**CS631 DATA MANAGEMENT SYSTEM DESIGN**

**The WALLET Payment Network**

**Project Deliverable 3**

**GROUP MEMBERS**

|  |  |
| --- | --- |
| **Nehali Parulekar** | **np67** |
| **Harshal Bhole** | **hb32** |

* **GOAL:**

In this phase of the project, we implement the WALLET Payment Network. The android application is implemented using Node JS, MySQL as a backend in order to interact with the database and React at the frontend (UI). All the functions have been included in the application are as mentioned

* Users can sign up with WALLET by providing their name, SSN, an email address, and a phone number.
* Users can link multiple bank accounts with their account but they should define one of them as the primary funding source.
* In the WALLET Network, two or more users can be associated with the same bank account in case of joint accounts.
* There are two forms of actions that WALLET supports:   
   a. Send money to a user   
   b. Request money from users
* A payment to an email address or phone number that isn't associated with a WALLET   
  account is considered as payment to a new user. The recipient can accept the money   
  within 15 days by signing up to WALLET or by adding the additional email address to   
  an existing WALLET account. After 15 days the payment is cancelled and the funds are   
  returned to the sender. The same happens for money sent to an email address or   
  phone number which is not verified.
* Payments can be cancelled within 10 minutes after they are initiated. Cancelled   
  payments should be recorded in the database as well as the reason they are cancelled.
* Users’ payment history should be organized in monthly statements.
* **IMPLEMENTATION:**

This application satisfies all the needs of a typical e-wallet which helps all the people involved in manual processing of the information.

E-wallet Database application has been made to provide the admin and users with better facilities to perform the functionalities mentioned in the goals of the project. The main purpose of this is to provide easy and convenient access to the users to send or request money from other users and also track the transaction history.

The system provides information about the transactions between the users and also the status of the transaction with a new user who has not installed the app or created an account. The transaction is successful if the new user creates the account within 15 days of the transaction else the amount is sent back to the sender.

We started our process with creating the sql queries in MySQL, we first created all the tables without any constraints and determining the primary key and foreign keys. Then we inserted random data, making sure that the inserts are successful.

We then created the front end with React, by creating a simple looking screen for main menu, login page, Account, Statement.

After doing the major work, we connected the front UI with the backend of SQL, so that all the data can be easily accessed by the end user. So, while clicking any button or link it will fetch the details in the backend (DB) and after processing it will display the result to the user.

* **DIFFICULTIES WE FACED:**
* Creating APIs was an issue which we faced since this is the first app that we have developed.
* We also faced an issue with filtering out the data based on date and time.
* We were initially unable to track the issue while adding the REQUEST money tab, but later on we were successful in implementing it.
* We faced difficulties in optimizing the queries so as to make the apk of smaller size.
* **SQL COMMANDS FOR THE TABLES:**

1. **Users Table**

*CREATE TABLE users (*

*id int NOT NULL AUTO\_INCREMENT,*

*created\_at datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,*

*updated\_at datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*

*name varchar (255),*

*role varchar (50) DEFAULT 'user',*

*email varchar (255) NOT NULL UNIQUE,*

*phone varchar (255) NOT NULL UNIQUE,*

*password varchar (255),*

*balance decimal (10,2) NOT NULL,*

*PRIMARY KEY (id)*

*) AUTO\_INCREMENT = 1;*

2. **Transactions Table**

*CREATE TABLE transactions (*

*id int NOT NULL AUTO\_INCREMENT,*

*created\_at datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,*

*updated\_at datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*

*sender\_id int NOT NULL,*

*receiver\_id int NOT NULL,*

*amount decimal (8,2) NOT NULL,*

*transaction\_date datetime NOT NULL,*

*transaction\_type varchar (50) NOT NULL,*

*comments varchar (255),*

*PRIMARY KEY (id),*

*CONSTRAINT FK\_sender\_id FOREIGN KEY (sender\_id) REFERENCES users(id),*

*CONSTRAINT FK\_receiver\_id FOREIGN KEY (receiver\_id) REFERENCES users(id)*

*) AUTO\_INCREMENT = 1;*

3. **Transactions\_in\_progress Table**

*CREATE TABLE transactions\_in\_progress (*

*id int NOT NULL AUTO\_INCREMENT,*

*created\_at datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,*

*updated\_at datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*

*sender\_id int NOT NULL,*

*receiver varchar (255) NOT NULL,*

*receiver\_identifier varchar (50) NOT NULL,*

*amount decimal (8,2) NOT NULL,*

*transaction\_date datetime,*

*transaction\_type varchar (50) NOT NULL,*

*comments varchar (255),*

*PRIMARY KEY (id),*

*CONSTRAINT FK\_transactions\_in\_progress\_sender\_id FOREIGN KEY (sender\_id) REFERENCES users(id)*

*) AUTO\_INCREMENT = 1;*

**4. Requests Table**

*CREATE TABLE requests (*

*id int NOT NULL AUTO\_INCREMENT,*

*created\_at datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP,*

*updated\_at datetime NOT NULL DEFAULT CURRENT\_TIMESTAMP ON UPDATE CURRENT\_TIMESTAMP,*

*sender\_id int NOT NULL,*

*receiver\_id int NOT NULL,*

*amount decimal (8,2) NOT NULL,*

*comments varchar (255),*

*`status` varchar (255) NOT NULL DEFAULT 'active',*

*PRIMARY KEY (id),*

*CONSTRAINT FK\_requests\_sender\_id FOREIGN KEY (sender\_id) REFERENCES users(id),*

*CONSTRAINT FK\_requests\_receiver\_id FOREIGN KEY (receiver\_id) REFERENCES users(id)*

*) AUTO\_INCREMENT = 1;*

* **SQL COMMANDS FOR POPULATING THE TABLES:**

INSERT INTO transactions (sender\_id, receiver\_id, amount, transaction\_date, transaction\_type, comments) VALUES (${req.user.id}, ${receiverUser.id}, ${amount}, now(), '${transactionMode}', ${comments});

INSERT INTO transactions\_in\_progress (sender\_id, receiver, receiver\_identifier, amount, transaction\_date, transaction\_type, comments) VALUES (${req.user.id}, '${receiver}', '${receiverIdentifier}', ${amount}, null, '${transactionMode}', ${comments});

INSERT INTO requests (sender\_id, receiver\_id, amount, comments, status) VALUES (${req.user.id}, ${receiverUser.id}, ${amount}, ${comments}, "active");

* **SOURCE CODE:**

**1. mysql.js**

const mysql = require("mysql2")

const con = mysql.createConnection({

host: "db4free.net",

user: "harshal",

password: "harshal123",

database: "sqlproj",

multipleStatements: true

})

con.connect((err) => {

if (err) {

console.log(err)

return

}

console.log("MySql Database Connected")

})

module.exports = con

**2. auth.js**

const { dbQuery } = require("../common/utils")

const auth = (roleKey) => {

return async (req, res, next) => {

var userId = req.params[roleKey]

req.userId = userId

let data = await dbQuery(`select \* from users where id = ${req.userId}`)

if (!data.length) {

res.status(404).send("User not found")

return True

}

req.user = data[0]

next()

}

}

module.exports = auth

**3. index.js**

const app = require("./app")

const port = process.env.PORT

app.listen(port, () => {

console.log("Server is up on port " + port)

})

**4. user.js**

const express = require("express")

const auth = require("../middleware/auth")

const { dbQuery } = require("../common/utils")

const router = new express.Router()

router.post("/user/login", async (req, res) => {

const email = req.body.email

const password = req.body.password

if (!(email && password)) {

res.status(400).send("Please provide Email and Password")

return true

}

results = await dbQuery(

`SELECT \* FROM users WHERE email='${email}' AND password='${password}';`

)

if (!results.length) {

res.status(401).send({ error: "Unauthorized" })

return true

}

user = results[0]

data = {

userId: user.id,

name: user.name,

email: user.email,

phone: user.phone,

balance: user.balance

}

res.send(data)

})

router.get("/user/:userId/profile", auth("userId"), async (req, res) => {

data = {

userId: req.user.id,

name: req.user.name,

email: req.user.email,

phone: req.user.phone,

balance: req.user.balance

}

res.send(data)

})

router.post("/user/:userId/profile-update", auth("userId"), async (req, res) => {

try {

query = `UPDATE users SET name='${req.body.name}', email='${req.body.email}', phone='${req.body.phone}' WHERE id=${req.user.id};`

await dbQuery(query)

data = req.body

data.userId = req.user.id

data.balance = req.user.balance

res.send(data)

} catch (e) {

res.status(400).send("Update Failed")

return true

}

})

router.post("/user/:userId/get-total-amount", auth("userId"), async (req, res) => {

const fromDate = req.body.fromDate

const toDate = req.body.toDate

if (!(fromDate && toDate)) {

res.status(400).send("Please provide Date Range")

return true

}

results = await dbQuery(

`select sum(case when sender\_id = ${req.user.id} then amount else null end) as amount\_sent, sum(case when receiver\_id = ${req.user.id} then amount else null end) as amount\_received from transactions where sender\_id = ${req.user.id} or receiver\_id = ${req.user.id} and date(transaction\_date) >= '${fromDate}' and date(transaction\_date) <= '${toDate}';`

)

if (!results.length) {

res.status(404).send("No Data Found")

return true

}

data = results[0]

res.send(data)

})

router.post("/user/:userId/transactions", auth("userId"), async (req, res) => {

const fromDate = req.body.fromDate

const toDate = req.body.toDate

const month = req.body.month

const receiverName = req.query.receiverName ? req.query.receiverName : ""

const receiverEmail = req.query.receiverEmail ? req.query.receiverEmail : ""

const receiverPhone = req.query.receiverPhone ? req.query.receiverPhone : ""

const type = req.query.type ? req.query.type : ""

const userId = req.query.userId ? req.query.userId : null

query = `SELECT \* FROM (SELECT a.transaction\_id, IF(a.type = 'debit', receiver.id, sender.id) AS account\_id, IF(a.type = 'debit', receiver.name, sender.name) AS name, a.type, a.amount, a.transaction\_date, a.transaction\_type AS mode FROM (SELECT transactions.id AS transaction\_id, transactions.amount, transactions.transaction\_date, transactions.transaction\_type, transactions.sender\_id, transactions.receiver\_id, IF(transactions.sender\_id = ${req.user.id}, 'debit', 'credit') AS type FROM transactions WHERE (transactions.sender\_id = ${req.user.id} OR transactions.receiver\_id = ${req.user.id})) a, users sender, users receiver WHERE a.sender\_id = sender.id AND a.receiver\_id = receiver.id AND receiver.name LIKE "%${receiverName}%" AND receiver.email LIKE "%${receiverEmail}%" AND receiver.phone LIKE "%${receiverPhone}%" AND a.type LIKE "%${type}%"`

if (fromDate) query += ` AND DATE(transaction\_date) >= '${fromDate}'`

if (toDate) query += ` AND DATE(transaction\_date) <= '${toDate}'`

if (month) query += ` AND MONTH(transaction\_date) = '${month}'`

query += `) b`

if (userId) query += ` WHERE account\_id = ${userId}`

query += " ORDER BY transaction\_date DESC;"

results = await dbQuery(query)

if (!results.length) {

res.status(404).send("No Data Found")

return true

}

data = results

res.send(data)

})

router.get("/get-best-users", async (req, res) => {

results = await dbQuery(

`SELECT u.name, SUM(t.amount) AS total\_transactions\_amount FROM users u LEFT JOIN transactions t ON (u.id = t.sender\_id OR u.id = t.receiver\_id) group by u.id ORDER BY SUM(t.amount) DESC LIMIT 3;`

)

res.send(results)

})

router.post("/user/:userId/amount-per-month", auth("userId"), async (req, res) => {

const month = req.body.month

if (!month) {

res.status(400).send("Please provide Month")

return true

}

results = await dbQuery(

`select sum(case when t.sender\_id = ${req.user.id} then t.amount else null end) as total\_amount\_sent, sum(case when t.receiver\_id = ${req.user.id} then t.amount else null end) as total\_amount\_received, avg(case when t.sender\_id = ${req.user.id} then t.amount else null end) as average\_amount\_sent, avg(case when t.receiver\_id = ${req.user.id} then t.amount else null end) as average\_amount\_received from transactions t where t.sender\_id = ${req.user.id} or t.receiver\_id = ${req.user.id} and month(t.transaction\_date) = ${month};`

)

if (!results.length) {

res.status(404).send("No Data Found")

return true

}

data = results[0]

res.send(data)

})

router.post("/max-amount-for-month", async (req, res) => {

const month = req.body.month

if (!month) {

res.status(400).send("Please provide Month")

return true

}

results = await dbQuery(

`select t.id as transaction\_id, s.name as sender, r.name as receiver, t.amount, t.transaction\_date, t.transaction\_type from transactions t, users s, users r where t.sender\_id = s.id and t.receiver\_id = r.id and amount = (select max(amount) from transactions where month(transaction\_date) = ${month});`

)

if (!results.length) {

res.status(404).send("No Data Found")

return true

}

res.send(results)

})

router.post("/user/:userId/send-money", auth("userId"), async (req, res) => {

const amount = req.body.amount

const transactionMode = req.body.transactionMode

const receiverIdentifier = req.body.receiverIdentifier

const receiver = req.body.receiver

const comments = req.body.comments ? "'" + req.body.comments + "'" : "''"

if (!(amount && transactionMode)) {

res.status(400).send("Please provide all transactions details")

return true

}

if (Number(amount) > 10000) {

res.status(400).send("Max Amount is 10000")

return true

}

if (!(receiver && receiverIdentifier)) {

res.status(400).send("Please provide Receiver Details")

return true

}

currentBalance = Number(req.user.balance)

newBalance = currentBalance - amount

if (newBalance < 0) {

res.status(400).send("You don't have enough balance")

return true

}

if (receiverIdentifier == "accountId") {

if (!typeof receiver == "number") {

res.status(400).send("Account ID should be Number")

return true

}

results = await dbQuery(`SELECT \* FROM users WHERE id = ${receiver};`)

if (!results.length) {

res.status(400).send("Reciver Account ID does not exist")

return true

}

receiverUser = results[0]

if (receiverUser.id == req.user.id) {

res.status(400).send("Cannot send money to same account")

return true

}

await dbQuery(

`INSERT INTO transactions (sender\_id, receiver\_id, amount, transaction\_date, transaction\_type, comments) VALUES (${req.user.id}, ${receiverUser.id}, ${amount}, now(), '${transactionMode}', ${comments});`

)

await dbQuery(`UPDATE users SET balance = ${newBalance} WHERE id = ${req.user.id}`)

newReceiverBalance = Number(receiverUser.balance) + amount

await dbQuery(

`UPDATE users SET balance = ${newReceiverBalance} WHERE id = ${receiverUser.id}`

)

} else {

results = await dbQuery(

`SELECT \* FROM users WHERE email = '${receiver}' OR phone = '${receiver}';`

)

if (results.length) {

receiverUser = results[0]

if (receiverUser.id == req.user.id) {

res.status(400).send("Cannot send money to same account")

return true

}

await dbQuery(

`INSERT INTO transactions (sender\_id, receiver\_id, amount, transaction\_date, transaction\_type, comments) VALUES (${req.user.id}, ${receiverUser.id}, ${amount}, now(), '${transactionMode}', ${comments});`

)

await dbQuery(`UPDATE users SET balance = ${newBalance} WHERE id = ${req.user.id}`)

newReceiverBalance = Number(receiverUser.balance) + amount

await dbQuery(

`UPDATE users SET balance = ${newReceiverBalance} WHERE id = ${receiverUser.id}`

)

} else {

await dbQuery(

`INSERT INTO transactions\_in\_progress (sender\_id, receiver, receiver\_identifier, amount, transaction\_date, transaction\_type, comments) VALUES (${req.user.id}, '${receiver}', '${receiverIdentifier}', ${amount}, null, '${transactionMode}', ${comments});`

)

await dbQuery(`UPDATE users SET balance = ${newBalance} WHERE id = ${req.user.id}`)

res.send(

"Account with given details doesn't exist. If user signs up in 15 days money will be transferred, else it will be refunded"

)

return true

}

}

res.send("Transaction Completed Succesfully")

})

router.post("/user/:userId/request-money", auth("userId"), async (req, res) => {

const amount = req.body.amount

const receiverIdentifier = req.body.receiverIdentifier

const receiver = req.body.receiver

const comments = req.body.comments ? "'" + req.body.comments + "'" : "''"

if (!amount) {

res.status(400).send("Please provide amount")

return true

}

if (Number(amount) > 10000) {

res.status(400).send("Max Amount is 10000")

return true

}

if (!(receiver && receiverIdentifier)) {

res.status(400).send("Please provide Receiver Details")

return true

}

if (receiverIdentifier == "accountId") {

if (!typeof receiver == "number") {

res.status(400).send("Account ID should be Number")

return true

}

results = await dbQuery(`SELECT \* FROM users WHERE id = ${receiver};`)

if (!results.length) {

res.status(400).send("Reciver Account ID does not exist")

return true

}

receiverUser = results[0]

if (receiverUser.id == req.user.id) {

res.status(400).send("Cannot request money from same account")

return true

} else {

await dbQuery(

`INSERT INTO requests (sender\_id, receiver\_id, amount, comments, status) VALUES (${req.user.id}, ${receiverUser.id}, ${amount}, ${comments}, "active");`

)

}

} else {

results = await dbQuery(

`SELECT \* FROM users WHERE email = '${receiver}' OR phone = '${receiver}';`

)

if (!results.length) {

res.status(404).send("Account with given details doesn't exist")

return true

}

receiverUser = results[0]

if (receiverUser.id == req.user.id) {

res.status(400).send("Cannot request money from same account")

return true

}

await dbQuery(

`INSERT INTO requests (sender\_id, receiver\_id, amount, comments, status) VALUES (${req.user.id}, ${receiverUser.id}, ${amount}, ${comments}, "active");`

)

}

res.send("Request Sent Successfully")

})

router.get("/user/:userId/requests", auth("userId"), async (req, res) => {

const status = req.query.status ? req.query.status : ""

results = await dbQuery(

`SELECT r.id as request\_id, r.created\_at as requested\_time, r.sender\_id, s.name as request\_sender, r.amount, r.comments, r.status FROM requests r, users s, users rec WHERE r.sender\_id = s.id AND r.receiver\_id = rec.id AND rec.id = ${req.user.id} AND r.status LIKE "%${status}%"`

)

res.send(results)

})

router.post("/user/:userId/accept-request", auth("userId"), async (req, res) => {

const transactionMode = req.body.transactionMode

const requestId = req.body.requestId

const action = req.body.action

const comments = req.body.comments ? "'" + req.body.comments + "'" : "''"

if (!(requestId && action)) {

res.status(400).send("Invalid Request")

return true

}

results = await dbQuery(`SELECT \* FROM requests WHERE id = ${requestId};`)

if (!results.length) {

res.status(400).send("Invalid Request ID")

return true

}

requestObject = results[0]

if (req.user.id != requestObject.receiver\_id || requestObject.status != "active") {

res.status(400).send("Invalid Transaction")

return true

}

if (action == "reject") {

await dbQuery(`UPDATE requests SET status = 'rejected' WHERE id = ${requestObject.id}`)

res.send("Request Rejected Succesfully")

return true

}

if (!transactionMode) {

res.status(400).send("Please select mode of transaction")

return true

}

amount = Number(requestObject.amount)

currentBalance = Number(req.user.balance)

newBalance = currentBalance - amount

if (newBalance < 0) {

res.status(400).send("You don't have enough balance to accept this request")

return true

}

results = await dbQuery(`SELECT \* FROM users WHERE id = ${requestObject.receiver\_id};`)

receiverUser = results[0]

await dbQuery(

`INSERT INTO transactions (sender\_id, receiver\_id, amount, transaction\_date, transaction\_type, comments) VALUES (${req.user.id}, ${requestObject.sender\_id}, ${amount}, now(), '${transactionMode}', ${comments});`

)

await dbQuery(`UPDATE users SET balance = ${newBalance} WHERE id = ${req.user.id}`)

newReceiverBalance = Number(receiverUser.balance) + amount

await dbQuery(`UPDATE users SET balance = ${newReceiverBalance} WHERE id = ${receiverUser.id}`)

await dbQuery(`UPDATE requests SET status = 'accepted' WHERE id = ${requestObject.id}`)

res.send("Request Accepted Successfully")

})

module.exports = router

**5. package.js**

{

"name": "lq-live-node",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"start": "node src/index.js",

"dev": "env-cmd -f ./config/dev.env nodemon src/index.js"

},

"keywords": [],

"author": "",

"license": "ISC",

"dependencies": {

"express": "latest",

"mysql2": "latest"

},

"devDependencies": {

"nodemon": "latest",

"env-cmd": "latest"

}

}

**6. Account.js**

import {

View,

Text,

StyleSheet,

TextInput,

TouchableOpacity,

ActivityIndicator,

ToastAndroid,

} from 'react-native';

import React, {Component} from 'react';

import DropDownPicker from 'react-native-dropdown-picker';

var axios = require('axios');

export default class HomeScreen extends React.Component {

constructor(props) {

super(props);

this.state = {

name: '',

email: '',

phone: '',

items: [

{label: 'January', value: 'January'},

{label: 'February', value: 'February'},

{label: 'March', value: 'March'},

{label: 'April', value: 'April'},

{label: 'May', value: 'May'},

{label: 'June', value: 'June'},

{label: 'July', value: 'July'},

{label: 'August', value: 'August'},

{label: 'September', value: 'September'},

{label: 'October', value: 'October'},

{label: 'November', value: 'November'},

{label: 'December', value: 'December'},

],

itemsSearch: [

{label: 'SSN', value: 'SSn'},

{label: 'Email', value: 'Email'},

{label: 'Phone Number', value: 'Phone'},

{label: 'Type of transaction', value: 'type'},

{label: 'Time/Date', value: 'Time'},

],

selectedMonth: null,

};

}

componentDidMount() {

this.fetchProfile();

}

fetchProfile = () => {

axios

.get(

`https://harshal-trasactions-project.herokuapp.com/user/${this.props.id}/profile`,

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

},

)

.then(response => {

console.log('response', response.data);

this.setState({

email: response.data && response.data.email,

name: response.data && response.data.name,

phone: response.data && response.data.phone,

balance: response.data && response.data.balance,

});

})

.catch(e => console.log('op', e));

};

handleOnchange = (type, data) => {

this.setState({[type]: data});

};

updateDetails = () => {

this.setState({loader: true});

axios

.post(

`https://harshal-trasactions-project.herokuapp.com/user/${this.props.id}/profile-update`,

{

email: this.state.email,

phone: this.state.phone,

name: this.state.name,

},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

},

)

.then(response => {

ToastAndroid.show('Data Updated', ToastAndroid.SHORT);

this.setState({loader: false});

this.fetchProfile();

})

.catch(e => {

if (typeof e.response.data == 'string') {

ToastAndroid.show(e.response.data, ToastAndroid.SHORT);

} else {

ToastAndroid.show(e.response.data.error, ToastAndroid.SHORT);

}

this.setState({loader: false});

});

};

render() {

console.log('this.state', this.state.valueSearch);

return (

<View style={styles.container}>

<Text style={{fontSize: 20, color: '#fff', paddingTop: 15}}>

Account Functions

</Text>

<View

style={{

flexDirection: 'row',

justifyContent: 'space-between',

paddingVertical: 10,

alignItems: 'center',

}}>

<Text style={{color: '#fff', marginRight: 20}}>Name:</Text>

<TextInput

placeholder="Enter Name"

style={[

{

width: '50%',

marginVertical: 10,

justifyContent: 'center',

alignItems: 'center',

borderWidth: 1,

color: '#000',

backgroundColor: '#fff',

},

]}

onChangeText={text => this.handleOnchange('name', text)}

value={this.state.name}

/>

</View>

<View

style={{

flexDirection: 'row',

justifyContent: 'space-between',

paddingVertical: 10,

alignItems: 'center',

}}>

<Text style={{color: '#fff', marginRight: 20}}>Phone:</Text>

<TextInput

placeholder="Enter Phone"

style={[

{

width: '50%',

marginVertical: 10,

justifyContent: 'center',

alignItems: 'center',

borderWidth: 1,

color: '#000',

backgroundColor: '#fff',

},

]}

onChangeText={text => this.handleOnchange('phone', text)}

value={this.state.phone}

/>

</View>

<View

style={{

flexDirection: 'row',

justifyContent: 'space-between',

paddingVertical: 10,

alignItems: 'center',

}}>

<Text style={{color: '#fff', marginRight: 20}}>Email:</Text>

<TextInput

placeholder="Enter Email"

style={[

{

width: '50%',

marginVertical: 10,

justifyContent: 'center',

alignItems: 'center',

borderWidth: 1,

color: '#000',

backgroundColor: '#fff',

},

]}

onChangeText={text => this.handleOnchange('email', text)}

value={this.state.email}

/>

</View>

<View

style={{

flexDirection: 'row',

justifyContent: 'space-between',

paddingVertical: 10,

alignItems: 'center',

}}>

<Text style={{color: '#fff', marginRight: 20}}>Balance:</Text>

<Text style={{color: '#fff', marginRight: 20}}>

{this.state.balance}

</Text>

</View>

<View style={styles.buttonContainer}>

<TouchableOpacity

onPress={() => {

this.updateDetails();

}}>

{this.state.loader ? (

<ActivityIndicator color={'silver'} size="small" />

) : (

<Text style={{color: '#fff', textAlign: 'center'}}>Update</Text>

)}

</TouchableOpacity>

</View>

</View>

);

}

}

const styles = StyleSheet.create({

container: {

flex: 1,

alignItems: 'center',

justifyContent: 'center',

paddingHorizontal: 15,

},

textInputStyle: {

borderWidth: 1,

width: '40%',

},

buttonContainer: {

backgroundColor: '#000',

color: '#fff',

padding: 10,

margin: 8,

borderRadius: 5,

width: '40%',

},

});

6. Home.js

import {

View,

Text,

StyleSheet,

TextInput,

TouchableOpacity,

ToastAndroid,

ActivityIndicator,

ScrollView,

} from 'react-native';

import React, {Component} from 'react';

import DropDownPicker from 'react-native-dropdown-picker';

var axios = require('axios');

import moment from 'moment';

import Modal from 'react-native-modal';

import DateTimePicker from '@react-native-community/datetimepicker';

export default class HomeScreen extends React.Component {

constructor(props) {

super(props);

this.state = {

userData: null,

text: '',

open: false,

openSearch: false,

value: null,

email\_phone: '',

amount: '',

from\_date: '',

to\_date: '',

items: [

{label: 'January', value: '1'},

{label: 'February', value: '2'},

{label: 'March', value: '3'},

{label: 'April', value: '4'},

{label: 'May', value: '5'},

{label: 'June', value: '6'},

{label: 'July', value: '7'},

{label: 'August', value: '8'},

{label: 'September', value: '9'},

{label: 'October', value: '10'},

{label: 'November', value: '11'},

{label: 'December', value: '12'},

],

itemsSearch: [

{label: 'SSN', value: 'ssn'},

{label: 'Email', value: 'email'},

{label: 'Phone Number', value: 'phone'},

{label: 'Type of transaction', value: 'type'},

{label: 'Time/Date', value: 'time'},

],

transactionItems: [

{label: 'Credit Card', value: 'credit\_card'},

{label: 'Debit Card', value: 'debit\_card'},

{label: 'UPI', value: 'upi'},

],

Identifiers: [

{label: 'Email', value: 'email'},

{label: 'Phone', value: 'phone'},

],

selectedMonth: null,

};

}

componentDidMount() {

axios

.get(

`https://harshal-trasactions-project.herokuapp.com/user/${this.props.id}/profile`,

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

},

)

.then(response => {

console.log('response', response.data);

this.setState({userData: response.data});

})

.catch(e => console.log('op', e));

}

handleClassSelection = value => {

console.log('value', value);

this.setState({

selectedMonth: value,

});

};

handleOnchange = (type, data) => {

this.setState({[type]: data});

};

sendMoney = () => {

this.setState({sendMoneyLoader: true});

console.log(

'hjhcdcjkdgjcdkjc',

this.state.amount,

this.state.valueTransaction,

this.state.valueIdentifier,

this.state.email\_phone,

);

axios

.post(

`https://harshal-trasactions-project.herokuapp.com/user/${this.props.id}/send-money`,

{

amount: Number(this.state.amount),

transactionMode: this.state.valueTransaction,

receiverIdentifier: this.state.valueIdentifier,

receiver: this.state.email\_phone,

comments: this.state.comment,

},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

},

)

.then(response => {

this.setState({sendMoneyLoader: false});

console.log('response request', response.data);

if (typeof response.data == 'string') {

ToastAndroid.show(response.data, ToastAndroid.SHORT);

} else {

ToastAndroid.show('Money Requested', ToastAndroid.SHORT);

}

})

.catch(e => {

if (typeof e.response.data == 'string') {

ToastAndroid.show(e.response.data, ToastAndroid.SHORT);

} else {

ToastAndroid.show(e.response.data.error, ToastAndroid.SHORT);

}

this.setState({sendMoneyLoader: false});

});

};

acceptReject = (id, action) => {

this.setState({acceptRejectLoader: true});

console.log(id, action);

axios

.post(

`https://harshal-trasactions-project.herokuapp.com/user/${this.props.id}/accept-request`,

{

requestId: id,

transactionMode: 'upi',

action: action,

comments: 'done',

},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

},

)

.then(response => {

this.setState({acceptRejectLoader: false, requestsData: null});

console.log('response', response.data);

ToastAndroid.show(response.data, ToastAndroid.SHORT);

})

.catch(e => {

console.log('e', e.response);

if (typeof e.response.data == 'string') {

ToastAndroid.show(e.response.data, ToastAndroid.SHORT);

} else {

ToastAndroid.show(e.response.data.error, ToastAndroid.SHORT);

}

this.setState({acceptRejectLoader: false, requestsData: null});

});

};

renderModal = () => {

console.log('requestsData', this.state.requestsData);

return (

<Modal

animationInTiming={800}

animationOutTiming={600}

backdropTransitionInTiming={700}

backdropTransitionOutTiming={400}

animationIn={'slideInUp'}

animationOut={'slideOutDown'}

isVisible={this.state.requestsData}

style={{

width: '90%',

flexDirection: 'column',

zIndex: 1,

}}

onBackButtonPress={() => {

this.setState({requestsData: null});

}}

statusBarTranslucent={true}

onBackdropPress={() => {

this.setState({requestsData: null});

}}

useNativeDriver={true}

hideModalContentWhileAnimating={true}>

<View style={{backgroundColor: '#fff', padding: 20}}>

<ScrollView contentInsetAdjustmentBehavior="automatic">

{this.state.requestsData &&

this.state.requestsData.map(a => (

<View

style={{

borderWidth: 1,

backgroundColor: '#5b5c60',

borderRadius: 5,

marginVertical: 10,

padding: 5,

}}>

<Text style={{color: '#fff'}}>Name : {a.request\_sender}</Text>

<Text style={{color: '#fff'}}>

Amount Requested : {a.amount}

</Text>

<Text style={{color: '#fff'}}>Comment : {a.comments}</Text>

{a.status !== 'active' ? (

<Text

style={{

color: a.status == 'accepted' ? '#53B64B' : '#ED3833',

paddingVertical: 10,

}}>

{a.status == 'accepted' ? `✓ Accepted` : '✖ Rejected'}

</Text>

) : null}

{a.status == 'active' ? (

<View

style={{

flexDirection: 'row',

justifyContent: 'space-around',

width: '100%',

}}>

<View

style={[

styles.buttonContainer,

{backgroundColor: '#53B64B'},

]}>

<TouchableOpacity

onPress={() => {

this.acceptReject(a.request\_id, 'accept');

}}>

{this.state.acceptRejectLoader ? (

<ActivityIndicator color={'silver'} size="small" />

) : (

<Text style={{color: '#fff', textAlign: 'center'}}>

Accept

</Text>

)}

</TouchableOpacity>

</View>

<View

style={[

styles.buttonContainer,

{backgroundColor: '#ED3833'},

]}>

<TouchableOpacity

onPress={() => {

this.acceptReject(a.request\_id, 'reject');

}}>

{this.state.acceptRejectLoader ? (

<ActivityIndicator color={'silver'} size="small" />

) : (

<Text style={{color: '#fff', textAlign: 'center'}}>

Reject

</Text>

)}

</TouchableOpacity>

</View>

</View>

) : null}

</View>

))}

</ScrollView>

</View>

</Modal>

);

};

requestMoney = () => {

this.setState({requestMoneyLoader: true});

axios

.post(

`https://harshal-trasactions-project.herokuapp.com/user/${this.props.id}/request-money`,

{

amount: Number(this.state.amount),

receiverIdentifier: this.state.valueIdentifier,

receiver: this.state.email\_phone,

comments: this.state.comment,

},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

},

)

.then(response => {

this.setState({requestMoneyLoader: false});

console.log('response request', response.data);

if (typeof response.data == 'string') {

ToastAndroid.show(response.data, ToastAndroid.SHORT);

} else {

ToastAndroid.show('Money Requested', ToastAndroid.SHORT);

}

})

.catch(e => {

if (typeof e.response.data == 'string') {

ToastAndroid.show(e.response.data, ToastAndroid.SHORT);

} else {

ToastAndroid.show(e.response.data.error, ToastAndroid.SHORT);

}

this.setState({requestMoneyLoader: false});

});

};

setOpen = open => {

console.log('open', open);

this.setState({

open,

});

};

setValue = callback => {

this.setState(state => ({

value: callback(state.value),

}));

};

setItems = callback => {

this.setState(state => ({

items: callback(state.items),

}));

};

setOpenSearch = open => {

console.log('', open);

this.setState({

openSearch: open,

});

};

setValueSearch = callback => {

this.setState(state => ({

valueSearch: callback(state.valueSearch),

}));

};

setItemsSearch = callback => {

this.setState(state => ({

itemsSearch: callback(state.itemsSearch),

}));

};

setTransactionOpen = open => {

console.log('', open);

this.setState({

transactionOpen: open,

});

};

setValueTransaction = callback => {

this.setState(state => ({

valueTransaction: callback(state.valueTransaction),

}));

};

setTransactionItem = callback => {

this.setState(state => ({

transactionItems: callback(state.transactionItems),

}));

};

setIdentifierOpen = open => {

console.log('', open);

this.setState({

openIdentifier: open,

});

};

setValueIdentifier = callback => {

this.setState(state => ({

valueIdentifier: callback(state.valueTransaction),

}));

};

setIdentifierItem = callback => {

this.setState(state => ({

Identifiers: callback(state.Identifiers),

}));

};

getStatement = () => {

this.setState({statementLoader: true});

console.log('this.state.value', this.state.value);

axios

.post(

`https://harshal-trasactions-project.herokuapp.com/user/${this.props.id}/transactions`,

{month: this.state.value},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

},

)

.then(response => {

this.setState({statementLoader: false, statementData: response.data});

console.log('response request', response.data);

if (typeof response.data == 'string') {

ToastAndroid.show(response.data, ToastAndroid.SHORT);

} else {

ToastAndroid.show('Statement Fetched', ToastAndroid.SHORT);

}

})

.catch(e => {

console.log('e', e.response.data);

if (typeof e.response.data == 'string') {

ToastAndroid.show(e.response.data, ToastAndroid.SHORT);

} else {

ToastAndroid.show(e.response.data.error, ToastAndroid.SHORT);

}

this.setState({statementLoader: false});

});

};

fetchRequests = () => {

this.setState({fetchRequestLoader: true});

axios

.get(

`https://harshal-trasactions-project.herokuapp.com/user/${this.props.id}/requests`,

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

},

)

.then(response => {

this.setState({fetchRequestLoader: false});

console.log('response request', response.data);

if (response.data && response.data.length == 0) {

ToastAndroid.show('No requests found', ToastAndroid.SHORT);

} else {

this.setState({requestsData: response.data});

ToastAndroid.show('Requests Fetched', ToastAndroid.SHORT);

}

})

.catch(e => {

if (typeof e.response.data == 'string') {

ToastAndroid.show(e.response.data, ToastAndroid.SHORT);

} else {

ToastAndroid.show(e.response.data.error, ToastAndroid.SHORT);

}

this.setState({fetchRequestLoader: false});

});

};

onChange = (event, date) => {

if (date === undefined) {

this.setState({showFromPicker: false, showToPicker: false});

}

if (this.state.showFromPicker) {

this.setState({

from\_date:

date.getFullYear() +

'-' +

(date.getMonth() + 1) +

'-' +

date.getDate(),

dobProvided: date,

showFromPicker: false,

});

}

if (this.state.showToPicker) {

this.setState({

to\_date:

date.getFullYear() +

'-' +

(date.getMonth() + 1) +

'-' +

date.getDate(),

dobProvided: date,

showToPicker: false,

});

}

};

searchTransaction = () => {

this.setState({searchLoader: true});

let params = '';

console.log('searchValue', this.state.valueTransaction);

if (this.state.valueSearch == 'email') {

params = `?receiverEmail=${this.state.searchValue}`;

} else if (this.state.valueSearch == 'phone') {

params = `?receiverPhone=${this.state.searchValue}`;

} else if (this.state.valueSearch == 'ssn') {

params = `?userId=${this.state.searchValue}`;

} else if (this.state.valueSearch == 'type') {

if (this.state.valueTransaction == 'credit\_card') {

params = `?type=credit`;

} else if (this.state.valueTransaction == 'debit\_card') {

params = `?type=debit`;

} else {

params = `?type=upi`;

}

}

axios

.post(

`https://harshal-trasactions-project.herokuapp.com/user/${this.props.id}/transactions${params}`,

this.state.valueSearch == 'time'

? {fromDate: this.state.from\_date, toDate: this.state.to\_date}

: {},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

},

)

.then(response => {

this.setState({searchLoader: false, searchData: response.data});

console.log('response request', response.data);

if (typeof response.data == 'string') {

ToastAndroid.show(response.data, ToastAndroid.SHORT);

} else {

ToastAndroid.show('Fetched data', ToastAndroid.SHORT);

}

})

.catch(e => {

if (typeof e.response.data == 'string') {

ToastAndroid.show(e.response.data, ToastAndroid.SHORT);

} else {

ToastAndroid.show(e.response.data.error, ToastAndroid.SHORT);

}

this.setState({searchLoader: false});

});

};

render() {

const {open, value, items} = this.state;

const {openSearch, valueSearch, itemsSearch} = this.state;

const {transactionOpen, valueTransaction, transactionItems} = this.state;

console.log('statementData', this.state.statementData);

return (

<View style={styles.container}>

{this.renderModal()}

<Text style={{fontSize: 20, color: '#fff', paddingTop: 15}}>

Main Menu

</Text>

<Text style={{color: '#fff'}}>

Name : {this.state.userData && this.state.userData.name}

</Text>

<Text style={{color: '#fff'}}>

Email : {this.state.userData && this.state.userData.email}

</Text>

<Text style={{color: '#fff'}}>

Phone : {this.state.userData && this.state.userData.phone}

</Text>

<Text style={{color: '#fff'}}>

Balance : {this.state.userData && this.state.userData.balance}

</Text>

<View style={styles.buttonContainer}>

<TouchableOpacity

onPress={() => {

this.fetchRequests();

}}>

{this.state.fetchRequestLoader ? (

<ActivityIndicator color={'silver'} size="small" />

) : (

<Text style={{color: '#fff', textAlign: 'center'}}>

Check Requests

</Text>

)}

</TouchableOpacity>

</View>

<Text style={{fontSize: 17, color: '#fff', paddingTop: 15}}>

Send/Request Money

</Text>

<View

style={{

flexDirection: 'row',

justifyContent: 'space-between',

paddingVertical: 10,

alignItems: 'center',

}}>

<Text style={{color: '#fff', marginRight: 20, width: '40%'}}>

Amount:

</Text>

<TextInput

placeholder="Enter Amount"

style={[

{

width: '50%',

marginVertical: 10,

justifyContent: 'center',

alignItems: 'center',

borderWidth: 1,

color: '#000',

backgroundColor: '#fff',

},

]}

onChangeText={text => this.handleOnchange('amount', text)}

value={this.state.amount}

/>

</View>

<View

style={{

flexDirection: 'row',

justifyContent: 'space-between',

paddingVertical: 10,

alignItems: 'center',

}}>

<Text style={{color: '#fff', marginRight: 20, width: '40%'}}>

Select Identifier:

</Text>

<DropDownPicker

open={this.state.openIdentifier}

value={this.state.valueIdentifier}

items={this.state.Identifiers}

setOpen={this.setIdentifierOpen}

setValue={this.setValueIdentifier}

setItems={this.setIdentifierItem}

containerProps={{zIndex: 99999999999999999, width: '50%'}}

listMode="SCROLLVIEW"

scrollViewProps={{

nestedScrollEnabled: true,

}}

/>

</View>

{this.state.valueIdentifier ? (

<View

style={{

flexDirection: 'row',

justifyContent: 'space-between',

paddingVertical: 10,

alignItems: 'center',

}}>

<Text style={{color: '#fff', marginRight: 20, width: '40%'}}>

Enter {this.state.valueIdentifier}

</Text>

<TextInput

placeholder={`Enter ${this.state.valueIdentifier}`}

style={[

{

width: '50%',

marginVertical: 10,

justifyContent: 'center',

alignItems: 'center',

borderWidth: 1,

color: '#000',

backgroundColor: '#fff',

},

]}

onChangeText={text => this.handleOnchange('email\_phone', text)}

value={this.state.email\_phone}

/>

</View>

) : null}

<View

style={{

flexDirection: 'row',

justifyContent: 'space-between',

paddingVertical: 10,

alignItems: 'center',

}}>

<Text style={{color: '#fff', marginRight: 20, width: '40%'}}>

Comment

</Text>

<TextInput

placeholder={`Enter Comment`}

style={[

{

width: '50%',

marginVertical: 10,

justifyContent: 'center',

alignItems: 'center',

borderWidth: 1,

color: '#000',

backgroundColor: '#fff',

},

]}

onChangeText={text => this.handleOnchange('comment', text)}

value={this.state.comment}

/>

</View>

<View

style={{

flexDirection: 'row',

justifyContent: 'space-between',

paddingVertical: 10,

alignItems: 'center',

}}>

<Text style={{color: '#fff', marginRight: 20, width: '40%'}}>

Transaction Mode:

</Text>

<DropDownPicker

open={transactionOpen}

value={valueTransaction}

items={transactionItems}

setOpen={this.setTransactionOpen}

setValue={this.setValueTransaction}

setItems={this.setTransactionItem}

containerProps={{zIndex: 9999999, width: '50%'}}

listMode="SCROLLVIEW"

scrollViewProps={{

nestedScrollEnabled: true,

}}

/>

</View>

<View

style={{

flexDirection: 'row',

justifyContent: 'space-around',

width: '100%',

}}>

<View style={styles.buttonContainer}>

<TouchableOpacity

onPress={() => {

this.sendMoney();

}}>

{this.state.sendMoneyLoader ? (

<ActivityIndicator color={'silver'} size="small" />

) : (

<Text style={{color: '#fff', textAlign: 'center'}}>Send</Text>

)}

</TouchableOpacity>

</View>

<View style={styles.buttonContainer}>

<TouchableOpacity

onPress={() => {

this.requestMoney();

}}>

{this.state.requestMoneyLoader ? (

<ActivityIndicator color={'silver'} size="small" />

) : (

<Text style={{color: '#fff', textAlign: 'center'}}>

Request

</Text>

)}

</TouchableOpacity>

</View>

</View>

<Text style={{fontSize: 17, color: '#fff', paddingVertical: 15}}>

Statements

</Text>

<View style={{flexDirection: 'row', alignItems: 'center'}}>

<DropDownPicker

open={open}

value={value}

items={items}

setOpen={this.setOpen}

setValue={this.setValue}

setItems={this.setItems}

containerProps={{}}

listMode="SCROLLVIEW"

scrollViewProps={{

nestedScrollEnabled: true,

}}

/>

</View>

<View

style={[

styles.buttonContainer,

{alignSelf: 'center', marginTop: 25},

]}>

<TouchableOpacity

onPress={() => {

this.getStatement();

}}>

{this.state.statementLoader ? (

<ActivityIndicator color={'silver'} size="small" />

) : (

<Text style={{color: '#fff', textAlign: 'center'}}>

Get Statement

</Text>

)}

</TouchableOpacity>

</View>

<View style={{flexDirection: 'row'}}>

{this.state.statementData &&

Object.keys(this.state.statementData) &&

Object.keys(this.state.statementData[0]).map(a => (

<Text

style={{

color: '#fff',

width: '15.2%',

textAlign: 'left',

paddingHorizontal: 2,

fontSize: 10,

borderWidth: 0.2,

borderColor: '#fff',

}}>

{a}

</Text>

))}

</View>

{this.state.statementData &&

this.state.statementData.map(a => (

<View style={{flexDirection: 'row'}}>

<Text

style={{

color: '#fff',

width: '15.2%',

textAlign: 'left',

paddingHorizontal: 2,

fontSize: 10,

borderWidth: 0.2,

borderColor: '#fff',

}}>

{a.transaction\_id}

</Text>

<Text

style={{

color: '#fff',

width: '15.2%',

textAlign: 'left',

paddingHorizontal: 2,

fontSize: 10,

borderWidth: 0.2,

borderColor: '#fff',

}}>

{a.account\_id}

</Text>

<Text

style={{

color: '#fff',

width: '15.2%',

textAlign: 'left',

paddingHorizontal: 2,

fontSize: 10,

borderWidth: 0.2,

borderColor: '#fff',

}}>

{a.name}

</Text>

<Text

style={{

color: '#fff',

width: '15.2%',

textAlign: 'left',

paddingHorizontal: 2,

fontSize: 10,

borderWidth: 0.2,

borderColor: '#fff',

}}>

{a.type}

</Text>

<Text

style={{

color: '#fff',

width: '15.2%',

textAlign: 'left',

paddingHorizontal: 2,

fontSize: 10,

borderWidth: 0.2,

borderColor: '#fff',

}}>

{a.amount}

</Text>

<Text

style={{

color: '#fff',

width: '15.2%',

textAlign: 'left',

paddingHorizontal: 2,

fontSize: 10,

borderWidth: 0.2,

borderColor: '#fff',

}}>

{moment(a.transaction\_date).format('MMMM Do YYYY')}

</Text>

<Text

style={{

color: '#fff',

width: '15.2%',

textAlign: 'left',

paddingHorizontal: 2,

fontSize: 10,

borderWidth: 0.2,

borderColor: '#fff',

}}>

{a.mode}

</Text>

</View>

))}

<Text style={{fontSize: 17, color: '#fff', paddingVertical: 15}}>

Search Transactions

</Text>

<View

style={{

flexDirection: 'row',

justifyContent: 'space-between',

paddingVertical: 10,

alignItems: 'center',

}}>

<Text style={{color: '#fff', marginRight: 20, width: '40%'}}>

Type of Search

</Text>

<DropDownPicker

open={openSearch}

value={valueSearch}

items={itemsSearch}

setOpen={this.setOpenSearch}

setValue={this.setValueSearch}

setItems={this.setItemsSearch}

containerProps={{

width: '50%',

}}

listMode="SCROLLVIEW"

scrollViewProps={{

nestedScrollEnabled: true,

}}

/>

</View>

{this.state.valueSearch !== 'type' &&

this.state.valueSearch !== 'time' &&

this.state.valueSearch !== undefined ? (

<View

style={{

flexDirection: 'row',

justifyContent: 'space-between',

paddingVertical: 10,

alignItems: 'center',

}}>

<Text style={{color: '#fff', marginRight: 20, width: '40%'}}>

Enter {valueSearch}

</Text>

<TextInput

placeholder={`Enter ${valueSearch}`}

style={[

{

width: '50%',

justifyContent: 'center',

alignItems: 'center',

borderWidth: 1,

color: '#000',

backgroundColor: '#fff',

},

]}

onChangeText={text => this.handleOnchange('searchValue', text)}

value={this.state.searchValue}

/>

</View>

) : null}

{this.state.showFromPicker || this.state.showToPicker ? (

<DateTimePicker

testID="dateTimePicker"

value={new Date()}

maximumDate={new Date()}

mode={'date'}

is24Hour={true}

display="default"

onChange={this.onChange}

/>

) : null}

{this.state.valueSearch == 'time' ? (

<View

style={{

flexDirection: 'row',

justifyContent: 'space-around',

width: '100%',

}}>

<View style={{width: '40%'}}>

<Text style={{color: '#fff'}}>From Date</Text>

<TextInput

style={[

{

width: '100%',

marginVertical: 10,

justifyContent: 'center',

alignItems: 'center',

borderWidth: 1,

color: '#000',

backgroundColor: '#fff',

},

]}

onFocus={() => this.setState({showFromPicker: true})}

// onChangeText={text => this.handleOnchange('from\_date', text)}

value={this.state.from\_date}

/>

</View>

<View style={{width: '40%'}}>

<Text style={{color: '#fff'}}>To Date</Text>

<TextInput

style={[

{

width: '100%',

marginVertical: 10,

justifyContent: 'center',

alignItems: 'center',

borderWidth: 1,

color: '#000',

backgroundColor: '#fff',

},

]}

onFocus={() => this.setState({showToPicker: true})}

// onChangeText={text => this.handleOnchange('to\_date', text)}

value={this.state.to\_date}

/>

</View>

</View>

) : null}

{this.state.valueSearch == 'type' ? (

<View

style={{

flexDirection: 'row',

justifyContent: 'space-between',

paddingVertical: 10,

alignItems: 'center',

}}>

<Text style={{color: '#fff', marginRight: 20, width: '40%'}}>

Transaction Mode:

</Text>

<DropDownPicker

open={transactionOpen}

value={valueTransaction}

items={transactionItems}

setOpen={this.setTransactionOpen}

setValue={this.setValueTransaction}

setItems={this.setTransactionItem}

containerProps={{zIndex: 9999999, width: '50%'}}

listMode="SCROLLVIEW"

scrollViewProps={{

nestedScrollEnabled: true,

}}

/>

</View>

) : null}

{this.state.valueSearch ? (

<View style={[styles.buttonContainer, {alignSelf: 'center'}]}>

<TouchableOpacity

onPress={() => {

this.searchTransaction();

}}>

{this.state.searchLoader ? (

<ActivityIndicator color={'silver'} size="small" />

) : (

<Text style={{color: '#fff', textAlign: 'center'}}>Search</Text>

)}

</TouchableOpacity>

</View>

) : null}

<View style={{flexDirection: 'row'}}>

{this.state.searchData &&

Object.keys(this.state.searchData) &&

Object.keys(this.state.searchData[0]).map(a => (

<Text

style={{

color: '#fff',

width: '15.2%',

textAlign: 'left',

paddingHorizontal: 2,

fontSize: 10,

borderWidth: 0.2,

borderColor: '#fff',

}}>

{a}

</Text>

))}

</View>

{this.state.searchData &&

this.state.searchData.map(a => (

<View style={{flexDirection: 'row'}}>

<Text

style={{

color: '#fff',

width: '15.2%',

textAlign: 'left',

paddingHorizontal: 2,

fontSize: 10,

borderWidth: 0.2,

borderColor: '#fff',

}}>

{a.transaction\_id}

</Text>

<Text

style={{

color: '#fff',

width: '15.2%',

textAlign: 'left',

paddingHorizontal: 2,

fontSize: 10,

borderWidth: 0.2,

borderColor: '#fff',

}}>

{a.account\_id}

</Text>

<Text

style={{

color: '#fff',

width: '15.2%',

textAlign: 'left',

paddingHorizontal: 2,

fontSize: 10,

borderWidth: 0.2,

borderColor: '#fff',

}}>

{a.name}

</Text>

<Text

style={{

color: '#fff',

width: '15.2%',

textAlign: 'left',

paddingHorizontal: 2,

fontSize: 10,

borderWidth: 0.2,

borderColor: '#fff',

}}>

{a.type}

</Text>

<Text

style={{

color: '#fff',

width: '15.2%',

textAlign: 'left',

paddingHorizontal: 2,

fontSize: 10,

borderWidth: 0.2,

borderColor: '#fff',

}}>

{a.amount}

</Text>

<Text

style={{

color: '#fff',

width: '15.2%',

textAlign: 'left',

paddingHorizontal: 2,

fontSize: 10,

borderWidth: 0.2,

borderColor: '#fff',

}}>

{moment(a.transaction\_date).format('MMMM Do YYYY')}

</Text>

<Text

style={{

color: '#fff',

width: '15.2%',

textAlign: 'left',

paddingHorizontal: 2,

fontSize: 10,

borderWidth: 0.2,

borderColor: '#fff',

}}>

{a.mode}

</Text>

</View>

))}

<View style={[styles.buttonContainer, {alignSelf: 'center'}]}>

<TouchableOpacity onPress={() => this.props.handleLogout()}>

<Text style={{color: '#fff', textAlign: 'center'}}>Signout</Text>

</TouchableOpacity>

</View>

</View>

);

}

}

const styles = StyleSheet.create({

container: {

flex: 1,

alignItems: 'flex-start',

justifyContent: 'center',

paddingHorizontal: 15,

},

textInputStyle: {

borderWidth: 1,

width: '40%',

},

buttonContainer: {

backgroundColor: '#000',

color: '#fff',

padding: 10,

margin: 8,

borderRadius: 5,

width: '40%',

},

});

7. Login.js

import {

View,

Text,

StyleSheet,

TextInput,

TouchableOpacity,

ActivityIndicator,

ToastAndroid,

} from 'react-native';

import React, {Component} from 'react';

// import {LOGIN} from '../Routes';

var axios = require('axios');

export default class Login extends React.Component {

constructor(props) {

super(props);

this.state = {

email: '',

password: '',

};

}

componentDidMount() {}

handleOnchange = (type, data) => {

this.setState({[type]: data});

};

handleLogin = async () => {

this.setState({submitLogin: true});

axios

.post(

`https://harshal-trasactions-project.herokuapp.com/user/login`,

{

email: this.state.email,

password: this.state.password,

},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

},

)

.then(response => {

this.props.handleLOgin(response.data);

this.setState({submitLogin: false});

})

.catch(e => {

console.log('e', e.response.data);

if (typeof e.response.data == 'string') {

ToastAndroid.show(e.response.data, ToastAndroid.SHORT);

} else {

ToastAndroid.show(e.response.data.error, ToastAndroid.SHORT);

}

this.setState({submitLogin: false});

});

};

render() {

return (

<View style={styles.container}>

<View

style={{

marginHorizontal: 10,

backgroundColor: '#202124',

alignItems: 'center',

alignContent: 'center',

}}>

<Text

style={{

fontSize: 20,

color: '#fff',

paddingTop: 15,

width: '100%',

textAlign: 'center',

}}>

Login

</Text>

<View

style={{

flexDirection: 'row',

justifyContent: 'center',

paddingVertical: 10,

width: '100%',

alignItems: 'center',

}}>

<Text style={{color: '#fff', marginRight: 20, width: '22%'}}>

Email:

</Text>

<TextInput

placeholder="Enter Email"

style={[

{

width: '50%',

marginVertical: 10,

justifyContent: 'center',

alignItems: 'center',

borderWidth: 1,

color: '#000',

backgroundColor: '#fff',

},

]}

onChangeText={text => this.handleOnchange('email', text)}

value={this.state.email}

/>

</View>

<View

style={{

flexDirection: 'row',

justifyContent: 'center',

paddingVertical: 10,

width: '100%',

alignItems: 'center',

}}>

<Text style={{color: '#fff', marginRight: 20, width: '22%'}}>

Password:

</Text>

<TextInput

placeholder="Enter Password"

style={[

{

width: '50%',

marginVertical: 10,

justifyContent: 'center',

alignItems: 'center',

borderWidth: 1,

color: '#000',

backgroundColor: '#fff',

},

]}

onChangeText={text => this.handleOnchange('password', text)}

value={this.state.password}

/>

</View>

<View style={styles.buttonContainer}>

<TouchableOpacity

onPress={() => {

this.handleLogin();

}}>

{this.state.submitLogin ? (

<ActivityIndicator color={'silver'} size="small" />

) : (

<Text style={{color: '#fff', textAlign: 'center'}}>LOGIN</Text>

)}

</TouchableOpacity>

</View>

</View>

</View>

);

}

}

const styles = StyleSheet.create({

container: {

justifyContent: 'center',

marginTop: '50%',

paddingHorizontal: 15,

},

textInputStyle: {

borderWidth: 1,

width: '40%',

},

buttonContainer: {

alignSelf: 'center',

width: '40%',

backgroundColor: '#000',

color: '#fff',

padding: 10,

margin: 8,

borderRadius: 5,

},

});

8. Statement.js

import DateTimePicker from '@react-native-community/datetimepicker';

import {

View,

Text,

StyleSheet,

TextInput,

TouchableOpacity,

ToastAndroid,

ActivityIndicator,

} from 'react-native';

import React, {Component} from 'react';

import DropDownPicker from 'react-native-dropdown-picker';

// import LQDropdown from '../components/LQDropdown';

var axios = require('axios');

export default class HomeScreen extends React.Component {

constructor(props) {

super(props);

this.state = {

userData: null,

text: '',

open: false,

value: null,

from\_date: '',

to\_date: '',

amountReceived: null,

amountSent: null,

totalAvgObjPerMonth: null,

maxAmountObjPerMonth: null,

bestUsers: null,

items: [

{label: 'January', value: '1'},

{label: 'February', value: '2'},

{label: 'March', value: '3'},

{label: 'April', value: '4'},

{label: 'May', value: '5'},

{label: 'June', value: '6'},

{label: 'July', value: '7'},

{label: 'August', value: '8'},

{label: 'September', value: '9'},

{label: 'October', value: '10'},

{label: 'November', value: '11'},

{label: 'December', value: '12'},

],

selectedMonth: null,

};

}

componentDidMount() {

axios

.get(`https://harshal-trasactions-project.herokuapp.com/get-best-users`, {

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

})

.then(response => {

console.log('response', response.data);

this.setState({

bestUsers: response.data,

});

})

.catch(e => console.log('op', e));

}

handleClassSelection = value => {

console.log('value', value);

this.setState({

selectedMonth: value,

});

};

onChange = (event, date) => {

if (date === undefined) {

this.setState({showFromPicker: false, showToPicker: false});

}

if (this.state.showFromPicker) {

this.setState({

from\_date:

date.getFullYear() +

'-' +

(date.getMonth() + 1) +

'-' +

date.getDate(),

dobProvided: date,

showFromPicker: false,

});

}

if (this.state.showToPicker) {

this.setState({

to\_date:

date.getFullYear() +

'-' +

(date.getMonth() + 1) +

'-' +

date.getDate(),

dobProvided: date,

showToPicker: false,

});

}

};

setOpen = open => {

console.log('open', open);

this.setState({

open,

});

};

setValue = callback => {

this.setState(state => ({

value: callback(state.value),

}));

};

setItems = callback => {

this.setState(state => ({

items: callback(state.items),

}));

};

handleOnchange = data => {

console.log('change', data);

this.setState({[type]: data});

};

getTotalAmountInDate = () => {

this.setState({getTotalLoader: true});

axios

.post(

`https://harshal-trasactions-project.herokuapp.com/user/${this.props.id}/get-total-amount`,

{

fromDate: this.state.from\_date,

toDate: this.state.to\_date,

},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

},

)

.then(response => {

console.log('response', response.data);

ToastAndroid.show('amount fetched', ToastAndroid.SHORT);

this.setState({getTotalLoader: false});

this.setState({

amountReceived: response.data && response.data.amount\_received,

amountSent: response.data && response.data.amount\_sent,

});

})

.catch(e => {

if (typeof e.response.data == 'string') {

ToastAndroid.show(e.response.data, ToastAndroid.SHORT);

} else {

ToastAndroid.show(e.response.data.error, ToastAndroid.SHORT);

}

this.setState({getTotalLoader: false});

});

};

getTotalAmountPerMonth = () => {

this.setState({fetchtingTotalAvg: true});

axios

.post(

`https://harshal-trasactions-project.herokuapp.com/user/${this.props.id}/amount-per-month`,

{

month: this.state.value,

},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

},

)

.then(response => {

console.log('response', response.data);

this.setState({fetchtingTotalAvg: false});

ToastAndroid.show('Data Fetched', ToastAndroid.SHORT);

this.setState({

totalAvgObjPerMonth: response.data,

});

})

.catch(e => {

console.log('e', e.response.data);

if (typeof e.response.data == 'string') {

ToastAndroid.show(e.response.data, ToastAndroid.SHORT);

} else {

ToastAndroid.show(e.response.data.error, ToastAndroid.SHORT);

}

this.setState({fetchtingTotalAvg: false});

});

};

getMaxTransaction = () => {

this.setState({maxtransactionLoader: true});

axios

.post(

`https://harshal-trasactions-project.herokuapp.com/max-amount-for-month`,

{

month: this.state.value,

},

{

headers: {

Accept: 'application/json',

'Content-Type': 'application/json',

},

},

)

.then(response => {

console.log('response max amount', response.data);

this.setState({maxtransactionLoader: false});

ToastAndroid.show('Data Fetched', ToastAndroid.SHORT);

this.setState({

maxAmountObjPerMonth: response.data,

});

})

.catch(e => {

this.setState({maxtransactionLoader: false});

if (typeof e.response.data == 'string') {

ToastAndroid.show(e.response.data, ToastAndroid.SHORT);

} else {

ToastAndroid.show(e.response.data.error, ToastAndroid.SHORT);

}

});

};

render() {

console.log('this.state', this.state.maxAmountObjPerMonth);

const {open, value, items} = this.state;

return (

<View style={styles.container}>

<Text

style={{

fontSize: 18,

color: '#fff',

textAlign: 'center',

paddingVertical: 15,

}}>

Statement functions

</Text>

{this.state.showFromPicker || this.state.showToPicker ? (

<DateTimePicker

testID="dateTimePicker"

value={new Date()}

maximumDate={new Date()}

mode={'date'}

is24Hour={true}

display="default"

onChange={this.onChange}

/>

) : null}

<View

style={{

flexDirection: 'row',

justifyContent: 'space-around',

width: '100%',

}}>

<View style={{width: '40%'}}>

<Text style={{color: '#fff'}}>From Date</Text>

<TextInput

style={[

{

width: '100%',

marginVertical: 10,

justifyContent: 'center',

alignItems: 'center',

borderWidth: 1,

color: '#000',

backgroundColor: '#fff',

},

]}

onFocus={() => this.setState({showFromPicker: true})}

// onChangeText={text => this.handleOnchange('from\_date', text)}

value={this.state.from\_date}

/>

</View>

<View style={{width: '40%'}}>

<Text style={{color: '#fff'}}>To Date</Text>

<TextInput

style={[

{

width: '100%',

marginVertical: 10,

justifyContent: 'center',

alignItems: 'center',

borderWidth: 1,

color: '#000',

backgroundColor: '#fff',

},

]}

onFocus={() => this.setState({showToPicker: true})}

// onChangeText={text => this.handleOnchange('to\_date', text)}

value={this.state.to\_date}

/>

</View>

</View>

<View style={styles.buttonContainer}>

<TouchableOpacity

onPress={() => {

this.getTotalAmountInDate();

}}>

{this.state.getTotalLoader ? (

<ActivityIndicator color={'silver'} size="small" />

) : (

<Text style={{color: '#fff', textAlign: 'center'}}>

Get Total

</Text>

)}

</TouchableOpacity>

</View>

<View style={{color: '#fff', width: '100%'}}>

<View

style={{flexDirection: 'row', paddingVertical: 5, width: '100%'}}>

<Text style={{width: '50%', color: '#fff'}}>

Total Amount Received{' '}

</Text>

<Text style={{color: '#fff'}}>

{this.state.amountReceived ? this.state.amountReceived : '-'}

</Text>

</View>

<View

style={{flexDirection: 'row', paddingVertical: 5, width: '100%'}}>

<Text style={{width: '50%', color: '#fff'}}>

Total Amount Sent{' '}

</Text>

<Text style={{color: '#fff'}}>

{this.state.amountSent ? this.state.amountSent : '-'}

</Text>

</View>

</View>

<View

style={{

borderBottomColor: '#fff',

borderBottomWidth: 1,

width: '100%',

marginVertical: 20,

}}></View>

<DropDownPicker

open={open}

value={value}

items={items}

setOpen={this.setOpen}

setValue={this.setValue}

setItems={this.setItems}

containerProps={{

height: 100,

}}

listMode="SCROLLVIEW"

scrollViewProps={{

nestedScrollEnabled: true,

}}

/>

<View

style={{

flexDirection: 'row',

alignItems: 'center',

justifyContent: 'space-around',

width: '100%',

}}>

<View style={styles.buttonContainer}>

<TouchableOpacity

onPress={() => {

this.getTotalAmountPerMonth();

}}>

{this.state.fetchtingTotalAvg ? (

<ActivityIndicator color={'silver'} size="small" />

) : (

<Text style={{color: '#fff', textAlign: 'center'}}>

Get Total/Avg Per Month

</Text>

)}

</TouchableOpacity>

</View>

</View>

{this.state.totalAvgObjPerMonth ? (

<View>

<View

style={{flexDirection: 'row', paddingVertical: 5, width: '100%'}}>

<Text style={{width: '50%', color: '#fff'}}>

Avg amount received{' '}

</Text>

<Text style={{color: '#fff'}}>

{this.state.totalAvgObjPerMonth.average\_amount\_received}

</Text>

</View>

<View

style={{flexDirection: 'row', paddingVertical: 5, width: '100%'}}>

<Text style={{width: '50%', color: '#fff'}}>

Avg amount sent{' '}

</Text>

<Text style={{color: '#fff'}}>

{this.state.totalAvgObjPerMonth.average\_amount\_sent}

</Text>

</View>

<View

style={{flexDirection: 'row', paddingVertical: 5, width: '100%'}}>

<Text style={{width: '50%', color: '#fff'}}>

Total amount received{' '}

</Text>

<Text style={{color: '#fff'}}>

{this.state.totalAvgObjPerMonth.total\_amount\_received}

</Text>

</View>

<View

style={{flexDirection: 'row', paddingVertical: 5, width: '100%'}}>

<Text style={{width: '50%', color: '#fff'}}>

Total amount sent{' '}

</Text>

<Text style={{color: '#fff'}}>

{this.state.totalAvgObjPerMonth.total\_amount\_sent}{' '}

</Text>

</View>

</View>

) : null}

<View

style={{

flexDirection: 'row',

alignItems: 'center',

justifyContent: 'space-around',

width: '100%',

}}>

<View style={styles.buttonContainer}>

<TouchableOpacity

onPress={() => {

this.getMaxTransaction();

}}>

{this.state.maxtransactionLoader ? (

<ActivityIndicator color={'silver'} size="small" />

) : (

<Text style={{color: '#fff', textAlign: 'center'}}>

Get Max Transaction

</Text>

)}

</TouchableOpacity>

</View>

</View>

{this.state.maxAmountObjPerMonth &&

this.state.maxAmountObjPerMonth.map(a => (

<View>

<View

style={{

flexDirection: 'row',

paddingVertical: 5,

width: '100%',

}}>

<Text style={{width: '50%', color: '#fff'}}>Sender </Text>

<Text style={{color: '#fff'}}>{a.sender}</Text>

</View>

<View

style={{

flexDirection: 'row',

paddingVertical: 5,

width: '100%',

}}>

<Text style={{width: '50%', color: '#fff'}}>Receiver </Text>

<Text style={{color: '#fff'}}>{a.receiver}</Text>

</View>

<View

style={{

flexDirection: 'row',

paddingVertical: 5,

width: '100%',

}}>

<Text style={{width: '50%', color: '#fff'}}>Amount </Text>

<Text style={{color: '#fff'}}>{a.amount}</Text>

</View>

</View>

))}

<View

style={{

borderBottomColor: '#fff',

borderBottomWidth: 1,

width: '100%',

marginVertical: 20,

}}></View>

{/\* <Text

style={{

fontSize: 18,

color: '#fff',

textAlign: 'center',

paddingVertical: 15,

}}>

Max Transaction per month

</Text> \*/}

{/\* <DropDownPicker

open={open}

value={value}

items={items}

setOpen={this.setOpen}

setValue={this.setValue}

setItems={this.setItems}

containerProps={{

height: 100,

}}

/> \*/}

<Text

style={{

fontSize: 18,

color: '#fff',

textAlign: 'center',

paddingVertical: 15,

}}>

Best Users

</Text>

{this.state.bestUsers &&

this.state.bestUsers.map(a => (

<View

style={{

flexDirection: 'row',

justifyContent: 'space-around',

width: '70%',

paddingVertical: 10,

}}>

<Text style={{color: '#fff'}}>{a.name}</Text>

<Text style={{color: '#fff'}}>{a.total\_transactions\_amount}</Text>

</View>

))}

</View>

);

}

}

const styles = StyleSheet.create({

container: {

flex: 1,

alignItems: 'flex-start',

justifyContent: 'center',

paddingHorizontal: 15,

},

textInputStyle: {

borderWidth: 1,

width: '40%',

},

buttonContainer: {

backgroundColor: '#000',

color: '#fff',

padding: 10,

margin: 8,

borderRadius: 5,

alignSelf: 'center',

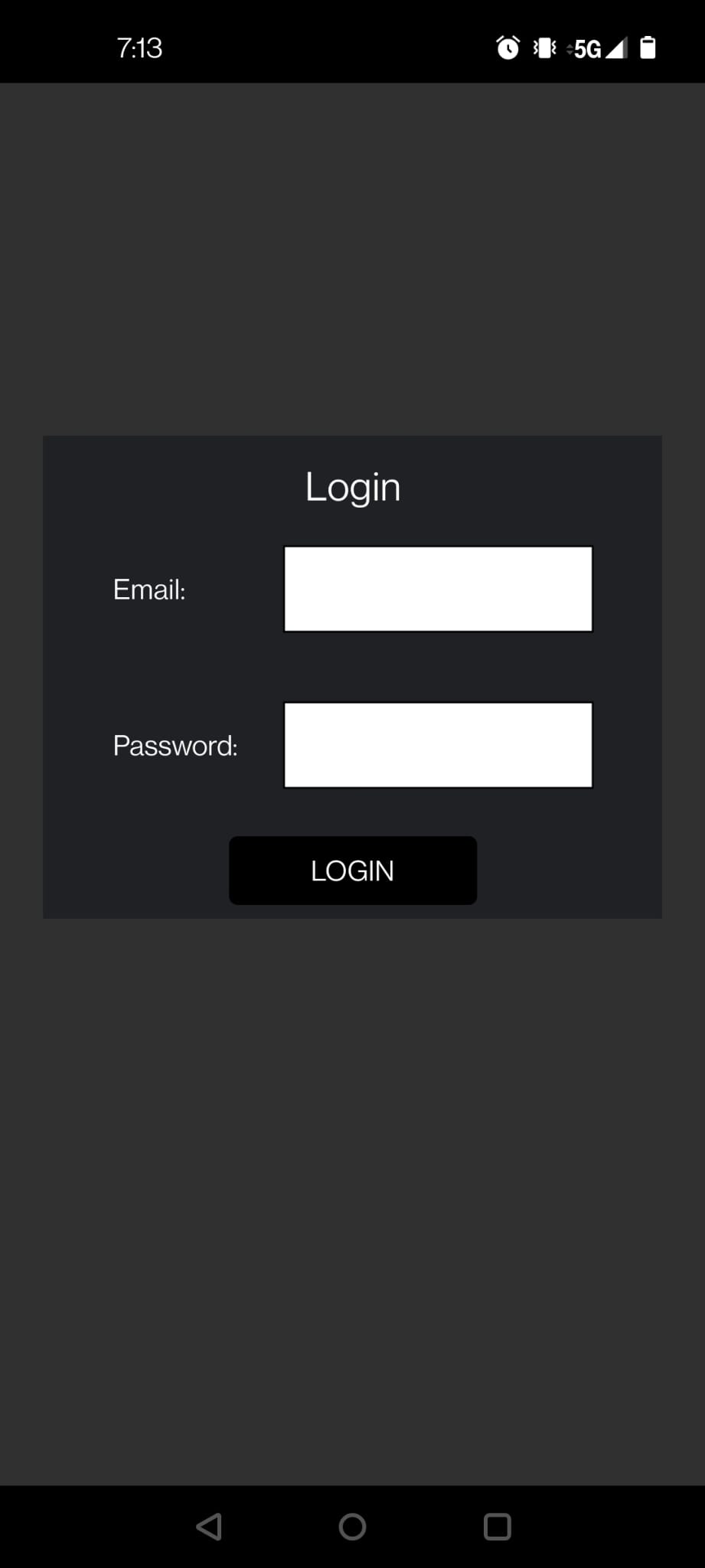
width: '55%',

},

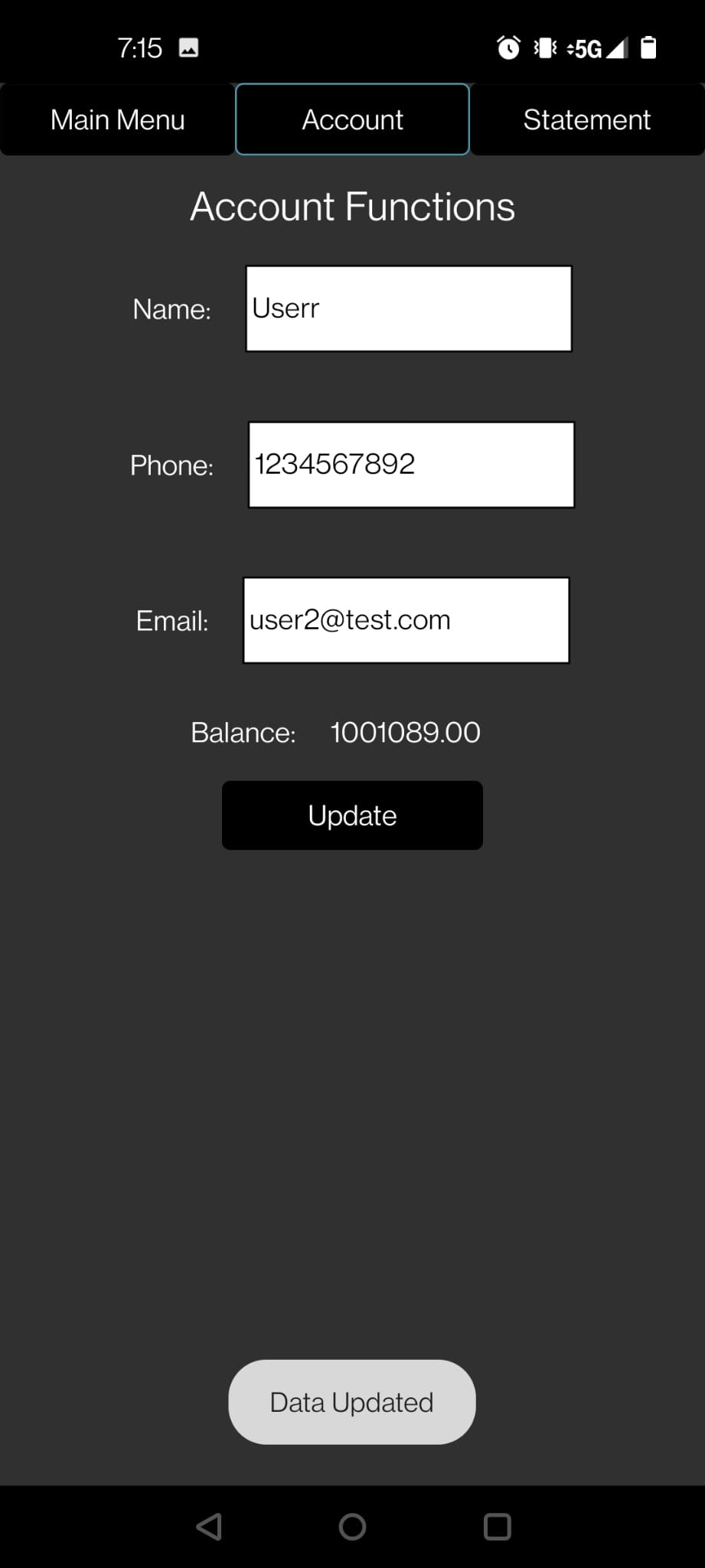
});

* **SCREENSHOTS:**

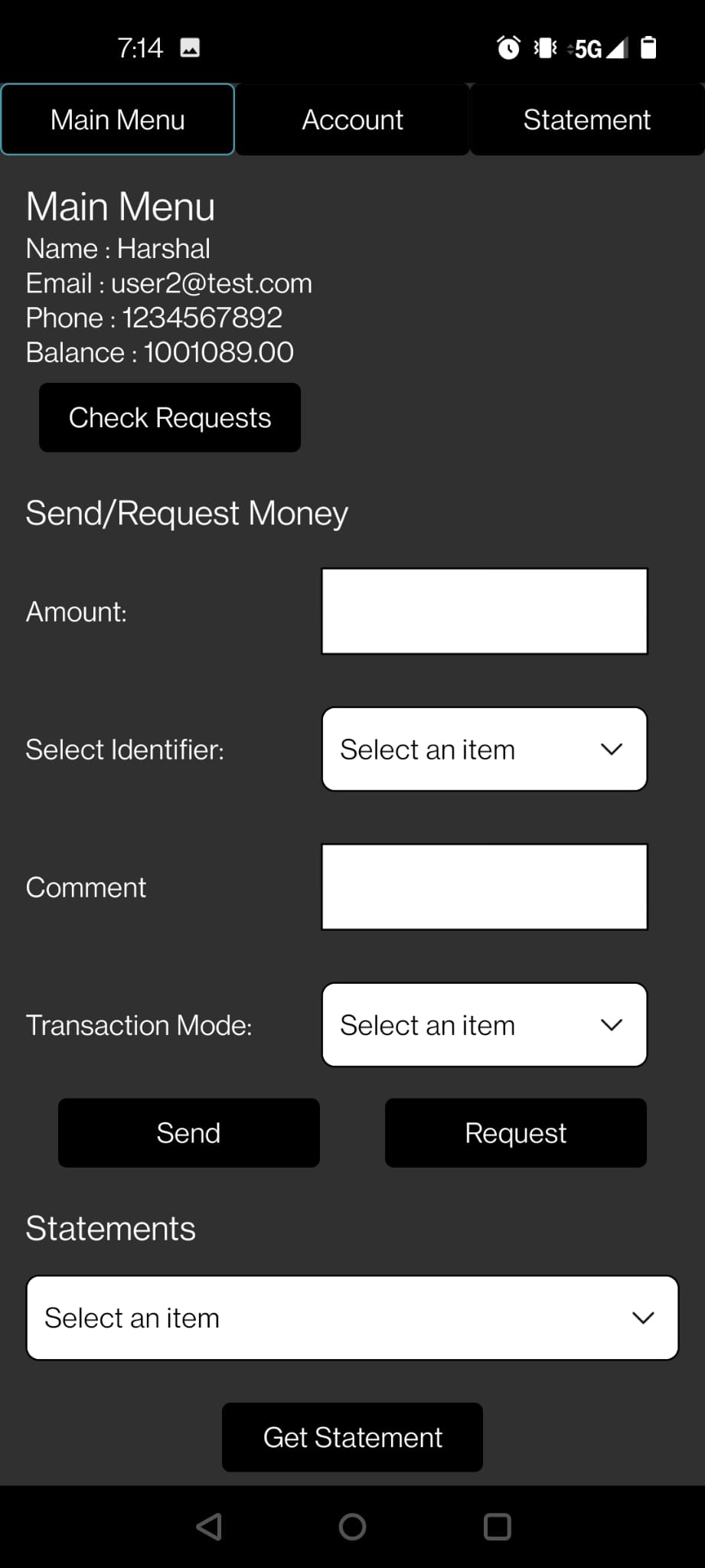
1. Login Page

****

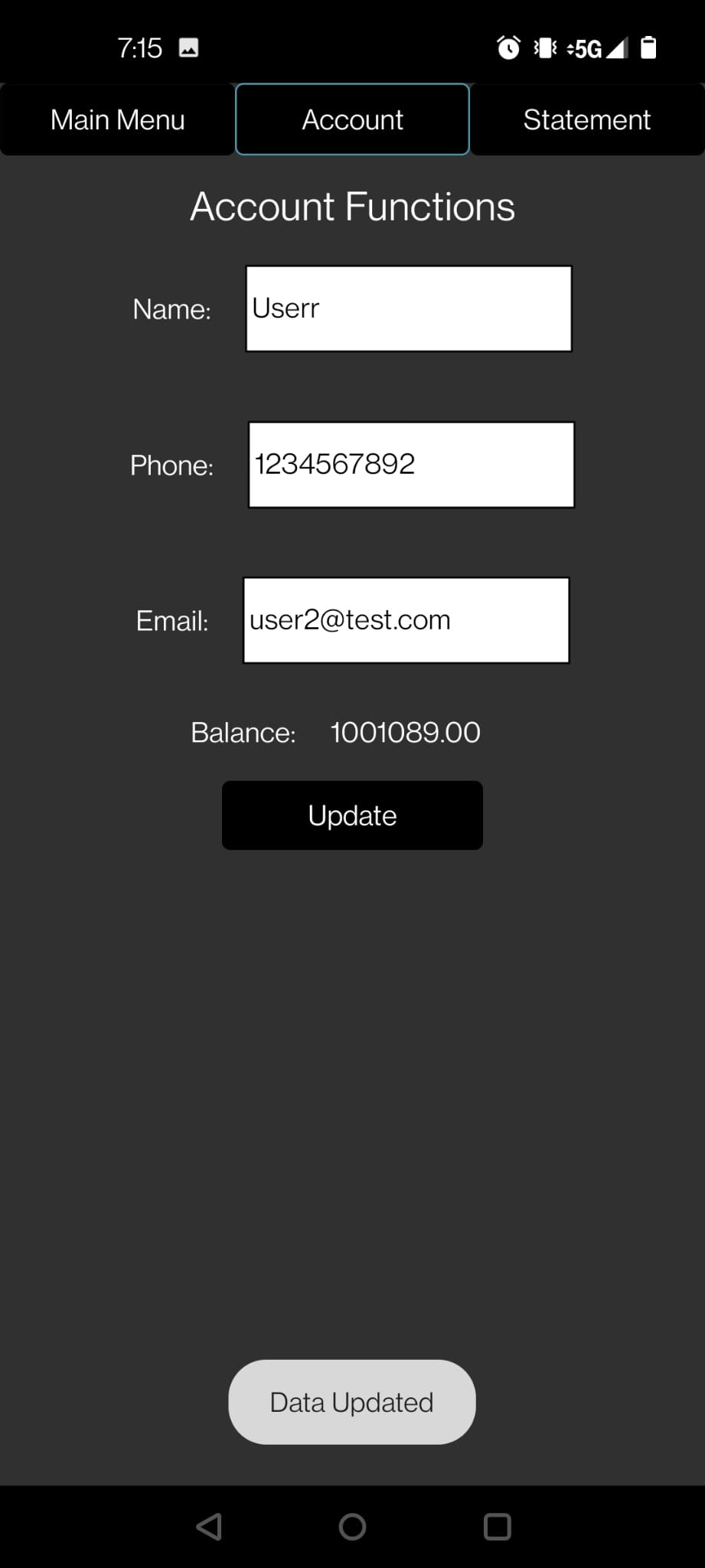
2. User Authentication



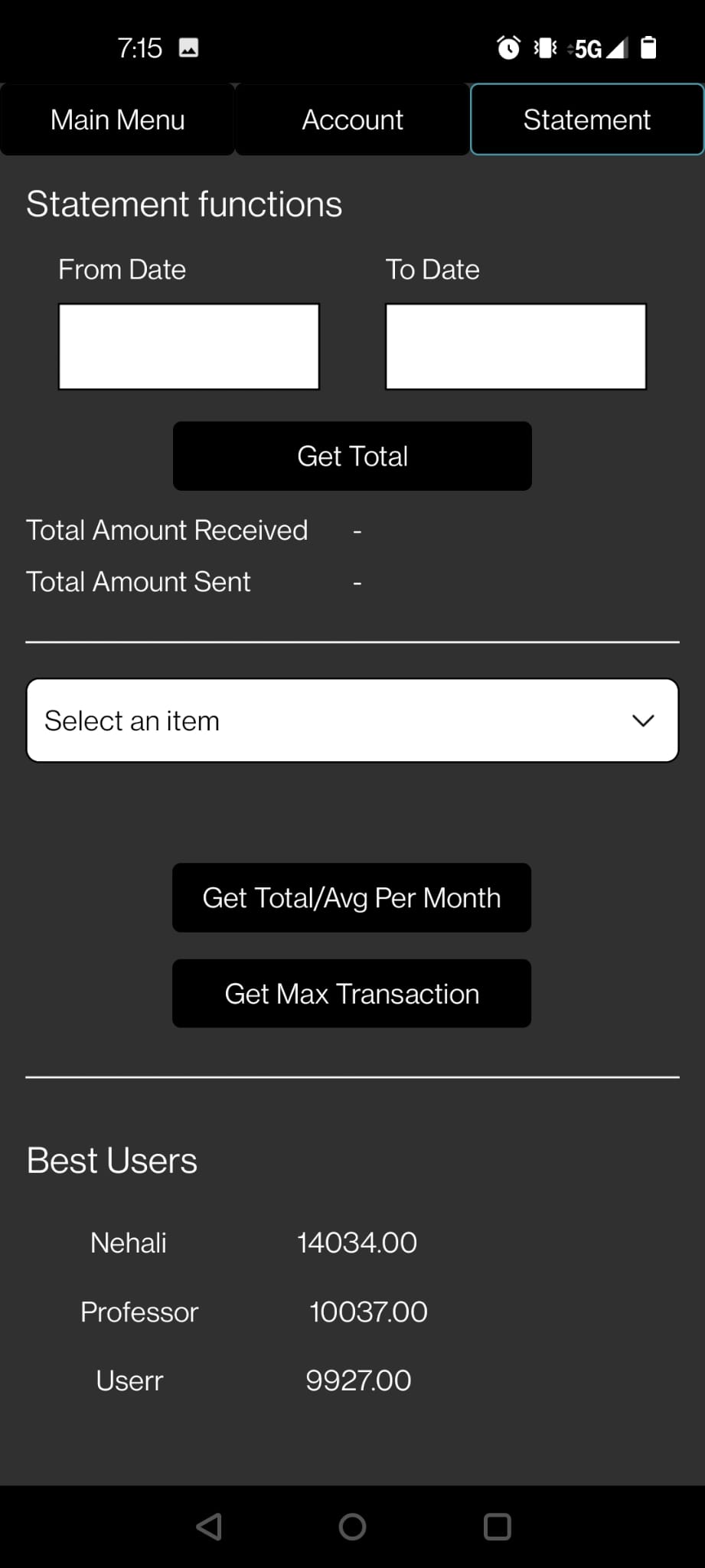
3. Main Menu



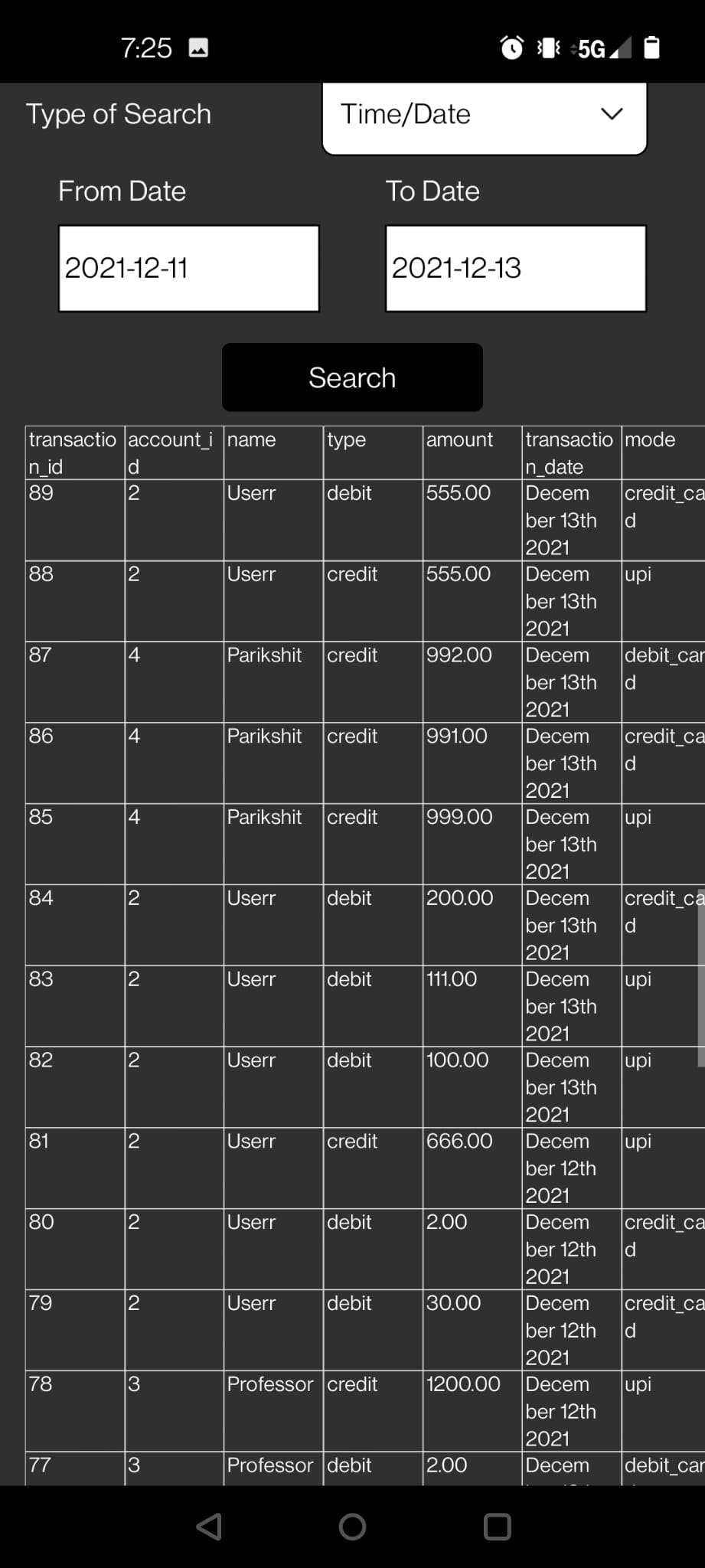
4. Update user information



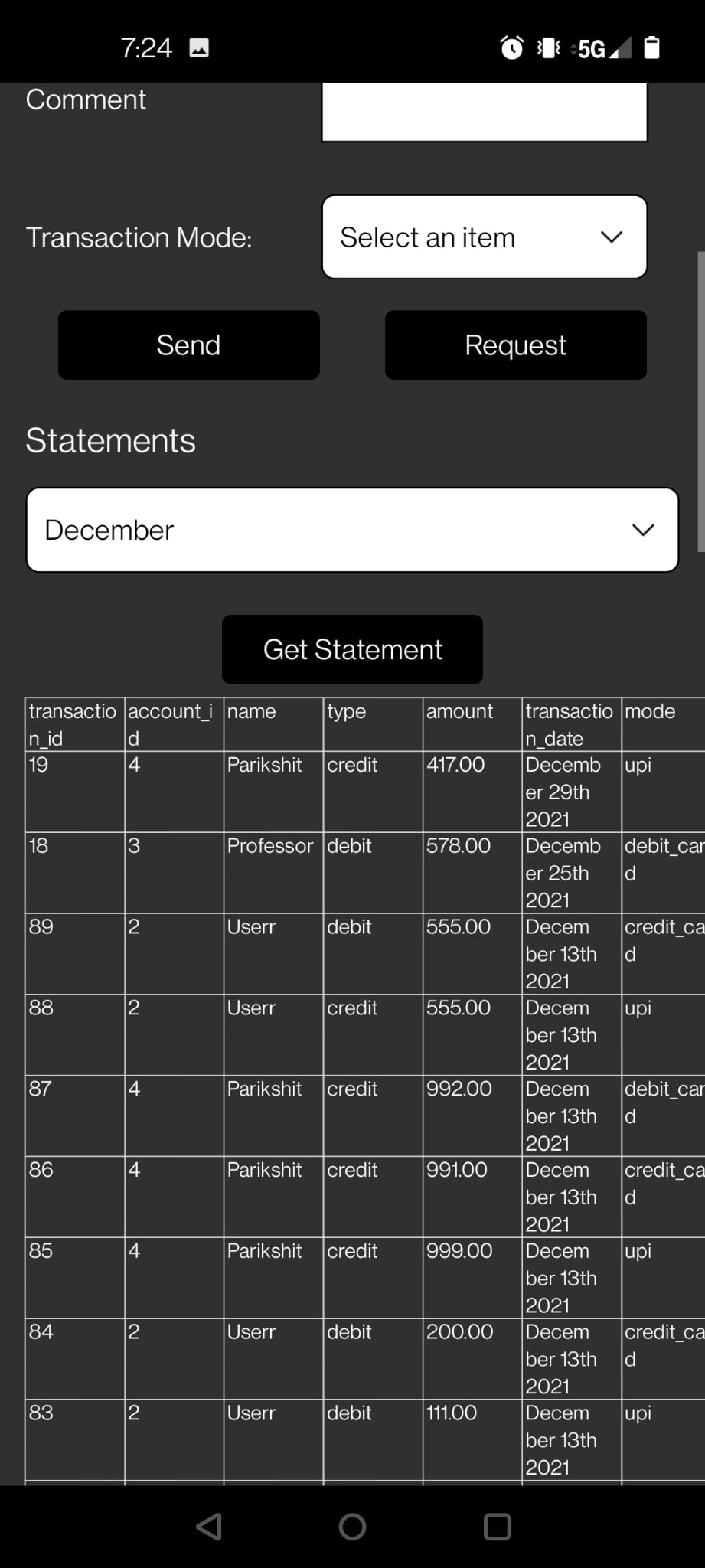
5. Statement Tab



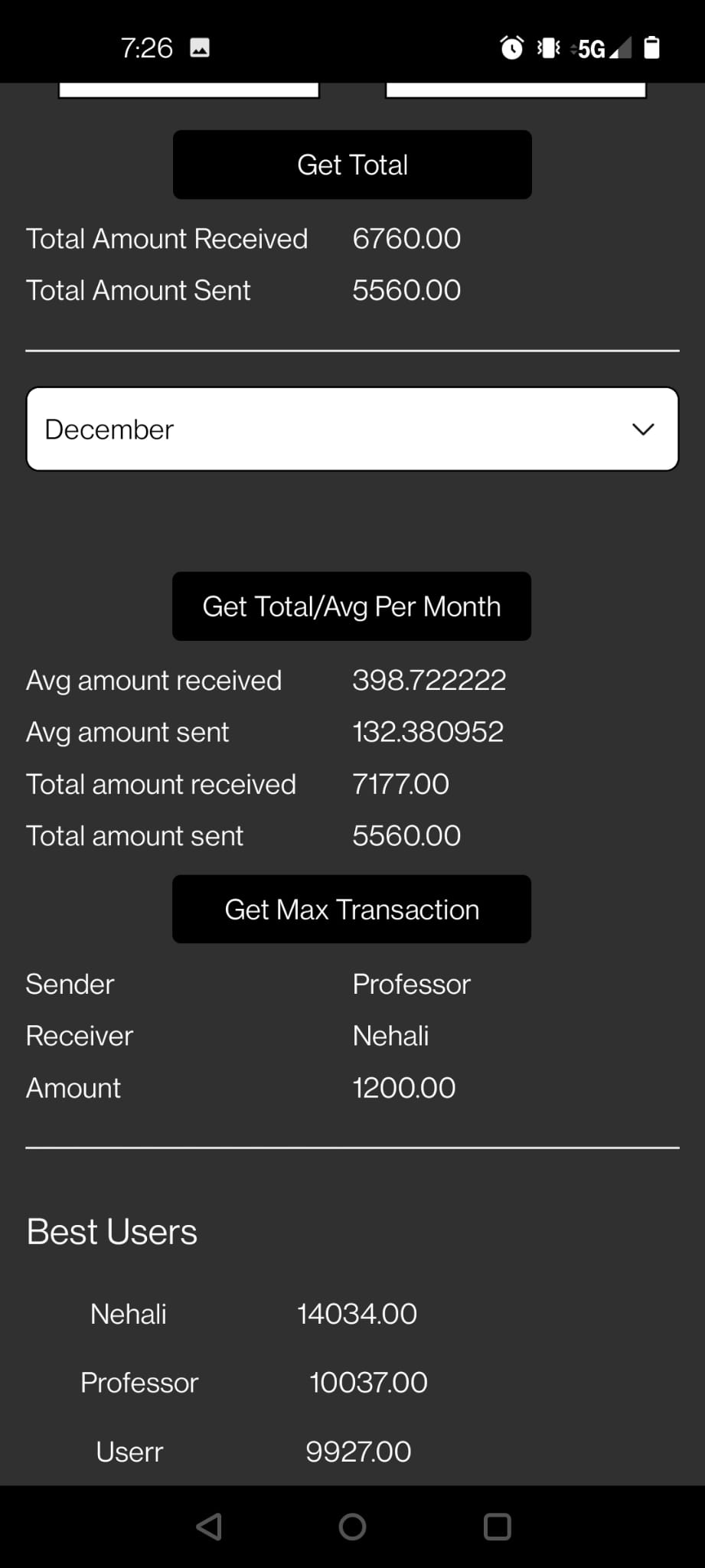
6. Filtering Statements by Date/Time



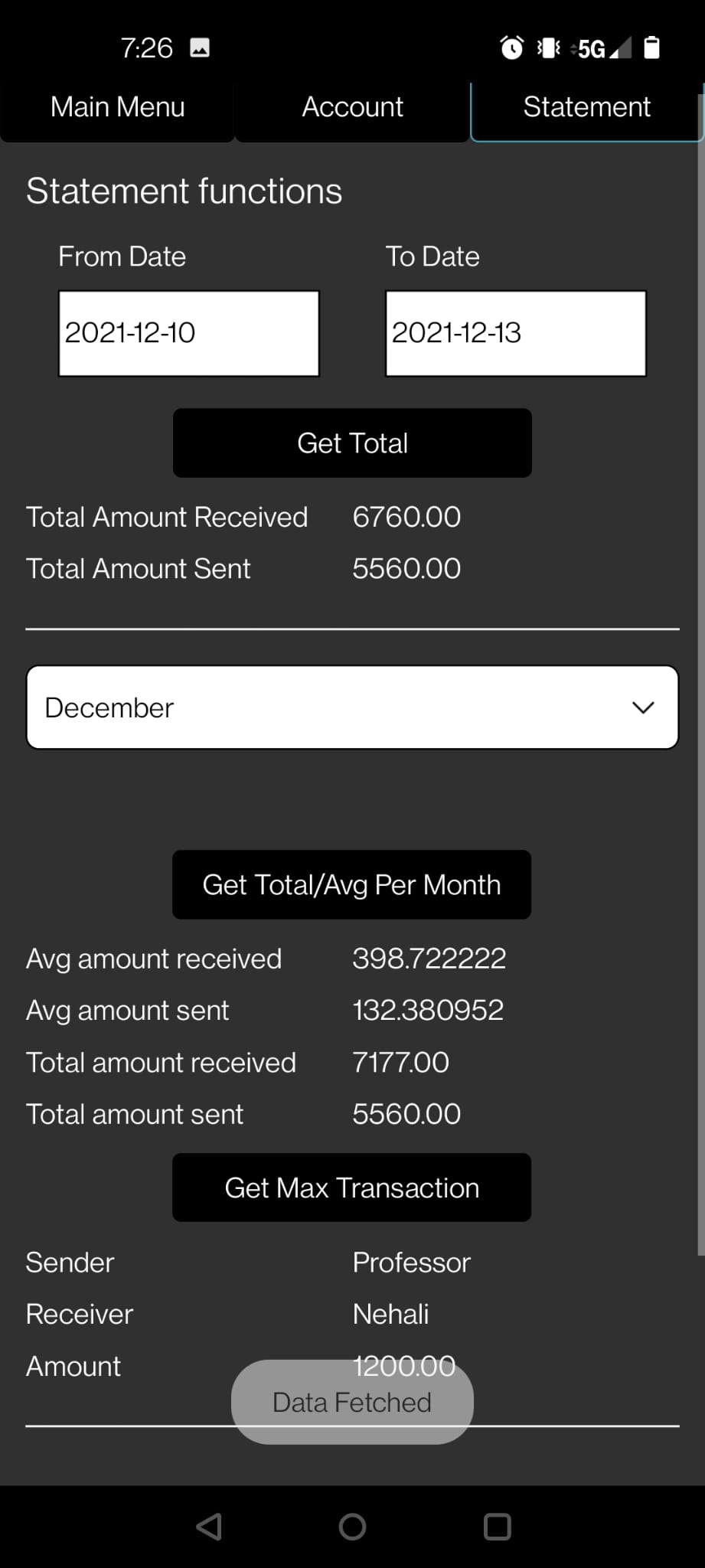
7. Monthly Statement



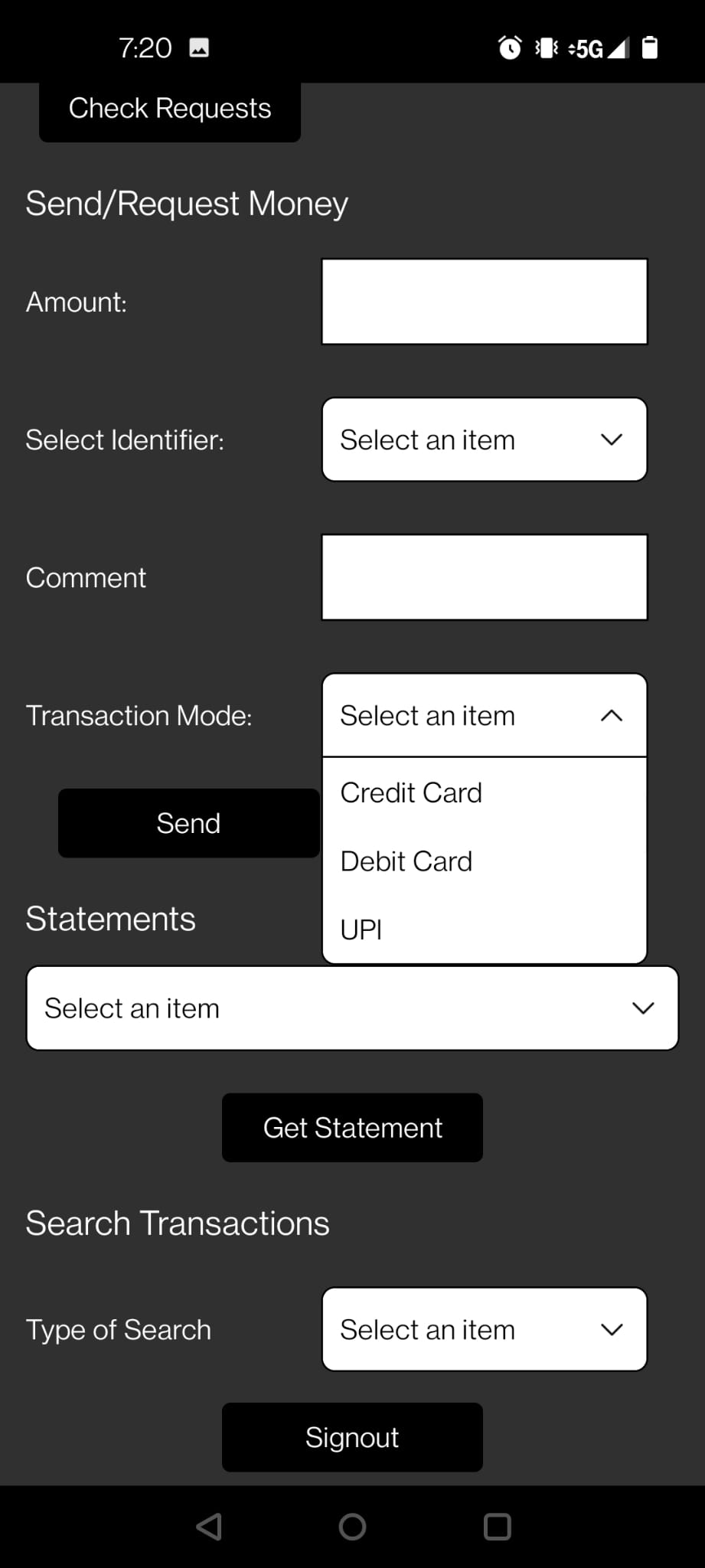
8. Best User



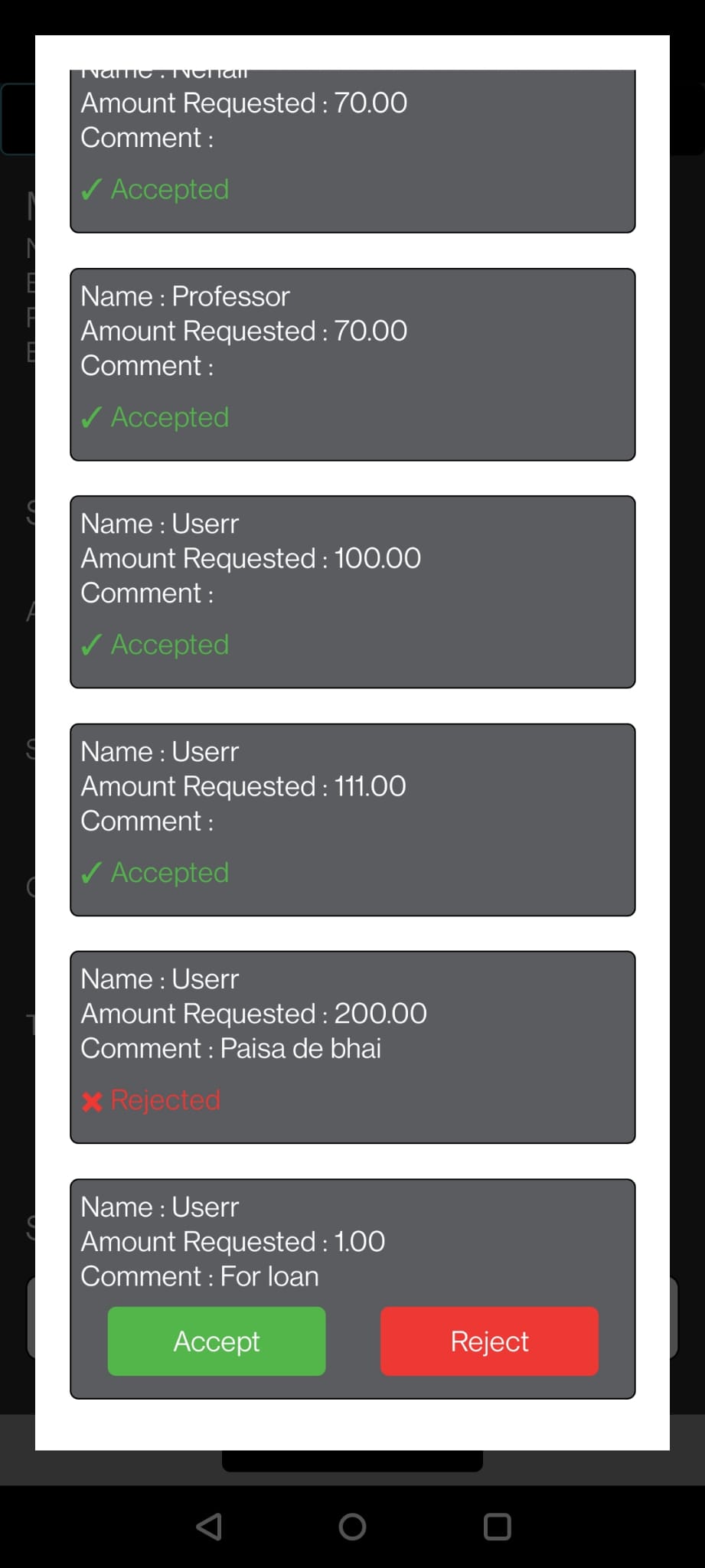
9. Max Transaction, Total & Average Per month



10. Modes of Transaction



11. Request to Accept or Reject Request



12. Identifier Type

