**Business Requirements Document (BRD)**

**Project Title:** SpaceSync: Integrated Space Management

**1. Project Overview**

SpaceSync is an innovative project aimed at transforming the management of physical spaces through a unified system with three key modules: parking, workspace, and event space management.

The parking module optimizes vehicle allocation with real-time availability updates, enhancing user convenience and reducing congestion. The workspace module streamlines the reservation of desks and meeting rooms, fostering a productive environment with an intuitive booking interface. Lastly, the event space management module facilitates efficient scheduling and organization of events, ensuring seamless resource allocation and attendee management.

By integrating these functionalities, SpaceSync provides a cohesive platform that enhances user experience, promotes efficiency, and maximizes the utilization of physical spaces, ultimately leading to a more organized and productive atmosphere.

**2. Business Objectives**

* **Optimize Space Utilization:** Maximize the effective use of parking, workspaces, and event spaces through efficient scheduling and resource allocation.
* **Enhance User Experience:** Develop an intuitive platform that simplifies the booking and management processes for users across all space types.
* **Facilitate Real-Time Availability Management:** Implement real-time tracking of space availability to minimize conflicts and ensure smooth transitions between bookings.
* **Streamline Booking Processes**: Simplify and standardize the booking process across parking, workspace, and event space modules to enhance efficiency and reduce scheduling conflicts.

**3. Stakeholders**

* **Employees:** Staff members who can book parking spaces and workspaces through the system, facilitating their daily commute and workspace needs for increased productivity.
* **Managers:** Team leaders who can book workspaces, meeting rooms, and event spaces, enabling effective collaboration and resource allocation for their teams.
* **Administrators:** System administrators responsible for managing the application’s functionality, user accounts, and overall maintenance to ensure a seamless experience for all users.
* **Security Personnel:** Staff tasked with managing parking space usage, ensuring that slots are automatically marked as unbooked when vehicles leave, maintaining accurate availability and security of the parking areas.

**4. Functional Requirements**

**4.1 User Authentication and Authorization**

* Users must be able to create accounts and log in securely.
* The system should provide role-based access, allowing different functionalities based on user roles.

**4.2 Parking Management**

* Users should be able to view real-time availability of parking spaces.
* Users must be able to book and cancel parking slots through the system.
* The system should automatically mark parking slots as unbooked when vehicles leave, managed by security personnel.
* Notifications should be sent to users for booking confirmations.

**4.3 Workspace Management**

* Employees must be able to search for and reserve workspaces based on shifts, availability and project.
* Users should be able to modify or cancel their workspace bookings.

**4.4 Event Space Management**

* Managers should be able to view available event spaces and book them as needed.
* The system must allow for detailed event scheduling, including setup and teardown times.

**4.5 Administrative Functions**

* Administrators should have access to manage user accounts, roles, and permissions.
* The system must provide reporting tools for analyzing space utilization and user activity.

**4.6 User Interface**

* The system must provide a responsive and user-friendly interface for all stakeholders.
* A dashboard should be available for users to view their bookings and upcoming events.

**4.7 Alerts**

* The system must send notifications for booking confirmations in-app alerts.

**5. Non-Functional Requirements**

**5.1 Security**

* **Basic Encryption:** Sensitive data is encrypted in transit using HTTPS.
* **Role-Based Access:** Implement role-based access control for authorized user actions.

**5.2 Usability**

* **User Interface**: The application will have an intuitive and clean design, optimized for desktop use.
* **Responsiveness:** The application will be fully responsive across devices and screen sizes.
* **Basic Accessibility:** The application will cater to users with varying technical abilities.

**5.3 Maintainability**

* **Code Simplicity:** The code will be modular, well-organized, and documented for easy maintenance.
* **Automated Testing:** Implement basic automated tests to ensure stability and ease future modifications.

**5.4 Scalability**

* **Basic Scalability:** The system will scale horizontally to accommodate more users and communities.

**6. Project Architecture**

The SpaceSync project follows a microservices architecture with a frontend built using React with TypeScript, and a backend developed with Node.js using TypeScript. The system utilizes multiple microservices to handle core functionalities such as parking management, workspace reservations, event scheduling, and user management. MongoDB is employed for its scalability and flexibility in managing dynamic data related to space usage and bookings.

* **Front End:** React with TypeScript for a type-safe user interface.
* **Back End:** Node.js with TypeScript, using RESTful APIs to manage business logic and facilitate communication between services.
* **Microservices:** Each core functionality is encapsulated in separate microservices to ensure modularity and maintainability.
* **Database:** MongoDB for scalable data storage and retrieval, managing entities like user profiles, bookings and events.

**7. API Documentation**

**7.1 User Management**

* **POST** /users - Create a new user.
* **GET** /users/{user\_id} - Retrieve user details by user ID.
* **PUT** /users/{user\_id} - Update user information by user ID.
* **DELETE** /users/{user\_id} - Delete a user by user ID.

**7.2 Event Space Management**

* **POST** /eventspaces - Create a new event space.
* **GET** /eventspaces/{eventspace\_id} - Retrieve event space details by ID.
* **PUT** /eventspaces/{eventspace\_id} - Update event space information by ID.
* **DELETE** /eventspaces/{eventspace\_id} - Delete an event space by ID.

**7.3 Event Space Bookings**

* **POST** /eventspace\_bookings - Create a new event space booking.
* **GET** /eventspace\_bookings/{eventspace\_reservation\_id} - Retrieve event space booking details by ID.
* **DELETE** /eventspace\_bookings/{eventspace\_reservation\_id} - Cancel an event space booking by ID.

**7.4 Workspace Management**

* **POST** /workspaces - Create a new workspace.
* **GET** /workspaces/{workspace\_id} - Retrieve workspace details by ID.
* **PUT** /workspaces/{workspace\_id} - Update workspace information by ID.
* **DELETE** /workspaces/{workspace\_id} - Delete a workspace by ID.

**7.5 Workspace Bookings**

* **POST** /workspace\_bookings - Create a new workspace booking.
* **GET** /workspace\_bookings/{workspace\_reservation\_id} - Retrieve workspace booking details by ID.
* **DELETE** /workspace\_bookings/{workspace\_reservation\_id} - Cancel a workspace booking by ID.

**7.6 Parking Slot Management**

* **POST** /parking\_slots - Create a new parking slot.
* **GET** /parking\_slots/{parking\_id} - Retrieve parking slot details by ID.
* **PUT** /parking\_slots/{parking\_id} - Update parking slot information by ID.
* **DELETE** /parking\_slots/{parking\_id} - Delete a parking slot by ID.

**7.7 Parking Reservations**

* **POST** /parking\_reservations - Create a new parking reservation.
* **GET** /parking\_reservations/{parking\_reservation\_id} - Retrieve parking reservation details by ID.
* **DELETE** /parking\_reservations/{parking\_reservation\_id} - Cancel a parking reservation by ID.

**7.8 Feedback Management**

* **POST** /feedback - Create new feedback.
* **GET** /feedback/{feedback\_id} - Retrieve feedback details by ID.
* **PUT** /feedback/{feedback\_id} - Update feedback by ID.
* **DELETE** /feedback/{feedback\_id} - Delete feedback by ID.

**8. Database Schema Overview**

The application will include the following collections in its MongoDB database schema:

**User Collection:** Stores essential information about users, including their credentials, personal details, and roles within the organization. This collection facilitates user authentication and management.

**EventSpace Collection:** Contains details about available event spaces, including their location, capacity, and availability status. This collection supports the management and booking of event venues.

**EventSpaceBookings Collection:** Tracks reservations made by users for event spaces, including booking times and event details. This collection ensures organized management of event scheduling and user commitments.

**Workspace Collection :**Holds information about available workspaces, including their location, availability, and associated project details. This collection enables users to reserve workspaces that fit their needs.

**WorkspaceBookings Collection:** Records reservations made by users for specific workspaces, detailing the booking time and date. This collection manages workspace allocation and ensures efficient usage.

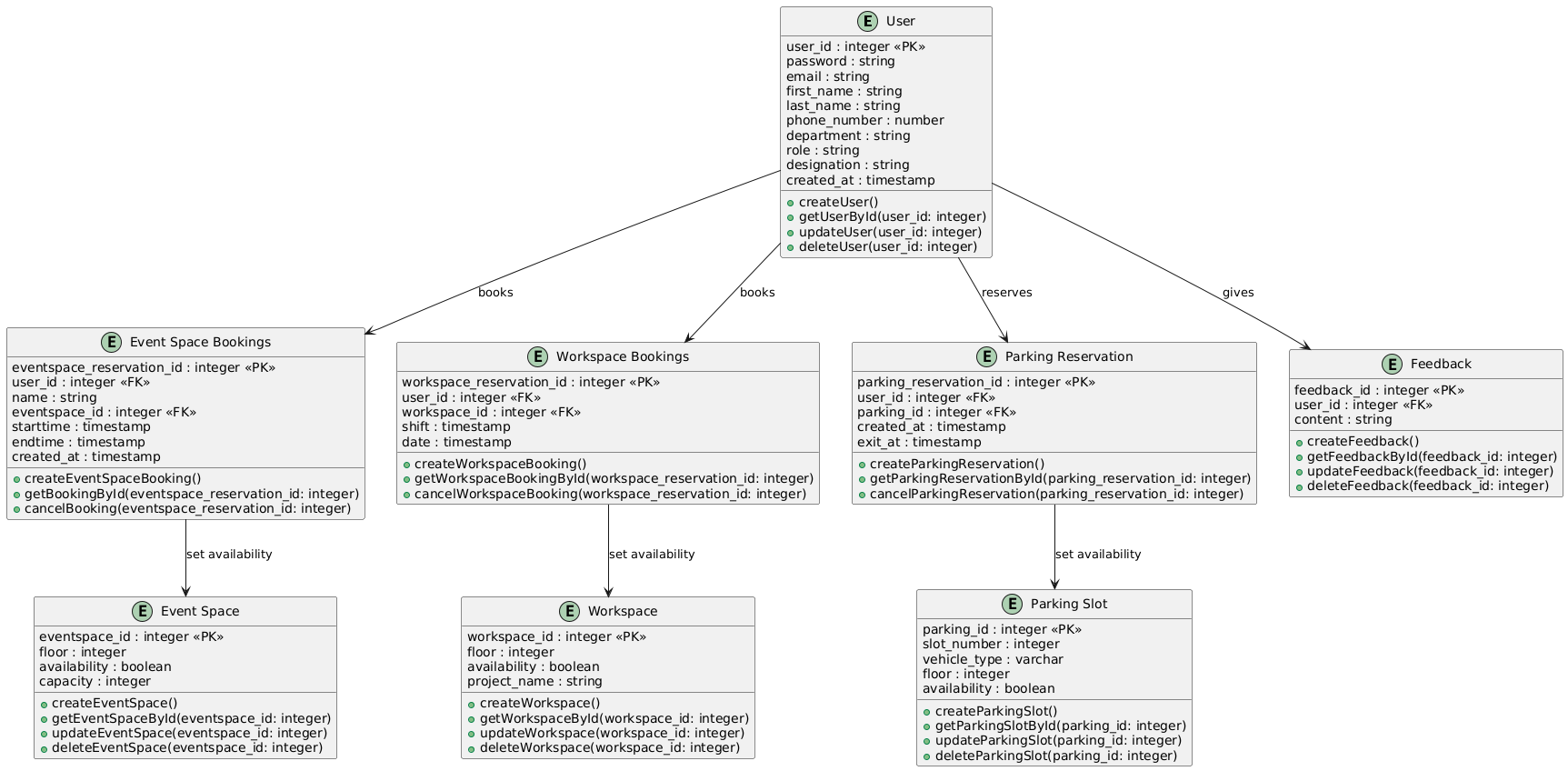
**ParkingSlot Collection:** Stores information about parking slots, including their number, type of vehicle accommodated, and availability status. This collection aids in managing parking resources for users.

**ParkingReservations Collection:** Tracks user reservations for parking slots, including timestamps for booking and exit. This collection helps manage parking availability and ensures accurate tracking of user parking.

**Feedback Collection:** Contains user feedback and comments about their experiences with the system and services. This collection allows for gathering insights and improving user satisfaction.



**9. UML Diagram**

****