**Business Requirements Document (BRD)**

**Project Title:** Flexi Work: Enterprise Resource and Event Management System

**1. Project Overview**

The project is about creating a system that helps a large company manage workspaces, parking slots, events, and employee feedback more efficiently. Right now, these tasks are done manually, leading to confusion and wasted resources. The system will allow employees to book workspaces and parking, register for events, and provide feedback through a single platform. It will also help administrators keep everything organized by automating resource allocation and generating reports.

Additionally, the solution will integrate external tools like Cloudinary for storing images, Mailchimp for sending notifications, and Google Calendar for event scheduling. This way, the company can improve communication, make better use of resources, and easily organize internal events, making life easier for both employees and management.

**2. Business Objectives**

 **Automate Resource Management:** Streamline workspace, parking, and vendor space management to reduce manual tasks and optimize resources.

 **Enhance Employee Engagement:** Simplify organizing events to boost participation and improve employee satisfaction.

 **Improve Operational Efficiency:** Centralize bookings for resources and events to enable quick access and real-time updates.

 **Data-Driven Decision Making:** Provide reports on resource usage and feedback to support informed management decisions.

 **Boost Communication and Collaboration:** Use tools like Mailchimp and Google Calendar for timely notifications and scheduling.

 **Minimize Resource Wastage:** Reduce unused resources with automated booking updates and cancellations.

**3. Stakeholders**

 **Employees:** End-users who book workspaces, parking slots, participate in events, and provide feedback.

 **Administrators:** Manage resources, bookings, workspace availability, and oversee system operations.

 **Event Organizers:** Plan and manage internal events, handle registrations, and track event participation.

 **Vendors:** Utilize and manage dedicated vendor spaces, including bookings and billing.

**4. Functional Requirements**

**4.1 User Management and Authentication**

**User Registration:** Employees, vendors, Manager can register on the platform.

**Profile Management:** Users can view their personal details.

**4.2 Workspace Management**

**Workspace Details :** Employees can search and book available workspaces, Admins can add, update, or delete workspace details and generate reports on workspace utilization.

**4.3 Parking Slot Management**

**Parking Details** : Employees can view and book parking slots, Admins can add new parking slots and can generate reports on parking utilization.

**4.4 Vendor Space Management**

**Vendor Space Details :** Admins can add and update vendor-specific workspaces, Vendor can book vendor spaces for specific durations.

**Billing Details :** Vendor can check the billing for the booked duration.

**4.5 Event Management**

**Event Calendar:** Event managercan create and update an event calendar.

**Access Event Calendar**: Employees can view the event calendar and stay updated on future events.

**External API:** We are using Cloudinary API for getting urls for the images uploaded by the administrator.

**LINK:** <https://api.cloudinary.com/v1_1/digbbzwlfx/image/upload>

**4.6 Feedback System**

**Raise Complaints:** Employees can submit complaints.

**Event Feedback:** Employees can submit feedback on events.

**Handle Feedback:** Admins review and use feedback for improvements.

**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.
* **Elasticity:** The system will dynamically allocate resources to maintain performance during peak usage.

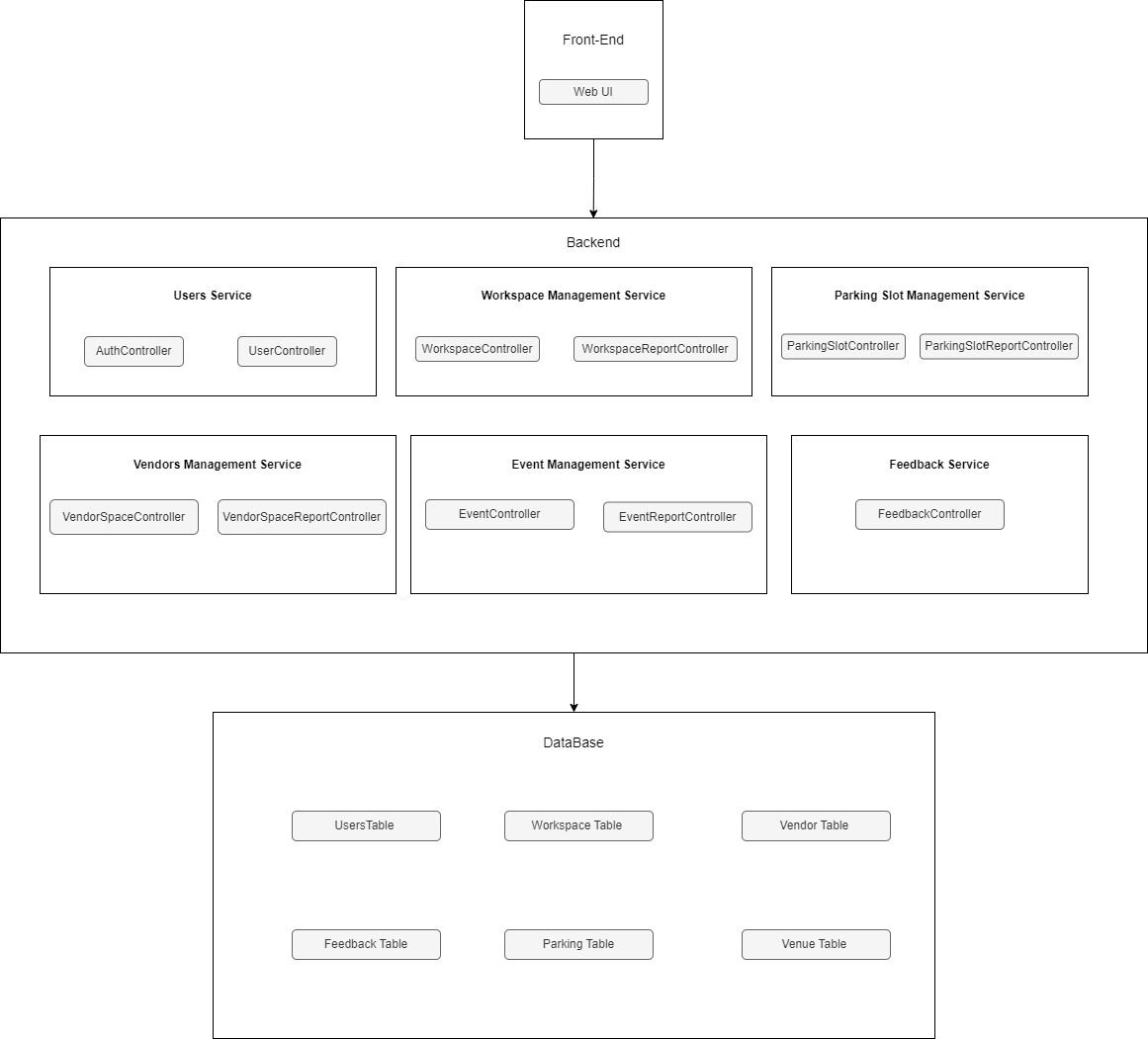
**5.5 Interoperability**

* **API Integration:** The system will expose RESTful APIs for integration with other services.
* **Modular Design:** The architecture will support modular components for easy integration of new features.

**6. Project Architecture**

The project architecture follows a microservices approach with a front end built using React and Redux for state management, and a back end developed with Nodejs Express. functionalities such as Workspace management, Parking slot allocation, Vendor space management and event management, complaint handling, and billing & payment processing. MongoDB is used for its scalability and flexibility in managing community-related data.

* **Front End**: React with Redux for state management (typescript).
* **Back End**: Nodejs Express RESTful APIs to manage the business logic.
* **Database:** MongoDB for efficient data storage and retrieval, managing



**7. Database Schema Overview**

The application will include the following tables :

**user**: Stores details of users including name, password, email, role, and contact information.

**vendor**: Contains information about vendors, the services they provide, and contract dates.

**vendor\_space**: Manages the details of vendor spaces, including location, capacity, and availability.

**vendor\_space\_booking**: Tracks bookings for vendor spaces, with associated user and time details.

**vendor\_billing**: Holds billing information for vendor space bookings, including total cost and payment status.

**workspace**: Represents workspaces with attributes like department, floor, location, and availability.

**workspace\_bookings**: Records bookings for workspaces, including start and end times for each booking.

**parking\_slot**: Stores information about parking slots, including their location, floor, and availability.

**parking\_reservation**: Tracks reservations made by users for parking slots, with start and end times.

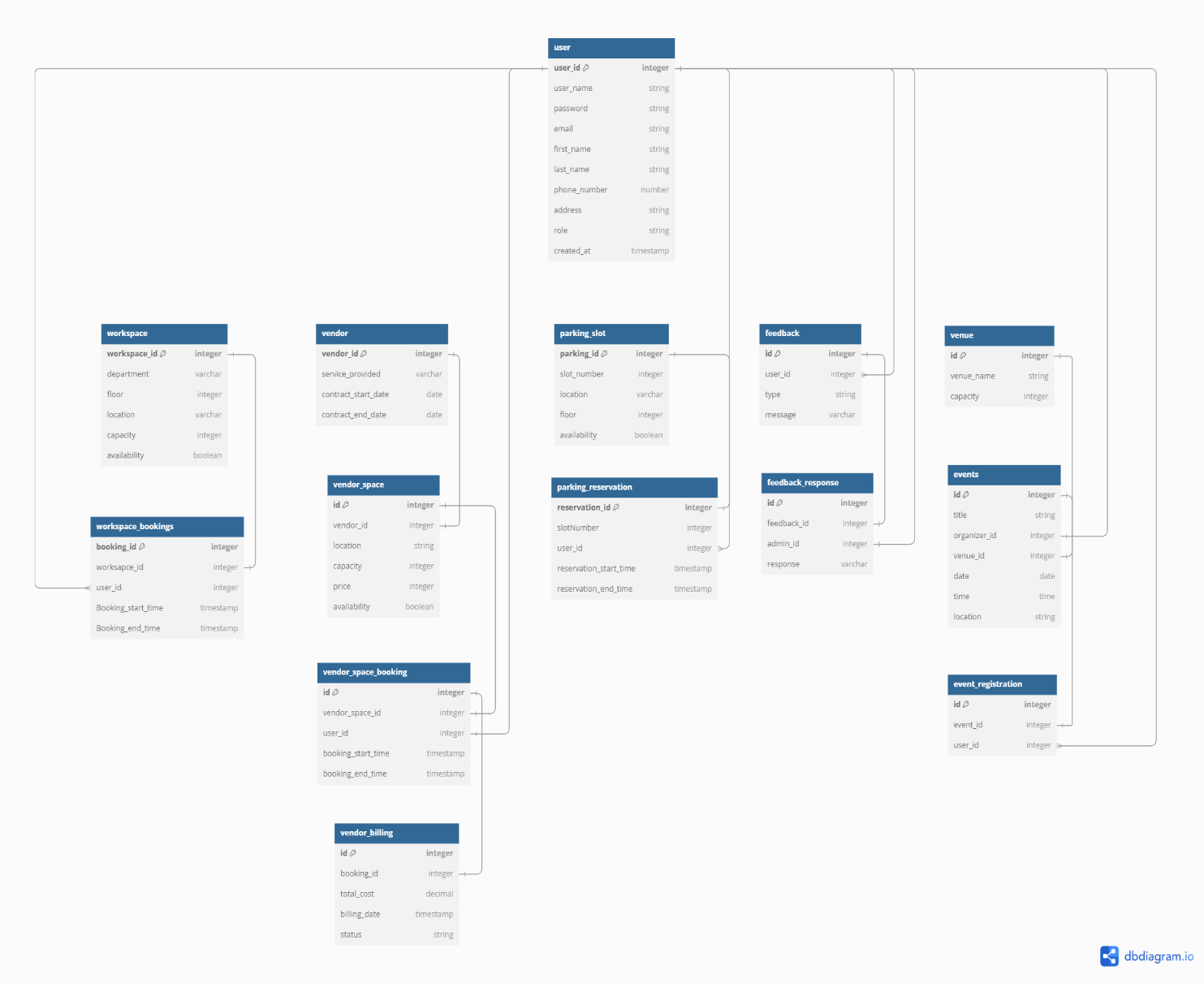
**feedback**: Captures user feedback, including the type of feedback and the associated message.

**feedback\_response**: Contains responses from admins to user feedback, linking to feedback messages.

**venue**: Manages event venues, including venue name and capacity details.

**events**: Stores details about events, such as title, organizer, venue, date, and time.

**event\_registration:** Tracks registrations of users for events, linking events with registered users.



**8. Assumptions**

* Stable Internet Connectivity: Users are expected to have access to stable internet connections to interact with the application effectively.
* Regular Input of Data: Users will regularly update their health, diet, and other relevant data to ensure accurate insights and recommendations.
* Integration with External APIs: The application will integrate with external APIs for functionalities like payment processing, WhatsApp notifications, and possibly other external services as needed.

**9. Constraints**

* Development and Deployment Timelines: Project milestones and deadlines must be adhered to, ensuring that the application is delivered and deployed within the specified time frame.
* Scalability and Performance: The system must be designed to handle varying loads, ensuring performance and scalability as the number of users and data volume grows.
* Compliance and Security: The application must comply with relevant data protection regulations and ensure the security of user data.

**10. UML Diagram**

