Team: NDY corp.

Progress Report 7 **Date: April 06, 2023**

		
Amangeldin Erasyl	**Amanzhol Nursultan**	**Shulanbay Darkhan**
id:200103252	**id:200103444**	**id:200103038**

In this report we added:

5) Giving rating for items

6) Commenting items

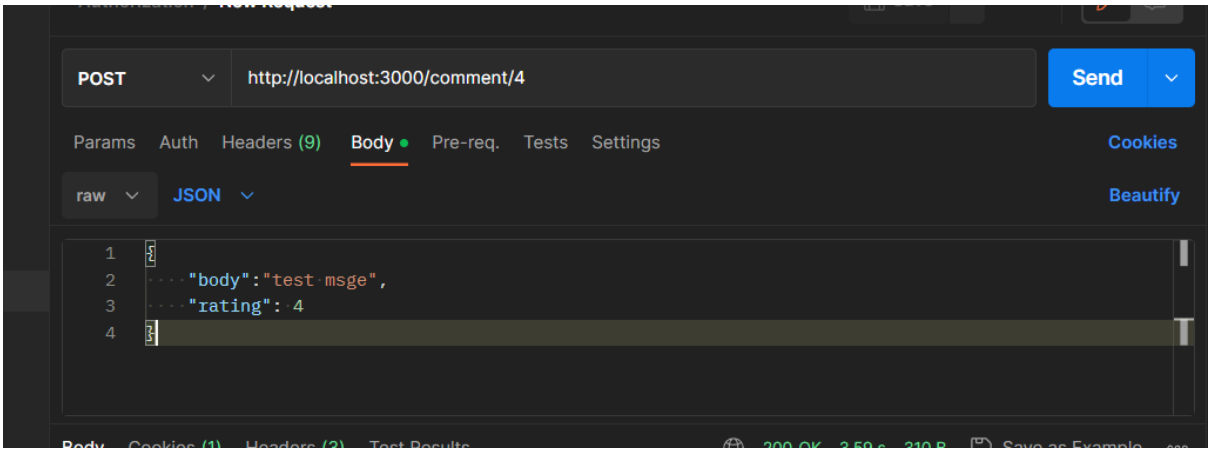
1)Here is our input of comment:

```
    "github.com/ndy-corp/1.src/midterm-1/src-code-nev
    "github.com/ndy-corp/1.src/midterm-1/src-code-nev
)

func CreateComment(c *gin.Context) {
    var body struct {
        Body   string
        Rating int
    }
    params := c.Params

    id, err := strconv.ParseInt(params.ByName("id"),
    if err != nil || id < 1 {
        return
    }

    if c.Bind(&body) != nil {
```

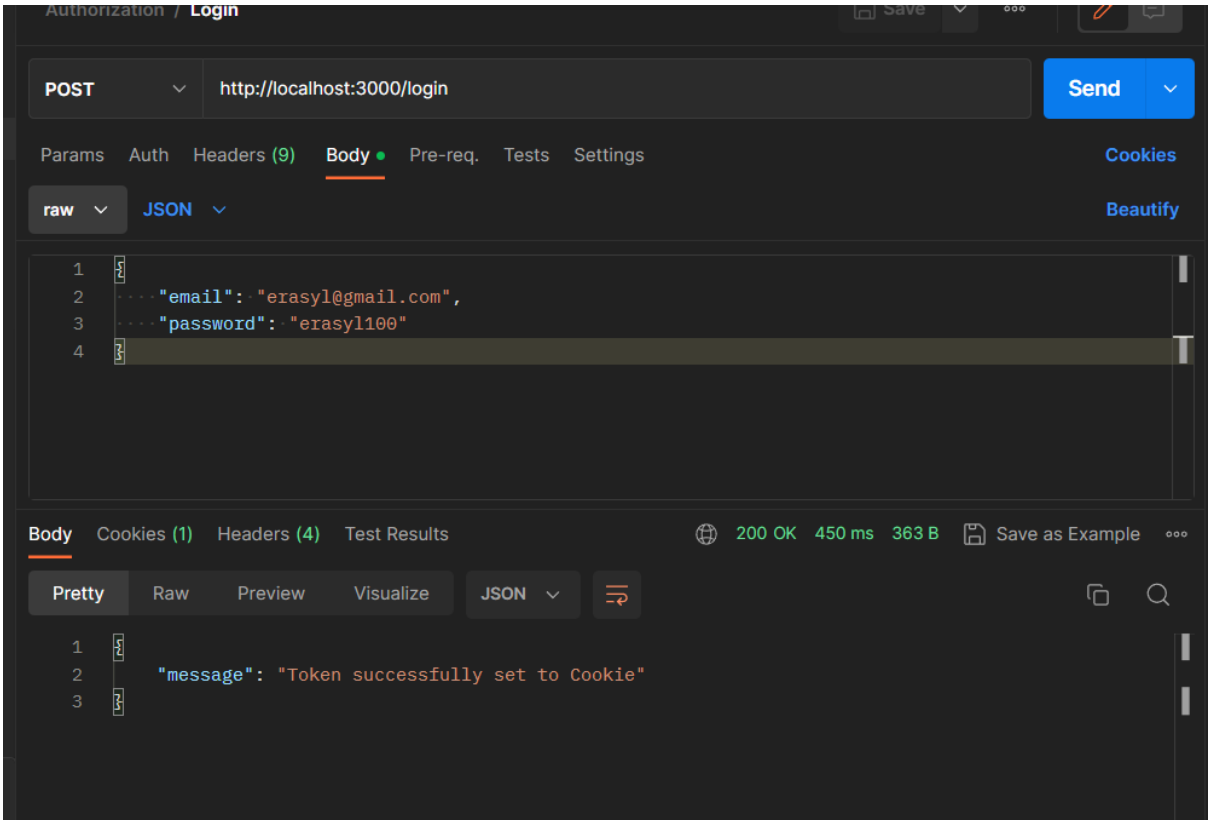
2)For example:

*https://localhost:3000/comment /**4***

here 4 means our products id.



3)And also we should be authorized (login).



4)Here we checking our products id:

```
    }
    params := c.Params

    id, err := strconv.ParseInt(params.ByName("id"), 10, 64)
    if err != nil || id < 1 {
        return
    }
```

5)Here our *BIND,* which input is put to the body. If it's incorrect, it will show an error message.

```
    if c.Bind(&body) != nil {
        c.JSON(http.StatusBadRequest, gin.H{
            "error": "Failed to read body",
        })
        return
    }
    user, _ := c.Get("user")
```

6)Here it is taking our user from cookie, and making a user model. Also checking whether it's correct or not.

```
    }
    user, _ := c.Get("user")

    usr, ok := user.(models.User)
    if !ok {
        c.JSON(http.StatusBadRequest, gin.H{
            "error": "Failed to assertion to type",
        })
        return
    }
    //usr := &models.User{}
```

7)Here we're finding our product by id,
then creating variables. After that, *We created a comment.*
After that we enter our data into the database

```go
    }
    //usr := &models.User{}
    var product models.Product
    initializers.DB.First(&product, "id = ?", id)
    var sum float32
    var cnt float32
    comment := models.Comment{Body: body.Body, Rating: body.Rating, User: usr.ID, Product: int(id)}

    result := initializers.DB.Create(&comment)
    if result.Error != nil {
        c.JSON(http.StatusBadRequest, gin.H{
            "error": "Failed to create comment",
        })

        return
    }
```

8)After the product rating has changed, we find the sum of all ratings and the number of all and find the average.
Then we're updating our rating.
And then just sending our comment.

```go
    initializers.DB.Table("comments").Where("product = ?", id).Select("sum(rating) as sm").Scan(&sum)
    initializers.DB.Table("comments").Where("product = ?", id).Select("count(*) as cnt").Scan(&cnt)
    avg := sum / cnt
    initializers.DB.Model(&product).Update("rating", avg)

    c.JSON(http.StatusOK, gin.H{"comment": comment})
}
```

```go
You, 32 minutes ago | 2 authors (You and others)
type Comment struct {
    gorm.Model
    Body    string `json:"body"`
    User    uint   `json:"user"`
    Rating  int    `json:"rating"`
    Product int    `json:"product"`
}
```

Created model for *Comment.*

```go
type Product struct {
    gorm.Model
    Title       string  `json:"title"`
    Description string  `json:"description"`
    Rating      float32 `json:"rating"`
}
```

Here we just added *Rating.*

And here our migrations.

```go
func SyncDatabase() {
    DB.AutoMigrate(&models.User{}, &models.Product{}, &models.Comment{})
}
```