**Milestone Four: Databases Enhancement**

Harrison Bergeron

CS 499: Computer Science Capstone

Prof. Brooke Goggin

July 31, 2022

For the databases category, I enhanced an artifact from IT 270 that was initially a website consisting of non-functional static webpages representing a fake real estate company. The home page featured a “Chat With an Agent” box at the bottom of the screen that allowed the user to enter text into an input field to mimic the concept of live customer service chats. I created this artifact to gain experience in web development by independently building and styling a website frontend over the course of a few weeks. I selected this artifact to enhance because while creating it, I took advantage of multiple learning opportunities to further my knowledge about HTML, CSS, and JavaScript, and this artifact showcases my skills in these subjects as well as provides a starting point for enhancements (CS-499-04 & CS-499-05). The “Chat With an Agent” feature included live JavaScript code but was client-side only and did not actually communicate with a remote server. To enhance this part of the artifact, I created a Node.js server and a MySQL server to accept chat message traffic from the original script and store those chat messages within a database. For company records, it may be desirable to store customer service chat logs so that they may be referenced in the future if necessary.

The original script only instantiated UI elements from chat message text and displayed them visually for the client with no real client-server interactions, as the feature was only conceptual. In enhancing this artifact, I created a new script to implement a localhost server with Node.js as the vehicle. In addition, I created a MySQL database server on the same machine. I updated the original script to send HTTP requests to the Node.js server, which then performed SQL queries on the MySQL server to store incoming chat messages from the client within a database. The Node.js server acted as a middleman between the client and the MySQL server. By setting up this MySQL server and managing the database programmatically in the website backend, I target the CS-499-04 course outcome by providing a database solution that delivers value and accomplishes a common industry goal of storing website traffic within a database. I utilized asynchronous methods to make HTTP requests on the client, process them on the server, and make SQL queries on the server, in order to avoid blocking the main thread unnecessarily. I also nested asynchronous callbacks to prevent SQL queries from beginning before a previous query completed. This demonstrates my ability to solve problems involving storing, manipulating, and accessing data effectively and targets the CS-499-04 outcome because asynchronous queries prevented database errors and inconsistencies from arising. Additionally, I used HTTPS and SSL within the Node.js server by using OpenSSL to generate a private key and certificate to allow for encrypted connections between the client and server. I included a hardcoded username and password in the Node.js script to establish a secure connection to the MySQL server. I also used prepared statements in server SQL queries to prevent injection-based attacks on the database. These parts of the enhancement target the CS-499-05 outcome and demonstrate my ability to address design, logical, and structural flaws related to database security by protecting client-server traffic as well as preventing malicious users from attempting to access the database through security loopholes. My enhancement also targets the CS-499-05 outcome because I use innovative skills and techniques to check for database and table existence before modification and create the necessary databases and tables if they don’t already exist, which addresses a design flaw that could impact the security of the database. Similarly, I mitigated design flaws in the client-side script that had the potential to cause index out of bounds errors and overconsumption of browser memory.

When enhancing this artifact, I learned a great deal about Node.js, SQL, and HTTPS in a single week. I learned how to create SSL certificates, create and send HTTP requests, setup a server that hosts files and processes client requests, connect to a MySQL database remotely, and create a variety of SQL queries to search, insert, and delete database records. I learned how to create prepared statements for SQL to prevent injection attacks for security purposes, which is a ubiquitous goal in database design. It took a considerable amount of time to familiarize myself with asynchronous programming in Node.js but doing so paid dividends in facilitating the creation of the server script and processing client requests and queries. A surprising challenge I faced was using the command line interface to not only generate a private key and certificate request, but to sign the certificate manually and apply the necessary extensions to make the certificate valid in Google Chrome browsers. I learned that OpenSSL documentation was widely considered to be lacking and in need of redesign. I spent the most time with OpenSSL documentation, followed by Node.js and SQL documentation, although I still needed refreshers on JavaScript DOM methods and the XMLHttpRequest object.

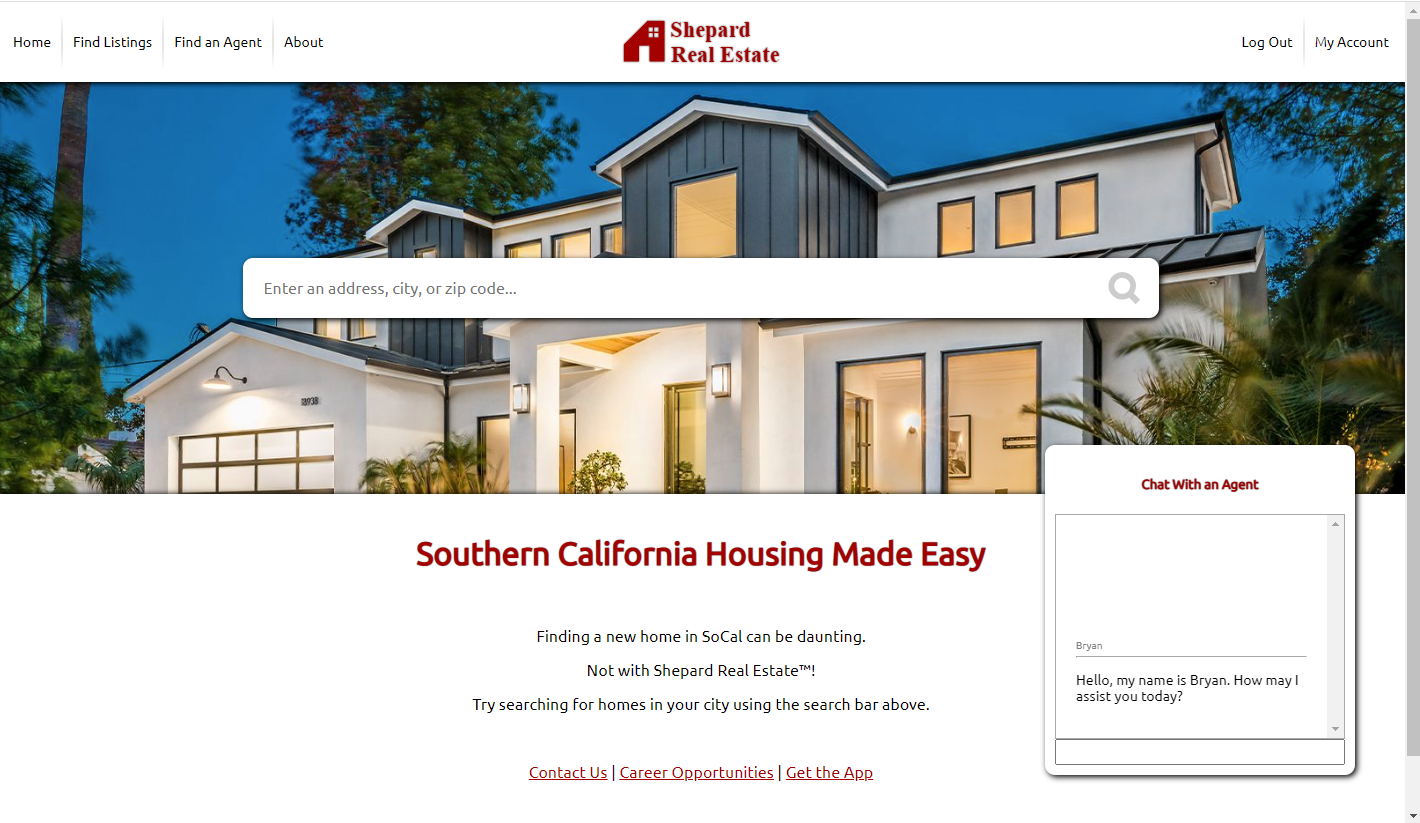
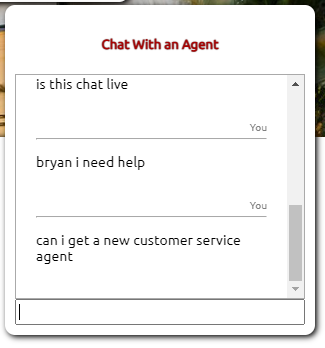
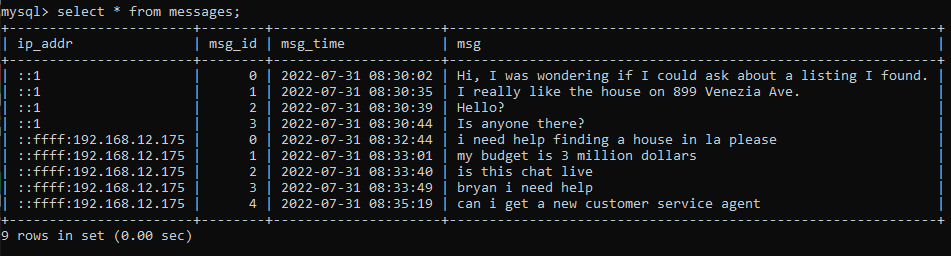


Figure 1: Original website

Graphical user interface, application

Description automatically generated

*Figure 2: User #1 and user #2 client chats*

**

*Figure 3: “Remote” MySQL database with chat messages from each user stored under different IP addresses*