Intelligent Floor Plan Management System (IFMS)

MoveInSync Project Assingment

Submitted by: Sridhar Vasudevan
Roll Number: IIT2022163
Institute: Indian Institute of Information Technology, Allahabad
Submission Date: 11th November 2025

Contents

1. Abstract

The *Intelligent Floor Plan Management System (IFMS)* is a full-stack web platform designed to streamline floor plan management and optimize meeting room bookings. It allows administrators to create, modify, and delete floor plans while ensuring data consistency using version control. The system provides users with intelligent meeting room suggestions based on capacity and availability. It integrates offline synchronization, authentication, and robust error-handling mechanisms to ensure reliability and scalability.

2. Introduction

2.1. Problem Statement

Organizations face inefficiencies in manual seat and meeting room management, often leading to scheduling conflicts and poor resource utilization. A centralized digital system can automate this process and improve coordination.

2.2. Objectives

- Develop a web-based system for managing floor plans and room bookings.
- Implement smart room suggestions based on participant requirements.
- Ensure consistency via version control and conflict resolution.
- Enable offline functionality and seamless synchronization.
- Maintain secure user authentication and authorization.

3. System Architecture

The system follows a client-server architecture with three main layers:

1. **Frontend:** Built with React.js and Material UI for responsive design.
2. **Backend:** Developed using Node.js and Express.js for handling REST APIs.
3. **Database:** MongoDB Atlas for cloud-based NoSQL storage.

4. Technology Stack

Component	**Technology Used**
Frontend	React.js, Material UI
Backend	Node.js, Express.js
Database	MongoDB Atlas
Authentication	JWT, bcrypt
Tools	Postman, Git, VS Code

5. Functional Modules

5.1. Authentication Module

Implements JWT-based authentication with bcrypt password hashing. Sessions are stored in LocalStorage, and login status is reflected dynamically in the navigation bar.

5.2. Admin Functionalities

- Add, modify, and delete floor plans.
- Manage meeting rooms.
- Resolve version conflicts during concurrent updates.
- Work offline and sync updates on reconnection.

5.3. User Functionalities

- Login/Signup using secure credentials.
- View all floor plans and their rooms.
- Get intelligent meeting room suggestions.
- Book or unbook rooms and view personal bookings.

5.4. Meeting Room Optimization

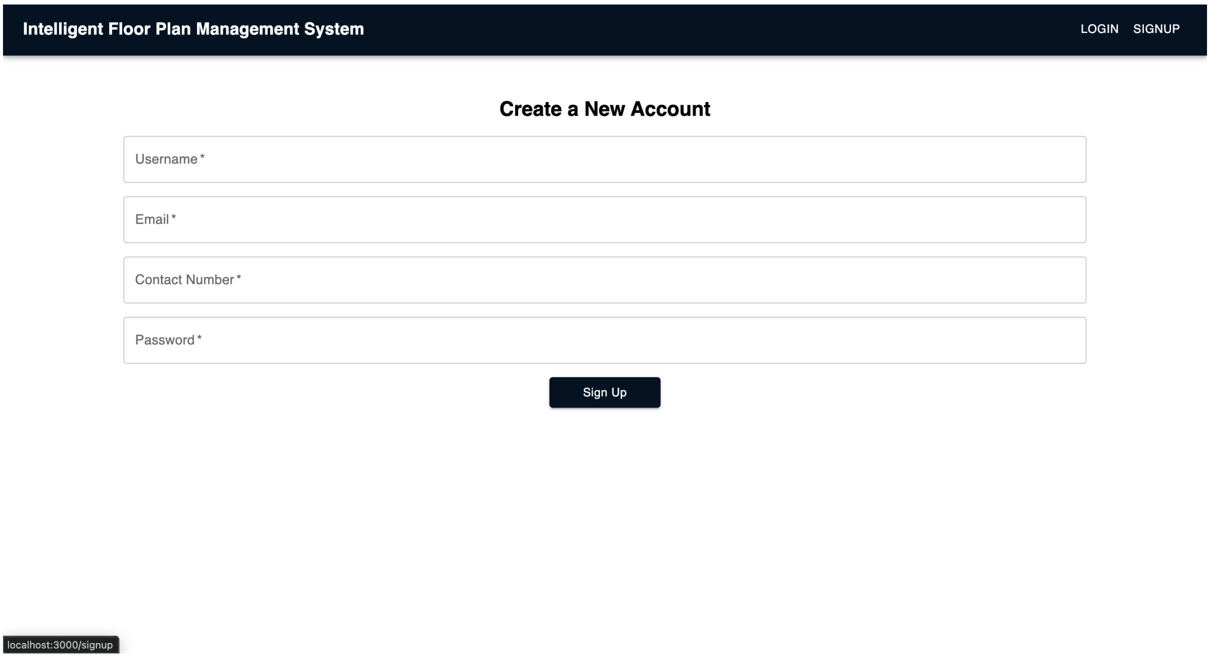
Meeting room suggestions are generated based on room capacity, usage frequency, and proximity. The system ranks available rooms and recommends the best match for given participants.

5.5. Offline and Version Control Mechanism

Offline changes are temporarily stored locally. Version control ensures conflict-free updates by comparing timestamps and version numbers.

6. Working Demo (Screenshots)

6.1. SignUp Page



Intelligent Floor Plan Management System

Create a New Account

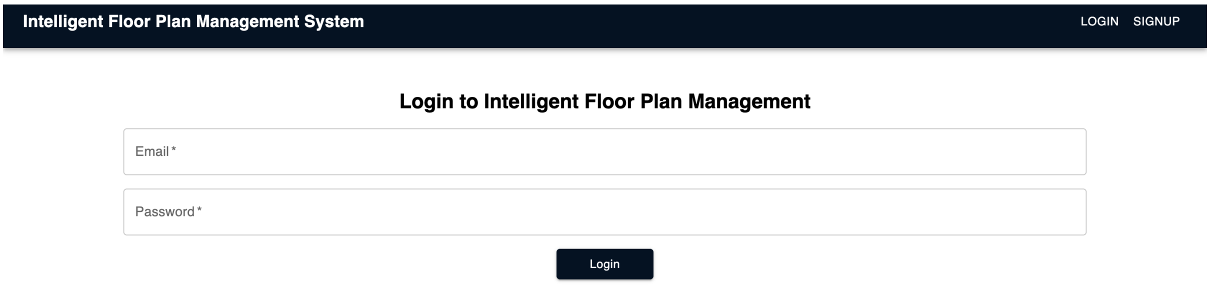
Username *

Email *

Contact Number *

Password *

6.2. Login Page



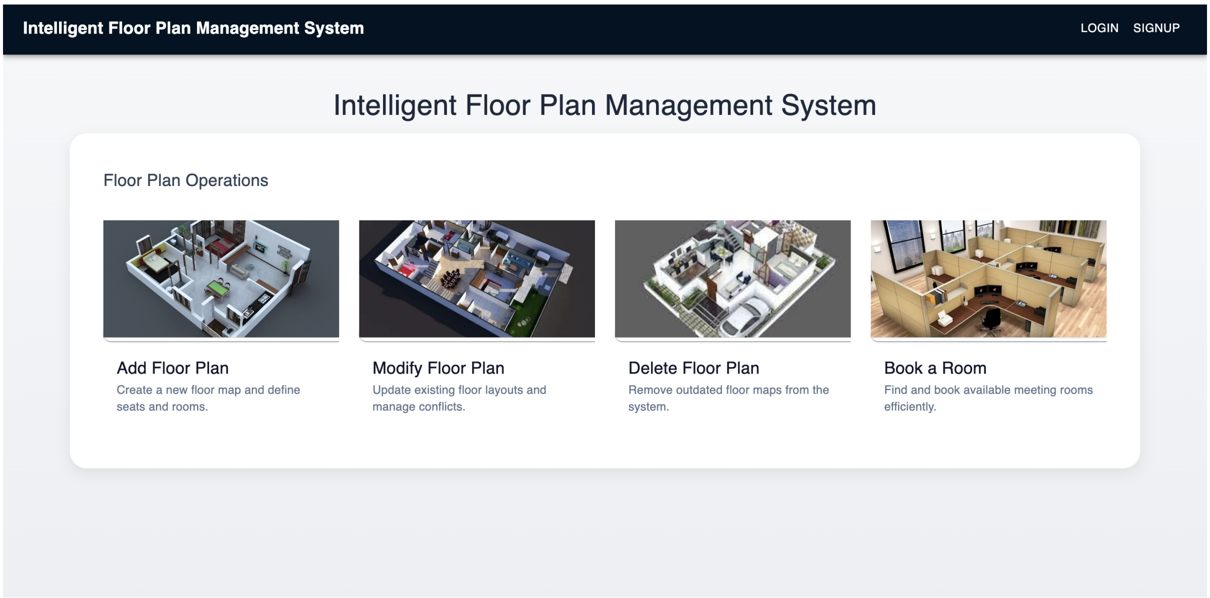
Intelligent Floor Plan Management System

Login to Intelligent Floor Plan Management

Email *

Password *

6.3. Home Page



Intelligent Floor Plan Management System

Floor Plan Operations

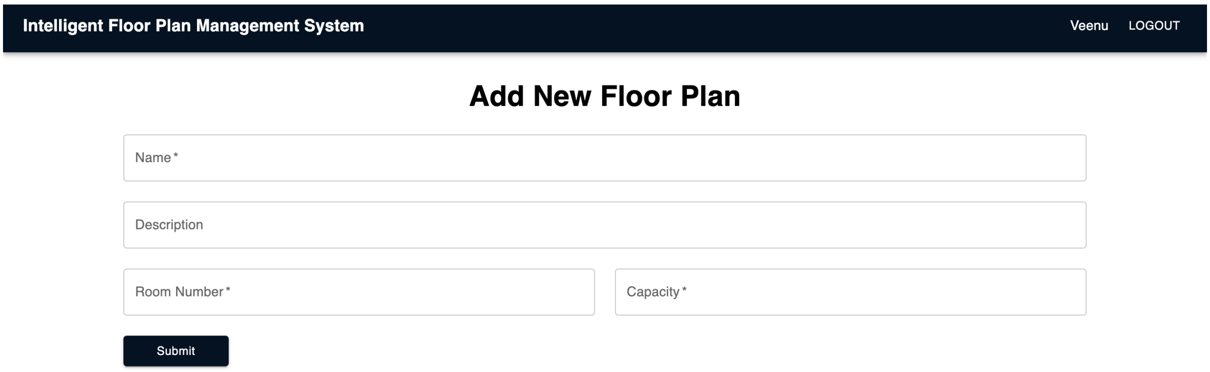
Add Floor Plan
Create a new floor map and define seats and rooms.

Modify Floor Plan
Update existing floor layouts and manage conflicts.

Delete Floor Plan
Remove outdated floor maps from the system.

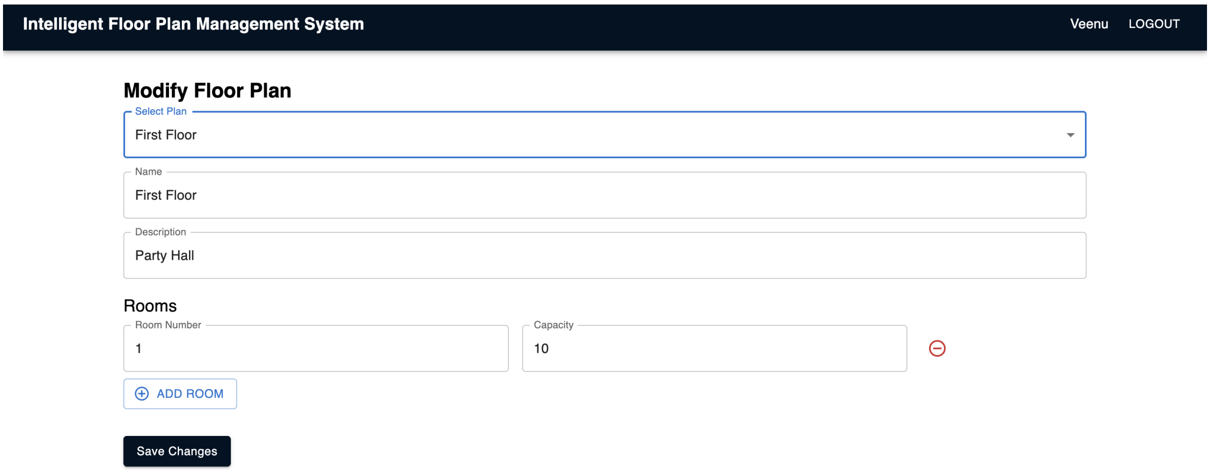
Book a Room
Find and book available meeting rooms efficiently.

6.4. Add Floor Plan Page



Add New Floor Plan

6.5. Modify Floor Plan Page



Modify Floor Plan

Select Plan: First Floor

Name: First Floor

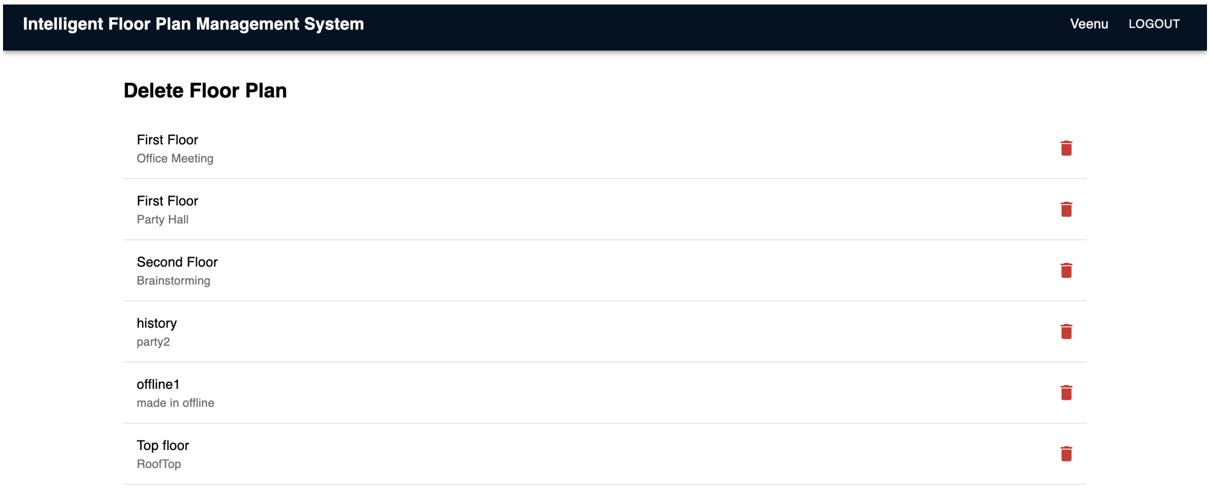
Description: Party Hall

Rooms

Room Number: 1

Capacity: 10

6.6. Delete Floor Plan Page



Delete Floor Plan

First Floor
Office Meeting

First Floor
Party Hall

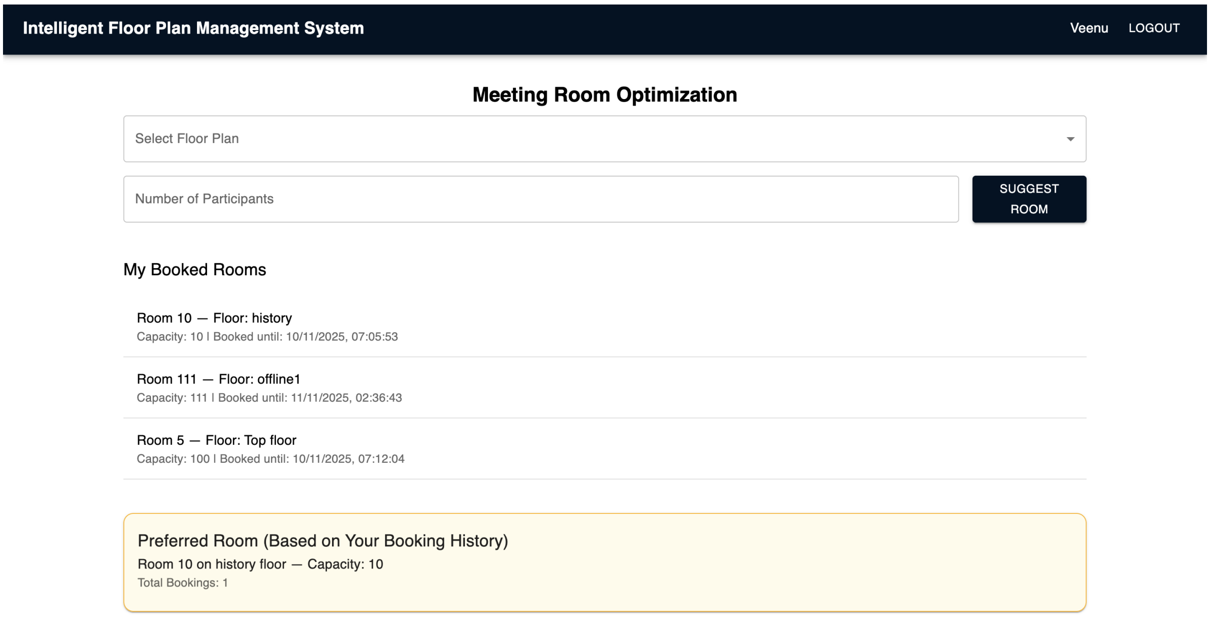
Second Floor
Brainstorming

history
party2

offline1
made in offline

Top floor
RoofTop

6.7. Book a Room Page



Intelligent Floor Plan Management System

Meeting Room Optimization

My Booked Rooms

Room 10 — Floor: history
Capacity: 10 | Booked until: 10/11/2025, 07:05:53

Room 111 — Floor: offline1
Capacity: 111 | Booked until: 11/11/2025, 02:36:43

Room 5 — Floor: Top floor
Capacity: 100 | Booked until: 10/11/2025, 07:12:04

Preferred Room (Based on Your Booking History)
Room 10 on history floor — Capacity: 10
Total Bookings: 1

7. System Design

7.1. Database Schema

Listing 1: Floor Plan Schema Example

```
const floorPlanSchema = new mongoose.Schema({
  name: String,
  description: String,
  version: { type: Number, default: 1 },
  rooms: [{
    roomNumber: Number,
    capacity: Number,
    booked: Boolean,
    bookedBy: String,
    bookingCount: { type: Number, default: 0 },
    lastBookedAt: Date
  }],
  seats: [{
    seatNumber: Number,
    occupied: Boolean
  }]
});
```

7.2. Component Structure

```
src/
 App.js
 Navbar.js
 Home.js
 AddPlan.js
```

```
ModifyPlan.js
DeletePlan.js
BookRoom.js
Login.js
Signup.js
Logout.js
```

8. Implementation Details

8.1. Authentication Example

Listing 2: Login Endpoint

```
app.post('/login', async (req, res) => {
  const { email, password } = req.body;
  const user = await User.findOne({ email });
  if (!user || !(await bcrypt.compare(password, user.password)))
    return res.status(401).json({ message: 'Invalid credentials' });

  const token = jwt.sign({ id: user._id }, process.env.JWT_SECRET, {
     ↪ expiresIn: '1h' });
  res.json({ message: 'Login successful', token, user });
});
```

8.2. Meeting Room Suggestion

Listing 3: Room Suggestion Algorithm

```
app.post('/floorplans/:id/suggest-room', async (req, res) => {
  const { participants } = req.body;
  const plan = await FloorPlan.findById(req.params.id);
  const available = plan.rooms.filter(r => !r.booked);
  const best = available
    .filter(r => r.capacity >= participants)
    .sort((a, b) => a.capacity - b.capacity)[0];
  res.json({ suggestedRoom: best });
});
```

9. Algorithm and Complexity Analysis

Feature	**Algorithm**	**Time Complexity**	**Space Complexity**
Room Suggestion	Greedy Selection (sort by capacity)	O(n log n)	O(n)
Version Control	Version Comparison	O(1)	O(1)
Offline Sync	Local Merge Algorithm	O(n)	O(n)

10. Error and Exception Handling

- All backend APIs use try-catch blocks for error resilience.

- HTTP codes used: 400 (Bad Request), 401 (Unauthorized), 404 (Not Found), 409 (Conflict), 500 (Server Error).
- Frontend alerts provide descriptive error messages to the user.

11. System Failure Recovery

- Offline changes stored locally until connection is restored.
- Version control prevents conflicting writes.
- System designed for minimal downtime and consistent synchronization.

12. Performance Optimization

- Optimized Mongoose queries for lower latency.
- Frontend reuses state and avoids redundant fetch calls.
- Lightweight payloads to minimize bandwidth usage.

13. Trade-offs and Design Decisions

Trade-off	**Decision**	**Rationale**
NoSQL vs SQL	MongoDB	Flexible schema for nested objects (rooms, seats).
Real-time vs Version Control	Version Control	Simpler conflict management with version field.
LocalStorage vs IndexedDB	LocalStorage	Quick and simple for offline caching.

14. Results and Outputs

- Floor plans successfully added, modified, and deleted.
- Intelligent room suggestions function accurately.
- Version conflicts resolved seamlessly.
- Offline changes synchronize correctly on reconnection.

15. Conclusion

The *Intelligent Floor Plan Management System* automates office resource allocation and improves collaboration through intelligent booking and conflict resolution. Its modular, fault-tolerant, and secure design ensures scalability and adaptability for future enterprise-level integration.

16. Future Enhancements

- Integrate Redis caching for faster data access.
- Implement analytics for room usage patterns.
- Add role-based access control (Admin/User).
- Enable real-time collaboration with WebSockets.

17. References

- React.js Documentation: https://react.dev
- Node.js Documentation: https://nodejs.org
- MongoDB Atlas: https://www.mongodb.com/atlas
- Material UI: https://mui.com
- Express.js Documentation: https://expressjs.com