# CERTIFICATE

This is to certify that the project entitled

**"SOCIETY HUB"**

Has been satisfactorily completed by

1) Ashwini Kanse
2) Aniket Jadhav
3) Akshay Shetty
4) Shashank Surve
5) Rutuja Fale

Under supervision and guidance for partial fulfillment of the

**PG Diploma in Advanced Computing (DAC)**

**(24 weeks FullTime) Course**

of

**Centre for Development of Advanced Computing (C-DAC), Pune.**

at

**Academy of Information Technology**

**(YCP) Nariman Point, Mumbai – 400 021.**

**Faculty** **Course Co-ordinator**

Batch: PG DAC March 2024 Batch
Date: August 2024.

# ACKNOWLEDGEMENT

I wish to extend my deepest gratitude and special thanks to my project guide, for giving their generous support, necessary inputs and companionship during my project work.

I would like to convey my special thanks to the Management and all the staff of the college for providing the required infrastructure and resource to enable the completion and enrichment of my project.

I am extremely grateful to the CDAC Mumbai for having prescribed this project work to me as a part of the academic requirement.

Finally I thank all my fellow friends who have directly or indirectly helped me in completing my project.

# Introduction

**Society Management System** is provided to society members who can get all the updates related to their society. The members also get notified with notices and events held in society and can see information about members in society. Members can also post complaints regarding any issue in society. The only admin has right to modify the database which holds information of members. Also, the admin can perform tasks  that are done by the normal members of this system.

## 2.1 Objective of the Project:

Objective of Society Management System is to manage all day to day society works, tasks, and for maintaining information about flats, set up reminders for events, and also to maintain complaint register about society and flats in society.

## 2.2 Description of current system:

In the current framework, all the work is done physically. The notice board is traditional. Hence there is problem of unreachable information & more time consuming. No attention to the individual complaints about flat issues.

## 2.3 Limitations of current system:

Daily notices, monthly meetings, cultural events, miscellaneous contacts for daily needs, high priority communication and many others which may not be conveyed properly in current scenario as most of the things are getting handled manually, and this lacks transparency and creates conflicts between members.

## 2.4 Description of Proposed system:

In the proposed system, virtual notice board will be implemented so that notices can be seen from anywhere and will be available to everyone through website. There will be two logins admin and member to ensure the authenticity of things. This will make the entire process paperless and easy to handle by everyone.

## 2.5 Advantages of Proposed system:

To get attention and notify to important issues and complaints about existing problems (Water, cleanliness, technical works).To maintain transparency between society members and management and to minimize the paperwork load.

It also helps to reduce the efforts for manually communicating with each  society member for providing notifications and important information. To keep track society members through database stored. It also helps to manage events, complaints, and to keep society well organized and secure.

# <u>Requirement Specification.</u>

## 3.1 Software Requirement:

- Windows operating system
- Any browser preferably Google Chrome.
- Springboot ,VScode, Eclipse-IDE, MYSQL database

## 3.2 Hardware Requirement:

- Server with minimum 2 GB space
- 4 GB ram

## 3.3 Data Requirement:

- Username
- Password
- Email id
- Mobile number
- Flat number

## 3.4 Fact Finding Questions:

- What work does this system do overall?
- How many people can play a role of admin in this system?
- Who will be managing all works in this system?
- Is our data on this system secured?
- What is the limit for number of members in the society?
- How can I update my information?
- Can admin use this system as normal user?
- I still have some more queries, where do I contact?

## System Design Details

### 4.1 Event Table:

| No. | Event | Trigger | Source | Activity | Response | Destination |
|---|---|---|---|---|---|---|
| 1. | Register now | form-container | User | Creates new account | Adds new account to db, redirect to home | Server |
| 2. | Login | form-container | User | Checks username and password to login | Redirect to home | Server |
| 3. | Admin | btn | User | Login to admin page | Redirect to admin | Server |
| 4. | User wants to view notice board | View notice | User | Click on to view notice button | Notice section | Local |
| 5. | User wants to submit complaint | Btn-Complaint | User | Submit complaint of user | Add complaint to db, redirect to home | Server |
| 6. | User wants to upload photos | Btn-upload | User,admin | Click to upload photo | To view photo gallery | Server |
| 7. | Logout | Btn logout | User | Logout from account | Redirect to login page | Server |

| | | | | | | |
|---|---|---|---|---|---|---|
| 8. | Add member | Btn-add | Admin | Click to add newMember | Redirect to manage members | Server |
| 9. | Update member | Btn-update | Admin | Click to update member | Redirect to manage members | Server |
| 10. | Delete member | Btn-Delete | Admin | Click to delete member. | Redirect to manage members | Server |
| 11. | Add notice | Btn-notice | Admin | Click to add notice. | Redirect to add notice | Server |
| 12. | View complaints | Btn-complaints | Admin | Click to view complaints | Redirect to View Complaints | Server |
| 13. | Logout | Btn-Logout | Admin | Click to logout | Rdirect to admin login page | Server |

## 4.2 Class Diagram:

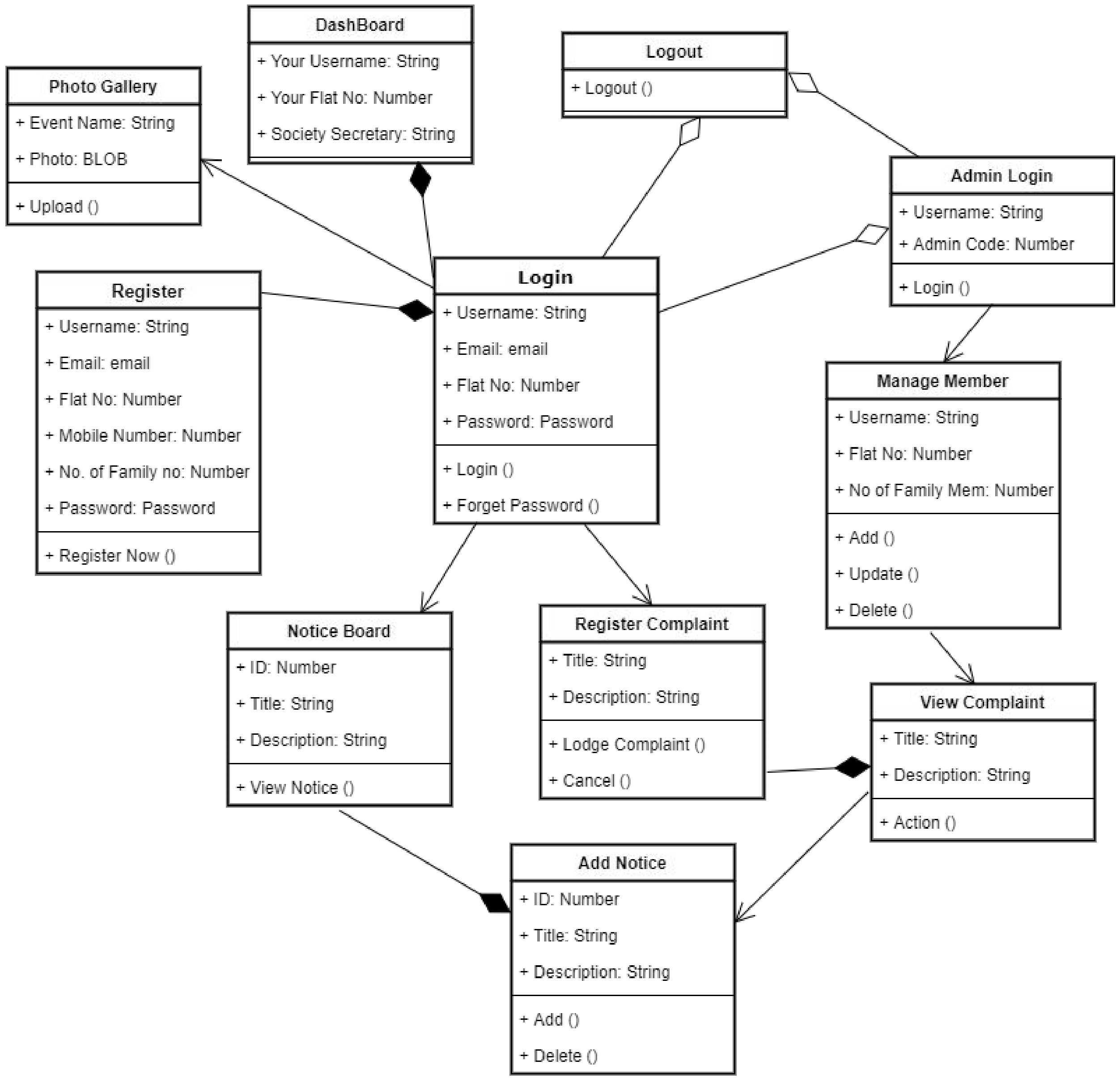


Figure 4.2: flow chart

## 4.3 Use Case Diagram:

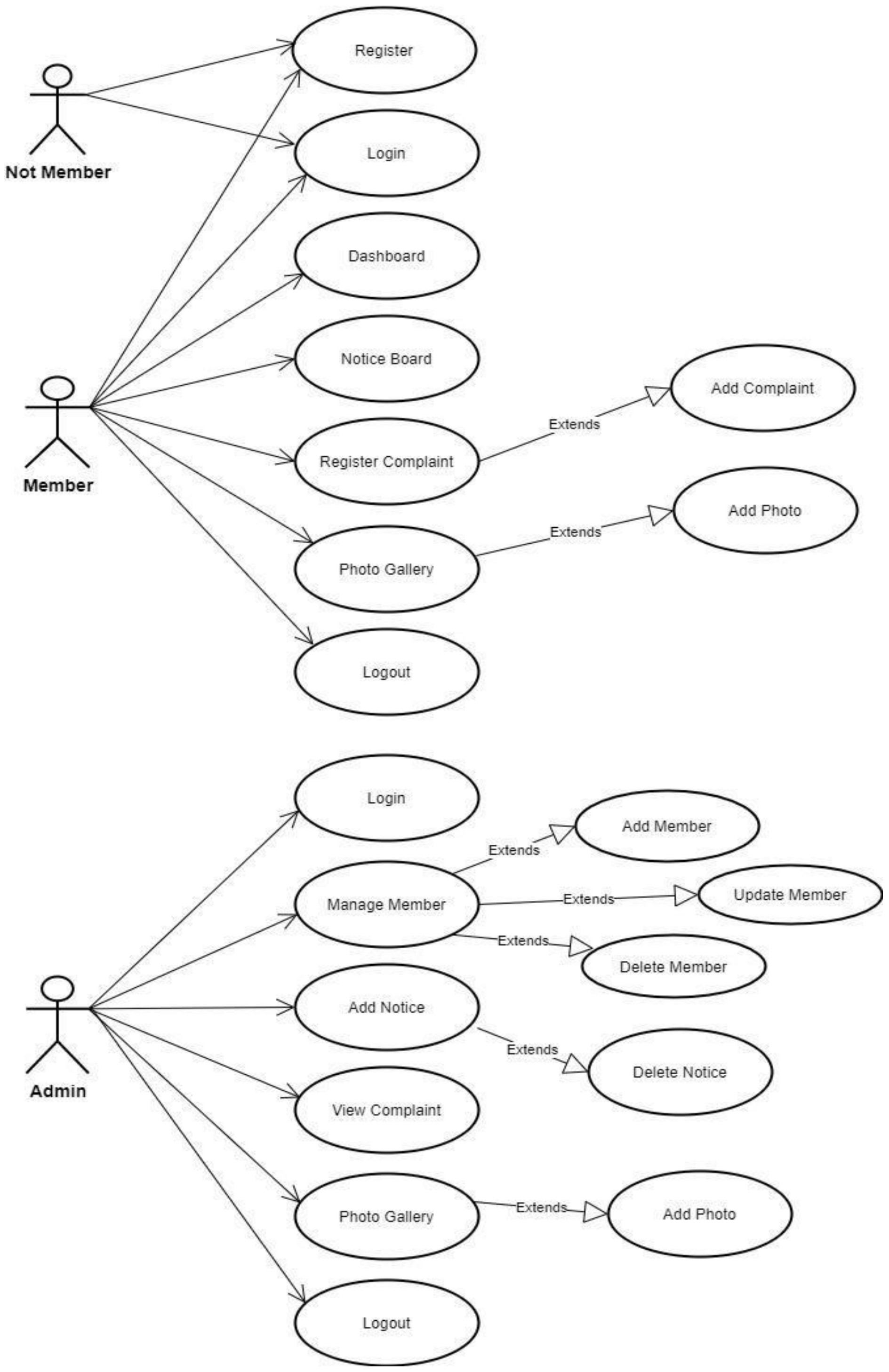


Figure 4.3: Use Case Diagram

## 4.4 Sequence Diagram:

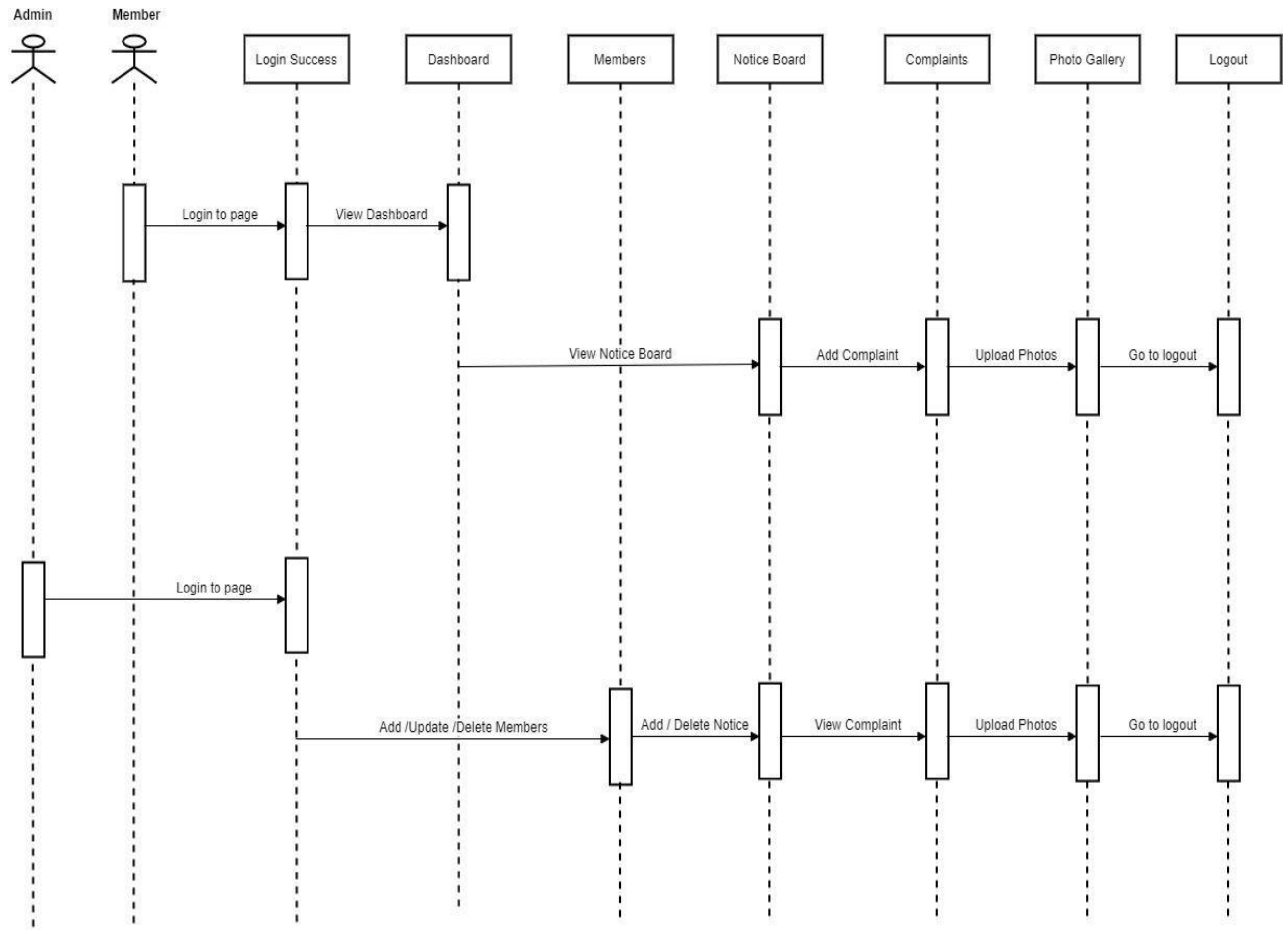


Figure 4.4: Sequence Diagram

## 4.5 Activity Diagram:

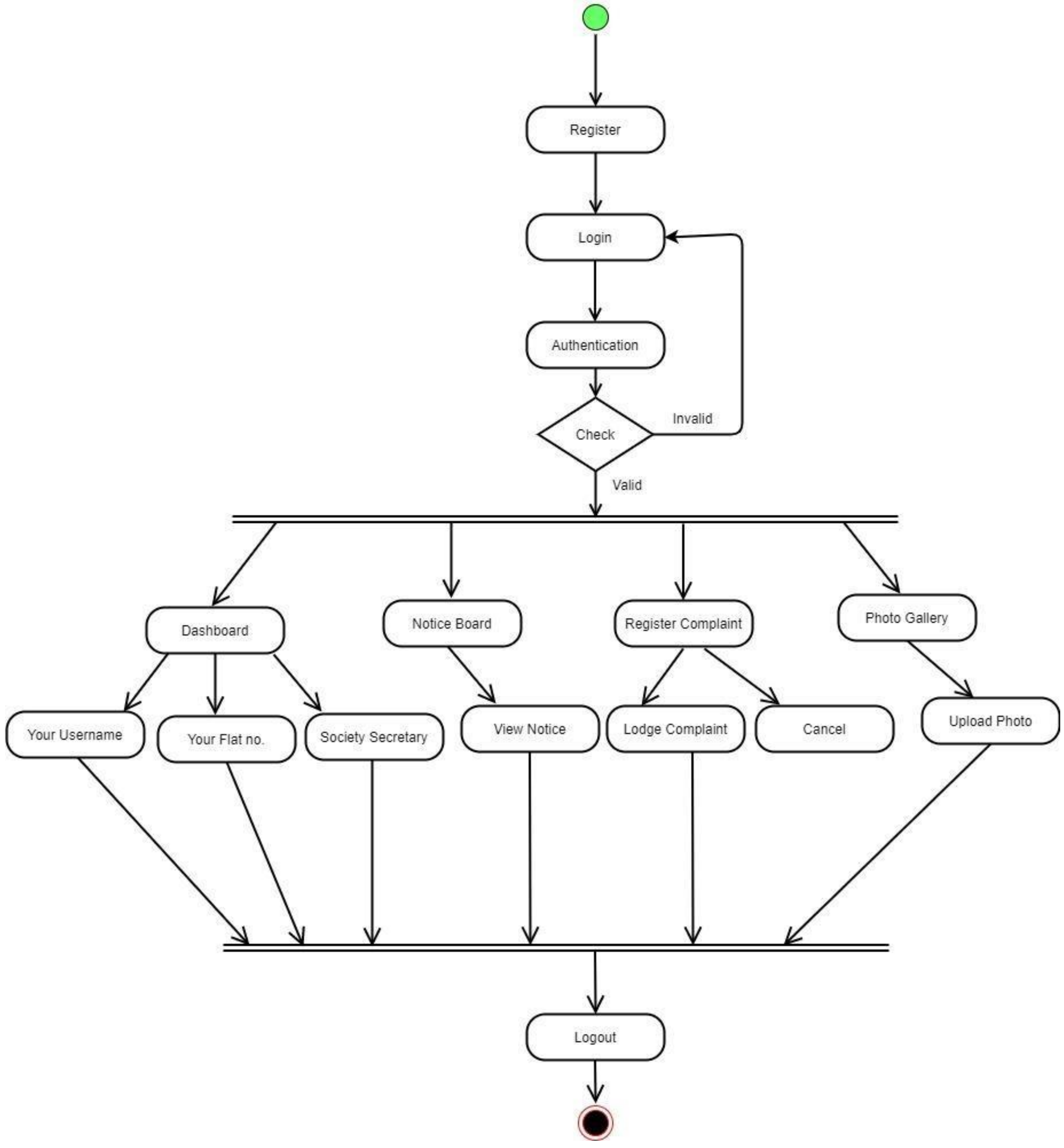


Figure 4.5: Activity Diagram for User

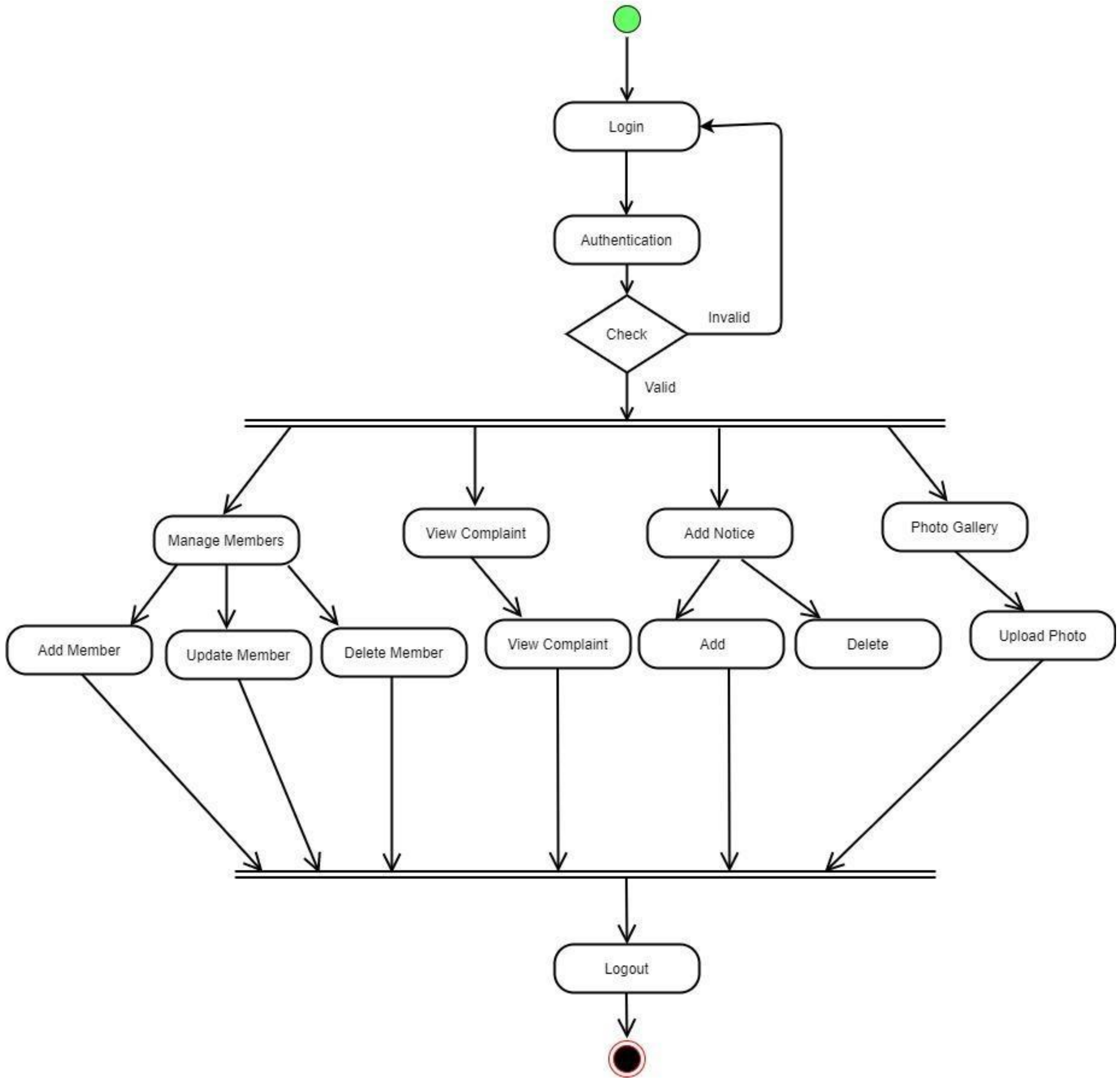


Figure 4.5: Activity Diagram for Admin

## 4.6 State Diagram:

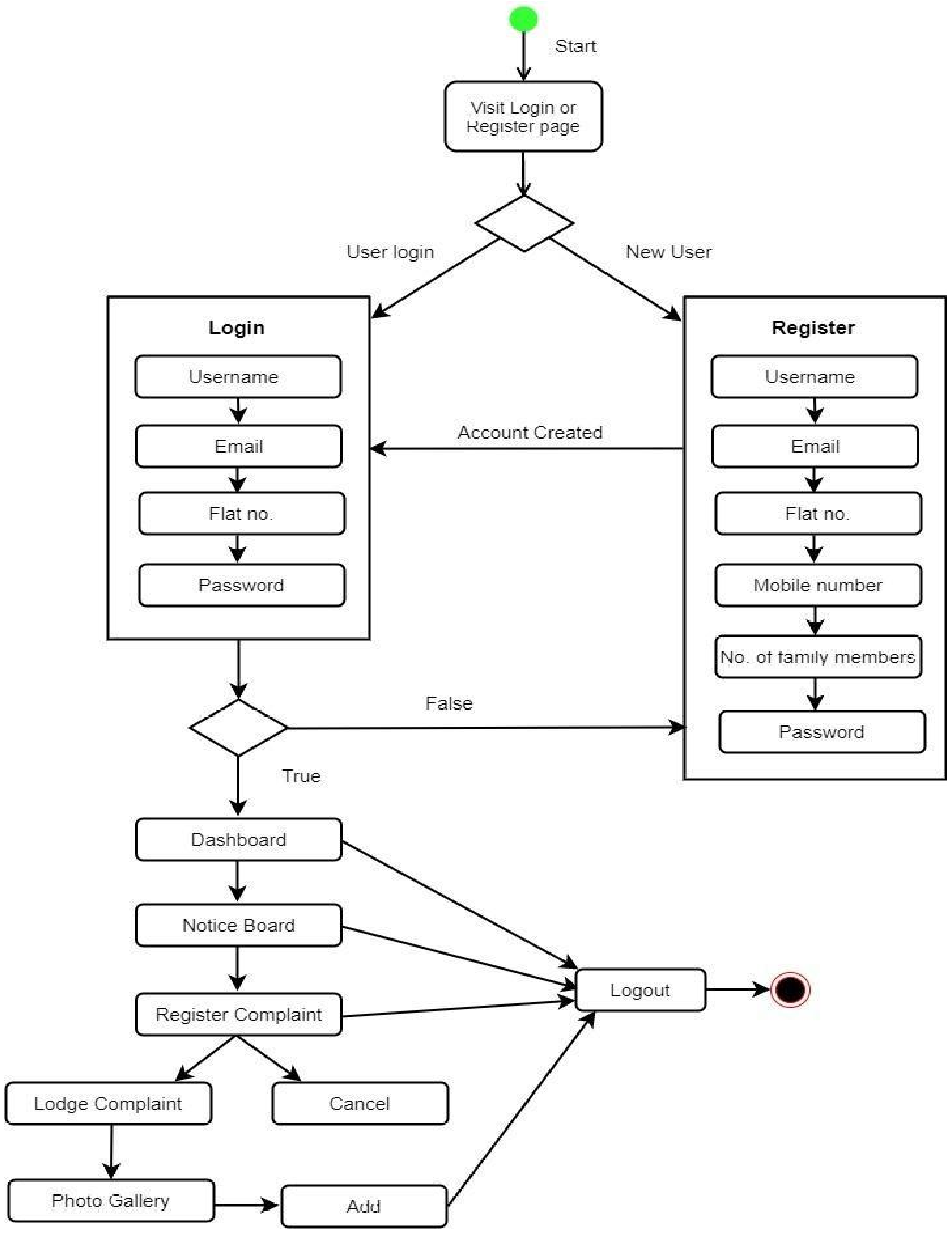


Figure 4.6: State Diagram User

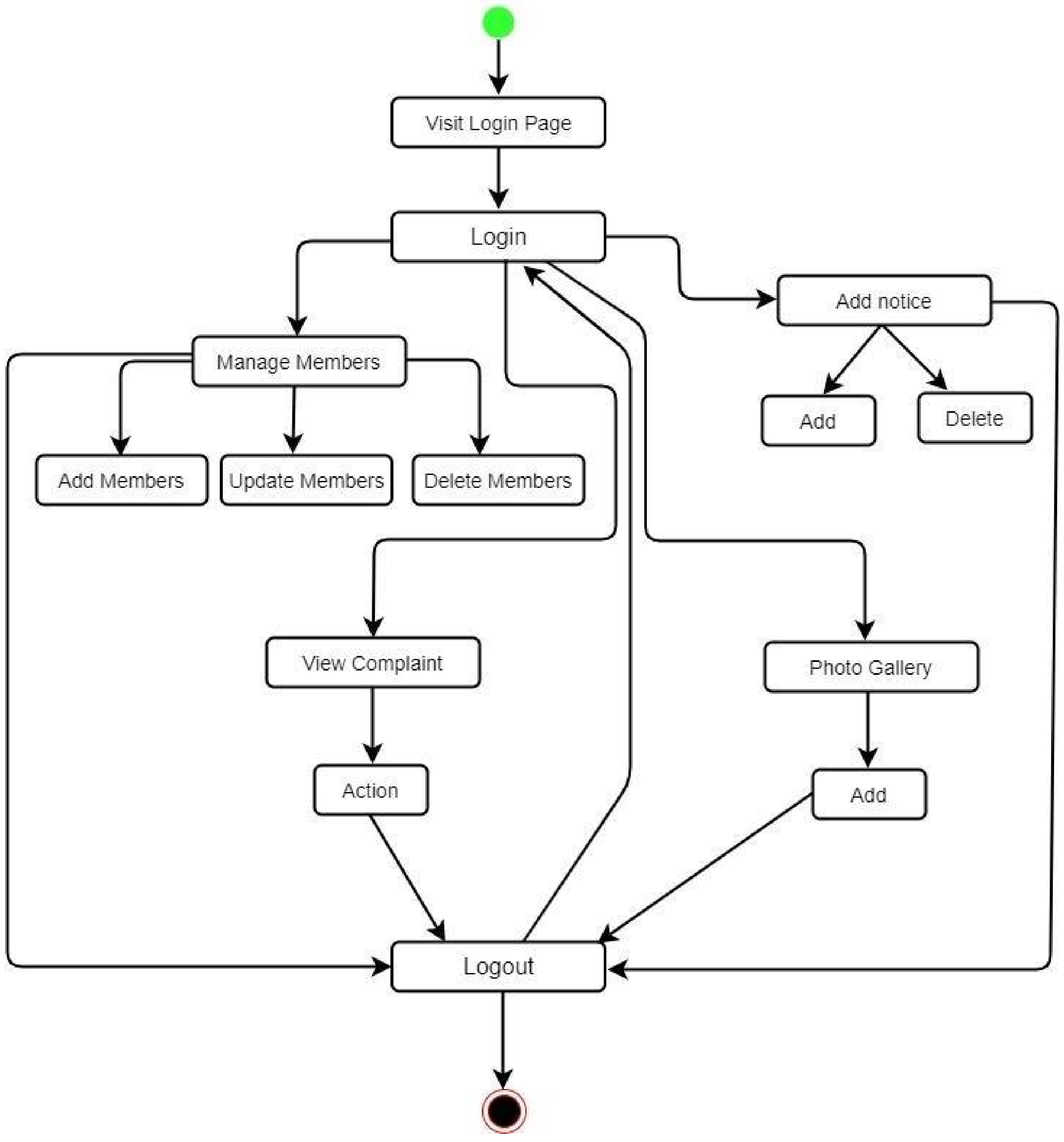


Figure 4.6: State Diagram for Admin

## 4.7 Component Diagram:

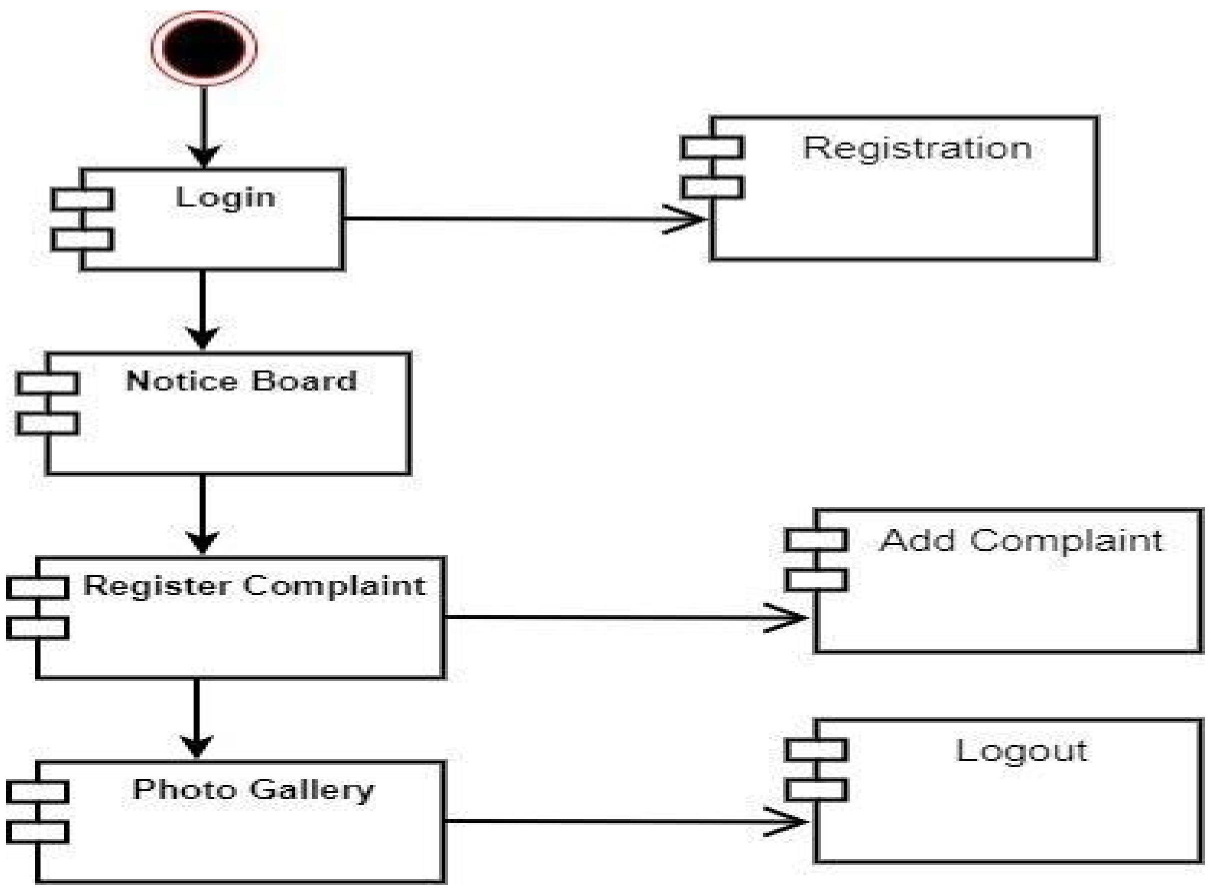


Figure 4.8: Component Diagram

## 4.8 Deployment Diagram:

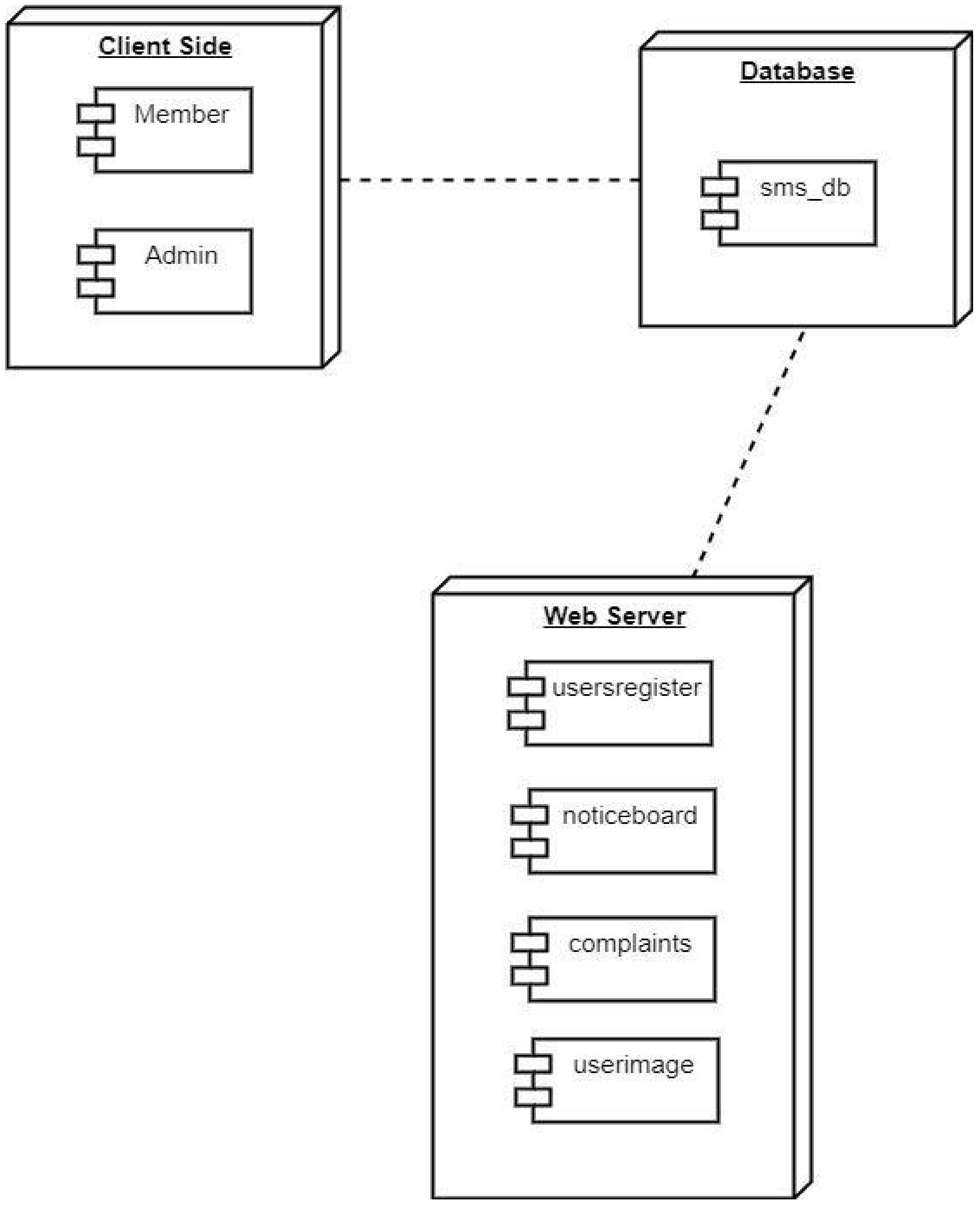
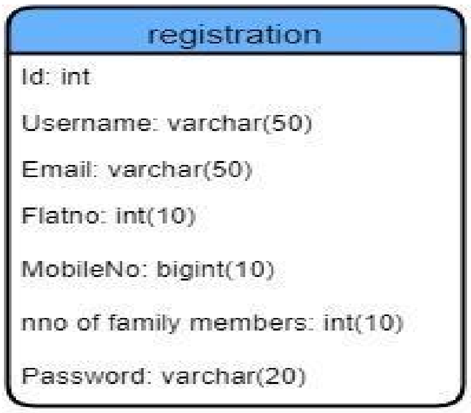


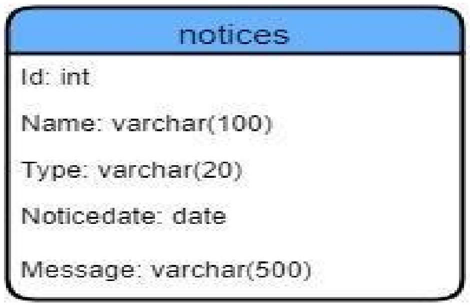
Figure 4.9: Deployment Diagram

## 4.9 Database Design:



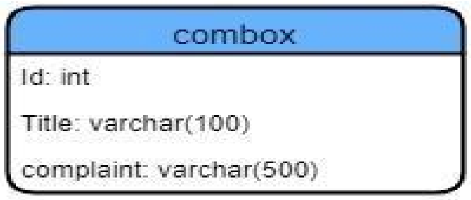
| registration |
|---|
| Id: int |
| Username: varchar(50) |
| Email: varchar(50) |
| Flatno: int(10) |
| MobileNo: bigint(10) |
| nno of family members: int(10) |
| Password: varchar(20) |

**4.10.1 Members Account Table**



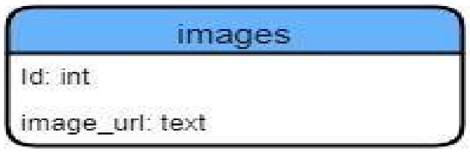
| notices |
|---|
| Id: int |
| Name: varchar(100) |
| Type: varchar(20) |
| Noticedate: date |
| Message: varchar(500) |

**4.10.2 Notices Table**



| combox |
|---|
| Id: int |
| Title: varchar(100) |
| complaint: varchar(500) |

**4.10.3 Complaints Table**



| images |
|---|
| Id: int |
| image_url: text |

**4.10.4 ImagesTable**

Figure 4.10: Database Design

# System Implementation.

## FRONT-END CODE :

### ADMIN FORM (.CSS)

```
/* Container styling */
.abc{
   background-image: url("../images/download.jpg")
}

div {
   display: flex;
   flex-direction: column;
   align-items: center;
   margin-top: 50px;

  }

  /* Form title styling */
  h2 {
   font-size: 24px;
   font-weight: bold;
   margin-bottom: 20px;
   color: #333;
  }

  /* Form styling */
  form {
   display: flex;
   flex-direction: column;
   width: 300px;
   background-color: #f9f9f9;
   padding: 20px;
   border-radius: 8px;
   box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  }

  /* Input field styling */
  input {
   margin-bottom: 15px;
   padding: 10px;
```

```
    font-size: 16px;
    border: 1px solid #ccc;
    border-radius: 4px;
    background-color: #fff;
    transition: border-color 0.3s;
  }

  input:focus {
    border-color: #007bff;
    outline: none;
  }

  /* Submit button styling */
  button {
    padding: 10px;
    font-size: 16px;
    font-weight: bold;
    color: #fff;
    background-color: #007bff;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    transition: background-color 0.3s;
  }

  button:hover {
    background-color: #0056b3;
  }
```

--------------------------------------------------------------------------------------------------------

**ADMIN FORM (.JS)**

```
import React, { useState } from "react";
import axios from "axios";
import "./AdminForm.css";

const AdminForm = ({ fetchAdmins }) => {
  const [admin, setAdmin] = useState({
    id: "",
    email: "",
    password: "",
    username: "",
    mobileNo: "",
    noOfFamilyMembers: 0,
  });

  const [responseString, setResponseString] = useState("");
const handleChange = (event) => {
  setAdmin({ ...admin, [event.target.name]: event.target.value });
};

const handleSubmit = async (event) => {
  event.preventDefault();
  try {
    const res = await axios.post("http://localhost:9999/api/registrations", admin);
    setResponseString(res.data);
    localStorage.setItem("token", JSON.stringify(res.data));
    if (res.data === "") {
      window.alert("Unsuccessful");
    } else if (typeof res.data === "object") {
      alert("Successful");
    }
  } catch (error) {
    console.error("Error creating admin", error);
  }
};

return (
  <div className="abc">
    <h2>Create User</h2>
    <form onSubmit={handleSubmit}>
      <input
        type="text"
        name="id"
        placeholder="ID"
        value={admin.id}
        onChange={handleChange}
      />
      <input
        type="text"
        name="username"
        placeholder="Username"
        value={admin.username}
        onChange={handleChange}
      />
      <input
        type="email"
        name="email"
        placeholder="Email"
        value={admin.email}
        onChange={handleChange}
      />
      <input
        type="password"
```

```
         name="password"
         placeholder="Password"
         value={admin.password}
         onChange={handleChange}
        />
        <input
         type="text"
         name="mobileNo"
         placeholder="Mobile No"
         value={admin.mobileNo}
         onChange={handleChange}
        />
        <input
         type="number"
         name="noOfFamilyMembers"
         placeholder="Number of Family Members"
         value={admin.noOfFamilyMembers}
         onChange={handleChange}
        />
        <button type="submit">Create User</button>
      </form>
    </div>
  );
};

export default AdminForm;
```

-----------------------------------------------------------------------------------------------------------

**ADMIN LIST (.JS)**

```
import React, { useEffect, useState } from "react";
import axios from "axios";

const AdminList = () => {
  const [admins, setAdmins] = useState([]);

  useEffect(() => {
    fetchAdmins();
  }, []);

  const fetchAdmins = async () => {
    try {
      const response = await axios.get("http://localhost:9999/api/registrations");
      setAdmins(response.data);
    } catch (error) {
      console.error("Error fetching admins", error);
    }
  };

  return (
    <div>
      <h2>User List</h2>
      <ul>
        {admins.map((admin) => (
          <li key={admin.id}>
            {admin.username} ({admin.email}) - Family Members: {admin.noOfFamilyMembers}
          </li>
        ))}
      </ul>
    </div>
  );
};

export default AdminList;
```

----------------------------------------------------------------------------------------------------------------

**<u>COMPLAINT FORM (.CSS)</u>**

```
.complaint-form-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    margin-top: 50px;
  }

  h2 {
    font-size: 24px;
    margin-bottom: 20px;
  }

  form {
    display: flex;
    flex-direction: column;
    width: 300px;
    background-color: #f9f9f9;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  }

  input, textarea {
    margin-bottom: 15px;
    padding: 10px;
    font-size: 16px;
    border: 1px solid #ccc;
    border-radius: 4px;
  }

  input:focus, textarea:focus {
    border-color: #007bff;
    outline: none;
  }

  button {
    padding: 10px;
    font-size: 16px;
    color: #fff;
    background-color: #007bff;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    transition: background-color 0.3s;
  }

  button:hover {
    background-color: #0056b3;
  }
```

---------------------------------------------------------------------------------------------------------------

**COMPLAINT FORM (.JS)**

```
import React, { useState } from "react";
import axios from "axios";
import "./ComplaintForm.css";

const ComplaintForm = ({ fetchComplaints }) => {
  const [complaint, setComplaint] = useState({
    id: "",
    title: "",
    complaint: "",
  });

  const [responseString, setResponseString] = useState("");

  const handleChange = (event) => {
    setComplaint({ ...complaint, [event.target.name]: event.target.value });
  };

  const handleSubmit = async (event) => {
    event.preventDefault();
    try {
     const res = await axios.post("http://localhost:9999/api/combox", complaint);
      setResponseString(res.data);
      localStorage.setItem("token", JSON.stringify(res.data));
      if (res.data === "") {
        window.alert("Unsuccessful");
      } else if (typeof res.data === "object") {
        alert("Successful");
      }
    } catch (error) {
      console.error("Error creating complaint", error);
      alert("Failed to create complaint");
    }
  };

  return (
    <div className="complaint-form-container">
      <h2>Create Complaint</h2>
      <form onSubmit={handleSubmit}>
        <input
          type="text"
          name="id"
          placeholder="ID"
          value={complaint.id}
          onChange={handleChange}
        />
        <input
          type="text"
          name="title"
          placeholder="Title"
          value={complaint.title}
          onChange={handleChange}
        />
        <textarea
          name="complaint"
          placeholder="Your Complaint"
          value={complaint.complaint}
          onChange={handleChange}
        ></textarea>
        <button type="submit">Submit Complaint</button>
      </form>
    </div>
  );
};

export default ComplaintForm;
```

---

**COMPLAINT LIST (.CSS)**

```
.complaint-list-container {
   display: flex;
   flex-direction: column;
   align-items: center;
   margin-top: 50px;
 }

 h2 {
   font-size: 24px;
   margin-bottom: 20px;
 }

 ul {
   list-style-type: none;
   padding: 0;
 }

 li {
   background-color: #f9f9f9;
   padding: 15px;
   margin-bottom: 10px;
   border-radius: 8px;
   box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
 }

 h3 {
   font-size: 18px;
   margin-bottom: 10px;
 }

 p {
   font-size: 16px;
 }
```

---

**COMPLAINT LIST (.JS)**

```
import React, { useEffect, useState } from "react";
import axios from "axios";
import "./ComplaintList.css";

const ComplaintList = () => {
  const [complaints, setComplaints] = useState([]);

  useEffect(() => {
    fetchComplaints();
  }, []);

  const fetchComplaints = async () => {
    try {
      const res = await axios.get("http://localhost:9999/api/combox");
      setComplaints(res.data);
    } catch (error) {
      console.error("Error fetching complaints", error);
    }
  };

  return (
    <div className="complaint-list-container">
      <h2>Complaint List</h2>
      <ul>
        {complaints.map((complaint) => (
          <li key={complaint.id}>
            <h3>{complaint.title}</h3>
            <p>{complaint.complaint}</p>
          </li>
        ))}
      </ul>
    </div>
  );
};

export default ComplaintList;
```

---------------------------------------------------------------------------------------------------------------

**IMAGE FORM (.CSS)**

```
.image-form-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    margin-top: 50px;
  }

  h2 {
    font-size: 24px;
    margin-bottom: 20px;
```

```
 }

 form {
   display: flex;
   flex-direction: column;
   width: 300px;
   background-color: #f9f9f9;
   padding: 20px;
   border-radius: 8px;
   box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
 }

 input {
   margin-bottom: 15px;
   padding: 10px;
   font-size: 16px;
   border: 1px solid #ccc;
   border-radius: 4px;
 }

 input:focus {
   border-color: #007bff;
   outline: none;
 }

 button {
   padding: 10px;
   font-size: 16px;
   color: #fff;
   background-color: #007bff;
   border: none;
   border-radius: 4px;
   cursor: pointer;
   transition: background-color 0.3s;
 }

 button:hover {
   background-color: #0056b3;
 }
```

---------------------------------------------------------------------------------------------------------

**IMAGE FORM (.JS)**

```
import React, { useState } from "react";
import axios from "axios";
import "./ImageForm.css";
```

```
const ImageForm = ({ fetchImages }) => {
  const [image, setImage] = useState({
    id: "",
    imageUrl: "",
  });

  const [responseString, setResponseString] = useState("");
  const handleChange = (event) => {
    setImage({ ...image, [event.target.name]: event.target.value });
  };

  const handleSubmit = async (event) => {
    event.preventDefault();
    try {
      const res = await axios.post("http://localhost:9999/api/images", image);
      setResponseString(res.data);
      localStorage.setItem("token", JSON.stringify(res.data));
      if (res.data === "") {
        window.alert("Unsuccessful");
      } else if (typeof res.data === "object") {
        alert("Successful");
      }
    } catch (error) {
      console.error("Error creating image", error);
      alert("Failed to create image");
    }
  };

  return (
    <div className="image-form-container">
      <h2>Create Image</h2>
      <form onSubmit={handleSubmit}>
        <input
          type="text"
          name="id"
          placeholder="ID"
          value={image.id}
          onChange={handleChange}
        />
        <input
          type="text"
          name="imageUrl"
          placeholder="Image URL"
          value={image.imageUrl}
          onChange={handleChange}
        />
        <button type="submit">Create Image</button>
      </form>
    </div>
  );
};

export default ImageForm;
```

----------------------------------------------------------------------------------------------------------

**IMAGE LIST (.CSS)**

```
.image-list-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    margin-top: 50px;
  }

  h2 {
    font-size: 24px;
    margin-bottom: 20px;
  }

  ul {
    list-style-type: none;
    padding: 0;
  }

  li {
    background-color: #f9f9f9;
    padding: 15px;
    margin-bottom: 10px;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  }

  img {
    max-width: 100%;
    border-radius: 4px;
  }
```

----------------------------------------------------------------------------------------------------------

**IMAGE LIST (.JS)**

```
import React, { useEffect, useState } from "react";
import axios from "axios";
import "./ImageList.css";

const ImageList = () => {
  const [images, setImages] = useState([]);

  useEffect(() => {
    fetchImages();
  }, []);

  const fetchImages = async () => {
    try {
      const res = await axios.get("http://localhost:9999/api/images");
      setImages(res.data);
    } catch (error) {
      console.error("Error fetching images", error);
    }
  };

  return (
    <div className="image-list-container">
      <h2>Image List</h2>
      <ul>
        {images.map((image) => (
          <li key={image.id}>
            <img src={image.imageUrl} alt={`Image ${image.id}`} />
          </li>
        ))}
      </ul>
    </div>
  );
};

export default ImageList;
```

------------------------------------------------------------------------------------------------------------

**NOTICE FROM (.CSS)**

```
.notice-form-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    margin-top: 50px;
  }

  h2 {
```

```
  font-size: 24px;
  margin-bottom: 20px;
}

form {
  display: flex;
  flex-direction: column;
  width: 300px;
  background-color: #f9f9f9;
  padding: 20px;
  border-radius: 8px;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

input,
textarea {
  margin-bottom: 15px;
  padding: 10px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

textarea {
  resize: none;
  height: 100px;
}

input:focus,
textarea:focus {
  border-color: #007bff;
  outline: none;
}

button {
  padding: 10px;
  font-size: 16px;
  color: #fff;
  background-color: #007bff;
  border: none;
  border-radius: 4px;
  cursor: pointer;
  transition: background-color 0.3s;
}

button:hover {
  background-color: #0056b3;
}
```

----------------------------------------------------------------------------------------------------------------

**NOTICE FORM (.JS)**

```
import React, { useState } from "react";
import axios from "axios";
import "./NoticeForm.css";

const NoticeForm = ({ fetchNotices }) => {
  const [notice, setNotice] = useState({
    id: "",
    name: "",
    type: "",
    noticedate: "",
    message: "",
  });

  const [responseString, setResponseString] = useState("");

  const handleChange = (event) => {
    setNotice({ ...notice, [event.target.name]: event.target.value });
  };

  const handleSubmit = async (event) => {
    event.preventDefault();
    try {
      const res = await axios.post("http://localhost:9999/api/notices", notice);
      setResponseString(res.data);
      localStorage.setItem("token", JSON.stringify(res.data));
      if (res.data === "") {
        window.alert("Unsuccessful");
      } else if (typeof res.data === "object") {
        alert("Successful");
      }
    } catch (error) {
      console.error("Error creating notice", error);
      alert("Failed to create notice");
    }
  };

  return (
    <div className="notice-form-container">
      <h2>Create Notice</h2>
      <form onSubmit={handleSubmit}>
        <input
```

```
          type="text"
          name="id"
          placeholder="ID"
          value={notice.id}
          onChange={handleChange}
        />
        <input
          type="text"
          name="name"
          placeholder="Name"
          value={notice.name}
          onChange={handleChange}
        />
        <input
          type="text"
          name="type"
          placeholder="Type"
          value={notice.type}
          onChange={handleChange}
        />
        <input
          type="date"
          name="noticedate"
          value={notice.noticedate}
          onChange={handleChange}
        />
        <textarea
          name="message"
          placeholder="Message"
          value={notice.message}
          onChange={handleChange}
        />
        <button type="submit">Create Notice</button>
      </form>
    </div>
  );
};

export default NoticeForm;
```

------------------------------------------------------------------------------------------------------------

**NOTICE LIST (.CSS)**

```
.notice-list-container {
    display: flex;
    flex-direction: column;
    align-items: center;
    margin-top: 50px;
  }

  h2 {
    font-size: 24px;
    margin-bottom: 20px;
  }

  ul {
    list-style-type: none;
    padding: 0;
  }

  li {
    background-color: #f9f9f9;
    padding: 15px;
    margin-bottom: 10px;
    width: 300px;
    border-radius: 8px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
  }

  h3 {
    margin: 0;
    font-size: 18px;
  }

  p {
    margin: 5px 0;
  }
```

------------------------------------------------------------------------------------------------------------

**NOTICE LIST (.JS)**

```
import React, { useEffect, useState } from "react";
import axios from "axios";
import "./NoticeList.css";

const NoticeList = () => {
  const [notices, setNotices] = useState([]);

  useEffect(() => {
    fetchNotices();
  }, []);
  const fetchNotices = async () => {
    try {
      const res = await axios.get("http://localhost:9999/api/notices");
      setNotices(res.data);
    } catch (error) {
      console.error("Error fetching notices", error);
    }
  };

  return (
    <div className="notice-list-container">
      <h2>Notice List</h2>
      <ul>
        {notices.map((notice) => (
          <li key={notice.id}>
            <h3>{notice.name}</h3>
            <p>{notice.type}</p>
            <p>{new Date(notice.noticedate).toLocaleDateString()}</p>
            <p>{notice.message}</p>
          </li>
        ))}
      </ul>
    </div>
  );
};

export default NoticeList;
```

-----------------------------------------------------------------------------------------------------------

**APP (.CSS)**

```
.App {
  text-align: center;
}

.App-logo {
  height: 40vmin;
  pointer-events: none;
}

@media (prefers-reduced-motion: no-preference) {
  .App-logo {
    animation: App-logo-spin infinite 20s linear;
  }
}

.App-header {
  background-color: #282c34;
  min-height: 100vh;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  font-size: calc(10px + 2vmin);
  color: white;
}

.App-link {
  color: #61dafb;
}

@keyframes App-logo-spin {
  from {
    transform: rotate(0deg);
  }
  to {
    transform: rotate(360deg);
  }
}
```

------------------------------------------------------------------------------------------------------------

**APP (.JS)**

```
import React from "react";
import { BrowserRouter as Router, Route, Routes, Link } from "react-router-dom";
import AdminForm from "./component/AdminForm";
import AdminList from "./component/AdminList";
import NoticeForm from "./component/NoticeForm";
import NoticeList from "./component/NoticeList";
import ImageForm from "./component/ImageForm";
import ImageList from "./component/ImageList";
import ComplaintForm from "./component/ComplaintForm";
import ComplaintList from "./component/ComplaintList";
function App() {
  return (
    <Router>
      <nav>
        <ul>
          <li>
          <Link to="/">Admin Dashboard</Link>
        </li>
        <li>
          <Link to="/create-admin">Create User</Link>
        </li>
        <li>
          <Link to="/admin-list">User List</Link>
        </li>
        <li>
          <Link to="/create-notice">Create Notice</Link>
        </li>
        <li>
          <Link to="/notices">Notice List</Link>
        </li>
        <li>
          <Link to="/create-image">Create Image</Link>
        </li>
        <li>
          <Link to="/images">Image List</Link>
        </li>
        <li>
          <Link to="/create-complaint">Create Complaint</Link>
        </li>
        <li>
          <Link to="/complaints">Complaint list</Link>
        </li>
      </ul>
    </nav>

    <Routes>
      <Route path="/" element={<AdminDashboard />} />
      <Route path="/create-admin" element={<AdminForm />} />
      <Route path="/admin-list" element={<AdminList />} />
      <Route path="/create-notice" element={<NoticeForm />} />
      <Route path="/notices" element={<NoticeList />} />
      <Route path="/create-image" element={<ImageForm />} />
      <Route path="/images" element={<ImageList />} />
      <Route path="/create-complaint" element={<ComplaintForm />} />
      <Route path="/complaints" element={<ComplaintList />} />
    </Routes>
  </Router>
 );
}

const AdminDashboard = () => {
 return (
  <div>
    <h2>Admin Dashboard</h2>
```

```
      <p>Welcome to the Admin Dashboard!</p>
    </div>
  );
};

export default App;
```

------------------------------------------------------------------------------------------------------------------------

## BACK-END CODE :

### package in.sp.main :

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class RestSpringBootCrudOpApplication {

	public static void main(String[] args) {

		SpringApplication.run(RestSpringBootCrudOpApplication.class, args);
	}

}
```

## //CONTROLLER//

### 1. COMPLAINT

```
package in.sp.main.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import in.sp.main.entity.Complaintbox;
import in.sp.main.repository.ComplaintRepository;

import java.util.List;
import java.util.Optional;

@Service
public class ComplaintController {

   @Autowired
   private ComplaintRepository comboxRepository;

   public List<Complaintbox> getAllComboxes() {
      return comboxRepository.findAll();
   }

   public Optional<Complaintbox> getComboxById(Integer id) {
      return comboxRepository.findById(id);
   }
```

```
    public Complaintbox saveCombox(Complaintbox combox) {
        return comboxRepository.save(combox);
    }

    public void deleteCombox(Integer id) {
        comboxRepository.deleteById(id);
    }
}
```

-------------------------------------------------------------------------------------------------------------------

**2. IMAGES**

```
package in.sp.main.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import in.sp.main.entity.Images;
import in.sp.main.service.ImagesService;
import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/api/images")
public class ImagesController {

    @Autowired
    private ImagesService imagesService;

    @GetMapping
    public List<Images> getAllImages() {
        return imagesService.getAllImages();
    }

    @GetMapping("/{id}")
    public Optional<Images> getImageById(@PathVariable Integer id) {
        return imagesService.getImageById(id);
    }

    @PostMapping
    public Images createImage(@RequestBody Images image) {
        return imagesService.saveImage(image);
    }

    @PutMapping("/{id}")
    public Images updateImage(@PathVariable Integer id, @RequestBody Images image) {
        image.setId(id);
        return imagesService.saveImage(image);
}}
```

**3. NOTICES**

```
package in.sp.main.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import in.sp.main.entity.Notice;
import in.sp.main.service.NoticesService;
import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/api/notices")
public class NoticesController {

@Autowired
    private NoticesService noticesService;

    @GetMapping
    public List<Notice> getAllNotices() {
        return noticesService.getAllNotices();
    }

    @GetMapping("/{id}")
    public Optional<Notice> getNoticeById(@PathVariable Integer id) {
        return noticesService.getNoticeById(id);
    }

    @PostMapping
    public Notice createNotice(@RequestBody Notice notice) {
        return noticesService.saveNotice(notice);
    }

    @PutMapping("/{id}")
    public Notice updateNotice(@PathVariable Integer id, @RequestBody Notice notice) {
        notice.setId(id);
        return noticesService.saveNotice(notice);
    }

    @DeleteMapping("/{id}")
    public void deleteNotice(@PathVariable Integer id) {
        noticesService.deleteNotice(id);
    }
}
```

## 4. REGISTRATIONS

```
package in.sp.main.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import in.sp.main.entity.Registration;
import in.sp.main.service.RegistrationService;
import java.util.List;
import java.util.Optional;

@RestController
@RequestMapping("/api/registrations")
public class RegistrationController {

    @Autowired
    private RegistrationService registrationService;

   @GetMapping
    public List<Registration> getAllRegistrations() {
        return registrationService.getAllRegistrations();
    }

    @GetMapping("/{id}")
    public Optional<Registration> getRegistrationById(@PathVariable Integer id) {
        return registrationService.getRegistrationById(id);
    }

    @PostMapping
    public Registration createRegistration(@RequestBody Registration registration) {
        return registrationService.saveRegistration(registration);
    }

    @PutMapping("/{id}")
    public Registration updateRegistration(@PathVariable Integer id, @RequestBody Registration
registration) {
        registration.setId(id);
        return registrationService.saveRegistration(registration);
    }

    @DeleteMapping("/{id}")
    public void deleteRegistration(@PathVariable Integer id) {
        registrationService.deleteRegistration(id);
    }
}
```

## Results

### 6.1 Validation and Naming Conventions:

| Sr.No | Control ID | Validation Used | Reason |
|---|---|---|---|
| 1. | Username | RequiredFieldValidator | Username Cannot be null |
| 2. | Email | RequiredFieldValidator | Email Cannot be Null |
| 3. | Flat no. | RequiredFieldValidator | Flat no. Cannot be null |
| 4. | Mobile no. | RequiredFieldValidator | Mobile no. Cannot be null |
| 5. | No. of family members | RequiredFieldValidator | No.of family Members cannot be null |
| 6. | Password | RequiredFieldValidator | Password cannot be null |
| 7. | Title | RequiredFieldValidator | Title cannot be null |
| 8. | Subject | RequiredFieldValidator | Subject cannot be null |
| 9. | Notice type | RequiredFieldValidator | Notice Type cannot be null |

## 6.1 Reports:

### ❖ Registration Table

- Registration table is created where  all the registered members details has been stored.
- This table contains Id, Username, Email, Flat no, Mobile No, no of family members, Password.
- This data is retrieved from database through PHP and MySQL.
- The retrieved data from this table is displayed on the website in manage member page.

### ❖ Notice Table.

- Notice table is created where all the notices has been stored.
- This table contains ID, Name, Type, Notice date, Message.
- This data is retrieved from database through PHP and MySQL.
- The retrieved data from this table is displayed on the website in  noticeboard page.

### ❖ Complaint Table

- Complaint table is created where all the complaints by user has been stored.
- This table contains id, title, complaint.
- This data is retrieved from database through Springboot and mysql.
- The retrieved data from this table is displayed on the website in  view complaint page.

## Future Enhancement

- This project can be enhanced further by adding online maintenance payment and receipt for that should be generated for every member to reduce the extra work of admin.
- Message and email alerts for various happenings in the society can be added to the system so that users do not miss the updates and happenings of the society.
- Parking management should be added to reduce parking issues.
- Guest entry for every flat should be registered so that there will be no security issues.
- Attending meeting online should be added to avoid get-together at meeting.

## Conclusion

- This project reduces the manual work for managing society members records, passing notices to each member and also registering complaints.

## References

- https://www.w3schools.com/php/
- https://www.youtube.com/
- https://www.tutorialspoint.com/
- https://www.javatpoint.com/php-tutorial
- https://www.w3schools.com/php/

## Annexure

### 10.1 Figure List:

### 10.2 Table List: