SOFTWARE ENGINEERING

SOFTWARE REQUIREMENT SPECIFICATIOIN

GROUP MEMBERS-

Aashish Raghav (IIT2022010)
Abhimanyu Choudhary (IIT2022075)
Kovid Kumar Juyal (IIT2022063)
Yashraj Singh Panwar (IIT2022059)
Saurabh Dubey (IIT2022107)

The Software Design Specification

1. Introduction

The Software Design Document is a document to provide documentation which will be used to aid in software development by providing the details for how the software should be built. Within the Software Design Document are narrative and graphical documentation of the software design for the project including use case models, sequence diagrams, collaboration models, object behaviour models, and other supporting requirement information.

Purpose of this document

This document will define the design of the one runway simulator. It contains specific information about the expected input, output, classes, and functions. The interaction between the classes to meet the desired requirements are outlined in detailed figures at the end of the document.

Scope of the development project

We describe what features are in the scope of the software and what are not in the scope of the software to be developed

In Scope:

a) Organizers request to Administration to book a event and required venue.
b) Administration has power to approve or reject the event.
c) User (all students) can retrieve information about all events (upcoming and currently going) in the college in no time
d) Even past events can also be accessed by users.

Definitions, Acronyms and Abbreviations:

Acronyms and Abbreviations:

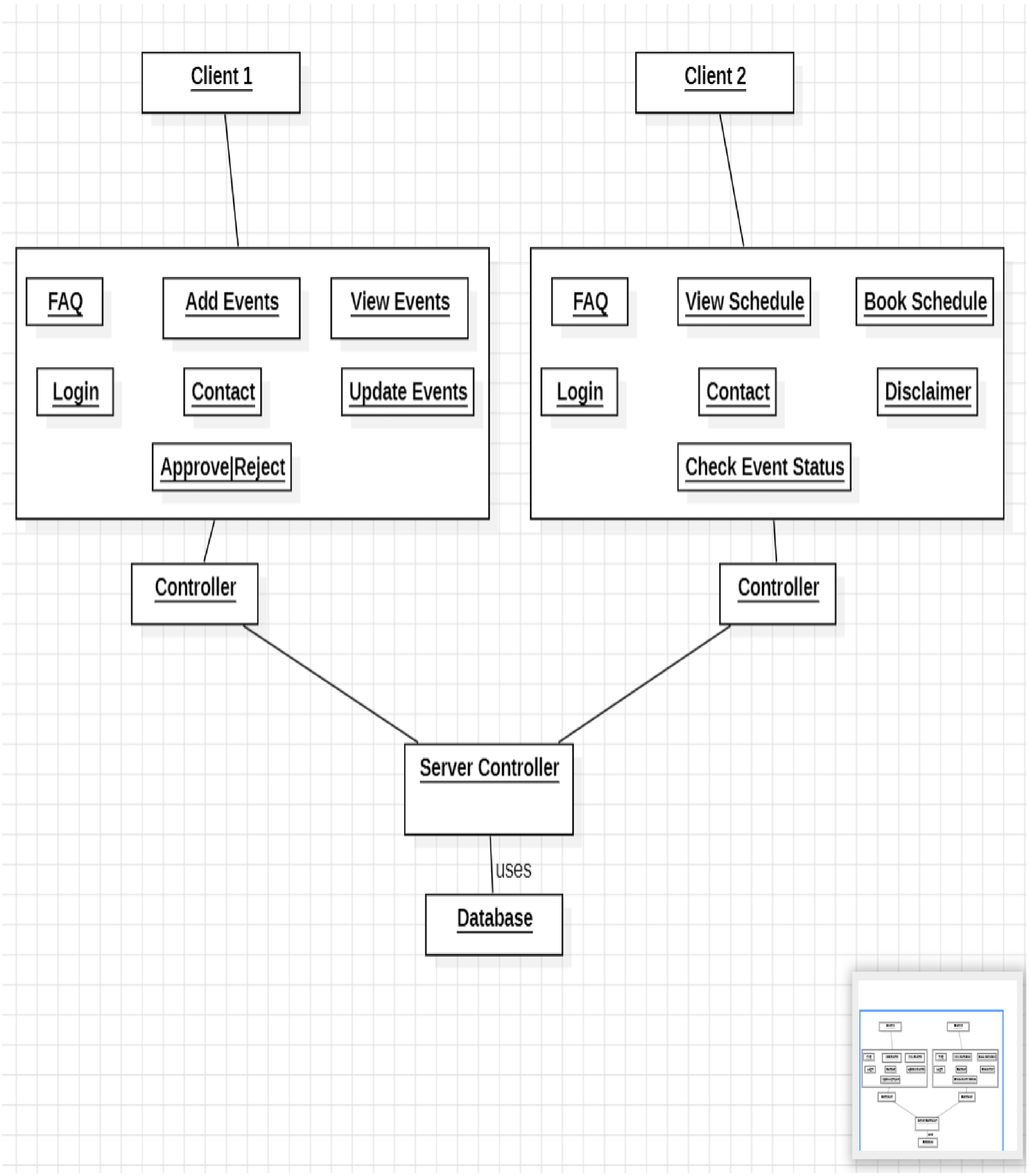
a) SRS: Software Requirement Specification
b) CC3,CC2,CC1: Computer Centre
c) SAC: Student Activity Centre
d) AAA: Administration Block
e) CBREMS: College Building Reservation and Event Management System

Definitions:

a) Student Gymkhana: Student body of IIITA
b) AAA: Administration Section of IIITA

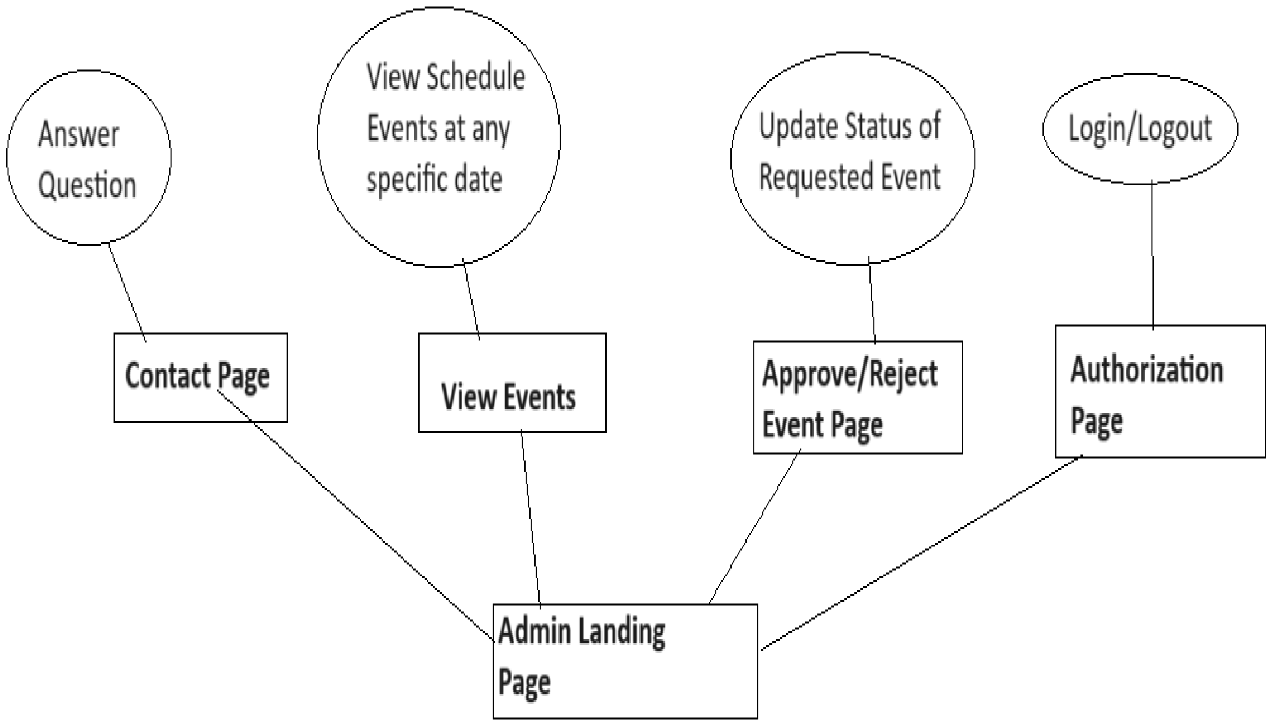
Conceptual Architecture/Architecture Diagram:

Architecture Diagram 1:



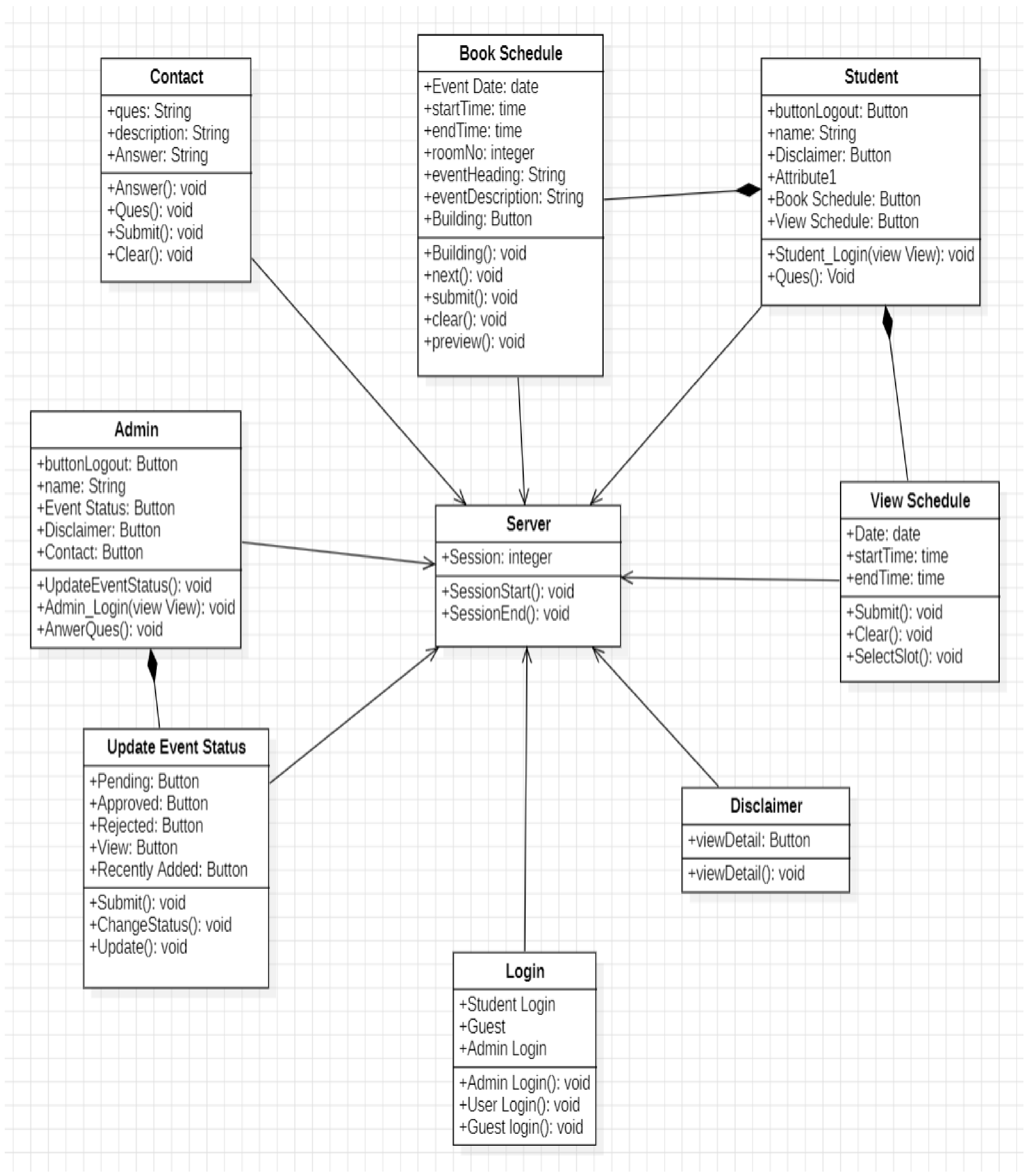
Structure and Relationships

Admin’s Side:


Answer Question
View Schedule Events at any specific date
Update Status of Requested Event
Login/Logout
Contact Page
View Events
Approve/Reject Event Page
Authorization Page
Admin Landing Page

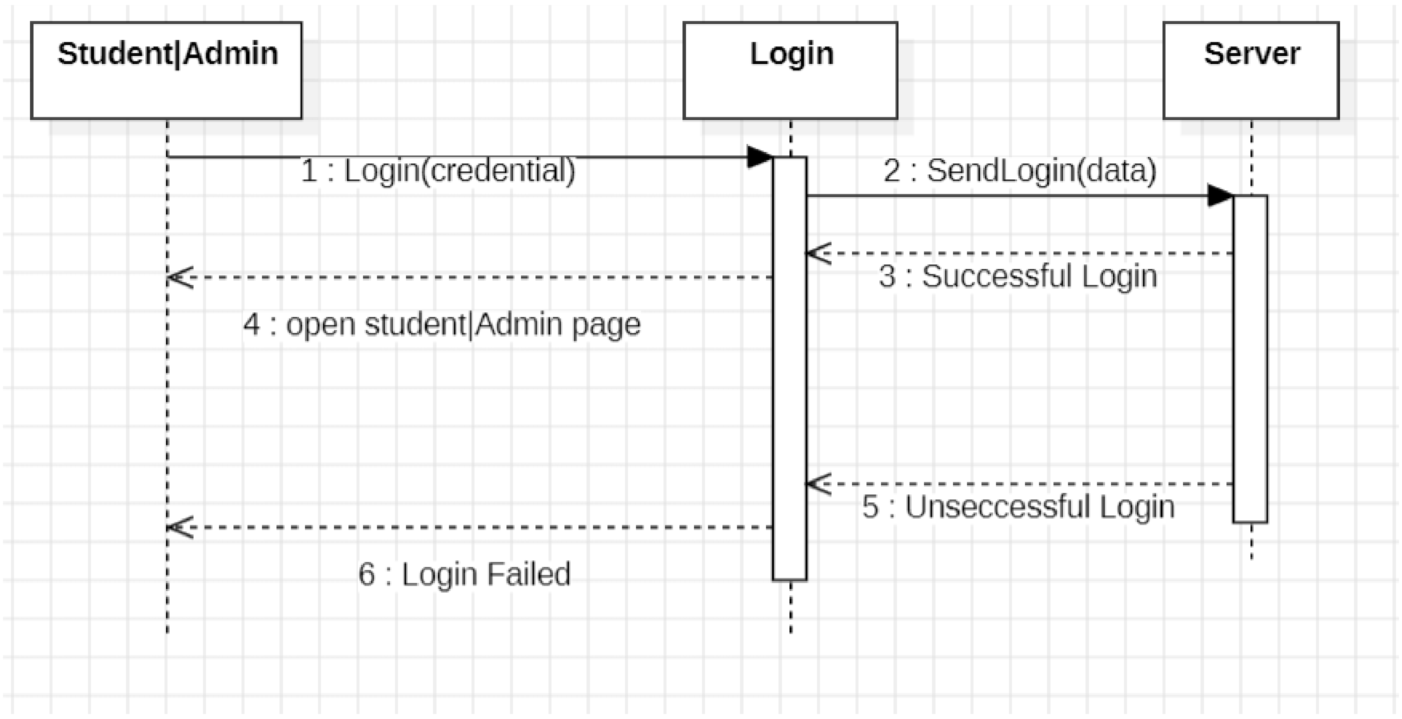
Logical Architecture (Class Diagram, Sequence Diagram, State Diagram)

Class Diagram:

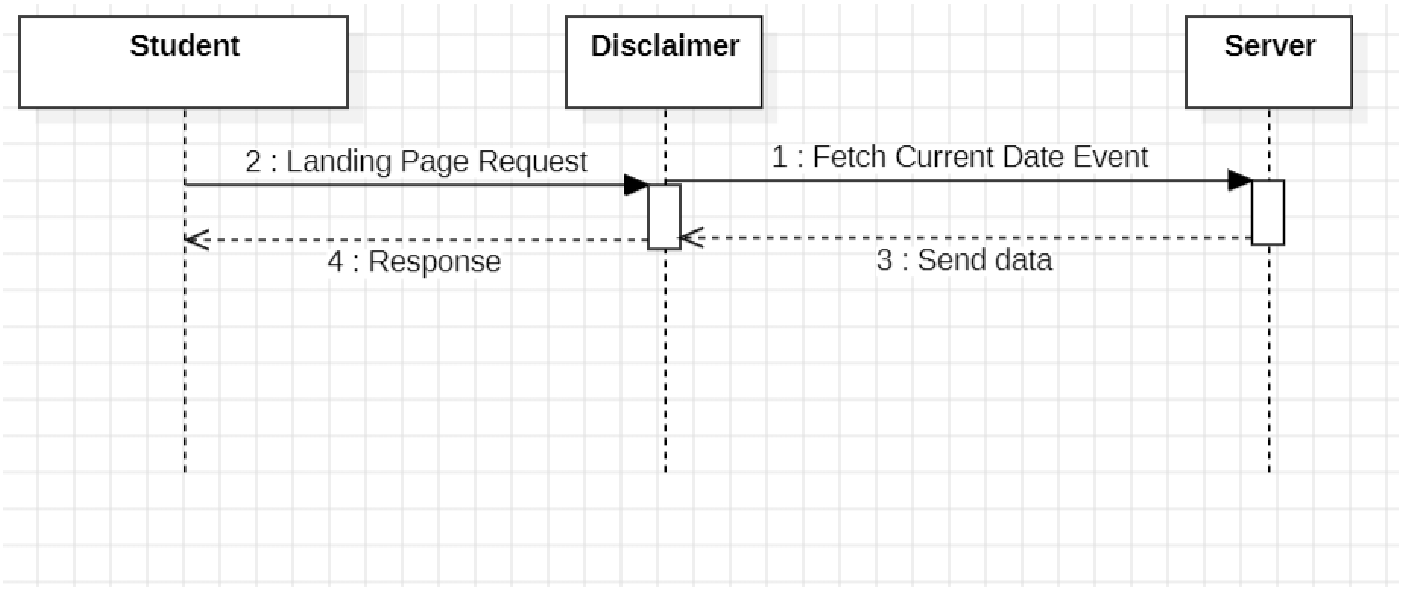


Sequence Diagram:

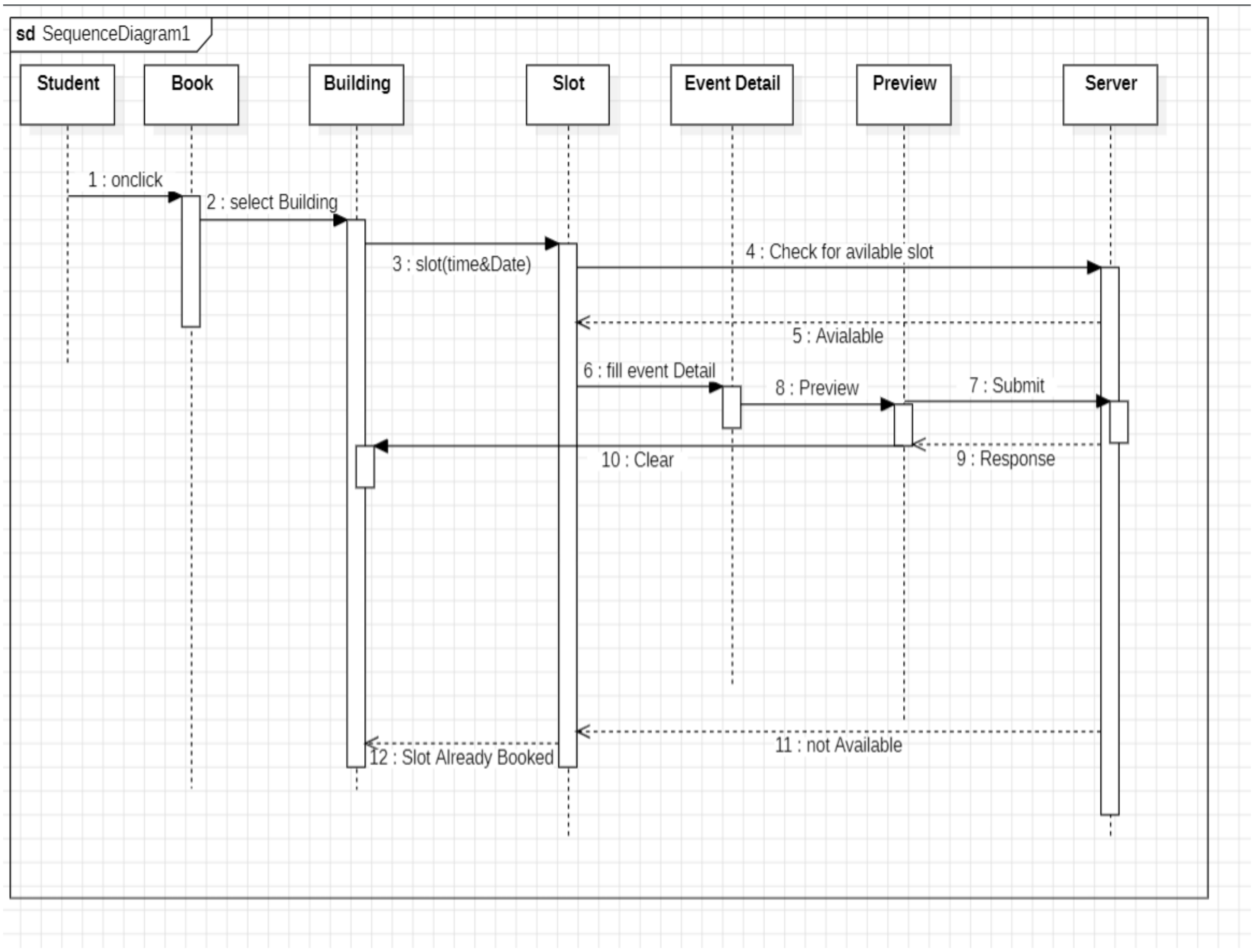
Login Page:



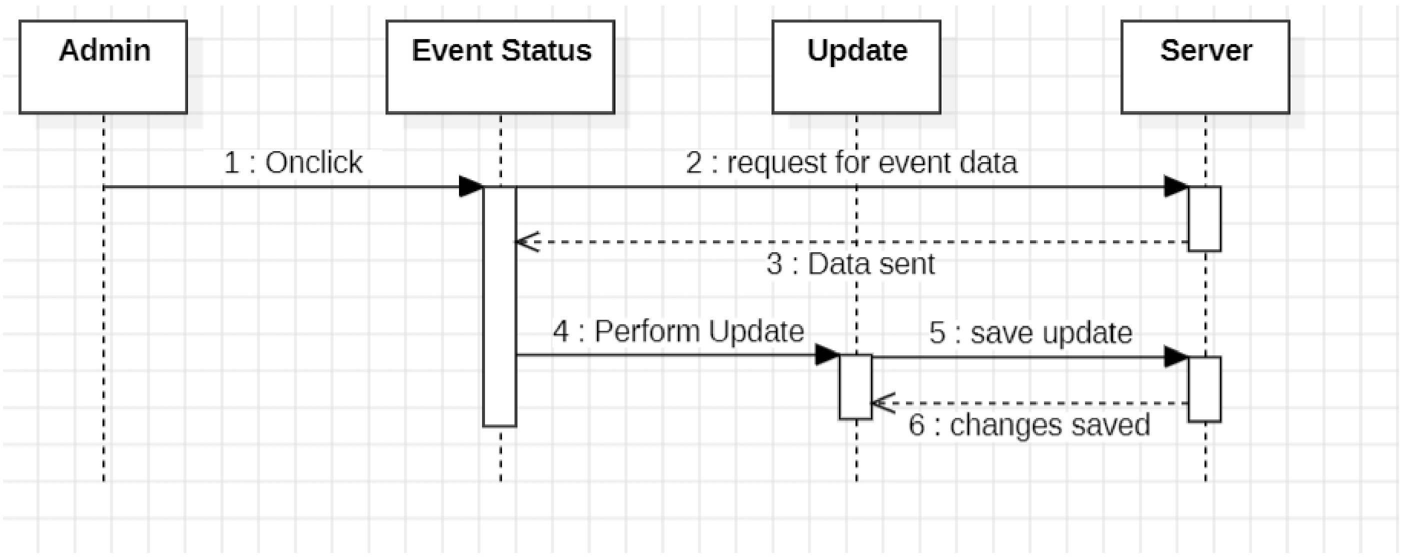
Disclaimer Page:



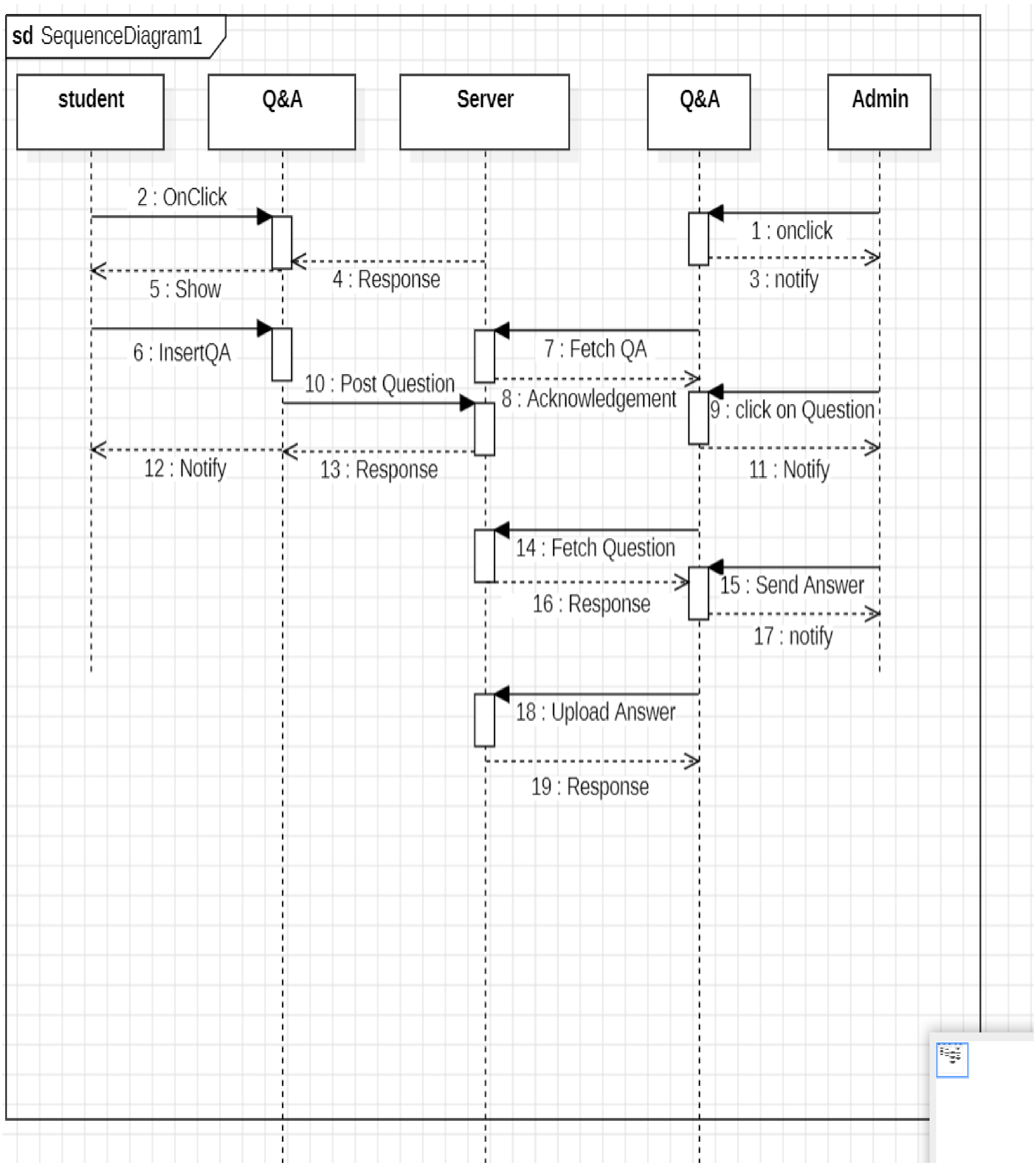
Book Schedule Page:



Event Status Page:



Contact Page:



Class Diagram Explaination :

Most of the classes extends Sever class which is being shown by direct association linkage. It is being shown by black-coloured Arrow.
The classes which extends Server are: Login, Disclaimer, Contact, Event Status, Book Schedule, View Schedule, Admin, Student , Update Event Status.

There are also some composition linkage, shown by lines with Black Diamong, which signifies that if the parent class is removed, the child class also loses it's existence.
These are: Book Schedule, View Schedule depends on Student.
The lines without arrow shows that the connection is bi-direction. It signifies that one class just makes instance of other class, but not dependent on each other in any way.

1. Class name: Login

Description: This class allows the user to enter the system by authenticating the entered credentials.

Method 1: Student_Login()
Input: view email,password
Output: Student Landing Page of login successfully
Method Description:
This method takes input as email and password from the user and check whether it is authorized login or not and if the result is successful it leads to another activity page.

Method 2: Guest ()
Input: view email,password
Output: Guest Landing Page.
Method Description:
This method is for those who just wants to see upcoming events.

Method 3: Admin_Login()

Input: view email,password

Output: Admin Landing Page of login successfully

Method Description:

This method takes input as email and password from the user and check whether it is authorized login or not and if the result is successful it leads to another activity page.

2. Class Name: Admin

Description:

This class enables the admin to enter into the subsystem (Landing page) after authenticating the entered credentials.

Method 1: Logout ()

Input: OnClick.

Output: Landing Page after logout successfully

Method Description:

This method allows admin to logout of active session.

Method 2: Update_Event_Status()

Input: NULL

Output: Launch the Activity

Method Description:

This method allows the admin to change current status of event requested by the student.

For particular event admin can mark its status as:

Approved

Rejected

Method 3: Answer_Ques()

Input: SavedInstanceState , OnClickListener of Q/A button

Output: Launch the Activity.

Method Description:
1) When a user click on Q/A button, It will call a method from button class setOnClickListener.
2) Inside this method(setOnClickListener) It will create a object of Intent class.
3) Intent class object opens a new activity, which will get its content from “qa.js” file.
4) Then a new page is opened using object of Intent class like - startActivityForResult(intent, 0)

3. Class Name: Update_Event_Status

Description:
This class enables admin to change status of event.

Method 1: Pending ()

Input: NULL.
Output: Change status of event to Pending.
Method Description:
This method allows admin to set event status as Pending.

Method 2: Approved ()

Input: Onclick
Output: Change status of event to Approved.
Method Description:
This method allows admin to set event status as Approved.

Method 3: Rejected ()

Input: Onclick.
Output: Change status of event to Rejected.
Method Description:
This method allows admin to set event status as Rejected.

Method 4: View ()

Input: Onclick.

Output: View events on basis of dates and time.

Method Description:

This method allows admin to view events on basis of their requested date and time.

Method 5: Recently_Added ()

Input: Onclick.

Output: Recently requested events.

Method Description:

This method allows admin to view events which are recently requested to be approved.

Method 6: Submit ()

Input: Onclick.

Output: Lands on new Activity Page.

Method Description:

This method allows admin to save the changes and same are updated in database and reflected to guests and students.

4. Class Name: Contact

Description:

This class enables is act as a common link between admin and student. Student can have queries, which can be uploaded as question. Admin can reply to those queries in same Q&A section.
Also, most FAQ would also be presents as comments in bottom of landing page.

Method 1: Answer ()

Input: Id of question to be answered.

Output: Q/A appeared in asked section box.

Method Description:

This method allows admin to answer questions that are asked in contact question.

Method 2: Ques ()

Input: NULL.

Output: Question is being sent to admin.

Method Description:

This method allows student to ask any doubt about event status or registering new event

Method 3: Submit ()

Input: Onclick.

Output: Completes the process of Q/A.

Method Description:

This method allows both admin and student to complete their respective Q/A tasks.

Method 4: Clear ()

Input: Onclick.

Output: Reset the process of Q/A.

Method Description:

This method allows both admin and student to reset their respective Q/A tasks.

5. Class Name: Student

Description:

This class enables the student to enter into the subsystem (Landing page) after authenticating the entered credentials.

Method 1: Logout ()

Input: OnClick.

Output: Landing Page after logout successfully

Method Description:

This method allows admin to logout of active session.

Method 2: Book_Schedule()

Input: onClick

Output: Landing after clicking.

Method Description:

This method allows the student to trigger booking process of event.

Method 3: view_Schedule()

Input: OnClickListener of view Schedule button

Output: Launch the Activity.

Method Description:

This method routes web page to view_Schedule page.

Method 3: Ques()

Input: OnClickListener of ques button

Output: Launch the Activity.

Method Description:

This method routes web page Q/A or contact page.

5. Class Name: Book Schedule

Description:

This class enables the student to start booking process, finalize event description and sent a request for event to be approved.

Method 1: Building ()

Input: Id of building currently booking slot for.

Output: Landing Page after selecting building successfully

Method Description:

This method allows student to continue booking process for specific building.

Method 2: next()

Input: onClick

Output: Landing after clicking.

Method Description:

There are various stages involved in booking.

Stage 1: building

Stage 2: Slot booking
Stage 3: Event description
Stage 4; Preview and submit.

This next method allows to process step by step to each stage.

Method 3: Submit()

Input: OnClickListener of submit button

Output: Launch the Activity.

Method Description:

After successfully adding all description of event and checking available slots and previewing event, finally can submit event to get approved.

Method 4: Clear()

Input: OnClickListener of clear button

Output: Launch the Activity.

Method Description:

Clear the filled form and start a new form to book event from scratch.

6. Class Name: View Schedule

Description:

This class enables the student to view all events on basis of date and time.

Method 1: SelectSlot()

Input: Time and date of slot searching for

Output: All those events which are occurring in selected time frame

Method Description:

This method takes time and date of slot and returns all events that overlap with those slots.

Method 2: Submit()

Input: onClick

Output: Landing after clicking.

Method Description:

This methods proceeds selectSlot method to sent request to server and get desired output.

Method 4: Clear()

Input: OnClickListener of clear button

Output: Launch the Activity.

Method Description:

Clear the filled slots and allows to enter new slots for searching.

6. Class Name: Disclaimer

Description:

This class begins as a home page for student where all events occurring on current date are reflected.

Method 1: ViewDetail()

Input: onClick

Output: Detailed Description of events

Method Description:

In Disclaimer all events would be reflected, view Details enables user to get in detail description of that particular event.

Code for some components:

Client:

Index.js

```
import React from "react";
import ReactDOM from "react-dom";

//REDUX;
import { Provider } from "react-redux";
import store from "./app/store";

import App from "./App";
import "./index.css";

const container = document.getElementById("root");
const root = ReactDOM.createRoot(container);

root.render(
  <Provider store = {store}>
    <App />
  </Provider>
);
```

App.js

```
import React, { useEffect } from "react";
import Disclaimer from "./components/Disclaimer/Disclaimer";
import Buildings from "./components/Buildings/Buildings";
import Schedule from "./components/Schedule/Schedule";
import {Navbar,Container,Nav} from "react-bootstrap";
import 'bootstrap/dist/css/bootstrap.min.css'
import { useDispatch } from "react-redux";
import BookingForm from "./components/BookingForm/BookingForm";
import {
   BrowserRouter as Router,
   Route,
   Routes,
   Link,
} from "react-router-dom";
import { getBookings } from "./actions/bookings";
function App(){
   const[curid,setId] = React.useState("0");
   const dispatch = useDispatch();
   useEffect(()=>{
      dispatch(getBookings("1"));
   },[dispatch])
   return (
      <Router>
         <Container className = "mt-4 container" >
            <Navbar expand = "sm"   >
                <Container className="d-flex bd-highlight">
                    <svg className="icon" xmlns="http://www.w3.org/2000/svg" viewBox="0 0 
512 512"><path d="M160 96a96 96 0 1 1 192 0A96 96 0 1 1 160 96zm80 152V512l-48.4-24.2c-
20.9-10.4-43.5-17-66.8-19.3l-96-9.6C12.5 457.2 0 443.5 0 427V224c0-17.7 14.3-32 32-
32H62.3c63.6 0 125.6 19.6 177.7 56zm32 264V248c52.1-36.4 114.1-56 177.7-56H480c17.7 0 32 
14.3 32 32V427c0 16.4-12.5 30.2-28.8 31.8l-96 9.6c-23.2 2.3-45.9 8.9-66.8 19.3L272 
512z"/></svg>
                    <div className= "heading me-auto">ELITE</div>
                    <div className= "heading me-auto">Feel Free to Book & Explore</div>
                        <Nav>
                            <Link className = "routeLink" to ="/">Home</Link>
                            <Link className = "routeLink" to ="/book">Book Schedule</Link>
                            <Link className = "routeLink" to ="/view">View Schedule</Link>
                        </Nav>
                    <div>
                        <svg className="icon" xmlns="http://www.w3.org/2000/svg" 
viewBox="0 0 448 512"><path d="M224 256A128 128 0 1 0 224 0a128 128 0 1 0 0 256zm-45.7 
48C79.8 304 0 383.8 0 482.3C0 498.7 13.3 512 29.7 512H418.3c16.4 0 29.7-13.3 29.7-29.7C448 
383.8 368.2 304 269.7 304H178.3z"/></svg>
                    </div>
                </Container>

            </Navbar>
        </Container>
         <div>
            <Routes>
               <Route path="/" element={<Disclaimer/>} />
               <Route path="/book" element={
                  <>
                     <Buildings  curid = {curid} setId = {setId} buttonValue="Book"/>
                  </>
               } />
               <Route path="/view"  element={<Buildings curid = {curid} setId = {setId} 
buttonValue="View"/>} />
               <Route path="/view/id/:id" element={
                  <>
                     <Schedule curid={curid} />
                  </>
               }/>
               <Route path= "/book/id/:id" element={
                  <>
                     <BookingForm curid = {curid}/>
                  </>
               } />
            </Routes>
         </div>
      </Router>
   );
}
```

```
export default App;
```

Booking Form:

```
import React,{useState} from "react";
import useStyles from "./styles";
import { useSelector } from "react-redux";
import SlotBooking from "./Booking/SlotBooking";
import EventBooking from "./Booking/EventBooking";
import FinalBooking from "./Booking/FinalBooking";
import { nanoid } from "@reduxjs/toolkit";

function BookingForm({ curid }) {
    const classes = useStyles();
    //have booking
    const {bookings} = useSelector((state)=>state.bookings);

    //have booking of building
    const [bookingB,setBookingB] = useState(
        bookings.filter((booking)=>{
            return (booking.building_id === curid);
        })
    );
    //new Event Entry
    const [eventSlot,setEventSlot] = useState({
        evDate : new Date(),
        startTime : new Date(),
        endTime : new Date(),
        building_id : curid,
    });
    const [eventDetail,setEventDetail] = useState({
        room : " ",
        heading : " ",
        body : " ",
        building_id : curid,
        event_id : nanoid()
    })
    const [formStep,setFormStep] = useState({
        slotBook : false,
        eventBook : false,
        finalBook : false
    });
    return (
        <div>


            {/* step 1 */}
                {formStep.slotBook === false ? <SlotBooking curid={curid}
                                    eventSlot = {eventSlot} setEventSlot={setEventSlot}
                                    bookingB={bookingB} setBookingB={setBookingB}
                                    formStep={formStep} setFormStep={setFormStep}/> :
<></>}
            {/* step 2 */}
            {(formStep.slotBook === true && formStep.eventBook === false) ? <EventBooking
curid={curid}
                                eventDetail = {eventDetail}
setEventDetail={setEventDetail}
                                formStep={formStep} setFormStep={setFormStep}/> : <></>}

            {/* step 3 */}
            {(formStep.slotBook === true && formStep.eventBook === true &&
formStep.finalBook === false) ? <FinalBooking curid={curid}
                                eventDetail = {eventDetail} eventSlot = {eventSlot}
            /> : <></>}
        </div>
    );
}
export default BookingForm;
```

Slot Booking:

```
import React,{useState} from "react";
import {Paper,Box,Button,} from "@mui/material";
import buildingTrack from "../../../constants/constant";
import useStyles from "./styles";
import dayjs from "dayjs";
import { AdapterDayjs } from "@mui/x-date-pickers/AdapterDayjs";
import { LocalizationProvider } from "@mui/x-date-pickers/LocalizationProvider";
import { DatePicker } from "@mui/x-date-pickers/DatePicker";
import { TimePicker } from "@mui/x-date-pickers/TimePicker";

function SlotBooking({
curid,eventSlot,setEventSlot,bookingB,setBookingB,formStep,setFormStep}) {
    console.log(bookingB);
    const classes = useStyles();
    const handleSubmit = (event)=>{
        event.preventDefault();

        if (eventSlot.startTime.getTime() > eventSlot.endTime.getTime()){
            alert("Start Time should be less than endTime");
            clear();
        }
        else{
            //check already booked
            let flag = 0;
            console.log(bookingB);
            bookingB.forEach(element => {
                if (flag === 0 && element.event_day === event.evDate){
                    if (element.start_time > event.endTime || element.end_time <
event.startTime){}
```

```
                    else{
                        console.log(element);
                        flag = 1;
                        alert("Slot Already Booked! Please Check Booked Schedule!");
                    }

                }
            });
            //if not already booked
            if (flag === 0){
                alert("Slot Available Can Proceed");
                console.log(eventSlot);
                setFormStep({...formStep,slotBook : true});
            }

        }
    }
    const clear = ()=>{
        setEventSlot({evDate : new Date(),startTime : new Date(), endTime : new Date()});

    }
    return (
        <Paper className={classes.paper}>
        <h1 class= "Heading">{buildingTrack[curid]}</h1>
        <form
            autoComplete="off"
            noValidate
            className={`${classes.root} ${classes.form}`}
            onSubmit={handleSubmit}>
            <LocalizationProvider dateAdapter={AdapterDayjs}>
                <DatePicker className={classes.entryField} label = "Event Date"
                    value={dayjs(eventSlot.evDate)}
                    onChange={(newValue)=> setEventSlot({...eventSlot,evDate : new
Date(newValue)})}
                />
                <TimePicker className={classes.entryField}  label = "Start Time"
                    value = {dayjs(eventSlot.startTime)}
                    onChange= {(newValue)=> setEventSlot({...eventSlot,startTime : new
Date(newValue)})}
                    />
                <TimePicker className={classes.entryField} label = "End Time"
                    value = {dayjs(eventSlot.endTime)}
                    onChange= {(newValue)=> setEventSlot({...eventSlot,endTime : new
Date(newValue)})}
                />
            </LocalizationProvider>

            <Box sx={{ width: "100%" }} mb={1}>
            <Button variant="contained" color="primary" size="large" type="submit"
fullWidth
            >
                Next
            </Button>
            </Box>
        </form>
        </Paper>
    );
}
export default SlotBooking;
```

Event Booking:

```
import React from "react";
import {Paper,Box,Button, TextField,} from "@mui/material";
import useStyles from "./styles";
import buildingTrack from "../../../constants/constant";

function EventBooking({curid,eventDetail,setEventDetail,formStep,setFormStep}){
    const classes = useStyles();
    const handleSubmit = (event)=>{
        setFormStep({...formStep,eventBook : true});
    }
    return(
        <div>
            <Paper className={classes.paper}>
                <h1 class= "Heading"> {buildingTrack[curid]} </h1>
                <form autoComplete="off"
                noValidate
                className={`${classes.root} ${classes.form}`}
                onSubmit={handleSubmit}>

                    <TextField variant="outlined" className={classes.entryField} value =
{eventDetail.room} label="Room No" onChange={(e)=> setEventDetail({...eventDetail,room :
e.target.value})}></TextField>
                    <TextField variant="outlined" className={classes.entryField} value =
{eventDetail.heading} required label="Event Heading"
onChange={(e)=>setEventDetail({...eventDetail,heading : e.target.value})}></TextField>
                    <TextField variant="outlined" className={classes.entryField} value =
{eventDetail.body} required multiline rows={5}label="Event Details"
onChange={(e)=>setEventDetail({...eventDetail,body : e.target.value})}></TextField>
                    <Box sx={{ width: "100%" }} mb={1}>
                        <Button variant="contained" color="primary" size="large"
type="submit" fullWidth>
                            Next
                        </Button>
                    </Box>
                </form>
            </Paper>
        </div>
    )
}
export default EventBooking;
```

Final Booking:

```
import React from "react";
import {Paper,Box,Button, TextField,} from "@mui/material";
import useStyles from "./styles.js";
import buildingTrack from "../../../constants/constant";
import {  useDispatch} from "react-redux";
import { requestBooking } from "../../../actions/bookings.js";
import { useNavigate } from 'react-router-dom';


function FinalBooking({curid,eventDetail,eventSlot}){
    const classes = useStyles();
    const sTime =  new Date(eventSlot.startTime).toLocaleTimeString();
    const day =  new Date(eventSlot.evDate).toLocaleDateString();
    const eTime = new Date(eventSlot.endTime).toLocaleTimeString();
    console.log(eTime);
    console.log(eventSlot);
    const dispatch = useDispatch();
    const navigate = useNavigate();

    const handleSubmit = (event)=>{
        event.preventDefault();
        navigate("/");
        const eventBook = {...eventDetail,...eventSlot};
        dispatch(requestBooking(eventBook));
        console.log(eventBook);
        alert("Request has been Sent.Thank You For booking");
    }
    return(
        <Paper className={classes.paper }>
                <h1 class= "Heading"> {buildingTrack[curid]} </h1>
                <form autoComplete="off"
                noValidate
                className={`${classes.root} ${classes.form}`}
                onSubmit={handleSubmit}
                >
                    <TextField variant="outlined" className={classes.showInfoEntry}
InputProps={{readOnly: true}} value = {eventDetail.event_id} label="Event-Id"
></TextField>
                    <TextField variant="outlined" className={classes.showInfoEntry}
InputProps={{readOnly: true}} value = {day} label="Day" ></TextField>
                    <TextField variant="outlined" className={classes.showInfoEntry}
InputProps={{readOnly: true}} value = {sTime} label="Start Time" ></TextField>
                    <TextField variant="outlined" className={classes.showInfoEntry}
InputProps={{readOnly: true}} value = {eTime} label="End Time" ></TextField>
                    <TextField variant="outlined" className={classes.showInfoEntry}
InputProps={{readOnly: true}} value = {eventDetail.room} label="Room No" ></TextField>
                    <TextField variant="outlined" className={classes.showInfoEntry}
InputProps={{readOnly: true}} value = {eventDetail.heading} required label="Event
Heading"></TextField>
                    <TextField variant="outlined" className={classes.entryField}
InputProps={{readOnly: true}} value = {eventDetail.body} required multiline
rows={5}label="Event Details" ></TextField>
```

```
                    <Box sx={{ width: "100%" }} mb={1}>
                        <Button variant="contained" color="primary" size="large"
type="submit" fullWidth>
                            Request a Booking
                        </Button>
                    </Box>
                </form>
            </Paper>
    )
}
export default FinalBooking;
```

Disclaimer:

```
import { Paper,List } from "@mui/material";
import React from "react";
import Event from "./Event/Event";
import { useSelector} from "react-redux";
function Disclaimer(){
    const {bookings} = useSelector((state)=> state.bookings);
    const [bookingToday,setBooking] = React.useState(
        bookings.filter((booking)=>{
            console.log(booking);
            const currDate = new Date(booking.event_day);
            console.log(currDate.getDate());
            return ((booking.booking_status === "APPROVED") && (currDate.getDate() === new
Date().getDate()));
        })
    );
    return(
        <Paper elevation={3} square={false} sx = {{margin : "20px auto",maxWidth : "60%"
}}>
            <List sx= {{
                // py: 0,
                // width: '100%',
                // maxWidth: 360,
                borderRadius: 2,
                border: '1px solid black',
                borderColor: 'divider',
                backgroundColor: 'background.paper',
                margin : "auto",
            }}>
            {bookingToday.map((entry)=>{
                return <Event key = {entry.booking_id} event = {entry}/>
            })}
                <Event/>
            </List>
        </Paper>
    )
}

export default Disclaimer;
```

Event:

```
import React from "react";
import { Divider,ListItemText,ListItemButton,Collapse,List } from "@mui/material";
import ExpandLess from '@mui/icons-material/ExpandLess';
import ExpandMore from '@mui/icons-material/ExpandMore';
function Event({event}){
  const [open,setOpen] = React.useState(false);
  function handleClick(){
    setOpen(!open)
  }
  if (event === undefined) return;
    const venue = "Venue : " + event.building_name;
    console.log(venue);
    const time = "Time : " + new Date(event.start_time).toLocaleTimeString() + " - " + new
Date(event.end_time).toLocaleTimeString();
    return (
        <>
        <ListItemButton onClick={handleClick}>
        <ListItemText primary={event.desc_Heading} />
            {open ? <ExpandLess /> : <ExpandMore />}
      </ListItemButton>
      <Collapse in={open} timeout="auto" unmountOnExit>
        <List component="div" disablePadding>
          <ListItemButton sx={{ pl: 4 }}>
            <ListItemText sx={{ m: 0,p : 0,mt : -1}}secondary={venue} />
          </ListItemButton>
          <ListItemButton sx={{ pl: 4}}>
            <ListItemText sx={{ m : 0, p : 0,mt : -1}}secondary={time} />
          </ListItemButton>
          <ListItemButton sx={{ pl: 4}}>
            <ListItemText sx={{ m : 0, p : 0,mt : -1}}secondary={event.desc_body} />
          </ListItemButton>
        </List>
      </Collapse>
        <Divider variant="middle" />
        </>

    );
}

export default Event;
```

Schedule:

```
import React from "react";
import { Paper , Typography,Box,TextField,Button} from "@mui/material";
import { useSelector } from "react-redux";
import useStyles from "./styles";
import today from "./today";
import FullCalendar from '@fullcalendar/react'
import dayGridPlugin from '@fullcalendar/daygrid' // a plugin!
import timeGridPlugin from "@fullcalendar/timegrid";

function Schedule({curid}){
    const {bookings} = useSelector((state)=>state.bookings);
    bookings.map(ele=>{

    })
    const [curDate,setDate] = React.useState(new Date());
    const [currBooking,setBooking] = React.useState(
        bookings.filter((booking)=>{
            const checkDate = new Date(booking.event_day);

            return (booking.building_id === curid && booking.booking_status === "APPROVED"
);
        })
    );
    function handleOnChange(event){
        const val = event.target.value;
        setDate(new Date(val));
    }
    function handleOnSubmit(event){
        event.preventDefault();
        console.log(currBooking);
        setBooking(
            bookings.filter((booking)=>{
                const checkDate = new Date(booking.event_day);
                return (booking.building_id === curid && booking.booking_status ===
"APPROVED" && (checkDate.getDate() === curDate.getDate()));
            })
        )
    }
    console.log(currBooking);
    const classes = useStyles();
    return (
        <>
            <Paper className={classes.paper}>
            {/* <form autoComplete="off" noValidate className={`${classes.root}
${classes.form}`} onSubmit={handleOnSubmit}>
                <Typography variant="h6">
                    Select Date
                </Typography>
                <input type = "date" onChange={handleOnChange} required value =
{today(curDate)}></input>
                <Box mb={1}>
                    <Button variant="contained" color="primary" size="small" type="submit"
>Submit</Button>
                </Box>
            </form> */}
            <FullCalendar
  plugins={[ timeGridPlugin ]}
  initialView="timeGridWeek"
  headerToolbar = {
    {
        start : 'prev,next',
        center : 'title',
        end : 'today',
    }
  }
  weekends={false}
  events= {
    [
        {
      title  : 'event1',
      start  : '2024-01-25'
    },
    {
      title  : 'event2',
      start  : '2024-01-23',
      end    : '2024-01-24'
    },
    {
      title  : 'event3',
      start  : '2024-01-23T12:30:00',
      end  : '2024-01-23T10:30:00',
      allDay : false // will make the time show
    }
    ]
  }
/>

        </Paper>
        </>
    )
}

export default Schedule;
```