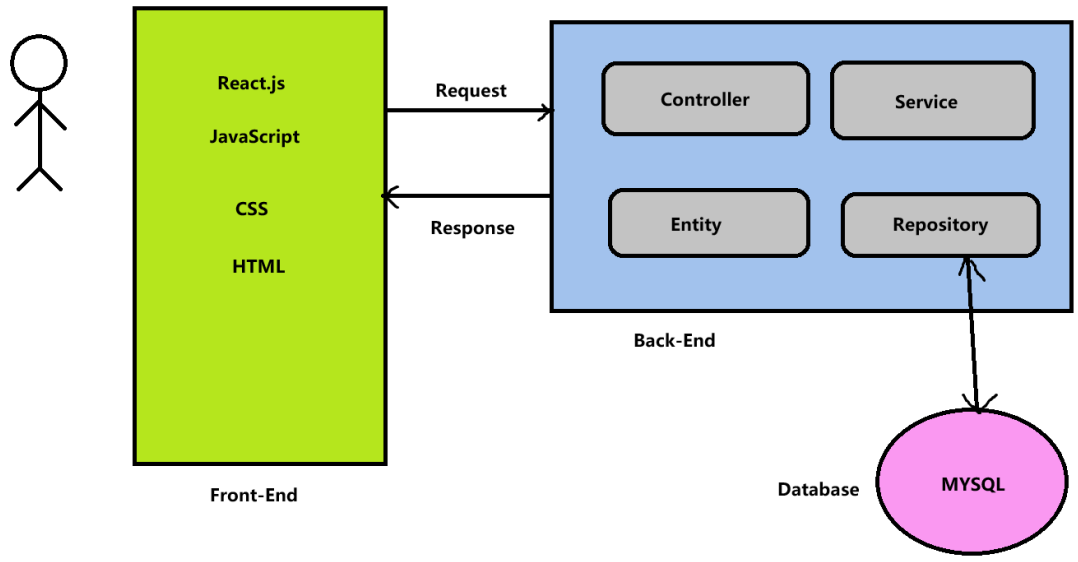
Software Design Document (SDD)

1. Introduction:

- **Purpose and functionality:** We worked to develop an event listing platform to be used by students who are interested in catering, event hosting or even attending parties. We propose the development of an Event Listing Platform for students that caters to event hosts and attendees. This platform will serve as a centralized hub where they can list their events, manage attendees, and promote their gatherings. Later, the attendees can rate the host based on the atmosphere and environment presented to them, the management of the event, etc. On the other hand, attendees can easily discover and register for events that interest them, as long as they meet the requirements such as age, dress code, or the minimum rating of the attendee. This project aims to simplify event management and attendance, fostering a vibrant event community.
- **Scope:** We are focusing on students who are interested in both generating some revenue as hosts and attending social gatherings. Students who want a safe and controlled way of attending events

2. Architectural design of the software:

Our web application's architectural design leverages modern web design technologies, including Spring Boot, MyBatis-Plus, React.js, CSS, HTML, and JavaScript, to create a feature-rich and user-friendly event management platform to help every user host or participate in events more effectively.



Backend Architecture:

We use Spring Boot as the foundation for our server-side architecture. It provides a microservices framework for easy management of the APIs and microservices required by the website. In the project, MyBatis-Plus acts as the Object-Relational Mapping (ORM) tool, facilitating seamless interaction with the MySQL database:

- The controller layer typically receives requests from the client, processes request parameters, and then calls methods in the service layer to execute business logic. It does not interact directly with MyBatis-Plus but calls methods in the service layer to pass the request for processing.
- The service layer contains the business logic of the web application. In the service layer, database operations are performed by invoking data access methods provided by MyBatis-Plus. The service layer is where direct interaction with the data access layer (DAO) takes place. We define various service layer methods that encapsulate MyBatis-Plus's query, update, insert, and delete operations to perform data-related tasks.
- Finally, we define data access objects (Mappers) in the form of annotations and XML provided by MyBatis-Plus. In this layer, we define the mapping relationships of database tables, write SQL query statements, and call APIs provided by MyBatis-Plus to execute database operations.

Frontend Architecture:

On the front end, we use React.js as the framework to build dynamic and responsive user interfaces. HTML and CSS are used for structuring content and styling, while JavaScript enhances frontend and backend interactions. The backend and front end are connected via RESTful APIs. We provide a unique URL for each backend API and create dedicated API invocation methods for them. When users perform actions on the web page, the web page will automatically invoke the corresponding API methods for data processing and return the expected values to the user.

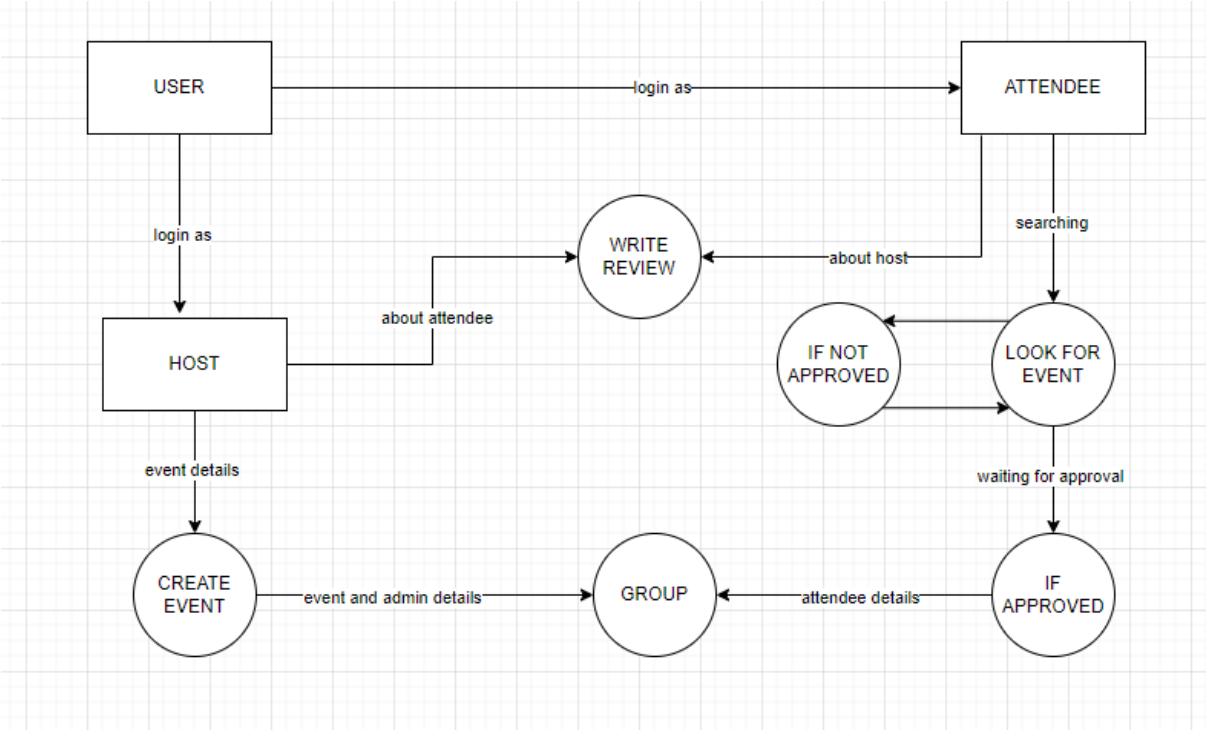
3. High-level (external-facing) design of the system:

3.1. User Interfaces (UI):

Users can interact with our design. During our meetings and when we start designing our website, we considered user experience (UX) principles for usability and accessibility such as the ones in the below image.

Reference: ***Design Accessibility Summit.*** **(2018). Retrieved 2023, from** https://th.bing.com/th/id/R.1e44bddf26633fa0ad09193a3e391952?rik=7yyiRWErhHyUhQ&riu=http%3a%2f%2fpinsourcing.com%2fwp-content%2fuploads%2f2018%2f02%2fAccessibility-infographic-1024x695.png&ehk=ZavvxlXxBd5VeVP8NG%2fxzlNLNd4sZZJXhFI7ZYZvNH8%3d&risl=&pid=ImgRaw&r=0.

3.2. Data Flow and Integration:



3.3. Statelessness: Each request from a user to the server must contain all the information needed to understand and process the request. The server should not store any user context between requests.

3.4. Data Format: In our project, we are using JavaScript (REACT) as the data format for request and response payloads.

3.5. Error Handling: We are planning to use a standard way of handling errors, such as 404 (Not Found) and 500 (Internal Server Error) to indicate the outcome of a request.

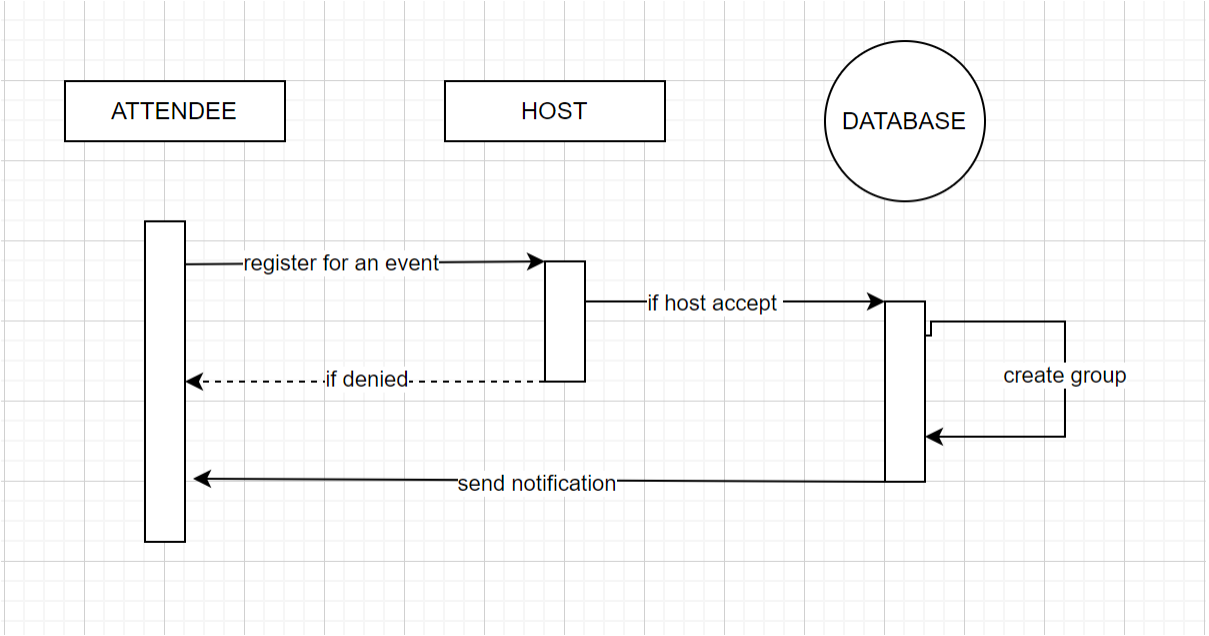
3.6. Pagination and Filtering: For some resources that may return many items such as the events page, we are providing a filtering option to help users retrieve only the data they need efficiently.

3.7. Versión Control: We Maintain version control for the API design of this project. We are currently using GitHub to manage any changes.

3.8. Feedback and User Support: We will develop a feedback and user support channel to collect user opinions and provide support.

3.9. Resource-Based Architecture: In a RESTful API, everything is treated as a resource. Resources are represented by URIs, and each resource should have a unique URI. In our case, (friends hub) have resources for events, users, posts, and more each with its own URI.

3.10. Sequence diagram:



3.11. Class Diagram: Please check the **.html file** present with our submission.

First demo examples:

Landing page: https://2d5nqx.csb.app/

Sign Up page: https://ttwfhq.csb.app/

4. Detailed (internal-facing) design of the system (e.g., classes, methods):

Following all the functional requirements from the requirement document and the attributes in the database, we have defined 9 distinct classes and multiple methods for the backend part.

Classes:

Attendee: The class that will be used to manage the feature for Attendees and event registration action.
Event: The class will be used to do event management such as adding and canceling events.
Event_Group: The class which will be used to get group information for each event. Both attendee and host will be invited into the group chat and Event_Group will save the group id to help users join it efficiently.
Event_registration: The class will be used to manage all registration from the attendees. There are three different states (accept, decline, and pending) that will help attendees to know if they can attend the event they registered or not.
Group_member: The class will be used to help the host manage all member information in each group.
Host: The class which will be used to manage the feature for hosts.
Rating: At the end of the event, both the host and attendee can rate the event and write a comment to talk about their experience. This class will manage all information and get them as part of the event's detailed information.
User: This is the core class representing the user information. It will be used to hold the information for both host and attendee users such as login information, email address, and other personal information.
Zip Code: While the attendee searches for an event to attend, he/she can filter out using zip code and look for events in any particular area.

We have added methods for adding, deleting, updating, and querying tables in the corresponding database to manage information in the database. We accomplish all database-related operations by adding SQL statements in XML files. When users perform corresponding actions on the front end, the system will automatically call the matching API to make changes to the database. In addition to the standard data management functions, we will also be adding more functional APIs to enhance the user experience of the website.

User login method: Since we encrypt user login passwords when storing information, when we attempt to match user input values with the passwords stored in the database, we use Spring Security for authentication. When the user's input values match perfectly with the database, the backend will return a token related to the user. This token will play a crucial role in accessing user information in the future.

We will also attempt to implement the Google Maps API. When users inquire about detailed event information, they can use the map's API to learn about the surrounding traffic

conditions, including the location of parking lots. This will significantly enhance the user experience of the website.

In the future, we will continue to explore more features to improve our website's design and development, aiming to gain more user recognition and usage of our product.