Software Design Specification

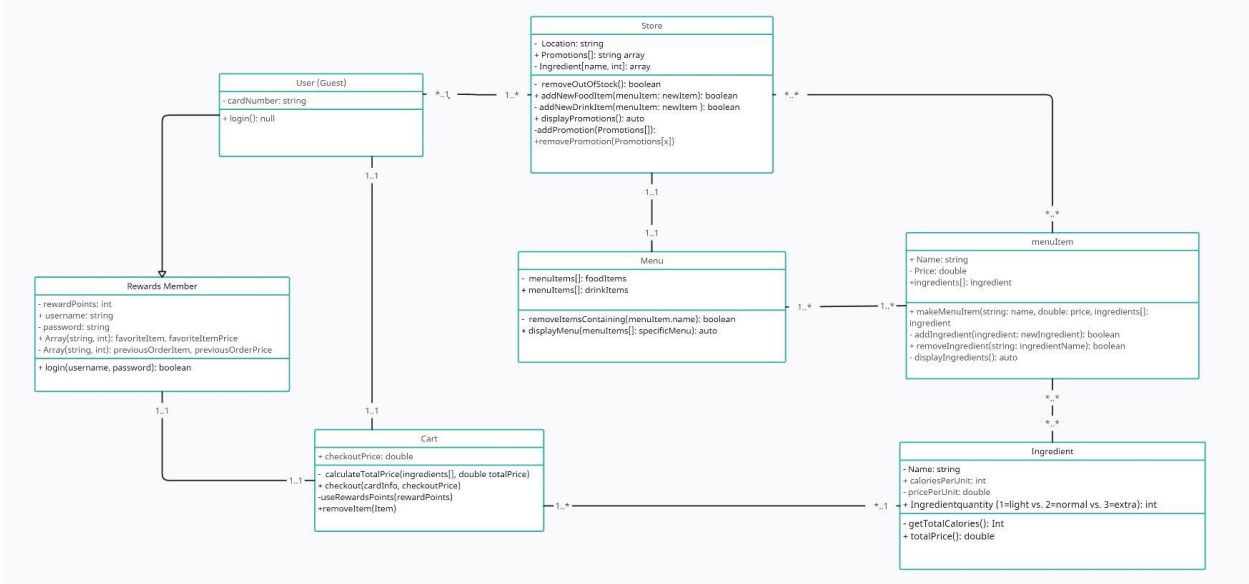
by Chaz Gabelman, Eric Soto,
Gerardo Reza, and Luke Patterson
(Group 9)

Software for a Cafe

Overview

This software is for customers to be able to use a guest or rewards account to order items from a Cafe. Users are able to add existing items to their cart, or customize their own items using the Cafe's listed ingredients. The software is also capable of distinguishing which items are in stock depending on the location they are ordering from. There is a menu page listing all items and ingredients for customers to add to their shopping cart, a shopping cart page to review your order and remove any items you don't want anymore, and a payment page to input name/card information and process payment. The software also contains a rewards system that takes in how many points members have accumulated, and allows them to use their points in the checkout page.

Classes



Class Descriptions

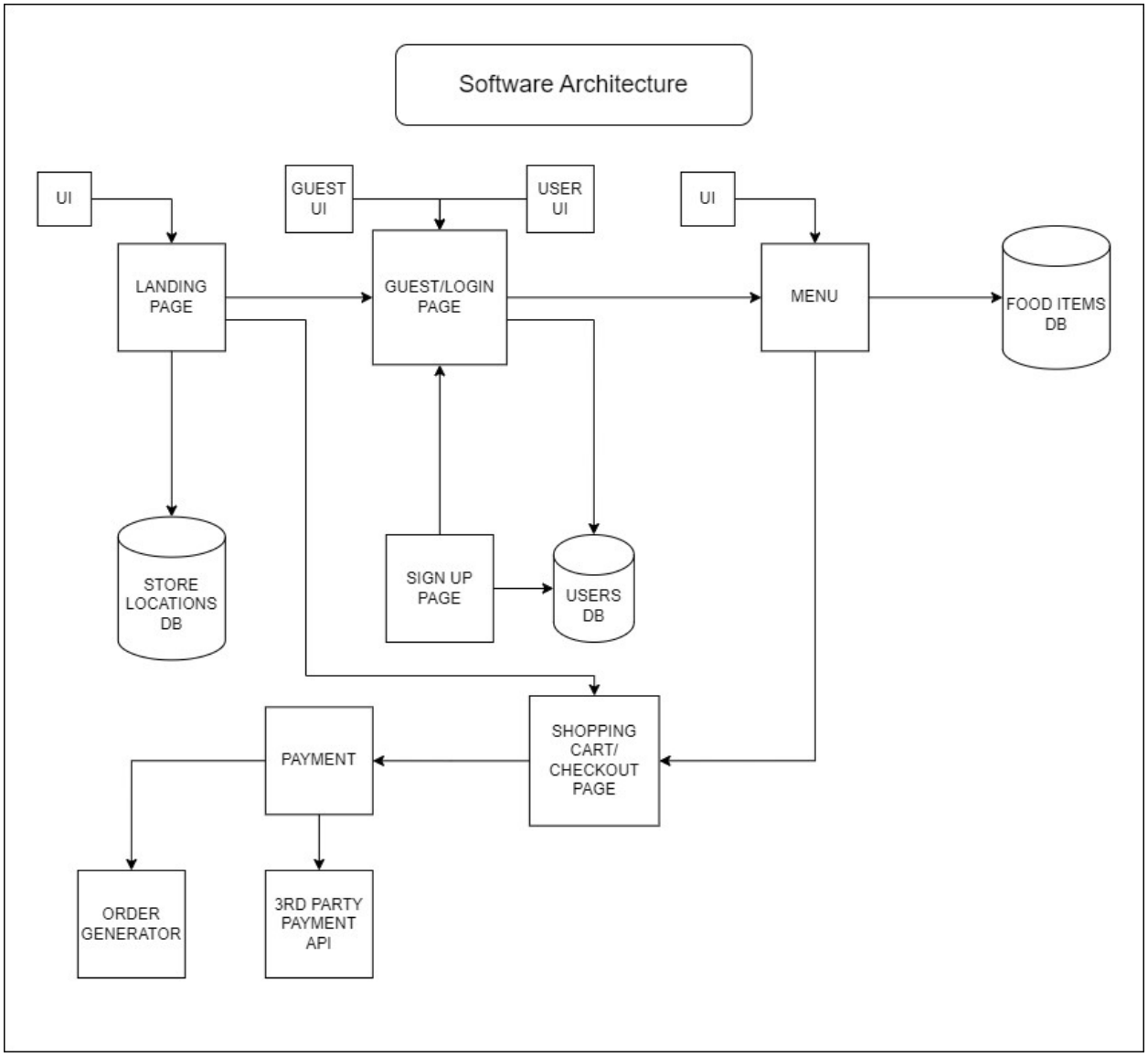
- User (Guest)
    - String: cardNumber
    - makePayment(String: cardNumber):boolean
- Rewards member subclass
    - Int: rewardPoints
    - String : "username"
    - String : "password"
    - Array[string, int] : [favorite item, favorite item price]

        - Array[string, int]: [previous order item, previous order price]
        - login(username, password):boolean

- User (Guest) ○ Attributes:
        - Has a string type attribute for Card Number, which would be sent to the Cart class in order to complete the checkout() method.
    - Methods:
        - Has a Login() method with no parameter to automatically login a guest
- Rewards Member
        - Subclass of the User class, inherits attributes and methods from User class ○ Attributes:
        - Has integer type attribute for RewardsPoints, calculates how many points the rewards member has for discounts
        - Has a string type attribute for Username and Password, used for logging into their rewards account
        - Has an array that takes in a string and integer for each element, string representing their favorite item, integer representing the price correlating to that item
        - Has another array that takes in a string and integer for each element, string representing their previous order item, and integer representing the price correlating to that item
    - Methods:
        - Has a login method, that takes in the Username and Password attributes as parameters, uses them to verify the rewards members account and login
- Store
    - Attributes:
        - Has string type attribute for Location, used for distinguishing between different store locations
        - Has an array of string type, used for storing relevant promotions
        - Has integer type attribute for Quantity, representing the amount of an ingredient left
    - Methods:
        - RemoveOutOfStock(MenuItem) to remove item from availability on ordering system
        - AddNewFoodItem() takes in a string and adds it to the MenuItem class, returns true if successful
        - AddNewDrinkItem() takes in a string and adds it to the MenuItem class, returns true if successful
        - DisplayPromotions() displays promotions from the promotions array
- Menu
    - Attributes:

        - Array of strings which stores all food items
    - Array of strings which stores all drink items ○ Methods:
        - RemoveItemsContaining() takes in ingredientName string and removes it from store availability
        - DisplayMenu() takes MenuItems array from MenuItem class and displays it
- Menu Item
    - Attributes:
        - Has a string attribute for Name of item
        - Has a double attribute for item's price
        - Has a string type ingredients array for storing all possible ingredients
    - Methods:
        - makeMenuItem() used for creating your own item from list of ingredients and displays price by combining prices from the correlating price
        - AddIngredient() takes in ingredient from ingredient array, used within makeMenuItem() method
        - RemoveIngredient() used to remove ingredient from customized order, in the makeMenuItem() method
        - DisplayIngredients() method is used to show all ingredients in the item you have created.
- Ingredient ○ Attributes:
        - Has a string attribute for the Name of the ingredient
        - Has an integer attribute for caloriesPerUnit of an ingredient
        - Has a double attribute for the pricePerUnit of an ingredient
    - Methods:
        - makeIngredient(String: name, int: quantity, int: caloriesPerUnit)
        - quantity () specifies how much of an ingredient they want in their item, 1 for light, 2 for regular, 3 for extra
        - getTotalCalories() provides an integer showing how many calories the specified item contains
        - totalPrice() displays the double, which is the price of the specified item
- Cart
    - Attributes:
        - A double variable that is the total price at checkout
    - Methods:
        - calculateTotalPrice(ingredients[], double totalPrice) receives an ingredients array and the total price
        - checkOut(cardInfo, checkoutPrice) receives the user's card information and the checkout price to charge the customer for their order
        - UseRewardsPoints() allows user to apply a discount to their price prior to checkout, depending on the amount of points they have accumulated

Software Architecture / Component Connections

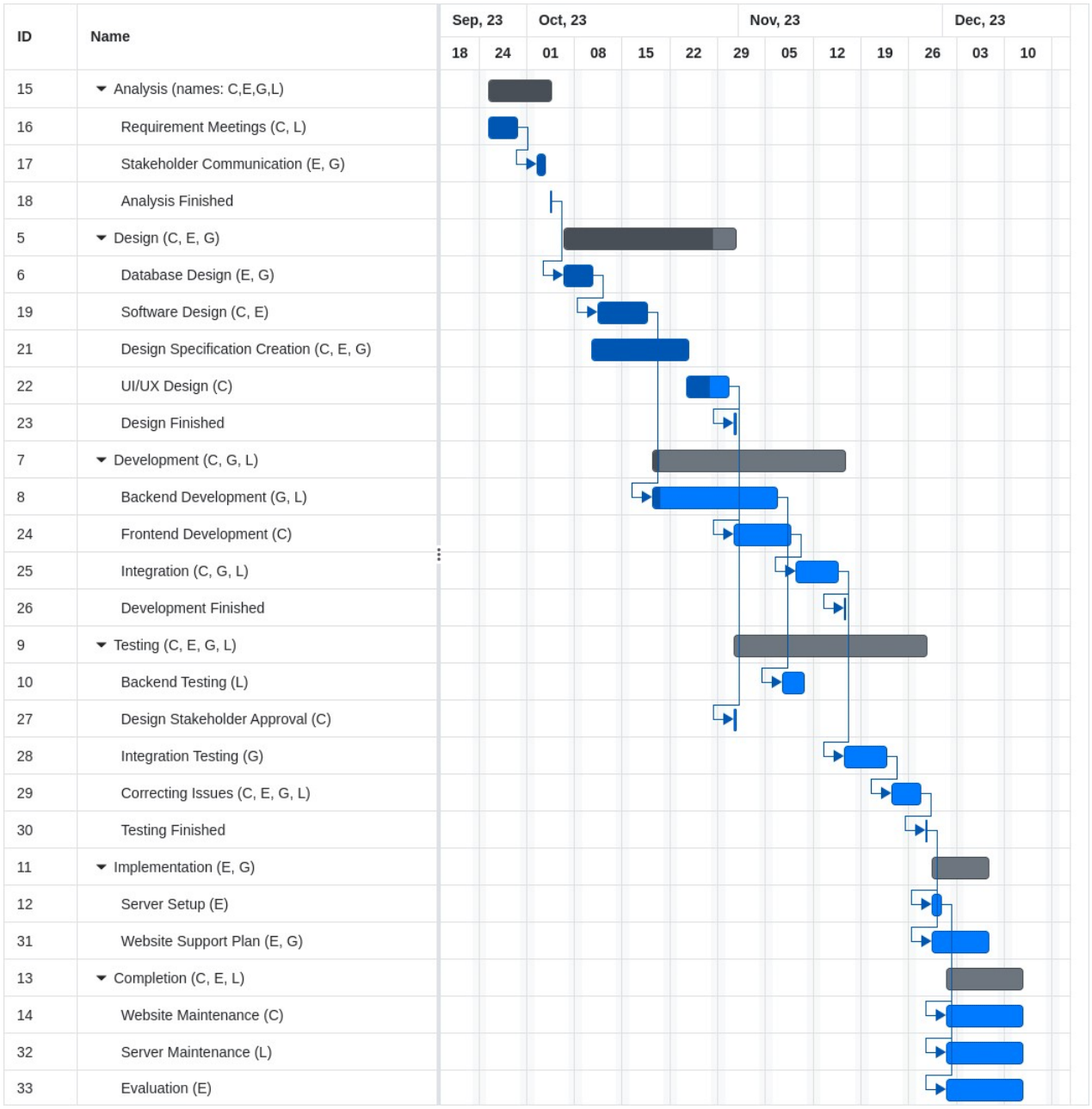


- The **landing page** is connected to a **store location DB and guest/login page.**
- The **guest/login page** is connected to **sign up page and users DB**.
- The **menu** connects to **food items DB and to shopping cart/checkout page** since you are adding menu items.
- The **landing page** would be connected to a **shopping cart/checkout page** if the user decides to access it from the home page.
- The **shopping cart/checkout page** would be connected to a **payment page** where the guest/user enters or uses saved payment information.
- The **payment page** connects to a **third party payment API** for processing payments.
- The **payment page** would also be connected to an **order generator** that sends an email receipt and order number.

Development Plan + Timeline

*(**please note**: Exporting the GANTT file to png/pdf collapsed several of the columns containing information including the names of those responsible, the expected start and stop date for each item, the duration of each item, and the progress on each item. The first initial of those responsible for each task was included following the name of*

each item (C=Chaz, E=Eric, G=Gerardo, L=Luke). The progress is represented by the portion of the bar being darkened. The full GANTT file is included in the repository.)



Data Management Strategy:
Mysql probably

Databases needed:

- Users database, to keep their information private and limit hacking vulnerability
- Menu Items database for all store related data including menu items, prices, etc.

Menu Items Database table:

- Menu meals table: stores the items we sell like drinks or dishes, and would access the “menu item” table through a foreign key

Primary key: ID: (Int)	Item Name: (VARCHAR)	Price: (Int)	Calories: (Int)	Stock: (Int)	Ingredients: (VARCHAR, ingredients separated by comma for parsing)
1	Vanilla Latte	4.99	200	50	Coffee, Cream, Vanilla Syrup
2					
3					
etc					

User Database tables:

- User table:
    - Username, Password, Location, card number (encrypted)

Primary Key: UserId: (Int)	Username: (VARCHAR, encrypted)	Password: (VARCHAR, encrypted)	Card Number: (Int, encrypted)
1	StoreEnjoyer12	goodpass145!	2134857384294

- Users Items table:

Primary Key Foreign Key: References UserId in User table (Int)	Foreign Key: MenuItemID: References Menu Item table (Int)
1	2