DESIGN PATTERNS
CSYE 7374

Ecommerce Management System

Presented By:

- Bhavya Prakash – 002774235
- Tiyasha Sen – 002727486
- Aditya Ramesh Mysore – 002774017
- Ridham Modh – 002752204

Technologies:

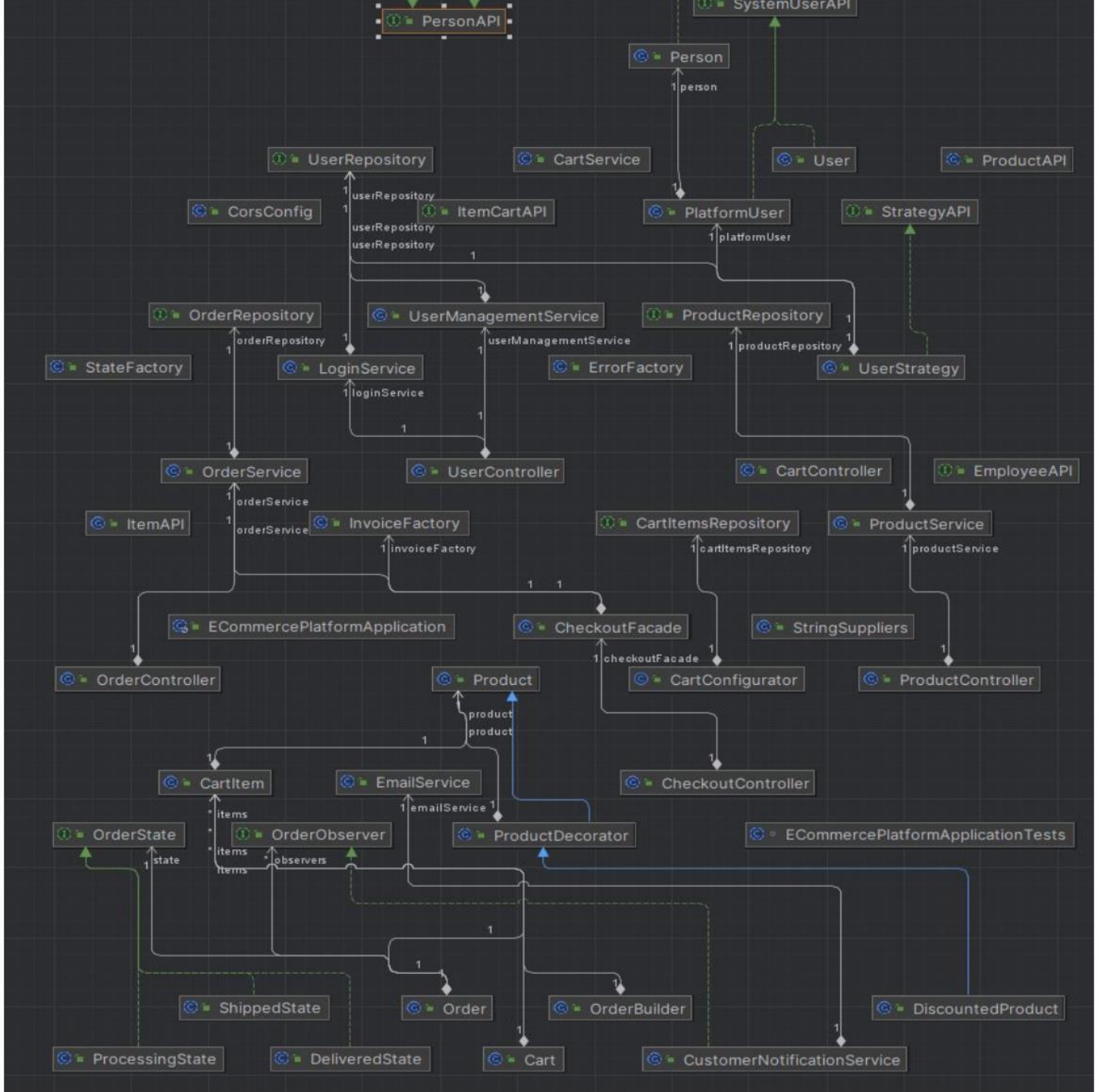
Frontend:
React

Backend:
Spring Boot

Database:
Postgresql

Database
Connectivity:
Hibernate

Server:
Apache
Tomcat



Class Diagram

Builder

- The Builder Design Pattern is used to construct complex objects step by step. It separates the construction of a product from its representation, allowing the same construction process to create various representations
- Implemented class OrderBuilder to build Order objects with Items added to the Shopping Cart.

Decorator

- The Decorator Design Pattern allows behavior to be added to individual objects, either statically or dynamically, without affecting the behavior of other objects from the same class.
- Designed abstract class ProductDecorator.
- Implemented DiscountProductDecorator which allows us to apply price discounts on individual products within the system.

Facade

- The Facade Design Pattern provides a simplified interface to a complex subsystem, making it easier to interact with by hiding the underlying complexity.
- Created class CheckoutFacade to simplify and abstract the checkout, email notification and invoice generation process.

Factory

- The Factory Design Pattern provides an interface for creating objects in a superclass, but allows subclasses to alter the type of objects that will be created.
- Designed multiple API error messages Factory classes and Order State Factory classes.

Observer

- The Observer Design Pattern establishes a one-to-many relationship between objects, ensuring that when one object changes its state, all dependent objects are automatically notified and updated.
- When a order's state is changed the user gets notified by the CustomerNotificationService class which implements OrderObserver interface.

Singleton

- The Singleton Design Pattern ensures that a class has only one instance while providing a global point of access to that instance. This pattern is commonly used to manage access to a shared resource across multiple parts of an application.
- Developed Cart using singleton pattern for holding product items through the order lifecycle.

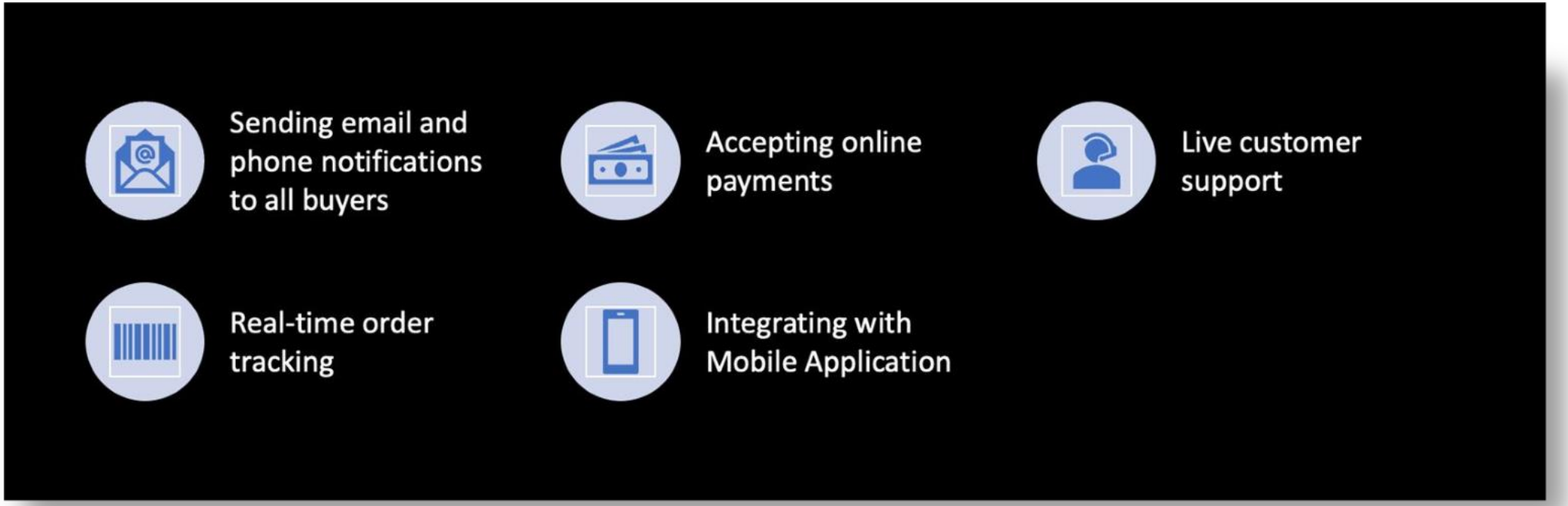
Strategy

- The Strategy Design Pattern enhances code flexibility, maintainability, and readability by enabling the dynamic selection of algorithms, promoting separation of concerns, and supporting the development of reusable and testable code.
- Created class User strategy which implements StrategyAPI which implements methods such as add, update and delete user for user management purpose.

State

- The State Design Pattern enables an object to change its behavior when its internal state shifts, enhancing modularity, scalability, and readability.
- Using this Pattern to check the order status whether it is Shipped, Processing Or Delivered after placing the order.
- Allowing User to have visibility into this order status and enabling system admin to change the status of the order.

Future Enhancements



TEAM CONTRIBUTIONS

Team Member	Contribution
Ridham Modh	State Design Pattern, Strategy Design Pattern, Backend
Bhavya Prakash	Builder Design Pattern, Decorator Design Pattern, Frontend
Tiyasha Sen	Facade Design Pattern, Factory Design Pattern, Backend
Aditya Ramesh Mysore	Observer Design Pattern, Singleton Design Pattern, Frontend

Thank You