# Weekly Progress Report - February 15, 2023

This week, I focused on connecting the front-end HTML with the back-end Golang code for my ecommerce wine selling website. Specifically, I created a login authentication page that allows users to securely log in to the website and access their account information.

To accomplish this, I first set up a basic HTML login form that collects the user's email address and password.

```
<form method="post" action="/login">
  <label for="email">Email:</label>
  <input type="email" name="email" required>

  <label for="password">Password:</label>
  <input type="password" name="password" required>

  <button type="submit">Log in</button>
</form>
```

Then, I used Golang's net/http package to create a server that listens for incoming requests to the login page. Next, I created a login handler function in Golang that reads the user's email and password from the form data, validates them against a database of registered users, and generates an authentication token that is stored in an HTTP cookie for future requests.

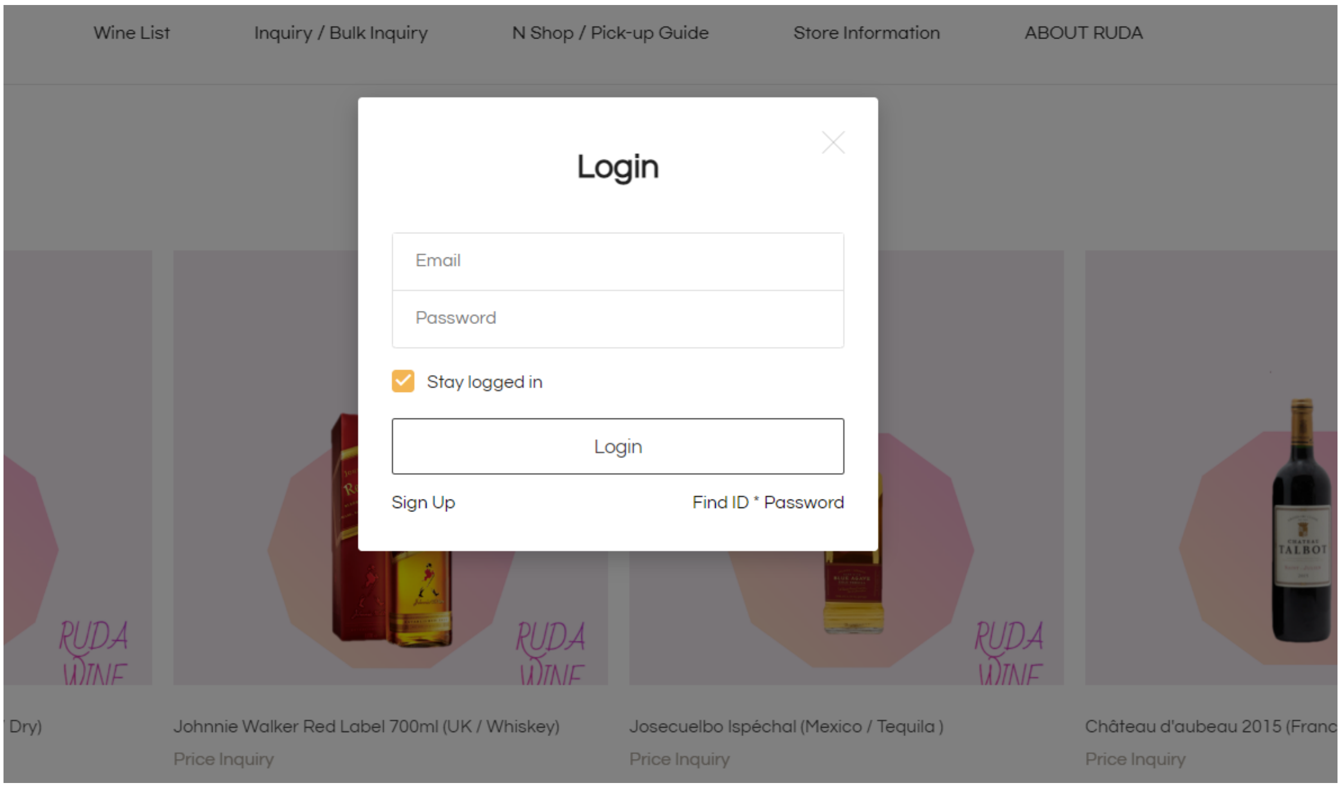
```
func loginHandler(w http.ResponseWriter, r *http.Request) {
    // Read email and password from form data
    email := r.FormValue("email")
    password := r.FormValue("password")

    // Validate user credentials
    if !validateUser(email, password) {
        http.Error(w, "Invalid email or password", http.StatusUnauthorized)
        return
    }

    // Generate authentication token and store in cookie
    authToken := generateAuthToken()
    http.SetCookie(w, &http.Cookie{
        Name:  "auth_token",
        Value: authToken,
    })

    // Redirect to user's account page
    http.Redirect(w, r, "/account", http.StatusSeeOther)
}
```

I also implemented some basic error handling to display error messages on the login page if the user's credentials are incorrect or if there is a problem with the server.

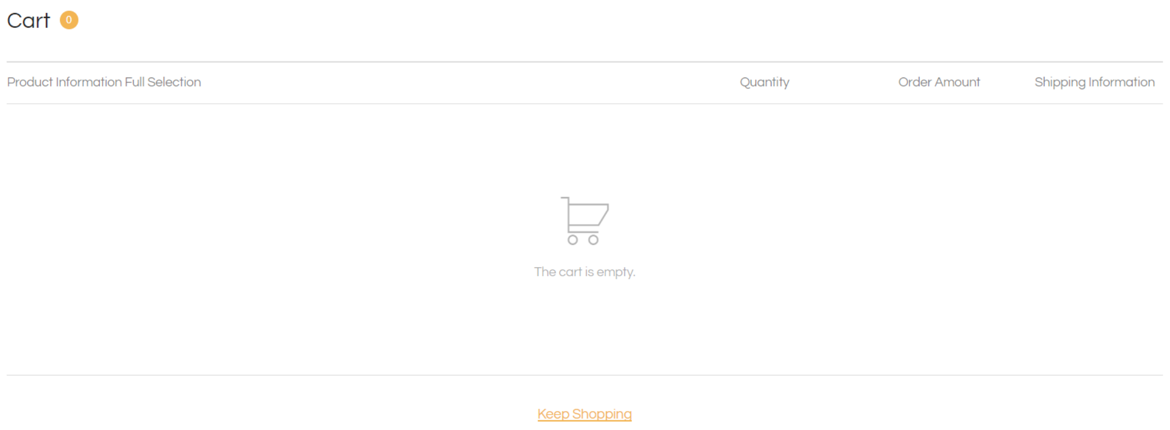


Overall:

1. Developed a login page that allows users to securely log in to the website and access their account information. This involved creating a login form using HTML, and implementing a login handler in Golang that reads the user's email and password from the form data, validates them against a database of registered users, and generates an authentication token that is stored in an HTTP cookie for future requests. I also implemented basic error handling to display error messages on the login page if the user's credentials are incorrect or if there is a problem with the server.
2. Added functionality for users to create a new account on the website. This involved creating a registration form using HTML, and implementing a registration handler in Golang that reads the user's name, email, and password from the form data, checks if the email is already in use, hashes the password using bcrypt, and stores the new user's information in the database. I also implemented basic error handling to display error messages on the registration page if the user's email is already in use or if there is a problem with the server.
3. Developed a product listing page that displays a list of available wines for purchase. This involved creating a simple HTML template for displaying wine information, and implementing a handler in Golang that retrieves wine data from the database and

populates the HTML template with the relevant information. I also implemented basic error handling to display a message on the product listing page if there are no wines available for purchase.

4. Added functionality for users to add wines to their cart and proceed to checkout. This involved creating an HTML form for adding wines to the cart, and implementing a handler in Golang that updates the user's cart information in the database and redirects them to the checkout page. I also implemented basic error handling to display a message on the product listing page if the user's cart is already full.
5. Developed a checkout page that allows users to review their order and enter their shipping and payment information. This involved creating a HTML form for collecting shipping and payment information, and implementing a handler in Golang that validates the user's information and processes the order. I also implemented basic error handling to display a message on the checkout page if the user's information is invalid or if there is a problem with the server.


Cart 0
Product Information Full Selection
Quantity
Order Amount
Shipping Information
The cart is empty.
Keep Shopping