VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELGAUM-590018

A Mini-Project Report

On

"BETTER BANKING"

A Mini-project report submitted in partial fulfillment of the requirements for the award of the degree of ***Bachelor of Engineering in Computer Science and Engineering*** *of Visvesvaraya Technological University, Belgaum.*

Submitted by:

Menahi Shayan	**1AM17CS101**
Nishank Swamy D N	**1AM17CS121**
Podaralla Revathi	**1AM17CS132**

Under the Guidance of:
Mr. Mushtaq Ahmed D M

(Asst. Prof., Dept. of CSE)

Department of Computer Science and Engineering
AMC Engineering College,
18th K.M, Bannerghatta Main Road, Bangalore-560 083

2020-2021

AMC Engineering College,

18th K.M, Bannerghatta Main Road, Bangalore-560 083

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the mini-project work entitled **"BETTER BANKING"** has been successfully carried out by Menahi Shayan (1AM17CS101), Nishank Swamy D N (1AM17CS121), Podaralla Revathi (1AM17CS132), the Bonafede students of **AMC Engineering College** in partial fulfilment of the requirements for the award of degree in **Bachelor of Engineering in Computer Science and Engineering** of **Visvesvaraya Technological University, Belgaum** during academic year 2020-2021. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library. The mini project report has been approved as it satisfies the academic requirements in respect of project work for the said degree.

Mr. MUSHTAQ AHMED DM	**Dr. LATHA C A**	**Dr. A.G. NATARAJ**
Asst. Prof., Dept. of CSE, AMCEC	**HOD, Dept. of CSE, AMCEC**	**Principal, AMCEC**

Examiners: **Signature with Date**

1. _____

2. _____

ACKNOWLEDGEMENT

The joy and satisfaction that accompany the successful completion of any task would be incomplete without the mention of those who made it possible. We are glad to express our gratitude towards our prestigious institution **AMC ENGINEERING COLLEGE** for providing us with utmost knowledge, encouragement and the maximum facilities in undertaking this project.

We wish to express sincere thanks to our respected chairman **Dr. K. R. Paramahamsa** and beloved principal **Dr. A.G. Nataraj** for all their support.

We express our deepest gratitude and special thanks to **Dr. Latha C.A.**, **H.O.D, Dept. Of Computer Science Engineering**, for all her guidance and encouragement.

We sincerely acknowledge the guidance and constant encouragement of our mini-project guide, **Mr. Mushtaq Ahmed D M, Assistant Prof., Dept. Of Computer Science Engineering.**

NAME	USN
Menahi Shayan	**1AM17CS101**
Nishank Swamy D N	**1AM17CS121**
Podaralla Revathi	**1AM17CS132**

ABSTRACT

The project entitled "**BETTER BANKING**" is made keeping in mind all the aspects of the Internet Banking. The system allows only authorized users to login and the new users are allowed to register on the website. The project provides most of the basic functionality required for Internet Banking. It allows the user to do various online transactions. It is be capable of doing all the necessary operations/functions that are done in any Internet Banking, adding beneficiary, login-logout, transactions, history of payments, minimalistic UI, Money Transfer, Spending Analytics, manages all your cards in a centralized manner.

The motive is to make such kind of a software that is very easy to use, minimalistic in design, with good UI/UX. All the transactions of the user will be stored for transaction analytics. Beneficiaries added so that the further transactions can be done in single click. Privacy and security of the user's data is maintained using secure authentication. All data is stored in the cloud and is secured using google security standards.

CONTENTS

Chapter 1

INTRODUCTION

This application is used by multiple authorized users who has an account in the specified Bank. The main aim is to centralize all Internet Banking at one place such as Adding Beneficiary, Login-Logout, Transactions, History of Payments, Minimalistic UI, Money Transfer, Spending Analytics.

The User can login to the login page and enter into the user profile where transactions can be made within a single click. The main aim of this project is to provide transactions on the bank with a simple and easy to use system along the transactions to merchants. It also features a tracking log that tracks all the user's transactions and tabulates them. It also creates real-time pie charts and analytics graph. It has also got a card menu, secure login via firebase to attain maximum security and even features a profile page for user customization. The software has been developed using ReactJS and Firebase.

This 'Better Banking' Project is a model Internet Banking Site. This site enables the customers to perform the basic banking transactions by sitting at their office or at homes through PC or laptop or phone. The customers can access the banks website for viewing their Account details and perform the transactions on account as per their requirements. With Internet Banking, the brick-and-mortar structure of the traditional banking gets converted into a click and portal model, thereby giving a concept of virtual banking a real shape. Thus, today's banking is no longer confined to branches. E-banking facilitates banking transactions by customers round the clock globally.

"BETTER BANKING" has been designed to computerize the following functions that are performed by the system:

- Registration details
- Security & Encryption
- Encrypted Database
- Secure Authentication
- Simple UI/UX
- Secure login
- Create Account
- Balance & Transaction History
- Cards
- Profile
- Edit Profile / change password
- Send / Receive Transactions
- Spending Analytics
- Virtual Credit Cards

1.1 OBJECTIVES

During several decades, personnel function has been transformed from a relatively obscure record keeping staff to a central and top-level management function. There are many factors that have influenced this transformation like technological advances, professionalism and general recognition of human beings as most important resources.

- Better Banking - An online banking website with a modern UX that is convenient and easy to use. Includes secure login, account creation, transactions and card support. Built on ReactJS and Firebase.

- This banking portal supports transactions to and from other users on the bank with a simple and easy to use system along with transactions to merchants.

- It also features a tracking log that tracks all the user's transactions and tabulates them. Additionally it creates real-time pie charts and analytics graphs

- It also has a card menu to manage credit cards, debit cards and virtual cards.

- It uses secure login via firebase authentication's encryption system.

- It even features a profile page for user customization

- The intention is to introduce more user-friendliness in the various activities such as transaction updating, maintenance and searching.

- The searching of transactions has been made quite simple as all the details of the users can be obtained in the user profile.

- Similarly, all the transactions updated automatically into the user profile in the transaction history page. Here the usage analytics is done via the same.

- The entire information has been maintained in the secured database.

1.2 SCOPE

PERFORMANCE: Manual handling of the record is time consuming and highly prone to error. Hence to improve the performance, computerized and user-friendly system is undertaken.

EFFICIENCY: Efficiency of the system is a basic need. So, whenever the new Transaction is done, it has to be updated automatically. It works on all platforms like Mac, Windows, Linux, iOS, Android, etc.

CONTROL: The complete control is under the user who has the authority to access, and illegal access is not permitted.

SECURITY: This is the main criteria of the proposed system. Since illegal access may cause unnecessary actions, so security has been enforced. Here, in this system, we are using 2048 bit Google SCRYPT encryption algorithm which is proprietary encryption algorithm. If any illegal login is detected then the account will be forbidden to access for specific period of time.

USER INTERFACE: The goal of user interface is to make the user's interaction as simple and efficient as possible, in terms of accomplishing user goals.

Chapter 2

SYSTEM SPECIFICATION

2.1 HARDWARE REQUIREMENTS

Minimum Req.

PROCESSOR: i3 4th gen

RAM: 4GB

HARD DISK: 40MB

Recommended Req.

PROCESSOR: i9 10^{th} gen

RAM: 32GB

HARD DISK: 4TB

2.2 SOFTWARE REQUIREMENTS

OS: Cross platform, 64 bit

CODING LANGUAGE:

- FRONT END: ReactJS

  ReactJS: React (also known as React.js or ReactJS) is an open-source, front end, JavaScript for building user interfaces or Ul components. It is maintained by Facebook and a community of individual developers and companies. React can be used as a base in the development of single-page or mobile applications. However, React is only concerned with state management and rendering that state to the DOM, so creating React applications usually requires the use of additional libraries for routing. React Router is an example of such a library.

  Node-JS:

  It is an open source, cross-platform, runtime environment that allows developers to create all kinds of server-side tools and applications in JavaScript. It uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

- BACK END: FIREBASE

  Firebase: Firebase is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help them develop quality apps, grow their user base, and earn profit. It is built on Google's infrastructure.

  Firebase is categorized as a NoSQL database program, which stores data in JSON-like documents.

Key Features:

1. Authentication

It supports authentication using passwords, phone numbers, Google, Facebook, Twitter, and more. The Firebase Authentication (SDK) can be used to manually integrate one or more sign-in methods into an app.

2. Realtime database

Data is synced across all clients in real-time and remains available even when an app goes offline.

3. Hosting

Firebase Hosting provides fast hosting for a web app; content is cached into content delivery networks worldwide.

4. Test lab

The application is tested on virtual and physical devices located in Google’s data centers.

5. Notifications

Notifications can be sent with firebase with no additional coding.

Chapter 3

DESIGN

Data Flow Diagram

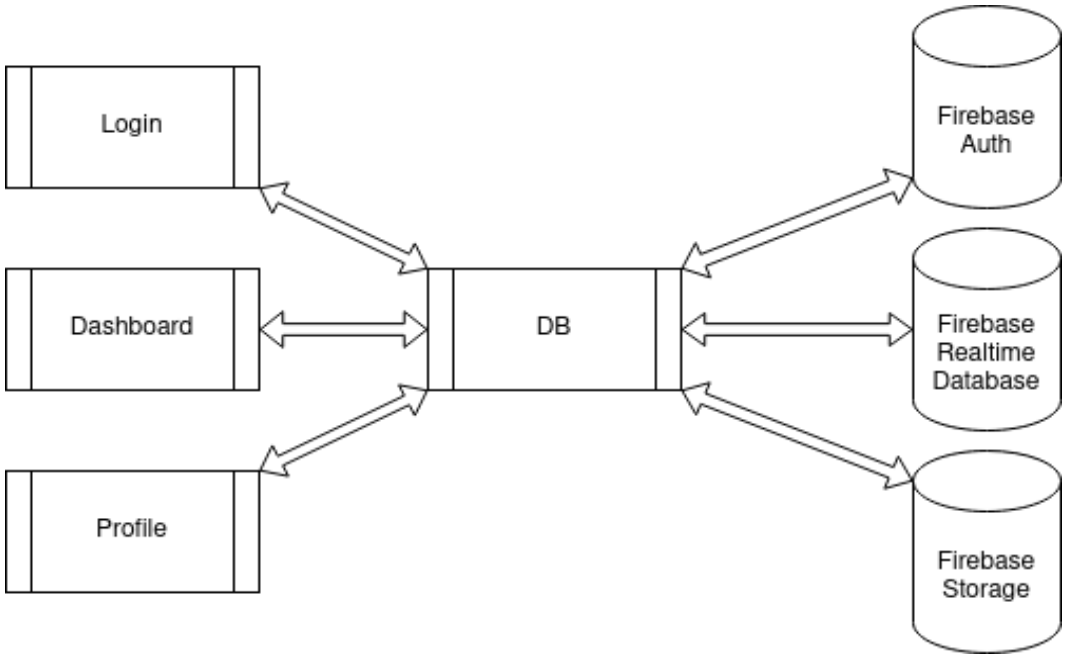


Fig (i): Data Flow Diagram

All web pages interact with the firebase database through the db class. The frontend classes are

- Login
- Dashboard
- Profile

The DB class contains API functions that allow all web pages in the app to use common functions that allow access to the database in a controlled and efficient manner.

Three different services are used from the firebase platform,

- Firebase Authentication
- Firebase Realtime Database
- Firebase Storage

Db class contains firebase configuration data and API keys and unifies and streamlines the interaction process between the frontend and the backend

Authentication:

Firebase authentication portal allows you to create and manage users. The selected sign in provider is email and password-based authentication. Each user is a allotted a randomly generated alphanumeric UID.

Identifier	Providers	Created	Signed In	User UID
cji@gmail.com		Nov 19, 2020	Nov 19, 2020	8s4PQxjtQLUlumT7jh69mCOQRZX2
ckt@gmail.com		Nov 20, 2020	Nov 20, 2020	D5p7RNYPZddv8XPI7fRpcKGo5ID3
revathipodaralla123@gmail....		Nov 12, 2020	Jan 12, 2021	GRxen20Hy2PWt00dxEj4LRk9z5V2
nishank.swamy@gmail.com		Nov 12, 2020	Jan 12, 2021	Gkv2Z8gyfINuWc5ofF8C1BVGwA...
ndc@gmail.com		Nov 20, 2020	Nov 20, 2020	J30zqYXjsqR0lDnbqUxbXwS3IMT2
ckft@gmail.com		Nov 20, 2020	Nov 20, 2020	MXcNqe7VI8Xu5DRU5qULM6xptQ...
menahi.shayan@gmail.com		Nov 12, 2020	Jan 12, 2021	P1UUbhJEx6T63ceqWw1RDGKXm...
123@gmail.com		Dec 1, 2020	Dec 1, 2020	PcrECzVrpdSxHgjsJDgLN5ZpD2B2
mnhgf@gmail.com		Nov 19, 2020	Nov 19, 2020	YRixOxwDvUQUHDevdJxAiz9Q1rB2
a@b.gmail.com		Dec 2, 2020	Dec 8, 2020	cS2osjwwaYdAZzseraoZaPpphEi1
abc@gmail.com		Nov 18, 2020	Nov 18, 2020	fiORT85xQ2T1EwUSv1zJdUxcp4Y2

Fig (ii): Firebase Authentication

If there are multiple failures in login the account will be frozen for several minutes. If the user forgets his password there is a way to recover it i.e., via reset password for this the reset password is send to the users registered email.

User
• name • email • photoUrl • emailVerified • uid • providerId • accessToken • refreshToken

Real-time database:

The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in real-time to every connected client. When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

Key capabilities:

Realtime	Instead of typical HTTP requests, the Firebase Realtime Database uses data synchronization—every time data changes, any connected device receives that update within milliseconds. Provide collaborative and immersive experiences without thinking about networking code.
Offline	Firebase apps remain responsive even when offline because the Firebase Realtime Database SDK persists your data to disk. Once connectivity is re-established, the client device receives any changes it missed, synchronizing it with the current server state.
Accessible from Client Devices	The Firebase Realtime Database can be accessed directly from a mobile device or web browser; there's no need for an application server. Security and data validation are available through the Firebase Realtime Database Security Rules, expression-based rules that are executed when data is read or written.
Scale across multiple databases	With Firebase Realtime Database on the Blaze pricing plan, you can support your app's data needs at scale by splitting your data across multiple database instances in the same Firebase project. Streamline authentication with Firebase Authentication on your project and authenticate users across your database instances. Control access to the data in each database with custom Firebase Realtime Database Rules for each database instance.

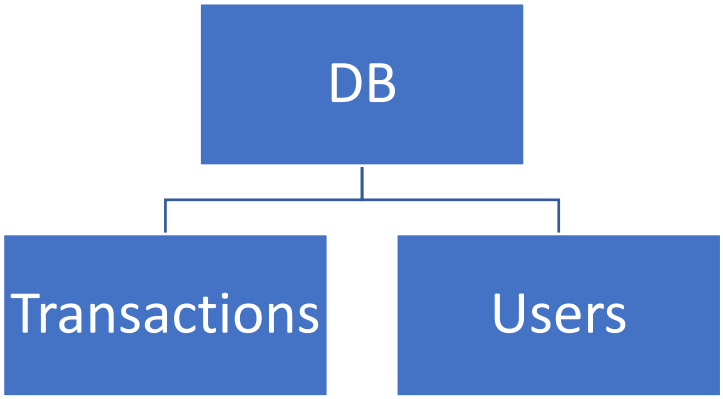


Fig (iii): Database Hierarchy

Transactions

- from
- amount
- category
- date
- description
- to
- name
- type

Users

- accno
- balance
- cards
  - cvv
  - expiry
  - name
  - number
  - provider
  - type
- history
- name
- shortname

Storage

Cloud Storage for Firebase is a powerful, simple, and cost-effective object storage service built for Google scale. The Firebase SDKs for Cloud Storage add Google security to file uploads and downloads for your Firebase apps, regardless of network quality. You can use our SDKs to store images, audio, video, or other user-generated content.

Key capabilities:

Robust operations	Firebase SDKs for Cloud Storage perform uploads and downloads regardless of network quality. Uploads and downloads are robust, meaning they restart where they stopped, saving your users time and bandwidth.
Strong security	Firebase SDKs for Cloud Storage integrate with Firebase Authentication to provide simple and intuitive authentication for developers. You can use our declarative security model to allow access based on filename, size, content type, and other metadata.
High scalability	Cloud Storage for Firebase is built for exabyte scale when your app goes viral. Effortlessly grow from prototype to production using the same infrastructure that powers Spotify and Google Photos.

Firebase storage is used to store profile picture of all the users and display when required.

Chapter 4

IMPLEMENTATION AND CODING

4.1 Source Code

FILE: App.js

```
import React from 'react';
import Dashboard from './Dashboard';
import Login from './Login';
import Profile from './Profile';
import { BrowserRouter as Router, Route } from 'react-router-
dom';

function App() {
  return (
    <Router>
      <div>
        <Route exact path="/" component={Login} />
        <Route path="/dashboard" render={(props) => <Dashboard
{...props} />} />
        <Route path="/profile" render={(props) => <Profile
{...props} />} />
      </div>
    </Router>
  );
}

export default App;
```

FILE: Component.css

```
.overlay {
    height: 100%;
    width: 100%;
    position: fixed;
    top: 0;
    left: 0;
    background-color: rgba(0, 0, 0, 0.35);
    overflow: hidden;
    z-index: 4;
    transition: all .2s ease;
    opacity: 0;
}

.overlay-container {
```

```
    position: relative;
    margin: -50% auto;
    background-color: white;
    width: 40%;
    z-index: 5;
    padding: 2% 3%;
    border-radius: 18px;
    text-align: center;
    -webkit-box-shadow: 0px 0px 13px 5px rgba(0, 0, 0, 0.17);
    box-shadow: 0px 0px 13px 5px rgba(0, 0, 0, 0.17);
}

.chart {
    margin: 5%;
    float: left;
}

.chart-tooltip {
    background-color: rgba(255, 255, 255, 0.726);
    text-align: center;
    border-radius: 15px;
    padding: 8px 20px;
}


.card {
    height: 180px;
    width: 310px;
    border-radius: 18px;
    background-color: #FAD961;
    background-image: linear-gradient(69deg, #FAD961 0%,
#F76B1C 100%);
    margin: 8% 5%;
    border-width: 0;
    padding: 8%;
    -webkit-box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.2);
    box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.2);
}

.card:hover {
    -webkit-box-shadow: 0px 0px 15px 0px rgba(0, 0, 0, 0.3);
    box-shadow: 0px 0px 15px 0px rgba(0, 0, 0, 0.3);
}

.card-no {
    color: white;
    position: relative;
    top: 60%;
    letter-spacing: 2px;
}
```

```
.card-type {
    position: absolute;
    top:10%;
    right: 8%;
}


.person {
    display: inline-block;
    margin: 0 2%;
    text-align: center;
}

.person-pic {
    height: 60px;
    width: 60px;
    border-radius: 50%;
    background-color: white;
    border: 3px solid rgb(48, 190, 233);
    margin-left: auto;
    margin-right: auto;
}

.person-name {
    user-select: none;
}
```

FILE: Component.js

```
import React, { Fragment } from 'react';
import { PieChart, Pie, Cell, Tooltip } from 'recharts';
import './Components.css'

export const COLORS = ['#0088FE', '#00C49F', '#FFBB28',
'#FF8042'];


export const Overlay = props => (
    <Fragment>
        <div className="overlay" onClick={props.bgClick}
style={{ opacity: (props.visible === true ? 1 : 0), display:
(props.visible === true ? 'block' : 'none') }}></div>
        <div className="overlay-container" style={{ display:
(props.visible === true ? 'block' : 'none') }}>
            {props.children}
        </div>
    </Fragment>
)

export const Chart = props => {
```

```
    const chartLabel = ({ cx, cy, midAngle, innerRadius,
outerRadius, percent, index }) => {
        const radius = innerRadius + (outerRadius -
innerRadius) * 0.25;
        const x = cx + radius * Math.cos(-midAngle * (Math.PI
/ 180));
        const y = cy + radius * Math.sin(-midAngle * (Math.PI
/ 180));

        return (
            <text x={x} y={y} fill="white" textAnchor={x > cx
? 'start' : 'end'} dominantBaseline="central">
                {props.chartData[index].name}
            </text>
        );
    };

    const chartTooltip = ({ active, payload, label }) => {
        if (active) {
            return (
                <div className="chart-tooltip">
                    <b>{`${payload[0].name}`}</b><br />
                    {` \u20B9${payload[0].value.toFixed(2)}`}
                </div>
            );
        }

        return null;
    };
    return (
        <PieChart width={300} height={300} className="chart">
            <Pie data={props.chartData} labelLine={false}
label={chartLabel} outerRadius={150} innerRadius={110}
fill="#8884d8" dataKey="value" >
                {props.chartData.map((entry, index) => <Cell
key={`cell-${index}`} fill={COLORS[index % COLORS.length]}
/>)}
            </Pie>
            <Tooltip content={chartTooltip} />
            {/* <Legend /> */}
        </PieChart>
    )
}

export const Card = props => {
    const maskedCardNo = (number) => {
        return number.substring(0, 4) + number.substring(4,
number.length - 2).replace(/\d/g, "\u2022") +
number.substring(number.length - 2);
    }
    return (
```

```
        <div className="card zoom-l">
            <div className="card-type">
                {
                    props.card.provider === 'mastercard' ?
                        <img src="mc-icon.svg" alt="mc"
height="24px" />
                        :
                        <img src="visa-icon.svg" alt="visa"
height="18px" />
                }
            </div>
            <div className="card-
no">{maskedCardNo(props.card.number)}</div>
        </div>
    )
}

export const PersonAvatar = props => (
    <div style={{ display: 'inline', position: props.inline ?
'relative' : 'static', left: '-3%' }}>
        <div className="person zoom-m" onClick={props.onClick}
style={[{ display: props.inline ? 'inline-block' : 'block' },
props.style]}>
            {
                props.person.img ?
                    <div>
                        <img src={props.person.img}
alt={props.person.shortname} className="person-pic" />
                    </div> :
                    <div className="person-pic"></div>
            }
            {!props.inline && <span className="person-
name">{props.person.shortname}</span>}
        </div>
        {
            props.inline &&
            <div style={{ display: 'inline-block', textAlign:
'left', marginLeft: '2%', position: 'relative', top: 12 }}>
                <span style={{ fontSize: 22, fontWeight: 600,
position: 'relative', top: 8 }}>{props.person.name}</span>
                <small style={{ position: 'relative', color:
'darkgrey' }}><br />{props.person.to}</small>
            </div>
        }
    </div>
)
```

FILE: Config.js

```
const config = {
```

```
    apiKey: "AIzaSyCr24EfY81mpT0MJx7tvkAzQUvCJPiJhD4",
    authDomain: "better-banking-c614b.firebaseapp.com",
    databaseURL: "https://better-banking-
c614b.firebaseio.com",
    projectId: "better-banking-c614b",
    storageBucket: "better-banking-c614b.appspot.com",
    messagingSenderId: "301515208758",
    appId: "1:301515208758:web:85b291fed0706a8f8cc097"
};

export default config;
```

FILE: Dashboard.css

```
.back-button {
    color: white;
    font-size: 26px;
    margin: 2%;
    float: left;
}

.dashboard-back {
    background-color: #0093E9;
    background-image: linear-gradient(135deg, #003feb 0%,
#80b9d0 100%);
}

.dashboard-main {
    padding: 3%;
    color: white;
}

.name {
    color: white;
    padding-top: 2.3%;
    float: right;
    font-size: 18px;
    font-weight: bold;
}

.profile-button {
    height: 35px;
    width: 35px;
    border-radius: 50%;
    background-color: white;
    float: right;
    display: block;
    margin: 2% 2.5%;
    margin-left: 1%;
}
```

```
.amount::before {
    content: '\20B9';
}

.neg-amount {
    color: #9c0808;
}

.neg-amount::before {
    content: '-\20B9';
}

.pos-amount {
    color: #0d5c09;
}

.pos-amount::before {
    content: '+\20B9';
}

.balance {
    font-size: 48px;
    margin: 4%;
    font-weight: 600;
}

.recents {
    width: 60%;
    float: right;
}

.new-pay-button {
    height: 63px;
    width: 63px;
    border-radius: 50%;
    background-color: rgba(255, 255, 255, 0.493);
    padding: 4px 0;
    text-align: center;
    display: block;
    font-size: 34px;
    font-weight: bold;
    display: inline-block;
    position: relative;
    top: -35px;
    margin-left: 2%;
}

.dashboard-subcontent {
    border-radius: 30px 30px 0 0;
    background-color: white;
```

```
    position: relative;
    padding: 3%;
    -webkit-box-shadow: 0px -2px 13px 5px rgba(0, 0, 0, 0.27);
    box-shadow: 0px -2px 13px 5px rgba(0, 0, 0, 0.27);
    color: #444;
    padding-bottom: 15%;
    display: inline-flex;
    width: 100%;
}

.card-section {
    border-left: 1px solid lightgray;
    padding-left: 4%;
    margin-left: -8%;
}

.history-list {
    overflow-y: scroll;
    overflow-x: hidden;
    height: 400px;
    width: 45%;
    display: inline-block;
}

.history {
    border-bottom: 1px solid lightgray;
    width: 100%;
    padding: 8px;
    padding-top: 15px;
}

.history-icon {
    height: 40px;
    width: 40px;
    border-radius: 50%;
    background-color: darkgray;
    float: left;
    display: block;
    margin: 0 2.5%;
    padding: 8px 0;
    text-align: center;
    color: white;
}

.pay-category-icon {
    height: 50px;
    width: 50px;
    border-radius: 50%;
    background-color: rgb(226, 225, 225);
    display: inline-block;
    margin: 0 1.5%;
```

```
    padding: 10px 0;
    text-align: center;
    color: grey;
    font-size: larger;
}

.selected-pay-category {
    background-color:#0088FE;
    color: white;
}

.pay-amount {
    font-weight: bold;
    font-size: 30px;
    position: relative;
    text-align: center;
}

.pay-amount::before {
    content: '\20B9';
}

.history-name {
    font-weight: bold;
    color: #444;
    display: block;
}

.history-description {
    color: darkgray;
    font-size: small;
    position: relative;
    top: -7px;
}

.history-amount {
    float: right;
    font-weight: bold;
    font-size: 22px;
    position: relative;
    top: -20px;
}

.zoom-m,
.zoom-l {
    transition: all .4s;
    cursor: pointer;
}

.zoom-m:hover {
    transform: scale(1.1);
```

```
}

.zoom-l:hover {
    transform: scale(1.05);
}
```

FILE: Dashboard.js

```
import './Dashboard.css';
import React, { useState, useEffect, Fragment } from 'react';
import { useForm } from "react-hook-form";
import Form from 'react-bootstrap/Form';
import Button from 'react-bootstrap/Button'
import Spinner from 'react-bootstrap/Spinner'
import { Redirect } from 'react-router';
import { FontAwesomeIcon } from '@fortawesome/react-
fontawesome'
import { faArrowLeft, faUtensils, faPlane, faShoppingBag,
faReceipt, faFilm, faHandHoldingMedical, faCubes } from
'@fortawesome/free-solid-svg-icons'
import DB from './DB'
import { Overlay, Chart, Card, PersonAvatar, COLORS } from
'./Components'

const db = new DB()

function Dashboard(props) {
    const [redirect, setRedirect] = useState();
    const [chartData, setChartData] = useState([]);
    const [recentPersons, setRecentPersons] = useState([]);
    const [profilePic, setProfilePic] = useState();
    const [payOverlay, setPayOverlay] = useState();
    const [newPayment, setNewPayment] = useState();
    const [selectedCategory, setSelectedCategory] =
useState();
    const [error, setError] = useState();
    const [loading, setLoading] = useState(false);
    const [history, setHistory] = useState(false);
    const { register, handleSubmit } = useForm()

    var user = props.location.state

    useEffect(() => {
        db.getProfilePic(user.accno, (url) => {
            setProfilePic(url)
        })
        const getChartData = (transactions) => {
            let items = {}, arrayItems = []

            transactions.forEach(his => {
```

```
                if (items[his.category])
items[his.category].value += his.amount
                else items[his.category] = { name:
his.category[0].toUpperCase() + his.category.slice(1), value:
his.amount }
            })

            for (let item in items) {
                arrayItems.push(items[item])
            }

            setChartData(arrayItems);
        }
        const getRecentPersons = (transactions) => {
            var recentPersonsSet = {}
            setRecentPersons([])
            transactions.forEach(his => {
                if (his.type === 'person') {
                    db.getUser(his.to !== user.accno ? his.to
: his.from, (u) =>
                        db.getProfilePic(his.to !== user.accno
? his.to : his.from, (url) => {
                            recentPersonsSet[u.accno] = { to:
u.accno, name: u.name, shortname: u.shortname, img: url }
                            let tempArray = []
                            for (let rps in recentPersonsSet)
{

tempArray.push(recentPersonsSet[rps])
                            }
                            setRecentPersons(tempArray)
                        })
                    )
                }
            })
        }
        db.getTransactions(user.accno, user.history,
transactions => {
            setHistory(transactions)
            getChartData(transactions)
            getRecentPersons(transactions)
        })
        // eslint-disable-next-line react-hooks/exhaustive-
deps
    }, [user.accno, user.history])


    const accnoCheck = (e) => {
        let accno = e.target.value
        if(!accno) setError('Please enter an account number')
```

```
        else if (accno.length < 13) setError('Please enter a
valid account number')
        else setError('')
    }

    const amountCheck = (e) => {
        let amount = e.target.value
        if(!amount) setError('Please enter a valid amount')
        else setError('')
    }

    const payHandler = (d) => {
            db.payHandler(user.accno, newPayment.person ?
newPayment.person.to : d.accno, d.amount, d.description, new
Date().toJSON().slice(0,10),
getCategoryName(selectedCategory))
    }

    const getCategoryIcon = (category) => {
        switch (category) {
            case 'food': return <FontAwesomeIcon
icon={faUtensils} />
            case 'travel': return <FontAwesomeIcon
icon={faPlane} />
            case 'shopping': return <FontAwesomeIcon
icon={faShoppingBag} />
            case 'medical': return <FontAwesomeIcon
icon={faHandHoldingMedical} />
            case 'entertainment': return <FontAwesomeIcon
icon={faFilm} />
            case 'bills': return <FontAwesomeIcon
icon={faReceipt} />
            default: return <FontAwesomeIcon icon={faCubes} />
        }
    }

    const getCategoryName = (category) => {
        switch (category) {
            case faUtensils: return 'food'
            case faPlane: return 'travel'
            case faShoppingBag: return 'shopping'
            case faHandHoldingMedical: return 'medical'
            case faFilm: return 'entertainment'
            case faReceipt: return 'bills'
            default: return 'other'
        }
    }

    if (redirect) return <Redirect push to={{ pathname:
redirect, state: user }} />
    return (
```

```
        <div className="dashboard-back">
            <FontAwesomeIcon icon={faArrowLeft}
className="back-button zoom-m" onClick={() => db.logout(() =>
setRedirect('/'))} />
            <img src={profilePic} className="profile-button
zoom-m" onClick={() => setRedirect('/profile')} alt="" />
            <div className="name">{user.shortname}</div>
            <br /><br />
            <div className="dashboard-main">
                <span className="balance
amount">{user.balance.toFixed(2)}</span>
                <div className="recents">
                    {
                        recentPersons.length > 1 &&
recentPersons.map((person, p) =>
                            <Fragment key={p}><PersonAvatar
person={person} onClick={() => { setPayOverlay(true);
setNewPayment({ person }); setSelectedCategory(faUtensils); }}
/></Fragment>
                        )
                    }
                    <div className="new-pay-button zoom-m"
onClick={() => { setPayOverlay(true); setNewPayment();
setSelectedCategory(faUtensils); }}>+</div>
                </div>
            </div>
            <div className="dashboard-subcontent">
                <div style={{ width: '78%' }}>
                    <h1>History</h1>
                    <Chart chartData={chartData} />
                    <div className="history-list">
                        { //add txn id & to and from tracking
on both sides
                            history && history.map((item, i)
=> (
                                <div className="history"
key={i}>
                                    <div className="history-
icon" style={{ backgroundColor: COLORS[i % COLORS.length]
}}>{getCategoryIcon(item.category)}</div>
                                    <span className="history-
name">{item.name || item.to}</span>
                                    <span className="history-
description">{item.description}</span>
                                    {
                                        item.to === user.accno
?
                                            <span
className="history-amount pos-amount">{item.amount}</span> :
<span className="history-amount neg-
amount">{item.amount}</span>
```

```
                                    }
                                </div>
                            ))
                        }
                    </div>
                </div>
                <div className="card-section">
                    <h1>Cards</h1>
                    {user.cards &&
                        <div>
                            <Card card={user.cards[0]} />
                        </div>
                    }
                </div>

            </div>
            { payOverlay &&
                <Overlay visible={payOverlay} bgClick={() =>
setPayOverlay(!payOverlay)}>
                    <Form onSubmit={handleSubmit(payHandler)}>
                        {newPayment ?
                            <Fragment>
                                <PersonAvatar
person={newPayment.person} inline />
                                <br />
                                <br />
                            </Fragment>
                            :
                            <Fragment>
                                <br />
                                <br />
                                <Form.Control type="text"
name='accno' placeholder='Account Number' className="field"
style={{ textAlign: 'center' }} ref={register({ required:
false })} onBlur={accnoCheck} />
                            </Fragment>
                        }

                        <Form.Control type="number"
name='amount' placeholder='0' className="field pay-amount"
ref={register({ required: true })} onBlur={amountCheck} />
                        <Form.Control type="text"
name='description' placeholder='Message' className="field"
style={{ textAlign: 'center' }} ref={register({ required: true
})} />

                        {
                            [faUtensils, faPlane,
faShoppingBag, faHandHoldingMedical, faFilm, faReceipt,
faCubes].map((item, i) =>
```

```
                                    <div key={i} className={`pay-
category-icon ${selectedCategory === item ? "selected-pay-
category" : ""}`} onClick={() =>
setSelectedCategory(item)}><FontAwesomeIcon icon={item}
/></div>
                                )
                            }

                            <br /><br /><br />
                            <Button type="submit"
className='submit'>Pay</Button>
                            <br /><br />
                            {error ? <p className="login-
alert">{error}</p> : <br />}
                        </Form>
                    </Overlay>
                }
            </div>
        );
}

export default Dashboard;
```

FILE: DB.js

```
import firebase from 'firebase/app';
import 'firebase/database';
import 'firebase/storage';
import 'firebase/auth';
import config from './config';
import React from 'react';

class DB extends React.Component {
     constructor(props) {
          super(props);
          if (!firebase.apps.length) {
               firebase.initializeApp(config);
          }
     }
     getUser = (accno, callback) => {
          let ref = firebase.database().ref('/users/' +
accno);
          ref.on('value', async (snapshot) => {
               // var user = firebase.auth().currentUser;
               // var data = await snapshot.val()
               // user.updateProfile({
               //   displayName: data.accno
               // }).then(function () {
               //   console.log("success");
               // }).catch(function (error) {
```

```
                //   console.log(error);
                // });
                console.log("Get User");
                callback && callback(await snapshot.val())
           })
     }
     getProfilePic = (accno, callback) => {
           let ref = firebase.storage().ref('dp/' + accno +
'.jpg');
           ref.getDownloadURL().then(callback).catch(err =>
console.log(err.code))
     }

     login = (email, pass, errorCallback) => {
           firebase.auth().signInWithEmailAndPassword(email,
pass).catch(errorCallback);
     }

     loginFetch = (callback) => {
           firebase.auth().onAuthStateChanged((user) => {
                if (user) {
                     this.getUser(user.displayName, callback)
                } else {
                     callback()
                }
           });
     }

     createUser = (email, password, callback) => {

     firebase.auth().createUserWithEmailAndPassword(email,
password).then((result) => {
                console.log(result);

                const numbers = '1234567890'
                let userID = ''
                for (let i = 0; i < 13; ++i)
                     userID +=
numbers.charAt(Math.floor(Math.random() * numbers.length))

                console.log(userID);

                let userProfile = {
                     accno: userID,
                     balance: 0,
                     email: email,
                     name: "",
                     shortname: "",
                     cards: [],
                     history: []
                }
```

```
                firebase.auth().onAuthStateChanged((user) => {
                     if (user) {
                          user.updateProfile({
                               displayName: userProfile.accno
                          }).then(function () {
                               console.log("success");
                               let ref =
firebase.database().ref('/users/' + userProfile.accno);
                               ref.set(userProfile)
                               callback(user)
                          }).catch(err => console.log(err));
                     }
                });

           }).catch((error) => {
                console.log("Error");
                console.log(error.message);
           });
     }

     getTransactions = (accno, ids, callback) => {
          let ref = firebase.database().ref('/transactions');
          ref.on('value', async (snapshot) => {
               let list = []
               let data  = await snapshot.val();

               for (let d in data)
                    if(ids.includes(d)) {
                         if(data[d].to === accno) {
                              this.getUser(data[d].from, user
=> {
                                   data[d].name = user.name
                              })
                         }
                         list.push(data[d])
                    }

               callback && callback(list)
          })
     }

     edit = (accno, d, callback) => {
          console.log(accno);
          this.getUser(accno, (data) => {
               console.log(data);
               var newData = {
                    accno: data.accno,
                    balance: data.balance,
                    history: [],
                    name: d.name || data.name,
```

```
                    shortname: d.shortname || data.shortname,
                    upi: d.upi || data.upi,
                    dob: d.dob || data.dob,
                    phone: d.phone || data.phone,
                    cards: []
               }
               firebase.database().ref('/users/' +
accno).update(newData, callback(newData))
          })
     }

     updatePass = (callback) => {
          firebase.auth().onAuthStateChanged((user) => {
               if (user) {
                    var newPassword;

     user.updatePassword(newPassword).then(function () {
                         console.log("success");
                    }).catch(function (error) { });
               } else {
                    callback()
               }
          });
     }

     payHandler = (from, to, amount, description, date,
category) => {
          console.log({from, to, amount, description, date,
category});
     }

     logout = (callback) => {
          firebase.auth().signOut().then(callback).catch(error
=> console.log(error));
     }
}
export default DB;
```

FILE: Login.css

```
.login-bg{
     background-color: #0093E9;
     background-image: linear-gradient(135deg, #003feb 0%,
#80b9d0 100%);
     height: 100%;
     width: 100%;
     text-align: center;
}

.title {
```

```
     color: white;
     font-weight: bolder;
     padding: 5% 0;
}
.login-container{
     text-align: center;
     margin: auto;
     width:30%;
     background-color: white;
     color: black;
     border-radius: 30px;
     padding: 3% 5%;
     margin-bottom: 20%;
     -webkit-box-shadow: 0px 0px 13px 5px rgba(0, 0, 0, 0.17);
    box-shadow: 0px 0px 13px 5px rgba(0, 0, 0, 0.17);
}
.field{
     margin: 3% auto;
     border-radius: 10px;
}
.submitLogin{
     border-radius: 10px;
     background-color: #003feb;
     padding: 2%;
     width: 100%;
     color: white;
     font-weight:600;
     font-size: 18px;
}

.login-alert {
     color: rgb(212, 5, 33);
     font-size: small;
     font-weight: 600;
}
```

FILE: Login.js

```
import React, { useState } from 'react';
import { Redirect } from 'react-router';
import { useForm } from "react-hook-form";
import Form from 'react-bootstrap/Form';
import Button from 'react-bootstrap/Button';
import Spinner from 'react-bootstrap/Spinner';
import './Login.css';
import DB from './DB';

const db = new DB()

function Login() {
```

```
    const [redirect, setRedirect] = useState();
    const [user, setUser] = useState();
    const [error, setError] = useState();
    const [signUp, setsignUp] = useState(false);
    const [login, setlogin] = useState("Login");
    const [loading, setLoading] = useState(false);
    const { register, handleSubmit, getValues } = useForm()

    const loginHandler = (d) => {
        setLoading(true)
        if (signUp) {
            db.createUser(d.email,d.password,(user) => {
                console.log(user);
            })
        }
        else {
            db.login(d.email, d.password, (error) => {
                setError(error.message)
                setLoading(false)
                console.log(error.code);
            })
        }
        db.loginFetch((user) => {
            if (user) {
                setUser(user);
                setRedirect('/dashboard')
            } else {
                setLoading(false)
            }
        })
    }

    const loginCheck = () => {
        var email=getValues('email')
        db.login(email,'0', (error) => {
            setLoading(false)
            if(error.code==='auth/user-not-found'){
                setsignUp(true)
                setlogin("Sign Up")
            } else {
                setsignUp(false)
                setlogin("Login")
            }
        })
    }

    if (redirect) return <Redirect push to={{ pathname:
redirect, state: user }} />
    return (
        <div className="login-bg">
            <h1 className="title">Better Banking</h1>
```

```
            <Form className='login-container'
onSubmit={handleSubmit(loginHandler)}>
                <h2>{login}</h2>
                <br />
                <Form.Control type="email" name='email'
placeholder='Email' className="textfield field"
ref={register({ required: true })} onBlur={() => loginCheck()
} />
                <Form.Control type="password" name='password'
placeholder='Password' className="textfield field"
ref={register({ required: true })} />
                {
                    signUp && <div>
                        <Form.Control type="password"
name='confirm_password' placeholder='Confirm Password'
className="textfield field" ref={register({ required: true })}
/>
                    </div>
                }
                <Button type="submit" className='submitLogin'>
                    {loading ? <Spinner
                        as="span"
                        animation="border"
                        role="status"
                        size="sm"
                    />: login}
                </Button>
                <br /><br />
                {error ? <p className="login-
alert">{error}</p> : <br />}
            </Form>
        </div>
    );
}

export default Login;
```

FILE: Profile.css

```
.profile-back {
    padding: 7%;
    color: white;
    background-color: #0093E9;
    background-image: linear-gradient(135deg, #003feb 0%,
#80b9d0 100%);
}
.efield{
    margin: 0 4% 0 2%;
    padding: 2%;
    border-radius: 10px;
```

```
    width: 35%;
}
.profile {
    display: block;
    margin: auto;
    margin-top: -8%;
    text-align: center;
}
.profile-pic {
    height: 120px;
    width: 120px;
    border-radius: 50%;
    background-color: white;
    border: 3px solid rgb(48, 190, 233);
    /* display: block; */
    margin: auto;
}
.profile-subcontent {
    border-radius: 30px 30px 0 0;
    background-color: white;
    position: relative;
    padding: 3% 3% 3% 7%;
    -webkit-box-shadow: 0px -2px 13px 5px rgba(0, 0, 0, 0.27);
    box-shadow: 0px -2px 13px 5px rgba(0, 0, 0, 0.27);
    color: #444;
    padding-bottom: 15%;
    display: block;
    width: 100%;
    margin-top: -2%;
    font-size: 18px;
}
.info{
    padding: 0 0 2% 0%;
}
.edit{
    height: 63px;
    width: 63px;
    border-radius: 50%;
    background-color: #97a19a;
    padding: 12px 0;
    text-align: center;
    display: block;
    font-size: 27px;
    font-weight: bold;
    /* display: inline-block; */
    color: white;
    position: relative;
    float: right;
    margin: -6% 0;
    /* margin-left: 2%; */
}
```

```
.change{
    border-radius: 18px;
    background-color: #003feb;
    padding: 1%;
    width: 15%;
    color: white;
    font-weight:600;
    font-size: 16px;
}
.submitProfile{
    border-radius: 10px;
    background-color: #003feb;
    /* padding: 1%; */
    width: 100%;
    margin: auto;
    color: white;
    font-weight:600;
    font-size: 18px;
}
```

FILE: Profile.js

```
import React, { useState, useEffect } from 'react';
import { Redirect, useHistory } from 'react-router';
import { FontAwesomeIcon } from '@fortawesome/react-
fontawesome'
import { useForm } from "react-hook-form";
import { faArrowLeft, faPencilAlt } from '@fortawesome/free-
solid-svg-icons'
import Button from 'react-bootstrap/Button'
import Form from 'react-bootstrap/Form';
import './Profile.css';
import DB from './DB'
import { Overlay } from './Components'

const db = new DB()

function Profile(props) {
    const [user, setUser] = useState(props.location.state);
    const [redirect, setRedirect] = useState();
    const [profilePic, setProfilePic] = useState();
    const [showEdit, setshowEdit] = useState(false);
    // const [showDetails, setshowDetails] = useState(true);
    const [changePassword, setchangePassword] =
useState(false);
    const { register, handleSubmit } = useForm()
    const [error, setError] = useState();
    const history = useHistory()

    useEffect(() => {
```

```
        console.log("USEEFFECT");

        db.getProfilePic(user.accno, (url) => {
            setProfilePic(url)
        })
    }, [])
    const passwordHandler = (d) => {
        console.log("PASSWORD HANDLER");
        db.updatePass(d)
    }

    const editdHandler = (d) => {
        console.log("EDIT HANDLER");
        db.edit(user.accno, d, (u) => {
                console.log(u);
                setUser(u)
                setshowEdit(false)
        })
        // setRedirect('/profile')

        // user = props.location.state;
        // console.log(user);

        // history.go(0)
    }

    if (redirect) return <Redirect push to={{ pathname:
redirect, state: user }} />

    return (
        <div>
            <FontAwesomeIcon icon={faArrowLeft}
className="back-button zoom-m" onClick={() =>
setRedirect('/dashboard')} />
            <div className='profile-back'>  </div>
            <div className='profile-subcontent'>
                {
                    !showEdit ? <div>
                        <div className="profile zoom-m">
                            <div ><img src={profilePic}
className="profile-pic" alt="" /></div>
                            <span className="person-
name">{user.name}</span>
                        </div><br />
                        <div className='edit zoom-m'
onClick={() => (setshowEdit(true))}><FontAwesomeIcon
icon={faPencilAlt} /></div>
                        <div className='info'><b>Short Name
:</b> {user.shortname}</div>
                        <div className='info'><b>Account
Number:</b> {user.accno}</div>
```

```
                        <div className='info'><b>Date of Birth
:</b> {user.dob}</div>
                        <div className='info'><b>Phone :</b>
{user.phone}</div>
                        <div className='info'><b>Email :</b>
{user.email}</div>
                        <div className='info'><b>UPI ID :</b>
{user.upi}</div>
                        <Button className='change' onClick={()
=> setchangePassword(true)}>Change Password</Button>
                        {
                            changePassword &&
                            <Overlay visible={changePassword}
bgClick={() => setchangePassword(!changePassword)} height={40}
width={50}>
                                <div style={{ display:
'inline-block', width: '100%', overflow: 'scroll', marginTop:
'-2%' }}>
                                    <Form
onSubmit={handleSubmit(passwordHandler)} autocomplete="off">
                                        <Form.Control
type="password" name='oldPassword' placeholder='Old Password'
className="textfield field" ref={register({ required: true })}
autocomplete="off" />
                                        <Form.Control
type="password" name='newPassword' placeholder='New Password'
className="textfield field" ref={register({ required: true })}
/>
                                        <Form.Control
type="password" name='confirmNewPassword' placeholder='Confirm
New Password' className="textfield field" ref={register({
required: true })} />
                                        <Button
className='submitProfile' type='submit'>Submit</Button>
                                    </Form>
                                </div>
                            </Overlay>
                        }
                    </div>:
                    <div visible={showEdit}>
                        <div className="profile zoom-m">
                            <div ><img src={profilePic}
className="profile-pic" alt="" /></div>
                            <span className="person-
name">{user.name}</span>
                        </div>
                        <FontAwesomeIcon icon={faArrowLeft}
style={{ color: 'black' }} className="back-button zoom-m"
onClick={() => (setshowEdit(false))} />
                        <Form
onSubmit={handleSubmit(editdHandler)} autocomplete="off" >
```

```
                                <Form.Group style={{ display:
'inline-flex', width: '100%', margin: '0 0% 1% 4%' }}>
                                    Name :<Form.Control
type="text" name='name' placeholder={user.name}
className="textfield efield" ref={register({ required: false
})} />
                                    Short Name :<Form.Control
type="text" name='shortname' placeholder={user.shortname}
className="textfield efield" ref={register({ required: false
})} style={{ marginLeft: '3.5%' }} />
                                </Form.Group>
                                <Form.Group style={{ display:
'inline-flex', width: '100%', margin: '0 0% 1% 4%' }}>
                                    UPI ID :<Form.Control
type="text" name='upi' placeholder={user.upi}
className="textfield efield" ref={register({ required: false
})} />
                                    Date of Birth :<Form.Control
type="text" name='dob' placeholder={user.dob}
className="textfield efield" ref={register({ required: false
})} style={{ marginLeft: '2.5%' }} />
                                </Form.Group>
                                <Form.Group style={{ display:
'inline-flex', width: '100%', margin: '0 0% 1% 4%' }}>
                                    Phone :<Form.Control
type="text" name='phone' placeholder={user.phone}
className="textfield efield" ref={register({ required: false
})} />
                                    {/* Account Num :<Form.Control
type="text" name='oldPassword' placeholder={user.accno}
className="textfield efield" ref={register({ required: false
})} /> */}
                                </Form.Group>
                                <Button className='submitProfile'
type='submit' style={{ width: '30%' }}>Submit</Button>
                            </Form>
                        </div>
                    }
                </div>
            </div>
        );
}
export default Profile;
{/* <div>
                    {
                        Object.keys(user).map((item, i) => (
                            <div>
                                {user[item].toString()}
                            </div>
                        ))
                    }
```

Chapter 5

SCREENSHOTS

Fig (iv): Signup

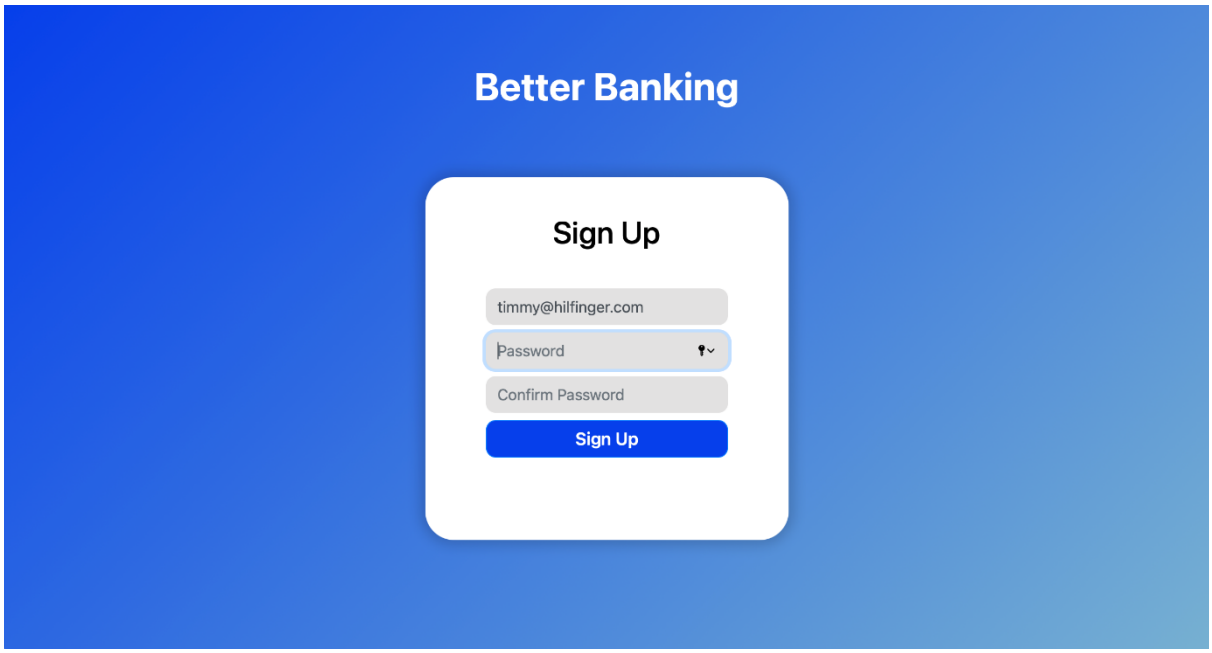


Fig (v): Login

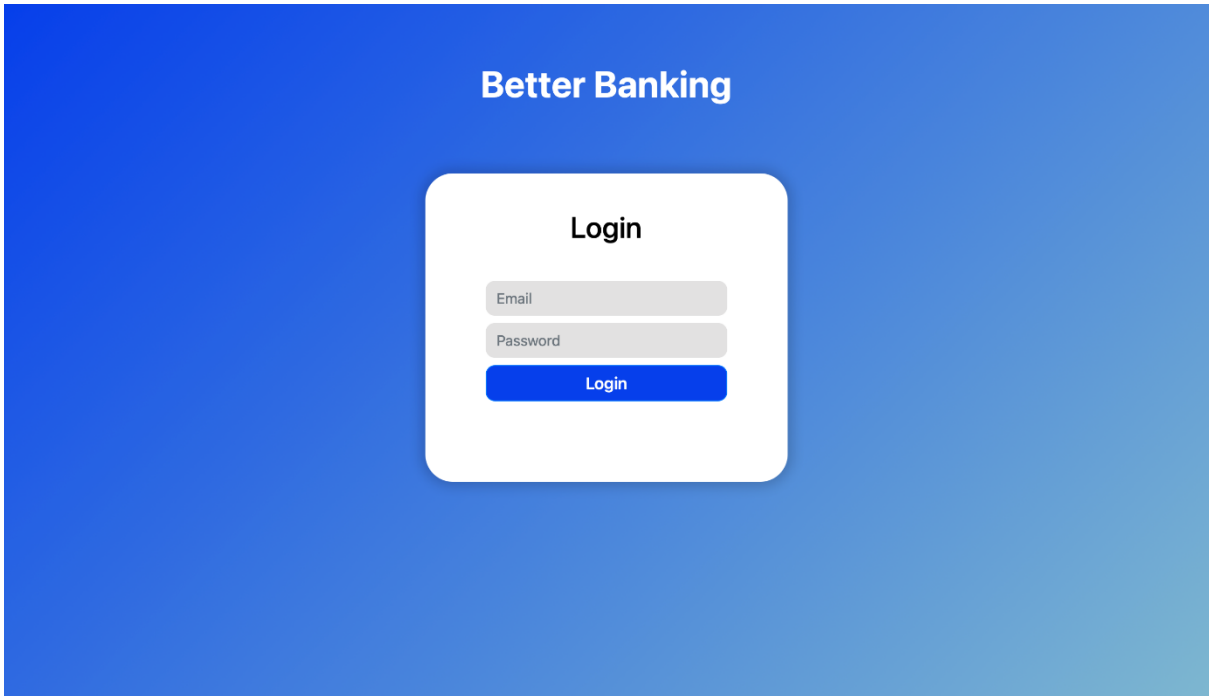


Fig (vi): Unauthorized Login

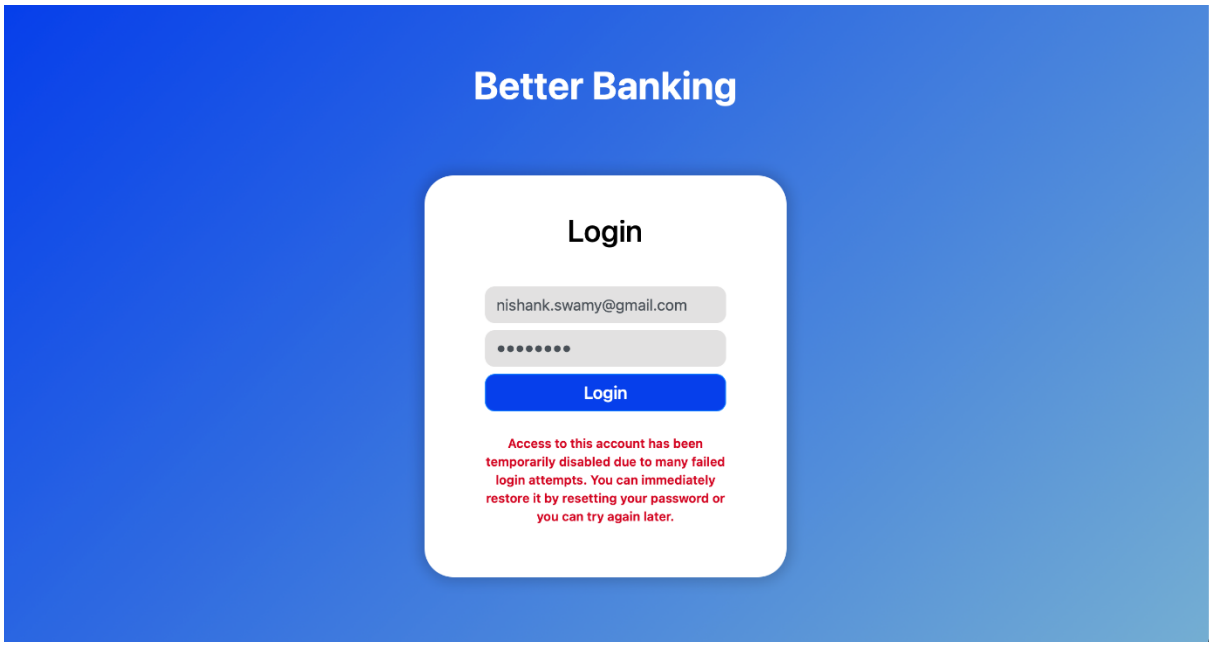


Fig (vii): Change Password

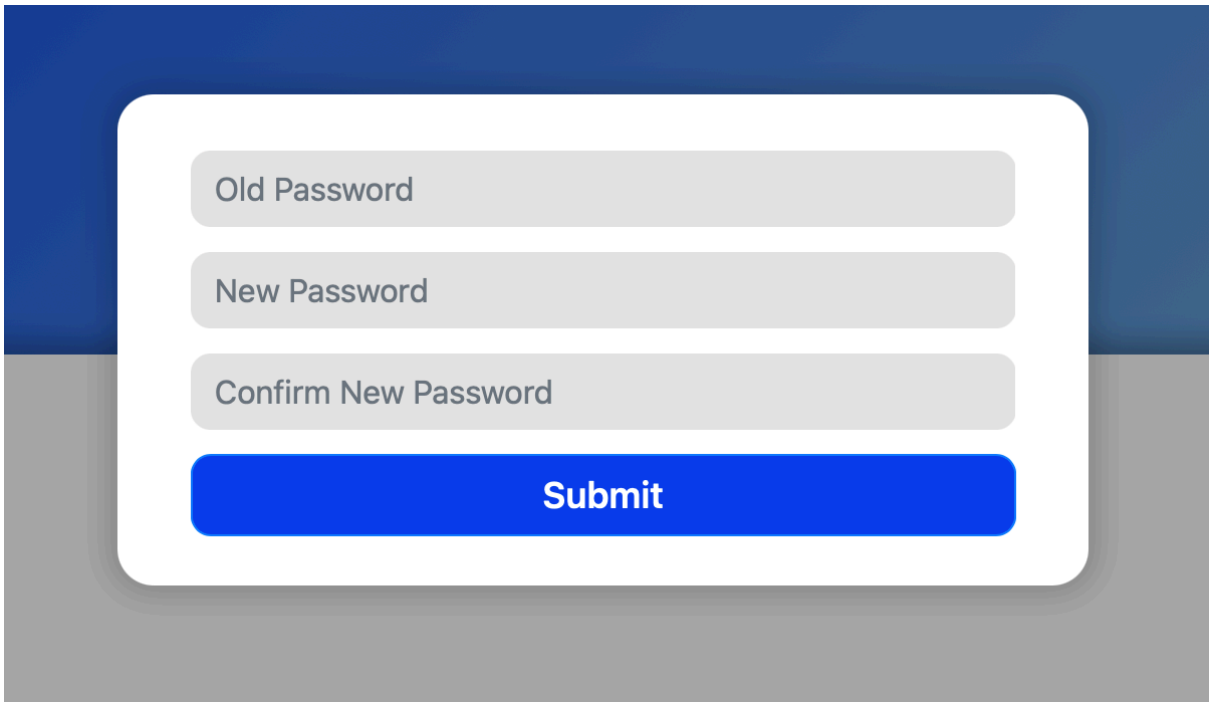


Fig (viii): Profile Home Page

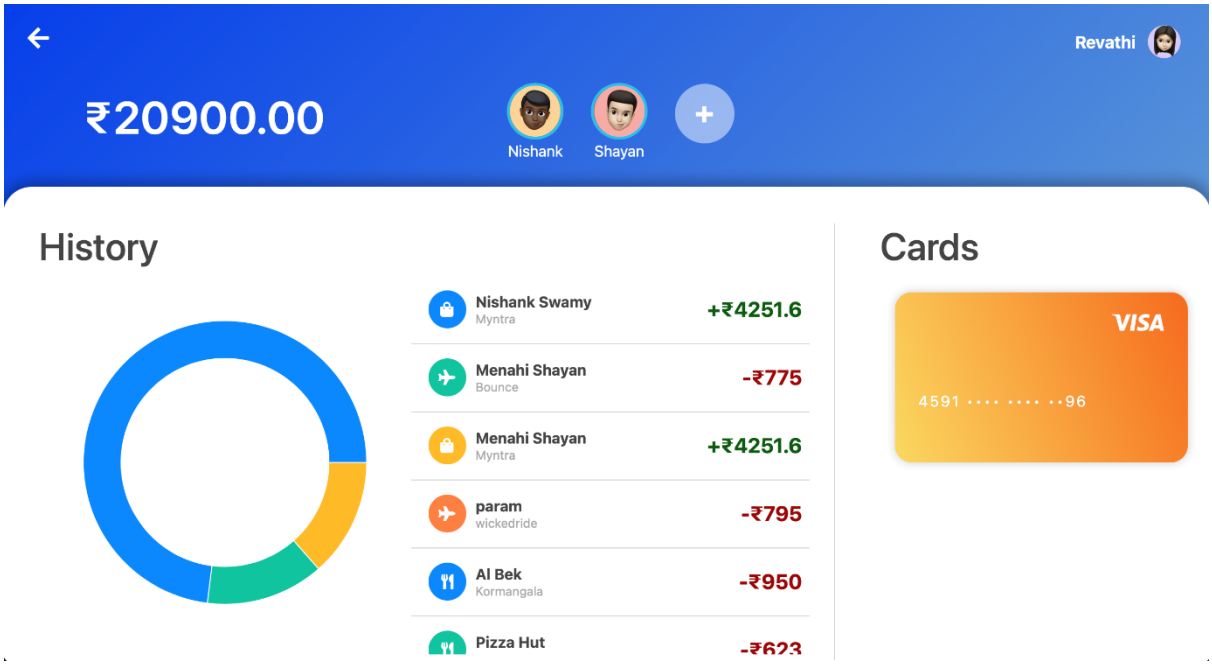


Fig (ix): View Profile

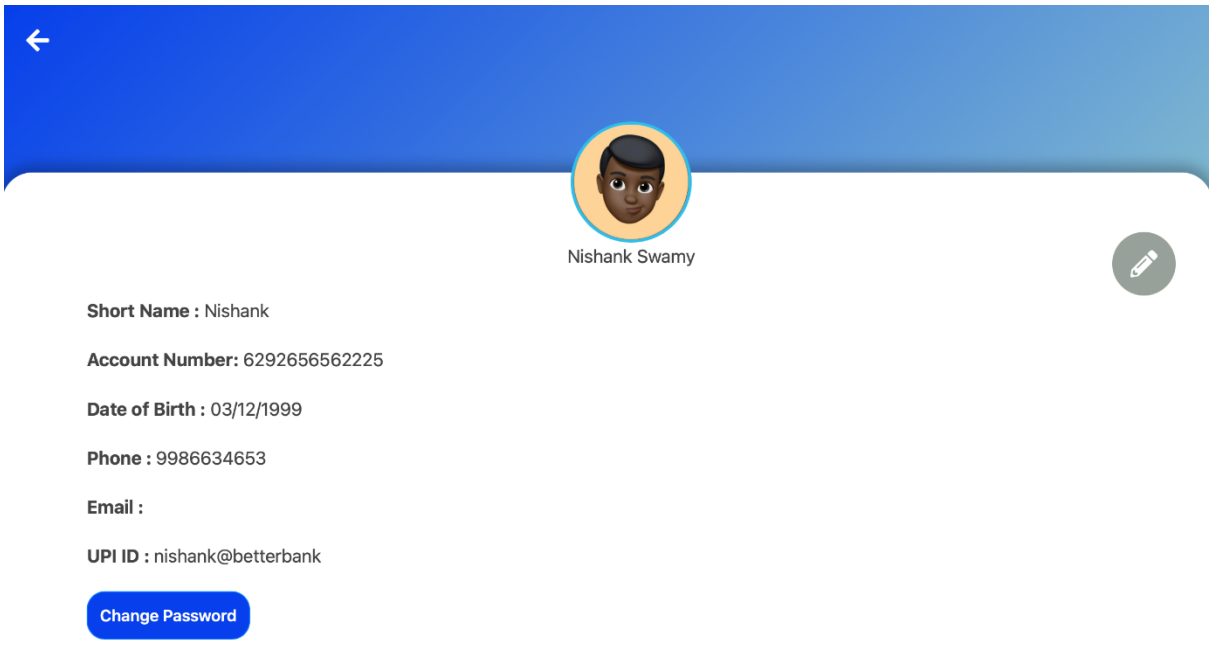


Fig (x): Edit Profile with Autofill Option

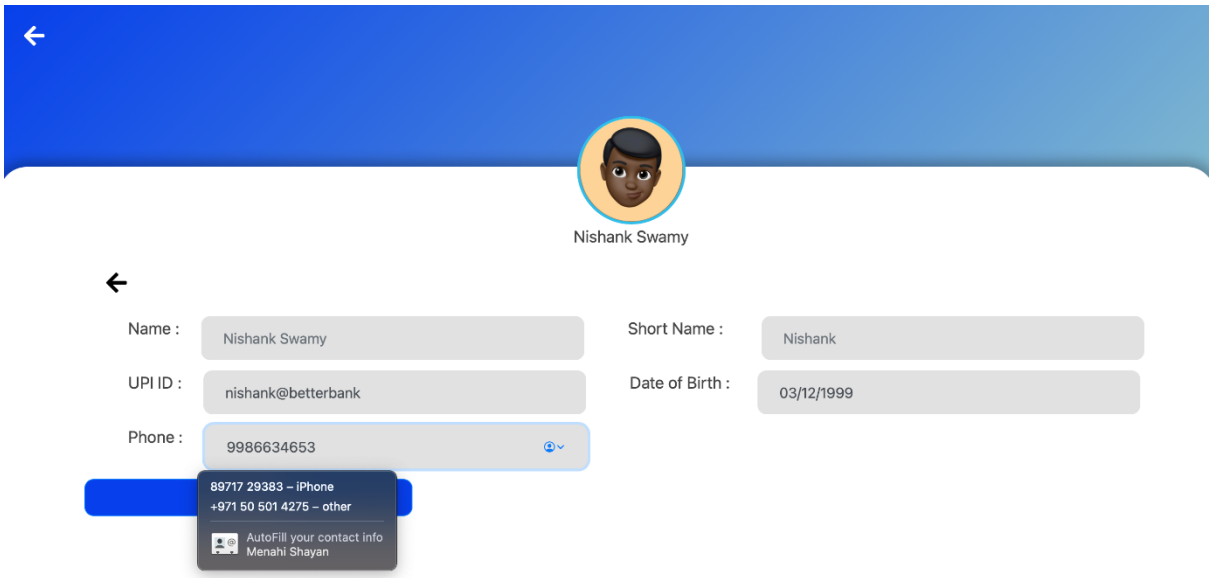


Fig (xi): Spending Analytics

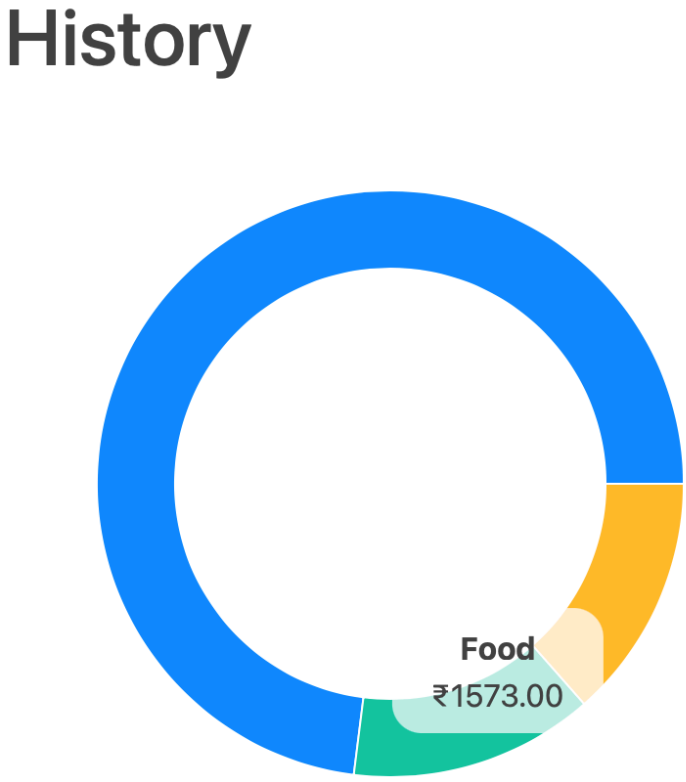


Fig (xii): Add Beneficiary

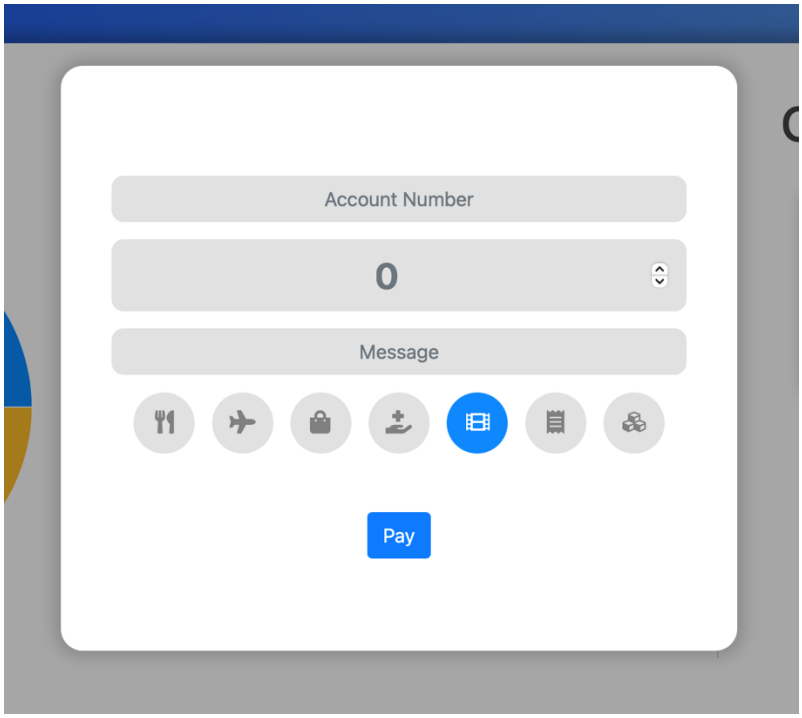


Fig (xiii): Transfer Amount to Beneficiary

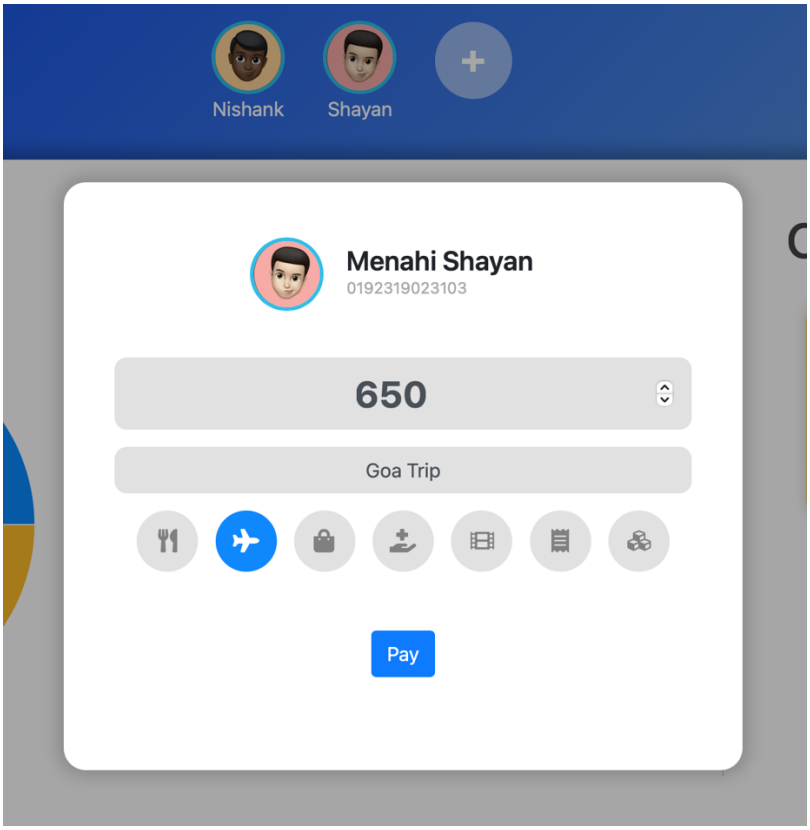
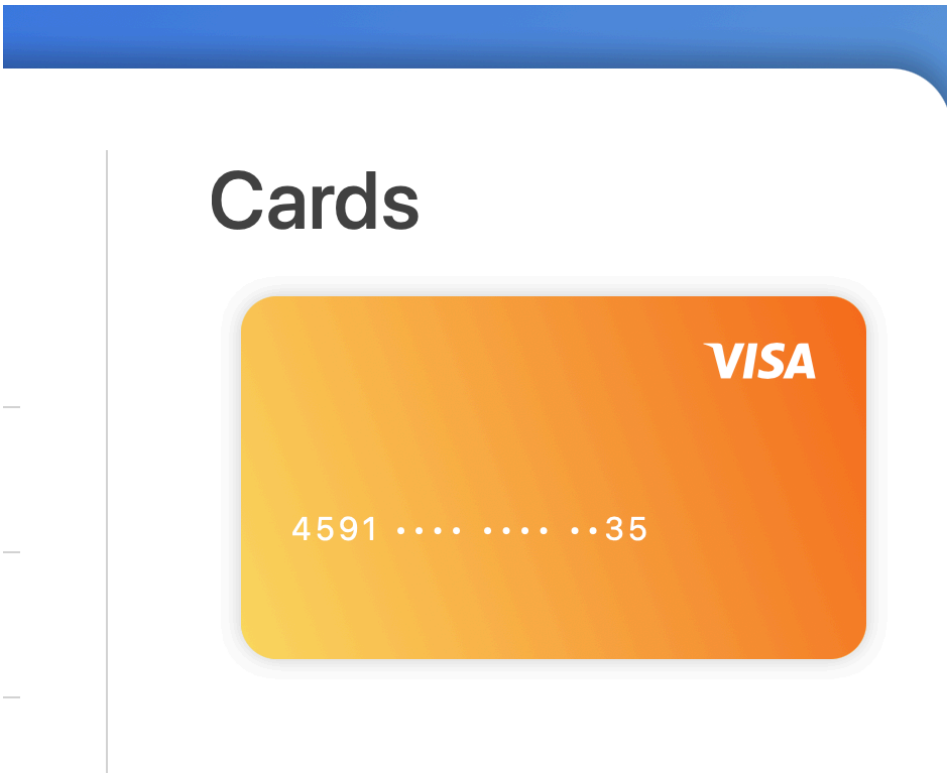


Fig (xiv): Virtual Credit/Debit Card



CONCLUSION

This project was successfully completed within the time span allotted. The project **Better Banking** has been developed using ReactJS and Firebase. All the modules are tested separately and put together to form the main system. Finally, the system is tested with real data and everything worked successfully. Thus the system has fulfilled the entire objective identified.

The system had been developed in an attractive dialogs fashion. So user with minimum knowledge about computers can also operate the system easily. It will make easy interactions between users and store. The speed and accuracy are maintained in proper way.

REFERENCES

1. www.google.com
2. https://firebase.google.com/docs/database
3. https://react-hook-form.com
4. https://react-bootstrap.github.io
5. www.w3schools.com
6. www.github.com
7. https://bootstrapdocs.com/
8. https://bootstrapdocs.com/
9. https://devdocs.io/html/
10. https://devdocs.io/javascript/
11. https://devdocs.io/css/
12. https://nodejs.org/en/docs/
13. https://reactjs.org