Functional and non-functional requirements with explanation, for the project “web interface to interact with assets”

With UML Diagram

Adam & Azeem

Users

View and filter assets: Users can view the list of assets and apply filters to narrow down the displayed results based on various criteria.

Search maps: Users can utilise search functionalities to locate specific assets on interactive maps.

Export data: Users can export asset data for external use.

Submit asset requests: Users can submit requests for new assets or modifications to existing assets.

Access departmental data: Users can access data related to specific departments as needed.

Registration screen for user accounts: New users can register their accounts through a dedicated registration screen.

Admins

Manage assets: Admins can add, update, and delete assets as needed.

Update datasets: Admins can update asset datasets to reflect the most current information.

Review logs: Admins have access to logs to review user activities and system events.

Manage users: Admins can manage user accounts, including creating, updating, and deleting accounts.

Set permissions: Admins can set permissions to control user access to different system functionalities.

Approve or reject user registrations: Admins are responsible for approving or rejecting new user registrations.

Perform bulk actions: Admins can efficiently perform bulk actions such as batch approvals, rejections, or deletions.

Functional Requirements

Map Display with Asset Integration

Integrate a JavaScript/PHP library to create interactive maps that can load and display asset data dynamically. Asset information, including latitude and longitude, will be loaded from a CSV file, and displayed on the map, allowing users to visualise asset locations in real time.

Asset Management

The system will enable comprehensive asset management, allowing users to categorise assets beyond the provided CSV data. Users can create and manage custom categories, select, and display assets, add new assets, and log job counts per asset. Administrators will have the authority to approve or reject modifications to assets, ensuring data integrity and oversight.

Data Integration

The system will support data integration by importing and parsing CSV/Excel files to extract asset data. This data will be stored in a structured database with category associations. An optional API will be available for dynamic data updates and external integrations. Data validation rules will be specified to handle missing latitude/longitude, duplicates, and incorrect formats, and the maximum CSV file size the system can manage will be clarified.

User Interface

The user interface will be designed to provide an intuitive experience for asset visualisation. It will include search and filtering capabilities with auto suggestions and ensure compliance with Bradford Council's branding and colours. The interface will align with accessibility standards (WCAG 2.1), be responsive on mobile devices, and include error handling for incorrect inputs, such as invalid asset searches or CSV format errors.

Non-Functional Requirements

Performance

Performance requirements ensure that the system operates efficiently, providing a smooth user experience without delays. This includes optimising map rendering and data loading speeds to ensure that assets are displayed quickly. To achieve this, the system should define the expected system load, specifying the number of concurrent users and assets that can be displayed without affecting performance. Techniques such as lazy loading and caching mechanisms should be implemented to load only necessary data at a time, reducing server load. Additionally, database indexing and optimised queries should be used to prevent slowdowns when fetching asset data, ensuring quick search and retrieval functionalities.

Scalability

Scalability ensures that the system can handle an increasing number of assets and users over time without performance degradation. As the asset database expands and more users access the system, it should be able to scale seamlessly by using efficient data structures, cloud computing solutions, and load-balancing techniques. This allows the system to adapt to growing demands without requiring a complete redesign.

Security

Security is critical for protecting user data and system integrity. The system must implement authentication and authorisation mechanisms, ensuring only authorised users can access specific features. Data storage and transmission should be secured using AES-256 encryption for data at rest and SSL/TLS encryption for data in transit. Additionally, security features like password reset functionality and Multi-Factor Authentication (MFA) should be implemented to enhance account security. To prevent unauthorised access, session timeouts should be defined for inactive users. Role-Based Access Control (RBAC) should be enforced to grant appropriate permissions based on user roles, while audit logs should track user and admin activities for accountability.

Usability

Usability ensures that the system is user-friendly and accessible to a diverse range of users. A well-designed interface should be intuitive, allowing users to interact with the system efficiently and effectively. Compatibility with screen readers and assistive technologies should be ensured to accommodate users with disabilities. Following best practices in UI/UX design helps enhance engagement and reduce learning curves for new users.

Maintainability

Maintainability refers to the system's ability to be easily updated, modified, and debugged. This requires developing clean, modular, and well-documented code, making it easier for developers to troubleshoot issues and implement updates. By following coding best practices and using standardised frameworks, the system can support future enhancements and long-term maintenance.

Testing and Quality Assurance

Testing ensures that the system functions correctly and meets user expectations. The system relies on external JS/PHP libraries for map display and structured CSV files for asset data, meaning these dependencies must be tested thoroughly. Constraints such as alignment with gov.uk design standards must also be met. Different testing strategies should be implemented, including unit tests (checking individual components), integration tests (verifying different modules work together), and user acceptance testing (ensuring the system meets business needs).

Security & Compliance

To comply with data protection regulations and ensure robust authentication and authorisation, the system must follow industry standards for data privacy and security. Encryption techniques should be applied to protect sensitive data at rest and in transit, reducing the risk of cyber threats or data breaches.

Performance & Scalability

The system must be designed to handle high traffic and large datasets efficiently, preventing slowdowns or crashes. Implementing load balancing, database optimisations, and cloud-based solutions ensures that system performance remains consistent under varying workloads.

User Interface & UX Design

A well-structured User Interface (UI) and User Experience (UX) design is essential for accessibility and ease of use. The system must adhere to gov.uk and Bradford Council branding, ensuring a consistent and professional look. Additionally, compliance with WCAG 2.1 accessibility standards guarantees that users with disabilities can navigate and use the platform effectively.

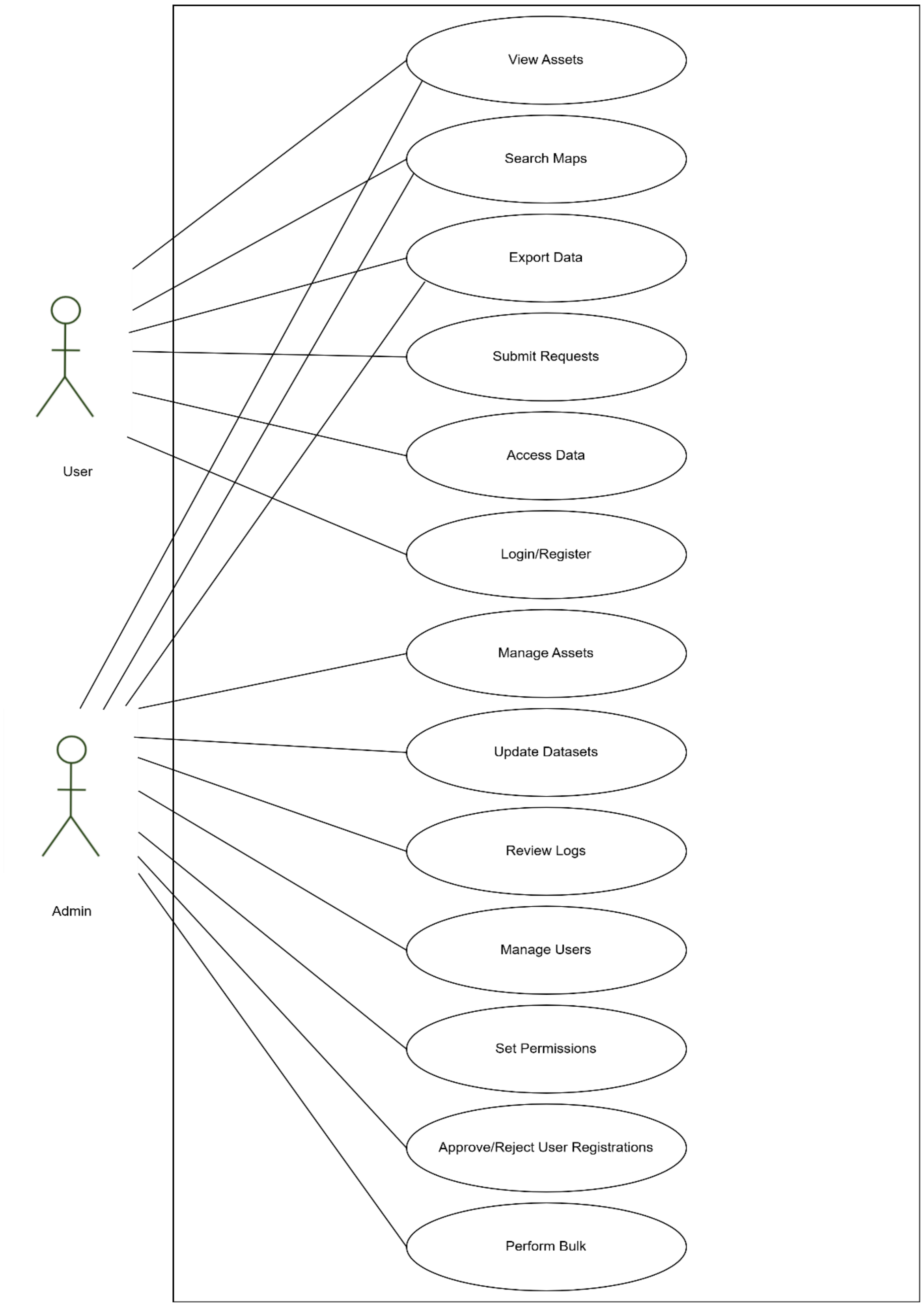
Data Management

Data management ensures that all asset information is securely stored, organised, and backed up. Backup and recovery mechanisms should be in place to prevent data loss due to system failures or cyberattacks. The system should also provide efficient data retrieval and storage techniques to maintain high performance.

Testing & Quality Assurance

Comprehensive testing strategies are required to ensure system reliability and compatibility across different devices and browsers. This includes conducting unit, integration, and user acceptance testing to validate that each feature works as expected. Load testing should assess the system's ability to handle high traffic, while cross-browser testing ensures a consistent experience across different web platforms.

web interface to interact with assets


View Assets
Search Maps
Export Data
Submit Requests
Access Data
Login/Register
Manage Assets
Update Datasets
Review Logs
Manage Users
Set Permissions
Approve/Reject User Registrations
Perform Bulk
User
Admin