

**Revision Table**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Modifications | Date | Members who modified |
| 1 | Skeleton document | Original submission 11/15 | Galen Yanofsky |
| 2 | Added Legend, Section Explanations | 11/16 | Galen Yanofsky |
| 3 | Added Event-Trace Diagram | 11/17 | Pierre Lucceus  Douglas Evert  Galen Yanofsky |
| 4 | Adding Risk / Mitigation Table | 11/18 | Douglas Evert  Pierre Lucceus |
| 5 | Initial class diagram and pseudocode, added appendix | 11/18 | Alexander DeForge |
| 6 | Appendix diagrams (php code) | 11/18 | Galen Yanofsky |
| 7 | Inserted digitized Trace Diagrams | 11/19 | Pierre Lucceus  Douglas Evert |
| 8 | Inserted proposed database schema | 11/19 | Galen Yanofsky |

**Project Design**

**(A) Class Diagram and Pseudocode:**

The following class diagram shows the conceptual structure of our application’s attributes and operations. The class diagram will eventually be used for deriving both pseudo, as well as functional application code. This pseudocode is supported by the database schema seen in Figure 3.

Index.html

<script src=”login.js”></script> (dynamic content)

<link rel=”stylesheet” href=”login.css”> (webpage style)

<?php

POST login credentials from username and password form

Redirect to appropriate page based upon authentication

?>

Admin.html

<script src=”admin.js”></script> (dynamic content)

<link rel=”stylesheet” href=”admin.css”> (webpage style)

<?php

(Initial state is all in the html/js/css)

On interaction, POST Add, Update, or Remove data for sanitizing

Sanitize POST’d data before SQLServer action

Act on SQLServer

GET results of action to be ECHO’d in page

Redirect to index.html (login page) upon logout

?>

User.html

<script src=”user.js”></script> (dynamic content)

<link rel=”stylesheet” href=”admin.css”> (webpage style)

<?php

Query SQLServer based upon user that logged in

GET results of query to display for initial state

On interaction, POST Update data for sanitizing

Sanitize POST’d data before SQLServer action

Act on SQLServer

GET results of action to be ECHO’d in page

Redirect to index.html (login page) upon logout

?>

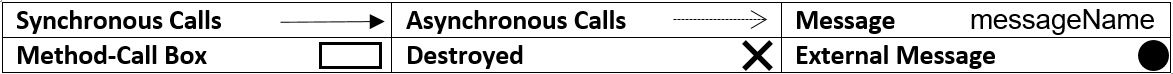
An interaction is a button press.

SQLServer action is reading or modifying the state of the database.

**(B) Event-Trace Diagram:**

An event-trace, or sequence diagram shows application specific entity interactions arranged in a chronological manner. Its role is to communicate the desired functionality of a system or application to a particular audience or customer. The legend below shows the different symbols that comprise this type of diagram.

**Diagram Legend**

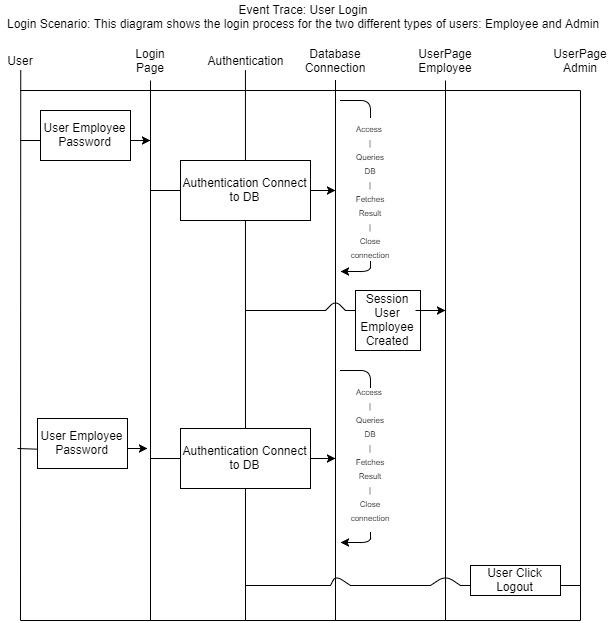
****

**Scenario 1**: Login

Description: the user enters his or her username and password in order to log on to the application

Precondition: The user have to open compatible browser to access to web application

Post-condition: The app will authenticate the user through the database, upon successful authentication, a session will be created and the user take to respective user page.

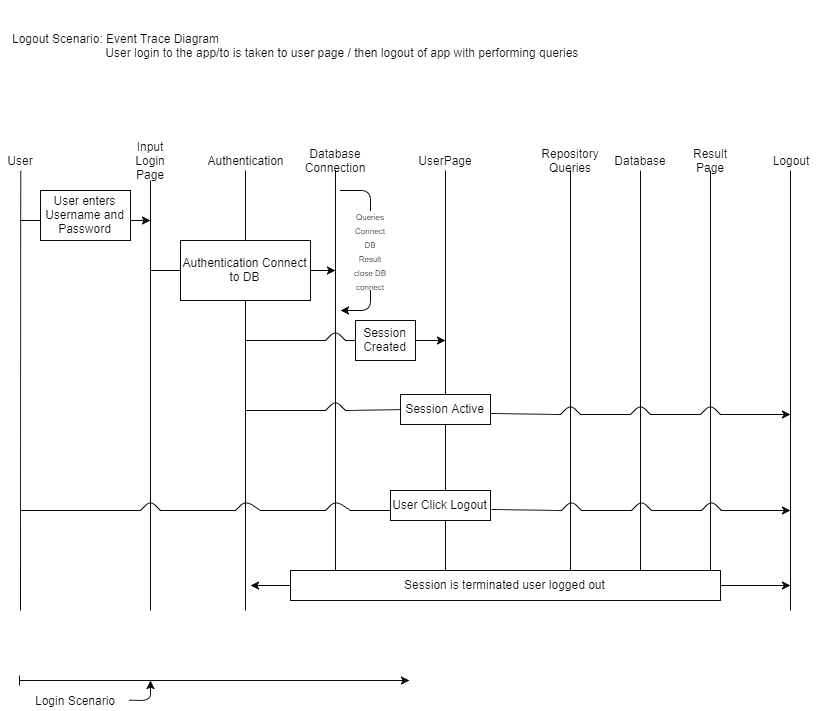


**Scenario 2**: Logout

Description: the user Click the logout button to terminate the session

Precondition: The user have an active session

Post-condition: The app will log the user out, terminate any open sessions as well as any connection to the database and the authentication will be revoked.

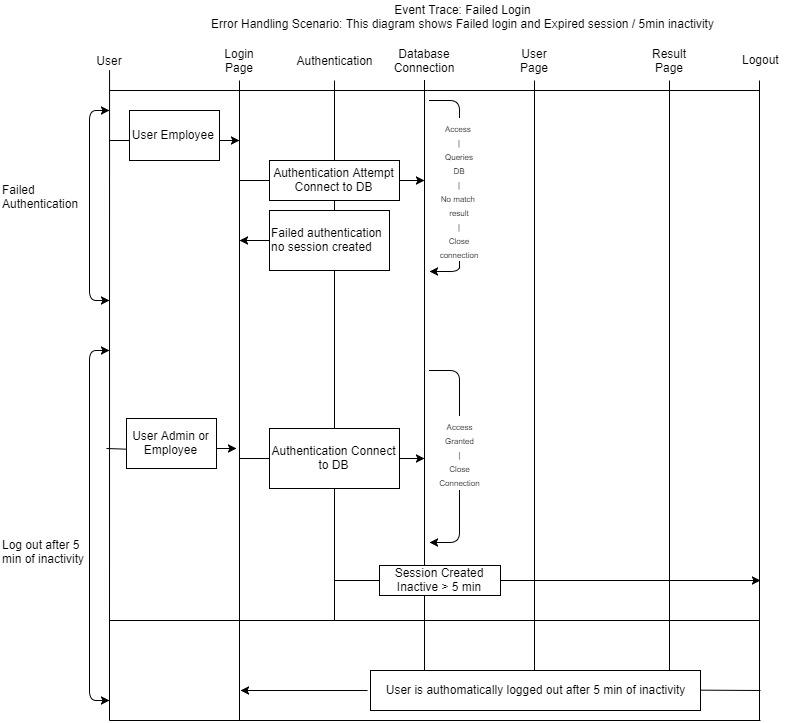


**Scenario 3**: Error handling

Description: the user enters wrong credentials

Precondition: The user have to open compatible browser to access to web application

Post-condition: The app will attempt to authenticate the user through the database by matching the combination of username and password. If there is no match, connection to DB is closed, session creation would fail and the user will be redirected to the login page

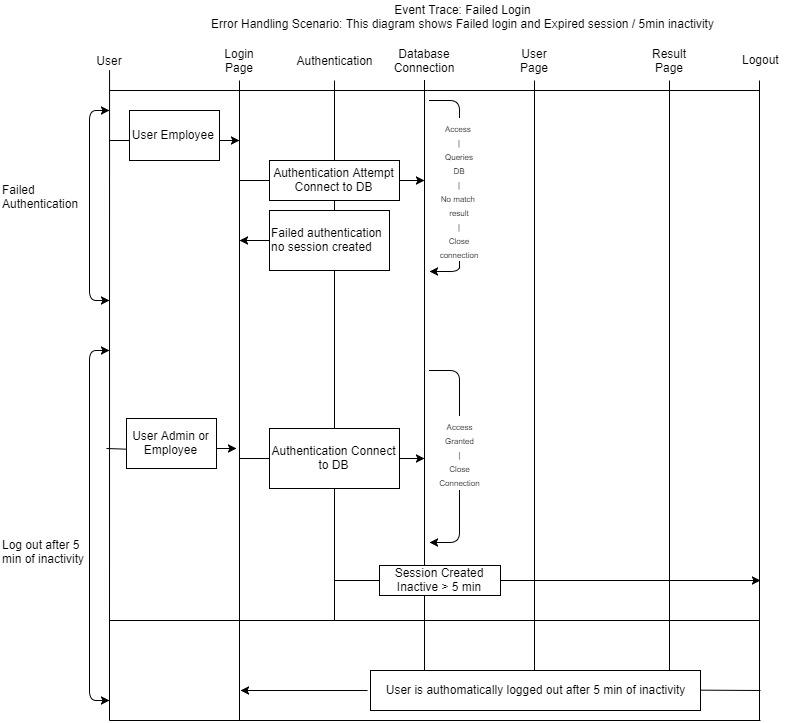


**Scenario 4**: > 5 min of inactivity

Description: The user is automatically logged out after more than 5 min of inactivity

Precondition: The user have an active session but does not perform any action for more than 5 min

Post-condition: The app will log the user out, terminate any open sessions as well as any connection to the database and the authentication will be revoked.



**(C) Unresolved Risks/Mitigation:**

|  |  |
| --- | --- |
| Risk | Possible Mitigation |
| Directory Transversal | Modify HTTPD to ensure this is blocked |
| Distributed Denial of Service (DDoS) | Load balancing and proper firewall configuration. |
| Social Engineering | Staff will have to be trained on proper handling of their credentials: not to share over the phone or through email etc… |
| Cross Site Scripting | Users can be informed not to access the application through links received from email but directly through a browser. |
| SQL injections | While we tried our best to develop a secure application, because of the time constraints the application may still be subject to highly sophisticated SQL injection attacks. We can use backup to insure that the data will not be lost. Also have a disclaimer to the client and discuss risk tolerance before going live. |
| Unavailability/Natural disaster | Because we are using a private server(VM) the application may not always be available for access due to internet connectivity issues caused by Natural disasters or technical problems from our personal computer that will be hosting the application. Possible mitigation is the use of cloud services such as AWS. |
| Deletion / Upload of Unauthorized material with HTTP methods | Ensure that the PUT / DELETE Methods are not active on the Web Server. |

Appendix

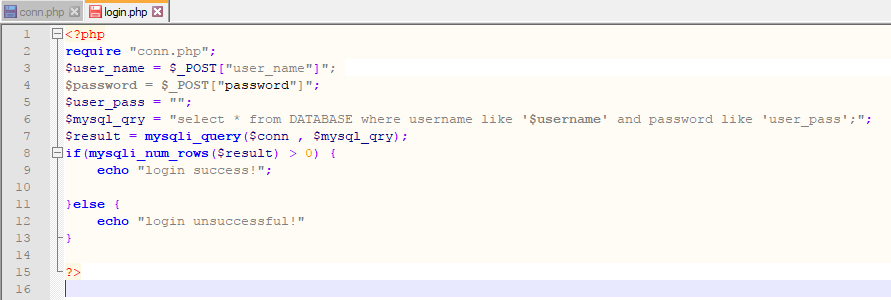


Figure 1. Additional reference with pseudocode

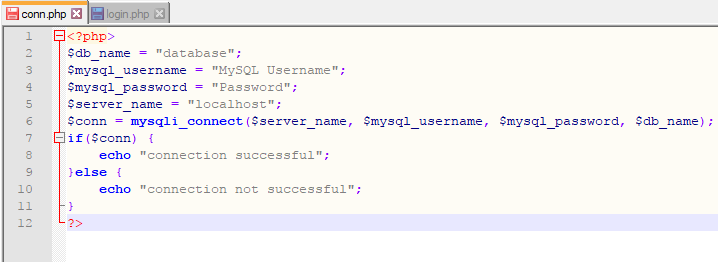


Figure 2. Further additional reference with pseudocode

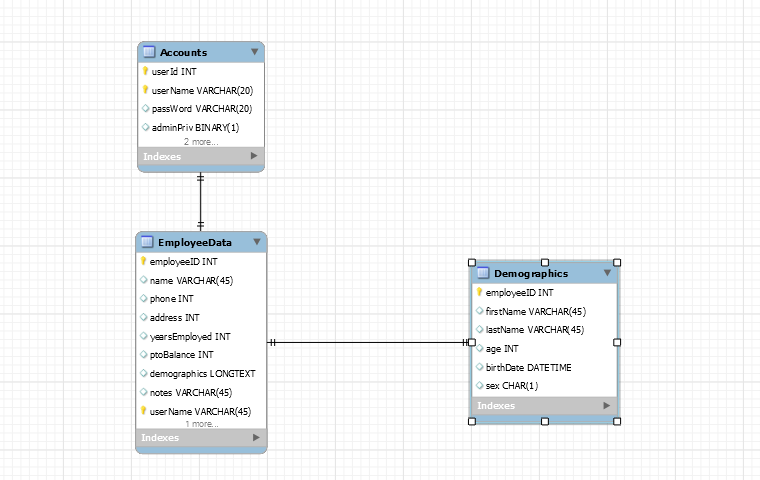


Figure 3. Proposed database schema