Comprehensive Exercise Report

Team TMDA

Tohir Davjanov, Diyorbek Kholmirzaev, Ayubkhon Salimov, Myroslav Suprun.

Requirements/Analysis

Week 2

Journal

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

- After reading the client's brief (possibly incomplete description), write one sentence that describes the project (expected software) and list the already known requirements.
    - The food ordering website have to provide a seamless and convenient online platform for users to order food from various restaurants. An online ordering should enable one to view menu items, pay, and order with ease.
        - The platform must allow customer registration and login.
        - The system should be able to show categorized menus.
        - Users must be able to place orders and make payments.
        - Order status should be trackable.
        - Admins should be able to manage menus and update order statuses.
- 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.
    - Q1: Should we support multiple restaurants or just one restaurant interface?
      A1: One restaurant is sufficient for this implementation.
    - Q2: Should the system support both card and cash payments?
      A2: Yes, both should be supported.
    - Q3: Do we need separate CRM system or just a panel which is available only for admins?
      A3: Yes, admin login is required and must be separated from customer login.
    - Q4: Do u need software to be optimized for both PC and mobile users?
      A4: Yes, we need both.
- Does the project cover topics you are unfamiliar with? If so, look up the topics and list your references.
    - Yes, initially we had really limited knowledge about payment processing and its real-world integration.
- Describe the users of this software (e.g., small child, high school teacher who is taking attendance).
    - Mostly office workers and other adults which order food like they serve
- Describe how each user would interact with the software
    - Just using desktop or mobile phone users can order using our software
    - Admins should be registered as admins, and they can interact with software using PC as admin panel is not yet optimized for phones
- What features must the software have? What should the users be able to do?
    - Register/log in
    - Browse food items by category
    - Add to cart and checkout with delivery details.
    - Choose payment method.
    - Track order status in real-time.
    - Log in securely to admin panel
    - Add/edit/delete food items
    - Update order statuses through the dashboard
- Other notes:
    - Admin and users should be separated.

- All online payments have to be verifie.

Software Requirements

Functional Requirements

User Log in

1. The system shall allow users to create an account using an email and password.
2. The system shall allow users to log in and log out securely.
3. The system shall allow users to reset their passwords if it is forgotten.

Order Placement

1. The system shall allow users to browse restaurant menus and select items.
2. The system shall allow users to add or remove items from their cart.
3. The system shall allow users to apply discount codes or promotions.
4. The system shall allow users to specify delivery or pickup options.

Payment Processing

1. The system shall allow users to make payments using credit/debit cards, PayPal, and other payment gateways.
2. The system shall securely process transactions.
3. The system shall generate and display order receipts.

Order Tracking

1. The system shall allow users to track their order status.
2. The system shall send real-time notifications to users on order status changes.
3. The system shall allow restaurant owners to update order status (preparing, ready, out for delivery, delivered).

Customer Support

1. The system shall allow users to contact customer support via chat or email.
2. The system shall provide a help center with FAQs.
3. The system shall allow users to submit complaints or feedback.

Reviews and Ratings

1. The system shall allow users to rate and review restaurants.
2. The system shall display average ratings and reviews for each restaurant.
3. The system shall allow restaurants to respond to customer reviews.

Non-functional requirements.

1. The system shall work fast and without lags.
2. The system shall support dynamic page update.
3. The system shall support at least 100 active users at a time.

Black-Box Testing

Instructions: Week 4

Journal

Remember: Black box tests should only be based on your requirements and should work independent of design.

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)?
    - Text inputs (email, passwords, names)
    - Numeric values(quantity, prices addresses)
    - Images (food images)
    - Button clicks (add to cart or place an order)
- What does output for the software look like (e.g., what type of data, how many pieces of data)?
    - Messages (Order Canceled, Order Placed or Login is successful)
    - Error messages (Invalid Log in information or Invalid payment information)
    - Page redirections
    - Updated order status
- What equivalence classes can the input be broken into?
    - Valid and invalid creditals
    - Valid and invalid payment details
    - Empty cart and populated cart
    - Authenticated user and guest user
    - Admin and Customer privileges
- What boundary values exist for the input?
    - Password length: min 6 characters, max 20 characters
    - Email format validation(example@example.com)
    - Address field max length (150 chars)
- Are there other cases that must be tested to test all requirements?
    - Attempt to place an order without being logged in
    - Attempt to pay with invalid card info
    - Admin trying to update order with invalid status
    - Checkout with missing required fields
- Other notes:
    - <<Insert notes>>

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
1	User Log in	Redirection to the homepage	Redirection to the homepage
2	User login with invalid credentials	Error message shown	Error message shown
3	Adding item to cart	Item appears in the cart	Item appears in the cart
4	Order placement with a valid payment card	Order confirmation	Order confirmation
5	Track order updates	Real-time status shown	Real-time status shown
6	Attempt checkout while not logged in	Redirection to the log in page	Redirection to the log in page
7	Attempt checkout with an empty cart	Redirection to the cart	Redirection to the cart

Design

Instructions: Week 6

Journal

Remember: You still will not be writing code at this point in the process.

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

- List the nouns from your requirements/analysis documentation.
    - User, Admin, Order, Cart, Food Item, Menu, Payment, Delivery
- Which nouns potentially may represent a class in your design?
    - User, Admin, Order, Cart, FoodItem, Payment, Address
- Which nouns potentially may represent attributes/fields in your design? Also list the class each attribute/field would be a part of.
    - User: name, email, password, cartData.
    - Admin: id, email, password.
    - Order: orderId, items, amount, paymentStatus, status, deliveryAddress.
    - Cart: items, totalAmount.
    - FoodItem: name, price, image, category.
    - Payment: method, success, timestamp.
    - Address: street, city, zip, country.
- Now that you have a list of possible classes, consider different design options (***lists of classes and attributes***) along with the pros and cons of each. 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.

    Design Option 1:

    - Classes and Attributes:

    User

        userID, name, email, password, address

    Order

        orderID, userID, items, totalPrice, status, timestamp

    Pros:

        It is straightforward and easy to follow.
        It's simpler for early development and debugging.
        Good for creating a draft or sample.

    Cons:

        It's not easy to add new restaurants, update menus, or enhance admin capabilities.
        Users believe something to be related to whatever they are asking.
        They're all about payment, dining and eating out.

    Design Option 2:

    Classes and Attributes:

    User

        userID, name, email, password, role, address

    Restaurant

        restaurantID, name, description, location, menuItems[]

MenuItem

itemID, restaurantID, name, description, price, imageURL

Order

orderID, userID, restaurantID, items[], totalAmount, status, createdAt

Payment

paymentID, orderID, method, status, timestamp

Pros:

It's very flexible and can change later, for example menus or types of restaurants.

Improved security, as admin and user levels are separate.

Scalable - you can easily add things like reviews, possible follow ups after you've made the delivery or loyalty programs.

It's easy to connect to other services, like payment services such as stripe.

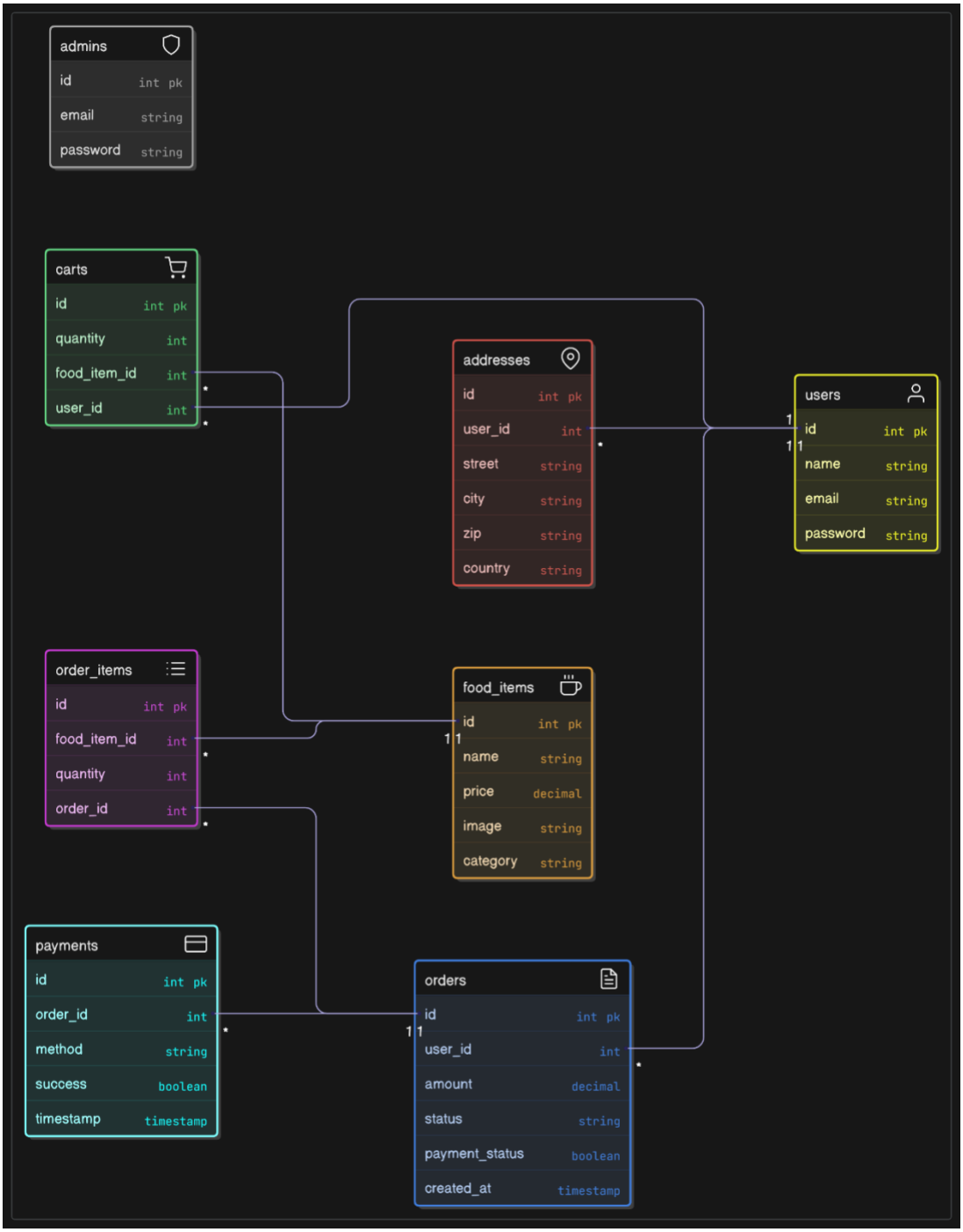
Cons:

More complicated to begin with while being designed.

Data has to prepped and in a standard format.

- Which design do you plan to use? Explain why you have chosen this design.
    - We chose Design Option 2 as it provides flexibility and space to expand in the long term. We have to develop a food delivery application that is able to support multiple restaurants, monitor updates in real time, and integrate with Stripe, so this configuration is required. It also aligns with Agile development concepts where we can create and deploy various components (backend and frontend) separately one by one. Although it is complex, it ensures stability, more structured code, and simplifies testing and debugging.
- List the verbs from your requirements/analysis documentation.
    - Register, Login, Logout, Browse, Add, Remove, Checkout, Pay, Track, Update
- Which verbs potentially may represent a method in your design? Also list the class each method would be part of.
    - User: register(), login(), logout()
    - Cart: addItem(), removeItem(), getTotal()
    - Admin: updateStatus(), manageMenu()
- Other notes:
    - <<Insert notes>>

Software Design



Use your notes from above to complete this section of the formal documentation by planning the classes, methods, and fields that will used in the software. Your design should include UML class diagrams along with method headers. ***Prior to starting the formal documentation, you should show your answers to the above prompts to your instructor.***>>

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):
    We used OOP to develop classes based on models of our users, restaurants, orders, and menu items. The advantage of OOP is the ability to encapsulate data and behavior. This means we can encapsulate models and organize the models we need for our back-end.

    Client-Server (Architecture):
    We used the term client-server architecture to describe our system, our client is using React.js to build the front-end, and we are using Node.js with Express for our server, and we are using RESTful APIs to communicate between the two.

    RESTful APIs:
    We are outlining our important RESTful APIs as hooks to connect our front-end to our back-end functionality, i.e., log-in (api), place an order (api), get menus (api), and so on.

    Asynchronous Programming (Promises/Async-Await):
    We adopted asynchronous programming (Promises/Async-Await) to module the API calls we used from our server. This includes REST API calls to get data from the MongoDB data collections, deal with payment providers such as Stripe, etc.without blocking the UI.

    Database Management (CRUD Operations):
    MongoDB is used to store the data associated with our users, food items, delivery orders etc., and all CRUD (Create, Read, Update, Delete) operations are carried out using Mongoose.

    Authentication & Authorization:
    This is needed to log in - and manage user sessions and service/admin and customer roles.

    Version Control (Git):
    We collaboratively coded with Git and GitHub, but also allowed our team to manage their own code.

Implementation Details

Food Delivery System allows restaurant customers to view restaurant menu, place an order online, pay securely using Stripe, and view order tracking in real-time. Restaurant managers can control the menus and see new orders from a specialized administration page.

User Roles
A customer can register, login, view food items, add food items to a shopping cart, and make order.
An administrator can add, update, or delete food items, update order status.

Frontend (User Interface)
Link: https://balts-t.onrender.com/
Myocardial bridges
Register/Login: secure registration and login functionality.
Explore Menu: Displays food from various restaurants.
Add to Cart: Users can select multiple items and quantities.
Payment can be made using your card information through Stripe.
Tracking your order: See where your order status is at: Getting ready → On its way → Delivered.

Admin Panel
Link: https://balts-admin.onrender.com/

Key Points
- Login: There is an admin authentication process.
- Manage Food Items: New items can be added, descriptions can be changed, prices can be modified, and old items can be deleted.
- Order Management: You can see the orders that are made and change their status.

Backend (API Overview) Built using Node.js with Express, connected to MongoDB.
POST /api/users/register - Register a new user
POST /api/users/login - Login and obtain a token.
GET /api/foods – Get all menu items
POST /api/orders - Create a new order
Fetch order by ID by going to /api/orders/:id.

POST /api/payments/stripe – Process payment using Stripe How to Use the System 1. You sign up to be a user → view menu → add items into cart 2. Proceed to checkout → input payment information using Stripe. 3. Verify order status in real time. 4. (Admin) : login to admin page → modify food/add food → modify order status

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.
    - Yes. We added Stripe integration.
- List the classes of your implementation. For each class, list equivalence classes, boundary values, and paths through code that you should test.
    - User
        - Valid/invalid credentials.
        - Duplicated email registration
    - Order
        - Valid/invalid cart
        - Payment type chosen
    - Payment
        - Valid/invalid payment
- Other notes:
    - <<Insert notes>>

Testing Details

- testLoginUser() — Validates user login logic and JWT token return
- testPlaceOrderCOD() — Places order and checks MongoDB record creation
- testStripeCheckout() — Sends mock cart and gets Stripe session URL
- testVerifyPaymentSuccess() — Verifies paid orders

Presentation

Instructions:Week 12

Preparation

The following prompts are meant to aid your thought process as you complete the presentation portion of this exercise. It is recommended that you examine the previous sections of the journal and your reflections as you work on the presentation as it is likely that you have already answered some of the following prompts elsewhere. Please respond to each of the prompts below and feel free to add additional notes.

- Give a brief description of your final project
    - The food ordering website aims to provide a seamless and convenient online platform for users to order food from various restaurants. An online ordering website enables one to view menu items, pay, and order with ease. It simplifies ordering and payment for customers.
- Describe your requirement assumptions/additions.
    - Stripe payments, admin panel, mobile and PC based responsive design, order tracking
- Describe your design options and decision. How did you weigh the pros and cons of the different designs to make your decision?
    - We chose MERN stack design with separated admin and user flows
- How did the extension affect your design?
    - We added Stripe which was not planned initially.
- Describe your tests (e.g., what you tested, equivalence classes).
    - Login, cart, payment, and status trackin are tested using API and User interface.
- What lessons did you learn from the comprehensive exercise (i.e., programming concepts, software process)?
- We got useful experience with software modularity, asynchronous programming, and secure API integration. Agile got us to track our task list incrementally, and we learned first-hand how to collaborate, commit, and test regularly at least reduces the chance of unforeseen issues in deployment. Stripe integration taught us how to actually handle security in the world and how to collaborate with a third-party service.
- What functionalities are you going to demo?
    - We will demonstrate the following:

      User flow:

      Register and log in
      Browse menu and add items to cart
      Proceed to checkout and make payment via Stripe
      View live order status

      Admin panel:

      Log in as admin
      Add a new food lite
      Update order status (e.g., “Food Processing” → “Delivered”)
- 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.)
    - Tohir Davjanov will discuss the project idea, aims and motivation.
    - Diyorbek Kholmirzaev will give an explanation of the system architecture, classes, and logic that happens in the backend.

- Myroslav Suprun will showcase the user interface (UI) of the frontend, and the features present in the user panel. Ayubkhon Salimov will exhibit the admin panel and finish off with some reflections on the research and future work.