**Project:** Pixelz Portal – Order Checkout Module  
**Author:** Đinh Nhật Minh  
**Date:** July 4, 2025

**🧭 Table of Contents**

1. [**Introduction**](#1-introduction)
2. [**Modules Overview**](#2-modules-overview)
3. [**Module 1: Search Order**](#3-module-1-search-order)
   * **3.1 Business Context**
   * **3.2 Functional Requirements**
   * **3.3 API Specification**
   * **3.4 Use Case Flow**
   * **3.5 Business Rules**
4. [**Module 2: Checkout Order**](#4-module-2-checkout-order)
   * **4.1 Business Context**
   * **4.2 Functional Requirements**
   * **4.3 API Specification**
   * **4.4 Use Case Flow**
   * **4.5 Post-payment Integrations**
   * **4.6 Business Rules**
5. [**Common Assumptions & Constraints**](#5-assumptions--constraints)
6. [**Glossary**](#6-glossary)

**1. Introduction**

This document defines the business and functional specifications for the Order Search and Checkout functionality in the Pixelz Portal. It serves as a foundation for development, testing, and delivery of the feature.

**2. Modules Overview**

| **Module** | **Description** |
| --- | --- |
| Search Order | Allow users (Art Directors) to find existing orders by name |
| Checkout Order | Enable users to submit payment for an order, followed by email, invoice, and internal system integration |

**3. Module 1: Search Order**

**3.1 Business Context**

Users (Art Directors) need to locate previously created orders in the system in order to proceed with actions such as reviewing or submitting checkout.

**3.2 Functional Requirements**

| **ID** | **Description** |
| --- | --- |
| SR-01 | The system shall allow filtering orders by name (case-insensitive) |
| SR-02 | The system shall support partial string matching |
| SR-03 | The system shall return basic order information: ID, name, amount, status |

**3.3 API Specification**

**GET /orders?name=...**

**Request parameters:**

* **name (query): Partial name of the order**

**Response (200):**

**json**

**CopyEdit**

**[**

**{**

**"id": "uuid",**

**"name": "Summer Campaign",**

**"status": "PENDING",**

**"amount": 150.00**

**}**

**]**

**3.4 Use Case Flow**

| **Step** | **Actor** | **Action** | **System Behavior** |
| --- | --- | --- | --- |
| 1 | User | Enters keyword in search bar | Calls GET /orders?name=... |
| 2 | Backend | Filters and returns matches | Returns matching order list |
| 3 | User | Views the list | Chooses one to proceed to checkout |

**3.5 Business Rules**

| **Rule ID** | **Rule** |
| --- | --- |
| SR-BR-01 | Search must ignore case and support partial match |
| SR-BR-02 | Orders must be sorted by creation date descending |

**4. Module 2: Checkout Order**

**4.1 Business Context**

Once an order is selected, the user can initiate a checkout process. If payment is successful, the system performs post-payment actions automatically.

**4.2 Functional Requirements**

| **ID** | **Description** |
| --- | --- |
| CR-01 | The system shall allow initiating checkout via POST /orders/:id/checkout |
| CR-02 | Only orders in PENDING status are allowed to be checked out |
| CR-03 | System shall call mocked Payment Service |
| CR-04 | If payment succeeds: • Update order to PAID • Send confirmation email • Generate invoice • Push order to Production System |
| CR-05 | If payment fails: • Update order to FAILED • Return failure message |
| CR-06 | All interactions must be logged for auditability |

**4.3 API Specification**

**POST /orders/:id/checkout**

**Request:  
None (order ID in URL)**

**Success Response (200):**

**json**

**CopyEdit**

**{**

**"success": true,**

**"message": "Payment successful",**

**"orderStatus": "PAID"**

**}**

**Failure Response (400/402):**

**json**

**CopyEdit**

**{**

**"success": false,**

**"message": "Payment failed or order already paid"**

**}**

**4.4 Use Case Flow**

**User → Selects order → POST /orders/:id/checkout**

**→ Validate order → Call Payment**

**→ If Failed: update to FAILED → return error**

**→ If Success:**

**- update to PAID**

**- send email**

**- create invoice**

**- push to Production**

**4.5 Post-payment Integrations**

| **Service** | **Type** | **Behavior** |
| --- | --- | --- |
| Payment Service | Mock | Simulates success or failure |
| Email Service | Mock | Sends confirmation email |
| Invoice Service | Mock | Creates invoice record |
| Production System | Mock | Accepts pushed order for internal processing |

**4.6 Business Rules**

| **Rule ID** | **Rule** |
| --- | --- |
| CR-BR-01 | An order may only be checked out once |
| CR-BR-02 | If post-payment integrations fail, order remains PAID, and failure is logged |
| CR-BR-03 | Checkout is not allowed if order is PAID or FAILED |

**5. Assumptions & Constraints**

* **Authentication is out of scope**
* **All external services are mocked**
* **No frontend or UI design is included**
* **Data persistence assumed via in-memory DB or mock DB**
* **Payment success/failure is deterministic (based on mock rules)**

**6. Glossary**

| **Term** | **Meaning** |
| --- | --- |
| Checkout | Submitting an order for payment and processing |
| Order | A set of assets (e.g., images) awaiting post-production |
| PAID | Order has been paid successfully |
| FAILED | Payment or validation failed |
| PENDING | Order is created but not paid |