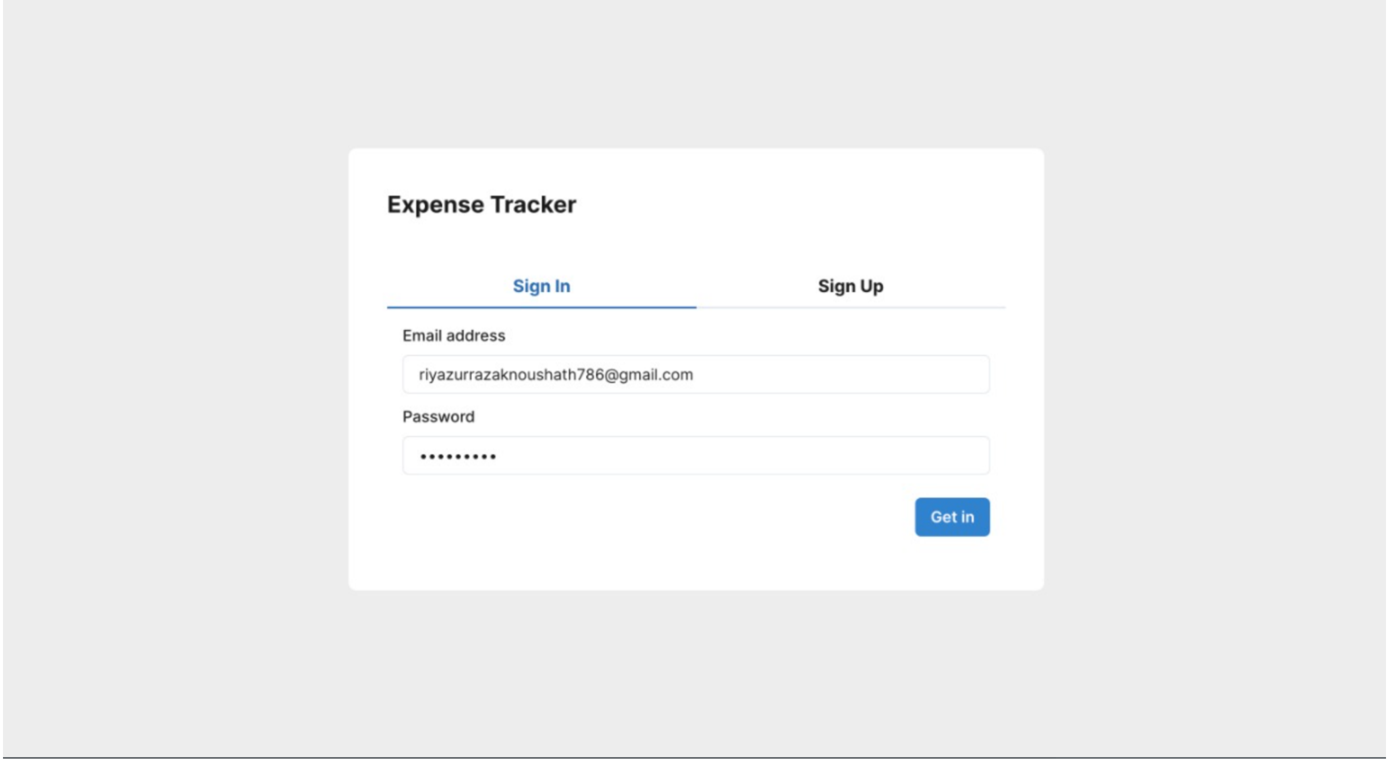
Sprint 1

Team ID	PNT2022TMID04671
Project Name	Personal Expense Tracker

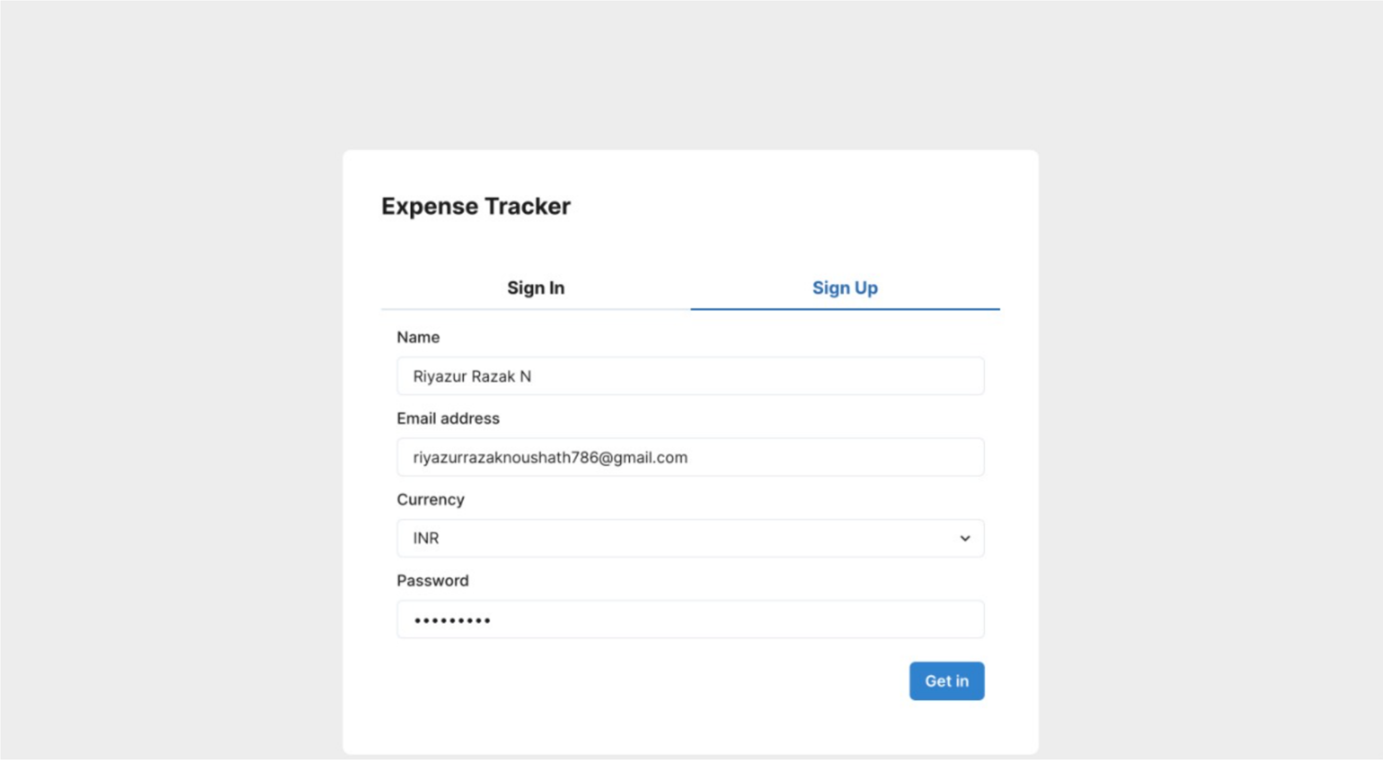
LOGIN & REGISTER PAGE SNAPSHOTS



Expense Tracker

Email address: riyazurrazaknoushath786@gmail.com

Password: •••••••••



Expense Tracker

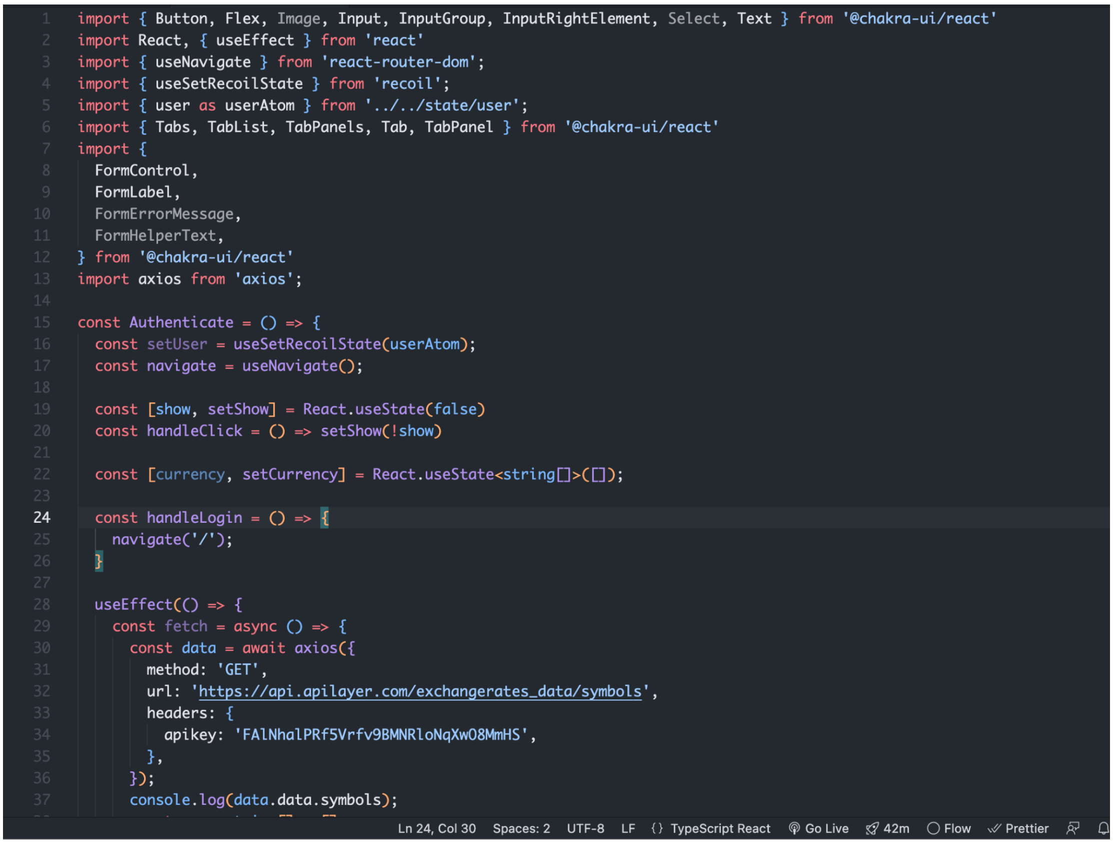
Name: Riyazur Razak N

Email address: riyazurrazaknoushath786@gmail.com

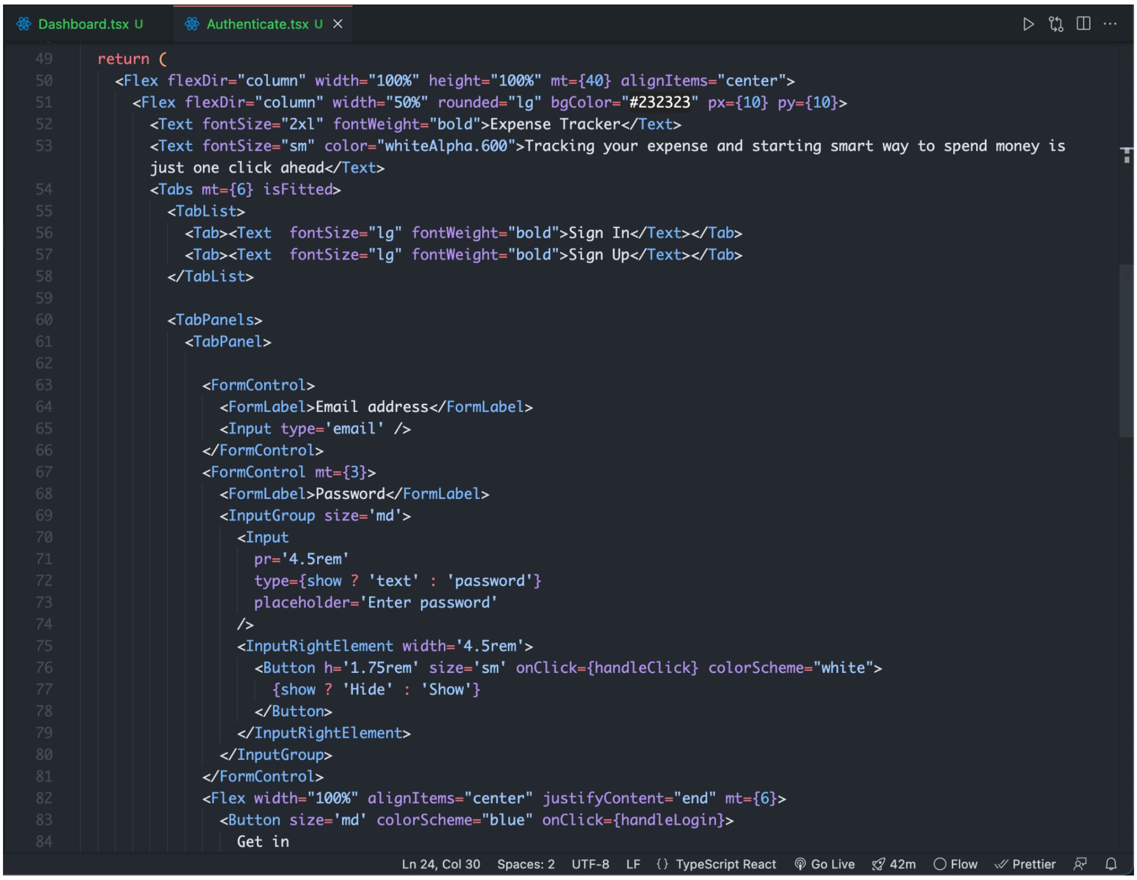
Currency: INR

Password: •••••••••

FRONTEND CODE SNAPSHOTS

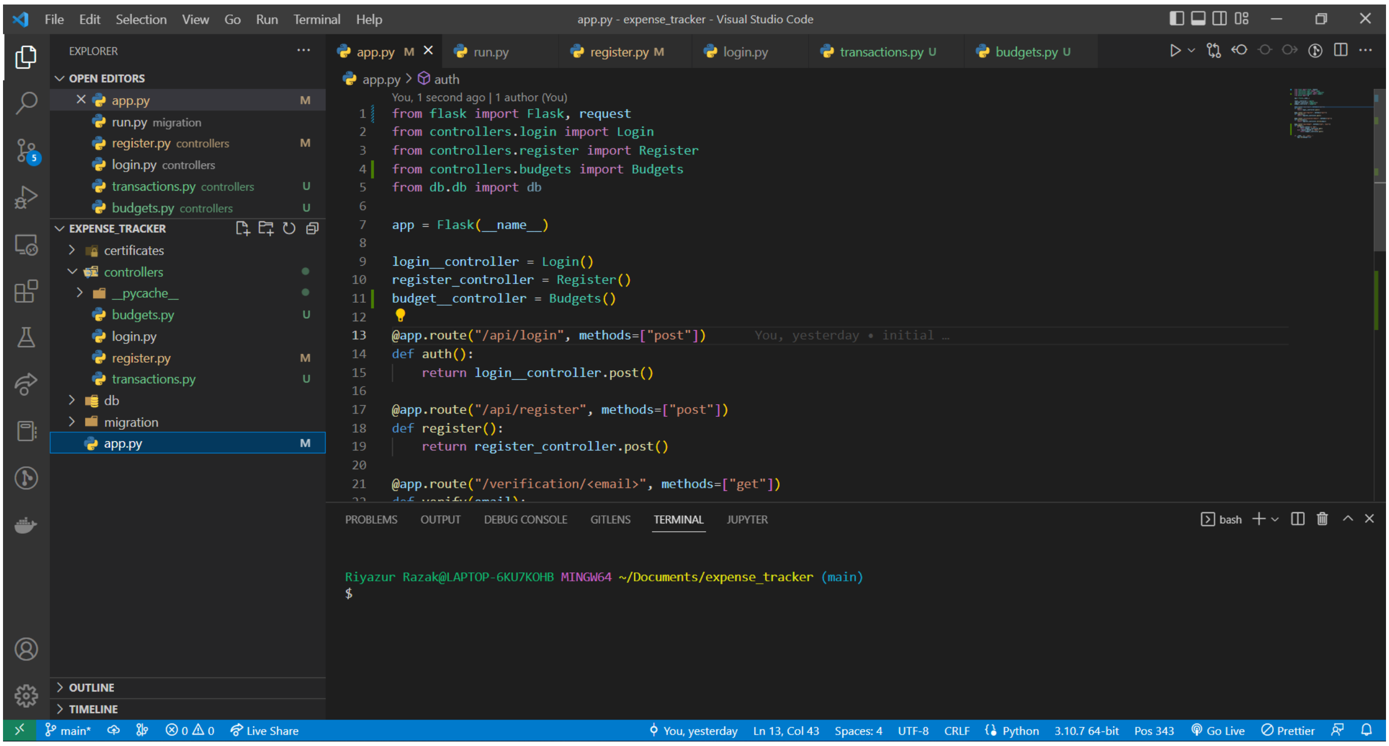


```tsx
import { Button, Flex, Image, Input, InputGroup, InputRightElement, Select, Text } from '@chakra-ui/react'
import React, { useEffect } from 'react'
import { useNavigate } from 'react-router-dom';
import { useSetRecoilState } from 'recoil';
import { user as userAtom } from '../../state/user';
import { Tabs, TabList, TabPanels, Tab, TabPanel } from '@chakra-ui/react'
import {
  FormControl,
  FormLabel,
  FormErrorMessage,
  FormHelperText,
} from '@chakra-ui/react'
import axios from 'axios';

const Authenticate = () => {
  const setUser = useSetRecoilState(userAtom);
  const navigate = useNavigate();

  const [show, setShow] = React.useState(false)
  const handleClick = () => setShow(!show)

  const [currency, setCurrency] = React.useState<string[]>([]);

  const handleLogin = () => {
    navigate('/');
  }

  useEffect(() => {
    const fetch = async () => {
      const data = await axios({
        method: 'GET',
        url: 'https://api.apilayer.com/exchangerates_data/symbols',
        headers: {
          apikey: 'FAlNhalPRf5Vrfv9BMNRloNqXwO8MmHS',
        },
      });
      console.log(data.data.symbols);
```



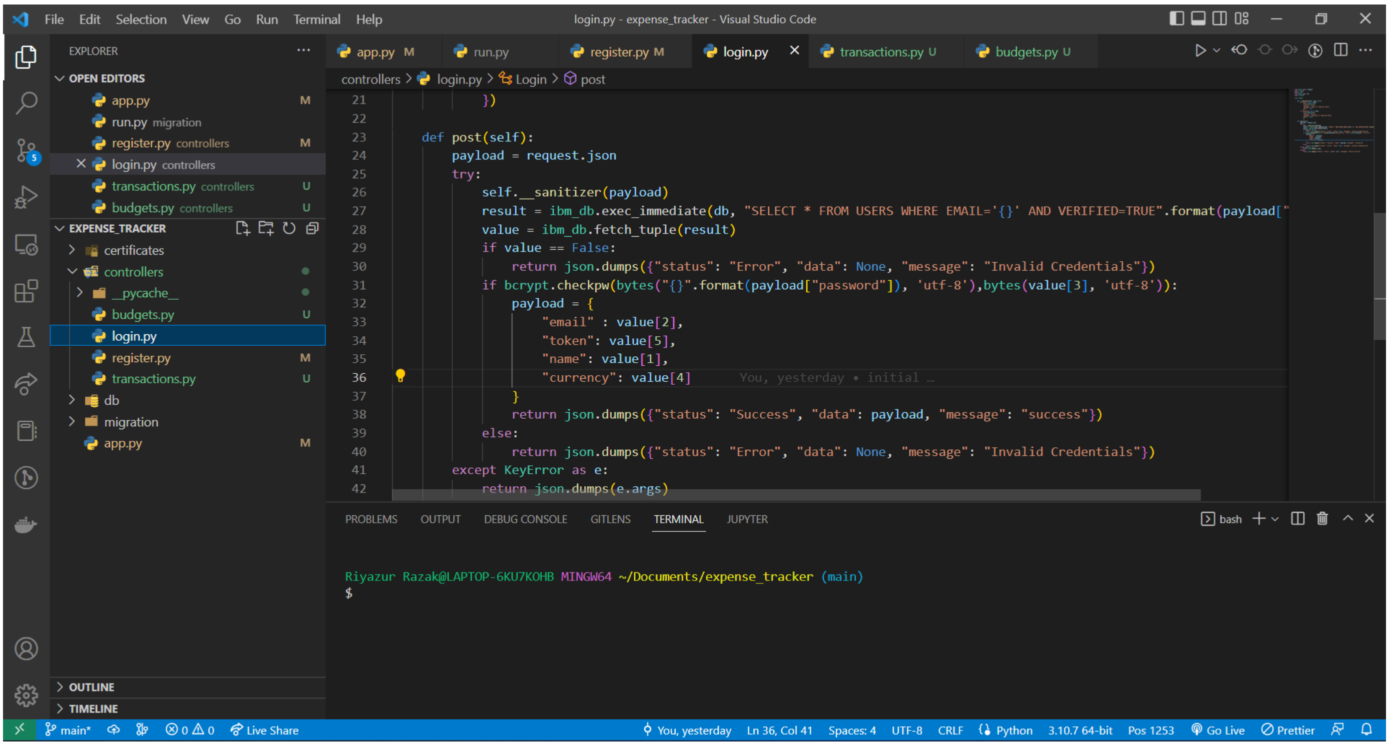
```tsx
  return (
    <Flex flexDir="column" width="100%" height="100%" mt={40} alignItems="center">
      <Flex flexDir="column" width="50%" rounded="lg" bgColor="#232323" px={10} py={10}>
        <Text fontSize="2xl" fontWeight="bold">Expense Tracker</Text>
        <Text fontSize="sm" color="whiteAlpha.600">Tracking your expense and starting smart way to spend money is just one click ahead</Text>
        <Tabs mt={6} isFitted>
          <TabList>
            <Tab><Text  fontSize="lg" fontWeight="bold">Sign In</Text></Tab>
            <Tab><Text  fontSize="lg" fontWeight="bold">Sign Up</Text></Tab>
          </TabList>

          <TabPanels>
            <TabPanel>

              <FormControl>
                <FormLabel>Email address</FormLabel>
                <Input type='email' />
              </FormControl>
              <FormControl mt={3}>
                <FormLabel>Password</FormLabel>
                <InputGroup size='md'>
                  <Input
                    pr='4.5rem'
                    type={show ? 'text' : 'password'}
                    placeholder='Enter password'
                  />
                  <InputRightElement width='4.5rem'>
                    <Button h='1.75rem' size='sm' onClick={handleClick} colorScheme="white">
                      {show ? 'Hide' : 'Show'}
                    </Button>
                  </InputRightElement>
                </InputGroup>
              </FormControl>
              <Flex width="100%" alignItems="center" justifyContent="end" mt={6}>
                <Button size='md' colorScheme="blue" onClick={handleLogin}>
                  Get in
```

BACKEND CODE SNAPSHOTS



```python
from flask import Flask, request
from controllers.login import Login
from controllers.register import Register
from controllers.budgets import Budgets
from db.db import db

app = Flask(__name__)

login__controller = Login()
register_controller = Register()
budget__controller = Budgets()

@app.route("/api/login", methods=["post"])
def auth():
    return login__controller.post()

@app.route("/api/register", methods=["post"])
def register():
    return register_controller.post()

@app.route("/verification/<email>", methods=["get"])
def verify(email):
```

```
Riyazur Razak@LAPTOP-6KU7KOHB MINGW64 ~/Documents/expense_tracker (main)
$
```



```python
            })

    def post(self):
        payload = request.json
        try:
            self.__sanitizer(payload)
            result = ibm_db.exec_immediate(db, "SELECT * FROM USERS WHERE EMAIL='{}' AND VERIFIED=TRUE".format(payload["
            value = ibm_db.fetch_tuple(result)
            if value == False:
                return json.dumps({"status": "Error", "data": None, "message": "Invalid Credentials"})
            if bcrypt.checkpw(bytes("{}".format(payload["password"]), 'utf-8'),bytes(value[3], 'utf-8')):
                payload = {
                    "email" : value[2],
                    "token": value[5],
                    "name": value[1],
                    "currency": value[4]
                }
                return json.dumps({"status": "Success", "data": payload, "message": "success"})
            else:
                return json.dumps({"status": "Error", "data": None, "message": "Invalid Credentials"})
        except KeyError as e:
            return json.dumps(e.args)
```

```
Riyazur Razak@LAPTOP-6KU7KOHB MINGW64 ~/Documents/expense_tracker (main)
$
```