SPL-2 Project Report

Intern & Placement Office Management(IPOM)

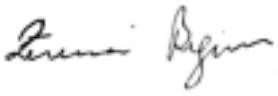
Submitted by

Mahir Faisal

(BSSE Roll No: 1316, BSSE Session: 2020-21)

Mostafizur Rahaman

(BSSE Roll No: 1320, BSSE Session: 2020-21)



Supervised by

Dr. Zerina Begum

Professor

Institute of Information Technology

Institute of Information Technology University of Dhaka

[13-06-2024]

Table of Contents

List Of Figure

1. Introduction

The Intern and Placement Office Management (IPOM) system at IIT is designed to facilitate the placement of 4th-year BSSE students into internships at reputed software companies. The internship semester spans from January 1st to June 30th each year. The IPOM involves the collaborative efforts of an IPO Committee (IPOC) led by the IPO head, who has the authority to enlist new companies. Companies interested in taking interns contact the IPO head, after which IPOC members enter relevant details about the companies and available intern positions into the system. Students upload their CVs and provide their skillsets, CGPA, and company preferences. Based on this information, IPOC members suggest suitable candidates for the IPO head, who then allocate students to companies for interviews. Post-interview, the system updates the status of the students and informs them via email. Selected students receive formal letters, and companies issue offer letters to interns, which are also recorded by the IPO head.

2. Background of the Project

For implementing this project, prior studies were necessary. These are given below:

2.1 Necessity of Prior Studies:

Before implementing the project, it was crucial to conduct thorough studies to understand the existing internship placement process and identify areas for improvement.

2.2 Key Studies Conducted:

- **Internship Placement Process Analysis:** An analysis of the current internship placement process at IIT was conducted to identify pain points, inefficiencies, and areas for enhancement.
- **Stakeholder Needs Assessment:** Interviews and surveys were conducted with stakeholders including students, IPOC members, and

company representatives to gather feedback on their needs, challenges, and expectations from the internship placement system.

- **Technology Assessment:** Research was conducted on existing technologies and systems used in internship placement management to identify best practices and potential technologies for implementation.

2.3 Objective of Prior Studies:

The objective of these prior studies was to gather insights and requirements that would inform the design and development of the Intern and Placement Office Management (IPOM) system. These studies provided a foundation for understanding the context, requirements, and challenges associated with internship placement management at IIT, thereby guiding the development process effectively.

2.4 Web Application Development

- **React**: A front-end JavaScript library for building user interfaces based on components. It enables efficient rendering and updating of the UI.
- **Node.js**: A JavaScript runtime built on Chrome's V8 engine, suitable for building scalable and efficient server-side applications.
- **MongoDB**: A source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas, making it flexible and scalable.
- **Express.js:** is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the development of RESTful APIs and handles routing, middleware, and HTTP requests efficiently.

3. Description of the Project

There is an Intern and Placement Office (IPO) at IIT, designated to send BSSE 4th Year students to reputed software companies as interns. The internship semester starts on 1st January and ends on 30th June of every year. The IPO Committee (IPOC) consists of three members, presided over by the head of IPO. The IPO head has the authority to enlist new companies.

3.1 Company Registration:

Intern-taking companies express their interest in taking interns by contacting the IPO head. Based on this interest, an IPOC member enters the intern-taking company names, HR's email address & password into the system. This information is stored for further processing.

3.2 Student Application Process

Students upload their CVs and, in a separate form, mention their skillsets and CGPA. They can also specify their company preferences(only one not multiple). This information is critical for matching students with appropriate internship opportunities.

3.3 Student Allocation

Based on a student's CGPA and skillset, the IPO head then allocates students to companies for interviews based on these suggestions. The IPO head retains the authority to place any student in any company.

3.4 Interview Process and Selection

Post-interview, companies update the system with the selection results, marking Add as Interns. Both students and the IPOC can view the status weather they are selected or not.

4. Implementation

The implementation of the IPOM system will involve the integration of various functionalities such as user management, company management, student management, allocation and interview management, status tracking, notifications, and document generation. Each of these components will be developed using secure, efficient, and user-friendly technologies to ensure a smooth and effective internship placement process.

4.1 Frontend

In the front-end development of the IPOM system, React is used as the primary library for building user interfaces. React offers several advantages for developing dynamic and interactive web applications, making it a suitable choice for the front end of the IPOM system.

4.1.1 Styling

Styling involves creating a visually appealing and user-friendly interface for the IPOM system. This includes designing forms, dashboards, and notification panels.

Figure 1:Tailwind styling for consistency

```json
{
  "name": "spl2",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  ▷ Debug
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "lint": "eslint . --ext js,jsx --report-unused-disable-directives --max-warnings 0",
    "preview": "vite preview"
  },
  "dependencies": {
    "@ant-design/icons": "^5.3.4",
    "antd": "^5.15.3",
    "axios": "^1.6.7",
    "bootstrap": "^5.3.3",
    "cors": "^2.8.5",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-router-dom": "^6.22.2"
  },
  "devDependencies": {
    "@types/react": "^18.2.56",
    "@types/react-dom": "^18.2.19",
    "@vitejs/plugin-react": "^4.2.1",
    "eslint": "^8.56.0",
    "eslint-plugin-react": "^7.33.2",
    "eslint-plugin-react-hooks": "^4.6.0",
    "eslint-plugin-react-refresh": "^0.4.5",
    "tailwindcss": "^3.4.3",
    "vite": "^5.2.10"
  }
}
```

Figure 2: Dependencies from package.json

4.1.2 Components Structure

The front end is built using a component-based architecture, enabling reusable and maintainable code. Components include login forms, dashboards, CV upload forms, and notification systems.

4.1.3 Interaction and Testing

The front end includes interactive elements for users to upload CVs, view statuses, and receive notifications. Comprehensive testing ensures that all components function correctly and provide a seamless user experience.

4.1.4 Code

Home

```
return (
  <Layout style={layoutStyle}>
    <Sider trigger={null} collapsible collapsed={collapsed} style={{ backgroundColor: '#1c2833' }}>
      <div className="demo-logo-vertical" />
      <Menu theme="dark" mode="inline" defaultSelectedKeys={['1']} style={{ backgroundColor: '#1c2833' }}>
        <Menu.Item key="1" icon={<UserOutlined />}>
          <Link to="/registration-type" className="btn btn-primary">Signup</Link>
        </Menu.Item>
        <Menu.Item key="2" icon={<UserOutlined />}>
          <Link to="/login" className="btn btn-primary">Login</Link>
        </Menu.Item>
        <Menu.Item key="3" icon={<MenuFoldOutlined />}>
          <Link to="/about" className="btn btn-primary">About</Link>
        </Menu.Item>
      </Menu>
    </Sider>
    <Layout>
      <Header style={headerStyle}>
        <Button
          type="text"
          icon={collapsed ? <MenuUnfoldOutlined /> : <MenuFoldOutlined />}
          onClick={() => setCollapsed(!collapsed)}
          style={buttonStyle}
        />
      </Header>
      <Content style={contentStyle}>
        <div style={boxStyle}>
        <Title level={2} style={textStyle}>Welcome to</Title>
        <h1 style={textStyle}>Intern & Placement Office Management</h1>
        </div>
      </Content>
    </Layout>
  </Layout>
);
```

Figure 3: Home page code

Authentication

```
function checkAuthentication(req, res, next) {
    try {
        const bearerToken = req.headers.authorization;
        const authToken = bearerToken ? bearerToken.split(" ")[1] : null;
        const cookie = req.cookies["access-token"];

        const token = cookie || authToken;

        if (!token) return res.status(401).json({ message: "Authentication failed" });

        const decoded = verifyJwtToken(token);

        if (!decoded) {
            return res.status(401).json({ message: "Authentication failed" });
        }

        console.log(decoded);

        req.user = decoded;

        next();
    } catch (err) {
        console.log(err);
        res.status(500).json({ message: "internal server error" });
    }
}

function isAdmin(req, res, next) {
    const user = req.user;
    if (user.userType === "admin") {
        next();
    } else {
        return res.status(403).json({ message: "you are not allowed" });
    }
}

function isTeacher(req, res, next) {
    const user = req.user;
    if (user.userType == "ipoc_head" || user.userType == "ipoc_member") {
```

Figure 4: Code for authentication

Login

```
useEffect(() => {
  const user = localStorage.getItem('userType');
  if (user === 'admin') {
    navigate('/ipoc-dashboard');
  } else if (user === 'student') {
    navigate('/stu_dashboard');
  } else if (user === 'company_manager') {
    navigate('/company-dashboard');
  } else {
    console.error('Invalid access token:', user);
  }
}, [navigate]);

const handleSubmit = async (e) => {
  e.preventDefault();
  if (!email || !password) {
    console.error('Email and password are required.');
    return;
  }
  try {
    const res = await login({ email, password });
    console.log('Log in ', res);
    localStorage.setItem('accessToken', res);

    // User info
    const info = await getUser();
    console.log(info, "infooooooooooooo");
    localStorage.setItem('userId', info.data._id);
    localStorage.setItem('userType', info.data.userType);

    navigate('/stu_dashboard');
  } catch (err) {
    console.log(err);
  }
};
```

Figure 5: Code for Login

Upload CV

```
const multer = require("multer");
const path = require("path");
const cvStorage = multer.diskStorage({
    destination: "./public/",
    filename: function (req, file, cb) {
        const ext = path.extname(file.originalname);
        if (ext != ".pdf" || ext != ".PDF") return cb(null, req.params.id + ext);
        else return cb(new Error("Only PDF files are allowed"));
    },
});

const cvUploader = multer({ storage: cvStorage });

module.exports = { cvUploader };
```

Figure 6: Code for Upload CV

Add Intern

```
const AddInterviews = () => {
  const [isSubmitting, setIsSubmitting] = useState(false);
  const navigate = useNavigate();
  const { id } = useParams();
  const [companyId, setCompanyId] = useState();

  //force pushh
  const handleSubmit = async (values) => {
    setIsSubmitting(true);
    try {
      const managerId = localStorage.getItem('userId');
      const res = await findCompanyManagerById(managerId); //this is hardcoded company
      setCompanyId(res.company._id);

      const response = await addForInterview(values, companyId);
      console.log('Response:', response);
      navigate('/company-dashboard');
    } catch (error) {
      message.error('Failed to add intern.');
      console.error('Error submitting form:', error);
    } finally {
      setIsSubmitting(false);
    }
  };
```

Figure 7: Code for Add intern student

Suggested Students

```
useEffect(() => {
  const fetchCompanies = async () => {
    try {
      const response = await getAllCompany();
      console.log(response);
      setCompanies(response.data);
    } catch (error) {
      console.error('Failed to fetch companies:', error);
    }
  };

  fetchCompanies();
}, []);

const handleCompanyChange = async (value) => {
  setSelectedCompany(value);
  setLoading(true);
  try {
    const response = await suggestStudent(studentId, value);
    // Process the response as needed
    console.log('Suggested students:', response.data);
    message.success('Students Referred successfully!');
  } catch (error) {
    console.error('Failed to Refer students:', error);
    message.error('Failed to Refer students');
  } finally {
    setLoading(false);
  }
};

const handleOk = () => {
  onClose();
};
```

Figure 8: Code for Suggested company

API

```
const ipocSignUp = async (data) => {
  try {
    const res = await axios.post("/users", data, { withCredentials: true });
    console.log("craeate User", res);
    return res.data;
  } catch (error) {
    console.error("error in creating User", error);
    throw error;
  }
};

const studentSignUp = async (data) => {
  try {
    const res = await axios.post("/students", data, { withCredentials: true });
    console.log("create student ", res);
    return res.data;
  } catch (error) {
    console.error("error in creating student", error);
    throw error;
  }
};
```

Figure 9: API call for signup

Dashboard

```
const IpocDashboard = () => {
  const [collapsed, setCollapsed] = useState(false);
  const navigate = useNavigate();

  useEffect(() => {
    const user = localStorage.getItem('userType');
    if (user === 'student' || user === 'company_manager') {
      navigate('/login');
    }
  }, [navigate]);

  const handleLogout = () => {
    localStorage.clear();

    document.cookie.split(";").forEach((c) => {
      document.cookie = c
        .replace(/^ +/, "")
        .replace(/=.*/, "=;expires=" + new Date().toUTCString() + ";path=/");
    });

    navigate('/login');
  };
```

Figure 10: Code for IpocDashboard

4.2 Backend

The backend manages data storage, business logic, and communication with the frontend. It includes:

- Node.js and Express for server-side logic.
- MongoDB for database management.
- Authentication and authorization mechanisms.
- API endpoints for handling requests.

Authentication checks

```
function checkAuthentication(req, res, next) {
    try {
        const bearerToken = req.headers.authorization;
        const authToken = bearerToken ? bearerToken.split(" ")[1] : null;
        const cookie = req.cookies["access-token"];

        const token = cookie || authToken;

        if (!token) return res.status(401).json({ message: "Authentication failed" });

        const decoded = verifyJwtToken(token);

        if (!decoded) {
            return res.status(401).json({ message: "Authentication failed" });
        }

        console.log(decoded);

        req.user = decoded;

        next();
    } catch (err) {
        console.log(err);
        res.status(500).json({ message: "internal server error" });
    }
}

function isAdmin(req, res, next) {
    const user = req.user;
    if (user.userType === "admin") {
        next();
    } else {
        return res.status(403).json({ message: "you are not allowed" });
    }
}
```

Figure 11: Code for authentication checks

Database Connection:

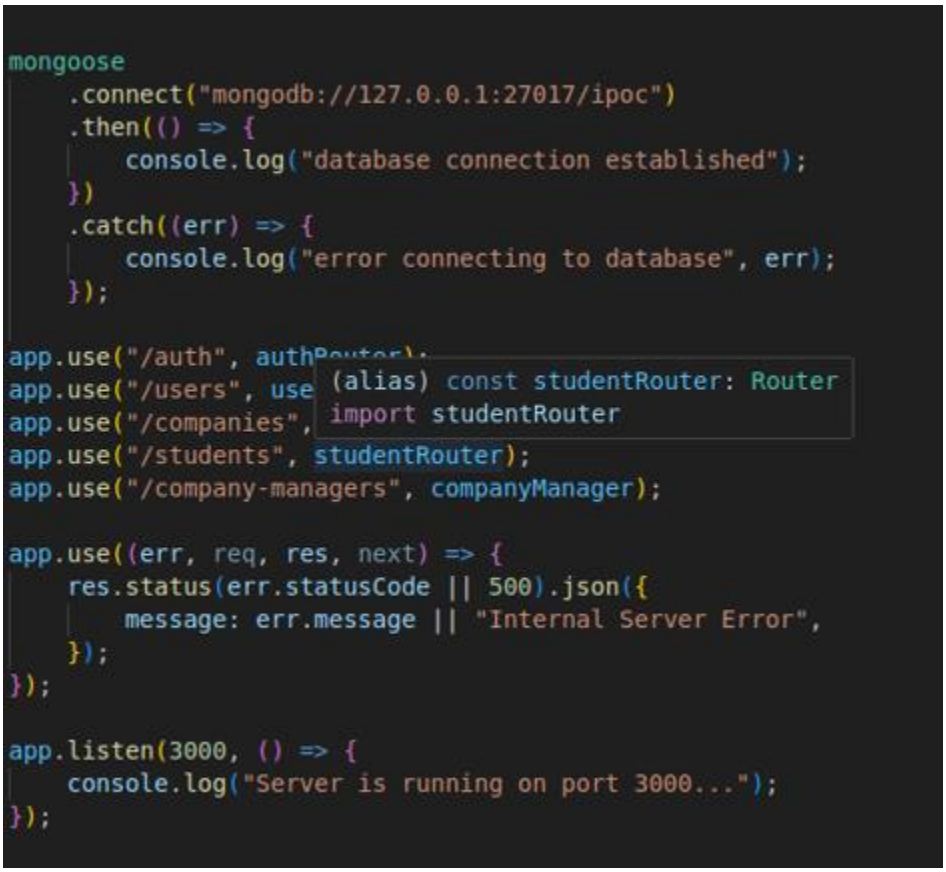


Figure 12: Code for database connection

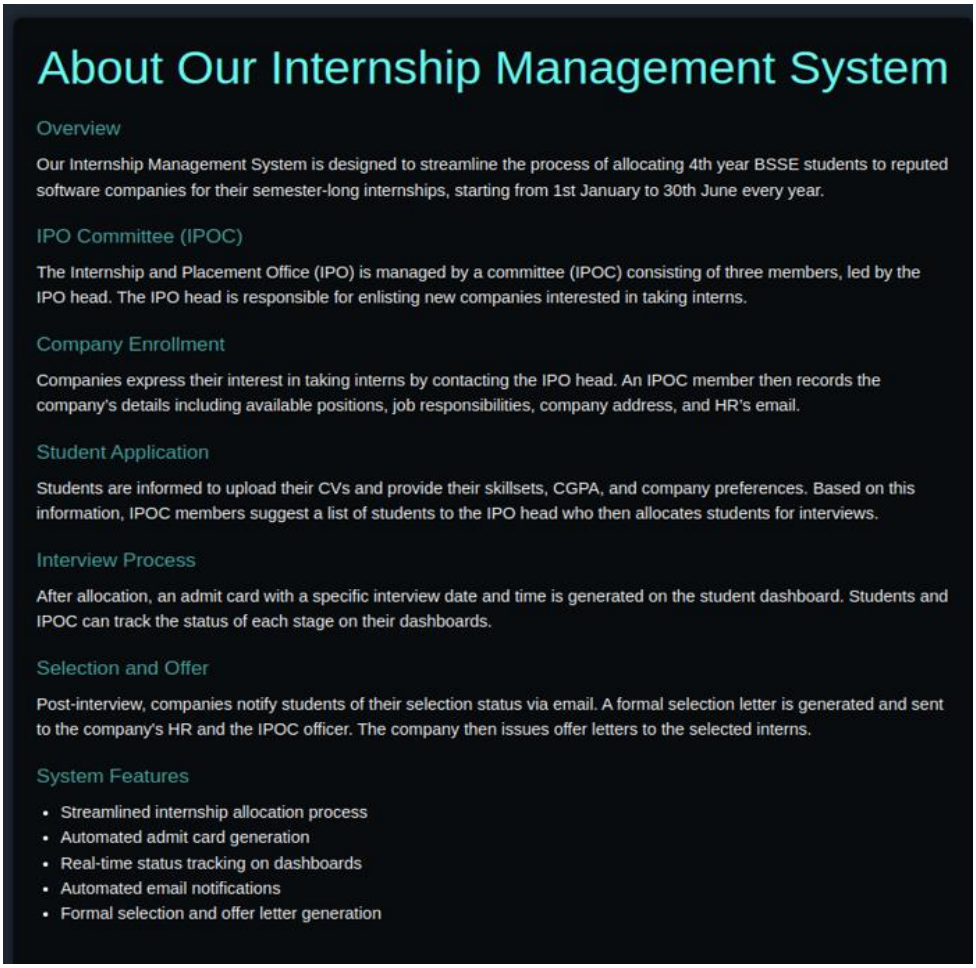
5. User Manual

The user manual provides detailed instructions for all users of the IPOM system, including students, IPOC members, and company HR representatives.

First, clone the code from https://github.com/MdMostafizurRahaman/SPL-II.git. Then open a terminal for frontend and backend then write npm install. Then front end npm run dev. Backend node index.js. It covers the following areas:

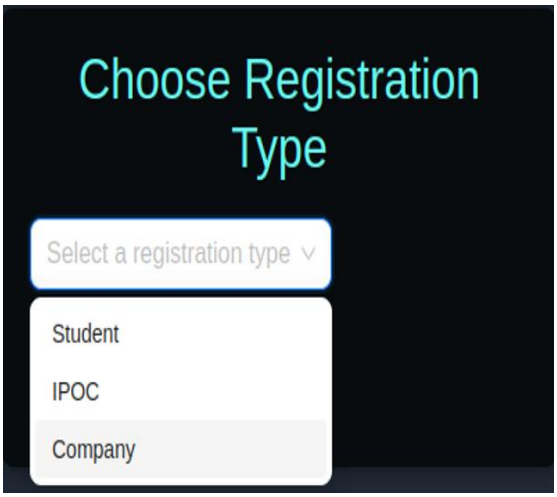
Website Interface

About our website

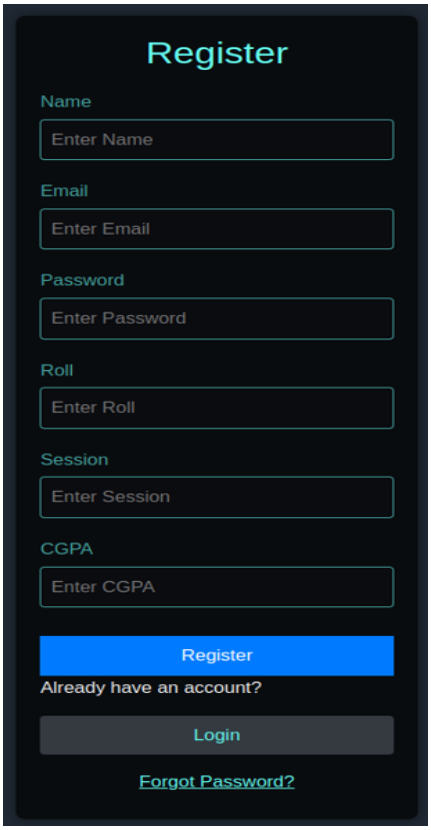


5.1 Student Guide

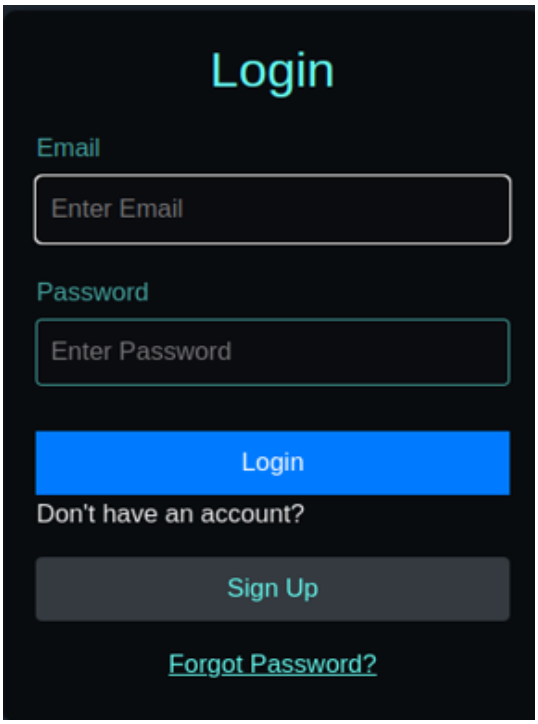
Create an account



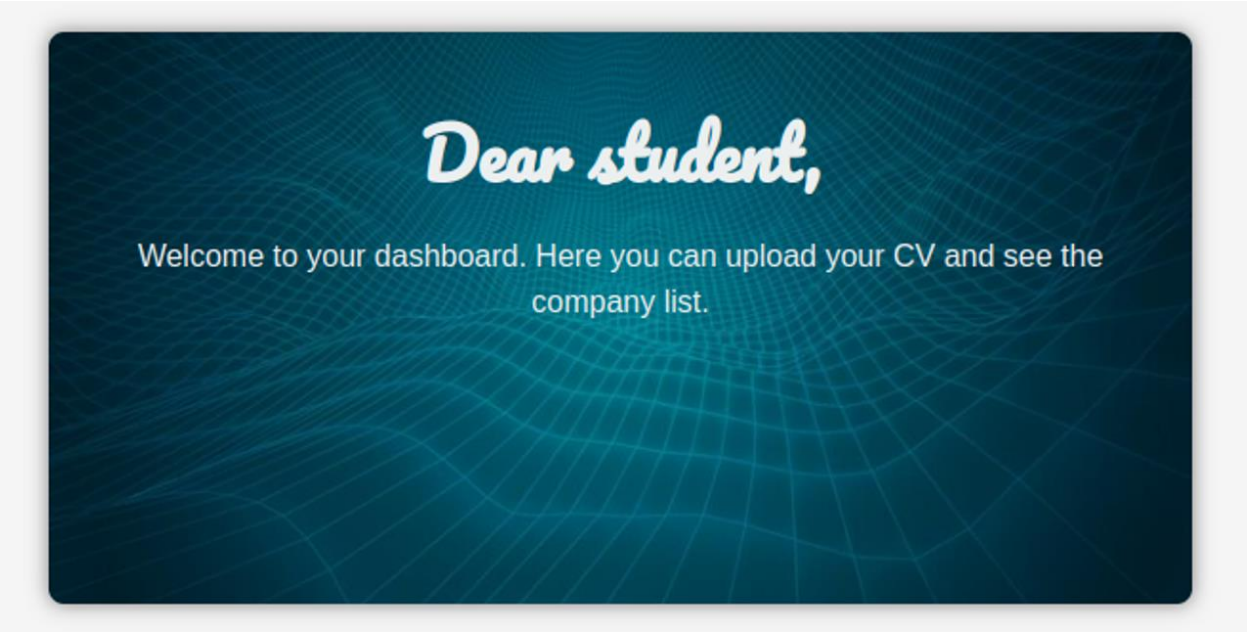
Select Student



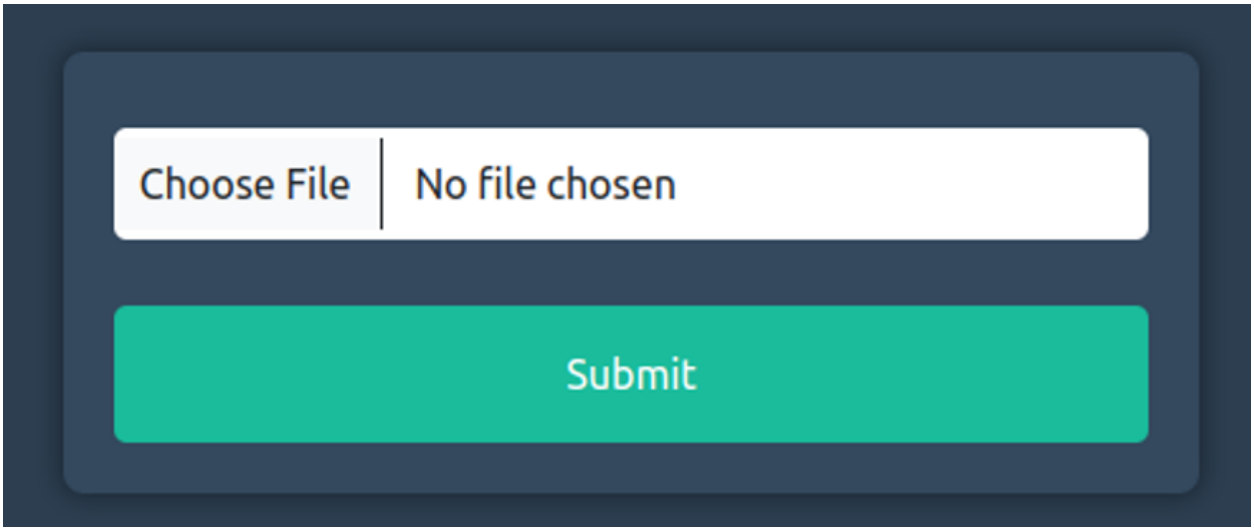
After completing the registration process, Login to the system



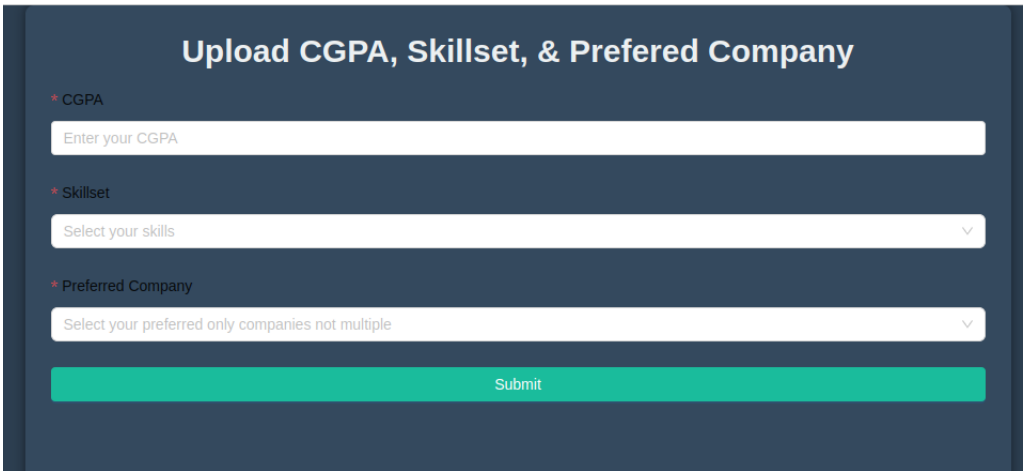
Successful Login goes to Student Dashboard



Upload CV

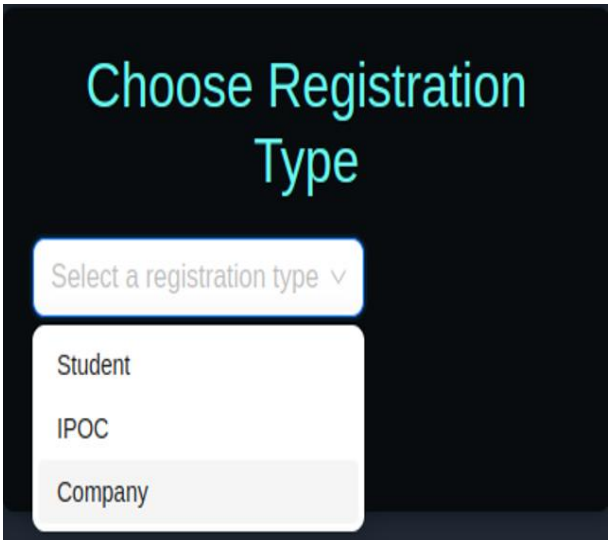


Upload CGPA, Skills & Preferred Company

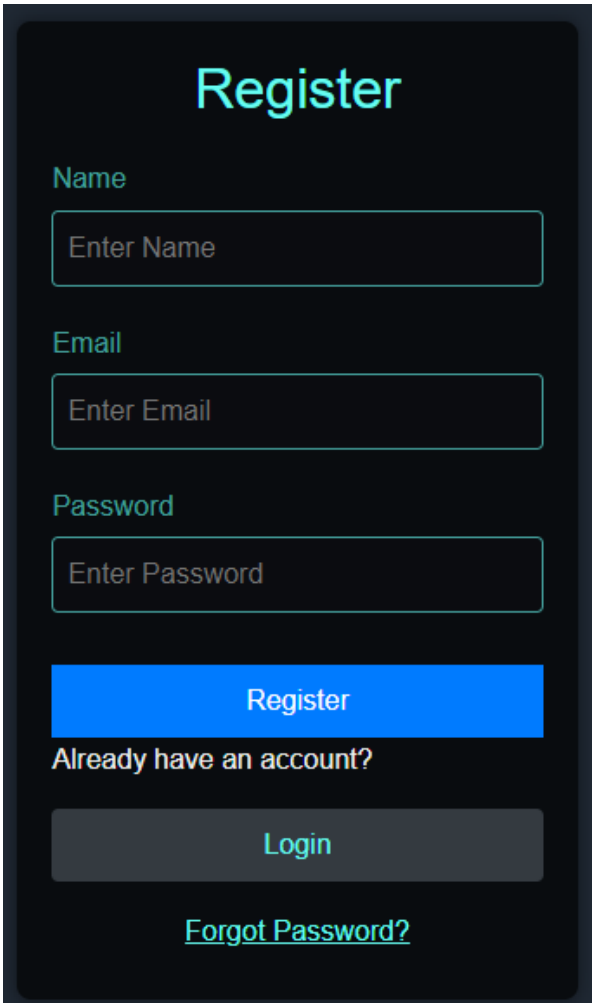


5.2 Admin Guide

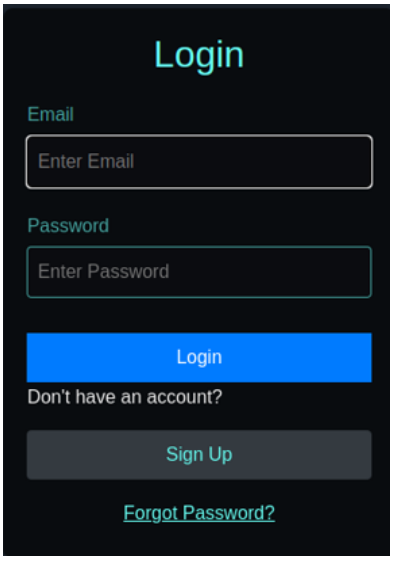
Create an account



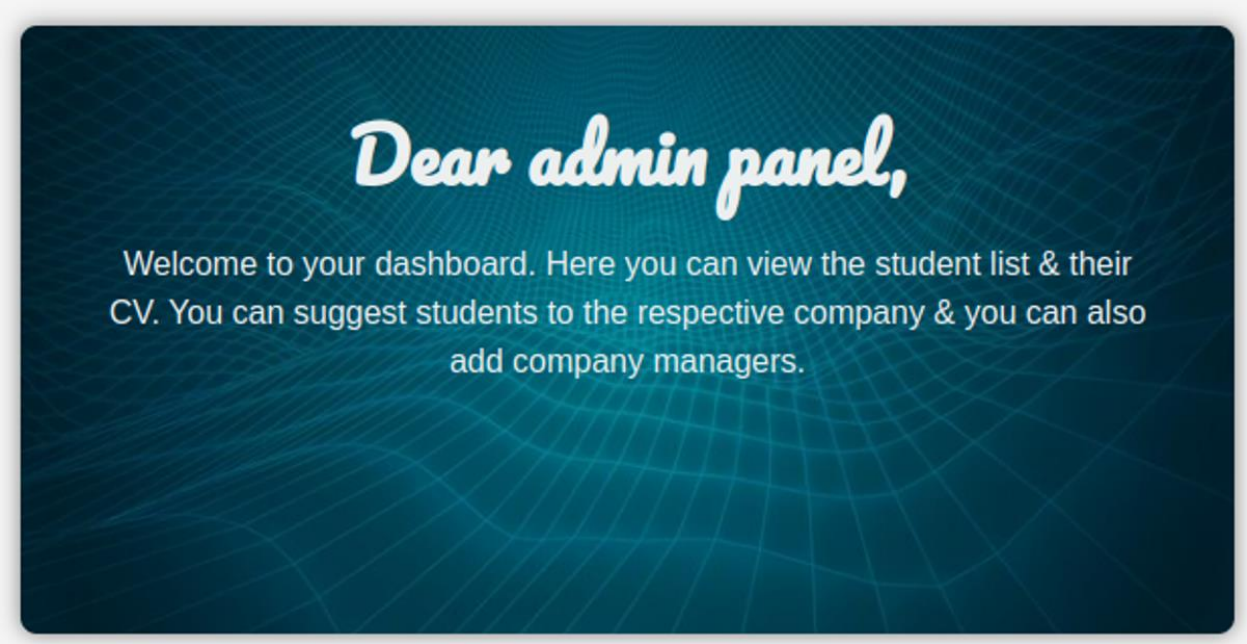
Select IPOC



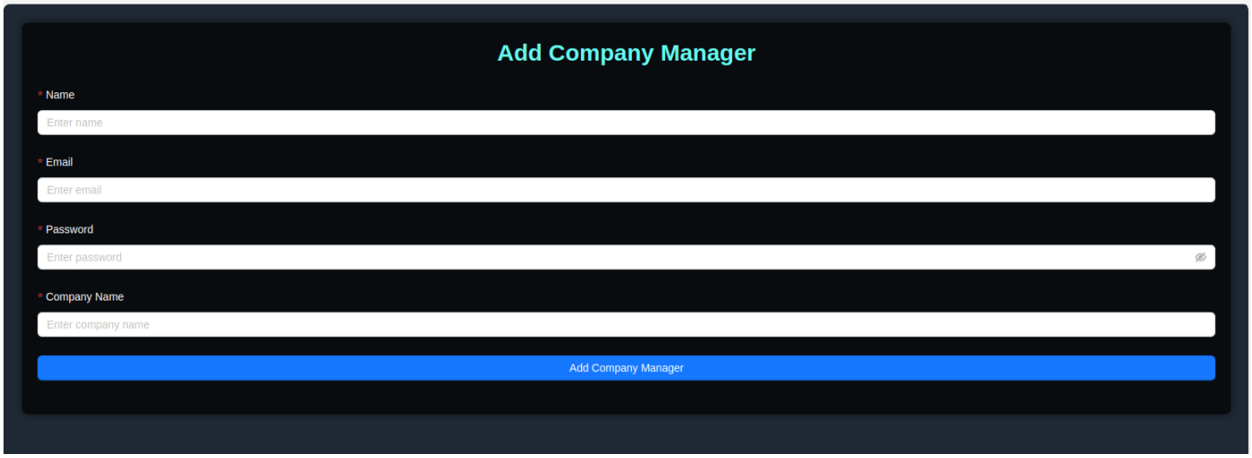
After completing the registration process, Login to the system



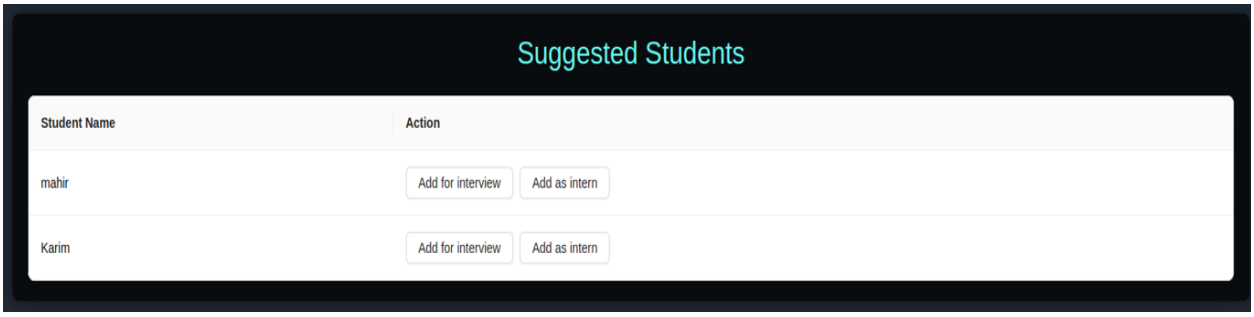
Successful Login goes to Admin panel



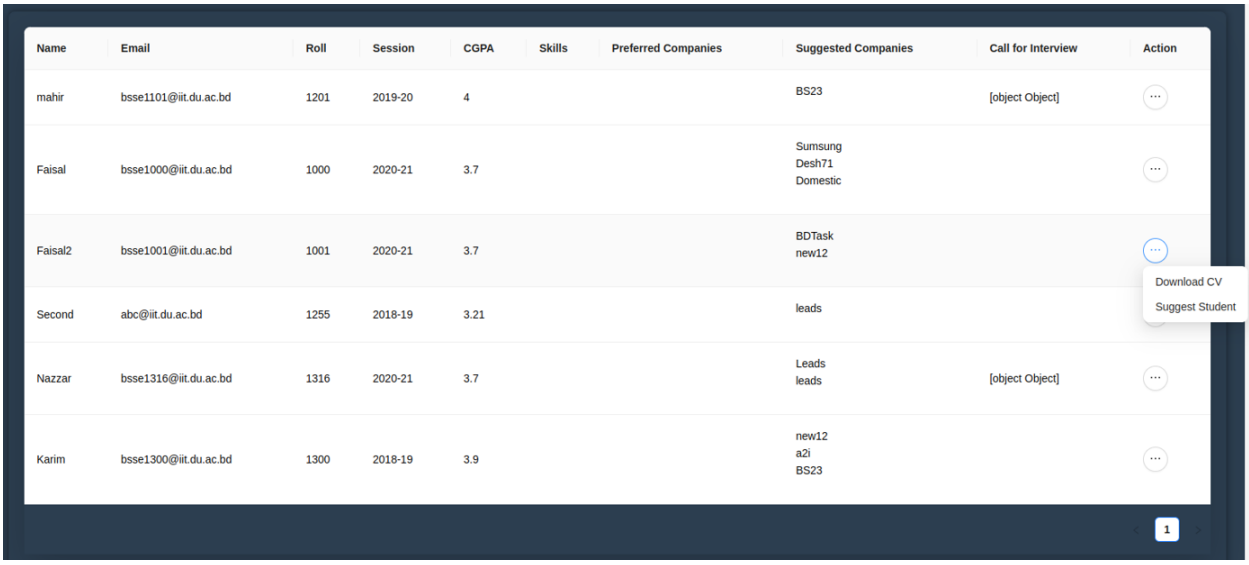
IPO head can add new company



IPOC members suggest candidates for interviews

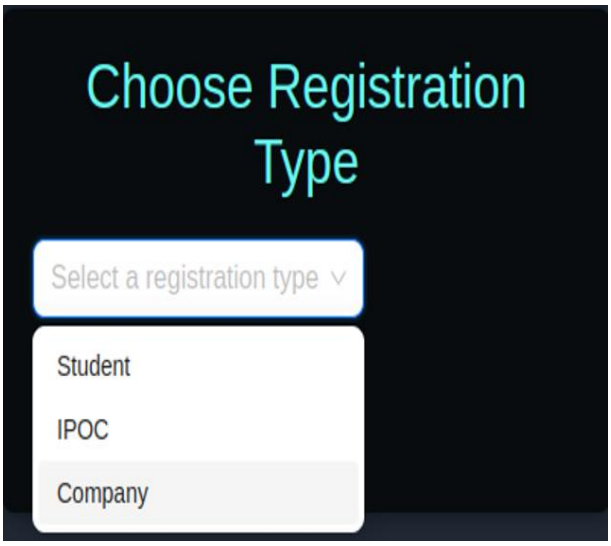


Views status

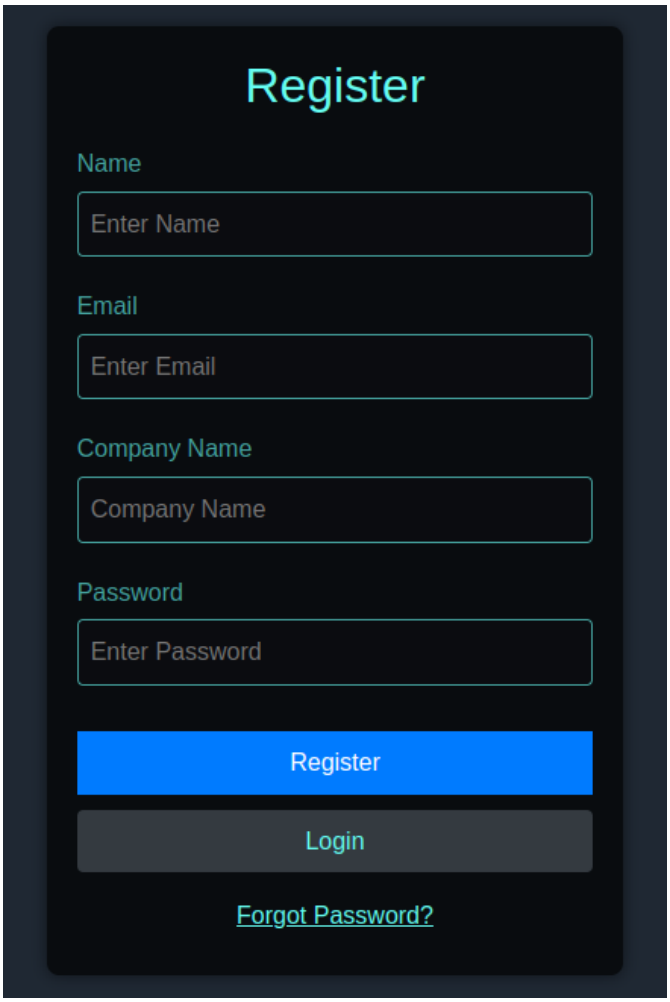


5.3 Company HR Guide

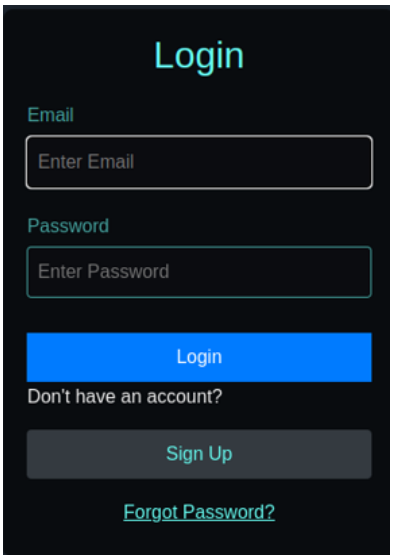
Create an account



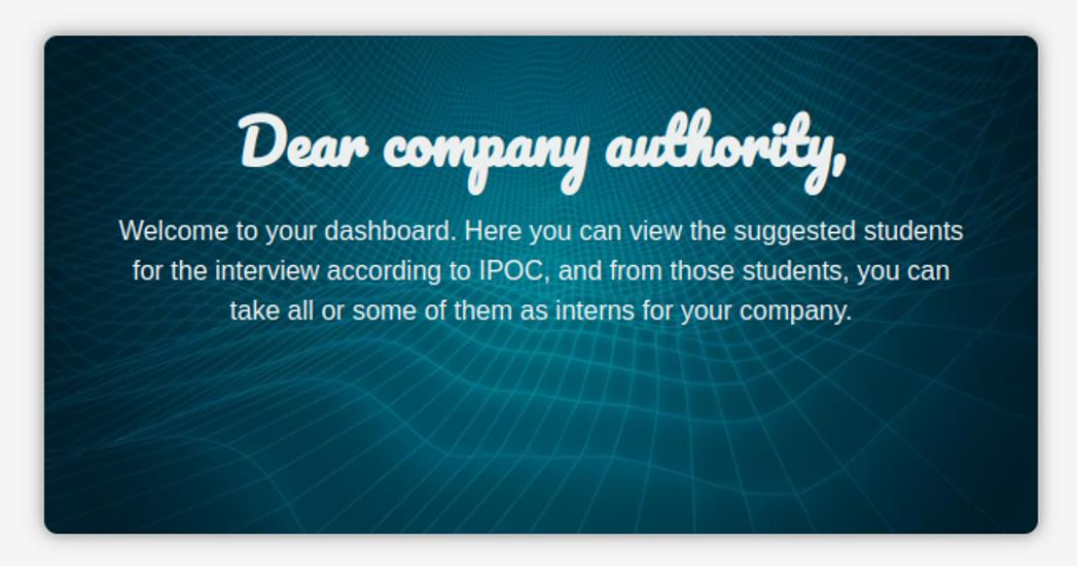
Select Company



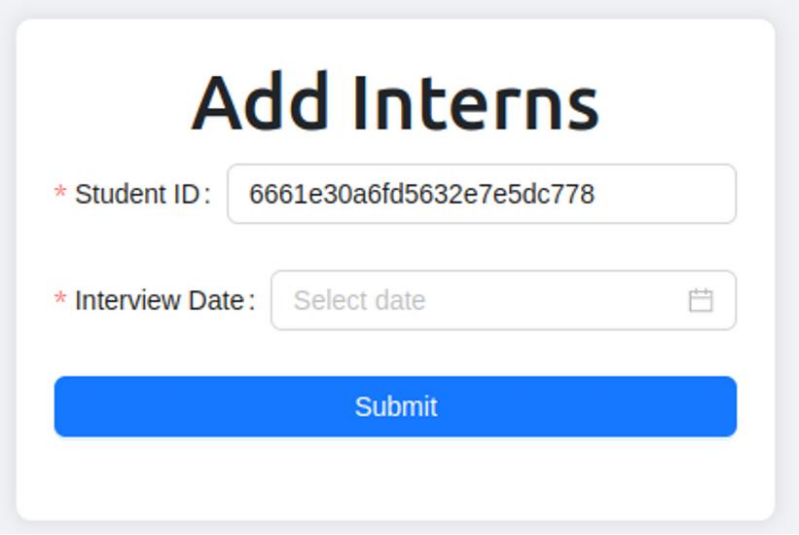
After completing the registration process, Login to the system



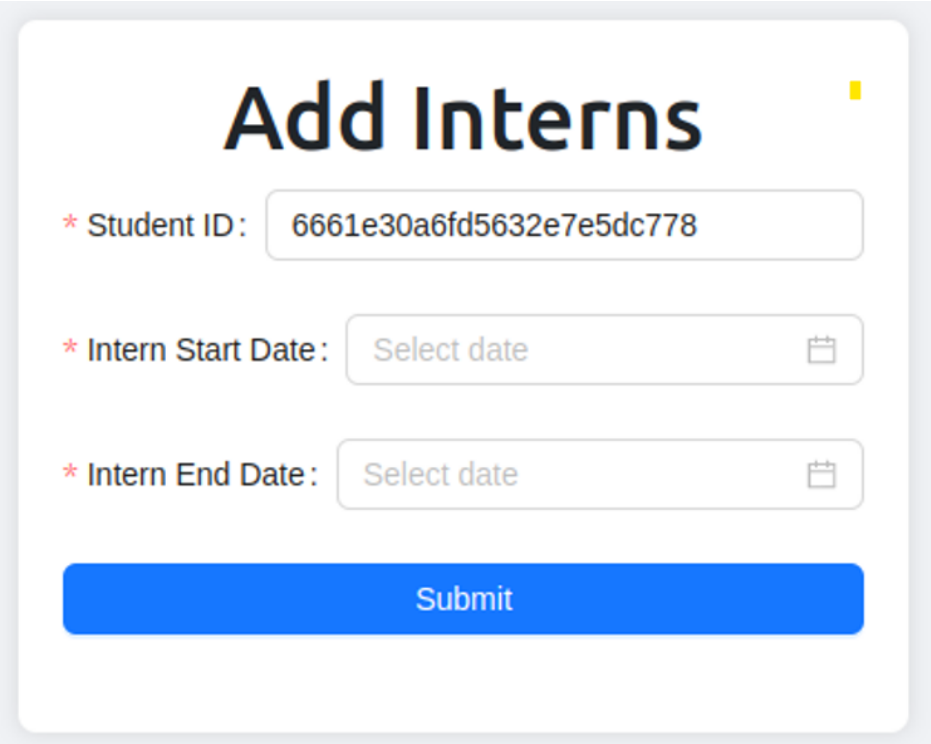
Successful Login goes to the Company Dashboard



Call for Interviews



Add as Interns



6. Challenges Faced

1. Multiple login systems based on user type.
2. Managing uploaded components from students.
3. Suggesting students from the admin dashboard to the company dashboard.
4. Adding interns from the company dashboard.
5. Selecting students for specific date ranges.
6. Displaying allocation status on both admin and student dashboards.

7. Limitations

1. The roles of IPOC members and the IPO head are merged to reduce complexity.
2. Feedback on CVs could not be implemented.
3. The addresses both of students & companies are not taken.
4. Appointment letters and continuous evaluation of interns by companies couldn't be implemented.

8. Conclusion and Future Work

The Intern and Placement Office Management (IPOM) system at IIT effectively facilitates the placement of BSSE 4th-year students into internships at reputed software companies. It streamlines processes for company registration, student applications, and status tracking, ensuring transparency and efficiency despite some limitations, like the merging of IPOC member and IPO head roles and the absence of CV feedback.

Future enhancements should focus on implementing feedback options, continuous intern performance evaluation, and generating appointment letters. Adding a mapping system for route and distance illustration between student addresses and companies will further enhance the system's functionality and user experience.

References

1. https://www.researchgate.net/publication/371501704_A_PROJECT_PROPOSAL_ON_Internship_Management_and_Portal_System_Submitted_By_Aashish_Katila_PAS076BCT001_Ashish_Acharya_PAS076BCT007_Nitesh_Devkota_PAS076BCT019_Saurab_Bhattarai_PAS076BCT037

2. https://www.academia.edu/34291111/Internships_Management_System

3. https://www.w3schools.com/