Report on eSalesOne Test Task Deployment and Demonstration

1. Introduction

This report outlines the eSalesOne test task, highlighting the two separate code repositories (front-end and back-end), the deployment process on a DigitalOcean server, and a short video demonstration covering three purchase scenarios (approved, declined, and failed). All components have been made publicly accessible to allow evaluation of functionality and user flows.

2. Project Structure

2.1 Front-End Repository

Purpose: Implements the user-facing interface for eSalesOne, handling product display, shopping cart functionality, and payment form integration.

Contents:

- HTML/CSS/JavaScript (or framework files) that render product listings.
- Routing logic to navigate between catalog, cart, and checkout pages.
- Integration with a payment gateway (simulated in the test environment) to submit purchase requests to the back end.
- Validation on input fields (e.g., credit-card data, billing/shipping address).

Access: The front-end code is hosted in its own Git repository:
https://github.com/farshadmrd/eSalesone

Reviewers can clone or fork this repository to inspect components, services, and build scripts.

2.2 Back-End Repository

Purpose: Provides API endpoints to process orders, handle payment logic, and return status responses (approved, declined, or failed). Includes minimal business logic to simulate a payment-processor integration.

Contents:

- A RESTful API Django
- A configuration file (or environment variables) specifying test credentials for the simulated payment gateway.
- Controllers or service modules that evaluate payment details and return one of three possible statuses:

1. Approved – Simulates a successful transaction.
2. Declined – Simulates a payment declined by the issuer/bank.
3. Failed – Simulates a system-level failure (e.g., network timeout, server error).

In the real world, a payment card number consists of 16 digits. However, since we're using fake payment data here, you can use the following single-digit codes to simulate different outcomes:

- Approved → insert in card number 1
- Declined → insert in card number 2
- Failed → insert in card number 3

Access: The back-end code is hosted in its own Git repository:
https://github.com/farshadmrd/eSalesone-backend

Reviewers can clone or inspect the repository to verify routing, middleware, and error-handling logic.

3. Deployment on DigitalOcean

Hosting Environment: Both front-end and back-end components have been deployed to a single DigitalOcean Droplet.

Configuration:

- Web Server: Nginx is configured as a reverse proxy to serve static assets (front-end) and forward API requests (e.g., /api/*) to the back-end application.
- Domain Setup: The domain **https://esalesone.live** has been pointed via an A record to the Droplet's public IP.

By navigating to **https://esalesone.live** users can access the full application: browse products, add to cart, and attempt purchases under the three test scenarios.

4. Video Demonstration

A concise video was recorded to showcase the primary purchase flows and system behavior under different transaction outcomes. The video highlights:

- Navigation:
- Browsing the available products.
- Adding items to the cart.
- Proceeding to checkout.
- Entering Payment Details:
- Filling in test credit-card information.
- Submitting the payment form.
- Outcome Display:
- Real-time feedback messages for approved, declined, and failed transactions.

The link to view the demonstration is:
https://drive.google.com/file/d/1dq3RKOU_htF9vZvQssXf0gNDFjGlmK2j/view?usp=drive_link

5. Conclusion

The eSalesOne test task successfully integrates separate front-end and back-end repositories into a deployed environment on DigitalOcean. By visiting https://esalesone.live/, evaluators can:
1. Clone or inspect the two Git repositories to review implementation details (front-end and back-end).
2. Navigate through the live site's product catalog and complete a checkout using test cards to observe three distinct transaction outcomes.
3. View a concise video demonstration (link above) that showcases the full user experience for "approved," "declined," and "failed" payment scenarios.

This separation of concerns (front-end vs. back-end) and the deployment on a publicly accessible domain allow for straightforward review of both the code and the live behavior. Evaluators can verify adherence to requirements, test the edge cases for payment handling, and assess the robustness of error handling in all three scenarios.

Prepared by:
Farshad Moradi Shahrbabak
Date: June 5, 2025