**Database Application Programming**

**Project Status and Design Report**

|  |  |  |
| --- | --- | --- |
| **Topic:** | CST-126 Blog Project | |
| **Date:** | January 16, 2021 | |
| **Revision:** | *1.5* | |
| **Team:** | 1. Casey Huz | |
|  | |
|  | |
|  | |
| **Team Status:** | |  |  |  |  | | --- | --- | --- | --- | | **Task** | **Team**  **Member** | **Hours**  **Worked** | **Hours Remaining** | | Update design report. | Casey Huz | 1 | 0 | | Code Testing | Casey Huz | 3 | 0 | | Fine Tuning of viewAllPosts.php page | Casey Huz | 2 | 0 | | Creation of deletePostHandler.php page | Casey Huz | 2 | 0 | | Creation of editPost.php page | Casey Huz | 2 | 0 | | Creation of editPostHandler.php page | Casey Huz | 3 | 0 | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | |  |  |  |  | | |
| **GIT URL:** | https://github.com/caseycase5/CST-126 | |
| **Hosting URL:** | https://chuz-cst126-blog.azurewebsites.net | |
| **Peer Review:** | **Y** | We acknowledge that our team has reviewed this Report and we agree to the approach we are all taking. |

**Supporting Design Documentation**

**Install Instructions:**

*Include step-by-step instructions for setting up your database, and configuring and deploying/installing your application. This section should also include detailed instructions for what configuration files are required by your application, what configuration settings need to be adjusted for various runtime (development or production) environments, and where the files need to be deployed to. This section should also contain detailed instructions for how to clone your application source code from Bitbucket and deploy the application to an externally hosted site.*

The program now consists of a home page (index.html) where the user can select whether they want to login or register, as well as a register page and login page. The register page will redirect to the index page once complete. Once logged in, the user has access to a page that allows them to add a new blog post, view all posts, or edit their user information. The project has now been hosted on Azure, allowing for access on any machine with access to the internet. A copy of the program and SQL server are stored on the Github Repository linked above.

**WEB PAGES ACCESSIBLE ON AZURE WEBSITE:**

Index.html

login.html

loginHandler.php

register.html

registerHandler.php

newpost.html

newPostHandler.php

editUserInfo.php

editUserInfoHandler.php

useraccount.html

viewAllPosts.php

editPost.php

editPostHandler.php

deletePostHandler.php

**General Technical Approach:**

*You should, in words, describe your approach and design here. You should also summarize any meeting notes, brain storming sessions, etc. that you want to retain through the design of your project.*

For the first milestone, I planned to create a registration page that requires the user to enter a username, password, name, role, email, and address. This led to me having to create inputs for 11 different user inputs as well as an auto-generated primary key on the SQL server. All user input will be required and will have maximum parameters that match the parameters on the SQL server. When the “register” button is hit, a php script will gather this information, assign it to variables, then push those variables to the sql server to create a new entry in the “user” table. A login page has been created with data validation, ensuring that the user inputs something into both the username and password fields. If the username and matching password are found, the user will be logged in. If not, the user will be notified that login was unsuccessful.

A new post page has now been created that allows the user to input a post title and post content, both of which are required before the form can be submitted. Once submitted, the contents of the fields are saved to a new SQL table within the same database (posts table). The user is then given a cue showing whether or not the post was successfully saved. In the future, this will link back to the main post page once that is created.

The project and SQL storage are both now hosted on Azure. Since the project is now in the public domain, additional security requirements were required to be implemented. When the project is updated locally, the files must be committed to the git, then pushed to the azure server in order to reflect the changes on the hosted website. I had issues getting my webpage to connect to my SQL server via Azure and haven’t found out how to get them working quite yet. The SQL server still works well on my local server, however.

For Milestone 5, I created a page that allows the user to log in and adjust certain profile settings. ID, username, and password cannot be changed from here. I also created a basic view all posts page that shows a list of all posts currently stored on the SQL server. Any time a new post is created, it will be added to this page.

In Milestone 6, I added two buttons, an “Edit” button and a “Delete” button. Each of these buttons have a corresponding php file that handles their actions accordingly. The “Edit” button redirects the user to a page similar to the page where they would create a new post, however it automatically fills in the information for that post in order to allow the user to edit it. Once changes are submitted, another php file executes the SQL script that updates the server with the new post data. The “Delete” button has a simple php file that executes an SQL script that removes the post with the corresponding ID from the SQL server and therefore the page.

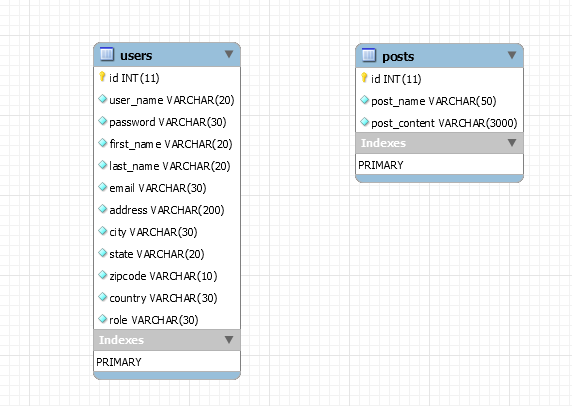
**Key Technical Design Decisions:**

*Any final technical design decisions, such as framework decisions and so forth, should be documented here. This should list the technology/framework, its purpose in the design, and why it was chosen.*

The blog site now consists multiple full pages, an index page which allow the user to create an account, edit their user settings, view all posts, and create a new post. The framework for the the pages is relatively finalized, but there may be some minor UI updates to clean up the presentation. All variables associated to the user will be required upon registration and will be stored in a “user” table in the SQL server. A separate login page has been created and requires the user to input both a username and password before making a query to the SQL server to determine if it is a valid account. The blog posts created on the new post page are saved to a separate table within the same database. In the future, the posts from this table will have their own page where they are displayed. Users and blog posts are saved to the SQL server. Anytime the user edits their user settings, the update is sent to the SQL server and will update the UI accordingly. The user also now update or delete any prior posts they saved to the SQL server. When they update a post, the new data is sent via an “Update” SQL command that simply updates the previous data stored on the server. The “Delete” function removes that post from the server.

**ER Diagram:**

*Include an image file of your ER database diagram.*



*A picture containing graphical user interface

Description automatically generated*

**DDL Scripts:**

*This should contain a link to Bitbucket where the DDL script can be downloaded.*

SQL server is included in the Github repo in order to import the database.

**Sitemap Diagram:**

*Include an image file of your Sitemap diagram.*

Diagram

Description automatically generated

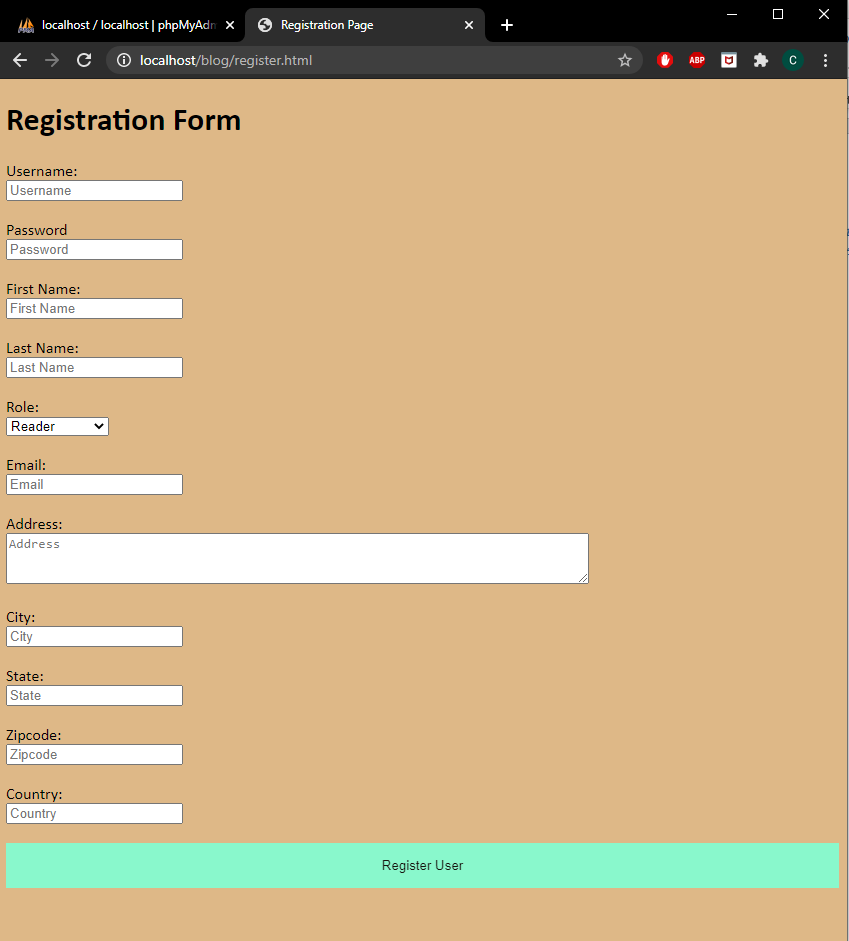
**User Interface Diagrams:**

*You should insert any wireframe drawings or white board concepts that were developed to support your application. If you have no supporting documentation, please explain the rationale for leaving this section as N/A.*

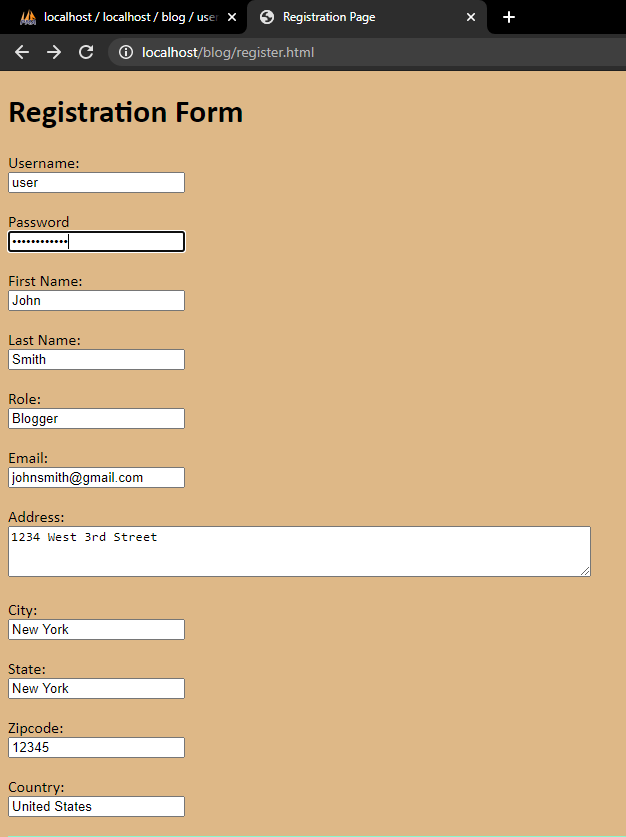
There are no user interface diagrams at this time. Screenshots of the current UI are provided below.

**Other Documentation:**

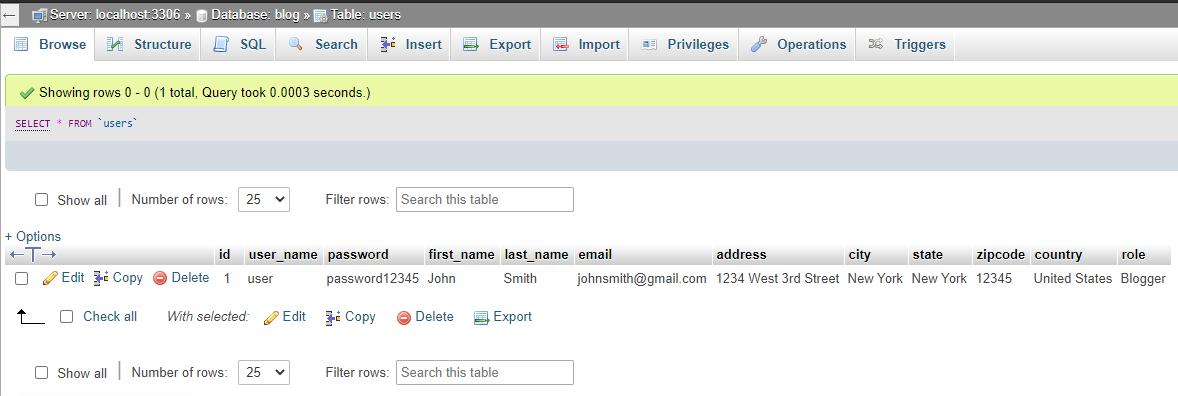
*You should insert any additional drawings, storyboards, white board pictures, project schedules, tasks lists, etc. that support your approach, design, and project. If you have no supporting documentation, please explain the rationale for leaving this section as N/A.*



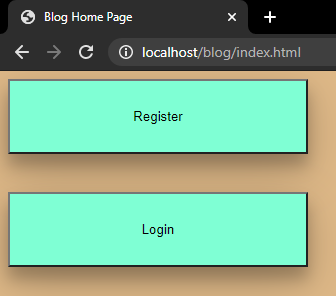
register.html accessed.



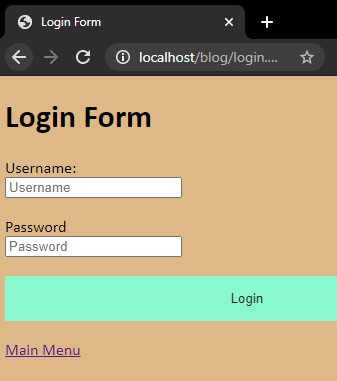
Filled in form



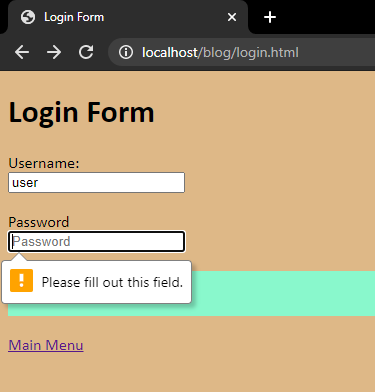
Data from form showing on SQL server after “submit” button is hit.



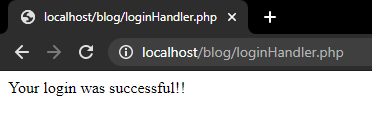
Index page that allows the user to choose what they want to do.



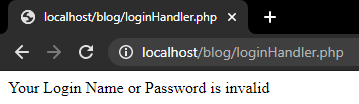
Login page with a link that leads to the main menu (index.html)



Data validation when filling out the login form



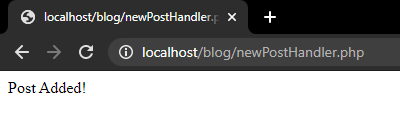
Successful login



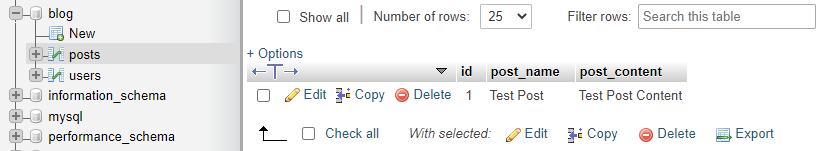
Unsuccessful login



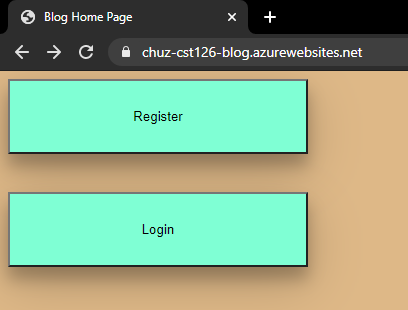
New post page



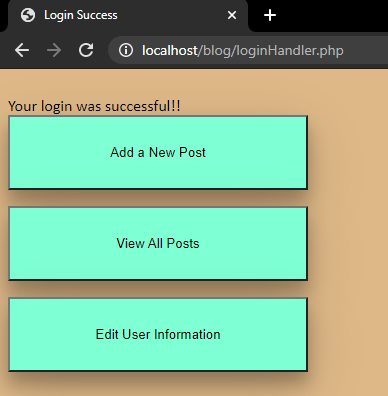
Confirmation that post was added to SQL server



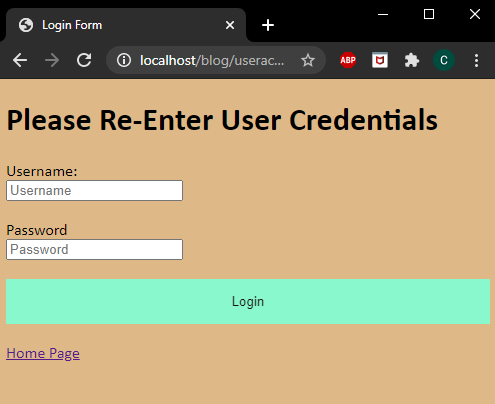
Post data in SQL table



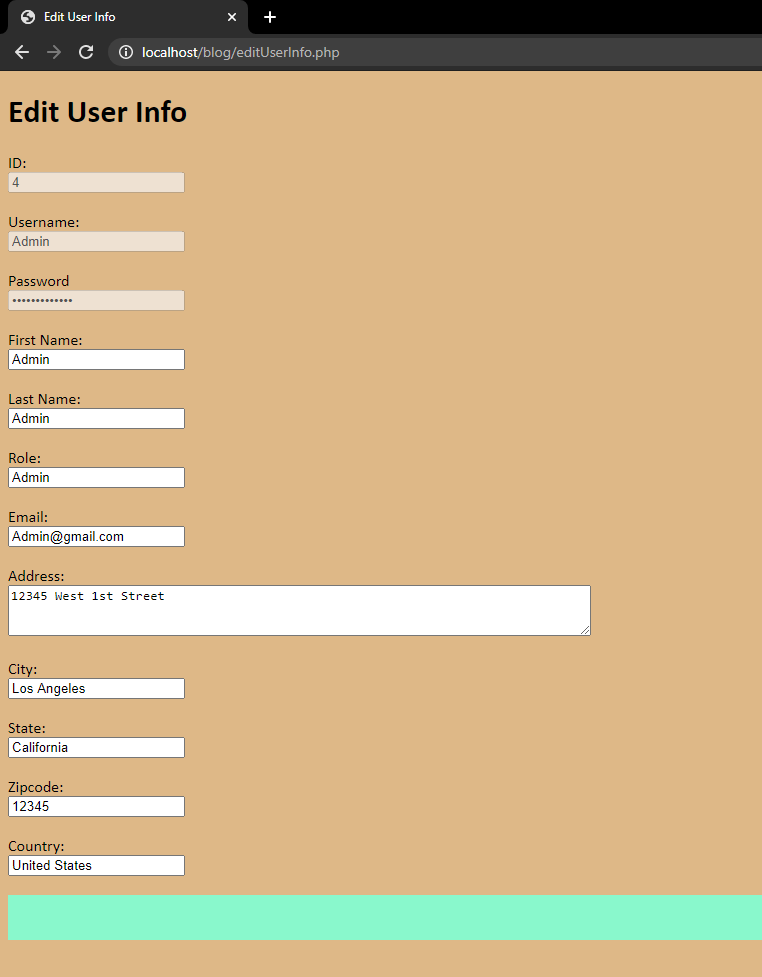
Webpage running on the Azure hosted site.



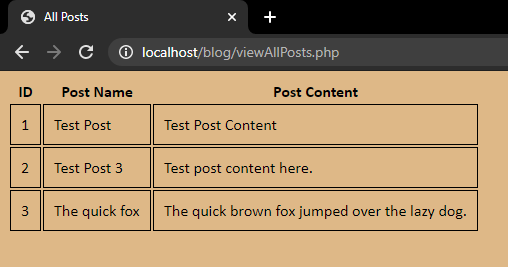
New Successful login page



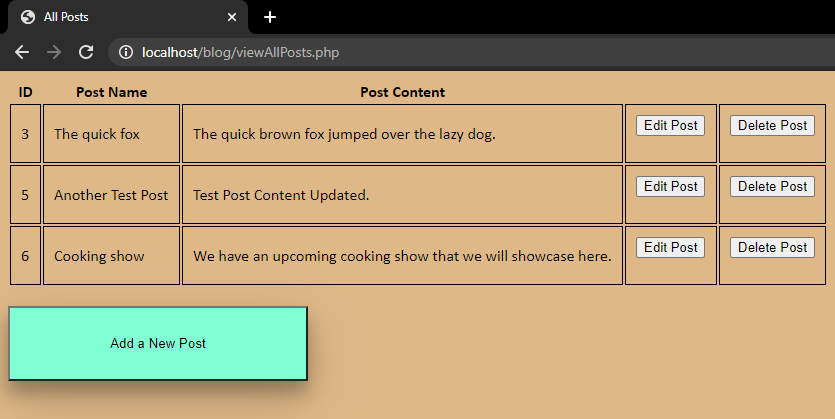
Attempting to edit credentials



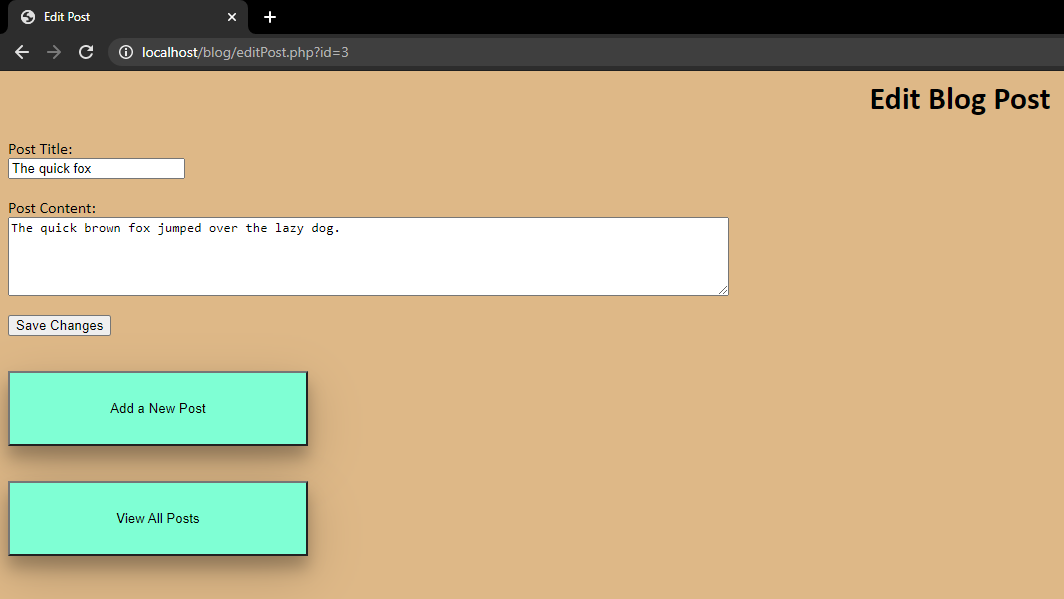
Edit user info page



Basic view all posts page



Updated View All Posts Page with Edit and Delete buttons.



Edit Post page with previous information automatically populated in the associated fields.