**What Do U Want?**

**Software Design Document**

John Melloni, Wyatt Doiron-LaRue, Kachi Onyenokwe, Ayodeji Marquis

**1. Introduction**

The purpose of this document is to layout the architectural structure that we intend to use as we embark on creating the What Do U Want? Application. This includes portraying various design considerations, class diagrams, state diagrams, deployment diagrams, data base knowledge, and possible libraries that will be utilized. Ultimately this document is meant to serve as a guideline for our eventual implementation.

**2. Design Considerations**

**2.1 Assumptions**

The software will require an internet connection to send data between users, but other than that, we have no real assumptions. We have backed away from utilizing cellular devices, at least as a part of this project. Further down the road, who knows?

**2.2 Constraints**

The system is designed for web-only access. The primary coding languages used were: PHP, JavaScript, HTML, and CSS. SQL was used to access the MySQL database, and JQuery and AJAX were used to complement our JavaScript.

**2.3 System Environment**

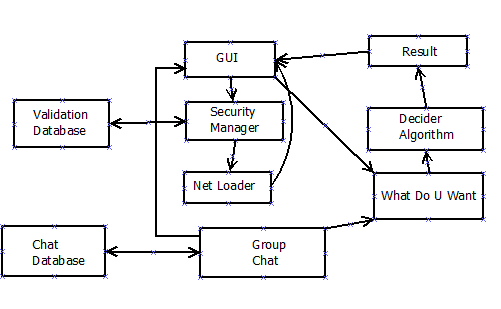
The required hardware would include some sort of way to connect to the internet. The system would require minimum memory because it will be web based and so a user wouldn’t need to download an app or any additional software. The only software required would be a web browser. Although realistically accessible on most web browsers, we feel the design looks best on Chrome.

**3. Architectural Design**

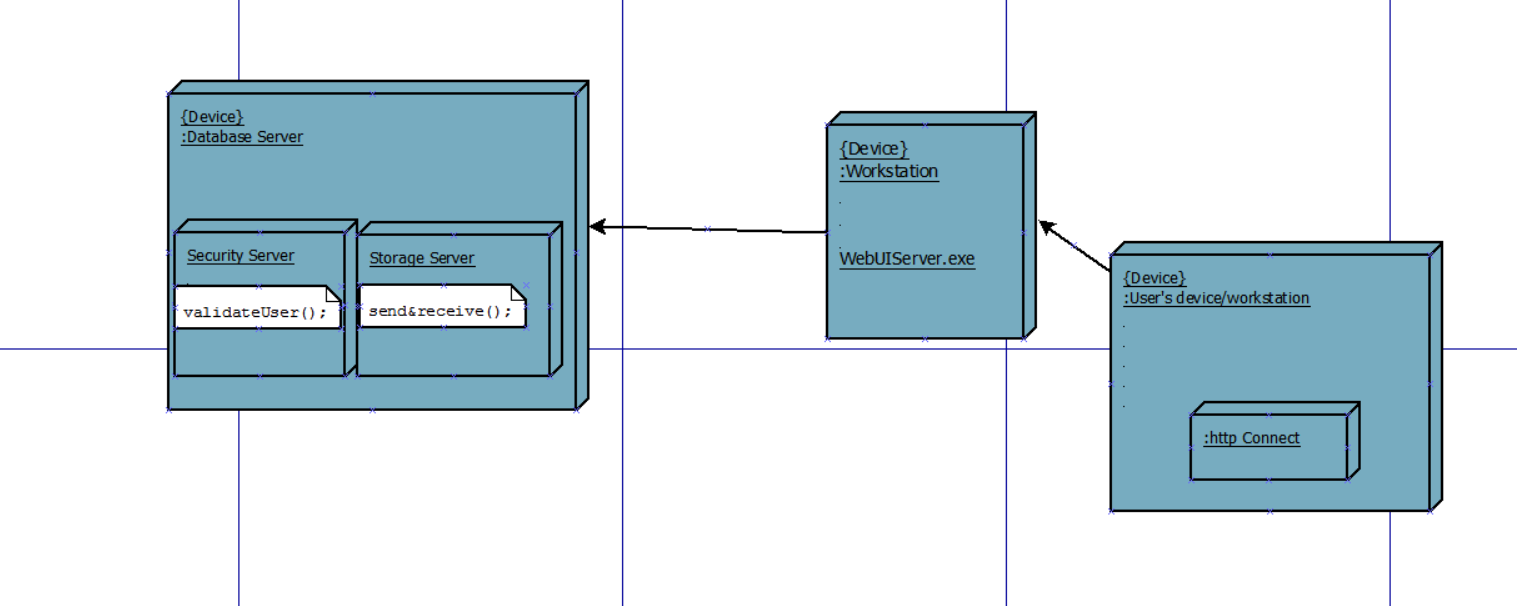
**3.1 Overview**

|  |  |  |
| --- | --- | --- |
| **Sl no.** | **Class** | **Main Responsibility** |
| 1 | GUI | Manages Graphical User Interface |
| 2 | SecurityManager | Manages user authorization |
| 3 | NetLoader | Transmits data between users and system via protocol requests |
| 4 | SecurityDatabase | Hold info regarding security of users |
| 5 | ChatDatabase | Hold info regarding chatrooms and their users |
| 6 | Groupchat | Allows users to input messages and displays messages sent between users |
| 7 | What Do U Want | Keeps track of users’ votes and updates GUI |
| 8 | DeciderAlgorithm | Makes the group decision based on the users’ input |
| 9 | Result | displays the result on GUI |

**3.2 High Level Design**

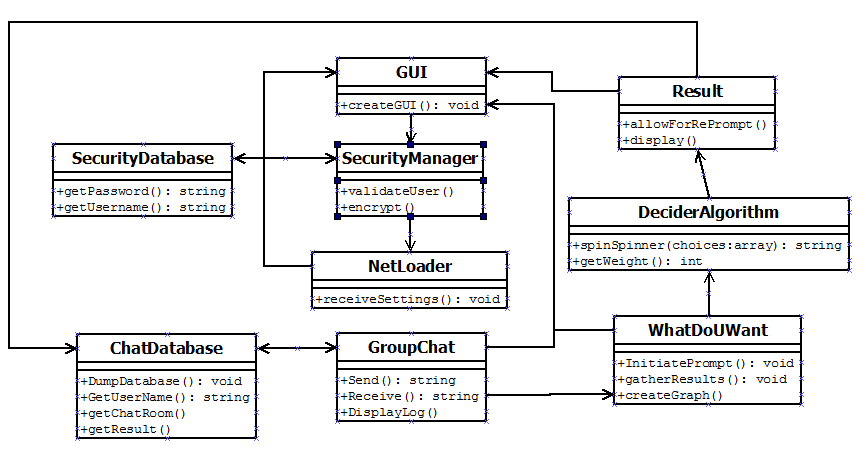


**3.3 Deployment Diagram**

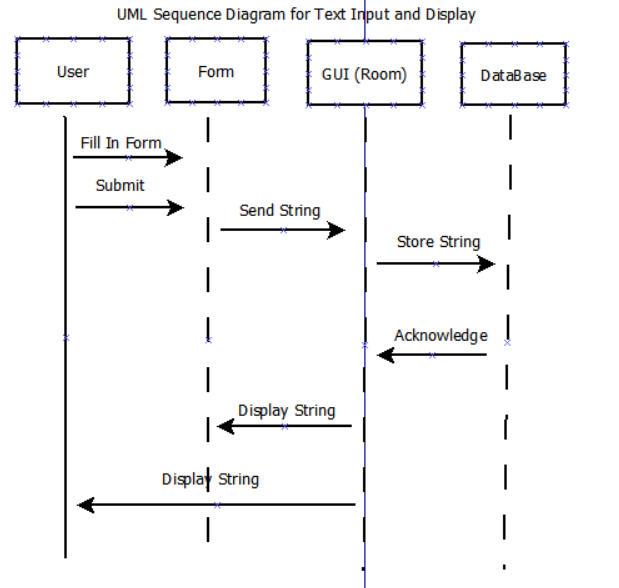


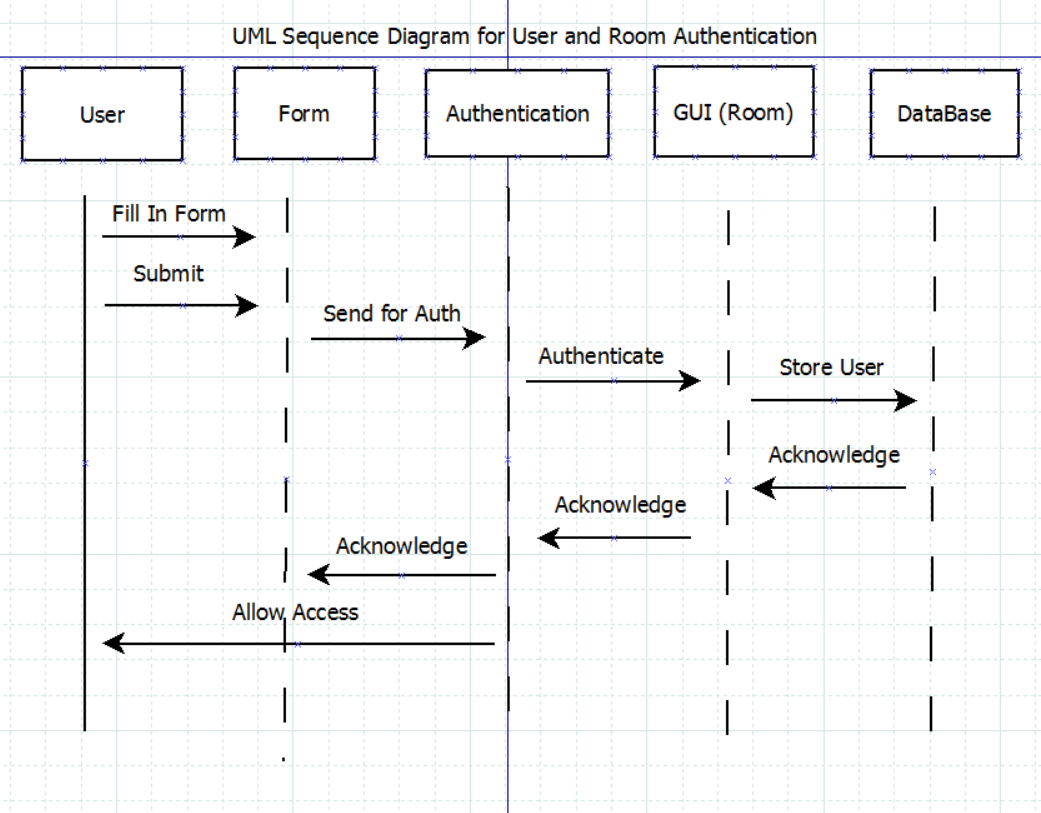
**4. Low Level Design**

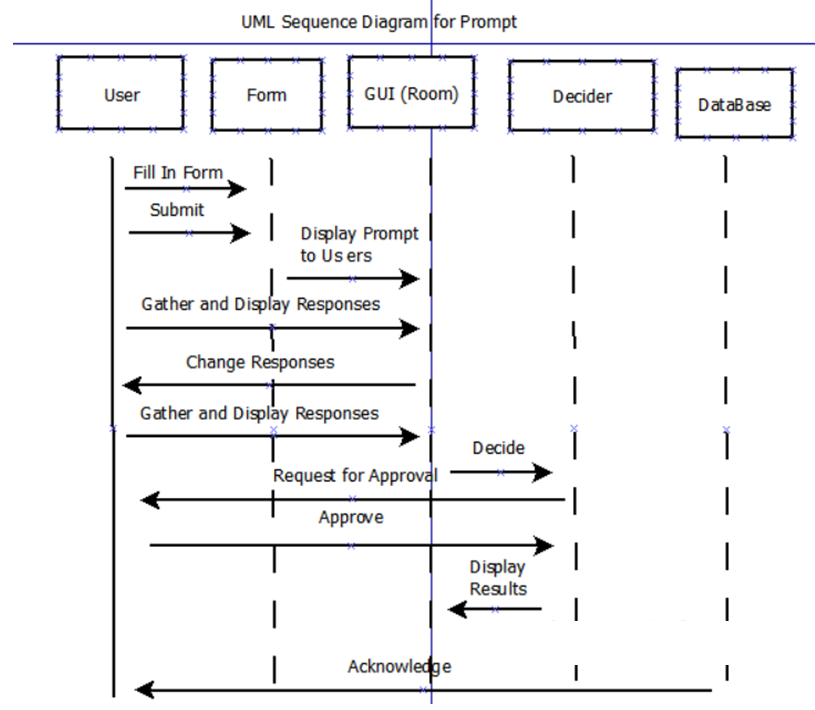
**4.1 Class Diagram**



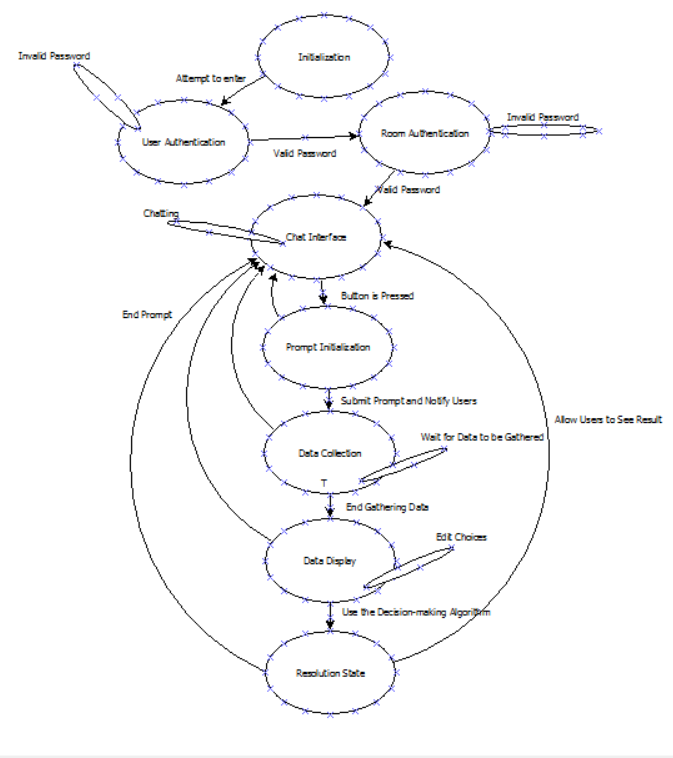
**4.2 Sequence Diagrams**







**4.3 State Diagram**



**5. User Interface Design**

Since our application is going to be deployed on web browsers, we have decided to design the Web UI using HTML, JavaScript and CSS.

The reasons for these languages are as follows:

* It’s one of the easiest languages to start with for web design. It has a syntax that is closest to the English language. It’s mainly object-oriented, that makes it easier to move data types around using the created objects.
* JavaScript uses client side execution. Meaning that the code is run on User’s computer microprocessor, hereby reducing the strain on web servers and increasing bandwidth efficiency. This whole process of client side execution makes javascript is relatively fast to the end user.
* JavaScript makes it easy to add extensions to web pages.
* CSS minimizes time, eases maintainability, and seems far superior to HTML for our implementation .

The only disadvantage to using JavaScript and CSS is that the client side execution can be taken advantage of, hence the user’s system can be manipulated by hackers and this creates security issues.

The login page of was built using a bootstrap theme. We implemented the javascript library and source code found on <https://colorlib.com/wp/html5-and-css3-login-forms/> Implementing this code to fit our website description was quite challenging. Most of the CSS code provided by the bootstrap had to tweaked a bit.

The main challenge came from building the chat window. We had to brainstorm what the chat room UI should look like. We decided to go with something similar to GroupMe/WhatsApp design with the pie chart to the right side of the screen.

This step was implemented incrementally. We first setup the chatbox with different users messages showing in the chatbox. This feature was introduced in V0.1.

We then segmented the webpage into 3 sections, 20% of the width was designated to show each user’s list of active room. It item in the list is presented in a button form, and on each click, the room changes to the specified room. The onClick() functions was implemented using JS. 50% for the chatbox and 30% for the pie chart. For more details about the pie chart, please refer to the “Implementing out Pie Chart” in the outline

**6. Data base**

**6.1 Our DBMS**

We have chosen to use MySQL because of its simplicity. Most other database management systems are more complicated but due to the relative simplicity of our app, MySQL is a smart choice. These are some more reasons we have chosen MySQL.

* MySQL saves into one file. This feature allows for easy management of the database file. It also allows for better portability because it can easily be stored and committed into Github.
* Great for websites: MySQL works great as the database engine for most low to medium traffic websites (most websites) in conjunction with PHP scripting.
* Client/server database engines are designed to live inside a lovingly-attended datacenter at the core of the network.
* Another perk with MySQL is the vast library of resources and tutorials for best practices.

**6.2 Base commands**

Here are some commands that we must use with our implementation of our database system.

* mysqli\_connect()
* mysqli\_query()
* mysqli\_real\_escape\_string()
* mysqli\_fetch\_array()
* mysqli\_num\_rows()

**6.3 DataBase Schema**

Our database will be broken down into 4 primary tables, along with an additional table created for each user as he joins the database.

**6.4 Registrar**

Here we will store all of the users and their password. The passwords are encrypted prior to entering our database.

**6.5 Rooms**

Here we will store the name and passwords for all rooms. Additionally each room will have a roomprompt and result column that is continuously updated and queried to allow all users to see the same information.

**6.6 Chat**

Here we store all of the messages, along with the user that sent it, and the room from which it was sent. The messages are queried by room, so that is how the multiple “rooms” are actually differentiated.

**6.7 Selections**

In this table, all of the choices of the various users are stored. Each selection contains the room, user, and choice. Additionally, another column called roomplususer is created, and serves as a unique identifier allowing us to limit each user to one choice for each room.

**6.8 User\_list**

Every time a new user joins the database, a new table is created titled User\_list (replace User with their username). Every time that user joins a new room successfully, that room will be added to their list. The list is displayed on the left side of the screen, and has a button for each room. Clicking said button will effectively change which room that user is in, without them having to enter the password again.

**7. Additional Library Considerations**

**GroupChat Capabilities:**

We used PHP, javascript, and MySQL to create the functionalityof our groupchat. We imported an ajax script from Google APIs to use some of the supported functions.

**Room Authorization:**

Room authorization didn’t require any special libraries. This portion was handled mainly by the mySQL database. When a user signs-up/create’s a room, we store the username/roomname and password in the database. But before entering the password in the DB, we first encrypt it using the md5() in php. This hashes the password to an integer value that is 32-40 characters long.

If a user tries to sign up with a name that already exists, our program is set to take the username and run it through the database, if there’s match, you get an ALERT. Same procedure also applies for room registration.

In order to validate a user/room, we run a query through the database using the username/roomname and password provided by the user. If there’s match, you get access to a room.

(Note we run the md5() method on the password, so as to decrypt before running the query)

**Implementing our Pie Chart for Display:**

In this app we made use of infographics to allow the user to easily interpret the results of taken polls. Specifically, we implemented a pie chart to display this information. After a room has been created and a poll question has been posted, the poll results (the selected choices) are displayed on a pie chart. This graphical representation is not available only after the poll has been ended but is updated realtime as the users in the room update their responses. For example, Suppose a 10 user room has been created to decide whether to go to the hampshire mall or to Big Y. Imagine that at first 5 people choose each place, then the pie chart would be split 50:50. A sudden change in mind of one person will cause the pie chart to reconfigure accordingly. We implemented this pie chart with the aid of javascript, MySQL, and PHP. We used PHP to collect user choice information and then load the data into our MySQL database. Then we accessed this data and created a pie chart from it using javascript.

* We imported the Google Visualisation API that was used to create the pie chart in javascript. It included many features including a way to handle selection events (when someone clicked on a slice of the pie chart).
* We connect the graph to the database using ajax that call PHP functions which get data to and from the database..
* After we established a connection to the database and then further ran queries to collect data from the database. We developed various data structures to hold our obtained values and passed them to the function that creates the pie chart.