Weekly progress report – April 14 (9th report)

For this week I've successfully implemented a payment system.

First I selected a payment gateway provider: I researched various payment gateway providers and decided to use Stripe due to its ease of use and strong dev documentation.

Next registered and obtained API credentials: I created an account with Stripe and obtained the necessary API credentials, including API keys and tokens.

After that integrated the Stripe API: I followed the Stripe API documentation to integrate the payment gateway API into my app. This involved making HTTP requests to the Stripe API endpoints, such as to create a new payment, process a payment, and retrieve payment details. I used the golang HTTP package to make these requests.

Also handled payment events: I configured my app to handle payment events, such as successful or failed payments, refunds, and chargebacks. I used Stripe webhooks to receive these events in real-time and updated the payment status in my SQL database accordingly.

```
    "github.com/stripe/stripe-go"
    "github.com/stripe/stripe-go/charge"
)

func processPayment(w http.ResponseWriter, r *http.Request) {
    // Set up Stripe API client
    stripe.Key = "sk_test_..."

    // Retrieve payment details from request body
    amount := r.FormValue("amount")
    currency := r.FormValue("currency")
    token := r.FormValue("token")

    // Create Stripe charge object
    params := &stripe.ChargeParams{
        Amount:      stripe.Int64(amount),
        Currency:    stripe.String(currency),
        Description: stripe.String("Wine purchase"),
        Source:      &stripe.SourceParams{Token: stripe.String(token)},
    }
    ch, err := charge.New(params)

    if err != nil {
        // Handle payment error
        http.Error(w, err.Error(), http.StatusInternalServerError)
        return
    }

    // Store payment details in database
```

Finally i stored payment data: created a SQL database schema to store payment data, including customer information, product details, and payment status. I used the Golang SQL package to interact with the database and execute SQL queries to store and retrieve payment data.

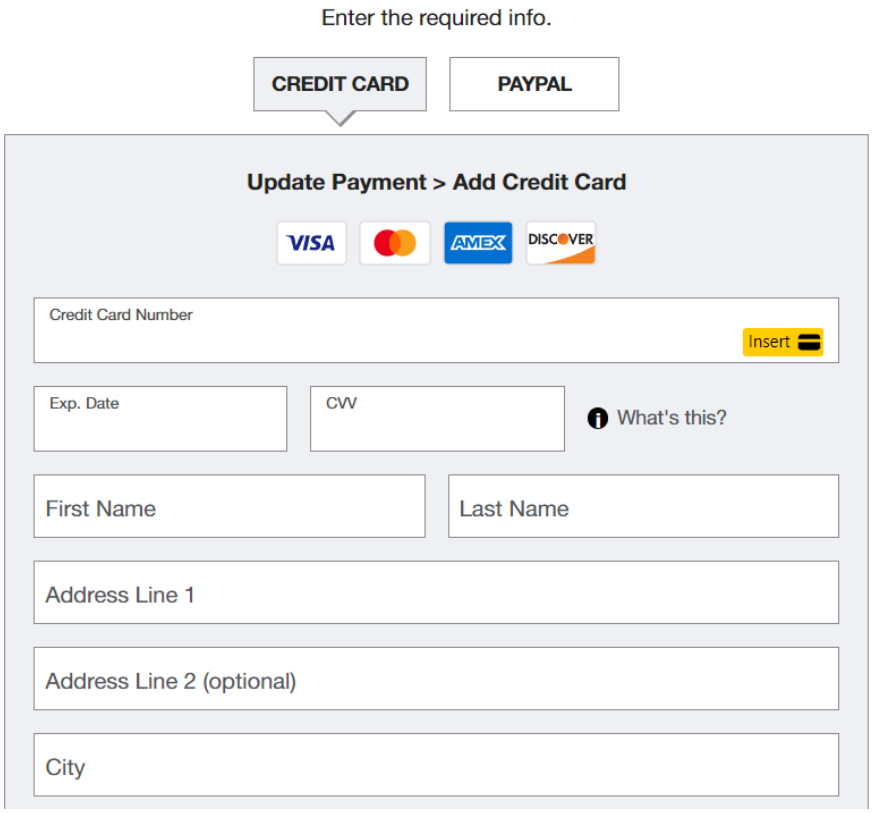
```
// Insert payment data into the database
stmt, err := db.Prepare("INSERT INTO payments (amount, currency, customer_id, product_id, status) VALUES (?, ?, ?, ?, ?)")
if err != nil {
    return err
}
defer stmt.Close()

_, err = stmt.Exec(amount, currency, customerID, productID, status)
if err != nil {
    return err
}

return nil
```

```go
func storePayment(amount int, currency string, customerID int, productID int, status string) error {
    // Connect to the MySQL database
    db, err := sql.Open("mysql", "username:password@tcp(127.0.0.1:3306)/wine_store")
    if err != nil {
        return err
    }
    defer db.Close()
```

the function storePayment takes in the payment data as parameters and inserts it into the payments table of the wine_store database. The sql.Open function is used to establish a connection to the MySQL database, and the sql.Prepare and stmt.Exec functions are used to prepare and execute the SQL INSERT statement.



I followed compliance with PCI-DSS regulations to ensure the security of payment data. Additionally, I used AWS hosting platform to host my app and Stripe integration.