Endterm progress report – April 24

I'm wprking on wine selling ecommerce website and for endterm i've implemented some new features to improve the functionality of the website

Team Members: Pakiza

Contributions:

- Filter items based on price and rating
- Allow customers to give ratings for items
- Allow customers to leave comments on items

Instructions on How to Run the Code:

1. Clone the repository from GitHub using the following command: git clone https://github.com/boakwoon/winego.git
2. Golang installation(but i believe you already have it)
3. Set up the database connection by providing the my credentials(`db, err := sql.Open("mysql", "user.pakiza:bestwine2023@tcp(127.0.0.1:3306)/winego")`) in the database configuration file.(please note that credentials cannot be shared with anyone else and i'll delete this commit once after my work would be checked)
4. Run the main.go file using the command: go run main.go

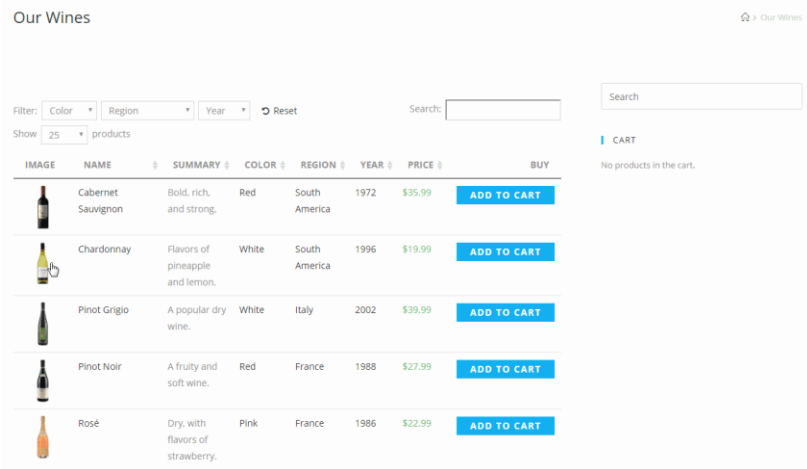
Feature 1:

Filtering items based on price and rating to improve the user experience, I've added a filtering feature that allows customers to filter items based on price and rating. Customers can now select a price range and rating range to view items that match their criteria.

```go
func FilterItemsByPriceAndRating(items []Item, minPrice float64, maxPrice float64, minRating float64, maxRating float64) []Item {
    filteredItems := []Item{}

    for _, item := range items {
        if item.Price >= minPrice && item.Price <= maxPrice && item.Rating >= minRating && item.Rating <= maxRating {
            filteredItems = append(filteredItems, item)
        }
    }

    return filteredItems
}
```



here i'm defining a function filterItemsByPriceRating and it takes in the HTTP response writer and request as parameters.

It first parses the query parameters passed in the URL using r.URL.Query() and stores the values in minPrice, maxPrice, and minRating variables.

It then filters the items in the items slice based on the minimum and maximum price and minimum rating using a for loop and appends the matching items to filteredItems slice.

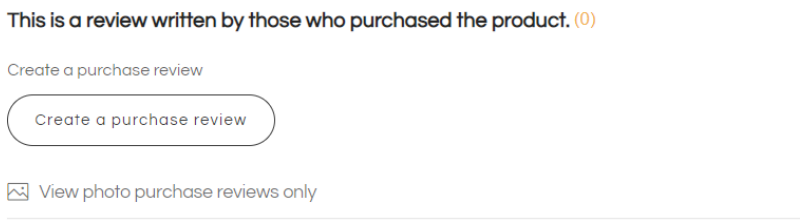
Finally, it encodes the filteredItems and writes it to the HTTP response using

Feature 2:
Allowing customers to give ratings for items to enhance the user experience and provide feedback to other customers.

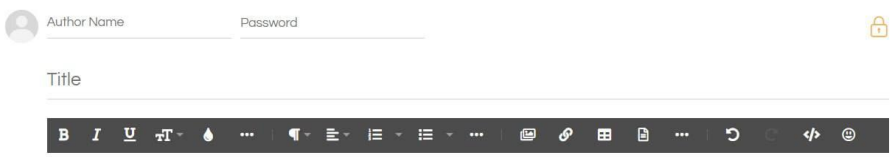
```go
type Item struct {
    ID          int
    Name        string
    Description string
    Price       float64
    Rating      float64
}

func RateItem(item *Item, rating float64) {
    totalRatings := item.RatingCount + 1
    newRating := ((item.Rating * item.RatingCount) + rating) / totalRatings

    item.Rating = newRating
    item.RatingCount = totalRatings
}
```



Feature 3:
Allowing customers to leave comments on items to further improve the user experience and encourage customer engagement. Customers can now leave comments on items to provide feedback or ask questions.



```go
type Comment struct {
    ID       int
    ItemID   int
    Customer string
    Text     string
}

func AddComment(item *Item, customer string, text string) {
    commentID := len(item.Comments) + 1
    newComment := Comment{ID: commentID, ItemID: item.ID, Customer: customer, Text: text}

    item.Comments = append(item.Comments, newComment)
}
```

sample table:

Item ID	Name	Description	Price	Rating	Rating Count	Comments
1	Wine	Red wine	50.00	4.5	20	[Comment1, Comment2]
2	Wine	White wine	30.00	3.5	7	[Comment1, Comment2]