Software Engineering: 14:332:452:01
Group 1
Home Kitchen Automation

https://github.com/max-legrand/chefcart

https://chefcart.herokuapp.com

Technical Documentation
1 April 2021

Team Members:

- Indrasish Moitra
- Milos Seskar
- Shreyas Heragu
- Kevin Lin
- Maxwell Legrand
- Jonathan Wong
- Allen Chang
- Elysia Heah
- Mark Stanik
- Brandon Luong

Table of Contents

User Models

User struct represents individual user object and contains the following:

- Email as String
- Password as String

UserInfo struct maintains additional user information found in Account Details and stores the following:

- ID as int
- City as string
- State as string
- Diets as StringArray
- Intolerances as StringArray
- QuantityThreshold as float 64

Ingredient struct represents a singular pantry ingredient and stores the following:

- UID as uint
- Name as string
- Quantity as string
- Weight as string
- Volume as string
- Expiration as string
- ImageLink as string

func ConnectDB() to connects to the database and updates the global database variable

Login & Login Services

type LoginCredentials struct is a struct to hold the email and passwords inputted by the user:
Email as string
Password as string

type LoginController interface is a struct that contains:
Login(ctx *gin.Context) as string

type loginController struct is a struct that contains:
service.JWTService as a JSon web token service

func LoginHandler(jWtService service.JWTService) starts a JSon web service

*func Login(controller *loginController)* authenticates the password and email fields from LoginCredentials and returns a json web token if user credentials are valid, and an empty string if invalid

type LoginService interface is an interface that contains:
The return value of LoginUser() as a boolean (checks authentication to see if user is logged in or not)

func LoginUser(email string, password string) takes in an email string and password string, checks if a user with a corresponding email and password are contained in the database, and returns: true and the user id if they exist or false, 0 if they do not.

Webserver

Package webapp
Runs webserver and displays content

func NewServer() starts a new server with Server struct

func AuthUser() verifies user from encrypted JSON web token

func GetPantry() gets pantry items for a valid user
Each pantry item is expressed as an Ingredient struct (see page 3)

func GetUserInfo() gets extra user info for a given user
Extra info is expressed as an UserInfo struct (see page 3)

Launch Server and User Functions

func LaunchServer() launches the server

*router.GET("/", func(c *gin.Context)* gets the base index page

*router.GET("/notfound/:type", func(c *gin.Context)* presents the not found page which occurs on invalid login or signup

*router.GET("/signup", func(c *gin.Context)* presents the signup form

*router.GET("/login", func(c *gin.Context)* presents the login form

*router.POST("/signup_user", func(c *gin.Context)* applies signup user logic. If signup logic fails, redirect to /notfound/signup (see page 6)

*router.POST("/login_user", func(c *gin.Context)* applies login user logic. If token generated is invalid (“”) then redirect to /notfound/login (see page 6)

*router.GET("/useredit", func(c *gin.Context)* displays user edit form

*router.POST("/edit_user", func(c *gin.Context)* performs user edit logic

*router.GET("/logout", func(c *gin.Context)* logs the user out. Deletes the token cookie and redirects back to the index page.

*router.GET("/pantry", func(c *gin.Context)* performs digital pantry logic. Deletes invalid ingredient entries with store.Delete()

*router.GET("/recipe", func(c *gin.Context)* displays recipe request form. Grabs diets and intolerances from userinfo.Diets and userinfo.Intolerances respectively.

*router.POST("/recipeSearch", func(c *gin.Context)* performs recipe search

myForm struct represents a singular recipe search which contains the following:

- Ingredients as a string array
- AdditionalIngredients as a string
- Diets as a string array
- Intolerances as a string array
- Cuisine as a string array

recipe struct represents a singular recipe that is displayed in a list. It contains the following fields:

- Name as a string

ID as a string
Used as a string
Missing as a string
ImageLink as a string

The following url is what is used to retrieve recipes in an API call:

url := "https://api.spoonacular.com/recipes/complexSearch?intolerances=" + strings.Join(formData.Intolerances, ",") + "&includeIngredients=" + ingredients + "&number=10&offset=" + strconv.Itoa(offset) + "&diet=" + strings.Join(formData.Diets, ",") + "&cuisine=" + strings.Join(formData.Cuisine, ",") + "&apiKey=" + os.Getenv("APIKEY")

*router.GET("/additem", func(c *gin.Context)* displays add item form

*router.POST("/.additem", func(c *gin.Context)* performs add item logic

The following url is what is used to check valid ingredients in an API call:
url := "https://spoonacular.com/api/tagFoods"

models.DB.Create() is used to create a new Ingredient struct (see page 3)

*router.GET("/edit/:id", func(c *gin.Context)* displays the edit pantry item form

*router.GET("/delete/:id", func(c *gin.Context)* is used to delete a pantry item

*router.POST("/edit/:id", func(c *gin.Context)* performs item edit logic

The following is used to update and save an item's Ingredient struct field:

```
pantry.Name = name
pantry.Quantity = quantity
pantry.Volume = volume
pantry.Weight = weight
pantry.ImageLink = image
pantry.Expiration = date
models.DB.Save(&pantry)
```

func invalidDate() determines if the date is in a valid format

func getLoginToken() gets the login token from a gin context and verify its integrity

Middleware

Utilized to encrypt and decrypt session tokens (primarily strings) and verify that the JWT (JSON web token) is still valid.

func Decrypt(encryptedString string) takes an encrypted token string as a parameter and returns ta JWT token

func Encrypt(stringToEncrypt string) will encrypt a jwt token into a new token

func ValidToken(c *gin.Context) checks whether the token from a cookie is valid. It takes in a gin context pointer and if the token is valid, it will return it.

func ValidTokenGRPC(tokenInput *Tokens.token) is identical in function to ValidToken but it takes in a protobuf Token object as its input instead.