

EXPENSER-PERSONAL FINANCE TOOL

PROJECT REPORT

Submitted in fulfilment for the J Component

of

CAL COURSE

ITE1016-Mobile Application Development

in

B.Tech – Information Technology

By

RAHUL RISHAV MOHANTI (15BIT0070)

Under the guidance of

Prof. Puvirarasi G

SITE



School of Information Technology and Engineering

Winter Semester 2016-17

Table of Contents

Chapter Number	Title	Page Number
	ABSTRACT	3
1	Problem Statement	4
2	System Design	5
3	Implementation	6
	3.1 Source Code	7
4	Results	
	4.1 Screenshots	27
5	Future Scope	32
6	References	32

ABSTRACT

Over the last decade there has been a tremendous improvement and refinement in the quality of life that the majority of the population leads. People have been spending on themselves more than ever and this has paved the way for more and more transaction options. Cash and card have given way to mobile apps such as PayTM, Google's Tez and other such alternate web based transaction apps. With such an increase in expenditure and the ways of transaction, there is a need to track one's personal finance on the go. With the increasing dependence that people have on their phones, a mobile app would best suit our purpose.

The project intends to create an app, that shall track a person's expenditure, by associating every expenditure with a timestamp and the method of transaction. Thus serving as a virtual wallet.

Problem Statement

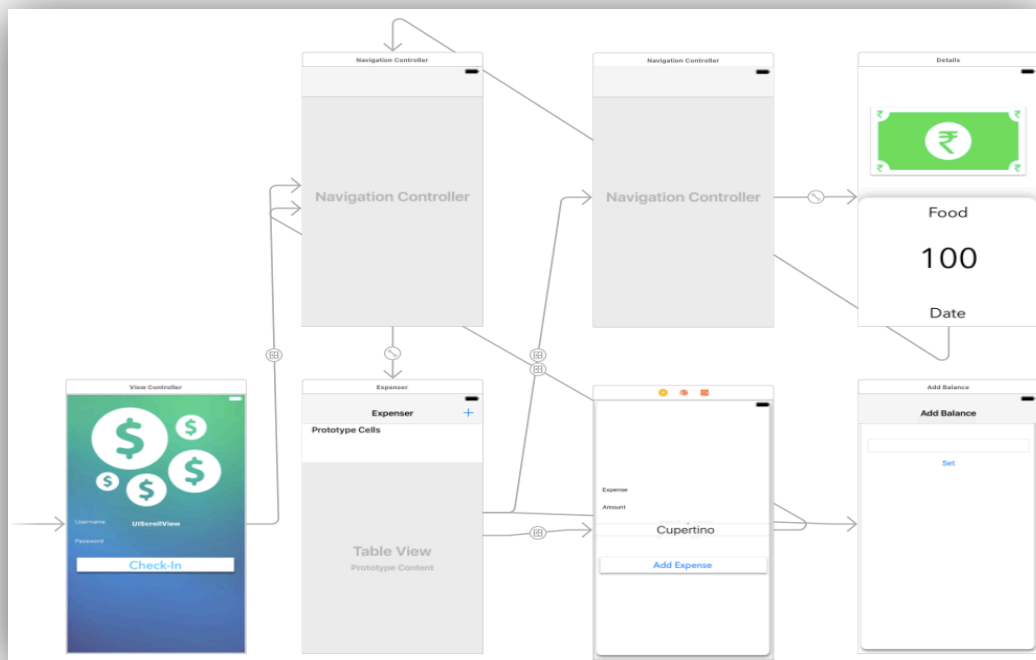
To create a minimal yet functional iOS app to track expenses, visualise spendings, set balance and keep check on the balance amount

Goals

- Option to input spendings with drop down for method of payment.
- Display Balance at all times.
- Expanded view of expenses on clicking on specific date.
- Real Time firebase database to track expenses
- Multiple user support.

System Design

- The application opens up to a Log-In/Sign Up screen. The screens simply requires the user's Email and Password.
- After successful login through Firebase Authentication a segue is initiated to the next View Controller which is a Table View Controller displaying the Current Balance and Expenditure Cards.
- Here the user has the following options:
 - Set Balance: On tapping on the Balance cell, the user is provided with a new View which allows the user to update his/her balance.
 - Add Expense: On tapping the '+' button at the top right corner of the Screen, the user is presented with a card which allows the user to add a new expense by filing in the expenditure name, expenditure amount and the method of payment using the drop down.
 - View Expense: The method of payment is shown in the form of a neat animation followed by a card containing the details of the each expenditure.

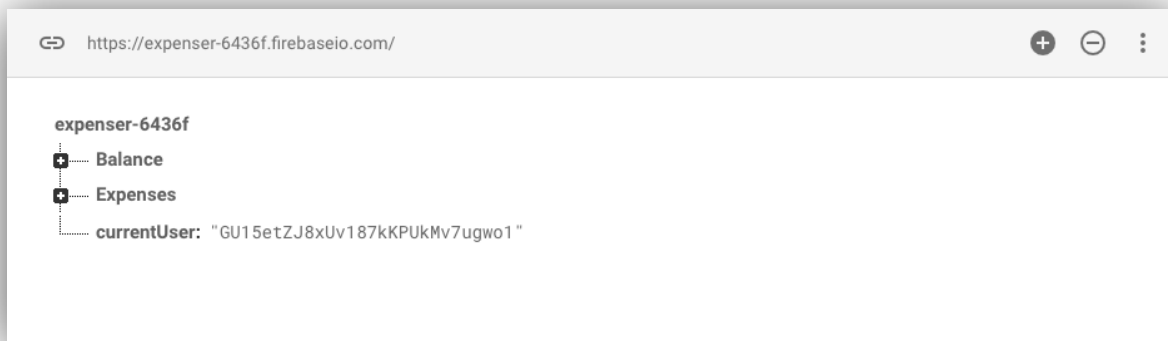


Storyboard Diagram provides the overall flow of the app

Implementation

The UI/UX for the app has been designed on Sketch and then implemented using Apple's Swift language on the Xcode Software. Extensive care has been given to the design of the app. All the icons, animations and background used in the app were designed with utmost attention to detail before implementation of the app.

For the backend a firebase database tree was created for tracking the expenses and balance of each user. The parent node was split into a Balance and Expenses node each containing child nodes for every user. The expenses node would track every expenditure.



During login Firebase Auth is used to verify the user's credentials. This returns a UserID token which is then used throughout the app to make DB transactions for carrying out tasks such as displaying balance, displaying expense cards, adding expenses and updating balance. Various methods and algorithms had been created and explored to deal with the asynchronous work flow of data and also to optimize the codes and database queries.

3.1 Source Code

AppDelegate.swift

```
//  
// AppDelegate.swift  
// Expenser  
//  
// Created by Rahul Rishav Mohanti on 17/09/17.  
// Copyright © 2017 Rahul Rishav Mohanti. All rights reserved.  
//  
  
import UIKit  
import Firebase  
  
@UIApplicationMain  
class AppDelegate: UIResponder, UIApplicationDelegate {  
  
    var window: UIWindow?  
  
    func application(_ application: UIApplication,  
didFinishLaunchingWithOptions launchOptions:  
[UIApplicationLaunchOptionsKey: Any]?) -> Bool {  
        // Override point for customization after application launch.  
        FirebaseApp.configure()  
        return true  
    }  
  
    func applicationWillResignActive(_ application: UIApplication) {
```

interruptions (such as an incoming phone call or SMS message) or when the user quits the application and it begins the transition to the background state.

// Use this method to pause ongoing tasks, disable timers, and invalidate graphics rendering callbacks. Games should use this method to pause the game.

}

func applicationDidEnterBackground(_ application: UIApplication) {

// Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your application to its current state in case it is terminated later.

// If your application supports background execution, this method is called instead of applicationWillTerminate: when the user quits.

}

func applicationWillEnterForeground(_ application: UIApplication) {

// Called as part of the transition from the background to the active state; here you can undo many of the changes made on entering the background.

}

func applicationDidBecomeActive(_ application: UIApplication) {

// Restart any tasks that were paused (or not yet started) while the application was inactive. If the application was previously in the background, optionally refresh the user interface.

}

func applicationWillTerminate(_ application: UIApplication) {

// Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:.

}


```
}
```

//Login page

ViewController.swift

```
//  
// ViewController.swift  
// Expenser  
//  
// Created by Rahul Rishav Mohanti on 17/09/17.  
// Copyright © 2017 Rahul Rishav Mohanti. All rights reserved.  
//  
  
import UIKit  
import Firebase  
import FirebaseDatabase  
import FirebaseAuth  
import Canvas  
var x = String()  
var y = String()  
var bal = ""  
var curr_user = String()  
var y_name = String()  
var y_amt = String()  
var y_method = String()  
var y_date = String()  
class ViewController: UIViewController, UITextFieldDelegate {  
    //@IBOutlet weak var checkInView: CSAnimationView!  
    @IBOutlet weak var Dollar: CSAnimationView!
```

```

@IBOutlet weak var ScrollView: UIScrollView!
    @IBOutlet weak var username: UITextField!
    @IBOutlet weak var Password: UITextField!

    @IBOutlet weak var butt: UIButton!
    override func viewDidLoad() {
        super.viewDidLoad()
        butt.layer.cornerRadius = 4
        // Do any additional setup after loading the view, typically
from a nib.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        textField.resignFirstResponder();

        return true
    }
    func textFieldDidBeginEditing(_ textField: UITextField) {
        if(textField == Password)
        {
            ScrollView.setContentOffset(CGPoint(x: 0, y: 200), animated:
true)
        }

```

```

}

```

```

func textFieldDidEndEditing(_ textField: UITextField) {

    ScrollView.setContentOffset(CGPoint(x: 0, y: 0),
animated:true)

}

@IBAction func checkIn(_ sender: Any)
{ //butt.setBackgroundImage(#imageLiteral(resourceName:
"tapped"), for: UIControlState.selected)
    print("clicked")

    Password.resignFirstResponder()
    Auth.auth().createUser(withEmail: username.text!, password:
Password.text!)
    { (user, error) in
        if error != nil
        {
            self.login()
        }
        else
        {
            print("User Created")
            var ref: DatabaseReference!
            ref = Database.database().reference()

ref.child("Expenses").child((user?.uid)!).child("0").updateChildValues
(["expense":"Food 200 Cash 6th"])

```

```

ref.child("Balance").updateChildValues([(user?.uid)!:"0"])

```

```

        self.login()
    }
}

}

func login(){
    Auth.auth().signIn(withEmail: username.text!, password:
Password.text!) { (user, error) in
        if error != nil
        {
            print("Incorrect Pass")
        }
        else
        {
            curr_user = (user?.uid)!
            //print (user?.email ?? "rahul")
            var ref: DatabaseReference!
            ref = Database.database().reference()
            ref.updateChildValues(["currentUser":user?.uid ??
"none"])

            x = (user?.uid)! as? String ?? ""
            self.Dollar.type = CSAnimationTypeZoomIn
            self.Dollar.duration = 2
            self.Dollar.delay = 0
            self.Dollar.startCanvasAnimation()
            self.performSegue(withIdentifier: "firstSegue",
sender: nil)

        }
    }
}

```

```
}
```

//Home Page

```
TVC.swift
//
//  TVC.swift
//
//
//  Created by Rahul Rishav Mohanti on 08/10/17.
//
//

import UIKit
import Firebase
import FirebaseDatabase
import FirebaseAuth
class TVC: UITableViewController {
    var ref: DatabaseReference!
    var refHandle = UInt()
    var refHandle2 = UInt()
    var expenseList = [User]()

    override func viewDidLoad() {
        super.viewDidLoad()
        ref = Database.database().reference()
        print("hey");
    }
}
```

```

        fetchFinance()

        // Uncomment the following line to preserve selection between
presentations

        // self.clearsSelectionOnViewWillAppear = false

        // Uncomment the following line to display an Edit button in
the navigation bar for this view controller.

        // self.navigationItem.rightBarButtonItem =
self.editButtonItem()
    }

    func fetchFinance(){
        print("fetchFinance called");

        refHandle =
ref.child("Expenses").child(x).observe(.childAdded, with: { (snapshot)
in
    if let dictionary = snapshot.value as? [String :
AnyObject]
    {
        print(dictionary)
        print("hey")
        let user = User()
        user.setValuesForKeys(dictionary)
        self.expenseList.append(user)
        print(self.expenseList[0].expense ?? "x")
        DispatchQueue.main.async() {
            print("hey2")

            self.tableView.reloadData()

        }

    }

})

ref.child("Balance").observe(.value, with: { (snap) in

```

```

        let value = snap.value as? NSDictionary
        bal = value?[x] as? String ?? ""
        DispatchQueue.main.async() {
            print("hey2")

            self.tableView.reloadData()
        }
    })
}

override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarning()
    // Dispose of any resources that can be recreated.
}

// MARK: - Table view data source

override func numberOfSections(in tableView: UITableView) -> Int {
    // #warning Incomplete implementation, return the number of
sections
    return 1
}

override func tableView(_ tableView: UITableView,
numberOfRowsInSection section: Int) -> Int {
    // #warning Incomplete implementation, return the number of
rows
    return expenseList.count+1
}

```

```

        override func tableView(_ tableView: UITableView, cellForRowAt
indexPath: IndexPath) -> UITableViewCell {

            /* let cell = UITableViewCell(style: .subtitle,
reuseIdentifier: "cell")

            // Set Cell Contents
            y = expenseList[indexPath.row].expense!
            var temp = y.characters.split{$0 == " "}.map(String.init)

            cell.textLabel?.text = temp[0] + " " + temp[1]*/

            if(indexPath.row>0)
            {
                let cell = Bundle.main.loadNibNamed("tvcell", owner: self,
options: nil)?.first as! tvcell
                y = expenseList[indexPath.row-1].expense!
                var temp = y.characters.split{$0 == " "}.map(String.init)

                cell.labelexpense.text! = temp[0] + " " + temp[1]

                return cell
            }
            else{
                let cell = UITableViewCell(style: .subtitle,
reuseIdentifier: "cell")
                y = "Balance:" + bal
                cell.textLabel?.text = y;
                return cell
            }
        }

        override func tableView(_ tableView: UITableView, didSelectRowAt
indexPath: IndexPath) {
            if(indexPath.row>0)
            {

```



```

        y = expenseList[indexPath.row-1].expense!
        var yarr = y.characters.split{$0 == " "}.map(String.init)
        y_name = yarr[0]
        y_amt = yarr[1]
        y_method = yarr[2]
        y_date = yarr[3]
        self.performSegue(withIdentifier: "seguecell", sender: nil)
    }
    else
    {
        self.performSegue(withIdentifier: "addBalSeg", sender:
nil)
    }
}

/*
    override func tableView(_ tableView: UITableView, cellForRowAt
indexPath: IndexPath) -> UITableViewCell {

        let cell = tableView.dequeueReusableCell(withIdentifier:
"reuseIdentifier", for: indexPath)

        // Configure the cell...

        return cell
    }
    */

/*
    // Override to support conditional editing of the table view.
    override func tableView(_ tableView: UITableView, canEditRowAt
indexPath: IndexPath) -> Bool {

        // Return false if you do not want the specified item to be
editable.

```

```

        return true
    }
    */

    /*
    // Override to support editing the table view.
    override func tableView(_ tableView: UITableView, commit
    editingStyle: UITableViewCellEditingStyle, forRowAt indexPath:
    IndexPath) {
        if editingStyle == .delete {
            // Delete the row from the data source
            tableView.deleteRows(at: [indexPath], with: .fade)
        } else if editingStyle == .insert {
            // Create a new instance of the appropriate class, insert
            it into the array, and add a new row to the table view
        }
    }
    */

    /*
    // Override to support rearranging the table view.
    override func tableView(_ tableView: UITableView, moveRowAt
    fromIndexPath: IndexPath, to: IndexPath) {

    }
    */

    /*
    // Override to support conditional rearranging of the table view.
    override func tableView(_ tableView: UITableView, canMoveRowAt
    indexPath: IndexPath) -> Bool {
        // Return false if you do not want the item to be re-
        orderable.

```

```

        return true
    }
    */

    /*
    // MARK: - Navigation

    // In a storyboard-based application, you will often want to do a
    little preparation before navigation
    override func prepare(for segue: UIStoryboardSegue, sender: Any?)
    {
        // Get the new view controller using
        segue.destinationViewController.
        // Pass the selected object to the new view controller.
    }
    */
}

```

//Expanded Expense View

cellView.swift

```

//
//  cellView.swift
//  Expenser
//
//  Created by Rahul Rishav Mohanti on 16/10/17.
//  Copyright © 2017 Rahul Rishav Mohanti. All rights reserved.
//

import UIKit

```

```

class cellView: UIViewController {

    @IBOutlet weak var payMethod: UIImageView!
    @IBOutlet weak var ExpenseName: UILabel!
    @IBOutlet weak var ExpenseAmt: UILabel!
    @IBOutlet weak var ExpenseDate: UILabel!
    override func viewDidLoad() {
        super.viewDidLoad()
        ExpenseName.text = y_name
        ExpenseDate.text = y_date
        ExpenseAmt.text = y_amt
        switch(y_method){
            case "Cash": payMethod.image = UIImage(named: "rupee")!;break
            case "Debit": payMethod.image = UIImage(named: "debit")!;break
            case "Credit": payMethod.image = UIImage(named:
"credit")!;break
            case "Cheque": payMethod.image = UIImage(named:
"cheque")!;break
            default: payMethod.image = UIImage(named: "rupee")!; break
        }
        // Do any additional setup after loading the view.
    }

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    /*
    // MARK: - Navigation

```

```

        // In a storyboard-based application, you will often want to do a
        little preparation before navigation
        override func prepare(for segue: UIStoryboardSegue, sender: Any?)
        {
            // Get the new view controller using
            segue.destinationViewController.
            // Pass the selected object to the new view controller.
        }
    */
}

```

//New Expense Form
addExpenseView.swift

```

//
//  addExpenseView.swift
//  Expenser
//
//  Created by Rahul Rishav Mohanti on 18/10/17.
//  Copyright © 2017 Rahul Rishav Mohanti. All rights reserved.
//

import UIKit
import FirebaseDatabase
import Firebase
class addExpenseView: UIViewController, UIPickerViewDelegate,
UIPickerViewDataSource, UITextFieldDelegate {

    @IBOutlet weak var payMethod: UIPickerView!
    @IBOutlet var expense_name: UITextField!

```

```

@IBOutlet var scrollView: UIScrollView!
@IBOutlet var expense_amt: UITextField!
var array = ["Cash","Debit","Credit","Cheque"]
var payM = ""
override func viewDidLoad() {
    super.viewDidLoad()
    payMethod.delegate = self
    payMethod.dataSource = self
    // Do any additional setup after loading the view.
}
func textFieldShouldReturn(_ textField: UITextField) -> Bool {
    textField.resignFirstResponder();

    return true
}
func textFieldDidBeginEditing(_ textField: UITextField) {
    if(textField == expense_amt || textField == expense_name)
    {
        scrollView.setContentOffset(CGPoint(x: 0, y: 100),
animated: true)
    }
}

func textFieldDidEndEditing(_ textField: UITextField) {

    scrollView.setContentOffset(CGPoint(x: 0, y: 0),
animated:true)

}

```

```

    override fun didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

    fun pickerView(_ pickerView: UIPickerView, titleForRow row: Int,
forComponent component: Int) -> String? {
        return array[row]
    }

    fun pickerView(_ pickerView: UIPickerView,
numberOfRowsInComponent component: Int) -> Int {
        return array.count
    }

    fun numberOfComponents(in pickerView: UIPickerView) -> Int {
        return 1
    }

    fun pickerView(_ pickerView: UIPickerView, didSelectRow row: Int,
inComponent component: Int) {
        payM = array[row]
    }

    @IBAction fun addExp(_ sender: Any) {
        let date = Date()
        let formatter = DateFormatter()
        formatter.dateFormat = "dd_MM_yyyy"
        let result = formatter.string(from: date)

        var final = expense_name.text! + " " + expense_amt.text! + " "
+ payM + " " + result
        print(final)
        var ref: DatabaseReference!
    }

```

```

        ref = Database.database().reference()

ref.child("Expenses").child(curr_user).child(expense_name.text!).updateChildValues(["expense":final])

        ref.child("Balance").observeSingleEvent(of: .value, with:
{ (snapshot) in
            // Get user value
            let value = snapshot.value as? NSDictionary
            var prevBal = Int(value?[x] as? String ?? "")
            prevBal = prevBal! - Int(self.expense_amt.text!)
            var fbal = String(prevBal!)
            ref.child("Balance").updateChildValues([x:fbal ?? ""])
            self.performSegue(withIdentifier: "addSegue", sender: nil)
        }) { (error) in
            print(error.localizedDescription)
        }

    }

    /*
    // MARK: - Navigation

    // In a storyboard-based application, you will often want to do a
    little preparation before navigation
    override func prepare(for segue: UIStoryboardSegue, sender: Any?)
    {
        // Get the new view controller using
        segue.destinationViewController.
        // Pass the selected object to the new view controller.
    }
    */
}

```


//SetBalance Form

addBalView.swift

```
//
//  addBalView.swift
//  Expenser
//
//  Created by Rahul Rishav Mohanti on 09/11/17.
//  Copyright © 2017 Rahul Rishav Mohanti. All rights reserved.
//

import UIKit
import Firebase
import FirebaseDatabase

class addBalView: UIViewController {

    @IBOutlet var balVal: UITextField!
    @IBAction func setBalance(_ sender: Any) {
        var ref: DatabaseReference!
        ref = Database.database().reference()
        ref.child("Balance").updateChildValues([(x):balVal.text ?? 0])
    }
    override func viewDidLoad() {
        super.viewDidLoad()

        // Do any additional setup after loading the view.
    }
}
```

```

    override func didReceiveMemoryWarning() {
        super.didReceiveMemoryWarning()
        // Dispose of any resources that can be recreated.
    }

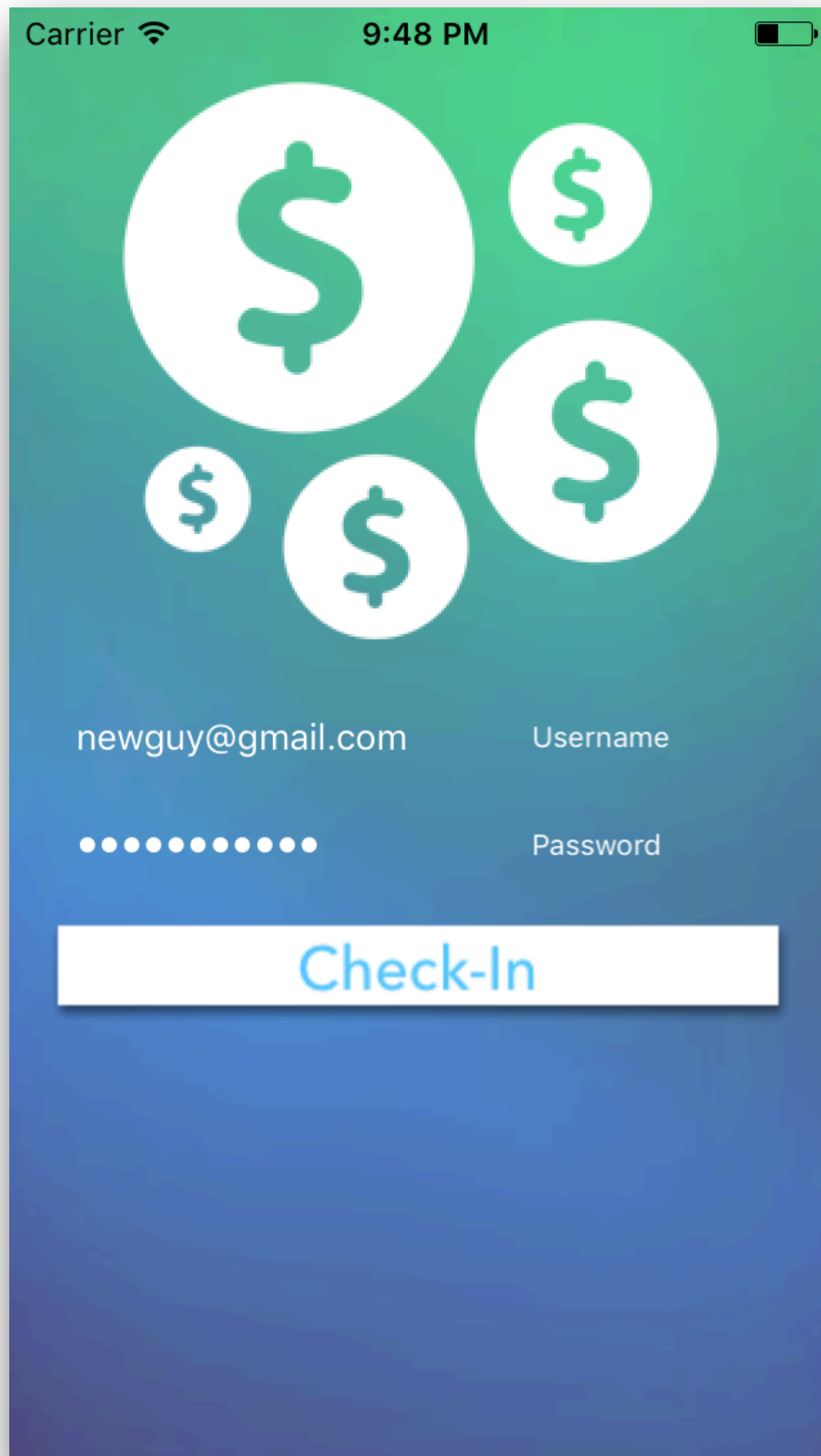
    /*
    // MARK: - Navigation

    // In a storyboard-based application, you will often want to do a
    little preparation before navigation
    override func prepare(for segue: UIStoryboardSegue, sender: Any?)
    {
        // Get the new view controller using
        segue.destinationViewController.
        // Pass the selected object to the new view controller.
    }
    */
}

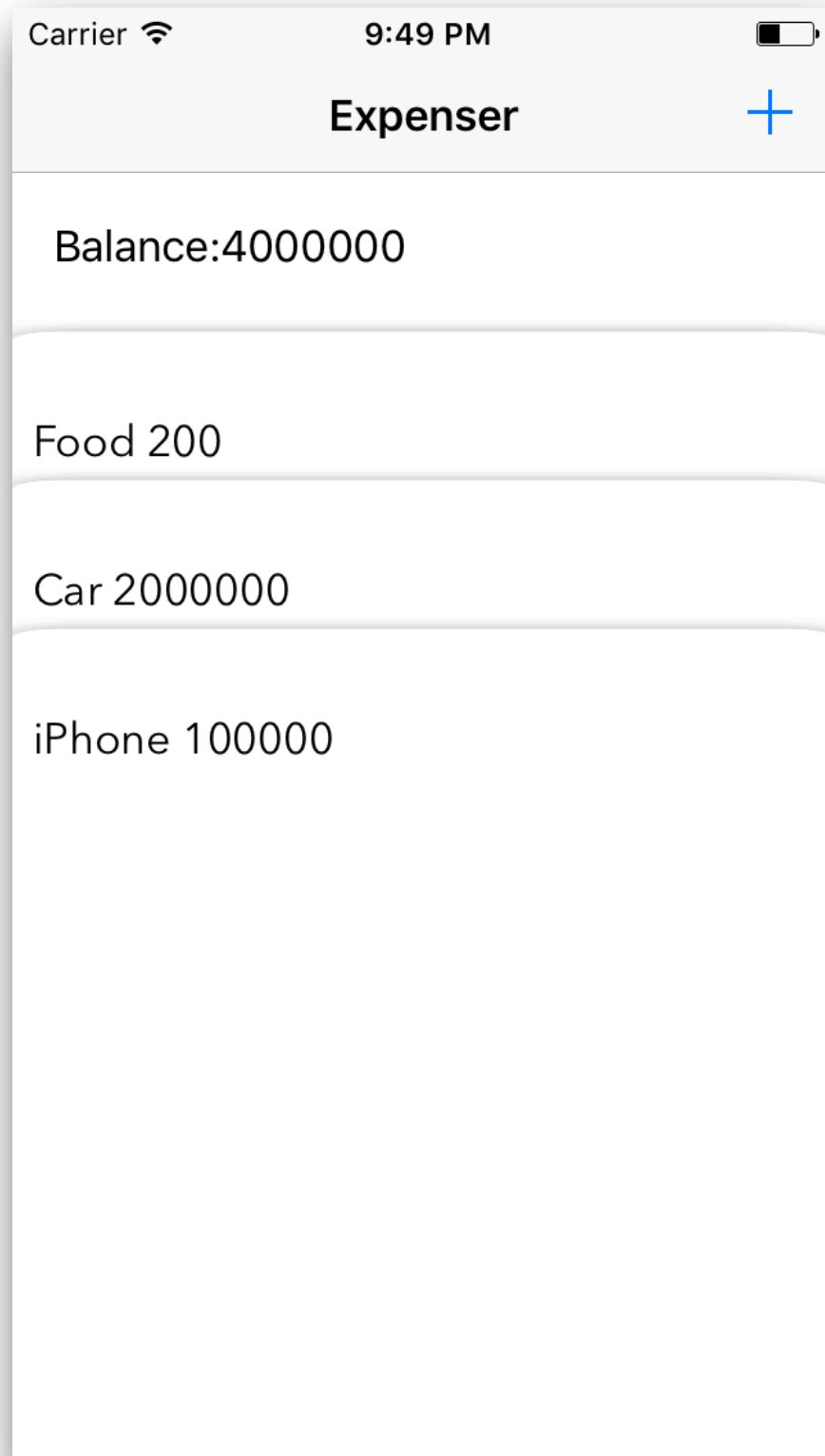
```

Screenshots:



Login/Sign Up Screen



Welcome Screen



Add Expense Screen

Carrier  9:49 PM 

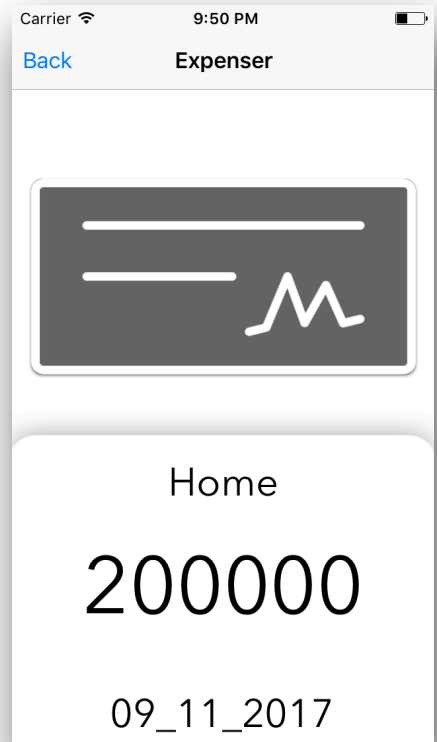
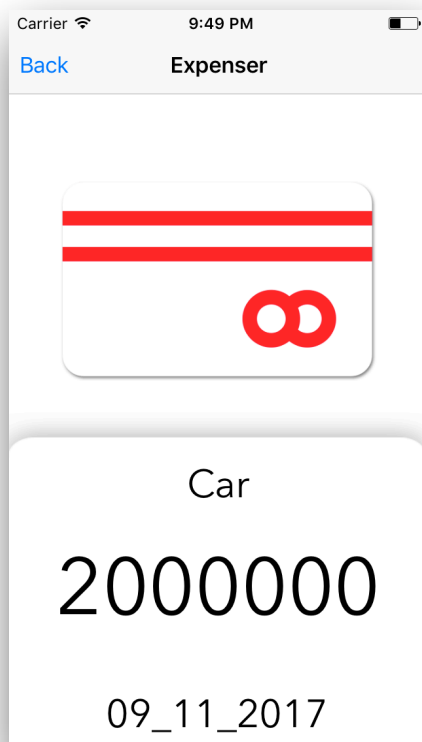
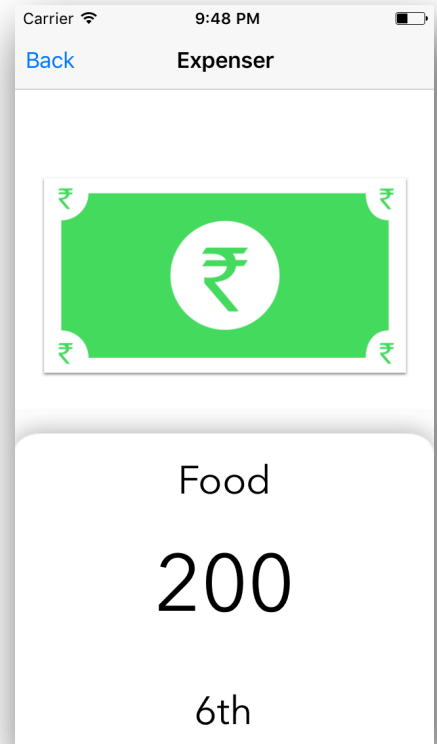
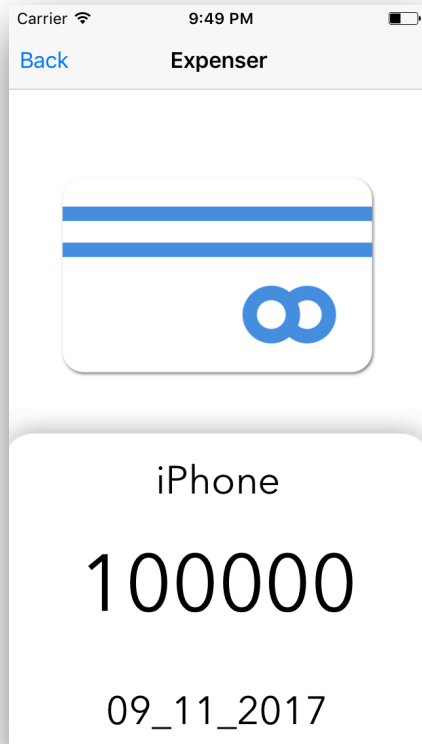
[< Expenser](#)

Car	Expense
2000000	Amount

Credit

Add Expense

Different expense Views



Update Balance Screen

The image shows a mobile application interface for updating a balance. At the top, a status bar displays 'Carrier' with a signal icon, the time '9:50 PM', and a battery level icon. Below this is a navigation bar with a blue back arrow, the text 'Expenser' in blue, and 'Add Balance' in black. The main content area has a title 'Set Balance' in a large, black, sans-serif font. Underneath the title is a wide, empty text input field with rounded corners and a thin gray border. At the bottom of the input area is a blue 'Set' button.

Carrier 9:50 PM

< Expenser Add Balance

Set Balance

Set

Future Scope

The app in its present state has a minimal yet functional approach. In the future the app can be made to connect with third party apps and even Apple Pay in order to automate the process of balance maintenance. Further an Apple Watch app can be created to make the app usable on the go. The UI/UX can be refined and the code can be refactored before being published on the App Store. The app although in its beta stage is unique in its functionality and thus has tremendous potential if evolved in the right ways.

References

<https://www.youtube.com/user/Archetapp>

<https://developer.apple.com/library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/>

<https://www.raywenderlich.com>

<https://firebase.google.com/docs/>

<https://stackoverflow.com/questions/18153702/uitableview-mix-of-static-and-dynamic-cells>

