Software Requirements Specification (SRS) document

CS673 Team-5 Project
TerrierConnect

Introduction

The Software Requirements Specification (SRS) document aims to present a detailed description of the Terrier Connect project. This document introduces the overview by defining the benefits, scope in detail. It also concentrates on the functions required by end-users and their needs while defining high-level requirements and use cases. The detailed requirements of the project are provided in this document. In addition, it provided High-level interface (GUI or API) designs. The GUI design gives users clearer expectations of our products. The API design gives developers standards and specifications for software design and development.

Benefits

1. **Centralized Information Hub**: TerrierConnect will serve as a social media for Boston University students, staff, and faculty, streamlining communication by providing a single platform for personal posting, event announcements, academic updates, and community discussions.

2. **Improved Community Engagement**: TerrierConnect fosters collaboration and engagement across various university departments and groups by allowing users to interact, comment, and share information easily. It strengthens connections within the BU community.

3. **Event Management**: TerrierConnect can be used for creating, promoting, and managing events, allowing users to RSVP directly within the system, which will streamline event organization and participation for both students and staff.

4. **Location Information Sharing**: TerrierConnect allows for seamless sharing of locations, enabling seamless coordination, and simplified campus navigation. This can benefit on making friends, even posting, academic discussions.

5. **Scalability for Future Expansion**: TerrierConnect will be designed to handle growing numbers of users, posts, and events, with the potential for expansion to accommodate future needs such as alumni networks, external partnerships, or new university initiatives.

Scope

The **TerrierConnect** project aims to develop a web application that serves as a centralized bulletin board for the Boston University community. It will allow students, faculty, and staff to share events, activities, and updates, offering features such as personalized recommendations, an interactive map for event location tracking, and user-generated posts. This platform is designed to enhance campus communication, strengthen social connections, and provide a user-friendly experience for staying informed about university-related activities.

The application will be built using React for the front-end development, Django RESTful API for the back-end, and SQLite for data management. Key features include integration with Google Maps to facilitate real-time event location sharing and campus navigation, along with account management functionalities like registration and login. The development approach will follow an iterative and incremental methodology, adopting agile practices to ensure regular progress, feedback, and enhancement throughout the project lifecycle.

The project scope includes delivering a fully functional web application accessible to all members of the BU community, along with technical documentation covering design, APIs, and user guides. Quality assurance will be a priority, with robust testing processes, including unit, integration, and user acceptance testing. This scope will help ensure that the final product is reliable, user-friendly, and meets the intended objectives of enhancing campus life.

High-level requirements

Functional Requirements

Since our project is built on the backend of Django API and the frontend framework of React. Therefore, we need to have a relatively complete design of the overall architecture of the project before we start to implement the project, so as to avoid encountering additional problems during the development of the project as much as possible.

User Management

User management is an essential function of a social network service, and it is also the first part of our project that we are going to complete in the first phase. The functions of this part mainly include the following:

- User registration
    - For each visitor, they can register a new user in any interface.
    - Since **TerrierConnect** is limited to BU campus user registration, when the user registers, the registration page will verify whether the email address used by the registrant ends with *@bu.edu*.
    - The password used by the registrant must meet the basic specifications.
- User profile page
    - Users can write relevant information about themselves on this page, including: display name, personal profile, and avatar.
    - Users can view their followers and followings on this page.
    - Users can view all their posts on this page, and can enter the detailed post page by clicking on any post.
    - Users can change their passwords on this page.

Home Page

The home page is the entry point after the user completes the login. The user can access other functions of the website on this page.

- This page will display the list of posts sent by other users followed by the currently logged in user.
- The top of the home page will display important information pinned by the website administrator.
- There is a sidebar on the left side of the home page. The sidebar will display the current hot topics (tags). The displayed tags are the tags that have been used most in the past 24 hours, so that users can join the discussion of hot topics.
- Users can click on a specific post on this interface to enter the corresponding detailed post page.
- Users can post new posts on this interface.
- Users can click on their own or other users' avatars to enter the corresponding user's user profile page.

Detailed Post Page

The detailed post page is used to display the specific content and related information of each post.

- Post detail contains the content of each post and the pictures sent with the text content and the sending time of the current post. If the content of the current post has been modified, the latest modification time of the current post will be displayed at the same time.

- The detailed post page will display the sender's name and avatar.
- If the user includes geographic coordinates when sending a post, the detailed post page will display a map view on the page through the Google Maps API.
- If the sender of this post is the currently logged in user, the detailed post page will display an additional edit button to allow the user to modify the content of the post.
- There will be a comment button below the post content, and clicking the button can send a new comment.
- Below the comment button is a list of comments from other users.

New Post Page

The New post page is used to send a new post to the user.

- Each post is required to have at least one title. However, the text content of the post is optional.
- The New post page can choose to add a sent picture.
- The New post page can choose to add geographic location information. If there is geographic location information, a map will be displayed in the post detail after it is published.
- New posts can set permissions: visible to everyone (Public), visible only to friends (Friends Only), visible only to oneself (Private). Friends Only is defined as a state where users are following each other.

Nonfunctional Requirements

The **nonfunctional requirements** for the TerrierConnect project define the key system qualities that ensure the application's performance, security, and usability. These requirements outline the expected standards for how the system should operate beyond its core functionalities.

The system must be designed for high performance, capable of supporting many concurrent users while providing quick response times. Security measures will protect user data and ensure only authorized access, while usability and accessibility will cater to the needs of all Boston University community members. Additionally, the application must maintain reliability, be easy to maintain and update, and be compatible with a range of browsers and devices to ensure a seamless user experience.

Performance

- The system must support at least 1,000 concurrent users without significant performance degradation.
- Response times for user actions must be under 2 seconds for most operations.

Scalability

- The system must be designed to easily scale to accommodate increasing numbers of users and events as the user base grows.
- The database structure must allow for efficient querying and storage of large volumes of user-generated content.

Security

- User data, including credentials, must be securely stored and encrypted.
- Authentication and authorization must ensure that only authorized users have access to restricted features or data.
- The application must be resilient against common web vulnerabilities, including SQL injection, XSS, and CSRF.

Usability

- The user interface must be intuitive, requiring minimal learning for BU students, faculty, and staff.

Reliability and Availability

- The application must be available 24/7, with minimal downtime for maintenance.
- The system must be able to recover from unexpected failures with minimal data loss.

Maintainability

- The codebase must be modular and documented to ensure ease of future maintenance and updates.
- Version control must be utilized for all components to manage changes and track the history of development effectively.

Compatibility

- The web application must be compatible with modern browsers (Chrome, Firefox, Safari, Edge) and responsive across different screen sizes and devices.
- The system must be deployable using Docker for consistent environments in development, testing, and production.

Use cases

User registration page

- The user registers on this page.
- On the login page, The user inputs their BU email address, and enters a valid password to complete the registration. Then the user clicks the Sign Up button.
- The system automatically login the account the user registered and jumps to the home page.

Home page

- The user clicks on START button
- The system jumps to Login page
- The user inputs email and password, and clicks sign in
- The system jumps back to the homepage
- The user clicks on a post button in the sidebar
- The system jumps to the detailed post page.

Detailed post page

- After The user clicks on the title of a post, he/she will enter the detailed post page.
- The user views the specific content of the current post on this page.
- Each post contains the following:
  - Title
  - Content
  - Published time
  - Edit button (if you are the author of this post)
  - Comment button
  - Comments List
- Click the "New Comment" button to create a new comment. After finishing writing the comment, click the "Publish" button to publish it.

New post page

- The user creates a new post on this page.
- A new post contains the following:
  - Title
  - Content
  - Pictures upload button

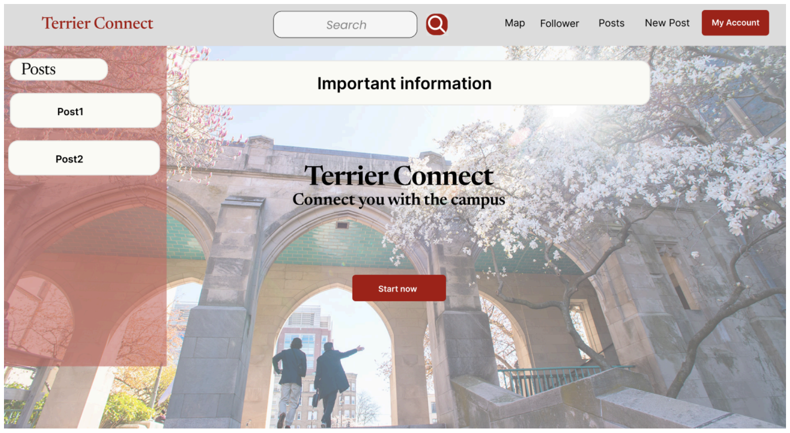
    - Add location button
- After finishing writing the title and content of this post, click the “Publish” button to publish it.

High-level interface (GUI or API) designs

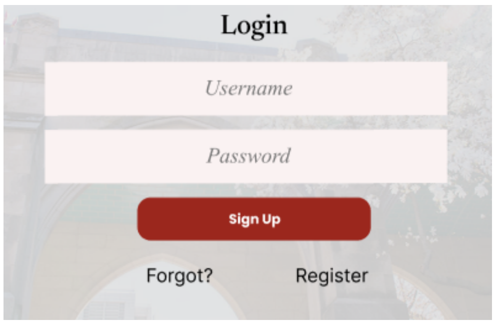
GUI design

Homepage

- Navigation Bar: Include links to My Account, New Post, Posts, Follower, Map
- Main content area: list of events/posts with brief information and START button, and display important information
- Sidebar: display the current hot topics.

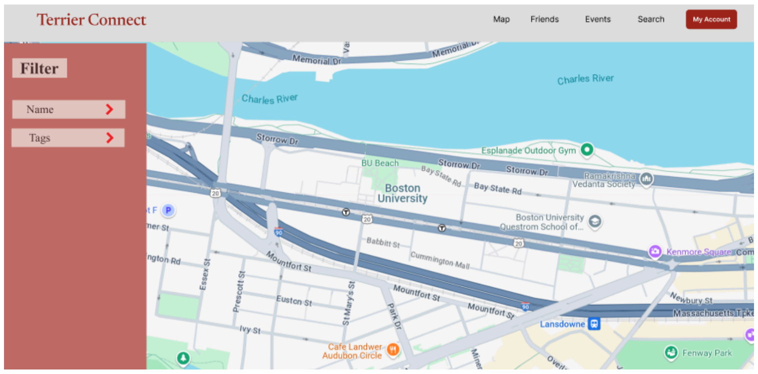


Login



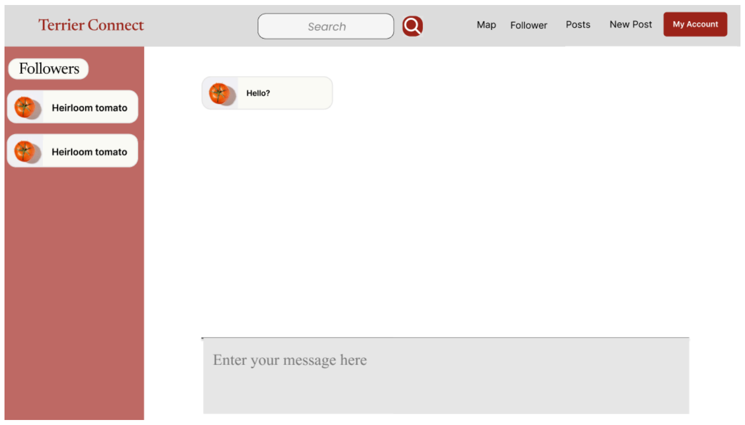
- Username/email
- Password
- Continue button
- Sign in button
- Forgot button

Map



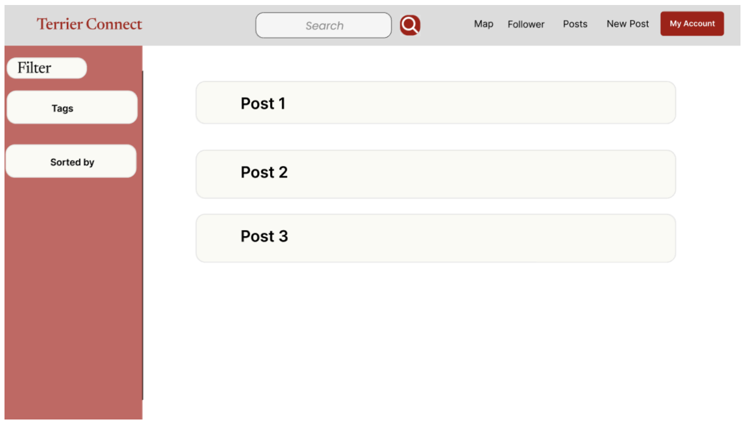
- Filter sidebar
- Map and mark
- Navigation

Follower



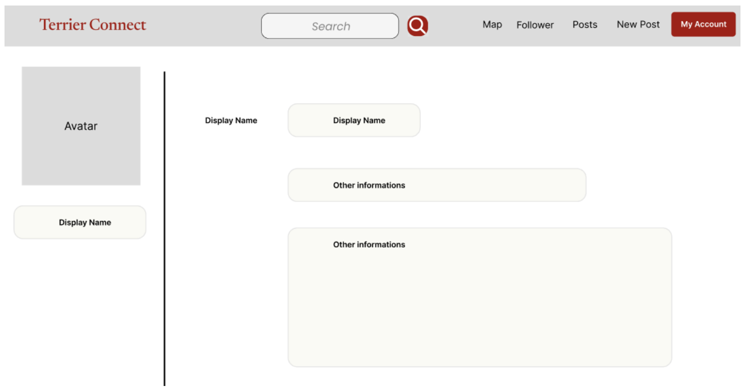
- Followers list(sidebar)
- Navigation
- Chat part

Posts



- Navigation
- Filter (sidebar)
- Posts list

My Account



- Navigation
- Avatar
- Other Information

API design

The overall API design should be **RESTful APIs** that adapt to HTTP verbs and expose key resources such as **posts, users, events, groups and maps**. The APIs will satisfy all high-level functional requirements from the frontend of the website.

- **Data Format:** All APIs will return data in **JSON format** to ensure compatibility with various client systems and to keep the data easy to work with.
- **One-to-One Mapping:** Each key requirement will have its own specific API endpoint, keeping the system organized and making it easy to understand and maintain.
- **Pagination:** For endpoints that return large sets of data (like lists of posts, users, or events), **pagination** will be used. This helps avoid performance issues by breaking down large amounts of data into smaller, manageable chunks, which also reduces the load on the server.
- **Error Handling:** We will standardize error messages using HTTP status codes (e.g., 400 for Bad Request, 401 for Unauthorized, 404 for Not Found, 500 for Internal Server Error). Clear and concise error messages in the response body will help developers quickly understand and resolve issues.
- **Logging and Monitoring:** The system will include logging and monitoring for all API calls. This will help us track how the system is being used, identify any issues, and optimize its performance over time.
- **HTTPS**: All API calls will be made over **HTTPS** to encrypt data and ensure secure communication between client and the server.

This API design will ensure that the TerrierConnect system is secure, scalable, and easy to maintain. By considering important aspects like security, error-handling etc. the API will provide a reliable and flexible foundation for future development and integration with other systems.