

# Project Report

Web App Link: <https://splitpayatul.herokuapp.com>

AWS Web App: [Splitpaynode-env.eba-nseqjsmp.ap-south-1.elasticbeanstalk.com](https://splitpaynode-env.eba-nseqjsmp.ap-south-1.elasticbeanstalk.com)

GitHub Repository: <https://github.com/atulragarwal/SplitPay-Internship-Saksoft>

## 1. Introduction

### 1. Purpose:

The purpose of the document is to state and identify the technologies, functional requirements, techniques used during the development of this project over the course of this internship. The product is a payments ledger for university students which maintains payments record between users.

### 2. Aim:

The aim of the project was to develop a Full Stack Web application, focusing primarily on backend development using JavaScript as the core language.

### 3. Objectives:

The objectives for the project are stated as:

- i. To understand and implement CRUD operations using NodeJS and MongoDB for the application.
- ii. To develop and maintain a non-relational database and to provide secure connectivity to the application.
- iii. To ensure the compliance with Schneiderman's Golden Rules for the frontend interface.
- iv. To implement NodeJS and ExpressJS middleware for functionalities such as Password hashing, file uploads, data fetching etc.
- v. To deploy and maintain the application using CI/CD techniques.
- vi. To deploy a cloud-based database system to maintain a secure cluster for all user data required for proper functioning of the application.

### 4. About SplitPay

SplitPay is a payments ledger. It aims to make the payments between university students' error free. A common scenario would be where one member of the group pay the bill for all after a meal. Any single user can simply add this payment to the Splits section of the web application. It is automatically added for all within the database. Hence, users can simply refer to the application to check their balance with other and settle them. This removes any confusion or misunderstanding between the group.

## 2. Project Development

### 1. Functional Requirements

Any user of the product will have some basic requirements and expectations to interact and use the product efficiently. This section specifies the function and methods expected to be implemented during this project:

- a. Login/Register
- b. Make Payment
- c. Make Split
- d. Settle Up
  - i. Cash
  - ii. Online UPI (Testing)
- e. View Profile

### 2. Development Environment

The main aim is to implement the project using MERN Stack technologies. However, due to time constraints and given that the frontend development is done, the backend technologies of MERN Stack, such as NodeJS and MongoDB will be implemented first.

The project is a web-based application with the following languages / technologies used for development:

- a. Frontend Development:
  - i. HTML
  - ii. CSS
  - iii. JavaScript

b. Backend Development:

- i. NodeJS
- ii. MongoDB
- iii. Express JS

The development environment used will be Visual Studio Code and the development progress will be stored/updated on GitHub.

The application's CI/CD operations are completed using Travis CI and the site is deployed using Heroku.

### 3. Features Specification

#### a. Login/Register

The project will require that a user registers on the portal to access other features of the site. This makes identification and record fetching from the database possible. Login/Register feature will represent the Create Operation in CRUD.

#### b. View Home

The landing page displaying the transaction history of the logged in user, providing them with the option to make new payment or new split.

#### c. Make payment

To create and update a payment record between two users. A record is created between the two users in the database and updated as the user makes new operations.

#### d. Add Splits

A feature allowing users to make splits among a group. The amount is split evenly among all and the transaction database is updated accordingly.

#### e. Settle Up

A feature allowing users to settle up their balance with any other user. There are two options to settle up. With cash or an online payment. However, the online payments option is only in testing mode as the API used requires registered secret key for production use.

#### f. View Profile

Allows a user to view their details stored in the database.

### 3. Resources Used

1. Official Documentations:
  - a. NodeJS: <https://nodejs.org/en/docs/guides/>
  - b. ExpressJS: <https://expressjs.com/en/guide/routing.html>
  - c. MongoDB: <https://docs.mongodb.com/manual/>
  - d. BcryptJS: <https://www.npmjs.com/package/bcryptjs>
  - e. AWS: AWS Academy Cloud Foundations
  - f. Travis CI: <https://docs.travis-ci.com/user/languages/javascript-with-nodejs/>
  - g. Travis CI/Heroku: <https://docs.travis-ci.com/user/deployment/heroku/>
  - h. Deployment Guide: <https://www.freecodecamp.org/news/learn-how-to-automate-deployment-on-github-pages-with-travis-ci/>
  - i. Multer NodeJS: <https://www.npmjs.com/package/multer>

## 2. Video Tutorials:

### a. Telusko CRUD in NodeJS and MongoDB:

<https://www.youtube.com/watch?v=eYVGoXPq2RA>

### b. Technical Thapa MongoDB Operations:

<https://youtube.com/playlist?list=PLwGdqUZWnOp1P9xSsJg7g3AY0CUjs-WOa>

### c. JWT Token Generation:

<https://www.youtube.com/watch?v=mbsmsi7l3r4>

### d. Multer File Upload:

[https://www.youtube.com/watch?v=sUUgbcHm\\_3c](https://www.youtube.com/watch?v=sUUgbcHm_3c)

### e. AWS Multer S3 Integration: [https://youtu.be/NZEIlg91l\\_ms](https://youtu.be/NZEIlg91l_ms)

- Variable Names in ENV file are fixed
- `AWS_ACCESS_KEY_ID`
- `AWS_SECRET_ACCESS_KEY`

### f. What Is CDN: <https://youtu.be/Bsq5cKkS33I>

### g. AWS CloudFront :

- [https://youtu.be/kky6zjL5\\_tU](https://youtu.be/kky6zjL5_tU)
- <https://youtu.be/sQNONcj0cvc>

### h. Deploy on AWS Elastic Beanstalk with GitHub Integration:

<https://youtu.be/b0g-FJ5Zbb8>

#### a. Engine Variable/Object in package.json file compulsory:

```
"engines": {  
  "node": "14.16.1"  
},
```

### i. Deploy on AWS EC2 Instance:

#### a. <https://youtu.be/46mFdtpy3NQ>

#### b. <https://youtu.be/FanoTGjkhQ>

#### c. <https://jasonwatmore.com/post/2019/11/18/react-nodejs-on-aws-how-to-deploy-a-mern-stack-app-to-amazon-ec2>

### 3. Deployment Resources Used:

#### a. Travis CI:

Ingrates GitHub with Heroku wo provide a consistent pipeline. On a successful push to GitHub Travis runs production commands to build the application and proceeds to deploy to selected service (Heroku). Provides with the ability to merge multiple GitHub pushes into a single pipeline.

#### b. Heroku:

Deployment server for the web application. Maintains a backend server running the NodeJS files while responding to a user's requests. Integrated with Travis CI using secret key to provide continuous deployment ensuring that the application is updated to current specifications.

#### c. MongoDB Atlas:

Cloud based non-relational database service provider. Uses services from AWS DynamoDB to maintain the could database. Connected to the application using a key based connection and provides data storage facility.

#### d. AWS Elastic Beanstalk:

EBS is an AWS service allowing users to deploy and host their web application to the AWS Cloud. It uses Two major AWS services namely, EC2 and S3. EC2 is a virtual server that will run the backend application. It can be configured specifically for the application and provides developers with the ability to scale their application based on the demand. AWS S3 is a storage service. EBS uses S3 to store and retrieve the static files of the application whenever required. In all EBS provides a complete deployment solution.



e. AWS CodePipeline:

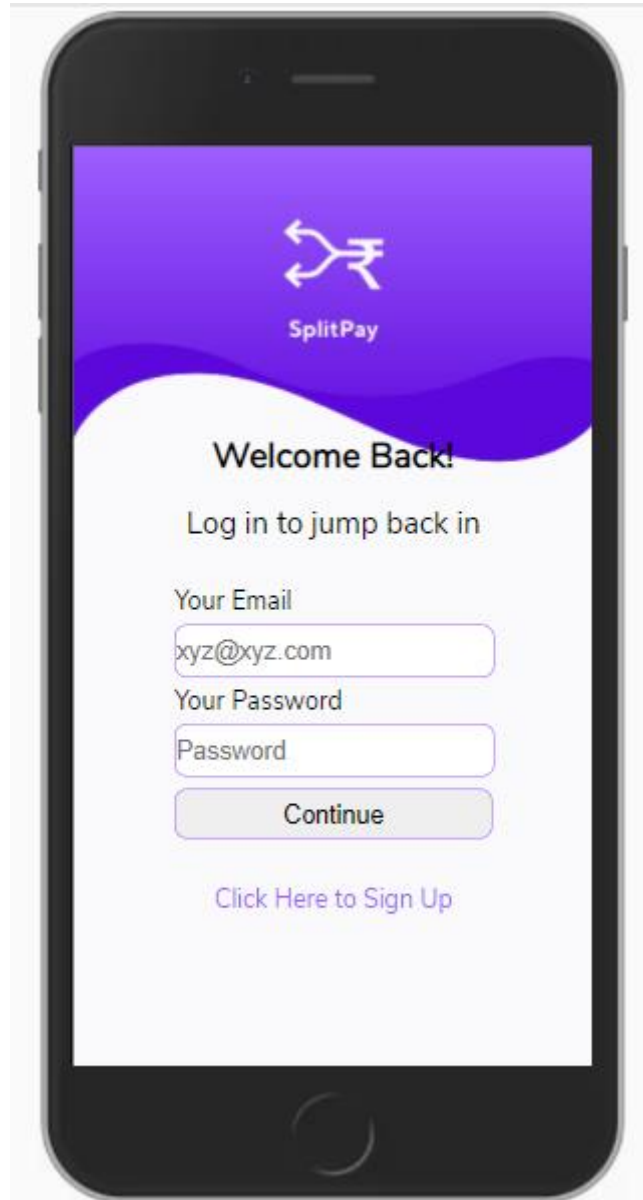
AWS CodePipeline works like Travis CI, integrating GitHub code to the EBS deployment. When code is pushed to GitHub, CodePipeline fetches the code, runs it and deploys the update to EBS hence achieving continuous integration and deployment.

f. AWS S3:


When a user uploads an image to the application, its details are stored in the MongoDB cloud however the image is stored in a S3 bucket made for the application. The image(objects) is added and fetched using their name, which becomes a key. A specific policy needs to be created for an application user to prevent misuse. Such a policy is created using AWS IAM console and restricts any user to only add and see an object in the bucket. The user key is passed along with every request as the bucket is private and not publicly accessible.

## 4. Screenshots

### Login Page



## Sign Up Page:



SplitPay

**Welcome!**

Let's get you started

Your Full Name

Your Phone Number

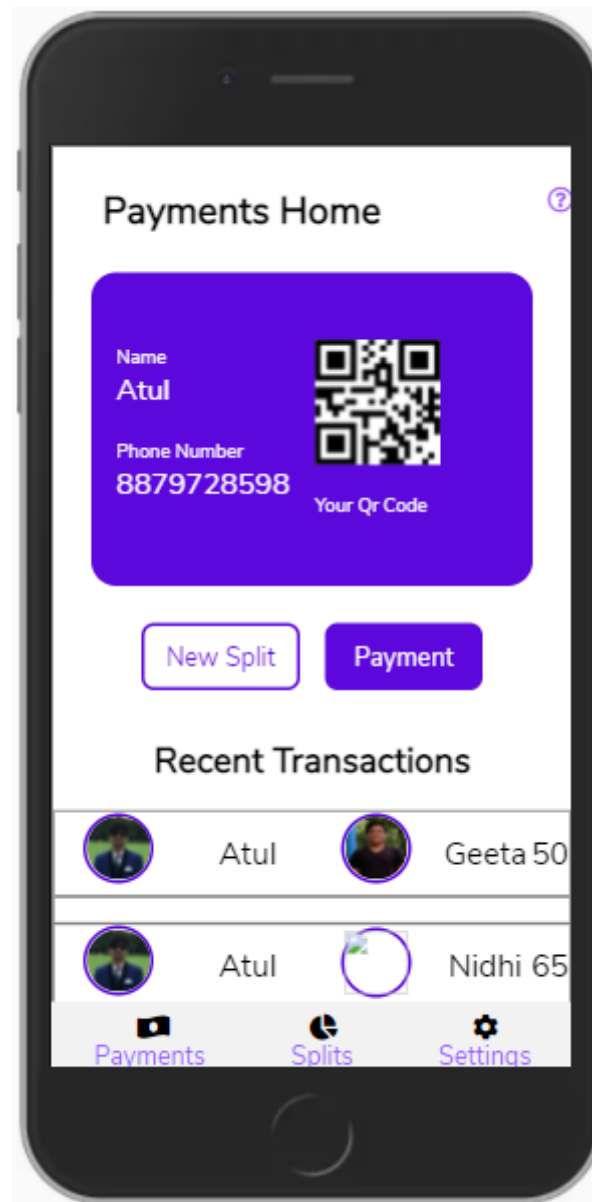
Your Email

Password

Confirm Password

Add Image

## Home Page:



## Make Payments

← Make Payments ?

Enter Phone Number:

XXXXXX XXXXX

Enter Amount

Rs.XX.XX

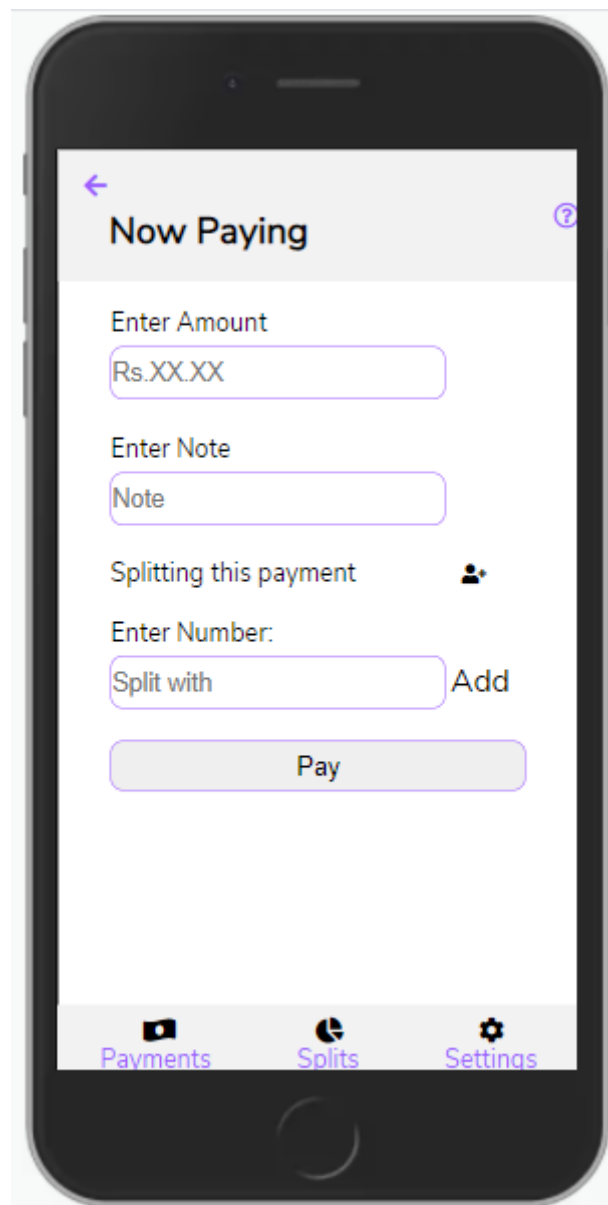
Enter Note

Note

Pay

Payments Splits Settings

## Make Splits




The image shows a smartphone screen with a financial application interface. The screen is titled "Now Paying" in a grey header bar, with a back arrow on the left and a help icon on the right. Below the header, there are three input fields: "Enter Amount" with a placeholder "Rs.XX.XX", "Enter Note" with a placeholder "Note", and "Splitting this payment" with a person icon. Below these is a section "Enter Number:" with a "Split with" input field and an "Add" button. At the bottom of the form is a large "Pay" button. The bottom of the screen features a navigation bar with three icons and labels: "Payments" (card icon), "Splits" (pie chart icon), and "Settings" (gear icon).

← Now Paying ?




Enter Amount  
Rs.XX.XX

Enter Note  
Note

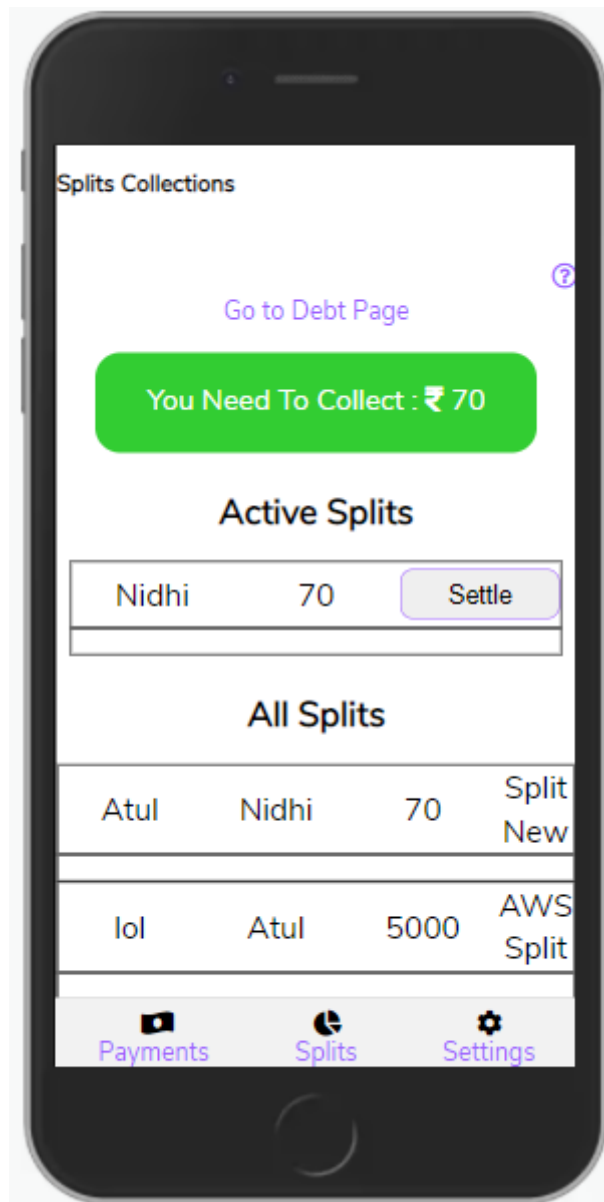
Splitting this payment 

Enter Number:  
Split with Add

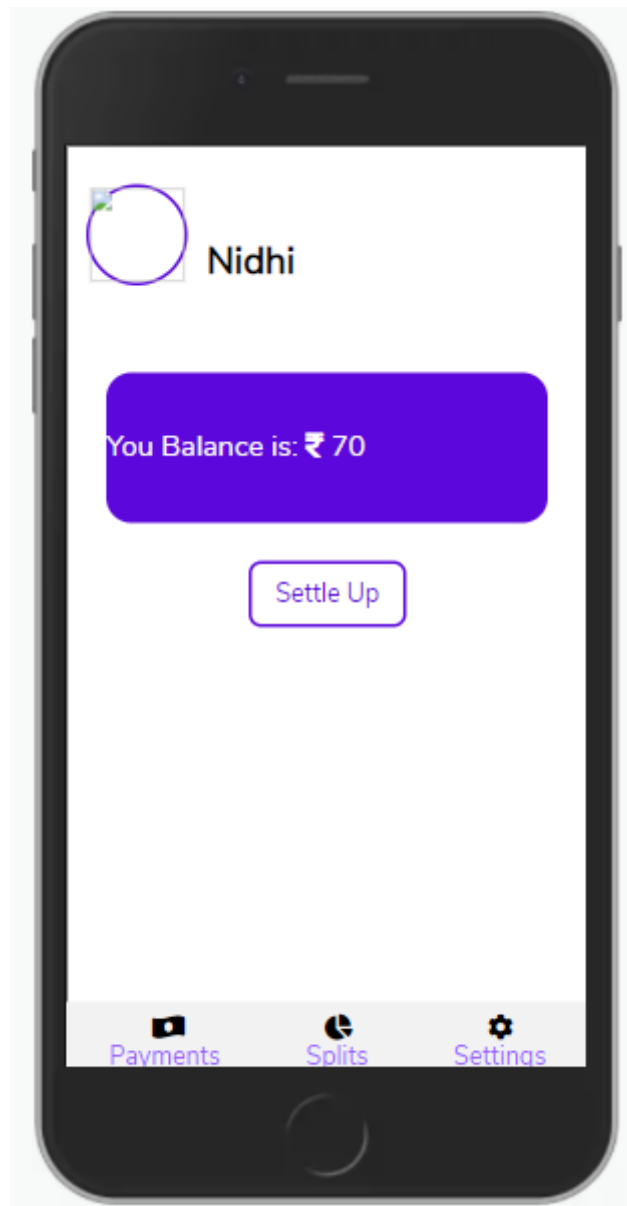
Pay

 Payments  Splits  Settings

## Split Collection View

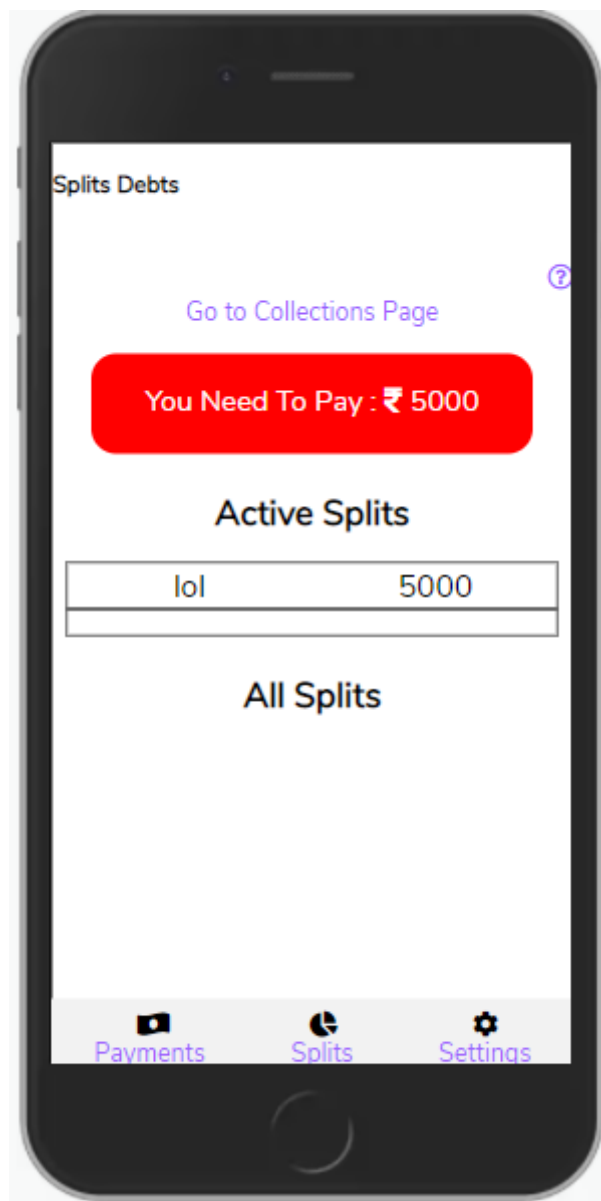


## Settle Split

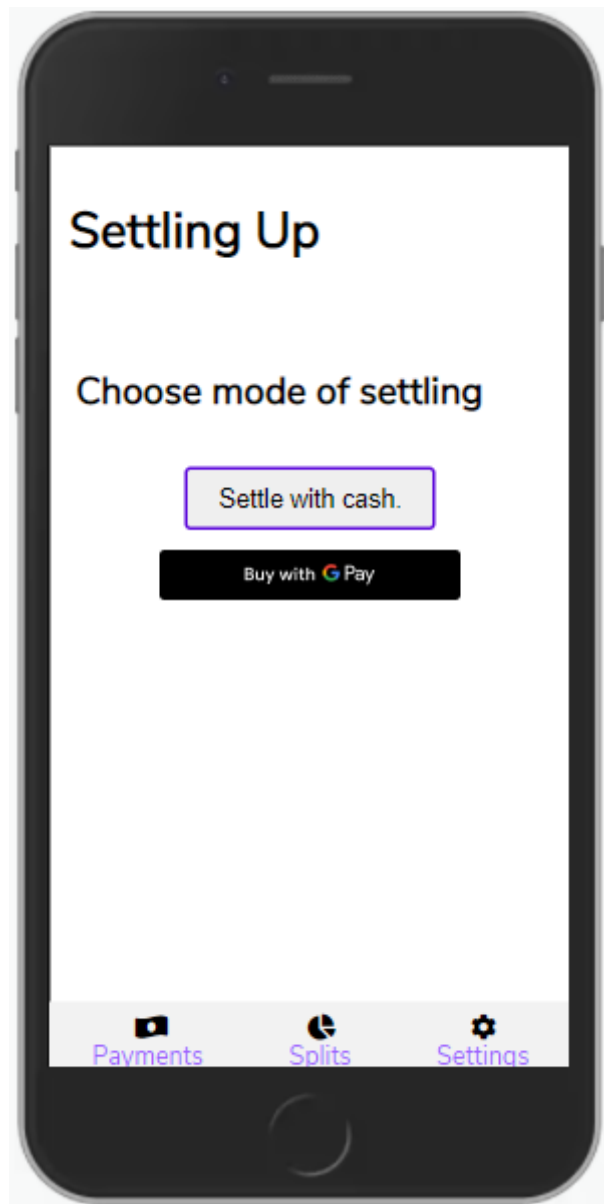




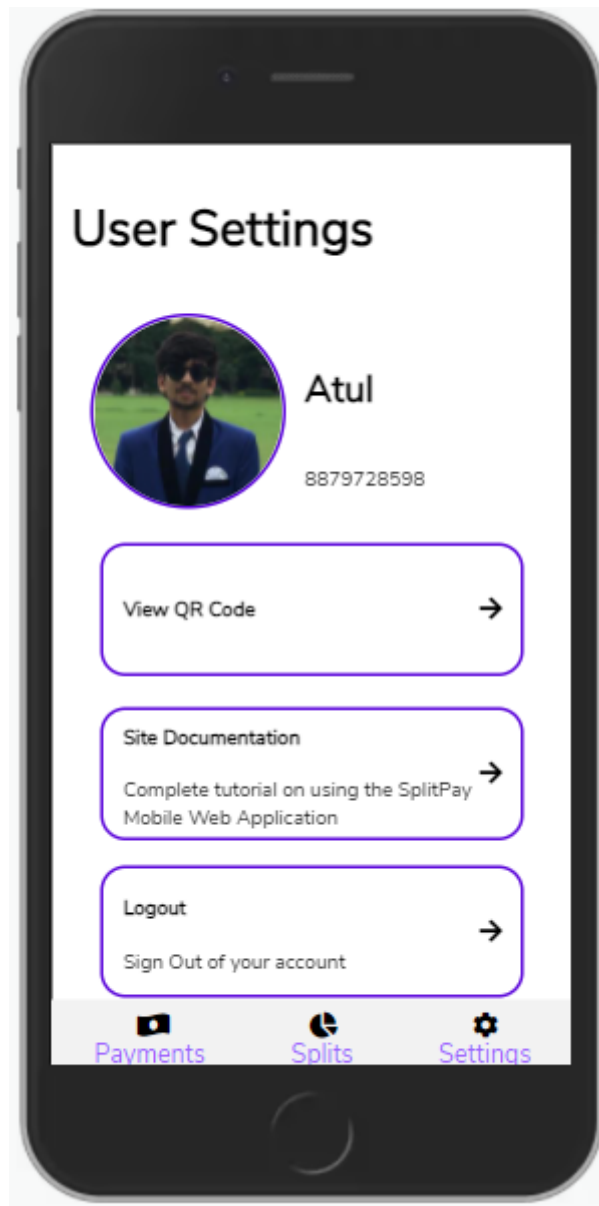
## Split Debts:



## Payment Methods:



## User Settings:



## 5. Progress

### Day 1:

1. Creating the project requirements document and establishing a project schedule.
2. Creating a GitHub repository
3. Establishing a file structure for the application

### Day 2:

- a. Beginning the implementation of CRUD operations for a module.
- b. Creating database schema and establishing application base file and system.
- c. Implementing Create and Read operations for a simple payment module.

### Day 3:

- a. Integrating the completed CRUD operation with existing frontend work
- b. Completing Render with frontend pages
- c. Initiating Update and Delete Operations for the module

### Day 4:

- a. Completion of complete CRUD Module
- b. Unit testing the implemented module
- c. Complete implementation and integration of Update and Delete module with the existing application framework

### Day 5:

- a. Research on express middleware for login token generation and authentication.
- b. Implementing Signup module and updating MongoDB database for the same
- c. Integrating signup module with the existing framework

### Day 6:

- a. Initiating implementation of Login Module for the application
- b. Using Bcrypt Middleware to implement user password hashing and encryption
- c. Using Passport JS with Local strategy to create login session

Day 7:

- a. Sprint Demo 1 for CRUD Module
- b. Change in Login Strategy from Passport JS Local strategy to JWT Wen Token system
- c. Research on JWT wen token for user authentication and user authorization.

Day 8:

- a. Initiated implementation of JWT strategy for user authentication.
- b. Integrating Login Module with existing work framework
- c. Committing changes to existing Payment Module (CRUD) to integrate with JWT token strategy

Day 9:

- a. Completing implementation of CRUD Module with the login token system
- b. Initiating implementation of Split Payments Module
- c. Research on file/image uploads with NodeJS and MongoDB

Day 10:

- a. Completion of Split Payments module and its integration in to the existing framework
- b. Creating a proper flow for application
- c. Initiating progress report documentation

Day 11:

- a. Initiating implementation of user profile image upload option using multer middleware.
- b. Completing logic for deleting/setting of payments from the split module
- c. Unit testing of existing work done.

Day 12:

- a. Sprint Demo 2 covering new functionalities for Signup, Login, user authentication, Split payment module and image upload.
- b. Improving multer code to fix image upload functionality bug.
- c. Initiating research on Travis CI and GitHub pages.

Day 13:

- a. Conducting research and creating a .yml file for Travis CI.
- b. Restructuring file structure to comply with requirements of the deployment service provider.
- c. Initiating a launch to GitHub Pages.

Day 14:

- a. Bug fixing for .yml file.
- b. Site deployed to GitHub Pages through Travis CI, with only frontend display ability.
- c. Research on Heroku initiated.

Day 15:

- a. Site deployed directly to Heroku.
- b. Setting up MongoDB Atlas to provide a cloud-based database.
- c. Research on integrating Travis CI with Heroku for CI/CD

Day 16:

- a. Updating Tavis CI .yml file to meet Heroku requirements.
- b. Adding .Procfile for Heroku identification.
- c. Creating a pipeline providing CI/CD from GitHub to Heroku.

Day 17:

- a. Unit Testing site integration through Travis CI and Heroku
- b. Modifying initial site routed to meet Heroku requirements.
- c. Updating CORS for site usage through HTTP requests.

Day 18:

- a. Updating site frontend to be responsive for mobile use.
- b. Updating Project Report on Travis CI and Heroku useage.

Day 19:

- a. Implementing frontend user validation with restrictions of password.
- b. Integrating existing work tree with main tree and deploying to Heroku.
- c. Research on using multer for production environment instead of development environment (using AWS S3 Bucket).

Day 20:

- a. Sprint Demo 3 for SplitPay.
- b. Initiating solving file upload bug using AWS S3 Bucket.
- c. Conducting research on using Amazon Web Services for hosting and deploying a web application.

Day 21:

- a. Attempting to deploy site using AWS Lambda
- b. Conducting researching on related AWS services:
  - EBS
  - Lambda
  - EC2
  - S3
  - Code Pipeline

Day 22:

- a. Changing AWS Service to EBS
- b. Working with EC2 and EBS to create instances
- c. Implementing image upload fix

Day 23:

- a. Implementing Deploy on EC2
- b. Implementing changes to package.json file to meet AWS requirements
- c. Attempting deploy to EBS

Day 24:

- a. Implementing Deploy to EBS
- b. Committing changes to S3 bucket image upload
- c. Conducting Research in AWS Code Pipeline

Day 25:

- a. Implementing Code Pipeline to achieve CI/CD for application
- b. Conducting Unit Testing deployed site.

Day 26:

- a. Conducting research on AWS CloudFront
- b. Improving frontend design

Day 27:

- a. Trying to deploy application using CloudFront
- b. Continuing research on CloudFront and CDN
- c. Understanding terms of content delivery network

Day 28:

- a. Implementing CloudFront for images
- b. Reverting deploy to Elastic Beanstalk

Day 29:

- a. Changing image load architecture to CloudFront based system
- b. Unit testing CloudFront

Day 30:

- a. Sprint Demo 4 for SplitPay
- b. Initiating research on creating new user with policy using IAM
- c. Initiating research on Cloud Formation

Day 31:

- a. Implementing changes to implement new user and policy for images
- b. Conducting research on Cloud Formation Templates

Day 32:

- a. Continuing research on Cloud Formation
- b. Understanding Cloud Formation Stacks



Day 33:

- a. Using CloudFormation Design templates to create S3 Buckets and EC2 instances
- b. Using CloudFormation .json file templates to create S3 Buckets and EC2 instances.

Day 34:

- a. Understanding SplitPay's AWS architecture on CloudFormation design template

Day 35:

- a. Final Demo
- b. Internship Conclusion

## 6. CONCLUSION

Over the course of this internship, I have gained invaluable skills in my field of work and the experience was very useful. All the goals and objectives set out at the start of the internship were met successfully. This internship has helped me identify my strengths and weaknesses as a developer. It has provided an all-rounded development for me in web development, deployment and maintenance. I am grateful to my mentors, Mr. Abhilash Bose, Mr. Vikas Jaiswal and Mr. Vijayakanthan V, for their guidance and support throughout this internship.