**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).

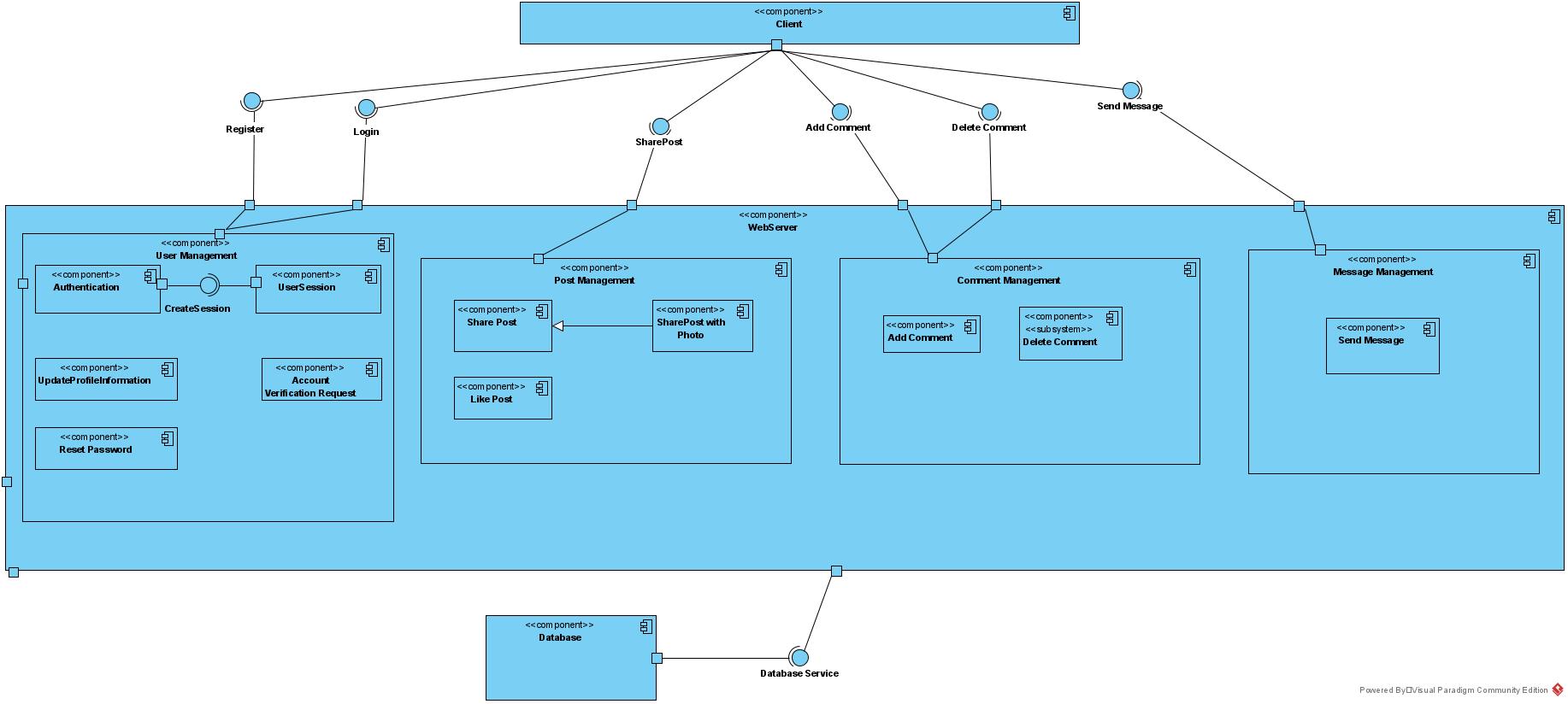
1. **Current Software Architecture**

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.

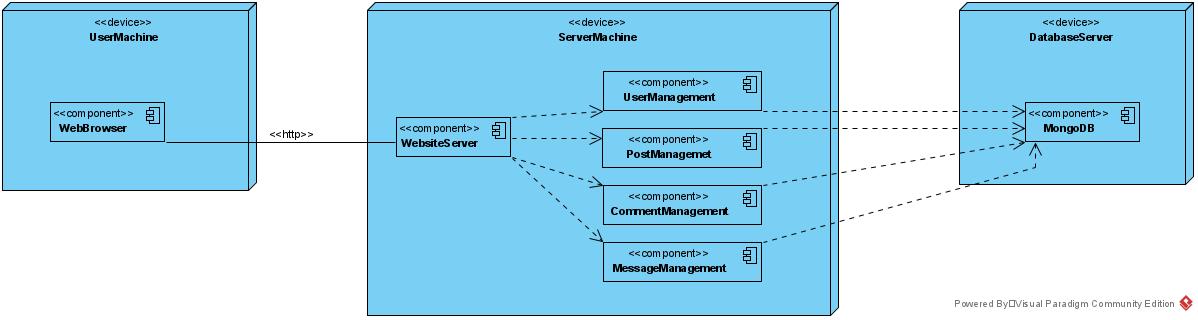
1. **Proposed Software Architecture**
   1. **Overview**

In our project, we used 'Single Thread Event Loop Model' Architecture. The project is essentially broken into three layers. These are Client, Server and Database. In Client layer, there is interface that user can trigger functions of the website. Second layer is Server. In this layer, server has essentially four different components such that UserManagement, PostManagement, CommentManagement and MessageManagement. First component UserManagement is responsible for authentication, updating of user profile, verification request for own account and resetting own password. Second component PostManagement is responsible for sharing post, sharing post with photo and liking post. Then, CommentManagement is responsible for adding and deleting comment. Lastly, MessageManagement is responsible for chatting. The last layer is Database. It stores incoming data from Server.

* 1. **System Decomposition**

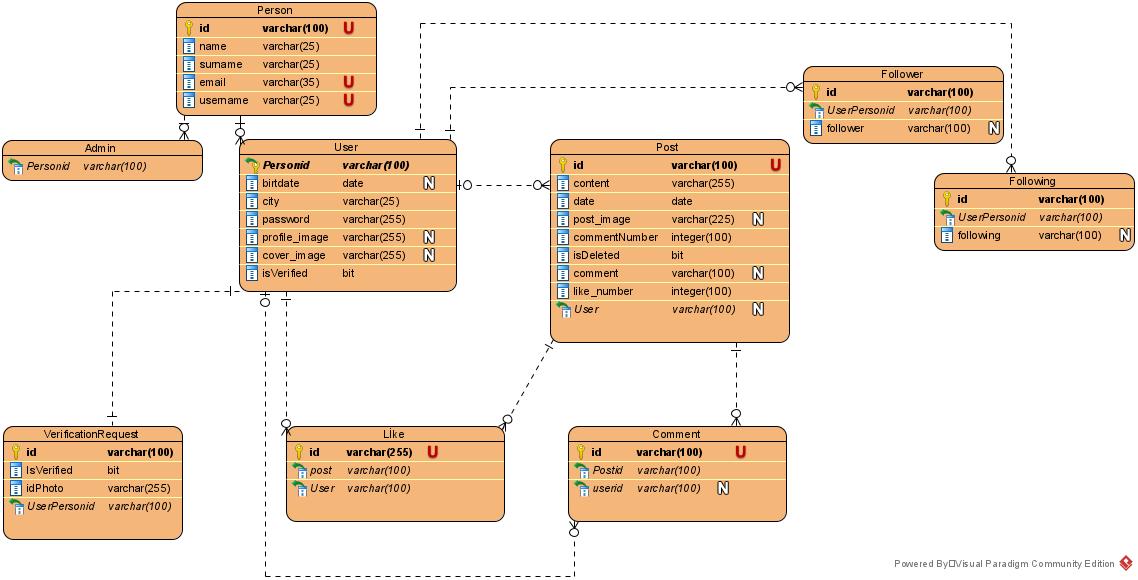
****

* 1. **Hardware/Software Mapping**



The first device in the figure is the User Machine that includes a Web Browser that users interact. Second device is the Server Machine that includes the web server of our website and UserManagement, PostManagement, CommentManagement, MessageManagement components. The third device is the Database Server that includes the database server of our website. In our project, we used the node.js as framework and for the storage, we used a MongoDB database server.

* 1. **Persistent Data Management**

****

Our website offers users the opportunity to share their thoughts with functions such as posting and commenting. Information about this situation is stored in the system. The objects in the end-to-end reconfigurable(E2R) diagram that should be permanent in the system are Person, Admin, User, Post, Like, Comment, Following, Follower and VerificationRequest. We choose MongoDB to store this data.

User table is created to store information of general users who use the website in the system. Admin table is created to store information of administrator in the system.

VerificationRequest table stores which user made the account verification request, id card photo of the user and the information of if it is verified. Post is a table containing which user created the post, when it was created, what content was written, how many comments and likes the post received, and the comments the post received. Like table stores information of which user liked which post. Comment table stores information of which user wrote a comment on which post. Follower table stores who a user's followers are. Following table stores who the user is following.

* 1. **Access Control and Security**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| OBJECTS  ACTORS | Post | Comment | Request | Account | Message |
| User | addPost()  deletePost()  likePost() | addComment()  deleteComment() | verificationRequest() | updateProfileInformation()  resetPassword() | sendMessage() |
| Admin | deletePost() | deleteComment() | accountVerification() |  |  |
| Visitor |  |  |  | signIn()  signUp() |  |

* 1. **Global Software Control**

In our project, we use NodeJS framework to build website by using JavaScript programming language. Also, we use ExpressJS web framework which based on NodeJS. Additionally, we use MongoDB for database, and we use MongoDB-Compass to manipulate database.

* 1. **Boundary Conditions**

**Start-up:** The system administrator runs the server on NodeJS. Then, with the records in the server database and the URL it has, it can be accessed, and its functions become applicable.

**Shutdown:** In case the server is running, the server administrator that providing direct access to the server can shut down the server without any data loss.

**Error Behavior:** If an error is occurred on the website or user try to access an invalid URL, user is directed to a page containing a 404-status code and listing the possible causes of the errors.

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.