**SYSTEM DESIGN DOCUMENT**

1. **Introduction**

We used “Single Threaded Event Loop” as an architecture in our project. Our architecture is broken into 3 parts. These are Client, Server and Database. Client part is interaction of user with frontside of the project. The main function of Server part is to take request from user and to process logical steps. It treats like a bridge between front-end and database. Lastly, Database part's responsible is to store data which comes with along user requests. Data are manipulating with user request in database.

Expectation from our design goals is briefly descripted below:

* The website should be able to appeal to everyone in terms of ease of use.
* The website shows the posts and comments within the page as quickly as possible, and the search function brings results quickly with the given information.
* In case of any error that may appear on the website, the user is directed to a page corresponding to the error.
  1. **Purpose of the System**

The main purpose of our website is, creating a social media platform for the people who are looking for website with gastronomy and culinary art concept. Users generally share their thoughts around the concept of gastronomy and interact with other users. The using of our website is open to everyone, registering and using the website is completely free.

* 1. **Design Goals**
* **Performance:**

User requests such as sharing, commenting or liking are made within 2 seconds at most after the request.

* **Dependability:**

Invalid inputs from the user are handled with error page which contain possible error reasons.

The server does not crash in case of possible errors on the website.

Users' data are securely protected in the database against possible errors that may be encountered.

* **Cost**

The cost of developing initial system is just purchased HTML template.

* **Maintenance**

Through the modular structure of the project, new functionalities can be added and modified easily.

The simple and responsive design of the website easily allows display from any device with internet access.

The programming language and framework we use allows any developer to easily understand the general structure of the system.

* **End User**

Both the redirects made between the pages and the requests that the user can make are available on each page in a simple and plain form.

If an action has been taken by the admin on the user's data such as account verification, deleting post or comment on the website, the user will be informed about this.

* 1. **Definitions, Acronyms, and Abbreviations**

**User Request:** The name given to the requests sent to the system by the user such as adding, removing or editing.

**User:** A person that can surf between pages on the website, share post, comment or like the posts.

**Admin:** A person that can manage the website with specific functions.

* 1. **References**

Especially, we referenced the general structure of Facebook (facebook.com) and LinkedIn (linkedin.com).

1. **Current Software Architecture**

In the project, we used Single Thread Event Loop Model as an architecture. General structure of the architecture is that. All requests from the client side are handled with a single thread. This thread is named as 'Event Queue'. The web server has an internal component also known as 'Event Loop'. The function of this loop is that receiving user requests and processing these requests. Event Loop checks whether requests from the client side in Event Queue. If there are any request, it takes any request from Event Queue. If not, Event Loop continuously waits for a user request. In case of user request, Event Loop selects an available threat from Thread Pool. Then, Event Loop gives incoming user request to selected available thread and Event Loop assigns the thread with this request. Then, thread takes this request and process this. After that, it executes to block IO operations if necessary. It prepares a user response to return client side. It sends to Event Loop. Event Loop returns the responses to client side, respectively.

1. **Proposed Software Architecture**
   1. **Overview**

Present a bird’s-eye view of the software architecture and briefly describes the assignment of functionality to each subsystem.

* 1. **System Decomposition**

Describe the decomposition into subsystems and the responsibilities of each. This is the main product of system design.

* 1. **Hardware/Software Mapping**

Describe how subsystems are assigned to hardware and off-the-shelf components. It also lists the issues introduced by multiple nodes and software reuse.

* 1. **Persistent Data Management**

Describe the persistent data stored by the system and the data management infrastructure required for it. This section typically includes the description of data schemes, the selection of a database, and the description of the encapsulation of the database.

* 1. **Access Control and Security**

Describe the user model of the system in terms of an access matrix. This section also describes security issues, such as the selection of an authentication mechanism, the use of encryption, and the management of keys.

* 1. **Global Software Control**

Describe how the global software control is implemented. In particular, this section should describe how requests are initiated and how subsystems synchronize. This section should list and address synchronization and concurrency issues.

* 1. **Boundary Conditions**

Describe the start-up, shutdown, and error behavior of the system. (If new use cases are discovered for system administration, these should be included in the requirements analysis document, not in this section.)

1. **Subsystem Services**

Describe the services provided by each subsystem. Although this section is usually empty or incomplete in the first versions of the SDD, this section serves as a reference for teams for the boundaries between their subsystems. The interface of each subsystem is derived from this section and detailed in the Object Design Document.

1. **References**

The following is an example of listing a book in this section. Check the text to see how it is cross referenced (The whole document is based on [1]).

1. Bruegge B. & Dutoit A.H.. (2010). *Object-Oriented Software Engineering Using UML, Patterns, and Java*, Prentice Hall, 3rd ed.