Indian Institute of Information Technology, Allahabad

Software Engineering

SOFTWARE DESIGN SPECIFICATION

An Occupancy Monitoring system

GROUP MEMBERS-

Pushkal Madan - IIT2020005

Ritej Dhamala - IIT2020006

Shubham Kumar Bhokta - IIT2020007

Avishkar Singh - IIT2020008

TABLE OF CONTENT

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 behavior models, and other supporting requirement information.

1.1 Purpose of this document

This document will define the design of our web application on occupancy monitoring. It contains specific information about the expected input, output, components, and functions. The interaction between the components to meet the desired requirements are outlined in detailed figures at the end of the document.

1.2 Scope of the development project

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

In Scope:

a. A GUI web Application for computing occupancy of buildings for managing various resources among them in IIITA campus.
b. Admin can see the graphs of occupancy of multiple buildings in a particular time.
c. Admin can view the time and frequency of users entering and exiting a particular building and hence manage the resources of the building like lights, fans, AC's, elevators, etc, thus saving energy consumption.

Out of Scope:

a. Switching ON/OFF the electrical devices.

1.3 Definitions, acronyms, and abbreviations

Acronyms and Abbreviations:

1. "Occupancy Monitoring system" : app name.

Definitions:

1. Buildings- auditorium ,library,cc1,cc2,cc3,etc.
2. "Occupancy Monitoring system" - An web application for buildings for effective energy management.

1.4 References

1.4.1 R. S. Pressman, Software Engineering: A Practitioner's Approach, 5th Ed, McGraw-Hill, 2001.

1.4.2 IEEE SDS template

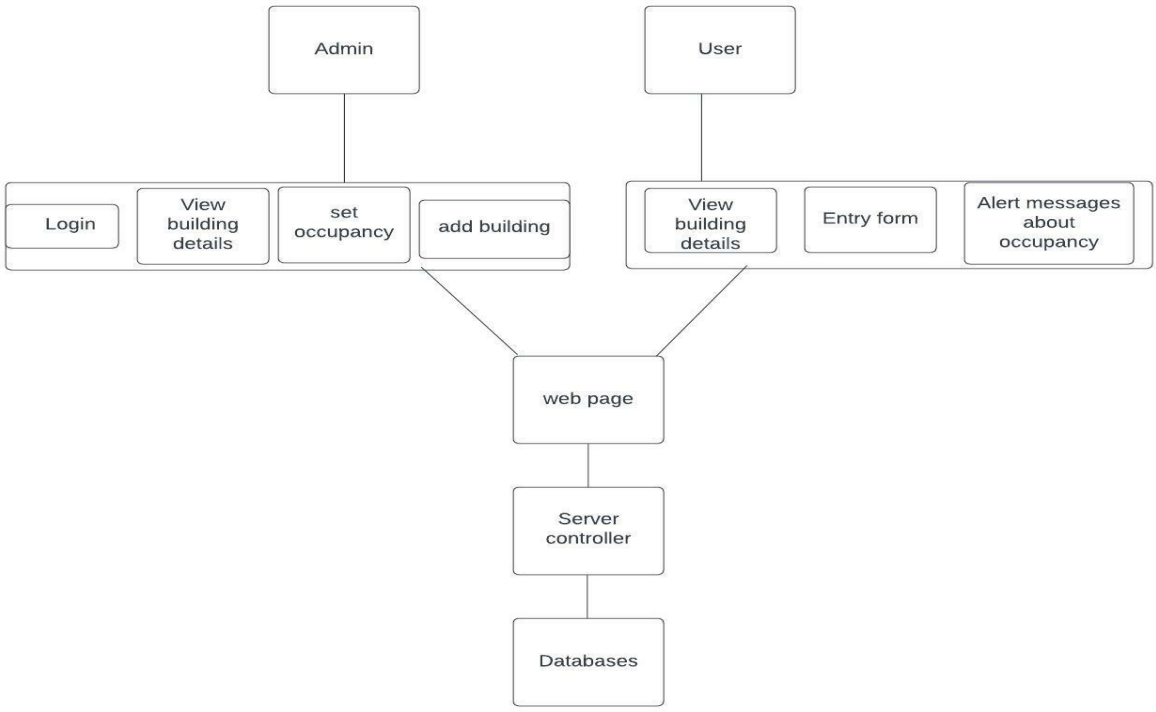
1.5 Overview of document

This SDS is divided into seven sections with various sub-sections. The sections of the Software Design Document are

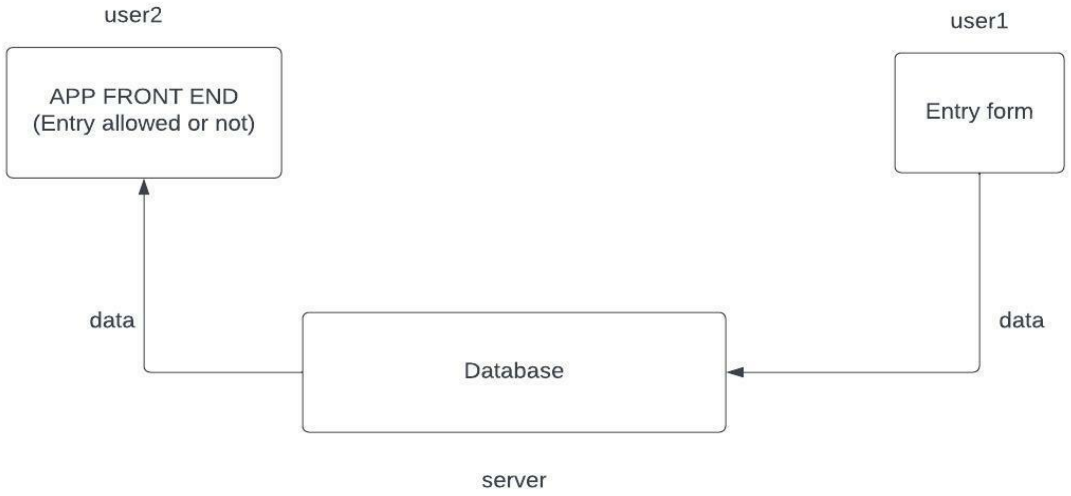
1. **Introduction**: describes about the document, purpose, scope of development project definitions and abbreviations used in the document.
2. **Conceptual Architecture/Architecture Diagram:** describes the overview of components, modules, structure and relationships and user interface issues.
3. **Logical Architecture:** describes Logical Architecture Description and Components.
4. **Execution Architecture:** defines the runtime environment, processes, deployment view.
5. **Design Decisions and Trade-offs:** describes the decisions taken along with the reason as to why they were chosen over other alternatives.
6. **Pseudocode for components:** describes pseudocode, as the name indicates.
7. **Appendices:** describes subsidiary matter if any.

2 Conceptual Architecture/Architecture

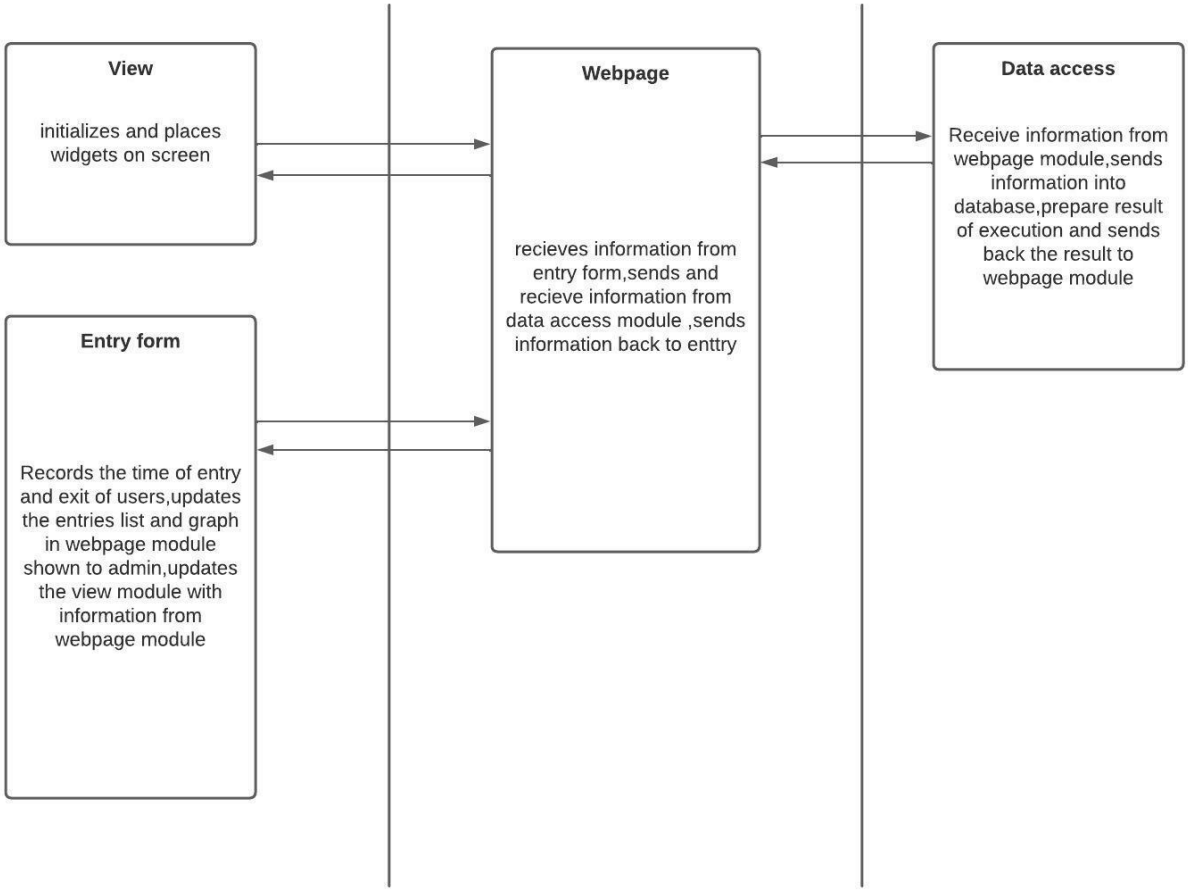
Architecture Diagram 1:



Architecture Diagram 2:



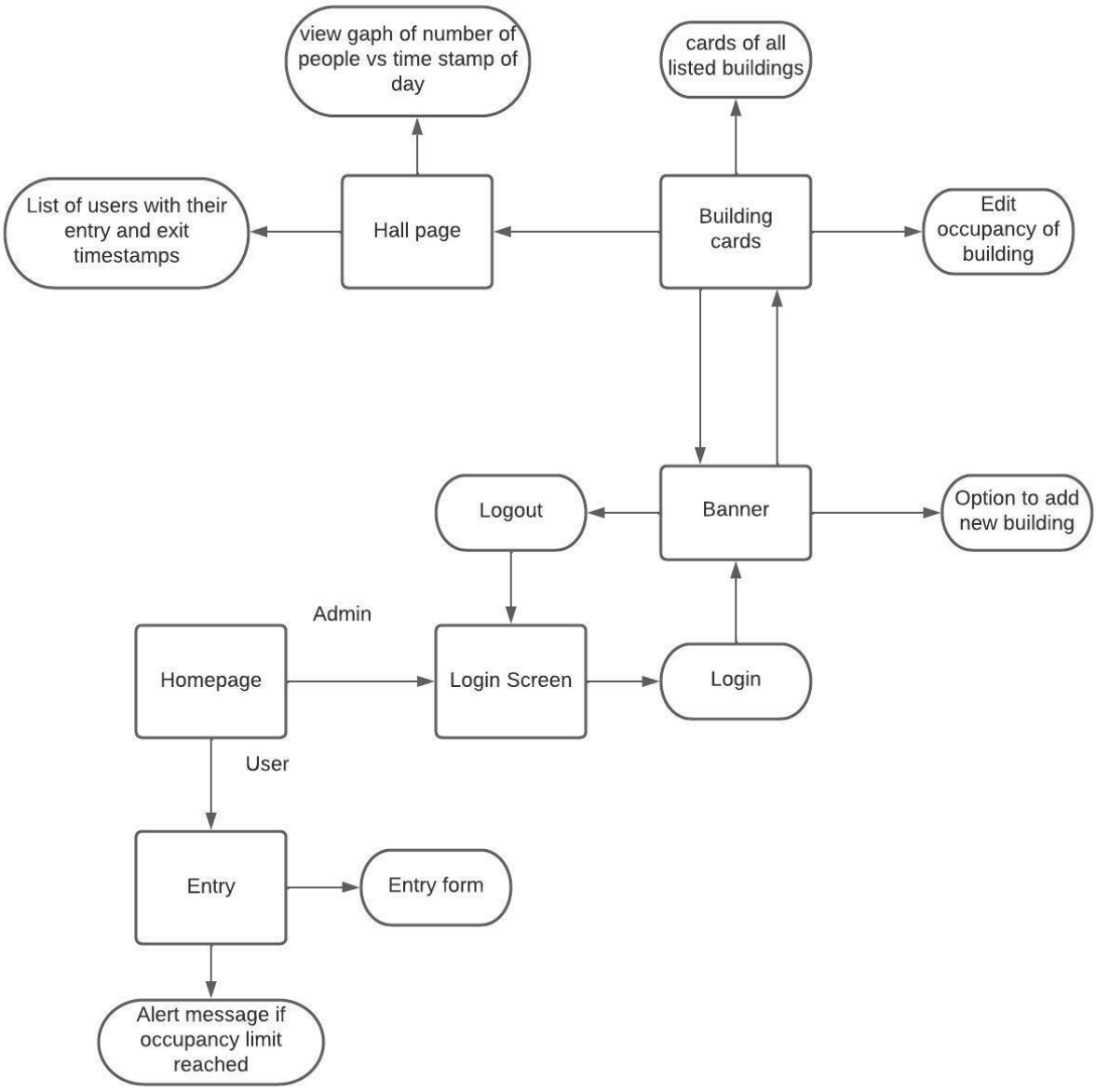
2.1 Overview of modules / components



NOTE:
The horizontal lines represent the separation of modules.

2.2 Structure and relationships

2.2.1 User's/Admin's Side



NOTE:
The boxes represent individual screens.
The circles represent actions that do not have screens. The arrows represent navigation between screens.

2.3 User interface issues

This section will address User Interface issues as they apply to the following hypothetical users

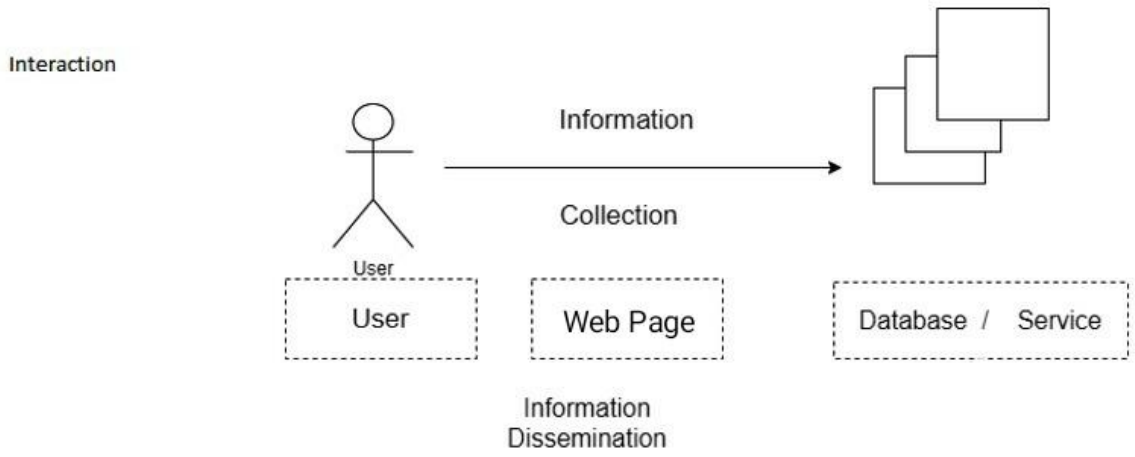
Occupancy Monitoring system .

User A is a 25-40 year old Care-Taker, of a building, who is fairly comfortable with technology. He/She is proficient with using most common computer applications, though not much aware of application development or dealing with problems. He may acknowledge that this application will be super useful for him to manage the electricity and various resources usage in the whole building.

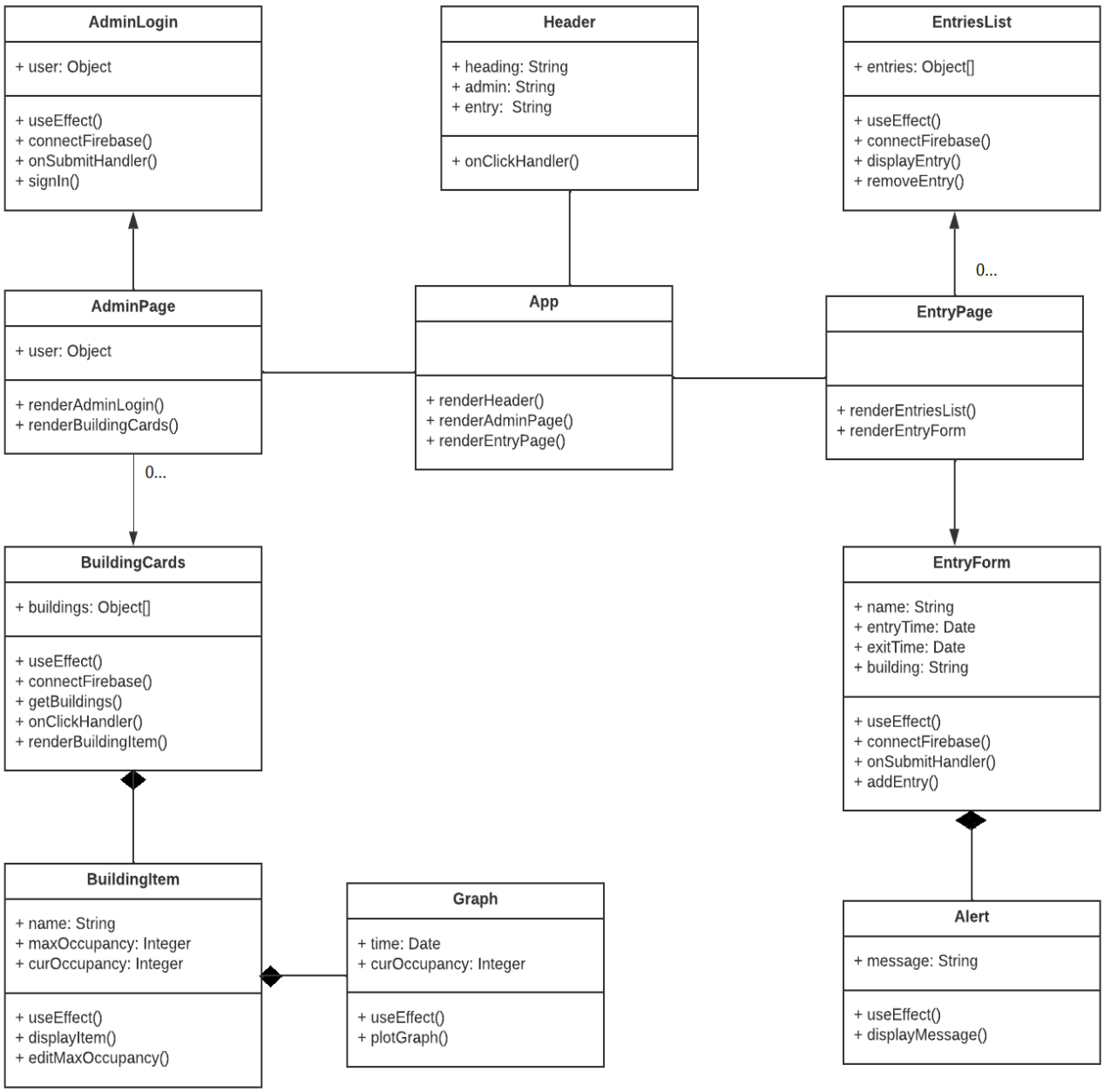
Since User A is familiar with using web applications,Occupancy Monitoring system will use common user interface conventions. For example, links between screens will use ordinary, easy to understand descriptions such as “Login”, ”Add building”,”change occupancy” etc. Color combination has been chosen that allows the user to read all of the text on the screen in direct sunlight. Text size is reasonably larger and, therefore, more readable. Again, information fetching would be a few clicks away with minimal/no extra efforts. The user will just have to open a particular hall/room of that building and the database will use the latest information obtained from entry forms and showcase the data numerically as well as drawing graphs.

3.0 Logical Architecture

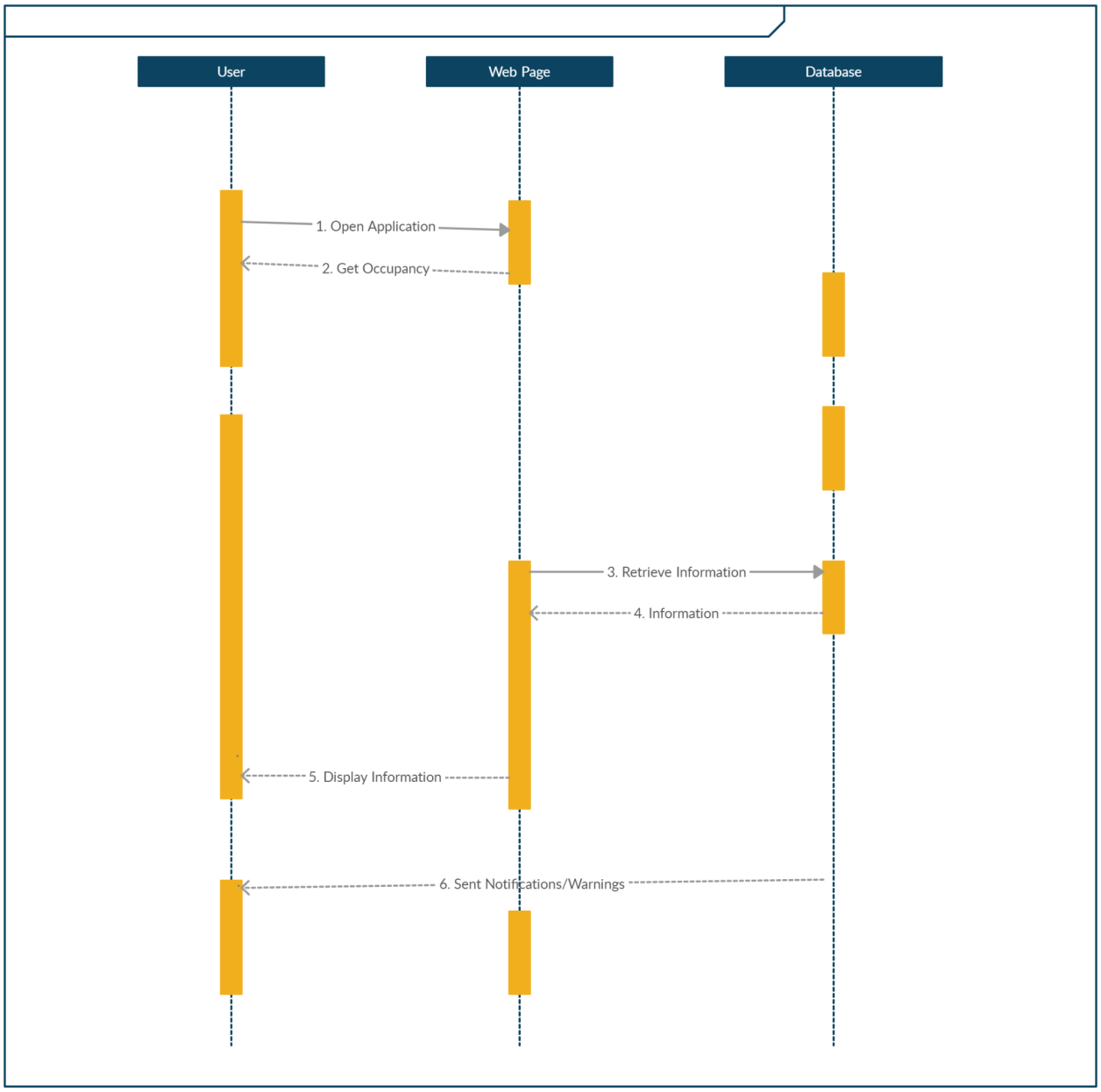
DataFlow Diagram:



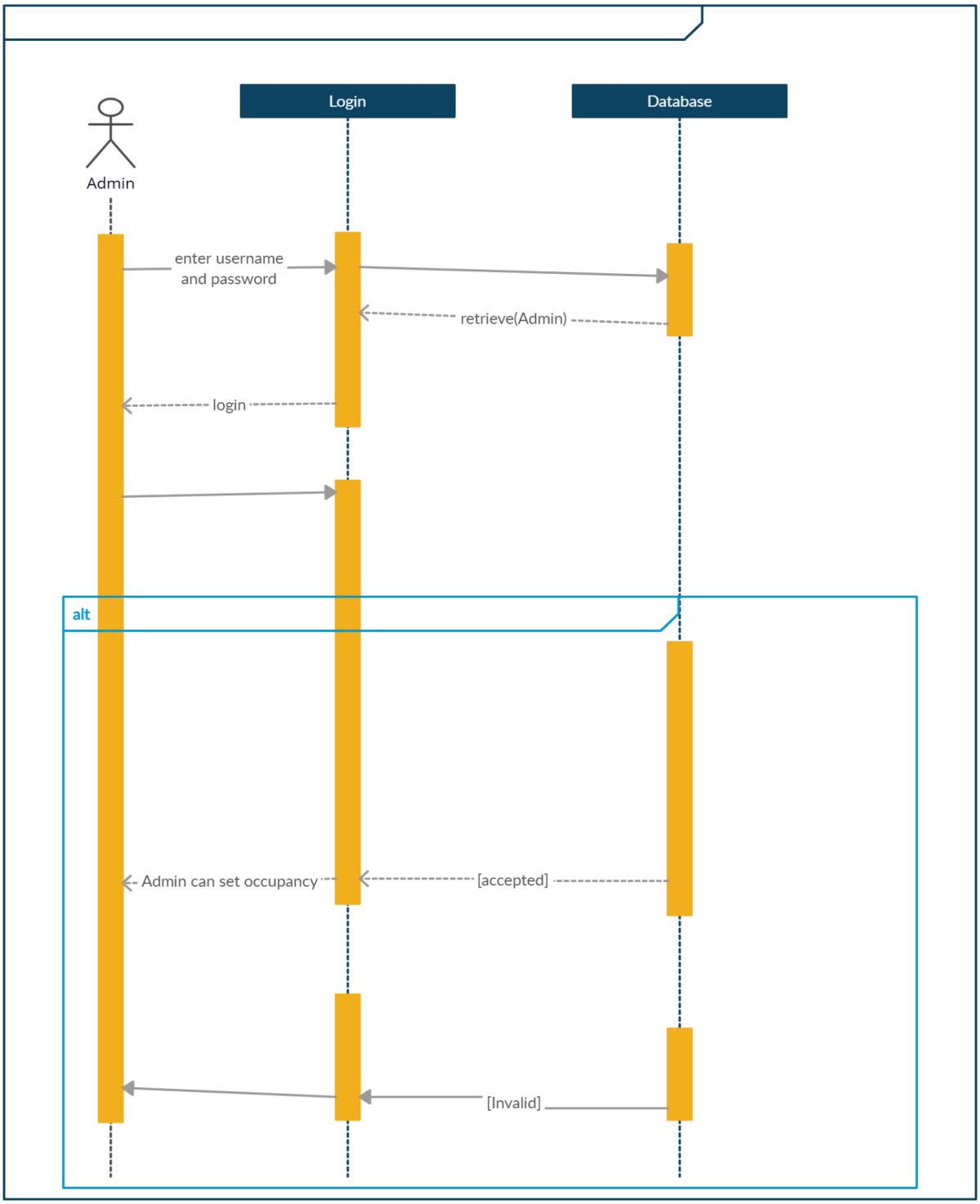
3.1 Class diagram



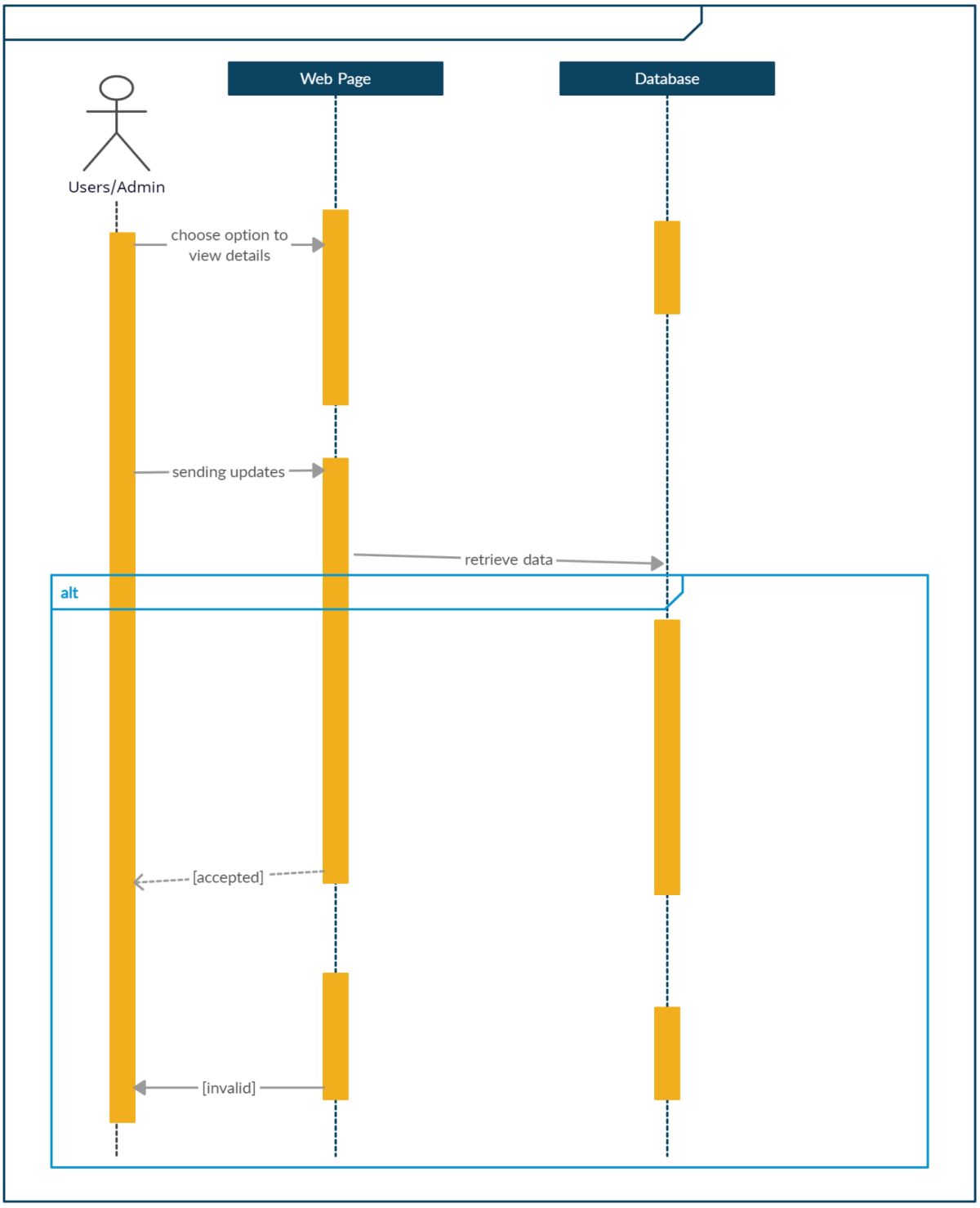
3.2 Sequence Diagrams:



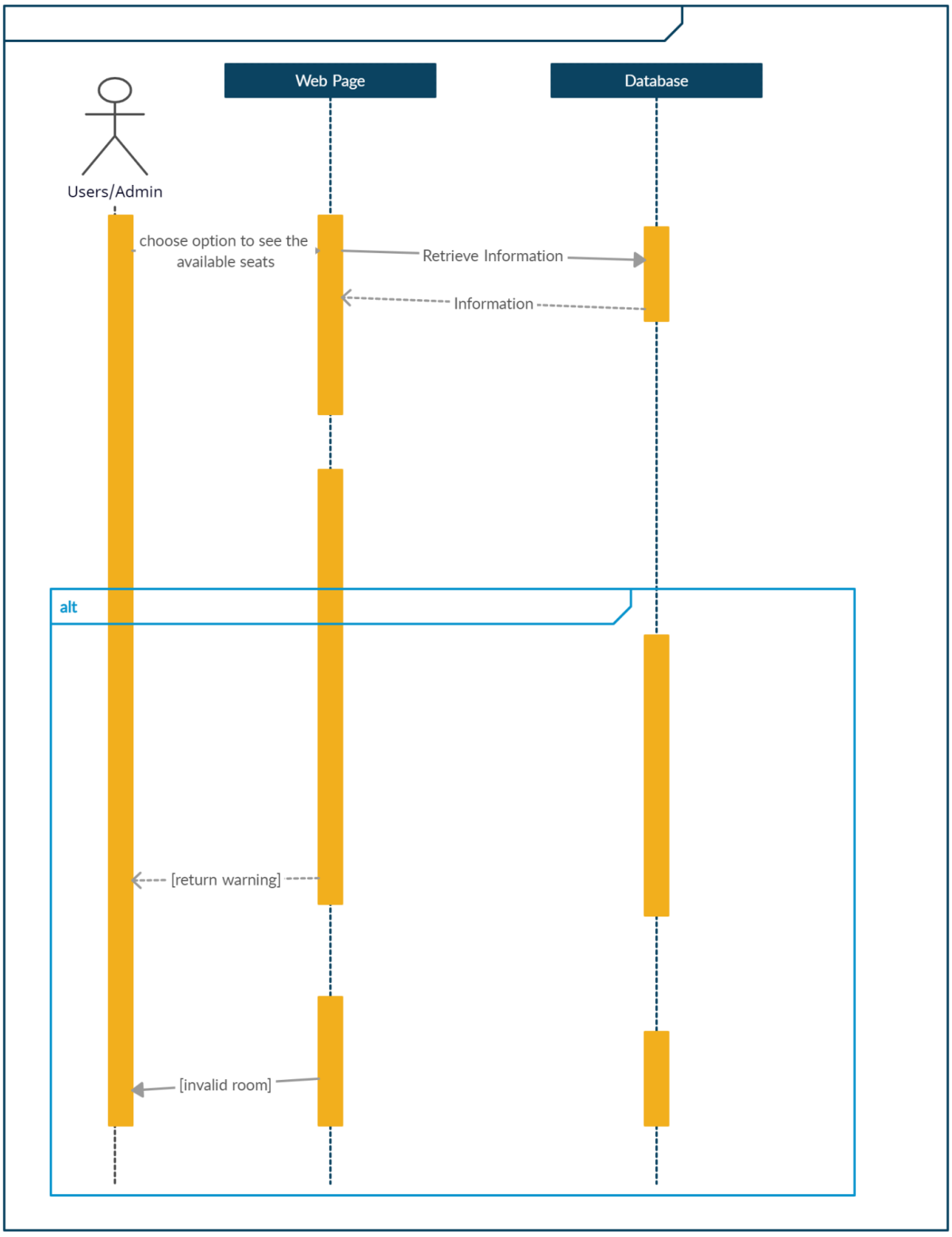
Sequence Diagram: Authentication



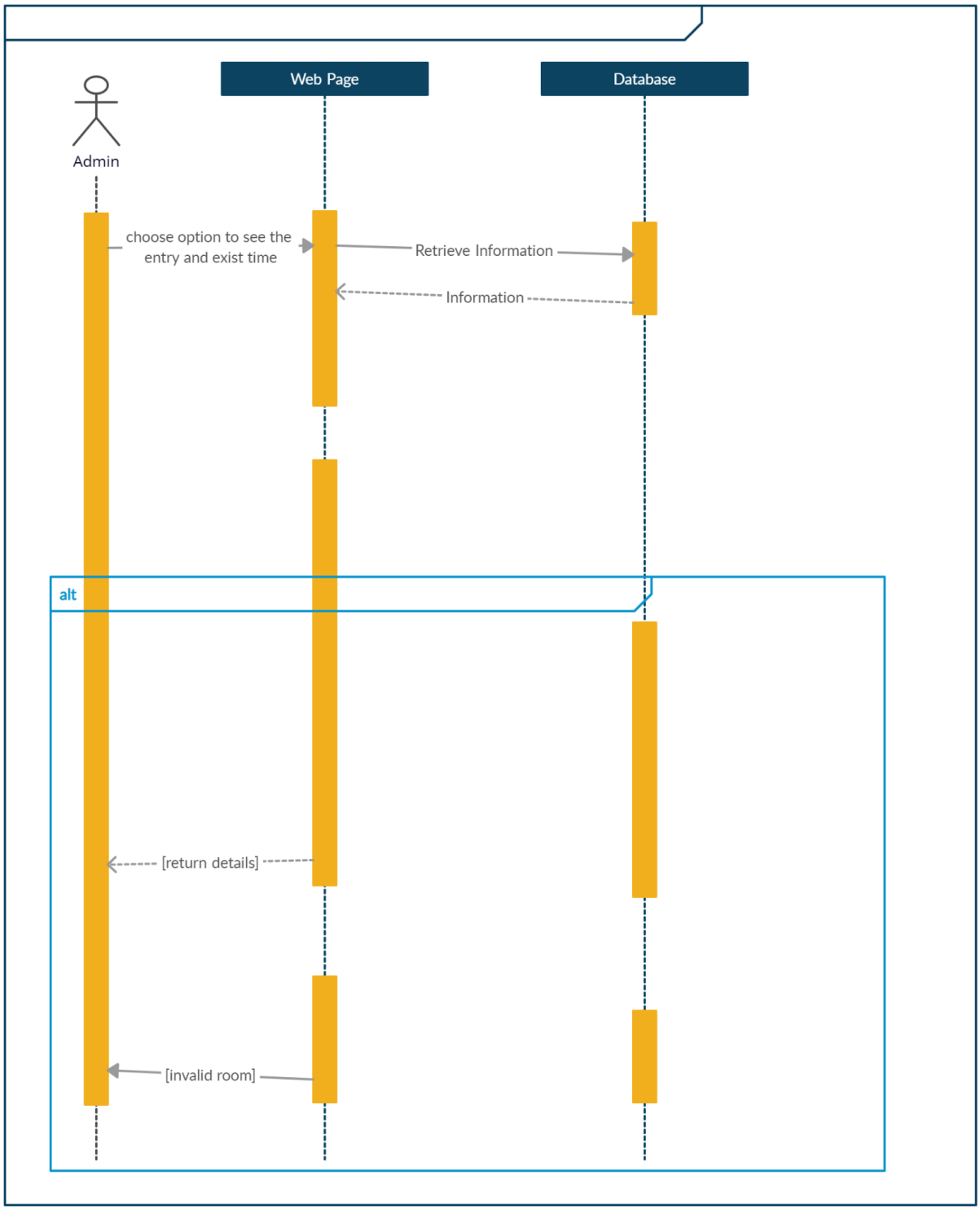
Sequence Diagram: View Building Details



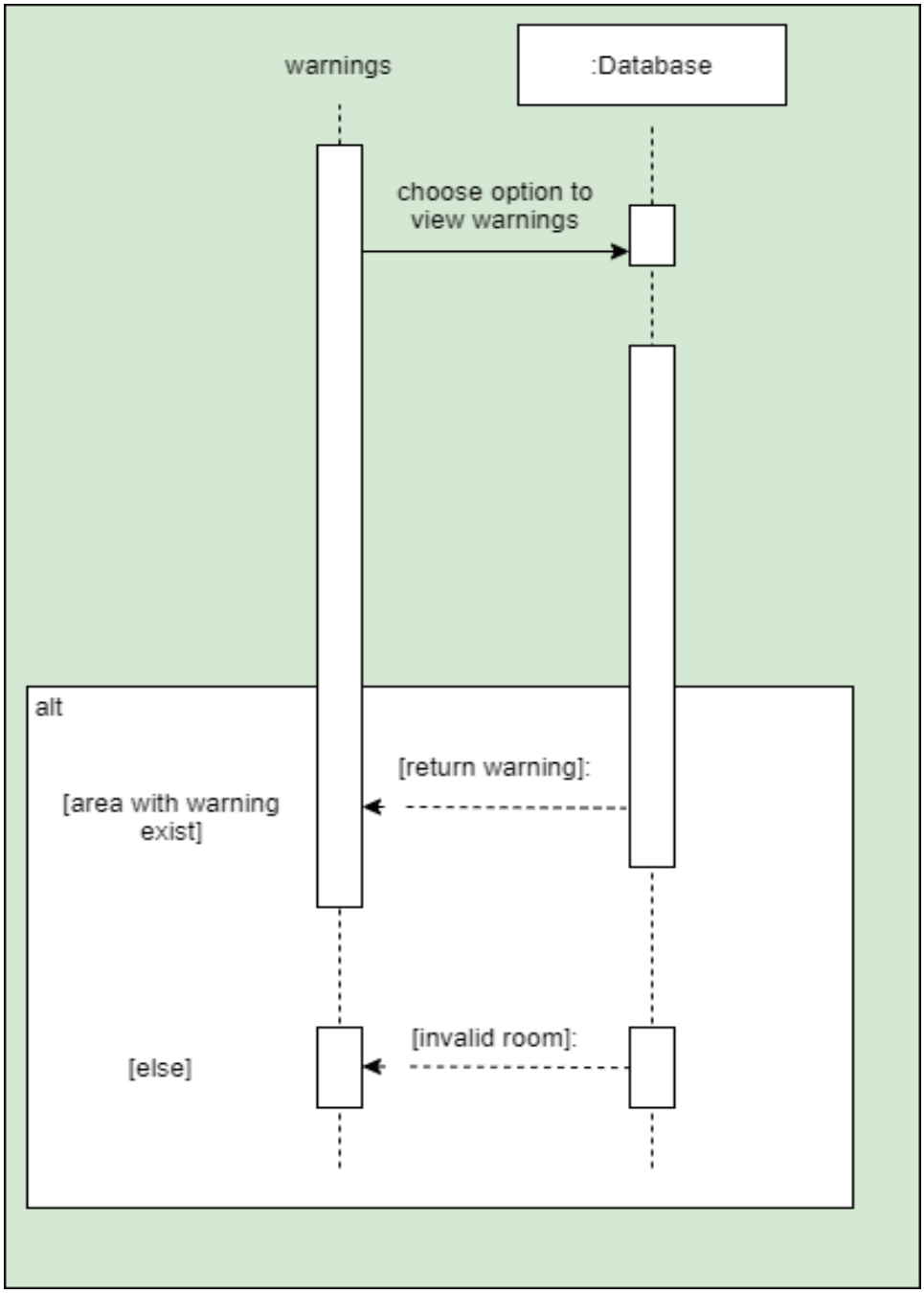
Sequence Diagram: Availability Checking Page



Sequence Diagram: Display Panel Page

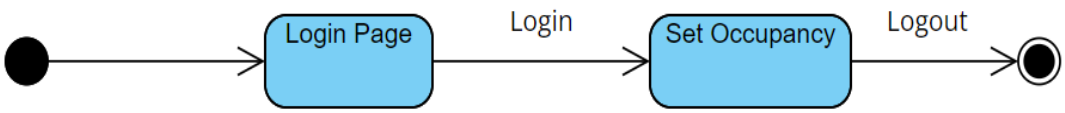


Sequence Diagram: Alert Window



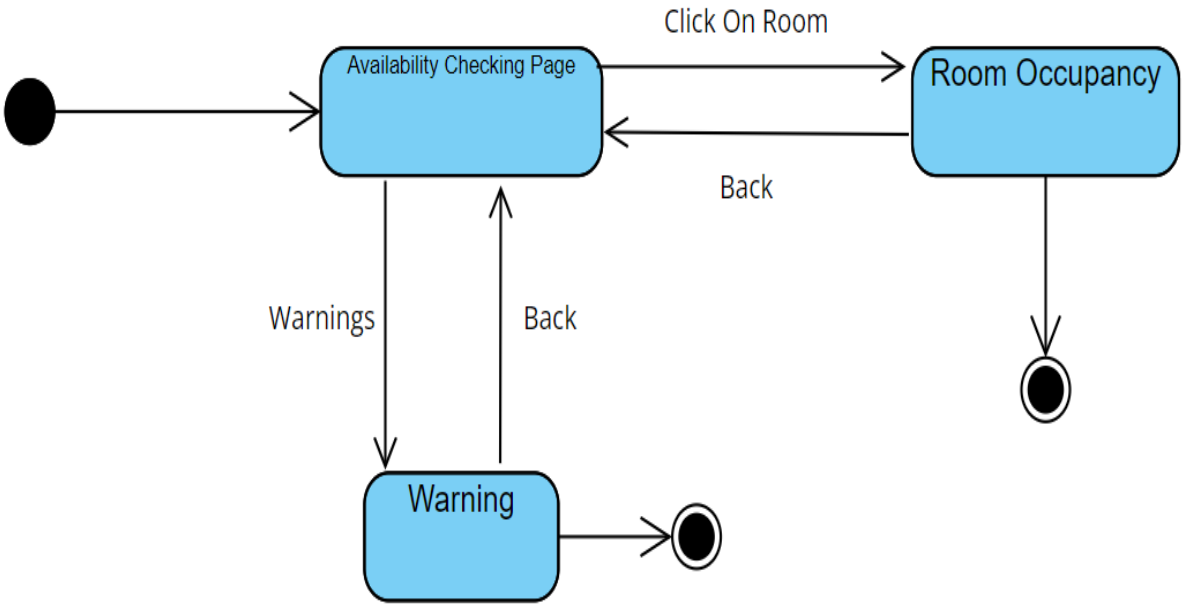
3.3 State Diagrams:

State Diagram: Authentication

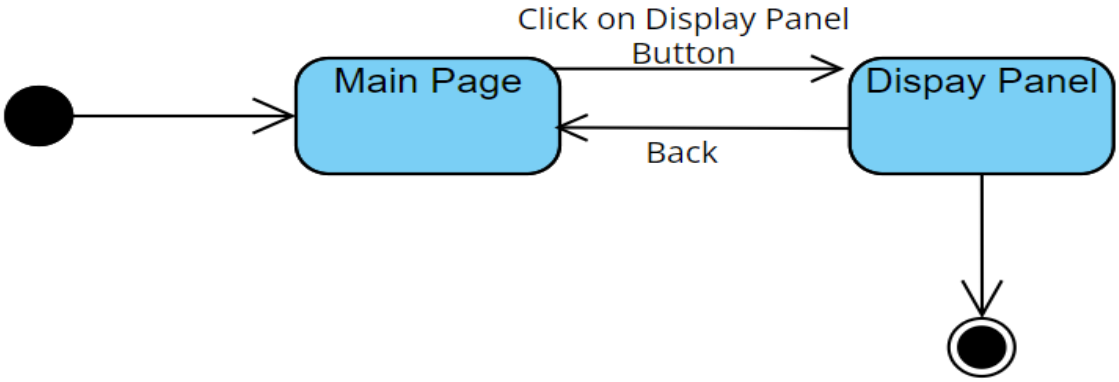


State Diagram: Building Details Page

State Diagram: Availability Checking Page



State Diagram: Display Panel Page



State Diagram: Warnings Page

3.4 Component Description

3.4.1 Component Name: App

Description: This component is the outermost component and all the other components will be built inside this component. It will wrap other components with routes and cover universal styling.

3.4.2 Component Name: Header

Description: This component acts as a Navigation Bar with applications Banner and buttons to change the routes between Entry_Page and Admin_page

Method 1: **onClickHandler()**

Input: event

Output: render the selected component

Method Description: When the user clicks on any of the two button to route into entry page or admin page this method is invoked and the selected component gets rendered into the screen.

3.4.3 Component Name: Admin_Login

Description: In this component admin will be shown an google sign in button so that he/she can be authenticated and upon signing in they will be redirected to Building_Cards component.

Method 1: useEffect()

Method Description: useEffect is generally invoked whenever a particular component is rendered. We can also apply logic to render a component eveytime state of the component is altered.

Method 2: connectFirebase()

Method

3.4.4 Component Name: Entry_Page

Description: the user has to select this component in order to enter a building ,this component will invoke Entry_form component .if the occupancy limit is already reached this component will invoke Alert component.

3.4.5 Component Name: Admin_Page

3.4.6 Component Name: Building_Cards

3.4.7 Component Name: Building_Item

3.4.8 Component Name: Graph

3.4.9 Component Name: Entry_Form

Description: this component will be invoked by the Entry_page component and will record the building,entry time of the user entering a building and will send that information to the database.

3.4.10 Component Name: Entires_List

Description:

3.4.11 Component Name: Edit_Occ

3.4.12 Component Name: Alert

Description: This will give the Users/Admin the details of the warning, i.e., if any room exceeds maximum occupancy this method shows those areas to the Users/Admin.

3.4.3 Component Name - Admin_Login

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

Method 1: onCreate()

Input: savedInstanceState, Username, Password

Output: Launch the Activity, SignIn

Method Description:

When an Activity first calls or launches then onCreate(Bundle savedInstanceState) method is responsible for creating the activity. Whenever orientation(i.e. from horizontal to vertical or vertical to horizontal) of activity gets changed the object of Bundle class will save the state of an Activity. Basically Bundle class is used to store the data of activity whenever the above condition occurs in the application. SetContentView is used to fill the window with the UI provided from the layout file. This method takes input as username and password and as a result opens the landing page if login is successful.

Method 2: Admin_Login()

Input: SignIn through Username and password

Output: Admin landing on the Set Occupancy Page.

Method Description:

Intents are asynchronous messages which allow application components to request functionality from other Web components. This method allows Admin to login through their account. SignIn Intent stores the resultant value by checking whether the sign from Username and password is authorized login or not from the database. If the Admin is not a client then the login will fail and if it is an authorized login then it will lead to a Admin landing page.

3.4.6 Class Name: Building Details

This class basically handles details of the building for the application.

3.2.1 Method1: oncreate()

Input: Listener(Button)

Output: Shows the building details like building name, number of rooms etc.

Method Description:

This method takes care of the new details that this is getting from whenever Admin will update the number of occupancy changes in the building in any area it will immediately update in the building details of the application.

3.2.3 Method3: goback()

Input: Click action on go back option

Output: landing page

Method Description-

This method is used to go back to the login page the user is in. By clicking this method the user will be able to redirect to the Main Page.

3.3 ClassName: Availability Checking

Class Description:

This class allows to check the availability of seats in that building

3.3.1 Method 1:oncreate()

Input: Button_Listener

Output: Shows present occupancy of every room in the form of cards.

Method Description:

It fetches data from the database, which has been regularly been put by Admin, and then plots the data in the cards (consisting of details of rooms).

Method 2: **show_warnings()**

Input: Click Action on warnings option.

Output: User is landed on the warnings page.

Method Description:

By clicking this method a warning will appear to the users.

3.4 Class Name: Room_x occupancy

Class Description:

This class will contain the occupancy in any particular room (for example, occupancy in room no. 1).

3.4.1 Method1:onCreate()

Input: Bundle object and button clicks.

Output:

Method Description:

It fetches data from the database, which has been regularly been put by Admin, and then plots the data into graphs of No. of people Vs. Time Stamp. Here we will plot the data with a timestamp of each hour of a day, for one week. So that we get a fair idea of occupancy of a particular room for the whole week, and displays on the screen.

3.4.2 Method2:goBack()

Input: Click Action on the go back option.

Output: User lands on the Main Page.

Method Description:

By clicking on this method the user will be redirected to the previous page. Users will be redirected to the Main Page.

3.5 Class Name: Warning

Class Description:

This will give the Users/Admin the details of the warning, i.e., if any room exceeds maximum occupancy this method shows those areas to the Users/Admin.

3.5.1 Method 1: onCreate()

Input: Bundle object and button clicks.

Output:

Method Description:

This method fetches data from the database and shows the warnings in ListView.

3.5.2 Method 2: goBack()

Input: Click action on go back option.

Output: Users/Admin will land to the Main Page.

Method Description:

By clicking on this method the Users/Admin will be redirected to the previous page. Users/Admin will be redirected to the Main Page.

6. Pseudocode for components

6.1 Class Name: App

function App () {
render jsx:
<BrowserRouter>
<render Banner>
<Switch>
    <Route to **Admin_Page**>
    <Route to **Entry_Form**>
</Switch>
</ BrowserRouter>
}

6.2 Class Name: Banner

function() {
render jsx:
<>

```
<div>
<h1>Occupancy Monitoring System</h1>
</div>
<button>
<Link to: Admin_Page /> Admin
<button>
<button>
<Link to: Entry_Page /> Entry
<button>
</>
}
```

6.3 Admin_login

```
function() {
const handleSubmit = () => {
        auth.signInWithPopup(provider).then((res) => { user=res.user})
.catch(err => alert(err.message));
}

render jsx:
        <>
<button onclick={handleSubmit}> Sign-in with Google <button>
</>
}
```

6.4 Entry_Page

```
function() {
const [users, setUsers] = useState([]);
const usersColletion = collection(db, "users");

 useEffect(() => {
    const getColletion = async () => {
      const data = await getDocs(usersColletion );
      setUsers(data.docs.map((doc) => ({ ...doc.data(), id: doc.id })))
    };
    getColletion ();
  }, [users]);

render jsx:
```

```
        <>
                <div>
        {users.map((user) => {
          return (
            <div key={user.id}>
              < Entries_List
name={user.name}
entryTime={user.entry}
exitTime={user.exit}
       />
    </div>
  )
                </div>
                <render Entriy_Form>
</>
}
```

6.5 Admin_Page

```
function() {
render jsx:
        <render Building_Cards>
}
```

6.6 Building_Cards

```
function() {
const [buildings, setBuildings] = useState([]);
const buildingsColletion = collection(db, "buildings");

 useEffect(() => {
     const getBuildings = async () => {
       const data = await getDocs(buildingsColletion);
       setBuildings(data.docs.map((doc) => ({ ...doc.data(), id: doc.id })))
     };
     getBuildings();
   }, [buildings]);

render jsx:
        <div>
        {buildings.map((building) => {
          return (
```

```
            <div key={building.id}>
              <Building_Item
name={building.name}
max={building.maxocc}
cur={building.curocc}
id={building.id}
      />
   </div>
 )
          </div>
}
```

6.7 Building_Item

```
function(building) {
render:
        <>
        <div key={building.id}>
                <img src=”assest->building image.png” />
                <div>
                        {building.name}
                </div>
<div>
                        Maximum Occupancy: {building.max}
                        Current Occupancy: {building.cur}
                </div>
                <Edit_Occ id={building.id} />
        </div>
        <Graph />
}
```

6.8 Component Name: Graph

```
function() {
const [users, setUsers] = useState([]);
const usersColletion = collection(db, "users");

 useEffect(() => {
    const getColletion = async () => {
      const data = await getDocs(usersColletion );
      setUsers(data.docs.map((doc) => ({ ...doc.data(), id: doc.id })))
    };
    getColletion ();
```

```
  }, [users]);

render:
        <Recharts x-axis:timestamp y-axis:number of user>
                {map(user.timestamp) => {
                        plot data
                }
</Recharts>
}
```

6.9 Component Name: Entry_Form

```
function() {
const [name, setName] = useState('');
const [building, setBuilding] = useState('');

const entryCollection = collection(db, "entries");

const handleSubmit = async (e) => {
    e.preventDefault();
    console.log(name, building);
    await addDoc(entryCollection, {
      name: name,
      building: building,
      entry: serverTimestamp()
    });
};

render:
    <div>
      <form >
        <input
          onChange={(e) => setName(e.target.value)}
          placeholder="Enter your name!"
        />
        <input
          onChange={(e) => setBuilding(e.target.value)}
          placeholder="Enter the building name!"
        />
        <button onClick={handleSubmit} type="submit">Enter</button>
      </form>
    </div>
```

```
}
```

6.10 Component Name: Entires_List

```
function(user) {
render:
        <div>
                Name: {user.name}
                EntryTimestamp: {user.entryTime}
                ExitTimestamp: {user.exitTime}
        </div>
}
```

6.11 Component Name: Edit_Occ

```
function() {
const [occ, setOcc] = useState(0);
const handleclick = (id}) {
        const buildingDoc = doc(db, "buildings", id);
        const newFields = { curocc: occe };
        await updateDoc(buildingDoc, newFields);
}
render:
        <input placeholder="Change maximum Occupancy"
                onChange={(event) => {setOcc(event.target.value) }}
        />
        <button onclick={() => {updateOcc(building.id)}>
Change maximum occupancy
</button>
}
```