Sri Lanka Institute of Information Technology

Programming Applications and Frameworks (IT3030)

Continuous Assignment – 2023, Semester 1

Initial Document

GROUP ID: **GRP_046**

- Nishshanka N.A.G.A.A **IT21032974**
- Jayasinghe K.A.K.N **IT21032806**
- Nishshanka N.A.P.K.R **IT21033032**
- Abeykoon R.M.S.P **IT21021602**

❖ brief project description

The social media platform for foodies is a web-based platform where food and beverage reviewers can share their experiences and opinions on various restaurants, cafes, bars, and other food-related businesses. The platform aims to connect foodies from around the world, allowing them to discover new dining experiences, share recommendations, and connect with like-minded individuals.

The platform will allow users to create profiles, share photos and reviews, follow other users, and participate in discussions on various food-related topics. It will also include a search feature that allows users to search for restaurants and food-related businesses based on location, cuisine, price range, and other criteria.

The platform will include both a REST API and a client web application, with the API handling data storage and retrieval and the client web application providing a user-friendly interface for users to interact with the platform. The project will prioritize scalability, reliability, security, and performance, while also focusing on usability and accessibility to ensure a seamless user experience.

❖ Functional requirements

- Functional requirements for the REST API

1. **Authentication and Authorization**: Users must be able to create an account, log in, and log out of the platform. The API should support authentication and authorization for secure access to user-specific data.
2. **User Management**: Users should be able to update their profile information, including their name, email, password, and profile picture.
3. **Post Management**: Users should be able to create, read, update, and delete posts. Each post should include a title, description, location, date, and pictures. Users should be able to tag the cuisine type, restaurant name, and location of the restaurant in each post.
4. **Commenting**: Users should be able to comment on posts made by other users.

5. **Rating System**: Users should be able to rate restaurants based on various criteria such as food quality, service, ambience, etc.
6. **Search Functionality**: Users should be able to search for posts and restaurants based on various criteria such as cuisine type, location, and ratings.
7. **Social Features**: Users should be able to follow other users, like and share posts made by other users, and view a newsfeed that displays the latest posts made by users they follow.

- Functional requirements for the client web application

1. **User Authentication and Authorization**: Users must be able to create an account, log in, and log out of the platform. The client web application should include a user interface for secure user authentication and authorization.
2. **Profile Management**: Users should be able to update their profile information, including their name, email, password, and profile picture. The client web application should include a user interface for managing user profiles.
3. **Post Management**: Users should be able to create, read, update, and delete posts. The client web application should include a user interface for creating and editing posts.
4. **Commenting**: Users should be able to comment on posts made by other users. The client web application should include a user interface for adding comments to posts.
5. **Rating System**: Users should be able to rate restaurants based on various criteria such as food quality, service, ambience, etc. The client web application should include a user interface for rating restaurants.
6. **Search Functionality**: Users should be able to search for posts and restaurants based on various criteria such as cuisine type, location, and ratings. The client web application should include a user interface for searching posts and restaurants.
7. **Social Features**: Users should be able to follow other users, like and share posts made by other users, and view a newsfeed that displays the latest posts made by users they follow. The client web application should include a user interface for social features.

❖ Non-Functional requirements

- Non-Functional requirements for the REST API

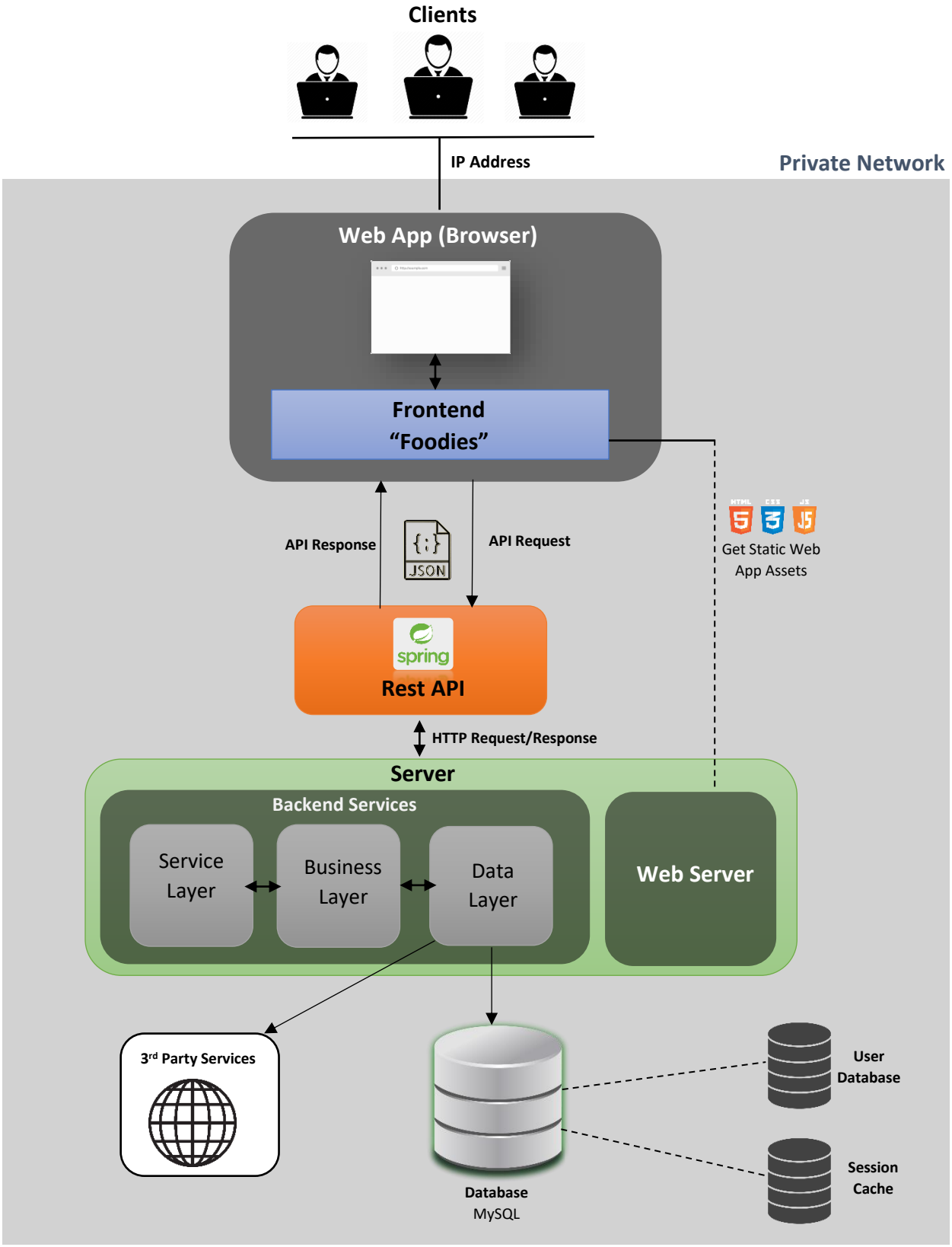
1. **Scalability**: The REST API should be scalable to handle many users and requests. It should be able to handle peak traffic times without slowing down or crashing.
2. **Reliability**: The REST API should be highly available and reliable. It should be able to handle unexpected failures and errors and provide seamless user experience.

3. **Security**: The REST API should be secure and protect user data. It should implement proper authentication and authorization mechanisms to prevent unauthorized access and data breaches.
4. **Performance**: The REST API should be highly performant and respond quickly to user requests. It should have low latency and be optimized for efficient data retrieval and processing.
5. **Documentation**: The REST API should be well-documented and easy to understand. It should include clear documentation for endpoints, request/response formats, and error handling.
6. **API Versioning**: The REST API should support versioning to allow for changes and updates without breaking existing client applications.

- Non-Functional requirements for the client web application

1. **Usability**: The client web application should be user-friendly and easy to use. It should have a simple and intuitive interface that allows users to easily navigate and use the platform.
2. **Accessibility**: The client web application should be accessible to users with disabilities. It should comply with accessibility standards such as WCAG 2.1 Level AA.
3. **Compatibility**: The client web application should be compatible with popular web browsers and devices. It should work seamlessly on desktop and mobile devices.
4. **Performance**: The client web application should be highly performant and respond quickly to user interactions. It should have low latency and be optimized for efficient data retrieval and processing.
5. **Security**: The client web application should be secure and protect user data. It should implement proper authentication and authorization mechanisms to prevent unauthorized access and data breaches.
6. **Internationalization**: The client web application should support multiple languages and cultures to cater to a global audience.
7. **Offline Support**: The client web application should support offline access and provide a seamless user experience even when the user is offline.

❖ Overall architecture diagram



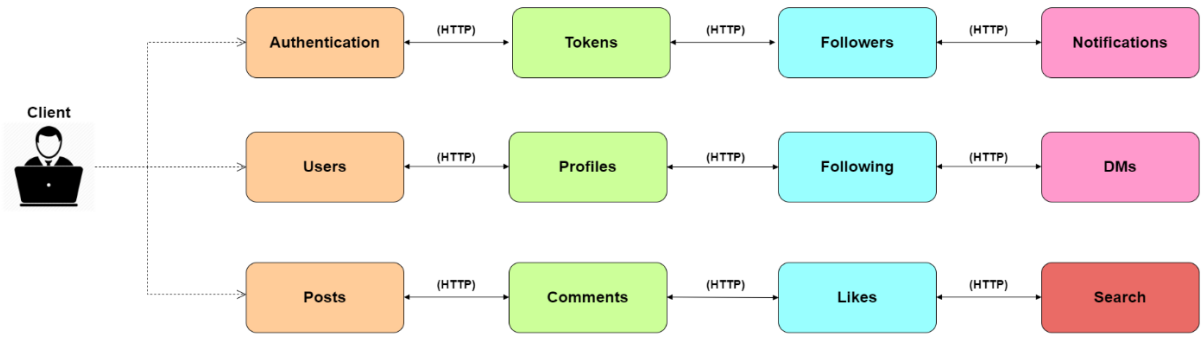
- **Clients**: End-users using the web app to interact with the social media platform.
- **Web App**: The front-end of the social media platform that displays information and allows user interactions.
- **REST API**: The interface that allows the client to communicate with the server.
- **Server**: The back end of the social media platform that manages data, business logic, and authentication.
- **Databases**: The storage systems that store and manage data used by the server.
- **User Database**: A database that stores user account information, such as usernames, passwords, and email addresses.
- **Session Cache**: A cache that stores session tokens to manage user authentication state.

The client sends requests to the server through the REST API, including authentication requests. The server processes these requests and returns responses to the client via the REST API. The server also interacts with the databases to store and retrieve data as needed.

When a user logs in or registers for an account, their credentials are checked against the User Database to verify their identity. If the user's credentials are valid, the server generates a session token and stores it in the Session Cache. The session token is sent back to the client, which includes it in future requests to authenticate the user. The server verifies the session token with the Session Cache to determine whether the user is authenticated or not.

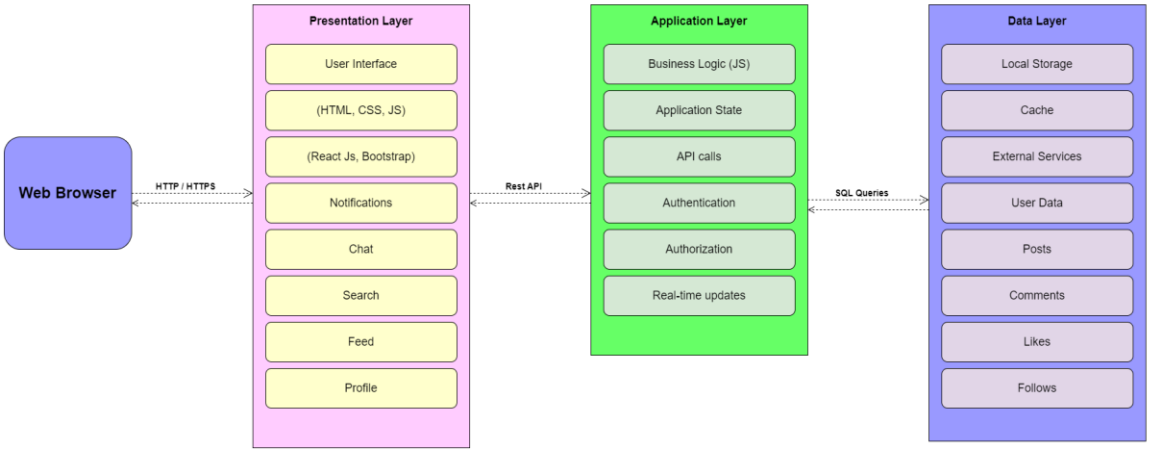
The web app is responsible for rendering the user interface for the client, which includes displaying user profiles, posts, comments, likes, and other social media-related features. The REST API provides a standardized interface for the client to interact with the server, enabling the development of various clients (mobile apps, desktop apps, etc.) that can all communicate with the server using the same API.

❖ Detailed architecture diagram(s) for the REST API



Each component exposes a set of REST API endpoints that clients can use to interact with the system. These endpoints are implemented using HTTP methods such as **GET, POST, PUT,** and **DELETE**.

❖ Detailed architecture diagram(s) for the client web application



In a "**Foodies**" social media platform, the client-side architecture is typically divided into three layers: the data layer, the application layer, and the presentation layer.

The **data layer** is responsible for handling the data that is received from the server. This layer handles the data transfer between the client and server and manages the storage of data locally. It includes technologies like HTTP requests, web sockets, and local storage. The **application layer** is responsible for managing the business logic of the social media platform. This layer determines how the data should be displayed, processed, and stored. The **presentation layer** is responsible for rendering the user interface (UI) of the social media platform. This layer takes the data and business logic from the application layer and displays it to the user. Overall, these three layers work together to provide a seamless user experience on social media platforms, with the data layer handling data transfer, the application layer handling business logic, and the presentation layer handling UI rendering.

❖ References:

1. Stack Overflow [https://stackoverflow.com/questions/34367031/functional-and-non-functional-requirements-relating-to-web-api]
2. ResearchGate[https://www.researchgate.net/figure/Reference-Architecture-of-an-Online-Social-Network-Platform_fig1_227226547]