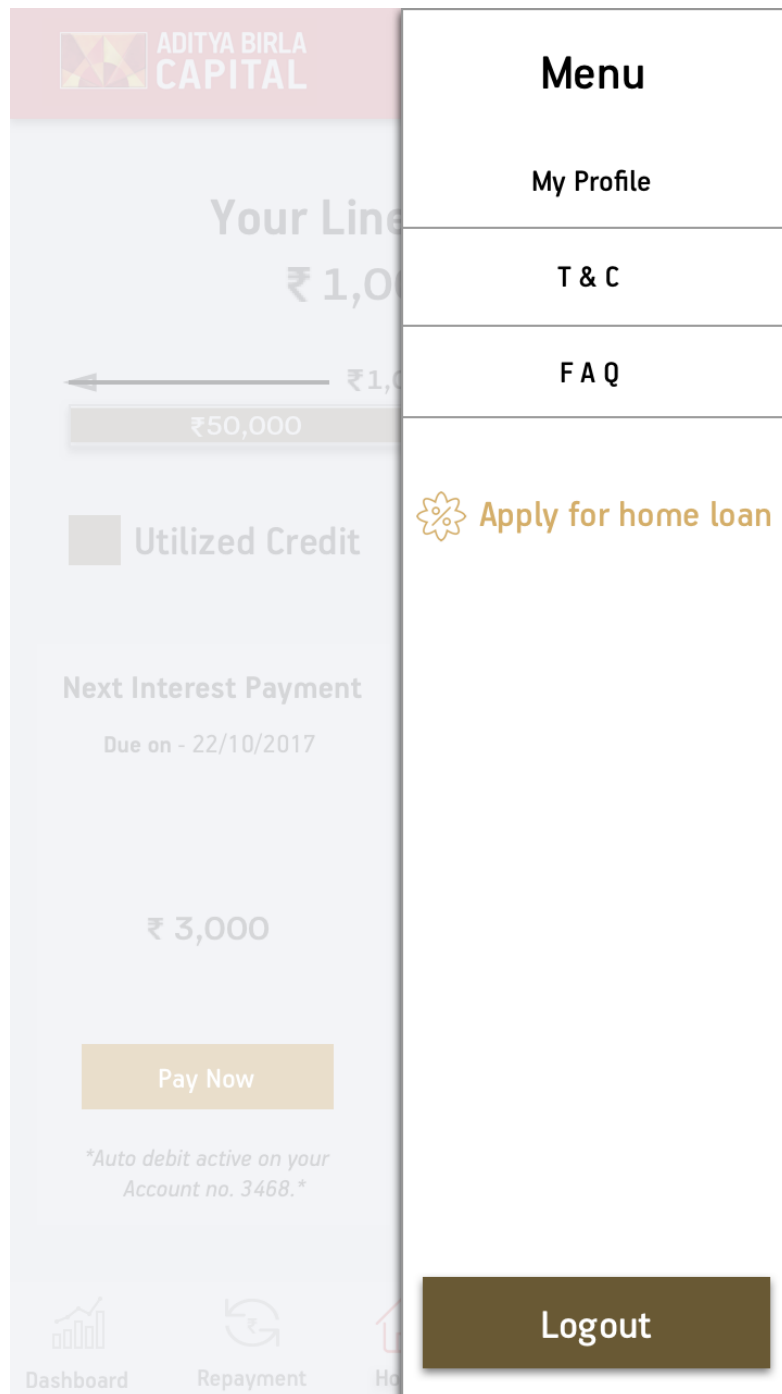
**Figure 3.13:** Activate/Deactivate Card Screen

23

### 3.14 Side Menu Screen

UI Element: ImageView, Label, TextField Button, TableView

Definition: User can use the side menu by sliding their finger. There are multiple options to the user.



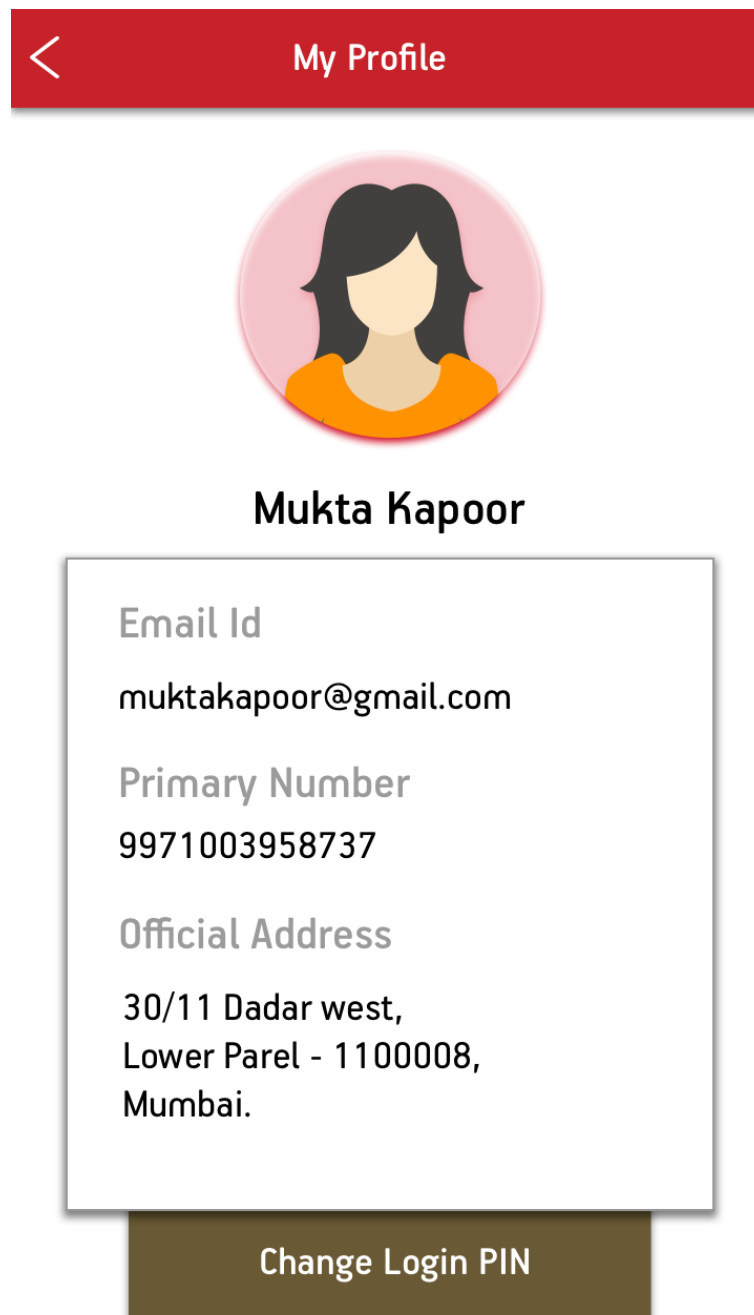
**Figure 3.14:** Side Menu Screen

### 3.15 My Profile Screen

UI Element: ImageView, Label , Button

Pods utilized: Alamofire – It is used to call API to get the profile details.

Definition: User can see his profile details and can also change login PIN.



**Figure 3.15:** My Profile Screen



### 3.16 Change Login PIN Screen

UI Element: TextField, Label , Button

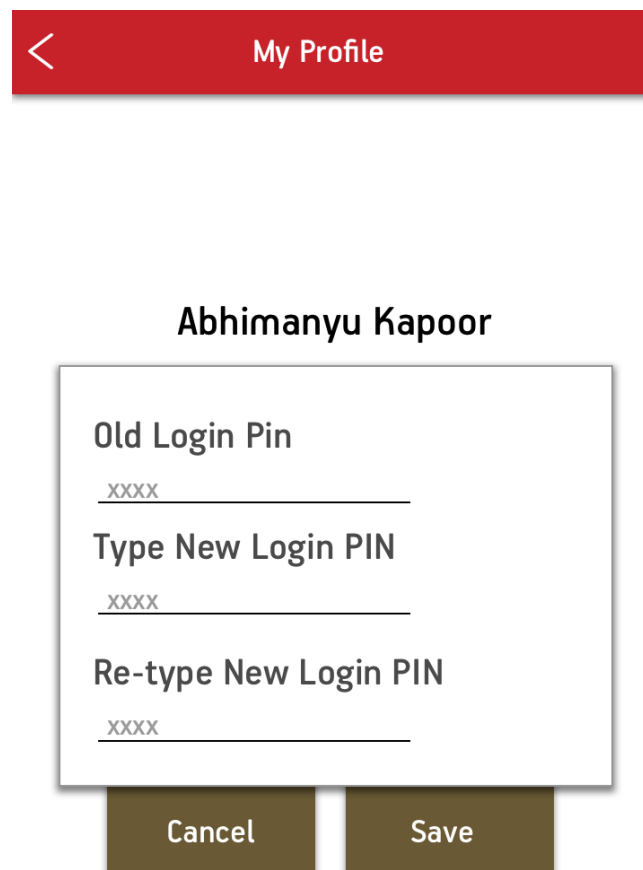
Pods utilized: Alamofire – It is used to call API to change the login PIN.

SkyFloatingLabelTextField - It is used for floating placeholder text and to pop up error if validation fails.

Delegation utilized: UITextFieldDelegate

Validation Conditions: Digits entered should be of 4 length

Definition: User enters the PIN which calls the API to update the pin of user.



The screenshot shows a mobile application interface. At the top, there is a red header bar with a white back arrow on the left and the text "My Profile" in the center. Below the header, the name "Abhimanyu Kapoor" is displayed. In the center of the screen, a white dialog box is shown with a thin grey border. The dialog box contains three text input fields, each with a floating placeholder text "XXXX". The labels for the fields are "Old Login Pin", "Type New Login PIN", and "Re-type New Login PIN". At the bottom of the dialog box, there are two buttons: "Cancel" and "Save".

**Figure 3.16:** Change Login PIN Screen

### 3.17 Apply Home Loan Screen

UI Element: TextField, Label , Button, ImageView, ScrollView, pickerView

Pods utilized: SkyFloatingLabelTextField - It is used for floating placeholder text and to pop up error if validation fails.

Delegation utilized: UITextFieldDelegate

Validation Conditions: Email id and mobile number should be valid.

Definition: User fills all the data to apply for home loan.

< Apply for home loan

Enter your details

Name

Enter your full name

Mobile Number

+91 Enter your phone number

Email ID

Enter your email id

Current city living

Select the city

Property city location

Select the city

Loan amount

₹ Enter Loan amount

Property identified

☒ Yes

☐ No

Select one

☒ Salaried

☐ Self Employed

Apply For Home Loan

**Figure 3.17:** Apply Home Loan Screen

### 3.18 Line Utilization/Interest Payment Screen

UI Element: ImageView, Label , Button

Pods utilized: Alamofire – It is used to call API to get the user data.

SwiftChart – It is used to construct chart from user data.

Definition: User can see his activities.

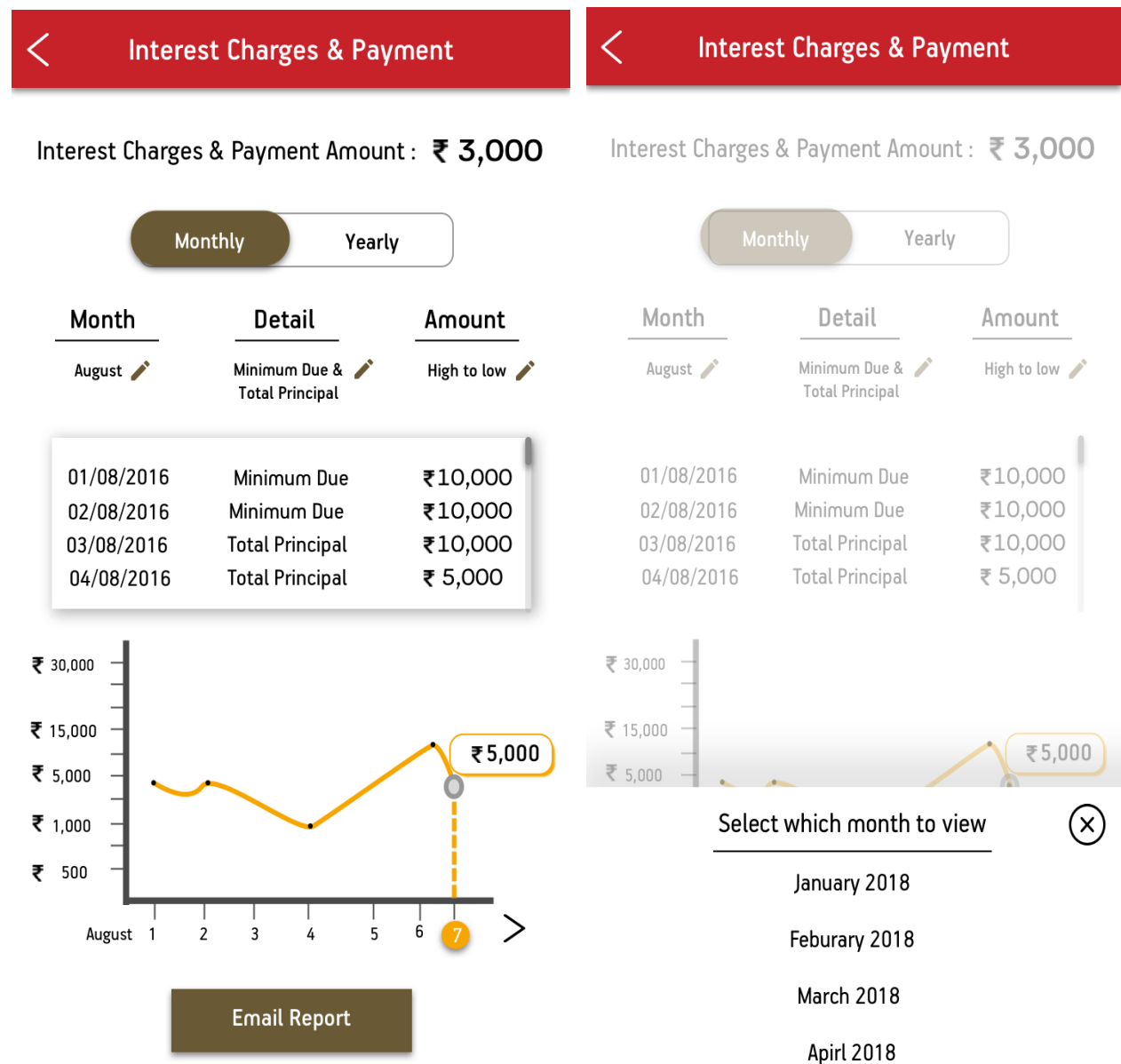


Figure 3.18: Line Utilization/Interest Payment Screen

### 3.19 Email Transaction Report Screen

UI Element: TextField, Label , Button, ImageView, DatePickerView

Pods utilized: SkyFloatingLabelTextField - It is used for floating placeholder text and to pop up error if validation fails.

Alamofire – It is used to call API to mail the user data.

Delegation utilized: UIPickerViewDelegate

Validation Conditions: From date should be less than To date and in possible range.

Definition: User should select atleast one option and then get the mail.

< Email Transaction Report

Choose report of

Card Bank Repayment

Card Bank Repayment

Select the date from DD/MM/YYYY to DD/MM/YYYY

Proceed

**Figure 3.19:** Email Transaction Report Screen

### 3.20 Forget Login PIN Screen

UI Element: TextField, Label , Button, ImageView, DatePickerView

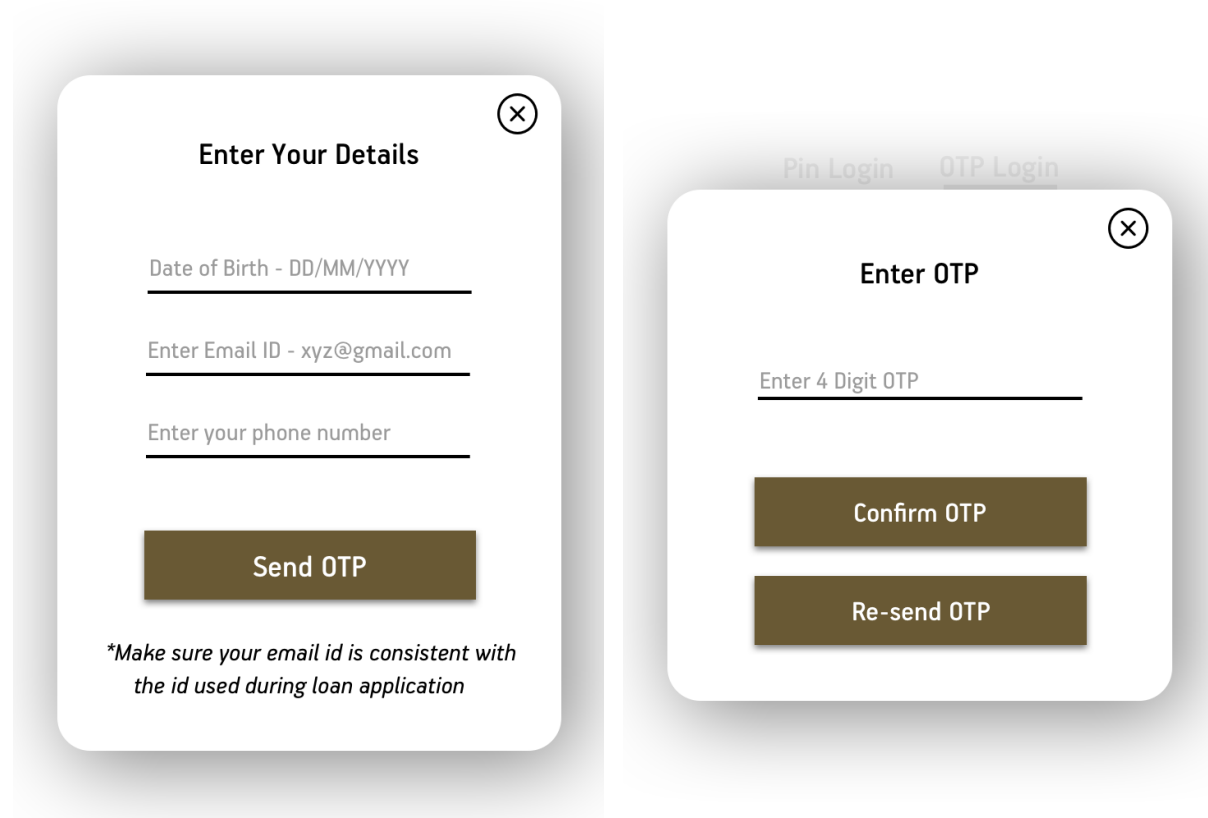
Pods utilized: SkyFloatingLabelTextField - It is used for floating placeholder text and to pop up error if validation fails.

Alamofire – It is used to call API to reset the login PIN.

Delegation utilized: UIPickerViewDelegate

Validation Conditions: Phone number should be of 10 digits and email id should be valid.

Definition: User can reset the PIN using OTP.



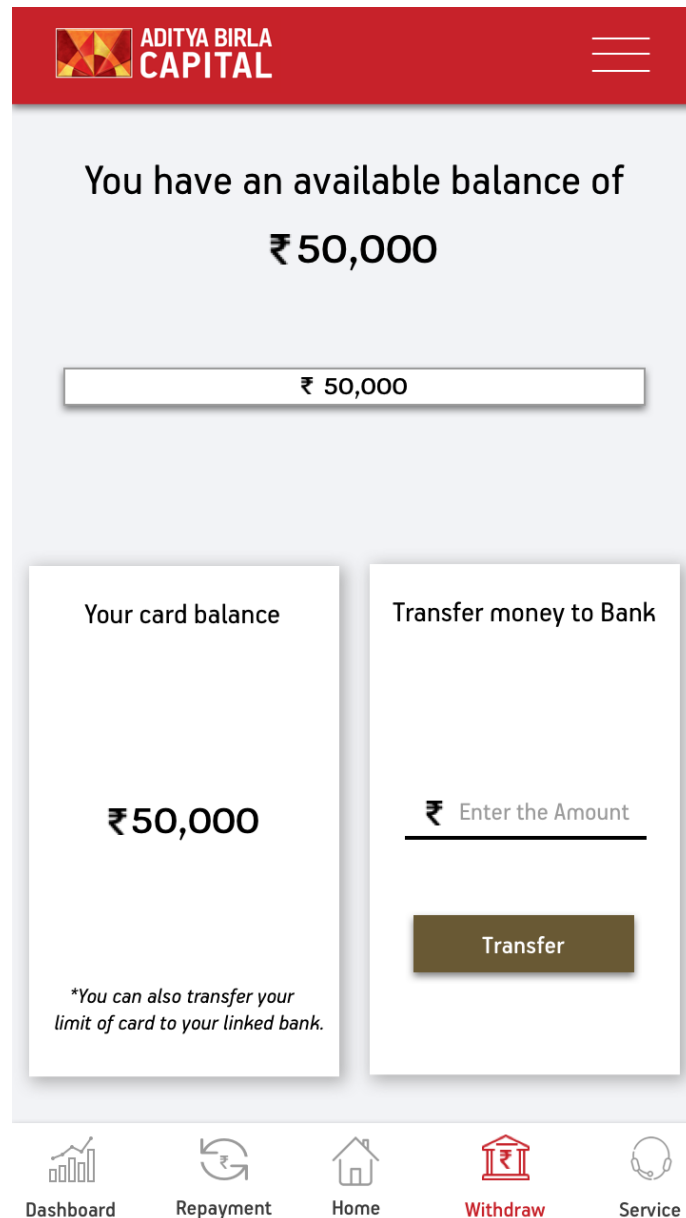
**Figure 3.20:** Forget Login PIN Screen

### 3.21 Withdraw Screen

UI Element: TextField, Label , Button, Custom View

Pods utilized: SkyFloatingLabelTextField - It is used for floating placeholder text and to pop up error if validation fails.

Definition: User can see his withdrawal details and transfer money to his different account.



**Figure 3.21:** Withdraw Screen

### 3.22 Repayment Screen

UI Element: TextField, Label , Button, Custom View

Pods utilized: SkyFloatingLabelTextField - It is used for floating placeholder text and to pop up error if validation fails.

Definition: User can see his repayment details and complete his interest payment.

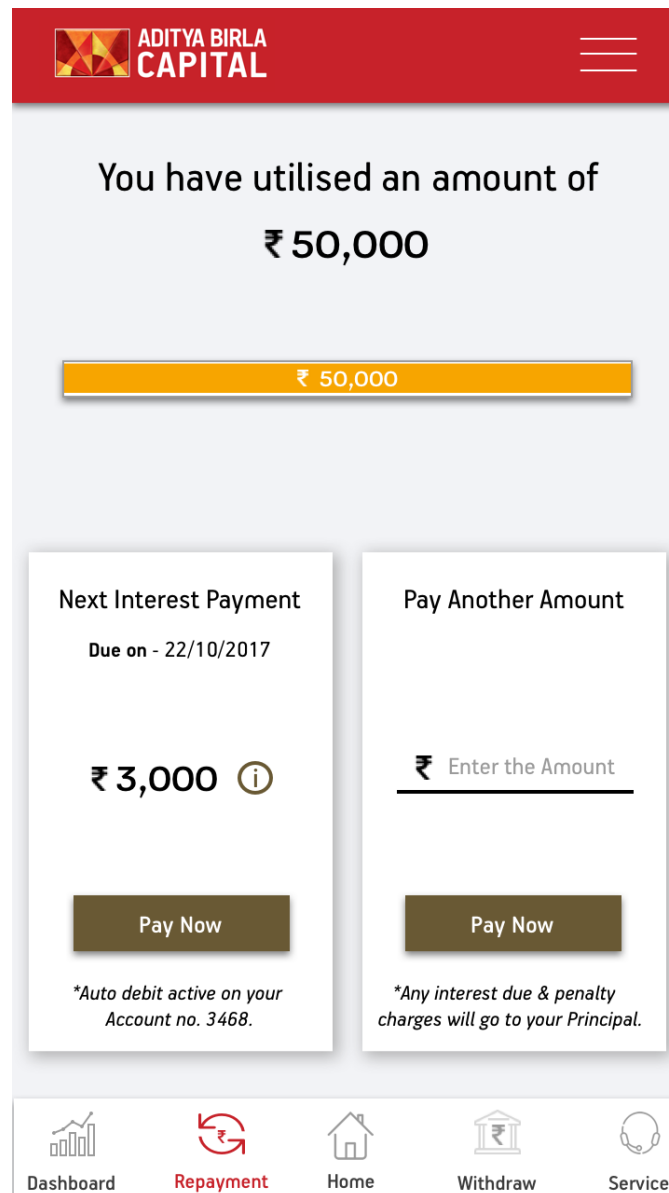


Figure 3.22: Repayment Screen

### 3.23 Find IFSC Code Screen

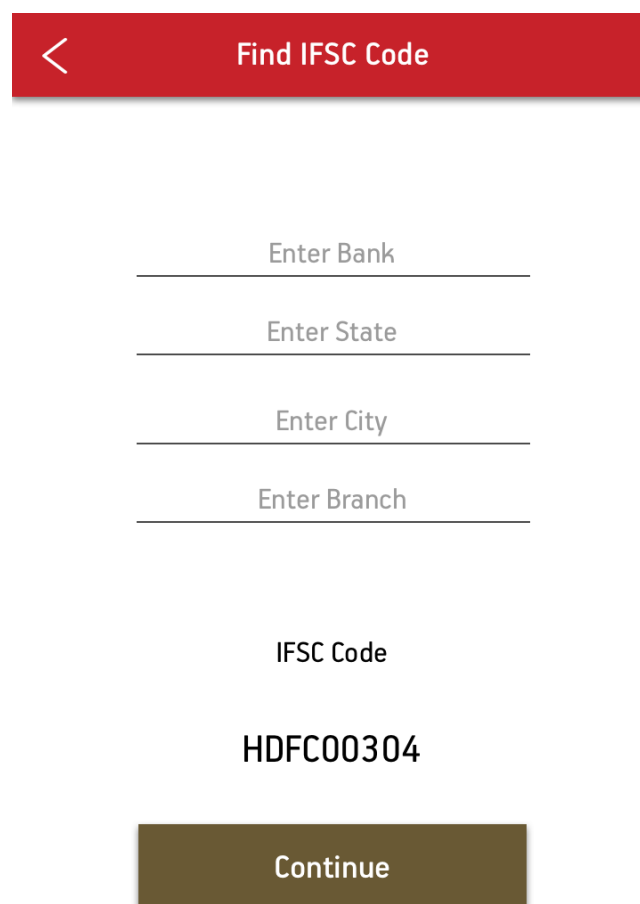
UI Element: TextField, Label , Button, UIPickerView

Pods utilized: SkyFloatingLabelTextField - It is used for floating placeholder text and to pop up error if validation fails.

Alamofire – It is used to call API to get the IFSC codes as the user enters the data.

Delegation utilized: UIPickerViewDelegate

Definition: User can get the IFSC code using bank and branch name.



The image shows a mobile application screen titled "Find IFSC Code". At the top, there is a red header bar with a white back arrow on the left and the title "Find IFSC Code" in white text. Below the header, there are four text input fields with floating placeholder labels: "Enter Bank", "Enter State", "Enter City", and "Enter Branch". Below these fields, the text "IFSC Code" is displayed, followed by the value "HDFC00304". At the bottom of the screen, there is a brown button with the text "Continue" in white.

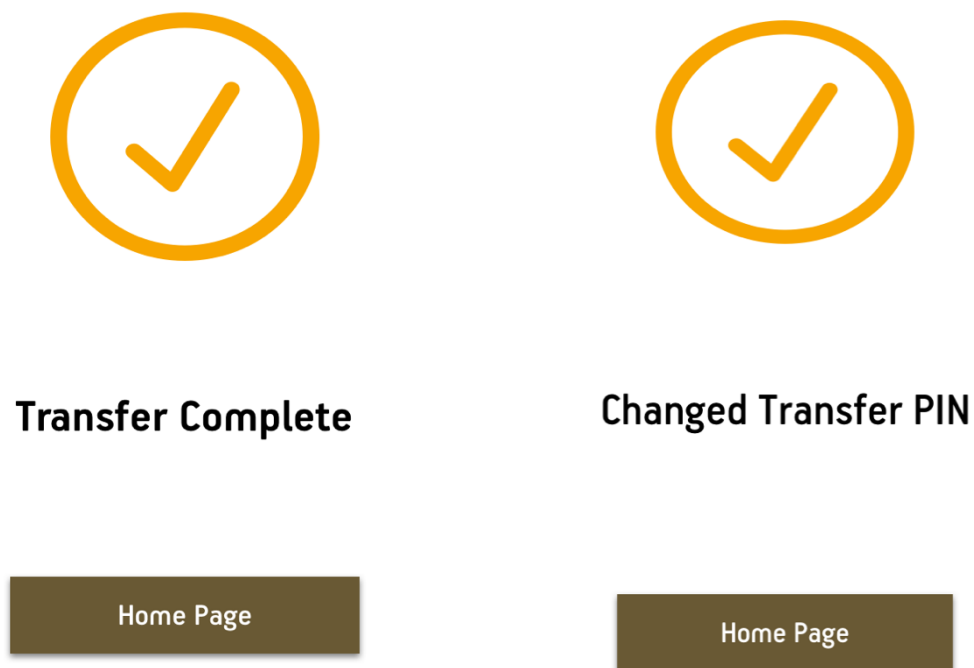
**Figure 3.23:** Find IFSC Code Screen



### 3.24 Completion Screen

UI Element: ImageView, Label , Button

Definition: This screen is shown at multiple places by changing the text of the label on completion.



**Figure 3.24:** Completion Screen



**Successful Transaction**

[Home Page](#)



**Your Pin Changed**

[Home Page](#)



Your Card Has Been Deactivated

Home Page



**Congratulations !**

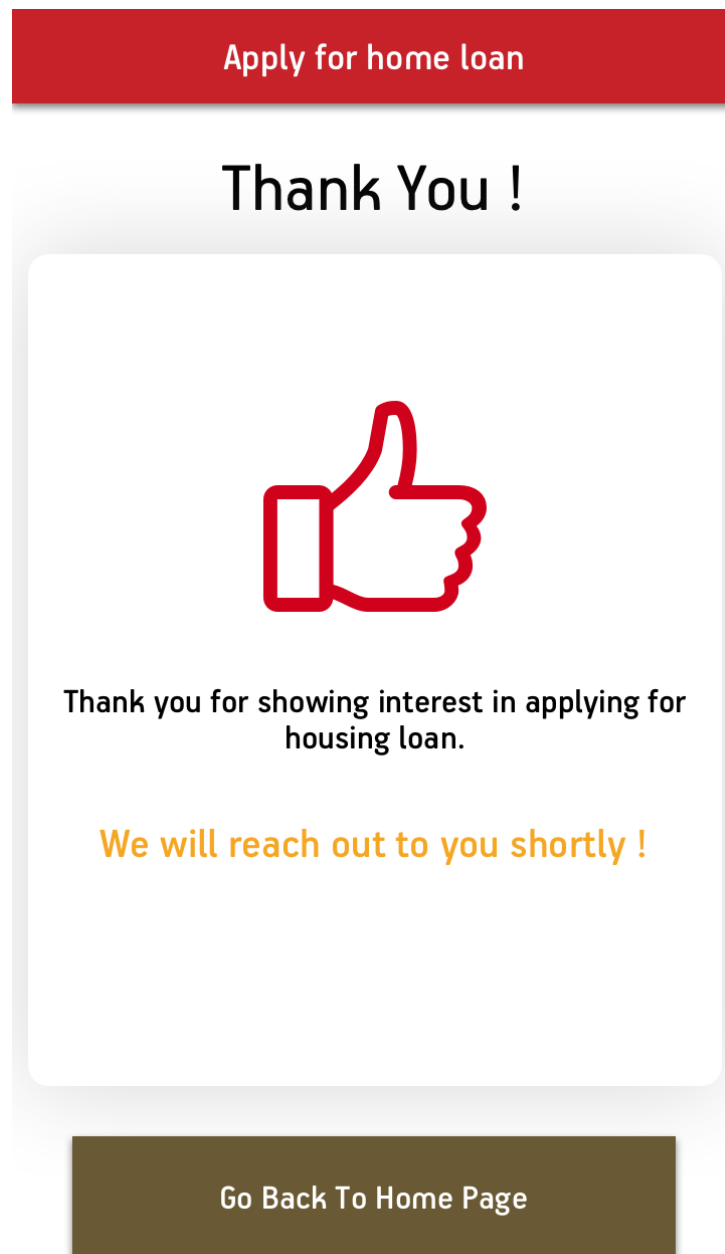
Your Card Is Active

Home Page

### 3.25 Apply Home Loan Completion Screen

UI Element: Label, Button, ImageView

Definition: This screen is shown when the user completes his apply home loan application.



**Figure 3.25:** Apply Home Loan Completion Screen

### 3.26 Network Error Screen

UI Element: Label, Button, ImageView

Definition: This screen is shown when the API call is rejected due to lack of internet or any other reason. Error code is shown in label inside image.



You are seeing this page because this  
URL does not exist.

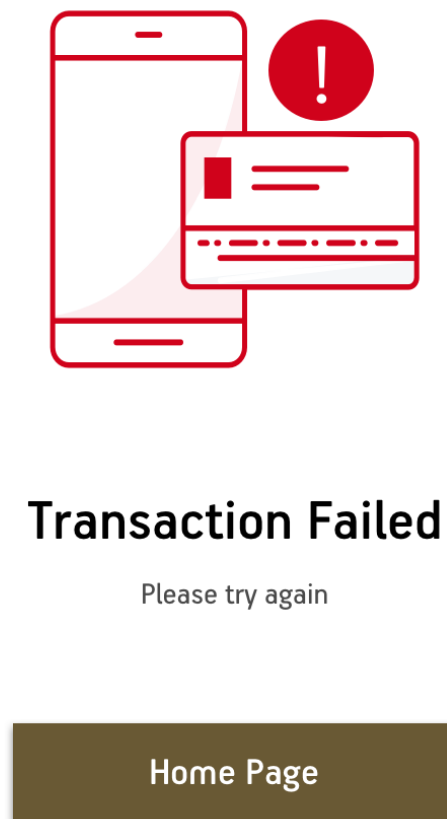
Home Page

**Figure 3.26:** Network Error Screen

### 3.27 Transaction Failure Screen

UI Element: Label, Button, ImageView

Definition: This screen is shown when the transaction gets failed and the reason is shown in the label below image.



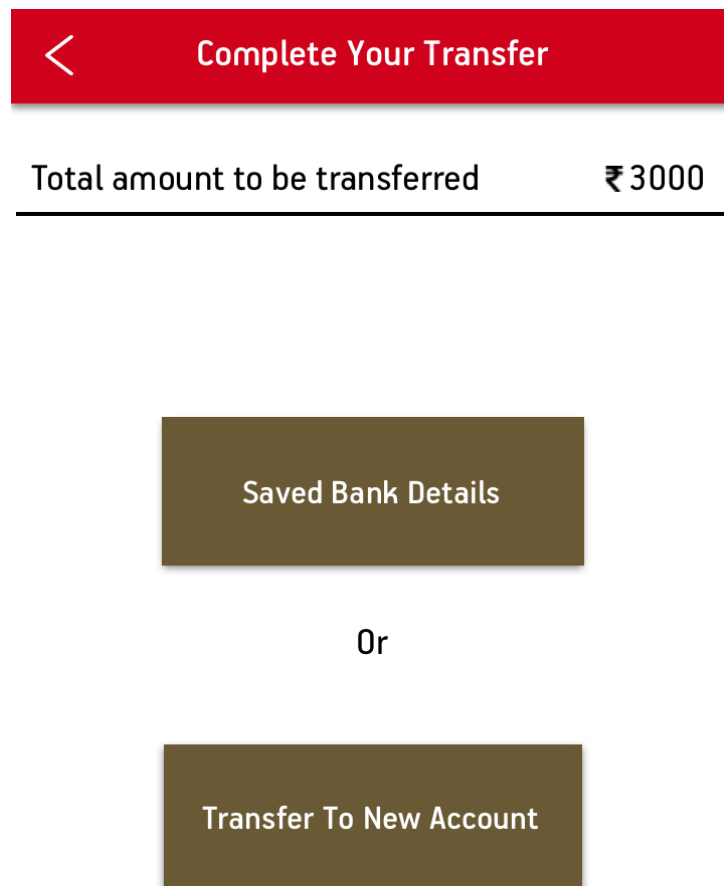
**Figure 3.27:** Transaction Failure Screen

### 3.28 Transaction Details Screen

UI Element: Label, Button, ImageView

Pods utilized: Alamofire – It is used to call API to get the saved bank data.

Definition: User can choose either the saved bank details option in which we get all the details from API or using new account where the user has to enter all the details again.



**Figure 3.28:** Transaction Details Screen

### 3.29 Saved Bank Transaction Screen

UI Element: Label, Button, TextField

Pods utilized: Alamofire – It is used to call API to get the saved bank data.

Definition: All the textfield are automatically completed and are not editable.

< Complete Your Transfer

Total amount to be transferred ₹ 3000

Saved Bank Details

007799079878

Abhimanyu Kapoor

PNB0065 [Find IFSC](#)

3,000

My Saving Account

Proceed

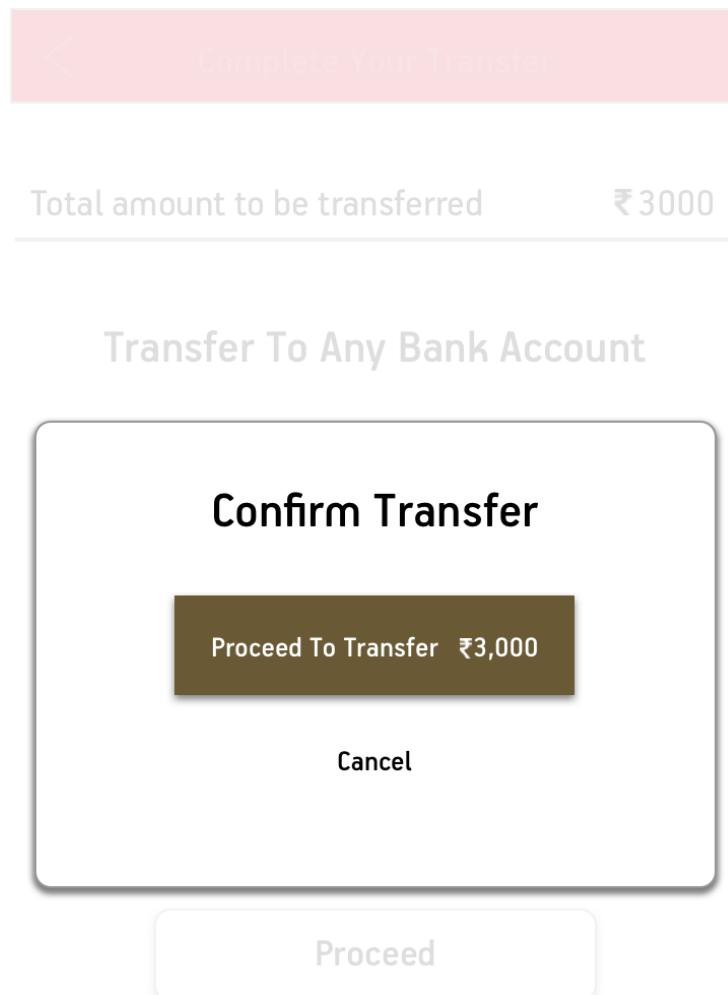
**Figure 3.29:** Saved Bank Transfer Screen



### 3.30 Transaction Confirmation Screen

UI Element: Label, Button

Definition: Transaction is confirmed before processing, backgrounds transparency is set to 0.5

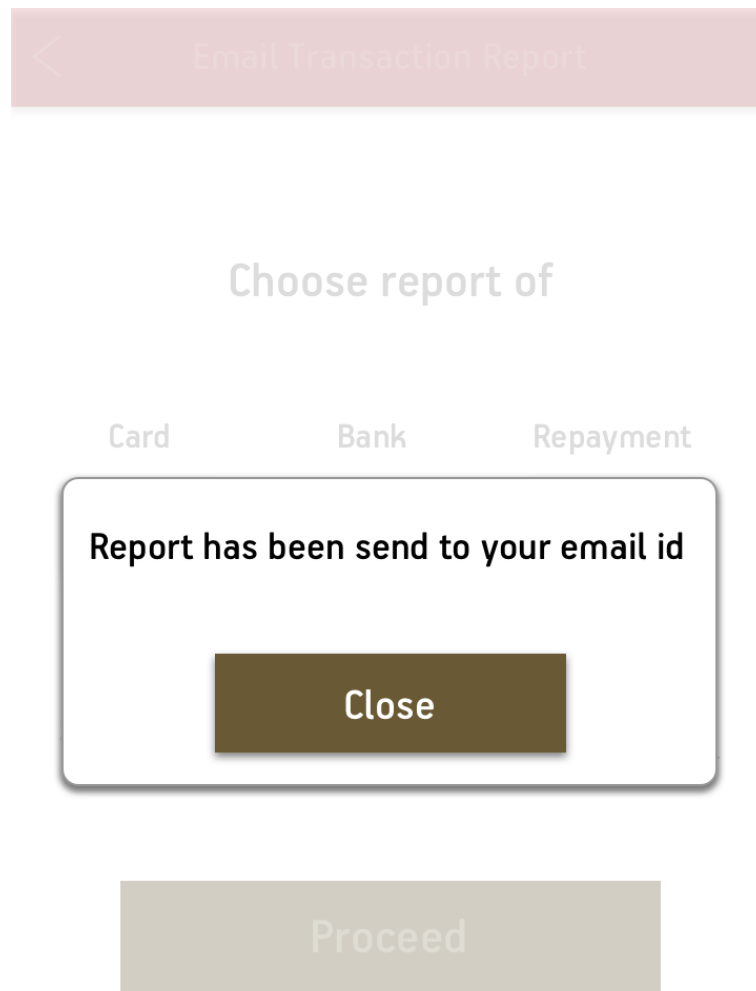


**Figure 3.30:** Transaction Confirmation Screen

### 3.31 Report Sent Screen

UI Element: Label, Button

Definition: This screen is displayed when the report has been successfully sent to the user which is verified by the API request response.



**Figure 3.31:** Report Sent Screen

### 3.32 Launch Screen

Definition: This screen is displayed at the launch of the app.



**Figure 3.32:** Launch Screen

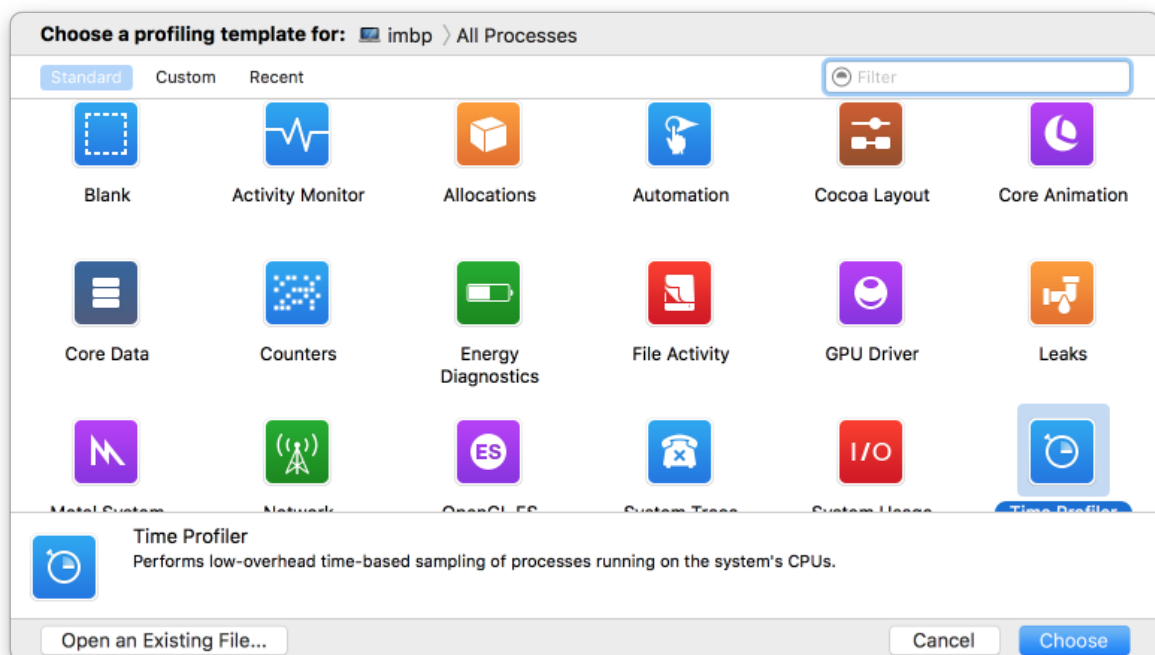
## CHAPTER 4

### PERFORMANCE ANALYSIS

The initial job to analyse any app is to identify un-optimised code that consumes large amount of CPU, GPU or memory. Apple has a great way to accomplish that: Instruments.

There are 4 main areas to focus on:

- CPU (“Time Profiler” tool)
- Power Usage (“Energy diagnostics” tool)
- Memory Usage (“Allocations” tool)
- GPU (“Core Animation” tool)



**Figure 4.1:** Instruments Screen

## **CHAPTER 5**

### **CONCLUSIONS**

#### **5.1 Conclusions**

In conclusion I might want to restore that I effectively finished a live task in the traverse of 4 months. In this 4 months, I took in a great deal, and in that time have changed from an amateur to an expert. Not exclusively did I find out about Protocol Oriented Programming in Swift, yet in addition how to incorporate API in applications. I likewise learnt about enlivening UI components, making geometric shape and vitalizing changes between screen. I am the main understudy who is dealing with the most recent innovation in iOS, FRP.

#### **5.2 Future Scope**

##### **5.2.1 Functional Reactive Programming**

Functional-Reactive Programming is an outline design where sees respond to change to information. These plan designs varies from each other. It is conceivable to utilize both in a solitary undertaking yet it is by and large dodged.

M.V.C. comprises of information Model, View and Controllers. Here, DataModels and Views don't interface with each other. Association in between the two is by means of controllers. FRP is only inverse to it concerning the fact that ViewModel and information can collaborate with others specifically. In quick, FRP is accomplished by Rx-Swift and Rx-Cocoa units.

##### **5.2.1 Swift Lint**

Composing code isn't as intense as composing spotless and justifiable code may be. Each dialect has some great practice of composing code. However, for the most part, designer doesn't take after those rules. To influence designer to power to compose code as indicated by rules, there are such frameworks utilized. For Swift-4, there is swiftlint. Swiftlint check the code for automatic and expressive blunders. This is most useful in recognizing some normal

and phenomenal missteps that are made amid coding. Swiftlint depends on rules from Swift-4 style direct.

## REFERENCES

[1] [https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/index.html#//apple\\_ref/doc/uid/TP40015214-CH2-SW1](https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/index.html#//apple_ref/doc/uid/TP40015214-CH2-SW1)

[2] <https://www.codeschool.com/learn/ios>

[3] <https://medium.com/@ansujain/interfaces-vs-inheritance-in-swift-1f48c85948b8>

[4] <https://medium.com/@ansujain/ios-common-layers-6c73321189ce>

[5] <https://medium.com/@ansujain/solid-principles-and-css-part-1-e73aeeb4bdc5>

[6] <https://medium.com/@ansujain/memory-management-in-ios-240d69f01c1e>

[7] <https://medium.com/@ansujain/ios-application-life-cycle-d517a3c44e7e>