Comprehensive Exercise Report

Team7 DIP392-burger_town of Section 007

Sujai Shanmagam - 231ADB016
Davyd Bogomolov - 221ADB232
Gasim Jabrayilov - 231ADB015
Abbas Rzayev - 211ADB023
Arsenii Prykhodko - 221ADB089
Rufi Aliyev - 221ADB153

Requirements/Analysis **1**
Journal 1
Software Requirements 3
Black-Box Testing **5**
Journal 5
Black-box Test Cases 7
Design **9**
Journal 9
Software Design 11
Implementation **12**
Journal 12
Implementation Details 13
Testing **13**
Journal 13
Testing Details 15
Presentation **17**
Preparation 17

Requirements/Analysis

Week 2

Journal

- After reading the client's brief (possibly incomplete description), write one sentence that describes the project (expected software) and list the already known requirements.
    - It is a restaurant website where customers can scan a QR code on the table and place an order.
        - It should allow customers to scan a QR code to access the menu.
        - It must have all the food items with the correct cost(including tax).
        - It should allow customers to place orders directly from the website.
        - A simple, smart , easy user interface for customers to access.
        - The website should be responsive(for mobile, tables, ipad etc..)
        - The restaurant workers (Chef, Staff) should receive orders in real time.

After reading the client's brief (possibly incomplete description), what questions do you have for the client? Are there any pieces that are unclear? After you have a list of questions, raise your hand and ask the client (your instructor) the questions; make sure to document his/her answers.

- Should the customer create an account to access the website or should it support the guest checkout?
  Answer: The website should allow both guest checkout and account creation which is optional for customers to use it and place order through it but both is fine.
- Should we include an admin panel for updating the menu items?
  Answer: Yes, you should include an admin panel for restaurant staff to update the menu items(foods) and price(cost).
- Do you want customer rating and also a feedback system for foods?
  Answer: Yes, I want it but if it's a simple rating system then it's no problem but it should include places like 1- 5 stars for food rating and for feedback it should contain around 50 words to express their feedback.
- Should the website support payment options/methods for payment?
  Answer: Not really , because if we add the payment option for payment then maybe the customer payment will fail or get any problem (incase) and also they will use different types of payment options so it will cost extra charge from the bank for us. So it's not needed right now, maybe we will add in future.
- Will the website support order tracking of food for customers?
  Answer: Yes, customers should see order status updates like preparing, ready , completed and maybe expected timing also good.

- Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.
    - QR Code Generation
      Understand and Read about QR code API for dynamic menu links.
    - Real time Order Management
      Explored WebSocket Implementation for live order updates for our website.
    - Payment Gateway Integration
      Research Paypal API documentation for future changes in the website.

- Describe the users of this software (e.g., small child, high school teacher who is taking attendance).
    - Customers who visit restaurant for food.
    - Restaurant Staff - who works in the kitchen and receptionist for managing orders.
    - Admin/Owner/Manager - who works in a restaurant for updating menu , managing restaurant settings.
- Describe how each user would interact with the software
    - Customer : Scan QR code , See menu , Place order with their name & table number and Track order status.(Can do without login)
    - Restaurant Staff : Receive new orders , Update order status and Notify customers when it's ready.
    - Admin/Owner/Manager : Login through their Id,password , Adjust prices & foods and manage restaurant settings.
- What features must the software have? What should the users be able to do?
    - OR code based menu access.
    - Online Order system.
    - Real time order notification for staff.
    - UI for mobile (simple and friendly).
    - Admin panel for menu updations.
- Other notes:
    - Support Multi language support other than english and latvian.
    - Special offers for frequent customers like points(if they login using their own account , not as a guest)

Software Requirements

Description of the Project :

The Burger Town restaurant website is designed to provide a good dining experience by allowing the customers to access the menu, place their orders using a QR code placed on their table.By scanning the QR code with their mobile , or searching the website link with their other devices like tablet or ipad , Customers are directed to the website and then they can browse the menu , customize their orders and place it. The website system will also allow the restaurant staff to receive & manage orders in real time. An admin panel will be available for restaurant managers/Admin/Owner to update the menu, manage order and adjust prices & cost and also settings as needed.

List of requirements:

➔ Functional requirements :

1. QR code menu access - The QR code must be dynamic, allowing for quick redirection to the menu for ordering.
2. Order Placement - The System should allow customers to modify their order before confirming and completing the purchase.
3. Order Notification - It must allow the staff to make the order as “In Progress” , “Ready” or “Order ready”.
4. Order Status - Customers should be able to track the status of their order.

➔ External Requirements :

1. Mobile Compatibility - The website must be responsive and optimized for use on smartphone and ipad or tablets. It should work in different mobile browsers (Chrome , safari , etc…)
2. Order Processing - The system must interact with the point of sale system to make sure orders are transmitted correctly to the kitsch staff.
3. QR code printing & generation - It must be generated dynamically based on the table. The system must interface with a QR code generation tool and make sure the proper printing of codes for tables.

➔ System Features :

1. Menu Management - Admin can update the menu like *Add new items , modify the prices & images and update descriptions* through an easy admin Interface. It should support categories for menu items for instance burgers , drinks , fries etc..
2. Order Placement & Customization - Customers can see the menu, add items to the cart and customize their orders (adding sauces , extra chicken , removing tomato , choosing toppings etc..). They can modify their orders before finalizing the purchase.
3. Real time Order Management - It should provide real time updates to restaurant staff when new orders are placed by customers. Staff can update the order status example, In progress, Ready , Completed and the status will be visible to the customers.
4. Order Tracking - Customers can track the progress of their orders through the website interface like (preparing, ready etc..)

➔ Non Functional Requirements :

1. Performance:
    - It should load within 3 to 4 seconds maximum on mobile devices.
    - It should be able to handle a maximum of 50 orders at the same time without lags in order processing.
2. Safety & Security:
    - Customer/Admin Datas must be protected and secure with strong encryption.
    - It must work with data protection regulations such as GDPR or CCPA.
    - The age of the user must be 12+ years(can understand the functionality).
3. Quality & Usability:
    - It should be user friendly for customers , restaurant staff and admins.
    - There should not be more than 3 steps between scanning the QR code and placing an order.
    - The language should be in English & Latvian and it may support multiple languages.
4. Reliability & Maintainability:
    - It should have an uptime of 95% with minimal downtime for maintenance.
    - It should be easy to maintain and update.

User Stories:

❖ Customer:

➢ Function : Scan QR code , browse menu and place order.

➢ Why this functionality : To quickly see the menu and order food without waiting for a server.

❖ Restaurant Staff:

➢ Function : Receive real time orders from customers and update order status

➢ Why this functionality : To manage orders and inform customers when their food is ready.

❖ Admin/Manager:

➢ Function : Update menu, edit cost & images and monitor sales.

➢ Why this functionality : To keep the menu up to date and track order activity for restaurant management.

Black-Box Testing

Instructions: Week 4

Journal

The following prompts are meant to aid your thought process as you complete the black box testing portion of this exercise. Please review your list of requirements and respond to each of the prompts below. Feel free to add additional notes.

- What does input for the software look like (e.g., what type of data, how many pieces of data)?
    - QR code for each table which is code that redirects the customers to the menu ,(1 piece per table).
    - Menu items data for food items, including names prices and description like burger with size,(up to the number of items in the restaurant's menu)
    - Order details data on selected items , quantity , special request like customization, (1 per order, with multiple items)
- What does output for the software look like (e.g., what type of data, how many pieces of data)?
    - Order confirmation message with the order number and estimated time for preparation, (1 piece per order)
    - Receipt details including the list of items ordered which means the total cost for payment , (1 piece per order)
    - Order status updates on the progress of the order like preparing , ready etc.., (1 per order , updated throughout the process)
    - For admin panel , it will display of current orders , statuses as a output and ability to modify the menu and also setting (multiple items like orders , status and payment details as a piece of data)
- What equivalence classes can the input be broken into?
    Valid and Invalid classes depends on the input , we clearly mentioned everything in the next line,
    - QR code input: Valid QR code(redirects to the correct menu).
    - Menu selection: Valid menu items(available items) , Invalid menu items (out of stock , removed from menu)
    - Order details: Valid order data (correct customization, quantity and payment details), Invalid order data (incorrect item quantity)
- What boundary values exist for the input?
    - QR code for table 1 to maximum number of tables available.(We think 10 tables)
    - Minimum order selection will be 1 item and maximum will be dependent which can handle in one order.
    - Customization like adding something will be the maximum value that exists for input and also minimum will be no customization.
    - Small amount like a drink is starting min and max payment details will be big meal or combination of items.

- Are there other cases that must be tested to test all requirements?
    - Multiple customers ordering at the same time must be tested to make sure the system can handle multiple orders at the same time.

    - Change or cancel the order before providing to the chef must be tested.
    - Should Test some cases like network failure, database failure.
    - Admin panel actions like update the menu or process order during the peak time must be tested
- Other notes:
    - UI responsiveness and correct error handling are a little hard to test because black box texting only focuses on functionality and user interaction.

Black-box Test Cases

Use your notes from above to complete the black-box test plan section of the formal documentation by writing black box test cases (other than actual results since no program currently exists). Remember to test each equivalence class, boundary value, and requirement.

Test ID	**Description**	**Expected Results**	**Actual Results**
T1	Test valid QR code scanning	The system should direct the customer to the restaurant's website menu.	Pass - qr code successfully redirected customer to menu page with correct table context
T2	Test valid menu item selection	The system should allow the user to add the items to their cart and proceed to checkout.	Pass - menu items added to cart successfully, checkout button enabled
T3	Test invalid menu item selection (out of stock items)	The system should show a message “This item is currently available and out of stock, please order different items , Have your food happily!”	Pass - error message displayed successfully
T4	Test selecting the only one item to order (whatever maybe drink or something)	The system should allow the customer to add exactly one item to cart and proceed to checkout.	Pass - single item successfully added to cart, checkout process initiated
T5	Test selecting maximum number of items allowed (15 items)	The system should allow the customer to add up to the maximum number of items to cart.	Pass - system accepted 15 items, blocked attempt to add 16th item with appropriate message
T6	Test adding special customization to an item (burger customization)	The system should allow customization(extra stuff inside burger) to be added to the item in the cart/	Pass - customization options displayed and applied correctly, price updated accordingly
T7	Test order status update (from “In progress” to “Ready”)	The system should update the order status in real time as the kitchen progresses with the order.	Partial pass - status updates work but with 2-3 second delay, not truly real-time
T8	Test order cancellation before proceed to kitchen	The system should allow customers to cancel their order before proceeding	Pass - cancellation successful within 5-minute window, refund

		to the kitchen.	processed automatically
T9	Test admin panel login and menu update	Admin should be able to login to the admin panel , update the menu and save the changes.	Pass - admin login successful, menu updates saved and reflected on customer interface immediately
T10	Test multiple customer ordering at the same time	The system should handle multiple orders simultaneously without lag and eros in processing.	Partial pass - system handled 35 concurrent orders successfully, some delays observed at 40+ concurrent users
T11	Test admin panel action during peak times (updating the menu during a rush)	Admin should still be able to update the menu without causing eros and changes should reflect immediately	Pass - menu updates successful during peak load, changes propagated to all active customer sessions

Design

Instructions: Week 6

Journal

- List the nouns from your requirements/analysis documentation.
    - Customer
    - QR Code
    - Menu
    - Menu Item
    - Order
    - Order Item
    - Customization
    - Cart
    - Table
    - Staff Member
    - Admin (Manager)
    - Category
    - Feedback / Rating
    - Notification
    - Order Status

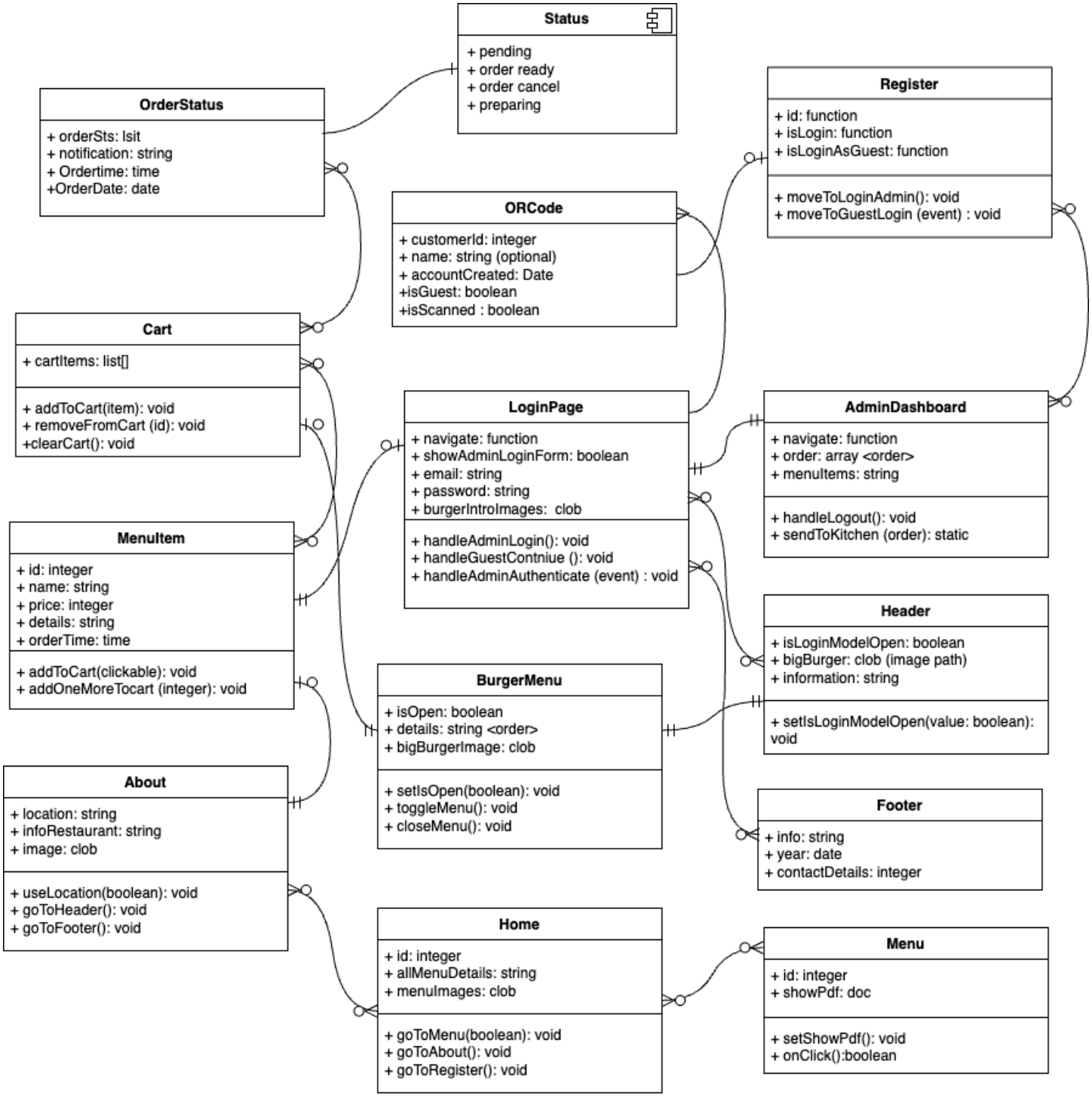
- Which nouns potentially may represent a class in your design?
    - Customer
    - QRCode
    - Menu
    - MenuItem
    - Category
    - Order
    - OrderItem
    - Customization
    - Cart
    - Table
    - StaffMember (inherits from User)
    - Admin (inherits from User)
    - Feedback
    - NotificationService
    - OrderStatus (enum)
- Which nouns potentially may represent attributes/fields in your design? Also list the class each attribute/field would be a part of.
    - Customer - customerId: String, name: String, accountCreated: Date?, isGuest: Boolean
    - QRCode - codeValue:String, tableNumber: int

    - Menu - items: List
    - MenuItem - itemId: String, name: String, description: String, pride: double, category: Category, imageUrl: String
    - Category - category_id: String, name: String
    - Order - orderId: String, customer: Customer, tableNumber: int, items: LIst, status: OrderStatus, timestamp: DateTime
    - OrderItem - manuItem: MenuItem, quantity: int, customizations: list
    - Customization: name: String, value: String
    - Cart: customer: Customer, items: List
    - Table - tableNumber: int, qrCode: QRCode
    - StaffMember - userId: String, username: String, passwordHash: String
    - Admin - userId: String, username: String, passwordHash: String
    - Feedback - feedbackId: string, customer: Customer, manuItem: MenuItem: MenuItem
- We often do not come up with the best design on our first attempt. Also consider whether any needed classes are missing. These two design options should not be GUI vs. non-GUI; instead you need to include the classes and attributes for each design. Reminder: Each design must include at least two classes that define object types.
    - Simple Domain Model
        - Classes: Customer, MenuItem, Order, OrderItem, Admin
        - Pros: Minimal set of classes; easier to implement quickly.
        - Cons: Lacks explicit support for QR codes, customization objects, feedback and separates responsibilities poorly.
    - Rich Domain Model with Service Layer
        - Classes: Customer, QRCode, Menu, MenuItem, Category, Order, OrderItem, Customization, Feedback, Table, User (abstract), StaffMember, Admin, NotificationService, OrderStatus enum
        - Pros: Clear separation of concerns; extensible (e.g., adding promotions, payment, multi-language); maps well to MVC architecture.
        - Cons: Higher initial complexity; more boilerplate code.
- Which design do you plan to use? Explain why you have chosen this design.
  We chose the Rich Domain Model with a Service Layer because it provides clear separation of concerns, supports real-time order updates, allows scalability for future enhancements like multi-language and promotions, and aligns well with MVC architecture. Although initially more complex, it allows modularity and better maintainability.
- List the verbs from your requirements/analysis documentation.
    - Access (Menu)
    - Place (Order)
    - Login (Admin Panel)
    - Modify (Menu Items)
    - Add (Items, Customizations)
    - Remove (Items)
    - Rate (Food)
    - Access (Menu)
    - Place (Order)
    - Customize (Items)
    - Receive (Orders)
    - Update (Order Status, Menu)

- Which verbs potentially may represent a method in your design? Also list the class each method would be part of.
    - Scan scanQRCode() QRCode
    - Access getMenu() MenuService
    - Customize addCustomization() OrderItem
    - Track trackOrderStatus() Customer
    - Receive receiveOrder() StaffMember
    - Rate rateFood() Feedback
- Other notes:
    - We ensured extensibility for future features such as payment integration, loyalty programs, and analytics. All user interactions (admin, customer, staff) are encapsulated within respective classes. Emphasis was placed on modularity and single-responsibility principle to simplify testing and maintenance. We plan to implement WebSocket-based real-time updates to enable instant order notifications and status changes.

Software Design

We used notes from above to complete this section of the formal documentation by planning the classes, methods, and fields that will be used in the software. Our design should include UML class diagrams along with method headers.



Implementation

Instructions: Week 8

Journal

The following prompts are meant to aid your thought process as you complete the implementation portion of this exercise. Please respond to each of the prompt below and feel free to add additional notes.

- What programming concepts from the course will you need to implement your design? Briefly explain how each will be used during implementation.

    - Object-Oriented Programming (OOP)
        - Team 7 used OOP principles to design classes such as MenuItem, Order, OrderItem, Customer, and Admin. Each class encapsulates relevant data and behavior (e.g., order status tracking, item customization, and menu management).
    - Client-Server Communication (HTTP APIs)
        - RESTful APIs were implemented using Node.js (backend) and consumed using Axios in the React frontend. These APIs enable communication between the frontend app and the backend database, like placing orders or updating menu items.
    - Frontend Development (React)
        - React was used to build dynamic UI components for the customer interface and admin dashboard.
    - State Management
        - React's state and context APIs were used to handle order details, selected items, and real-time updates across components like menu selection, order cart, order status display.
    - Authentication and Authorization
        - Basic authentication logic for admin login ensures only authorized users can modify menu items and access sensitive data.
    - Form Handling and Validation
        - Used in both customer and admin interfaces for tasks like order submission, item customization, and login validation. Prevents incomplete or invalid submissions.
    - Database Integration
        - MongoDB was used to store menu data, customer orders, and admin credentials. Mongoose was used to define schemas and manage data integrity.
    - Testing and Debugging Tools
        - Console logs, unit testing, and black-box testing were used throughout the implementation process to ensure the application logic behaves as expected.

- Other notes:
    - QR code simulation was tested using static table links for local testing.
    - Responsive design ensures mobile-first usability.
    - Admin panel features a real-time view of all current orders and menu status.
    - Code was modularized for reusability and clarity.

Implementation Details

(README for Users)

Welcome to Burger Town – Digital Restaurant Ordering Platform

This app allows in-house restaurant customers to scan a table-specific QR code, browse the menu, customize and place their order, and track it in real-time. Restaurant admins can manage menu items and monitor orders using the admin dashboard.

Customer Interface (How to Use)

1. Scan the QR Code at Your Table
    - Each table has a unique QR code. Scan it using your phone camera to access the Burger Town web app.
2. Browse the Menu
    - View items by category: Burgers, Fries, Drinks. Each item includes a name, description, price, and optional customization.
3. Add Items to Cart
    - Select your desired items. Customize if needed (e.g., add extra cheese or remove onions) and add to your order cart.
4. Place Order
    - Review your cart, make adjustments, and submit your order. Your table number is automatically included.
5. Track Your Order
    - After ordering, track your order's status: "In Progress" → "Ready" → "Completed". You may cancel before the order reaches the kitchen.

Admin Panel (For burger town Restaurant Staff)

1. Login
    - Navigate to /admin and log in with your credentials.
2. Manage Menu

    - Add new items, update prices or availability, and remove discontinued items.

3. Monitor Orders
    - View incoming orders, update statuses (e.g., mark as “Ready”), and review customer details

Technology Stack used by Team 7

- Frontend: React
- Backend: Node.js, Express.js
- Database: MongoDB with Mongoose
- Authentication: Basic login for admins
- Deployment : Localhost or hosted on platforms on Vercel (for frontend) and Railway (for backend)

Testing

Instructions: Week 10

Journal

The following prompts are meant to aid your thought process as you complete the testing portion of this exercise. Please respond to each of the prompts below and feel free to add additional notes.

- Have you changed any requirements since you completed the black box test plan? If so, list changes below and update your black-box test plan appropriately.
    - No requirements have been changed since completing the black box test plan
- List the classes of your implementation. For each class, list equivalence classes, boundary values, and paths through code that you should test.
    - MenuItem class
        - equivalence classes: available menu items, out-of-stock items, items with customization options, items without customization options, items in different categories (burgers, drinks, fries)
        - boundary values: minimum price ($0.01), maximum price (high-end items), items with no description, items with maximum description
        - paths to test: menu item display, price calculation, customization option handling
    - Order class
        - equivalence classes: valid orders (with available items), orders with customizations, cancelled orders, completed orders, orders in different statuses (in progress, ready, completed)
        - boundary values: minimum order (1 item), maximum order (15 items), orders with minimum payment amount, orders with maximum payment amount
        - paths to test: order creation, order status updates, order total calculation
    - Customer class
        - equivalence classes: valid customers (age 12+), invalid customers (under 12), customers with/without previous orders, customers using different devices (mobile, tablet, desktop)
        - boundary values: age 12 (minimum allowed age), maximum concurrent customers (50)
        - paths to test: customer session creation via qr scan, order placement workflow, order tracking, order modification before confirmation, order cancellation
    - OrderItem class
        - equivalence classes: valid order items, order items with customizations, order items with different quantities
        - boundary values: boundary values: quantity of 1 (minimum), maximum quantity per item
        - paths to test: adding items to order, applying customizations, quantity modification, price calculation with customizations, removing items from order
    - Admin class
        - equivalence classes: valid admin credentials, invalid admin credentials, different admin permission levels
        - boundary values: minimum admin privileges, maximum admin privileges
        - paths to test: admin login/logout, menu management (add/edit/delete items)

Testing Details

1. Test valid QR code scanning

This test validates that valid QR codes properly redirect customers to the restaurant's menu interface. It simulates scanning QR codes from different tables and verifies that each code successfully navigates to the correct menu page with the appropriate table context.

2. Test valid menu item selection

This test verifies that customers can successfully select available menu items and add them to their shopping cart. It tests the interaction between the Customer class and MenuItem class to ensure valid items are properly added and the checkout process can be initiated.

3. Test invalid menu item selection (out of stock items)

This test checks the system's behavior when customers attempt to select menu items that are currently out of stock. It validates that appropriate error messages are displayed and customers are guided to select alternative available items.

4. Test selecting the only one item to order (whatever maybe drink or something)

This test ensures that customers can successfully place orders with just a single item. It validates the minimum boundary condition for the Order class and confirms that the checkout process works correctly with the smallest possible order size.

5. Test selecting maximum number of items allowed (15 items)

This test validates that the system properly handles orders at the maximum limit of 15 items. It checks that the system accepts exactly 15 OrderItem objects within a single Order and prevents customers from exceeding this boundary.

6. Test adding special customization to an item (burger customization)

This test verifies that customers can successfully add customizations to menu items such as extra ingredients or special preparations. It tests the interaction between OrderItem and MenuItem classes to ensure customizations are properly applied and reflected in the final price.

7. Test order status update (from "In progress" to "Ready")

This test validates the real-time order status update functionality managed by the Admin class. It simulates admin users updating order statuses from "In Progress" to "Ready" and verifies that customers can see these updates reflected in their order tracking interface.

8. Test order cancellation before proceed to kitchen

This test ensures that customers can successfully cancel their orders before they are sent to the kitchen for preparation. It tests the Order class cancellation functionality and verifies that appropriate refunds are processed when applicable.

9. Test admin panel login and menu update

This test validates that admin users can successfully log into the administrative panel and perform menu management operations. It tests the Admin class authentication system and verifies that menu updates are properly saved and reflected across the system.

10. Test multiple customer ordering at the same time

This test simulates multiple customers placing orders simultaneously to validate the system's ability to handle concurrent operations. It tests the scalability of the Order and Customer classes under load conditions and ensures no data corruption occurs.

11. Test admin panel action during peak times (updating the menu during a rush)

This test verifies that admin operations continue to function properly during peak usage periods when multiple customers are actively using the system. It validates that Admin class operations like menu updates can be performed without disrupting ongoing customer orders or causing system errors.

Presentation

Instructions:Week 12

Preparation

➔ Give a brief description of Burger Town final project

Burger **Town** is a restaurant in Riga, Latvia. Team 7 created a new, digital restaurant ordering platform tailored for casual dining environments which aims to enhance and increase the customer experience and operational efficiency through a contactless ordering system as per the request of the owner who is named Praveen Krishnan. Key features include:

- QR Code-Based Access: Each table in the restaurant has a unique QR code that customers scan to open a web-based interface linked to their table number in the restaurant.

- Responsive Web App: The application is mobile-friendly and works seamlessly across smartphones, tablets, and desktops, ensuring accessibility for all users.

- Interactive Digital Menu: Customers can explore and view the categorized menu sections (burgers, fries, drinks), view item details, and apply special customizations like ingredient changes or extra toppings as per the restaurant wish.

- Seamless Order Placement: Customers who are coming to restaurants, can add items to their cart, review their selections, and place orders directly from their device which reduces the need for waiter intervention in the restaurant.

- Order Tracking: After the customer places an order, they will receive real time updates as their food progresses from "In Progress" to "Ready" and "Completed" status.

- Order Cancellation: Customers have the option to cancel their order before it's processed by the kitchen, which increases the customer control and flexibility.

- Admin Panel Access: Restaurant admins so called administrators can log into a secure portal to manage the menu, update item availability or prices, and monitor ongoing orders and can do whatever they want on the website.

- Scalable Architecture: It Designed by team 7 to handle high customer volume, especially during peak hours, without compromising performance or user experience.

- Focus on Usability: Simple, intuitive, good, better and fine user interface interface design ensures even first-time users can navigate the system easily.

So overall team 7’s project, Burger Town not only improves dining efficiency but also minimizes wait times, reduces human error, and delivers a modern dining experience suited for today's tech-savvy customers.

➔ Describe your requirement assumptions/additions.

- ◆ The Customers at least should be at 12 years old to place an order independently, but nowadays even kids are able to place so, from our team 7 side , we can say at least 12 years.
- ◆ Maximum of 15 items allowed per order to avoid lots of which means bulk abuse or overload.
- ◆ Admins of the restaurant can modify menu items dynamically during rush hours and when they want.
- ◆ The website system should handle at least 50 concurrent customer sessions.
- ◆ Customers should be able to cancel orders before they are sent to the kitchen.
- ◆ The QR code contains table-specific context for order tracking.

➔ Describe your design options and decisions. How did you weigh the pros and cons of the different designs to make your decision?

We explored two main designs:

1. Centralized Admin Dashboard with Real-time Sync
    - *Pros*: It is Easy to manage orders, fast updates, and scalable.
    - *Cons*: It requires complex backend logic and concurrency handling.
2. Static Menu with Local Changes
    - *Pros*: It is Simple to implement, good for MVP.
    - *Cons*: It is Difficult to manage updates and sync across users.

Decision: We, team 7 went with the centralized real-time design using Node.js backend and WebSocket support for updates. This allowed us to implement dynamic features like live order tracking and menu management.

➔ How did the extension affect your design?

➔ The requirement to support multiple simultaneous customers and real-time admin updates significantly influenced our architectural decisions. We , team 7 introduced in this project:

➔ WebSocket integration for real-time order updates.

➔ Mutex logic to prevent race conditions in order placement.

➔ Queue management to maintain order integrity.

- Describe your tests (e.g., what you tested, equivalence classes).

  We performed black-box and equivalence class testing across several scenarios:

➔ *MenuItem class:* Tested availability, price boundaries, and customization logic.

➔ *Order class:* Verified handling of min/max items, customizations, and price totals.

➔ *Customer class:* Tested order workflow, QR scanning, and cancellation.

➔ *OrderItem class:* Ensured proper price and customization handling.

➔ *Admin class:* Tested authentication and menu management.

  Highlights of Test Cases:

➔ QR code scanning for table-specific context.

➔ Adding 1 vs. 15 items (boundary tests).

➔ Customizations and price reflection.

➔ Concurrent customer orders.

➔ Admin updates during rush hours.

- What lessons did you learn from the comprehensive exercise (i.e., programming concepts, software process)?
  - ◆ Team 7 learnt the Importance of defining clear requirements and boundaries early.
  - ◆ And also team 7 learnt how to implement real-time communication in a multi-user environment.
  - ◆ Mainly the Experience with modular class design, including separation of concerns.
  - ◆ In this project, Team 7 gained understanding of black-box testing, equivalence partitioning, and concurrency handling.
  - ◆ Finally, team 7 Better grasp of agile development and iterative testing phases.
- What functionalities are you going to demo?
  - ◆ Customer scans QR and accesses menu
  - ◆ Item selection and customization

- Order placement and tracking
- Order cancellation before processing
- Admin login and menu update
- Real-time order status change by admin

- Who is going to speak about each portion of your presentation? (Recall: Each group will have ten minutes to present their work; minimum length of group presentation is seven minutes. Each student must present for at least two minutes of the presentation.)
  - Sujai (2+ minutes): Project overview, QR logic, Customer class, and customization demo
  - Davyd (2+ minutes): Front-end design, MenuItem class, testing strategy
  - Abbas (2+ minutes): Order and OrderItem class logic, handling concurrency
  - Arseni (2+ minutes): Admin class, menu management, real-time updates
  - Gasim (2+ minutes): System architecture, lessons learned, testing outcomes
  - Rufi (in addition, he will present and replying questions): Performance testing and system limitations and also future improvements

- Other notes:
  - Make sure the QR scanning works live or use a pre-recorded screen video if needed.
  - Each member must be prepared to handle the 1-2 audience questions.
  - Keep fallback screenshots ready in case real-time features fail during the live demo.
  - Final slide: “Burger Town – A Seamless Dining Experience Powered by Team 7tech.”