

Student Name: Muhammed Arsath C.N.

Seat No: 250

Project Id: 10

Project Title: Review Scheduling Portal for TAC

Technical Components:

Components	Tech Stack
1. Frontend	React Js
2. API	Rest API
3. Backend	Spring Boot
4. Database	MYSQL

1.PROBLEM STATEMENT:

Review Scheduling Portal for TAC: Build the system to schedule the review for the booked appointments domain wise.

2.WORK FLOW:

2.1. User Login:

- Users (students, teachers, admins) can Login to the portal using their Institute mail id (Bitsathy) only.
- User's (students, teachers, admins) can be inserted in the database.

2.2. Appointment Booking:

- Students can book appointments with teachers for reviews in specific domains.
- Teachers can view and manage their available slots for booking.

2.3. Slot Booking:

- Students can select an available slot for a review in a particular domain.
- The system should prevent students from booking more than one slot per domain.

2.4. Review Scheduling:

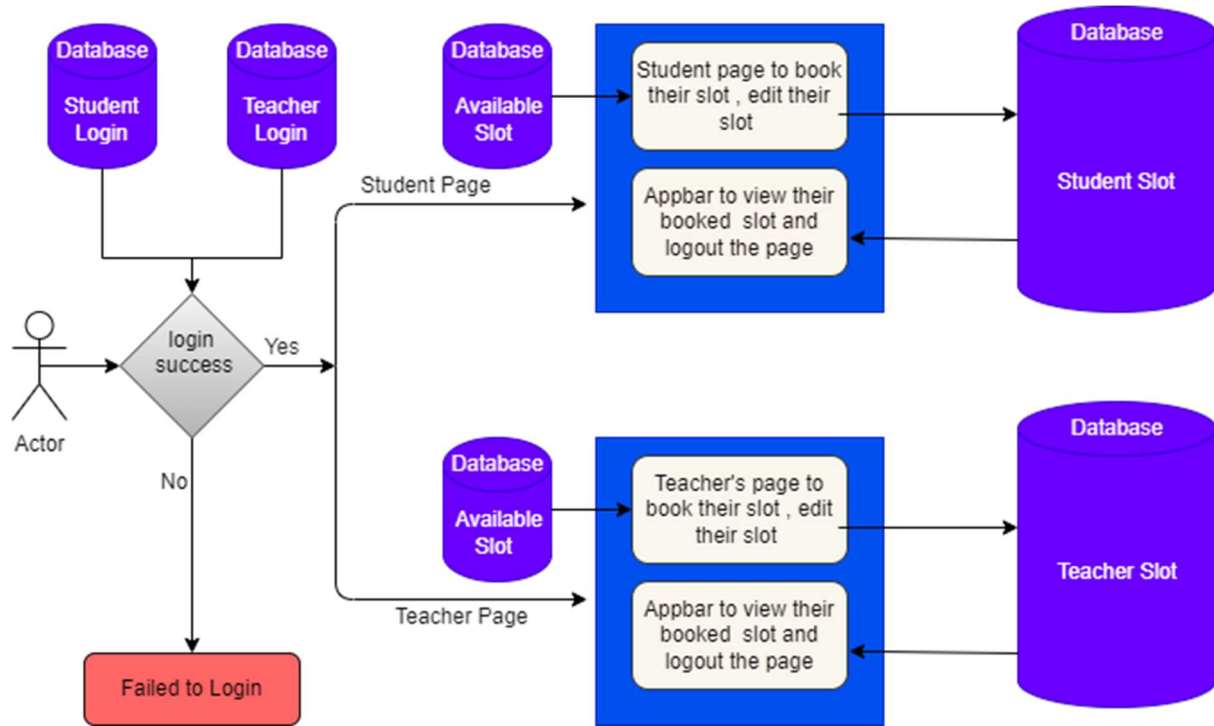
- After booking, the student's appointment is confirmed, and the review is scheduled.
- The review will be schedule based on one Teacher for 10 students.
- If the slot is over, it will not show to the students and teachers.

2.5. Admin Dashboard:

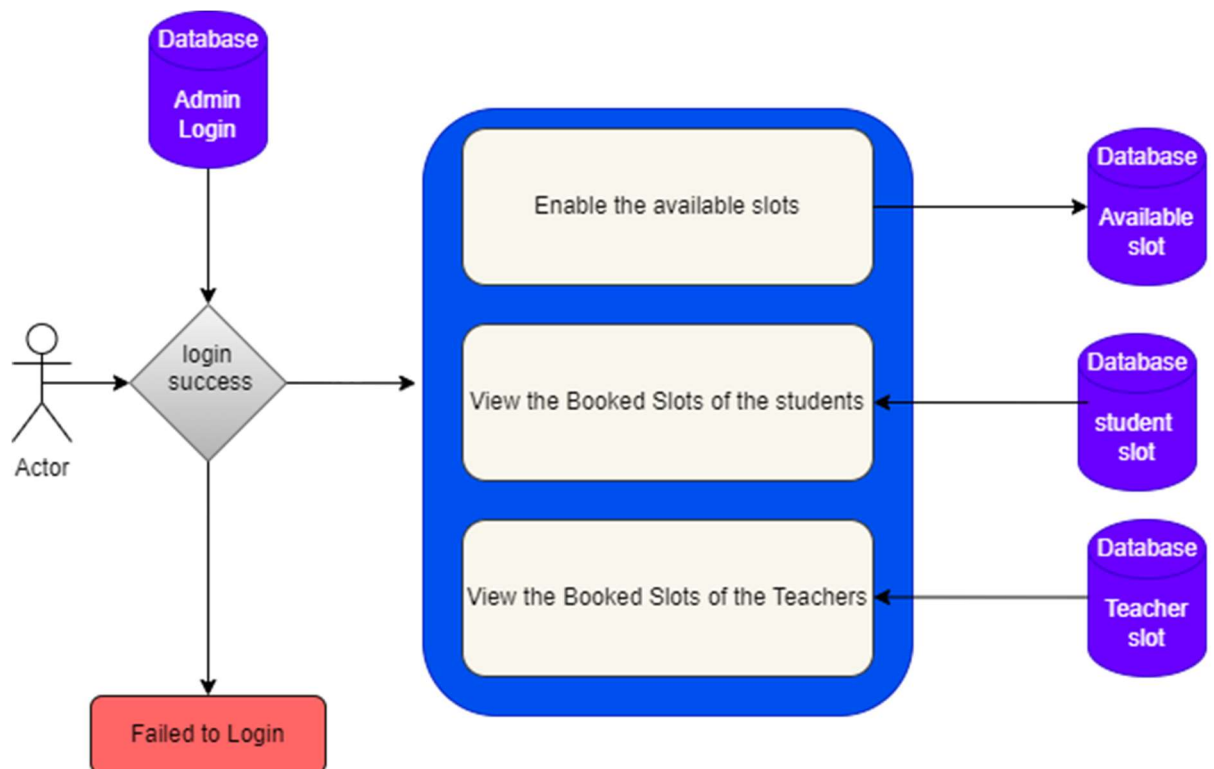
- Admins have access to a dashboard to manage users, appointments, and reviews.
- Admins can add, edit, or delete users, domains, and review slot.

4.FLOW CHART:

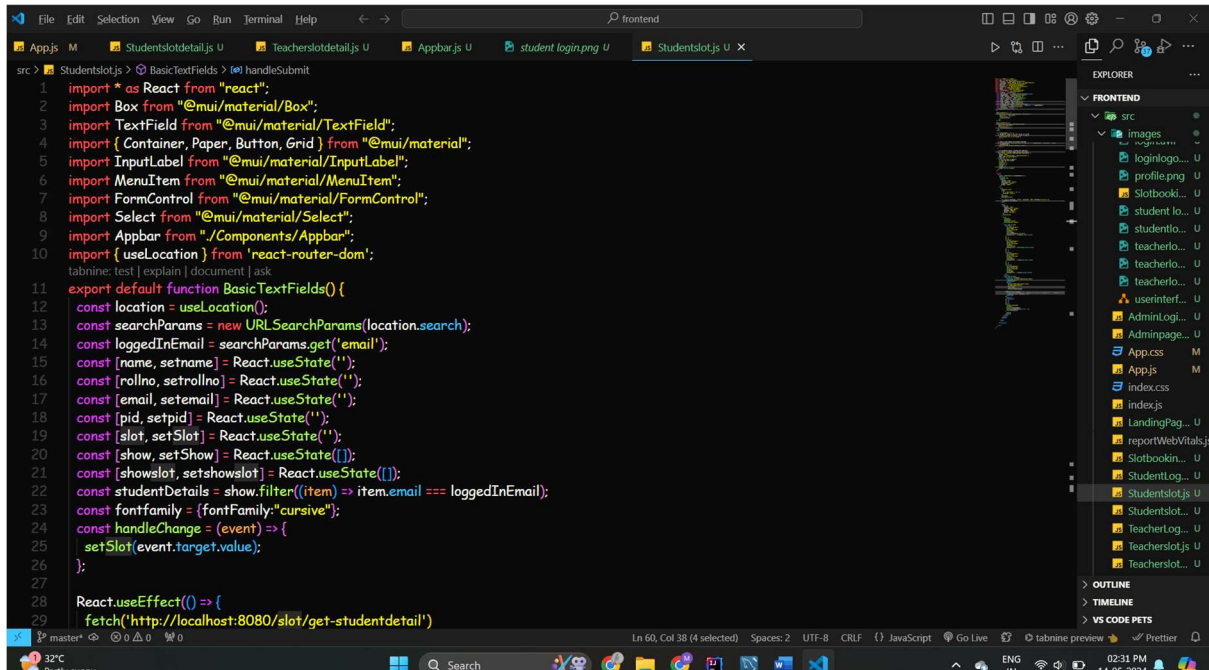
Student and Teacher Flow



Admin Flow

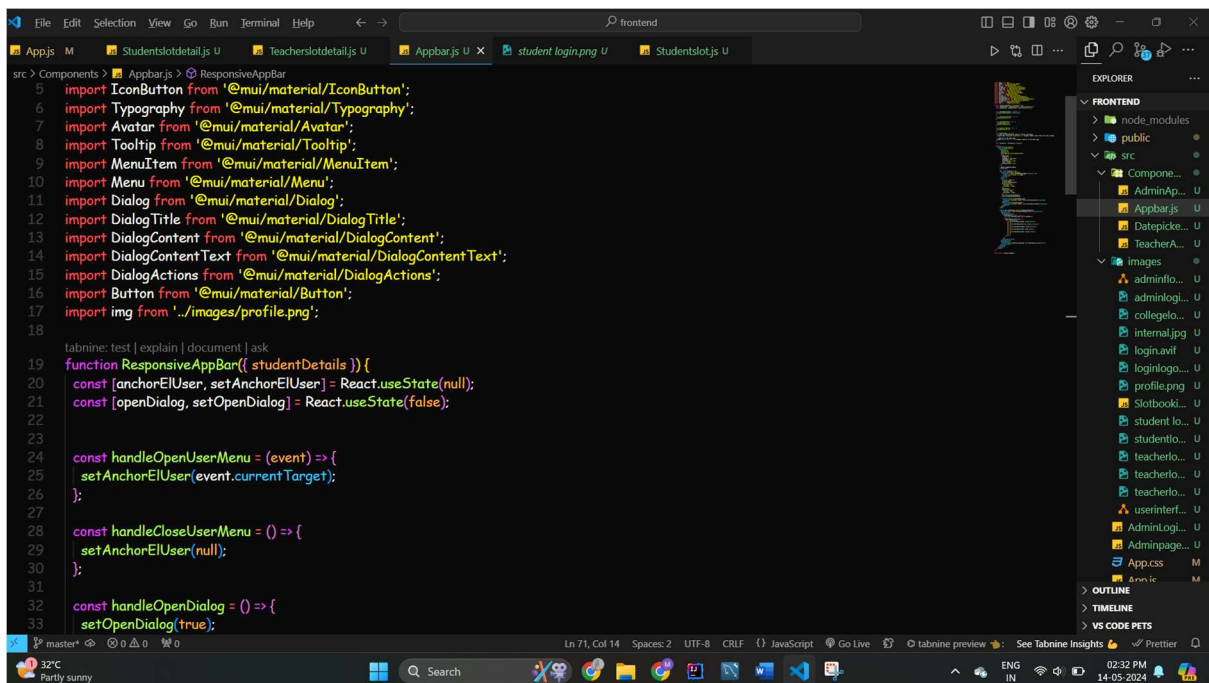


5.CODE (FRONTEND):



```
1 import * as React from "react";
2 import Box from "@mui/material/Box";
3 import TextField from "@mui/material/TextField";
4 import { Container, Paper, Button, Grid } from "@mui/material";
5 import InputLabel from "@mui/material/InputLabel";
6 import MenuItem from "@mui/material/MenuItem";
7 import FormControl from "@mui/material/FormControl";
8 import Select from "@mui/material/Select";
9 import AppBar from "../Components/Appbar";
10 import { useLocation } from "react-router-dom";
11 tabnine: test | explain | document | ask
12 export default function BasicTextFields() {
13   const location = useLocation();
14   const searchParams = new URLSearchParams(location.search);
15   const loggedInEmail = searchParams.get('email');
16   const [name, setName] = React.useState('');
17   const [rollno, setrollno] = React.useState('');
18   const [email, setEmail] = React.useState('');
19   const [pid, setpid] = React.useState('');
20   const [slot, setSlot] = React.useState('');
21   const [show, setShow] = React.useState(false);
22   const [showslot, setshowslot] = React.useState('');
23   const studentDetails = show.filter(item => item.email === loggedInEmail);
24   const fontFamily = {fontFamily: 'cursive'};
25   const handleChange = (event) => {
26     setSlot(event.target.value);
27   };
28   React.useEffect(() => {
29     fetch('http://localhost:8080/slot/get-studentdetail')
```

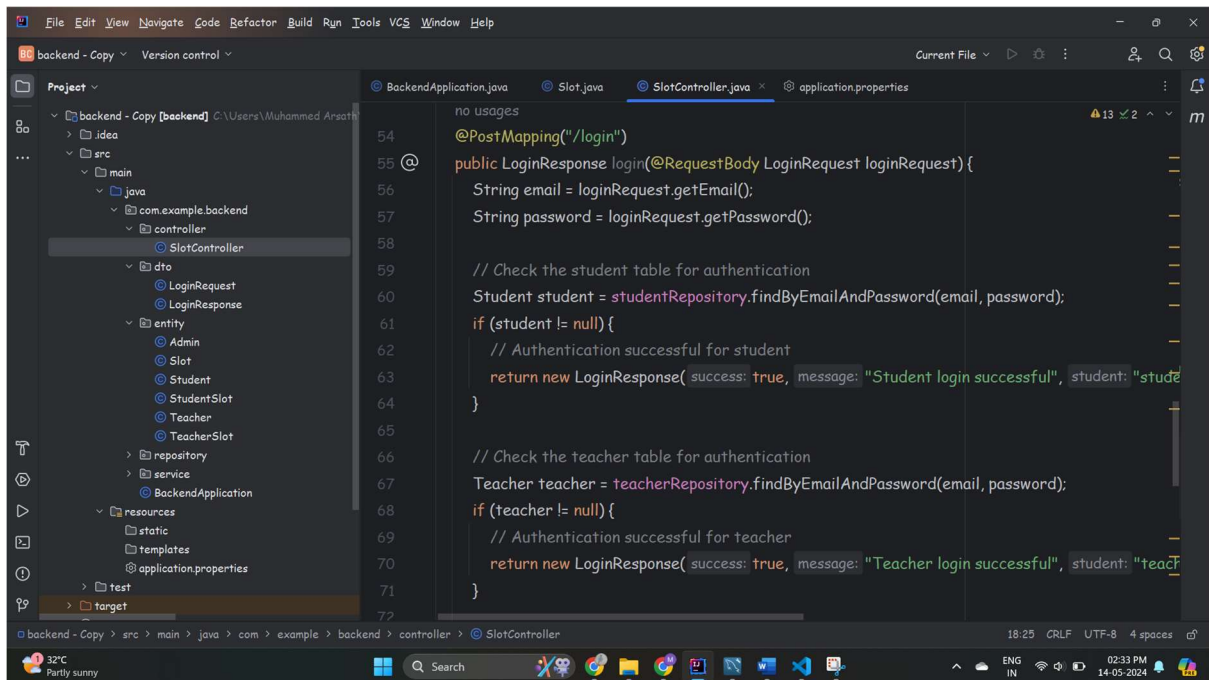
The screenshot shows a VS Code editor window with the file 'Studentslot.js' open. The code defines a function 'BasicTextFields' which uses 'useLocation' to get search parameters. It sets up state for name, rollno, email, pid, slot, and show. A 'handleChange' function is defined to update the 'slot' state. A 'useEffect' hook is used to fetch student details from a backend API based on the logged-in email.



```
5 import IconButton from '@mui/material/IconButton';
6 import Typography from '@mui/material/Typography';
7 import Avatar from '@mui/material/Avatar';
8 import Tooltip from '@mui/material/Tooltip';
9 import MenuItem from '@mui/material/MenuItem';
10 import Menu from '@mui/material/Menu';
11 import Dialog from '@mui/material/Dialog';
12 import DialogTitle from '@mui/material/DialogTitle';
13 import DialogContent from '@mui/material/DialogContent';
14 import DialogContentText from '@mui/material/DialogContentText';
15 import DialogActions from '@mui/material/DialogActions';
16 import Button from '@mui/material/Button';
17 import img from '../images/profile.png';
18
19 tabnine: test | explain | document | ask
20 function ResponsiveAppBar({ studentDetails }) {
21   const [anchorElUser, setAnchorElUser] = React.useState(null);
22   const [openDialog, setOpenDialog] = React.useState(false);
23
24   const handleOpenUserMenu = (event) => {
25     setAnchorElUser(event.currentTarget);
26   };
27
28   const handleCloseUserMenu = () => {
29     setAnchorElUser(null);
30   };
31
32   const handleOpenDialog = () => {
33     setOpenDialog(true);
```

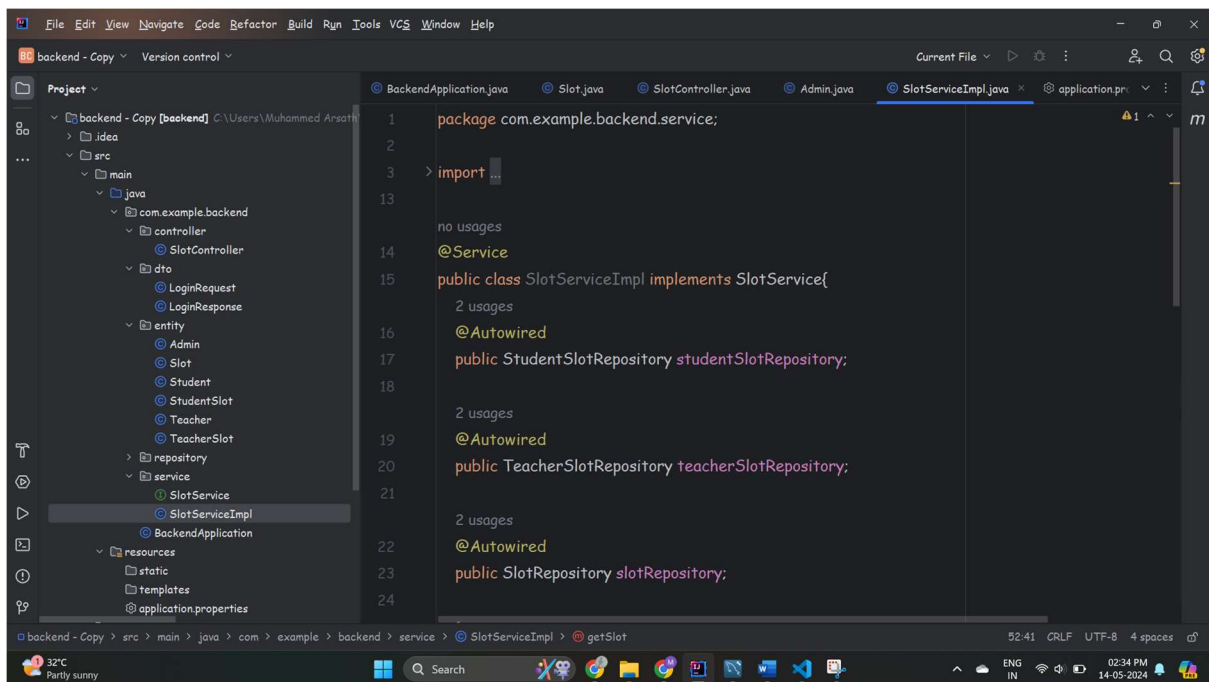
The screenshot shows a VS Code editor window with the file 'ResponsiveAppBar.js' open. The code defines a 'ResponsiveAppBar' function that takes 'studentDetails' as a prop. It sets up state for 'anchorElUser' and 'openDialog'. It defines 'handleOpenUserMenu' to open a user menu and 'handleCloseUserMenu' to close it. It also defines 'handleOpenDialog' to open a dialog box.

5.2 CODE(BACKEND):



```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
backend - Copy Version control
Project
  backend - Copy [backend] C:\Users\Muhammed Ansath
    src
      main
        java
          com.example.backend
            controller
              SlotController
            dto
              LoginRequest
              LoginResponse
            entity
              Admin
              Slot
              Student
              StudentSlot
              Teacher
              TeacherSlot
            repository
            service
              BackendApplication
            resources
            static
            templates
            application.properties
            test
            target

SlotController.java
no usages
54 @PostMapping("/login")
55 public LoginResponse login(@RequestBody LoginRequest loginRequest) {
56     String email = loginRequest.getEmail();
57     String password = loginRequest.getPassword();
58
59     // Check the student table for authentication
60     Student student = studentRepository.findByEmailAndPassword(email, password);
61     if (student != null) {
62         // Authentication successful for student
63         return new LoginResponse( success: true, message: "Student login successful", student: "stude
64     }
65
66     // Check the teacher table for authentication
67     Teacher teacher = teacherRepository.findByEmailAndPassword(email, password);
68     if (teacher != null) {
69         // Authentication successful for teacher
70         return new LoginResponse( success: true, message: "Teacher login successful", student: "teacF
71     }
72
18:25 GRLF UTF-8 4 spaces
backend - Copy > src > main > java > com > example > backend > controller > SlotController
```



```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
backend - Copy Version control
Project
  backend - Copy [backend] C:\Users\Muhammed Ansath
    src
      main
        java
          com.example.backend
            controller
              SlotController
            dto
              LoginRequest
              LoginResponse
            entity
              Admin
              Slot
              Student
              StudentSlot
              Teacher
              TeacherSlot
            repository
            service
              SlotService
              SlotServiceImpl
              BackendApplication
            resources
            static
            templates
            application.properties

SlotServiceImpl.java
1 package com.example.backend.service;
2
3 import
13 no usages
14 @Service
15 public class SlotServiceImpl implements SlotService{
16     2 usages
17     @Autowired
18     public StudentSlotRepository studentSlotRepository;
19
20     2 usages
21     @Autowired
22     public TeacherSlotRepository teacherSlotRepository;
23
24     2 usages
25     @Autowired
26     public SlotRepository slotRepository;
27
52:41 GRLF UTF-8 4 spaces
backend - Copy > src > main > java > com > example > backend > service > SlotServiceImpl > @getSlot
```

6.PROJECT OUTPUT:

