Case study of Intelligent Floor Plan Management System

Functional Requirement:

→ User should able to log in or register
→ User should able to search the requirements such as meeting room, visiting room, etc
→ The user should be able to see if the room is already booked or available.
→ If a room is available user should be able to book it.
→ User should able to book a parking lot based on availability.
→ confirmation and notification.

Not Functional Requirement:

→ Scalable
→ Available/reliable.

Actors:

→ Customers
→ Operators

Entity:

→ User
→ Floor
→ Room
→ location
→ booking
→ availability
→ Notification

Below is a basic representation of the provided Low-Level Design in C++

```
#include <iostream>
#include <vector
using namespace std;
// User Authentication Module
class User {
public:
   string username;
   string password;
   string role;
   User(string uname, string pwd, string userRole)
       : username(uname), password(pwd), role(userRole) {}
};
class AuthenticationManager {
public:
   AuthenticationManager() {}
   bool authenticateUser(string username, string password) {
       // Authentication logic
       return true; // Placeholder logic
   }
};
// Data Management Module
class FloorPlan {
public:
   int floorNumber;
   string floorData;
   map<string, bool> occupancyData;
   map<string, Reservation> reservationSchedule;
   void updateOccupancyData(string space, bool isOccupied) {
       // Update occupancy data logic
   }
   string getSpaceDetails(string space) {
       // Retrieve details about a specific space logic
       return ""; // Placeholder logic
   }
   void updateReservationSchedule(string space, Reservation reservation) {
       // Update reservation schedule logic
   }
};
class ReservationValidator {
public:
   ReservationValidator(FloorPlan& floorPlan) : floorPlan(floorPlan) {}
```

```
    bool validateReservation(Reservation reservation) {
        // Validate reservation logic
        return true; // Placeholder logic
    }
private:
    FloorPlan& floorPlan;
};
class ReservationManager {
public:
    ReservationManager(ReservationValidator& validator) :
reservationValidator(validator) {}
    void makeReservation(User user, string space, string startTime, string
endTime) {
        // Make reservation logic
    }
private:
    ReservationValidator& reservationValidator;
};
// Real-Time Updates Module
class RealTimeUpdater {
public:
    RealTimeUpdater(FloorPlan& floorPlan) : floorPlan(floorPlan) {}
    void notifyUpdates() {
        // Notify updates logic
    }
private:
    FloorPlan& floorPlan;
};
// IoT Integration Module
class IoTSensorInterface {
public:
    virtual string getSensorData() = 0;
};
class OccupancySensor : public IoTSensorInterface {
public:
    string getSensorData() override {
        // Implementation for occupancy sensor
        return ""; // Placeholder logic
    }
};
class EnvironmentSensor : public IoTSensorInterface {
public:
    string getSensorData() override {
```

```
        // Implementation for environment sensor
        return ""; // Placeholder logic
    }
};
// Notification System Module
class NotificationManager {
public:
    void sendNotification(User user, string message) {
        // Send notification logic
    }
};
// Conflict Resolution Mechanism
class ConflictResolver {
public:
    ConflictResolver(VersionControlSystem& versionControl) :
versionControl(versionControl) {}
    void resolveConflicts(FloorPlan newFloorPlan, FloorPlan
existingFloorPlan) {
        // Conflict resolution logic
    }
private:
    VersionControlSystem& versionControl;
};
// Version Control System
class VersionControlSystem {
public:
    vector<FloorPlan> versionHistory;
    void commitVersion(FloorPlan floorPlan, User user, string timestamp) {
        // Save the version with user and timestamp information
    }
    void mergeVersions(FloorPlan newVersion, FloorPlan existingVersion) {
        // Merge floor plans from two versions
    }
};
// Offline Mechanism for Admins
class OfflineStorageManager {
public:
    OfflineStorageManager(FloorPlan& floorPlan) : floorPlan(floorPlan) {}
    void saveLocally(FloorPlan floorPlan) {
        // Save floor plan locally
    }
    string getLocalChanges() {
        // Retrieve locally saved changes
```

```
        return ""; // Placeholder logic
    }
private:
    FloorPlan& floorPlan;
};
class SynchronizationManager {
public:
    SynchronizationManager(OfflineStorageManager& offlineStorage, string
serverCommunication)
        : offlineStorage(offlineStorage),
serverCommunication(serverCommunication) {}
    void syncChanges() {
        // Synchronize local changes with the server
    }
private:
    OfflineStorageManager& offlineStorage;
    string serverCommunication;
};
// Meeting Room Optimization
class MeetingRoomBookingSystem {
public:
    MeetingRoomBookingSystem(FloorPlan& floorPlan, RecommendationSystem&
recommendationSystem)
        : floorPlan(floorPlan), recommendationSystem(recommendationSystem)
{}
    void bookMeetingRoom(User user, string space, string startTime, string
endTime) {
        // Book meeting room based on participants and requirements
    }
private:
    FloorPlan& floorPlan;
    RecommendationSystem& recommendationSystem;
};
class RecommendationSystem {
public:
    RecommendationSystem(FloorPlan& floorPlan) : floorPlan(floorPlan) {}
    void suggestMeetingRoom(int participants) {
        // Suggest meeting room based on participants and availability
    }
    void updateSuggestions(string bookingChanges) {
        // Update suggestions dynamically based on bookings and capacity
changes
    }
```

```
private:
   FloorPlan& floorPlan;
};
int main() {
   // Placeholder main function
   return 0;
}
```