

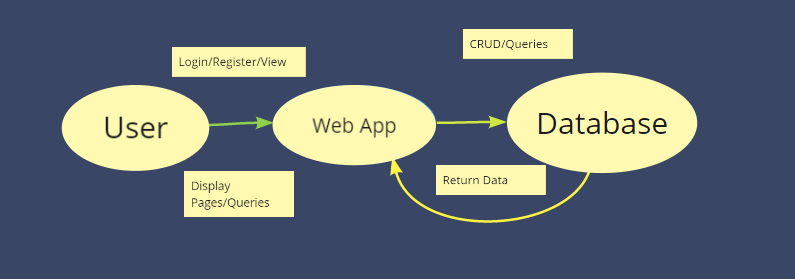




**Revision Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Modifications | Date | Members who modified |
| 1 | Skeleton document | Original submission 11/8 | Pierre Lucceus |
| 2 | [A] through [D] including Figures 1 through 3 | 11/8 | Alexander DeForge |
| 3 | [E] | 11/9 | Douglas Evert |
| 4 | [F] | 11/10 | Galen Yanofsky |
| 5 | [H] and [I] | 11/10 | Pierre Lucceus |
| 6 | Aggregating and formatting | 11/11 | Alexander DeForge |

**Project Analysis**

Our webapp will consist of a server hosting three frontend web pages and a backend containing a database. The three webpages are: LOGIN, USER VIEW, and ADMIN VIEW, as per our Requirement Specifications version 3. See *Figures* 1 through 3.

1. **What outside systems does this system interface with? [A]**

The client/user web browser

1. **What are the input data? [B]**

Weakly typed JavaScript variable text and numerals

1. **What are the sources of input data? [B]**
   1. **LOGIN WEBPAGE**
      1. Username text field
      2. Password text field
   2. **USER VIEW WEBPAGE**
      1. Demographic text field
      2. Buttons ADD/REMOVE Demographic
   3. **ADMIN VIEW WEBPAGE**
      1. Names/Addresses/Phones/SalaryRange/YearsEmployedRange/PTOBalanceRange/Demographics/Notes text search fields
      2. Buttons SEARCH/REMOVE/ADD/UPDATE/CLEAR
      3. Click selection of result (for display)
      4. Buttons ADD/REMOVE Names/Addresses/Phones/Demographics/Notes
      5. Name/Address/Phones/Salary/YearsEmployed/PTOBalance/Demographics/Notes ADD/UPDATE text fields
2. **What are the output data? [C]**

SQLServer query results

1. **What are the sources of output data? [C]**

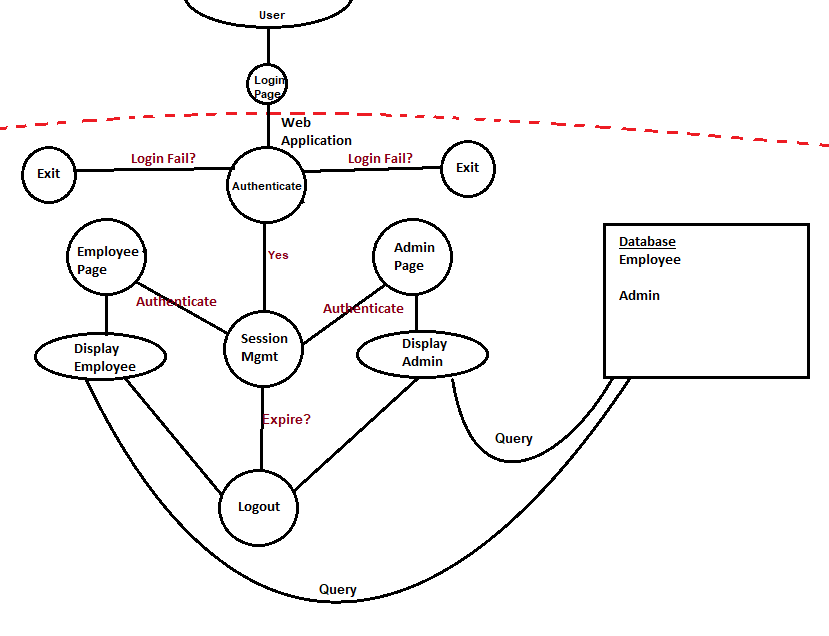
Backend SQLServer

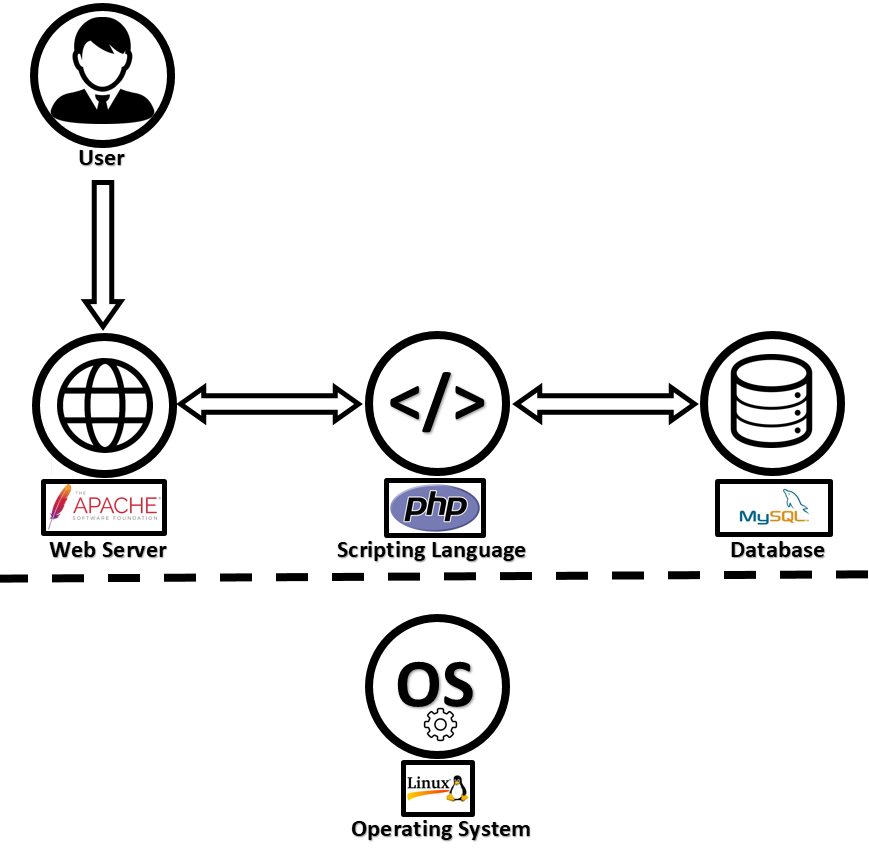
1. **What are the destinations of output data? [C]**

The client machine, via the webpage data being accessed and served through an HTTP connection from our server

1. **How do we convert the input data into output data? [D]**

PHP scripts will convert frontend data into queries for the database, and they will convert queries from the database into data that JavaScript can display in the webpage.

1. **What are the data processing subsystems? [E]**
   1. Authentication - this subsystem will take information from the initial login html form and through a PHP prepared SQL statement will potentially retrieve database information. If the password field matches the username from the field the user will be allowed to continue if not the user will be faced with a window that states password or username are incorrect.
   2. Retrieve User Information (RUI) - this subsystem is used to get data from the database. The RUI will get the correct user data by querying the database using a unique user ID that will be used in the database. The PHP query statement will use a prepared statement which selects information from the user table and returns it to an employee object.
   3. Displaying User Object - Once a user object has been created it will be displayed using the displaying user object subsystem. This system will break down the employee information into individual fields which will be used to dynamically create the HTML for the user’s frontend.
   4. Creating admin view - This subsystem will be used to create the administrative HTML page using PHP on this page the user will be able to access the user object as well as there will be added buttons that will include updating/adding/deleting employee information. This page will be dynamically created based on the administrators field in the users table inside of the SQL database.
   5. Updating employee record (Administrative System). The updating of an employee record will be conducted in this subsystem. After the user enters the appropriate information into the application frontend, either the user will click update (this option is only available to administrative users). after the button is clicked a php prepared statement will be created to update the employee record based on the employee ID in the employee table.
   6. Adding Employee Record (Administrative System). Creating a new employee record will be completed by this subsystem. The addition will take place after the user enters the new employee information into the HTML form and clicks the Add button (this option will only be available to the administrative users). After the add button is clicked the PHP will create a prepared insert statement into the employee table.
   7. Deleting Employee Record (Administrative System). The deletion subsystem will be used to remove records from the employee table. Once an administrator enters an employee’s name or ID and clicks the delete button on the HTML form (this option will only be available to the administrative users) a PHP prepared SQL statement for removing a record from the employee table where the employee ID or employee name equals what the administrator enters will be passed to the database.
2. **Identify interface data between each subsystem (and which subsystem processes the inputs, which subsystem does the output). [F]** 

Using LAMP as our software stack and for the basis of our program, we rely on the interaction between each component to drive web application processes. A solution stack like LAMP, consists of the following components to be functional.

* + 1. Operating System
    2. Web Server
    3. Database
    4. Programming Language

Each of these components processes unique data and accomplishes a resulting action. The diagram below illustrates how each of the components interface with each other. The application runs on the Linux OS and arbitrates all system calls made by the web browser/server.

Initially, the user accesses the application through a web browser. They provide user input to the application. For a login action, the user input is database specific and usually consists of a pair, or set of credentials that is unique to the user. In modern login applications, a session cookie is set with an expiration date. The web application doesn’t reach out to the database for subsequent request but trusts the credentials in the cookie as long as it is not expired. For our application, the front end will be HTML based. It will interface with a backend PHP Script that subsequently queries a MySQL database.

This simple interaction is how all processes are handled within a web application that uses the LAMP software stack. Each of the subsystem software components support each other to provide a complete user experience.

1. **Input Front-End => Back-End Response**

**HTML PHP/MySQL DB Connection**

1. **Back-End Validation =>** **Front-End Response**

**MySQL Query User Action Success**

**PHP User Input Validation User Action Failure**

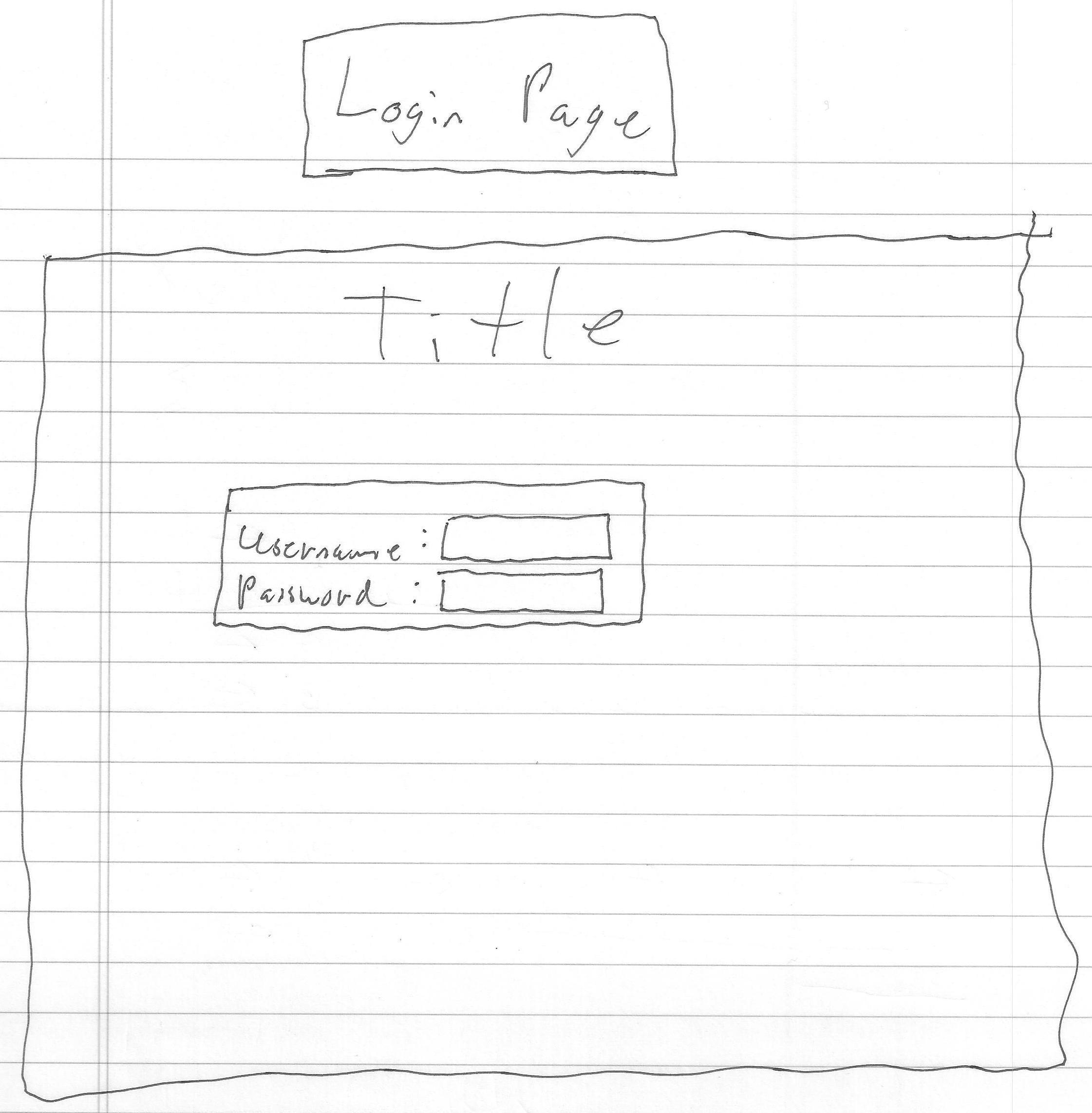
1. **Identify risks in your solution and possible ways to mitigate those risks [H]**

Web applications are susceptible to cyber attacks such as SQL injections since we are using a database to authenticate users and as well as getting the necessary data and make required changes. It is important that during the development process that we use secure coding practices and that we test the product for vulnerabilities(penetration Testing) using manual and automated test.

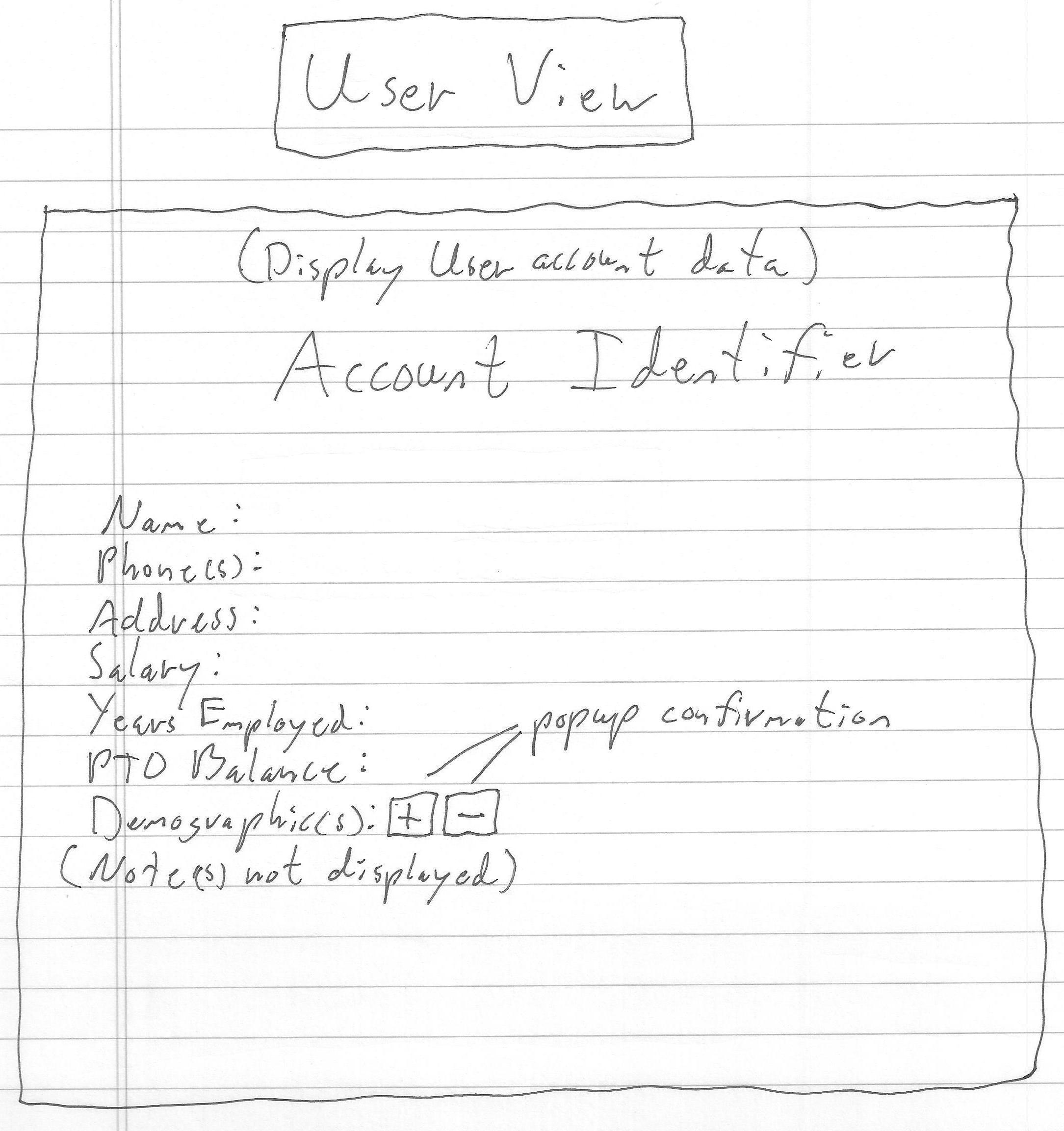
1. **Identify possible enhancements (new features) to the system; this is a way to get future work. [I]**

One possible new feature or enhancement to the system is additional layer of security during authentication. Instead of just relying on just username and password to log on we can add an email verification system that would generate a random number at login and send it to the user’s email. The user will be required to enter the random number in order to complete the authentication process. This will also help prevent brute force attack.

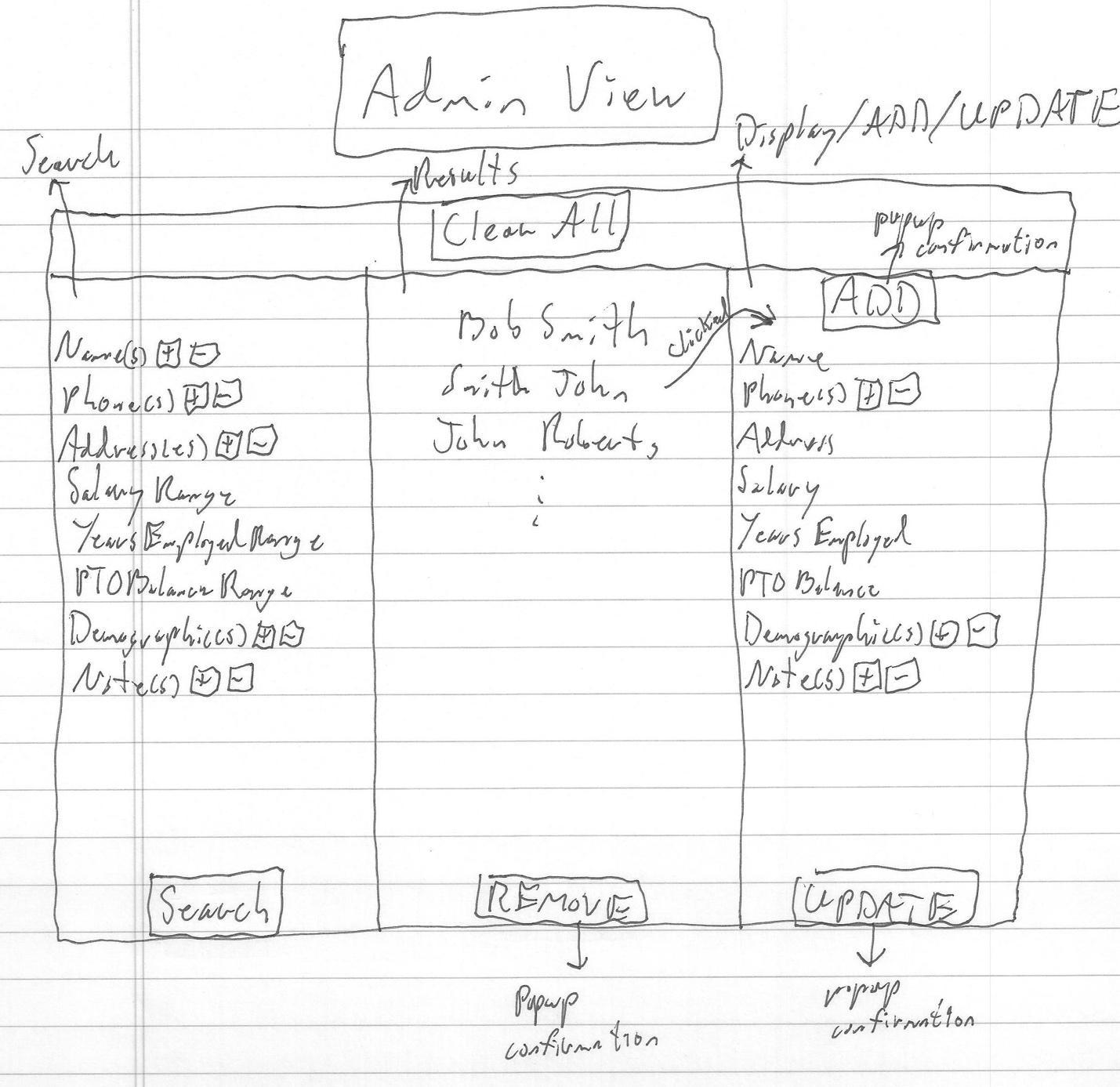
**Appendix**

****

*Figure 1. Login page mock up*

****

*Figure 2. User account webpage mock up*

****

*Figure 3. Administrator webpage mock up*