**C372 CA2 AI Interaction Journal**

|  |  |
| --- | --- |
| **Student ID** | 24045221 |
| **Student Name** | Christine Joy Teh Shi Hui |
| **Class** | E63C |

Overview

In CA2, you will enable payments in the Supermarket App that you have worked on in CA1.

The AI Interaction Journal is designed to help you document, reflect on, and evaluate your use of generative AI tools (such as ChatGPT, Gemini, or Copilot) during the development of CA2 deliverable.

It promotes responsible and thoughtful use of AI, encouraging you to develop key skills in prompt design, critical evaluation of AI responses, and independent problem-solving.

This journal demonstrates how you used AI as a collaborative thinking partner—to explore ideas, test solutions, and deepen your understanding of code, rather than simply to produce answers.

In this journal:

* You will use this journal to record your key prompts, results, and reflections as evidence of how you engaged with AI responsibly throughout your project.
* You will need to identify the tasks related to the feature(s) you are building
* For each task, update the status, objective, goal(s), preparation work(s) for mentioned task, AI tool(s) used, entity table(s), relevant source code files, prompt(s) and results.

**Note:**

* Please read through the entire document before you begin.
* This AI Interaction Journal is part of CA2 Deliverable submission.
* Do **not** include personal information, passwords, API keys, or any confidential data. This log is for **learning reflection and responsible AI use**.

*You may refer to* ***C372 Individual Graded Assignment AI Interaction Journal – SAMPLE.docx*** *for a reference on how to use this document.*

Table of Contents

[Overview 1](#_Toc218438179)

[Enabling Payment Solution(s) in SupermarketAppMVC 3](#_Toc218438180)

[Task Logs 4](#_Toc218438181)

[Task 1 – Task 1 Name 4](#_Toc218438182)

[OBJECTIVE 4](#_Toc218438183)

[PROMPT(S) USED 4](#_Toc218438184)

[AI RESPONSES/CODE GENERATED 4](#_Toc218438185)

[RESULTS 4](#_Toc218438186)

[Screenshots of the Payment Process Flow 5](#_Toc218438187)

[Reflection on the use of GenAI for Enabling Payments 6](#_Toc218438188)

**Note: Remember to update the table of contents for the latest updates to reflect above.**

# Enabling Payment Solution(s) in SupermarketAppMVC

1. Identify the payment solution(s) you plan to implement in SupermarketAppMVC.

|  |
| --- |
| * PayPal Checkout for online card and PayPal payments * NETS QR for local QR-based payments * In-app E-wallet, allowing users to top up via PayPal or NETS * Refund system where users can request refunds and admins can approve or reject them * Loyalty points system where users earn points after each purchase and can redeem them as discounts on future purchases |

1. Explain how you plan to implement the payment solution(s) identified in a).

|  |
| --- |
| I plan to implement PayPal by allowing users to pay online using PayPal or cards, where the system confirms the payment, records how the user paid, and then completes the order while giving loyalty points. NETS payments will work through a QR code that users scan to pay, and once the payment is confirmed, the order is created and loyalty points are awarded. The in-app E-wallet will store a balance for each user, which they can top up using PayPal or NETS, and when they pay with the wallet, the system checks if there is enough balance, deducts the amount, and completes the order. A refund feature will also be included, allowing users to request refunds for their purchases, with administrators deciding whether to approve or reject them. If a refund is approved, the money is credited to the in‑app e‑wallet regardless of original payment method |

1. In your chosen Generative AI tool (LLM), start a new chat session dedicated to your CA2 work.

Identify the tasks that are related to CA2. Add as many tasks as required.

|  |  |  |
| --- | --- | --- |
| **Task No.** | **Task Name** | **Task Description** |
| 1 | Generate PaymentController.js - that handles PayPal, NETS, wallet payments | Handle PayPal, NETS, wallet payments |
| 2 | Generate PayPalclient.js – connect to paypal | Create, capture, refund PayPal payments |
| 3 | Generate Netsclient.js – connect to NETS | Generate QR code and confirm payment. |
| 4 | Update App.js - Add payment routes | Connects payment routes in the server |
| 5 | Update Ordermodel.js - Save payment information | Store how the user paid and the payment ID |
| 6 | Update payment.ejs – check out page | Show PayPal, NETS, wallet options and points at checkout |
| 7 | Generate netQR.ejs - Create NETS QR page | Show the QR code and an error page if it fails. |
| 8 | Generate walletTopup.ejs - Create wallet top‑up page | Let users add money to their wallet |
| 9 | Update membershipPayment.ejs – update membership payment page | Let users pay for membership by PayPal or NETS |
| 10 | Update – ordersHistory.ejs – update order history | Show payment method and payment ID in orders |
| 11 | Generate refundController.js – add refund | Let admins/users request and process refunds |
| 12 | Update .env – add payment configuration | Put PayPal/NETS keys and URLs |
| 13 | Update orderController.js – add loyalty points | Earn and redeem points when paying |
| 14 | Generate refundModel.js - Refund mode | Store and retrieve refund data |
| 15 | Generate adminRefunds.ejs – Admin refunds page | Admin view to manage refund |
| 16 | Update invoice.ejs – invoice page | how payment info on invoice |
| 17 | Generate loyalty.ejs – loyalty page | Show points info and rules. |
| 18 | Generate netsQRfail.ejs – NETS QR fail page | Shows error if fail |
| 19 | Generate requestRefund-ejs – refund request page | User form to request refund |

**Entity Table(s)**

List and define the key information needed to support payment solutions in your application.

Table Name: Orders

|  |  |
| --- | --- |
| **Fields** | **Datatype** |
| id | INT (AUTO\_INCREMENT) |
| userId | INT |
| user\_id | INT |
| subtotal | DECIMAL(10,2) |
| total | DECIMAL(10,2) |
| savings | DECIMAL(10,2) |
| status | VARCHAR(50) |
| itemsJson | LONGTEXT |
| createdAt | TIMESTAMP |
| deliveryMethod | VARCHAR(50) |
| deliveryAddress | VARCHAR(255) |
| pickupOutlet | VARCHAR(255) |
| paymentMethod | VARCHAR(50) |
| paymentRef | VARCHAR(255) |

Table Name: Refunds

|  |  |
| --- | --- |
| **Fields** | **Datatype** |
| id | INT (AUTO\_INCREMENT) |
| orderId | INT |
| userId | INT |
| reason | TEXT |
| documentPath | VARCHAR(255) |
| status | VARCHAR(20) |
| adminNote | TEXT |
| createdAt | TIMESTAMP |
| updatedAt | TIMESTAMP |

# Task Logs

For each task, use the task template below:

## Task 1 - Generate PaymentController.js - that handles PayPal, NETS, wallet payments

Date Started(18/01/2026):

Date Updated(18/01/2026):

Status (Not started/In Progress/Completed): Completed

### OBJECTIVE

Create a backend controller that manages PayPal, NETS QR, and wallet payment flows for checkout, membership, and wallet top‑ups.

**GOAL(S)**

* Handle Paypal create and confirm payment
* Handle NETS QR generation and confirmation
* Handle wallet top-up and wallet payment

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Setting up routes in app.js
* Define payment fields in order model

**AI TOOL(S) OR API USED**

* CodeX
* Chagpt

**RELEVANT SOURCE CODE FILE(S)**

* paymentController.js
* paypalClient.js
* netsClient.js
* orderModel.js
* userModel.js

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Create a PayPal payment 2. Generate NETS QR codes and confirm NETS payments. 3. Support wallet top‑up via PayPal and NETS. 4. Process wallet payments at checkout with balance checks. 5. Update order records with payment method and payment ID. 6. Apply loyalty point earn/redeem during payment. 7. Handle errors and return clear responses to the frontend. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • PayPal Payments: The controller can start a PayPal payment and confirm it after PayPal says it’s paid. • NETS QR Payments: The controller generates a NETS QR code and confirms payment after the user scans and pays. • Wallet Top‑ups: Users can add money to their wallet using PayPal or NETS. • Wallet Checkout: Users can pay with their wallet, and the system checks if they have enough balance first. • Save Payment Info: The order is saved with payment method and payment ID. • Loyalty Points: Points can be earned or redeemed during payment. • Error Handling: Clear success/error responses are sent back to the frontend. * Recommendations: • Validate inputs like amounts before processing payments. • Use session data to track pending NETS payments. • Show clear error messages when payment fails. * Whether Used: • Input Validation: Used • Session Tracking for NETS: Used • Clear Error Messages: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| * It works as intended for the main flows (PayPal, NETS QR, wallet top‑up, wallet checkout, loyalty points). * Tested manually by running the app and completing each payment path. |

# 

# Task Logs

For each task, use the task template below:

## Task 2 – Generate PayPalclient.js – connect to paypal

Date Started(18/01/2026):

Date Updated(18/01/2026):

Status (Not started/In Progress/Completed): completed

### OBJECTIVE

Create a PayPal client module that connects to PayPal’s API to create, capture payments

**GOAL(S)**

* Get PayPal access token using client ID and secret
* Create PayPal orders for a given amount
* Capture PayPal payments after user pays

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Add PayPal credentials and API URL in .env
* Ensure payment routes call the PayPal client functions

**AI TOOL(S) OR API USED**

* Codex
* Chagpt

**RELEVANT SOURCE CODE FILE(S)**

* PaypalClient.js
* .env
* paymentController.js
* refundController.js

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Get a PayPal access token using client ID and secret. 2. Create a PayPal order for a given amount. 3. Capture a PayPal order after payment. 4. Return clear errors when PayPal calls fail. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • Token Handling: Request and use OAuth access token for API calls. • Create Order: Send amount to PayPal and return the order data. • Capture Order: Confirm payment and return capture info. • Error Handling: Return clear error messages on API failure. * Recommendations: • Store PayPal keys in .env and do not hardcode them. • Handle API errors and log useful messages. * Whether Used: • Env‑based credentials: Used • Error handling: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| Works as intended for create/capture calls Tested manually by running payment flows in the app. |

# Task Logs

For each task, use the task template below:

## Task 3 – Generate Netsclient.js – connect to NETS

Date Started(18/01/2026):

Date Updated(18/01/2026):

Status(Not started/In Progress/Completed): completed

### OBJECTIVE

Create a NETS client module that connects to the NETS API to generate QR codes and confirm payments

**GOAL(S)**

* Generate NETS QR codes for payment amounts
* Confirm NETS payments and return status
* Handle NETS API errors cleanly

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Add NETS API URL and credentials in .env
* Ensure payment routes call the NETS client functions

**AI TOOL(S) OR API USED**

* Codex
* Chatgpt

**RELEVANT SOURCE CODE FILE(S)**

* netsClient.js
* .env
* paymentController.js

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Generate a NETS QR code for a payment amount. 2. Confirm the NETS payment after the user scans and pays. 3. Return clear errors if the NETS API fails. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • QR Generation: Send amount to NETS API and return QR code data. • Payment Confirmation: Call NETS API to verify payment status. • Error Handling: Return clear error messages and fallback page when QR generation fails. * Recommendations: • Store NETS API details in .env instead of hardcoding. • Use clear user‑facing error messages on failures. * Whether Used: • Env‑based NETS config: Used • Error handling: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| Works as intended for QR generation and confirmation. Tested manually by generating a QR code and confirming payment flow |

# Task Logs

For each task, use the task template below:

## Task 4 – Update App.js - Add payment routes

Date Started(18/01/2026):

Date Updated(18/01/2026):

Status(Not started/In Progress/Completed): completed

### OBJECTIVE

Update the app routing file to connect all payment URLs to the correct controller actions.

**GOAL(S)**

* Add PayPal create and capture routes
* Add NETS QR and confirm routes
* Add wallet top‑up and wallet payment routes
* Add membership payment routes

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* paymentController.js
* identify all payment routes

**AI TOOL(S) OR API USED**

* codex
* chatgpt

**RELEVANT SOURCE CODE FILE(S)**

* app.js
* paymentController.js
* orderController.js
* userController.js

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Add PayPal routes for create‑order and capture‑order. 2. Add NETS routes for QR generation and confirmation. 3. Add wallet routes for top‑up and payment. 4. Add membership payment routes. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • Added routes in app.js that map payment URLs to controller functions. • Included routes for checkout, membership payments, and wallet top‑ups. • Used checkAuthenticated middleware to protect payment routes. * Recommendations: • Keep payment routes grouped and labeled for clarity. • Protect payment routes with authentication. * Whether Used: • Grouped routes: Used • Auth protection: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| • Routes work as intended and call the correct controller functions. • Tested manually by hitting the payment pages and actions. |

# Task Logs

For each task, use the task template below:

## Task 5 – Update Ordermodel.js - Save payment information

Date Started(DD/MM/YYYY): 18/01/2026

Date Updated(DD/MM/YYYY): 18/01/2026

Status(Not started/In Progress/Completed): completed

### OBJECTIVE

Update the order model to store payment method and payment ID with each order.

**GOAL(S)**

* Add paymentMethod and paymentRef fields in order creation
* Save payment data when an order is placed
* Retrieve payment info with orders when needed

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Identify which order fields are required for payment records
* Confirm database schema supports payment fields

**AI TOOL(S) OR API USED**

* Codex
* chatgpt

**RELEVANT SOURCE CODE FILE(S)**

* orderModel.js
* sql

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Add payment method and payment reference fields to order creation. 2. Save paymentMethod and paymentRef in the orders table. 3. Ensure order retrieval includes payment fields. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • Added paymentMethod and paymentRef parameters to createOrder. • Included these fields in SQL insert statements. • Ensured order reads include payment info. * Recommendations: • Keep default values as null when payment is not provided. • Ensure database columns exist for these fields. * Whether Used: • Default null values: Used • Column checks/migrations: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| * Payment info is saved with each order and shows in order data. * Tested by placing orders with PayPal/NETS/wallet. |

# Task Logs

For each task, use the task template below:

## Task 6 – Update payment.ejs – check out page

Date Started(DD/MM/YYYY): 18/01/2026

Date Updated(DD/MM/YYYY): 18/01/2026

Status(Not started/In Progress/Completed): completed

### OBJECTIVE

Update the checkout page to show PayPal, NETS QR, wallet payment options, and loyalty points.

**GOAL(S)**

* Display PayPal button and NETS QR flow
* Allow wallet payment and wallet top‑up
* Show loyalty points earn/redeem options
* Show totals and validation messages

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Ensure payment routes are ready
* Provide PayPal client ID to the view

**AI TOOL(S) OR API USED**

* Codex
* chatgpt

**RELEVANT SOURCE CODE FILE(S)**

* payment.ejs
* orderController.js
* paymentController.js

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Add PayPal button and call create/capture endpoints. 2. Add NETS QR payment form and confirmation. 3. Add wallet pay button and top‑up link. 4. Show loyalty points earn/redeem section. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • Embedded PayPal SDK and render PayPal button. • Added NETS QR payment form with hidden loyalty points. • Added wallet payment button with balance check and top‑up link. • Displayed loyalty points, max redeem and earned points. * Recommendations: • Disable payment buttons if cart has stock issues or insufficient wallet balance. • Keep loyalty input clamped to valid values. * Whether Used: • Button disabling: Used • Loyalty input clamping: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| Checkout shows all payment methods and loyalty info correctly. Tested manually by opening checkout and trying each payment option. |

# Task Logs

For each task, use the task template below:

## Task 7 – Generate netQR.ejs - Create NETS QR page

Date Started(DD/MM/YYYY): 18/01/2026

Date Updated(DD/MM/YYYY): 18/01/2026

Status(Not started/In Progress/Completed): completed

### OBJECTIVE

Create a NETS QR page that shows the QR code and lets users confirm payment.

**GOAL(S)**

* Display NETS QR code and payment details
* Provide confirm/cancel actions
* Show clear instructions for users

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Ensure NETS client generates QR data
* Add the NETS routes in app.js

**AI TOOL(S) OR API USED**

* Codex
* Chatgpt
* NETS API

**RELEVANT SOURCE CODE FILE(S)**

* netsQR.ejs
* paymentController.js
* netsClient.js

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Create a NETS QR page that shows the QR image and amount. 2. Add a confirm payment button and cancel link. 3. Show simple instructions for scanning. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • Rendered QR image and amount to pay. • Added confirm and cancel actions. • Included loyalty points info and hidden inputs for confirmation. * Recommendations: • Provide a clear fallback if QR data is missing. • Keep instructions short and visible. * Whether Used: • QR fallback handling: Used • Clear instructions: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| • QR page displays correctly and allows confirmation. • Tested manually by loading the QR page and submitting confirmation. |

# Task Logs

For each task, use the task template below:

## Task 8 – Generate walletTopup.ejs - Create wallet top up page

Date Started(DD/MM/YYYY): 18/01/2026

Date Updated(DD/MM/YYYY): 19/01/2026

Status(Not started/In Progress/Completed): completed

### OBJECTIVE

Create a wallet top‑up page so users can add money using PayPal or NETS.

**GOAL(S)**

* Show current wallet balance
* Allow top‑up via PayPal
* Allow top‑up via NETS QR
* Display clear instructions/errors

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Ensure wallet top‑up routes exist
* Provide PayPal client ID to the view
* Ensure NETS QR route exist
* Provide NETS QR data to the view from netsclient

**AI TOOL(S) OR API USED**

* Codex
* Chatgpt
* Paypal JS SDK
* NETS API

**RELEVANT SOURCE CODE FILE(S)**

* walletTopup.ejs
* paymentController.js

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Create a wallet top‑up page showing current balance. 2. Add a PayPal top‑up form and button. 3. Add a NETS QR top‑up option. 4. Show clear success/error messages. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • Displayed wallet balance and top‑up instructions. • Added PayPal button that calls create/capture endpoints. • Added NETS QR top‑up form with amount input. • Displayed warnings if PayPal is not configured. * Recommendations: • Enforce minimum top‑up amount. • Show clear feedback for invalid inputs. * Whether Used: • Minimum amount check: Used • Clear feedback: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| • Page works and allows top‑up with PayPal/NETS. • Tested manually by loading the page and submitting top‑ups. |

# Task Logs

For each task, use the task template below:

## Task 9 – Update membershipPayment.ejs – update membership payment page

Date Started(DD/MM/YYYY): 18/01/2026

Date Updated(DD/MM/YYYY): 18/01/2026

Status(Not started/In Progress/Completed): completed

### OBJECTIVE

Update the membership payment page to include PayPal, NETS QR, and E‑wallet payment options with clear pricing info.

**GOAL(S)**

* Show PayPal and NETS QR payment options
* Add E‑wallet payment option with balance check
* Display selected plan details and price

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Ensure membership payment routes exist
* Provide PayPal and NETS client ID and wallet balance to the view

**AI TOOL(S) OR API USED**

* CodeX
* ChatGPT
* PayPal JS SDK
* NETS API

**RELEVANT SOURCE CODE FILE(S)**

* membershipPayment.ejs
* paymentController.js
* userController.js
* app.js

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Add PayPal buttons for membership payment. 2. Add NETS QR form for membership payment. 3. Add E‑wallet option with balance check and top‑up link. 4. Show selected plan details and pricing. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • Added PayPal button and NETS QR form under the membership payment section. • Added E‑wallet card showing balance, pay button, and top‑up link. • Disabled wallet payment if balance is insufficient. • Passed wallet balance from controller to the view. * Recommendations: • Disable wallet payment when balance is too low. • Keep payment options grouped and easy to scan. * Whether Used: • Wallet balance check: Used • Clear layout grouping: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| • Membership payment page shows PayPal, NETS QR, and wallet options correctly. • Tested manually by loading the page and checking UI states for low balance. |

# Task Logs

For each task, use the task template below:

## Task 10 – Update – ordersHistory.ejs – update order history

Date Started(DD/MM/YYYY): 18/01/2026

Date Updated(DD/MM/YYYY): 18/01/2026

Status(Not started/In Progress/Completed): completed

### OBJECTIVE

Update the order history page to show how the user paid and the payment ID.

**GOAL(S)**

* Display payment method for each order
* Display payment ID/reference for each order
* Keep the order summary readable

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Ensure orders include paymentMethod and paymentRef
* Confirm data is passed to the history view

**AI TOOL(S) OR API USED**

* Codex
* chatgpt

**RELEVANT SOURCE CODE FILE(S)**

* ordersHistory.ejs
* orderController.js
* orderModel.js

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Show payment method and payment ID in order history. 2. Keep layout consistent with existing order details. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • Added “Payment method” and “Payment ID” lines to each order detail. • Used n/a when payment info is missing. * Recommendations: • Show fallback text when data is missing. * Whether Used: • Fallback text: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| • Order history now shows payment method and payment ID. • Tested by loading order history with existing orders. |

# Task Logs

For each task, use the task template below:

## Task 11 – Generate refundController.js – add refund

Date Started(DD/MM/YYYY): 18/01/2026

Date Updated(DD/MM/YYYY): 27/01/2026

Status(Not started/In Progress/Completed): completed

### OBJECTIVE

Create a refund controller to handle refund requests and approvals.

**GOAL(S)**

* Allow users to submit refund requests
* Let admins approve/deny refunds
* Process refunds by crediting the user’s e‑wallet.

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Refund model and refund routes exist
* Ensure payment methods and IDs are stored in orders

**AI TOOL(S) OR API USED**

* Codex
* ChatGpt

**RELEVANT SOURCE CODE FILE(S)**

* refundController.js
* refundModel.js
* userModel.js
* orderController.js

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Create a refund controller with routes to submit, approve, and deny refund requests. 2. Validate that the refund request belongs to the logged‑in user and prevent duplicate pending requests. 3. On approval, credit the user’s e‑wallet balance. 4. Store admin notes and update refund status (pending/approved/denied). 5. Return clear flash messages for success and error cases. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| 1. Key Implementation Details: • Added handlers for refund request submission and admin approval/denial. • Validated refund eligibility (PayPal or wallet only). • All approved refunds credit the user’s wallet balance • Stored refund status and notes.  * Recommendations: • Validate refund amount vs order total. • Log errors and show friendly messages. * Whether Used: • Refund validation: Used • Error handling: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| Refund requests work and approved refunds credit the e‑wallet. Tested manually by submitting a refund request and approving it in admin. |

# Task Logs

For each task, use the task template below:

## Task 12 – Update .env – add payment configuration

Date Started(DD/MM/YYYY): 18/01/2026

Date Updated(DD/MM/YYYY): 27/01/2026

Status(Not started/In Progress/Completed): completed

### OBJECTIVE

Add payment configuration values to .env for PayPal and NETS.

**GOAL(S)**

* Store PayPal client ID, secret, and API URL
* Store NETS API URL and required keys
* Do not write API keys/passwords directly in your .js files. Store them in .env instead, so it will not be exposed in the code.

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Identify required PayPal/NETS env variables
* Ensure the app reads config from .env
* Took lesson’s code to integrate and reference.

**AI TOOL(S) OR API USED**

* Codex
* Chatgpt
* Paypal REST API
* NETS API

**RELEVANT SOURCE CODE FILE(S)**

* .env
* paypalClient.js
* netsClient.js

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Add PayPal environment variables for client ID, secret, and API URL. 2. Add NETS environment variables for API URL and keys. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • Added PayPal and NETS config values into .env. • Services read these values at runtime. * Recommendations: • Do not hardcode keys in source files. • Keep .env out of version control. * Whether Used: • Env‑based config: Used • No hardcoded keys: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| • Payment configuration loads correctly from .env. • Tested by starting the app and verifying payment pages load. |

# Task Logs

For each task, use the task template below:

## Task 13 – Update orderController.js – add loyalty points

Date Started(DD/MM/YYYY): 18/01/2026

Date Updated(DD/MM/YYYY): 18/01/2026

Status(Not started/In Progress/Completed): completed

### OBJECTIVE

Add loyalty point earn/redeem logic during checkout and payment.

**GOAL(S)**

* Calculate points earned based on amount paid
* Allow points redemption with limits
* Apply savings from loyalty points in totals

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Ensure user model supports loyalty points
* Decide earn and redeem rules for example the rates and limits

**AI TOOL(S) OR API USED**

* Codex
* chatgpt

**RELEVANT SOURCE CODE FILE(S)**

* orderController.js
* userModel.js
* payment.ejs

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Add loyalty points earn/redeem logic to cart summary. 2. Limit redemption to a max percentage of total. 3. Calculate points earned after discounts. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • Added functions to calculate points earned and redemption savings. • Applied redemption to total and tracked points spent/earned. • Exposed loyalty data to the payment page. * Recommendations: • Clamp redemption to valid increments (e.g., 100 points). • Prevent negative totals and points. * Whether Used: • Redemption clamping: Used • Total safety checks: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| • Loyalty points calculate correctly and apply at checkout. • Tested manually by redeeming points and checking totals. |

# Task Logs

For each task, use the task template below:

## Task 14 – Generate refundModel.js - Refund model

Date Started(DD/MM/YYYY): 18/01/2026

Date Updated(DD/MM/YYYY): 18/01/2026

Status(Not started/In Progress/Completed): completed

### OBJECTIVE

Create a refund model to store and manage refund records in the database.

**GOAL(S)**

* Create refund records linked to orders and users
* Retrieve refund requests for users and admins
* Update refund status (pending/approved/denied)

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Ensure refunds table exists in the database

**AI TOOL(S) OR API USED**

* **Codex**
* **Chatgpt**

**RELEVANT SOURCE CODE FILE(S)**

* refundModel.js
* sql

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Create a refund model with CRUD functions. 2. Add methods to create, list, and update refund status. 3. Link refunds to orders and users. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • Added DB queries to create refund requests. • Added queries to list refunds for admin and user. • Added update method for refund status and admin notes. * Recommendations: • Validate required fields before insert. • Handle missing orders/users gracefully. * Whether Used: • Input validation: Used • Error handling: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| • Refund records can be created and updated successfully. • Tested manually by submitting and viewing refund requests. |

# Task Logs

For each task, use the task template below:

## Task 15 – Generate adminRefunds.ejs – Admin refunds page

Date Started(DD/MM/YYYY): 18/01/2026

Date Updated(DD/MM/YYYY): 27/01/2026

Status(Not started/In Progress/Completed): completed

### OBJECTIVE

Create an admin page to view and manage refund requests.

**GOAL(S)**

* List all refund requests with status
* Allow approve/deny actions
* Show refund details

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Ensure refund controller routes exist
* Ensure refund data is available to the view

**AI TOOL(S) OR API USED**

* Codex
* chatgpt

**RELEVANT SOURCE CODE FILE(S)**

* adminRefund.ejs
* refundController.js
* refundModel.js

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Create an admin refunds page listing requests and status. 2. Add approve/deny buttons with admin notes. 3. Show order ID, user, amount, and proof file link. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • Displayed refunds in a table with status badges. • Added approve/deny forms for each pending refund. • Included links to uploaded documents. * Recommendations: • Confirm admin actions with a note field. • Show clear status labels. * Whether Used: • Admin note input: Used • Status badges: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| • Admin can review and process refunds. • Tested manually by opening admin refunds page and submitting actions. |

# Task Logs

For each task, use the task template below:

## Task 16 – Update the invoice.ejs - invoice page to show payment method and payment ID

Date Started(DD/MM/YYYY): 18/01/2026

Date Updated(DD/MM/YYYY): 18/01/2026

Status(Not started/In Progress/Completed):

### OBJECTIVE

Update the invoice page to show payment method and payment ID.

**GOAL(S)**

* Display payment method on the invoice
* Display payment reference/ID on the invoice

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Ensure orders include paymentMethod and paymentRef
* Confirm invoice view receives order data

**AI TOOL(S) OR API USED**

* Codex
* chatgpt

**RELEVANT SOURCE CODE FILE(S)**

* invoice.ejs
* orderController.js
* orderModel.js

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Add payment method and payment ID to the invoice summary. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • Added “Payment Method” and “Payment ID” fields in the order summary section. • Used n/a if payment data is missing. * Recommendations: • Show fallback text when data is missing. * Whether Used: • Fallback text: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| Invoice now shows payment method and payment ID. Tested manually by opening an invoice. |

# Task Logs

For each task, use the task template below:

## Task 17 – Generate loyalty.ejs – loyalty page Show points info and rules

Date Started(DD/MM/YYYY): 18/01/2026

Date Updated(DD/MM/YYYY): 27/01/2026

Status(Not started/In Progress/Completed): completed

### OBJECTIVE

Create a loyalty page that shows the user’s points balance and explains how points are earned and redeemed.

**GOAL(S)**

* Show current loyalty points balance
* Explain earn/redeem rules and limits
* Link users back to the payment page

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Ensure user loyalty points are stored and retrievable
* Confirm the loyalty route passes points to the view

**AI TOOL(S) OR API USED**

* Codex
* Chatgpt

**RELEVANT SOURCE CODE FILE(S)**

* loyalty.ejs
* userController.js
* orderController.js

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Create a loyalty points page that shows the user’s current points balance. 2. Add simple rules for earning and redeeming points. 3. Include a link back to the payment page. 4. Add a short “How you can use points” section with bullet rules 5. Put it on the navbar |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • Displayed current points balance in a summary card. • Added rule list for redemption (100 points = $1 off, max 50%). • Added tips and a link to the payment page. * Recommendations: • Keep rules short and readable. • Show clear call-to-action to use points. * Whether Used: • Simple rules layout: Used • Call-to-action link: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| Loyalty page shows points and redemption rules correctly Tested manually by opening the loyalty page after login. |

# Task Logs

For each task, use the task template below:

## Task 18 – Generate netsQRfail.ejs – NETS QR fail page Shows error if fail

Date Started(DD/MM/YYYY): 18/01/2026

Date Updated(DD/MM/YYYY): 27/01/2026

Status(Not started/In Progress/Completed): completed

### OBJECTIVE

Create a NETS QR failure page that displays error details and gives users clear next steps.

**GOAL(S)**

* Show NETS error message and response code
* Provide actions to return to payment or continue shopping

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Ensure NETS error payload is passed to the view
* Confirm routes redirect to the fail page on timeout or error
* Used lesson code as reference

**AI TOOL(S) OR API USED**

* Codex
* Chatgpt

**RELEVANT SOURCE CODE FILE(S)**

* netsQrFail.ejs
* netsClient.js
* paymentController.js
* app.js

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Create a NETS QR failure page that shows the error message and response code. 2. Add a short instruction section if available from NETS. 3. Provide buttons to go back to payment or continue shopping |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • Displayed title, error message, response code, and optional instructions. • Added clear buttons to return to payment or keep shopping. * Recommendations: • Keep error details visible but concise. • Provide immediate navigation options. * Whether Used: • Error summary layout: Used • Navigation actions: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| Failure page shows error info and gives users recovery options. Tested by forcing a NETS timeout and loading the fail page |

# Task Logs

For each task, use the task template below:

## Task 19 – Generate requestRefund-ejs – refund request page User form to request refund

Date Started(DD/MM/YYYY): 18/01/2026

Date Updated(DD/MM/YYYY): 27/01/2026

Status(Not started/In Progress/Completed): completed

### OBJECTIVE

Create a refund request page where users can submit a reason and upload supporting documents.

**GOAL(S)**

* Show order summary and amount for the refund
* Allow users to submit a refund reason and optional document
* Inform users that approved refunds are credited to the e‑wallet

**PREPARATION WORK(S) FOR THE MENTIONED TASK**

* Ensure refund routes and controller exist
* Ensure order data is passed to the view

**AI TOOL(S) OR API USED**

* Codex
* Chatgpt

**RELEVANT SOURCE CODE FILE(S)**

* refundRequest.ejs
* refundController.js
* refundModel.js
* app.js

### PROMPT(S) USED

Paste the key prompt(s) you used

|  |
| --- |
| 1. Create a refund request form with reason input and optional document upload. 2. Show order summary (order ID, total, date). 3. Add a note that approved refunds go to the e‑wallet. 4. Provide submit and cancel actions. |

### AI RESPONSES/CODE GENERATED

Paste or summarize the AI-generated response/code. Note key implementation details, recommendations, and whether you used them.

|  |
| --- |
| * Key Implementation Details: • Displayed order summary and total at the top. • Added text area for refund reason and file upload for proof. • Included e‑wallet refund note and navigation actions. * Recommendations: • Require a reason before submission. • Keep the form short and clear. * Whether Used: • Required reason field: Used • Clear layout: Used |

### RESULTS

Describe if the output works as intended, how it was tested, and any remaining issues.

|  |
| --- |
| Refund request page works and submits the form correctly. Tested manually by opening the page and submitting a request. |

# Screenshots of the Payment Process Flow

In this section, you should include clear screenshots of your application that illustrate each step of the payment process in your application, from initiating a payment to receiving confirmation. These visuals should demonstrate the user experience and help explain how the payment feature works within your app.

|  |
| --- |
| This is the check out page    Applied 400 loyalty points, hence there is points discount of $4    Paid with e wallet. So the e wallet from $104 to $99.90    I am going to pay with NETS next    This is the NETS QR page where user use their phone to scan to pay    Payment will show successful on the phone and it will redirect the user to the invoice page on the website    Next, I will be paying with PAYPAL    It will ask the user to log in into their paypal    Then, will ask the user to confirm and pay    Once, successful, it will direct the user to the invoice page    Then, it will be reflected on user’s paypal’s transaction history |

# Reflection on the use of GenAI for Enabling Payments

Focus on how you engaged with generative AI as a learning partner while implementing payment functionality.

**Your responses should include specific examples from your code to demonstrate critical thinking, testing, and ethical awareness in your use of AI.**

1. **How did you use GenAI to understand payment integration requirements?**

*Describe how you used AI tools to research payment solutions, clarify requirements, or explore best practices for secure payment processing. What resources or guidance did AI provide that shaped your approach?*

|  |
| --- |
|  |

1. **What challenges did you face when enabling payments, and how did GenAI help you address them?**

*Reflect on any technical, security, or design challenges you encountered while implementing payments. How did you use AI to troubleshoot, generate code, or find solutions? Provide examples.*

|  |
| --- |
|  |

1. **How did you ensure the payment feature was secure and reliable?**

*Explain how you used AI to check for security vulnerabilities, validate payment flows, or follow industry standards. Did AI help you identify potential risks or suggest improvements?.*

|  |
| --- |
|  |

1. **How did you test and validate the payment functionality with the help of GenAI?**

*Describe your process for testing the payment feature. Did you use AI to generate test cases, debug errors, or verify that transactions were processed correctly? What changes did you make based on AI feedback?*

|  |
| --- |
|  |

1. **What limitations or risks did you notice in the AI’s suggestions for payment integration?**

*Critically evaluate the AI’s responses. Did it ever suggest insecure, outdated, or non-compliant solutions? How did you identify and address these issues?*

|  |
| --- |
|  |

1. **How did you balance AI assistance with your own learning and decision-making?**

*Discuss how you ensured that using AI enhanced your understanding of payment systems, rather than just providing answers. What steps did you take to independently review, test, or adapt AI-generated solutions?*

|  |
| --- |
|  |