**CIS 634 Object-Oriented Software Engineering**

**Course Project Proposal**

**Project: Power Management System**

Logo, company name

Description automatically generated

Instructor: Prof. Weidong Xiong

Group Number: 7

Group Members:

Tapan Desai CSU ID: 2777437

Shaishav Shah CSU ID: 2835285

Kinjal Patel CSU ID: 2794458

Mitee Patel CSU ID: 2817313

**1.0 Introduction**

The Power Management System is monitors and manages the electricity usage. It calculates the per unit charge for the customers and helps the administrator to manage each customers details from their registration to bill payment.

**1.1 Goals and objectives**

This web application provides monitoring platform for both Customers as well as the administrators. The central objective of this project is to provide a UI that is secure and simple to access. The goal of this report is to provide the team with specific details of all the features and design that are needed in this project.

**1.2 Statement of scopes**

The scope of the Power Management System includes its distinct features, its benefits, and its limitations. The system's distinct features allow to view the customer panel, look up electricity usage, pay the pending bills, and look up previous bills. The system enables the user to determine if there are any payments due.  **1.3 Software context**

Software required for this project are:

|  |  |
| --- | --- |
| Software | Role |
| Eclipse | Used to design Backend API |
| VS Code | Graphical User Interface, front end and react |
| Zamp | Servers(Apache and MySQL) |

* 1. **Major constraints**

The main constraint for this project will be that as we don’t have access to get all the actual bill data from an existing database. So, we will instead work with dummy data inputted by ourselves for this project.

* 1. **Design Choices**

Choosing a web application ensures that your application can easily run on a variety of devices and formats, which is beneficial for your project. Since this will be a commercial product, all technology used is free for commercial purposes. With this in mind, we have decided to build the application primarily using Java.

Java’s features should be useful during development, as they avoid many problems that arise when using older browser technologies. This way you can be more confident that your application will work correctly.

**2.0 Data design**

Configuration of a DBMS, Avoiding duplication of data. Each database should be automatically verifying user access to settings, networks, and data. We will be using relational databases store information with columns, rows, and tables. We will be using query to store the data on specific entity.

**2.1 Data structures**

**2.1.1 Backend**

The backend consists of a database, and a storage containing files. All interactions with either of these must go through the API. There are multiple tables spread across the database. Eclipse is used for designing the backend API’s. Zamp is used to start the server

**2.1.2 Frontend**

The Frontend is designed in Java. We have used different frameworks for the same. VS code is used for react.js and frontend.

**2.1.3 Components**

1) Web browser: The browser or client-side or front-end component is the primary component that facilitates user interaction with your application while interacting with the user, receiving input, and managing presentation logic. User input is also validated if necessary.

2) Web Server: Web server, also called back-end component or server-side component, handles user requests by managing business logic and routing requests to appropriate components to manage the overall application operation. . It can execute and monitor requests from various clients.

3) Database Server: The database server provides the data required by the application. Handle data-related tasks. In a tiered architecture, the database server can use stored procedures to manage business logic.

**2.2 Database description**

Each table of the database has a selected key that identifies the statistics withinside the table. To join one table to another, foreign keys are used. A foreign key's then related to a primary key.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table Name | foreign key | Data Type | Allow Nulls | Field Description |
| User | ID | Varchar(50) | No | USer is used to identify customer and admin |
| user\_role | ID | Varchar(50)  INT  DATE | No | Role is identified as customer or admin |
| list\_users | USERID | Varchar(100)  INT | No | it lists all the users present in the database |

**4.0 User interface design**

This Design represents what the user is going to see on the web application. The UI should be kept simple for user of every sector to interact with it with ease. There are no secondary features for the customer other than Payment or Viewing the previous records. All the update of the data i.e., Customer details or Bill change credentials must ONLY be done by administrator.

**4.1 Description of the user interface**

**4.1.1 Customer UI**

The customer can log-in using his credentials. If the user credentials are incorrect, he can request a password retrieval by contacting the administrator directly. The user can view his payment due, his information and a link to view his previous Bills. By clicking on the link, other web page opens which has filters for sorting the previous bills and a drop-down menu to select the month and year. Through the bill payment page, a payment gateway will open through which the payment can be made.

**4.1.2 Administrator UI**

The Administrator can log-in using the credentials provided. The administrator can view each customer and his details regarding the Bill Payment. Admin can enter the units used by each customer manually. The admin can view requests for Password Retrieval and provide with the password form the database. The admin can view the overdue bills of the user.

**4.2 Interface design rules**

The interface would be designed in HTML+CSS. Following are some design rules that must be followed up to a certain level. It not mandatory to follow all the rules.

**4.2.1 Consistency**

In comparable scenarios, equivalent action sequences should be needed; identical language should be used in prompts, menus, and help screens; and consistent color, layout, capitalization, typefaces, and so on should be utilized throughout. Exceptions, such as required confirmation of the delete command or no password echoing, should be understandable and restricted in number.

**4.2.2 Universal Usability**

Universal Usability is the design of things such that they are useful to as many people as possible. Recognize the demands of varied users and design for flexibility, allowing for content modification. The spectrum of criteria that leads design is enriched by novice to expert distinctions, age ranges, impairments, international variations, and technology variety. Including elements for both beginners and specialists, such as explanation and quicker pace, enhances the interface design and boosts perceived quality.

**4.2.3 Confirmation design for Closing**

Action sequences should be divided into groups with a beginning, middle, and end. Informative feedback upon the conclusion of a group of tasks provides users with a sense of success, a signal to abandon contingency preparations, and an indicator to prepare for the next series of actions. For this web application, for example, guide customers through payment confirmation, concluding with a clear confirmation page that completes the payment.

**4.2.4 Error Prevention**

Design the interface so that users can't make big mistakes as much as possible. For example, gray out menu items that are inappropriate and do not permit alphabetic characters in numerical entry fields. The user interface ought to provide straightforward, helpful, and specific recovery instructions if a user makes a mistake. If a user enters an incorrect username, for instance, they should not be required to retype the entire name-and-address form; rather, they should be directed to repair only the defective component. The interface's state should remain unchanged, or the interface should provide instructions for restoring the state in the event of an error**.**

**4.2.5 Reduce the load on short-term memory**

Designers should avoid creating interfaces that require users to remember information from one display and then use that information on another because of humans' limited capacity for information processing in short-term memory (the rule of thumb is that people can remember "seven plus or minus two chunks" of information).It means that long forms should be reduced to fit a single display, that phone numbers should not need to be entered again, and that web addresses should remain visible. For each web page, these fundamental principles must be interpreted, improved, and extended. They have some drawbacks, but they are a good place to start for web, desktop, and mobile designers.

**5.0 Restrictions and limitations**

**5.1 Restriction**

The Major restriction of this project is that it only involves interaction between the user and administrator. It does not involve any third party for technical support and maintenance.

**5.2 Limitation**

The data is manually entered into the database. Thus, the data can be un-synchronized or not structured. The retrieval of the data must be done manually by Administrator. For example, if the customer wants to change the DOB, or address it won’t be possible by the customer himself, but a request can be sent to the administrator, and he/she can change it and then sent the details back to the customer.

**5.2.1 Limited Mobile Access**

The display and features might not be synchronized or not ideal for viewing on Mobile or tablet. For example, the drop-down menu to select the month or the period of the bill generation – it would not be synchronized properly and thus the UI would not be easy to navigate.

**5.2.2 Storage limit**

The storage would be limited for the web Application. It includes the number of customers, the period for bill generation NOT exceeding 3 months for each customer and technical and maintenance request NOT exceeding 2 per customer.

**5.2.3 Reduced Speed**

A web program will most likely run at a somewhat slower speed than one hosted on a local server. As far as Browser Support is concerned, we don't all use the same browser, which is unfortunate. This implies that throughout development, you must guarantee that your software is compatible with a wide range of browsers.

**5.3 Manually Updating Required**

As it is a web application, all the updating would be done by the administrator manually. For example, if there is a change in per Unit charge or there are some changes in the records of customer, it would be done by the administrator himself. There is no UI available that can change the records or units. It must be done by the administrator on the required Database.

**7.1 Future Scope**

1.Complaint box: Provide a message box for customers to give their feedback or raise an issue to the admin

2.Payment: Provide payment gateway to the customer to pay the bills directly

3.Bill edit: Provide an editing feature to make correction on the existing bill

4.Notification: For payment due