Pizza Restaurant Online Ordering System

Group 2

Christian Gilmore, Devon Galloway, Sera J. , Frank ,Nick

3/30/2025

Table of Contents

Project Vision and Description:

The owners of a local pizzeria have approached our team with an exciting project: developing a responsive and user-friendly website. This new site will serve as a comprehensive resource for the restaurant, featuring a detailed description of their offerings, mouthwatering photos, a complete menu with pricing, and up-to-date information on promotions and discounts.

Our goal is to create an intuitive experience that allows customers to navigate the site effortlessly. They will be able to place orders tailored to their preferences, and once an order is completed, the website will automatically calculate and display the final bill. Secure payment processing will also be integrated for a seamless transaction. Additionally, the website will empower store managers by providing flexibility to modify menu items as needed, ensuring that the information remains current and accurate.

Team Roles:

Scrum Master: The Scrum Master will play a crucial role in overseeing the project, ensuring that the team embraces cross-functionality. They will help identify and eliminate any obstacles that may arise, while also fostering an environment where Scrum events are positive, productive, and completed within their designated time frames.

Product Owner: The Product Owner will efficiently manage the backlog by prioritizing items, creating tasks, and ensuring transparency. They will also communicate the product goals clearly to all parties involved.

Developers: The Developer team will be responsible for creating a comprehensive sprint plan that aligns with the expected quality standards to meet the Definition of Done requirements. This team will focus on developing both the front-end and back-end components of the website, as well as testing project items to ensure they function effectively and meet design specifications.

Collaboration Methodology:

To ensure efficient workflows and clear communication, our team will implement the following tools:

- **Communication Platform:** We'll use Microsoft Teams for our daily discussions, keeping everyone connected and informed.
- **Meetings:** We will hold weekly stand-up meetings on Microsoft Teams to review sprint progress, address any roadblocks, and discuss new story changes.
- **Version Control:** GitHub will be our go-to platform for managing code and tracking changes effectively.
- **Task Management:** To monitor our sprint progress and assign tasks, we will utilize Trello, which will help us stay organized.

The Definition of "Done":

The project will be considered done when:

Performance:

- The website loads within 3 seconds.
- The system can handle the expected load of customers (average customers per day + 50%)

Functionality:

- All acceptance criteria for the customer and store managers stories are met and demonstrated.
- All planned test cases pass.
- The website, and its interactions work correctly on all supported browsers.
- Early prototype testing has been conducted and any feedback addressed.
- Regression testing has been performed, and no existing functionality is broken.

Security:

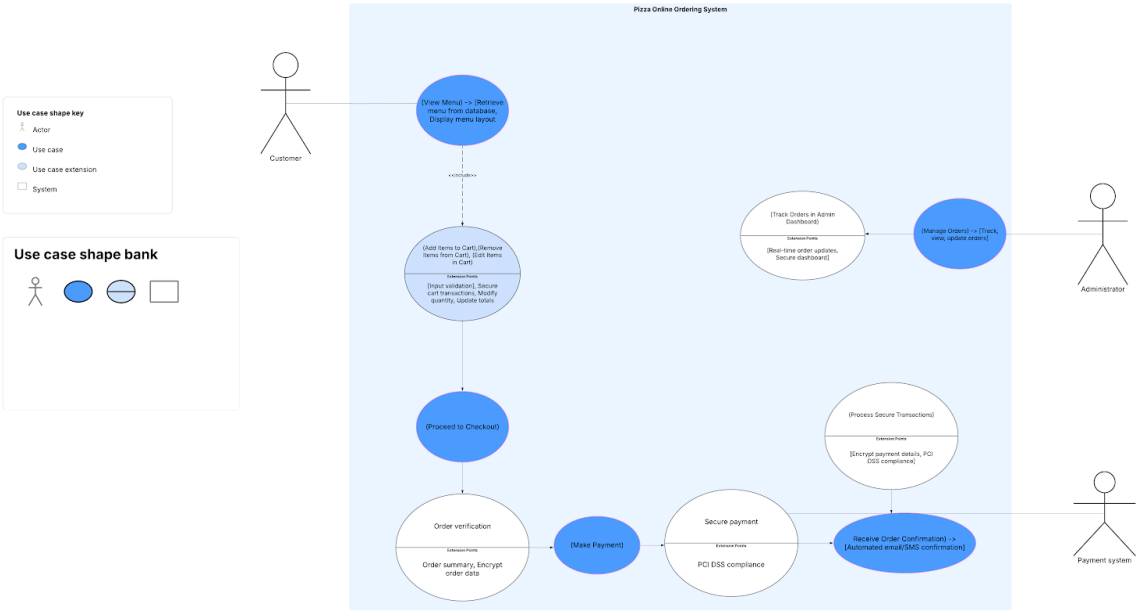
- Security best practices are followed.
- Vulnerabilities identified in testing are addressed.
- Payment processing complies with PCI DSS standards.
- Data is encrypted both in transit (HTTPS) and at rest.

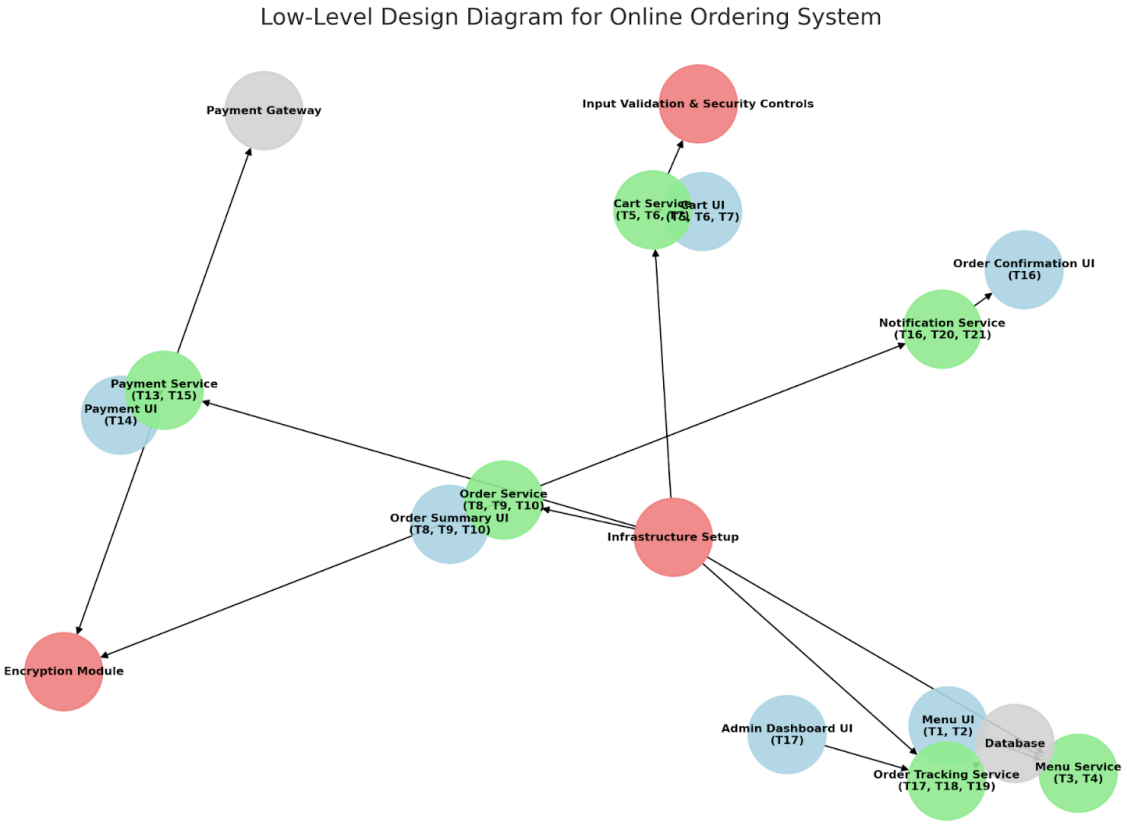
Product:

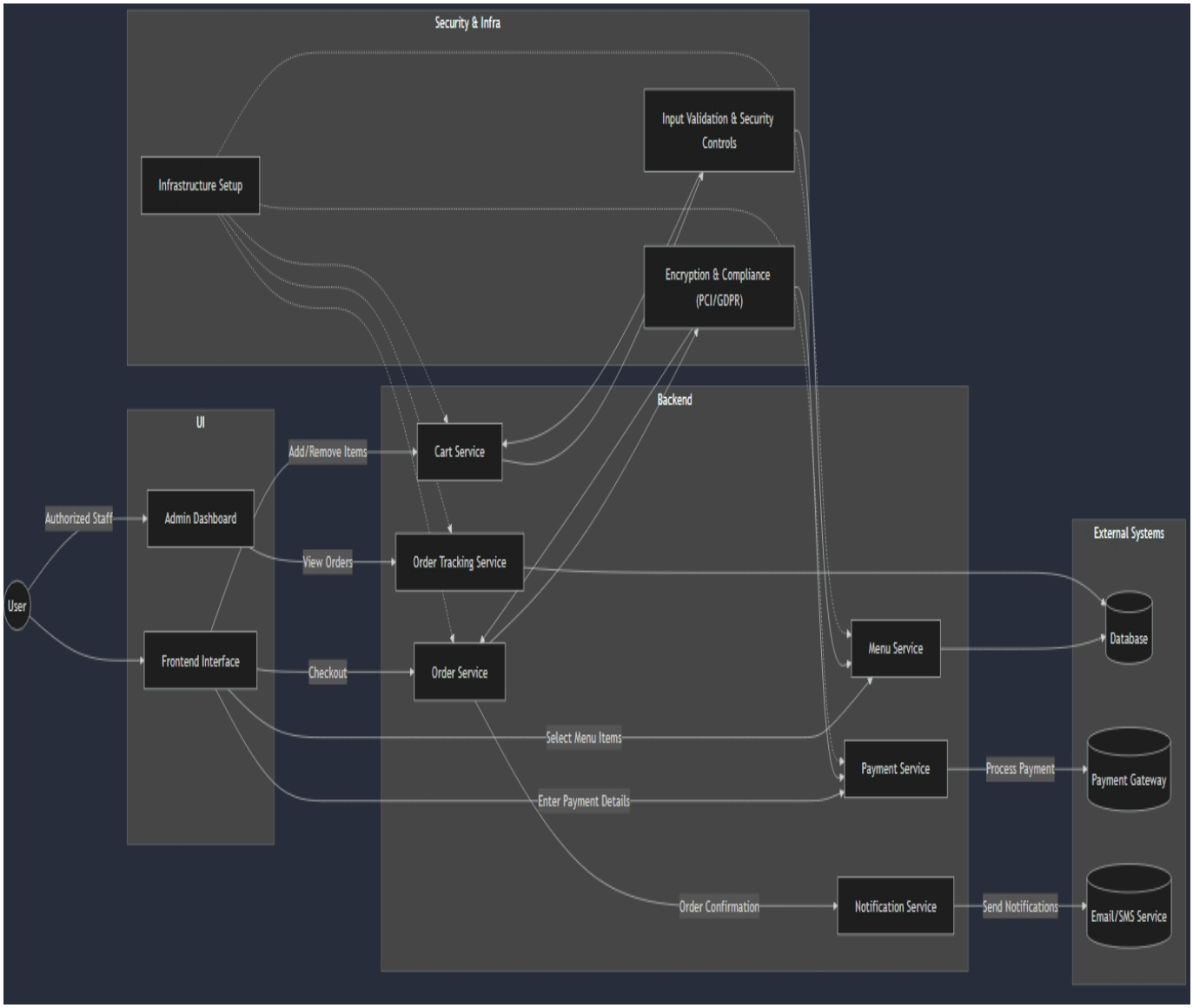
- The website has been deployed to a staging/testing environment first.

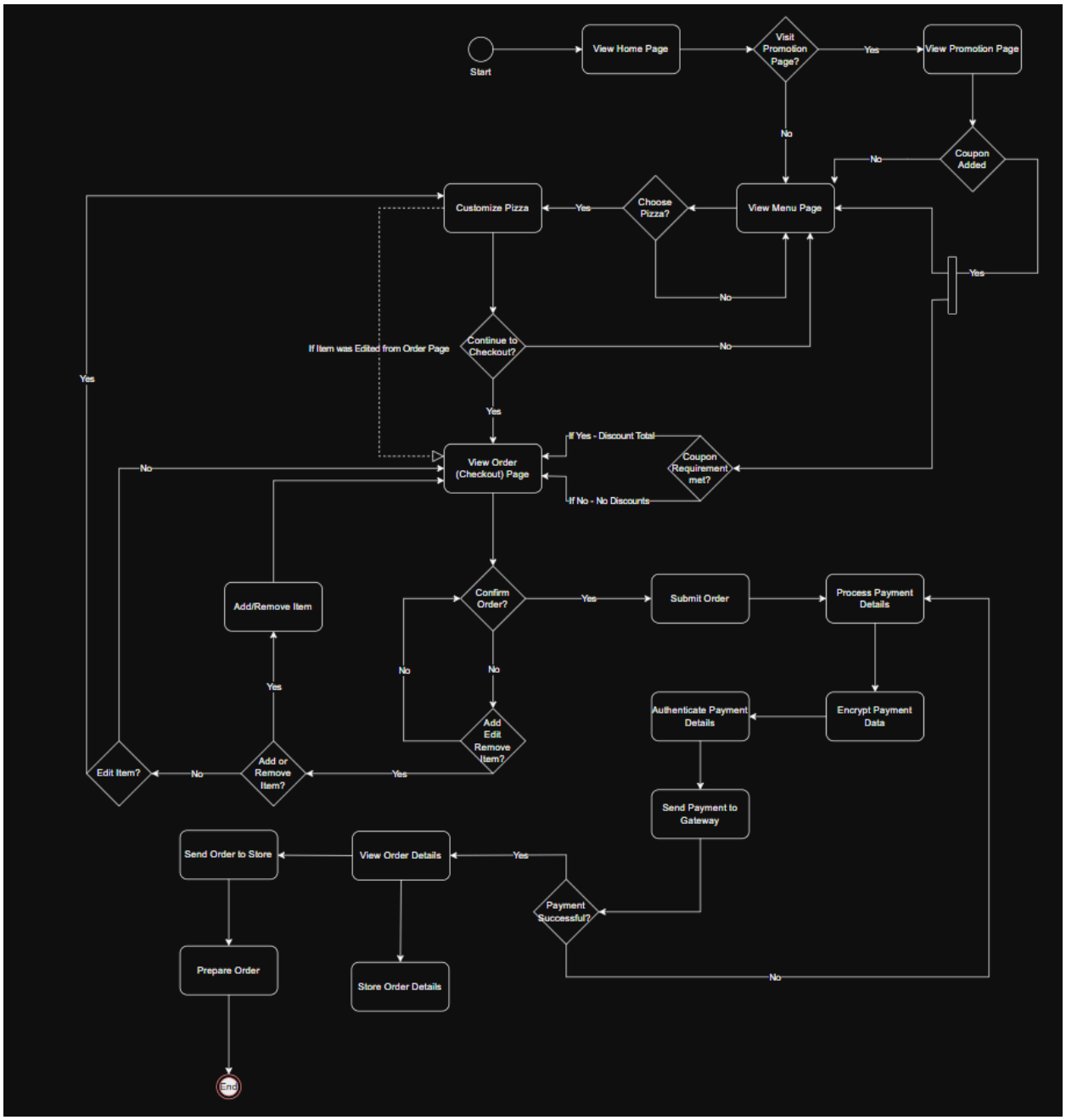
- The deployment process is automated and well documented.
- The staging environment closely mirrors the production environment.

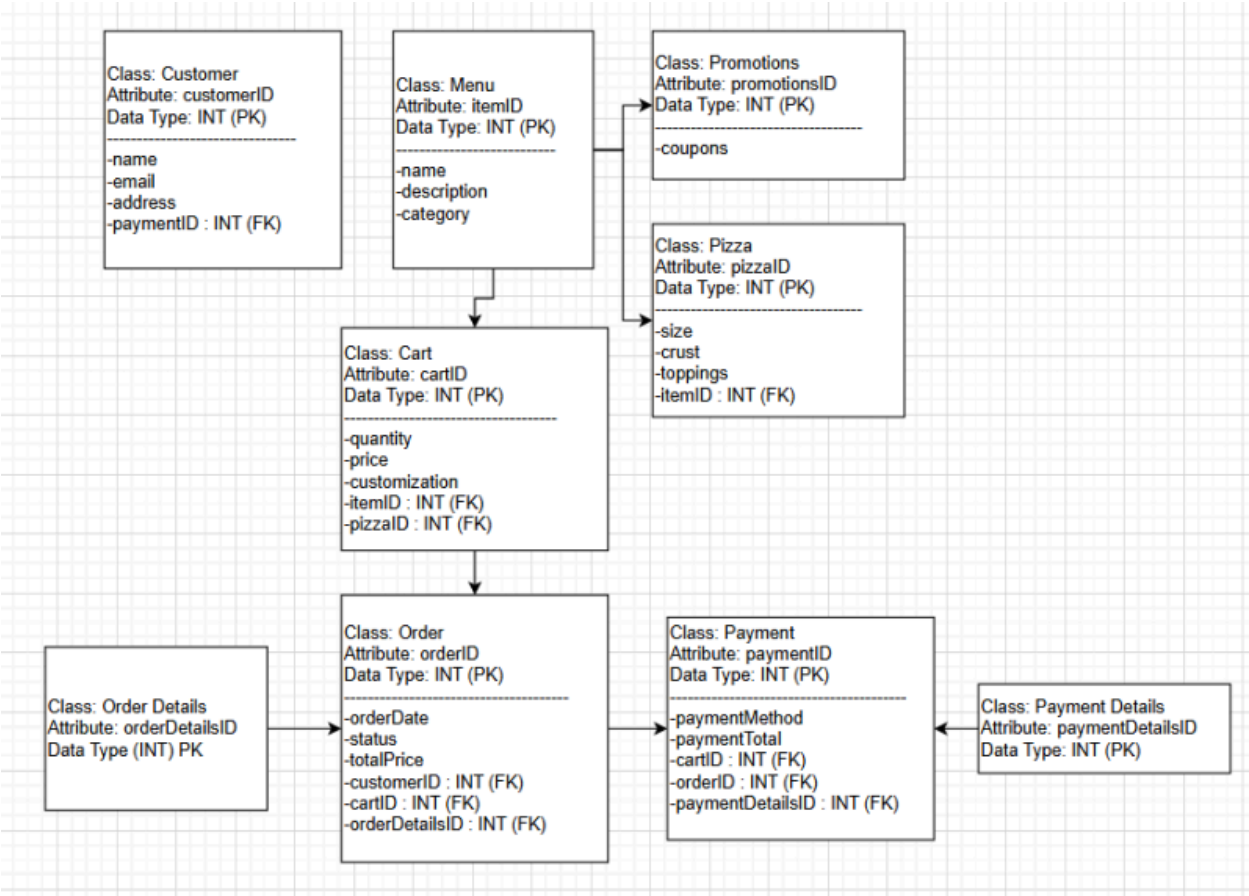
Product Design:






Security & Infra
Input Validation & Security Controls
Infrastructure Setup
Encryption & Compliance (PCI/GDPR)
Backend
UI
Add/Remove Items
Cart Service
Authorized Staff
Admin Dashboard
External Systems
View Orders
Order Tracking Service
User
Database
Frontend Interface
Checkout
Order Service
Menu Service
Select Menu Items
Payment Service
Process Payment
Payment Gateway
Enter Payment Details
Order Confirmation
Notification Service
Send Notifications
Email/SMS Service


Start
View Home Page
Visit Promotion Page?
Yes
View Promotion Page
No
Coupon Added
No
Customize Pizza
Yes
Choose Pizza?
View Menu Page
Yes
No
If Item was Edited from Order Page
Continue to Checkout?
No
Yes
Yes
If Yes - Discount Total
View Order (Checkout) Page
Coupon Requirement met?
No
If No - No Discounts
Add/Remove Item
Confirm Order?
Yes
Submit Order
Process Payment Details
No
No
Yes
Add Edit Remove Item?
Authenticate Payment Details
Encrypt Payment Data
Edit Item?
No
Add or Remove Item?
Yes
Send Payment to Gateway
Send Order to Store
View Order Details
Yes
Payment Successful?
Prepare Order
Store Order Details
No
End


Class: Customer
Attribute: customerID
Data Type: INT (PK)
-name
-email
-address
-paymentID : INT (FK)
Class: Menu
Attribute: itemID
Data Type: INT (PK)
-name
-description
-category
Class: Promotions
Attribute: promotionsID
Data Type: INT (PK)
-coupons
Class: Pizza
Attribute: pizzaID
Data Type: INT (PK)
-size
-crust
-toppings
-itemID : INT (FK)
Class: Cart
Attribute: cartID
Data Type: INT (PK)
-quantity
-price
-customization
-itemID : INT (FK)
-pizzaID : INT (FK)
Class: Order Details
Attribute: orderDetailsID
Data Type (INT) PK
Class: Order
Attribute: orderID
Data Type: INT (PK)
-orderDate
-status
-totalPrice
-customerID : INT (FK)
-cartID : INT (FK)
-orderDetailsID : INT (FK)
Class: Payment
Attribute: paymentID
Data Type: INT (PK)
-paymentMethod
-paymentTotal
-cartID : INT (FK)
-orderID : INT (FK)
-paymentDetailsID : INT (FK)
Class: Payment Details
Attribute: paymentDetailsID
Data Type: INT (PK)

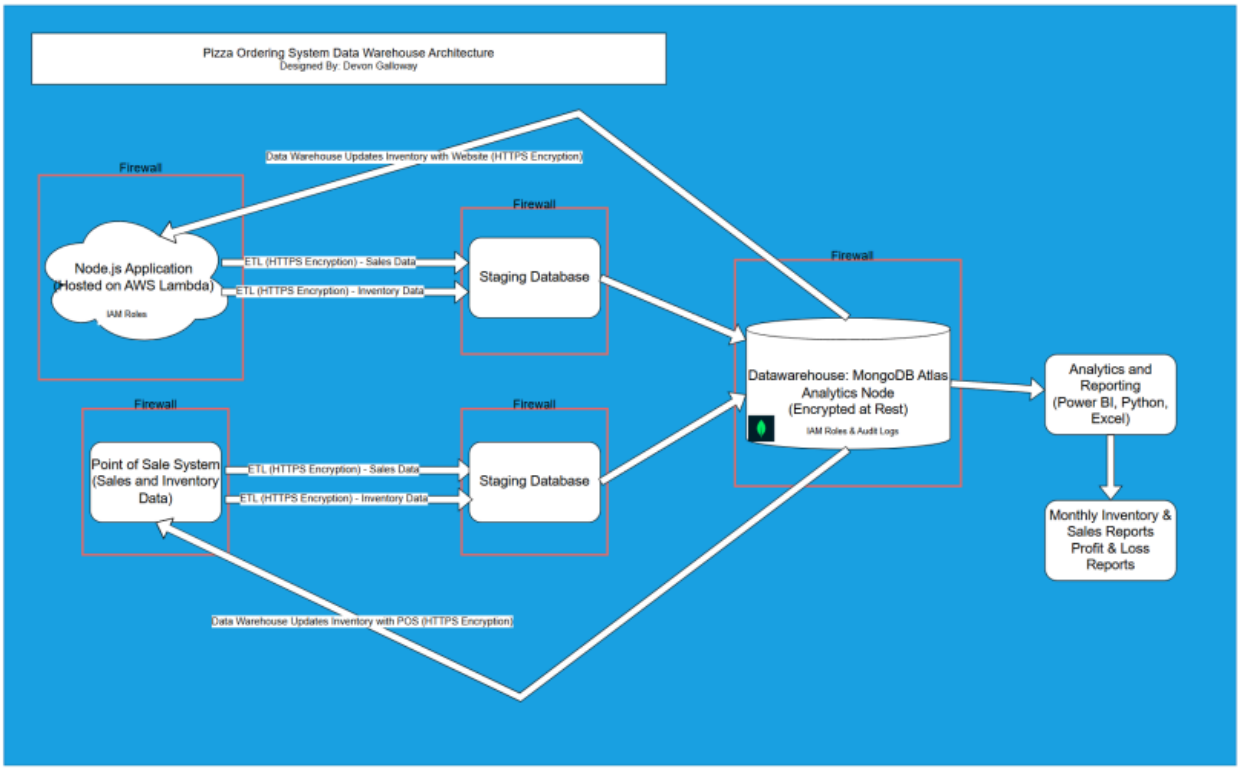

Pizza Ordering System Data Warehouse Architecture
Designed By: Devon Galloway
Data Warehouse Updates Inventory with Website (HTTPS Encryption)
Firewall
Node.js Application (Hosted on AWS Lambda)
IAM Roles
ETL (HTTPS Encryption) - Sales Data
ETL (HTTPS Encryption) - Inventory Data
Firewall
Staging Database
Firewall
Datawarehouse: MongoDB Atlas Analytics Node (Encrypted at Rest)
IAM Roles & Audit Logs
Analytics and Reporting (Power BI, Python, Excel)
Monthly Inventory & Sales Reports Profit & Loss Reports
Firewall
Point of Sale System (Sales and Inventory Data)
ETL (HTTPS Encryption) - Sales Data
ETL (HTTPS Encryption) - Inventory Data
Firewall
Staging Database
Data Warehouse Updates Inventory with POS (HTTPS Encryption)

Diagram Description:

Diagram 1: Product Design and User Stories

This diagram offers a visual breakdown of the product's functional and security requirements for the Cross-Country Pizzeria Online Ordering System. At its core, the diagram maps user stories to specific system functionalities that directly relate to the pizza ordering experience. For instance, it begins with a user story labeled P1, which details the customer's need to browse the restaurant's menu. The diagram highlights that the menu must display food categories, descriptions, prices, and images, and it emphasizes accessibility on both desktop and mobile platforms. This component directly informs the sprint plan for tasks such as designing the menu layout (T1) and implementing the menu display (T2) in Sprint 1.

Moreover, the diagram delineates subsequent user stories like P2 and P3, which cover the process of placing an order and reviewing the order before finalization. These segments illustrate how customers will add items to their cart, customize options (like pizza toppings and crust), and receive a dynamically updated order summary. Each acceptance criterion is depicted, ensuring that the functional aspects—such as dynamic price updates and cart modifications—are incorporated into the sprint tasks (T5–T10). Additionally, security considerations are integrated alongside each user story; for example, P1 and P2 include safeguards against unauthorized modifications and SQL injections. The diagram shows that encryption (for order data and payment information) and input validation are critical, aligning directly with Sprint 1's and Sprint 2's tasks related to data security (T4, T15, T21).

Overall, this diagram is pivotal to the entire project because it bridges the gap between high-level business objectives and detailed technical implementation. It provides a comprehensive blueprint that ensures every feature—from the user-friendly interface to secure transaction handling—is meticulously planned and executed. By aligning with the project vision, this diagram guarantees that the final system not only meets customer expectations but also adheres to rigorous security standards, thereby forming the backbone of the online ordering experience.

Diagram 2: Sprint Planning and Task Breakdown

The Sprint Planning and Task Breakdown diagram is a detailed visualization of how the project is organized across two primary sprints. This diagram serves as a roadmap for the iterative development of the pizza ordering system, breaking down the project into manageable segments that align with both functional goals and security standards.

In Sprint 1, the diagram illustrates tasks related to establishing the core functionalities of the online ordering system. It begins with initial planning activities, such as team meetings and infrastructure setup

(T11 and T12), followed by a sequence of development tasks focused on building the user interface. Key tasks include designing the menu layout (T1), implementing the menu display (T2), and developing a system for retrieving menu data from the database (T3). Further, Sprint 1 incorporates tasks that enhance the user experience, such as developing the cart functionality (T5, T6, T7) and creating an order summary page (T8, T9, T10). Each task is assigned a specific time estimate, which helps in gauging the project's progress and ensuring that resource allocation aligns with the project timeline.

Sprint 2 is dedicated to refining and extending the system's capabilities, with a significant emphasis on security and administrative features. The diagram shows tasks like integrating a secure payment gateway (T13), developing a secure payment entry form (T14), and implementing PCI DSS compliance measures (T15). Additionally, Sprint 2 includes tasks for creating an admin dashboard for order tracking (T17, T18, T19) and automating email/SMS confirmations (T20, T21). The clear demarcation between Sprint 1 and Sprint 2 in the diagram ensures that foundational functionalities are completed before advanced features are implemented. This staged approach minimizes risk and allows for thorough testing and validation of each component.

In relation to the overall project, this diagram is essential as it structures the entire development lifecycle. It ensures that every phase—from planning to execution—is methodically organized, promoting transparency, efficient task management, and adherence to both functionality and security requirements. This clear roadmap directly supports the project vision by ensuring that each sprint builds upon the previous one, leading to a robust and comprehensive online ordering system.

Diagram 3: Retrospective and Lessons Learned

The Retrospective and Lessons Learned diagram encapsulates the reflective process that the team undertakes at the end of each sprint. It is designed to visually organize feedback into three main categories: things that went well, areas for improvement, and surprising outcomes. The diagram is structured to compare and contrast these elements across the two sprints, providing clear insight into how each phase of development contributed to the overall progress of the pizza ordering system.

For Sprint 1, the diagram displays positive outcomes, such as the successful completion of core functionalities—like menu design, cart development, and order summary creation—as well as effective initial communication and planning. However, it also identifies challenges, such as issues with overly granular task breakdowns and coordination difficulties during the early phases. These insights are linked

directly to specific sprint tasks (T1–T12), emphasizing the importance of establishing a solid development foundation.

In the Sprint 2 section, the diagram shifts focus to advanced features and security enhancements. It highlights successes in implementing robust security measures, such as PCI DSS compliance for payment processing and the development of an intuitive admin dashboard for order tracking. At the same time, the diagram notes areas where further improvements could be made, including refining real-time update mechanisms and optimizing the feedback loop between developers and testers. The lessons learned from Sprint 2 are documented as actionable insights that can be applied to future iterations.

This retrospective diagram is critically related to the overall project because it fosters continuous improvement—a cornerstone of agile methodologies. By capturing both successes and challenges, the diagram not only serves as a record of the team's progress but also guides strategic adjustments for subsequent sprints. It ensures that the online ordering system evolves in response to practical feedback, ultimately contributing to a more refined, secure, and user-friendly final product.

Diagram 4: Devon Galloway Activity Diagram

This activity diagram provides a step-by-step visualization of the user journey within the Pizza Restaurant Online Ordering System, starting from the moment a user visits the home page to the point where an order is prepared and stored. The flow begins at the **Home Page**, where users can either view promotions or proceed to the menu. If they choose to explore the menu, the diagram shows how they can **select a pizza**, **customize it** with toppings or crust options, and then decide whether to continue browsing or move to the **Checkout** process.

A key branch appears when the user finishes customizing their pizza: they can either edit the item further or proceed to the checkout page. Upon reaching the **Checkout** page, the diagram highlights how users confirm their order and view the **cart total**, ensuring accuracy before finalizing. Once the user clicks **Submit Order**, the diagram transitions to payment-related activities, illustrating how **payment details are processed**, **encrypted**, and **sent to the payment gateway**. This underscores the system's focus on security by depicting encryption as a mandatory step before communicating with external services.

Following successful payment, the **order details** are routed to the store, where the pizza is prepared. Meanwhile, the user can view **order details**—such as order status or estimated preparation time—until the process concludes with the system **storing** all relevant data. By mapping out each action and decision point, this diagram clarifies the dependencies and logical flow among tasks. It also reflects key user stories from the project, including the need for secure payment processing, dynamic cart management, and real-time order updates. Overall, the activity diagram not only captures the core functionality of the online ordering experience but also ensures that each step is traceable, secure, and aligned with the system's overarching requirements.

Diagram 5: Sera J class diagram

This diagram shows the data model for the Pizza Restaurant Online Ordering System, illustrating how key entities in the application relate to one another. Below is a detailed explanation of each class and how it supports the overall project:

1. Customer
    - Attributes: customerID (primary key), address, paymentID (foreign key)
    - Purpose: Represents a user in the system who can place orders. Linking payment ID allows the application to reference specific payment methods or records associated with that customer.
2. Menu
    - Attributes: itemID (primary key), name, description, category
    - Purpose: Stores information about general menu items, such as appetizers, drinks, and desserts. This allows the system to display all available items to both customers and store managers.
3. Promotions
    - Attributes: promotionID (primary key), coupons
    - Purpose: Manages data about promotions or coupons. These can be applied to orders at checkout for discounts or special deals, helping to boost sales and offer marketing incentives.
4. Pizza
    - Attributes: pizzaID (primary key), size, toppings, crust, itemID (foreign key)
    - Purpose: Represents pizza-specific menu items. By referencing the item ID, each pizza entry is tied back to the Menu class, ensuring a consistent way to track both general menu items and custom pizza configurations.
5. Cart

- Attributes: cartID (primary key), quantity, price, customization, itemID (foreign key), pizzaID (foreign key), and possibly, userID or customerID (not explicitly shown but implied)
- Purpose: Holds the items that a user selects before finalizing an order. By linking itemID or pizzaID, the cart can contain either general menu items or pizzas with special customizations. Quantity, price, and other details are managed here until the user proceeds to checkout.

6. Order

    - Attributes: orderID (primary key), totalPrice, customerID (foreign key), orderDetailsID (foreign key)
    - Purpose: Represents a finalized purchase. When a user checks out, the cart information is transferred into an Order record. It references the Customer who placed the order and uses orderDetailsID to store or retrieve specific details about the items purchased.
7. Payment

    - Attributes: paymentID (primary key)
    - Purpose: Stores essential payment information. This links to one or more Orders or Customers, ensuring the system knows how a particular order was paid for, whether by credit card, digital wallet, or another method.
8. Payment Details

    - Attributes: orderDetailsID (primary key)
    - Purpose: Holds detailed transaction data for a given order, such as amounts, timestamps, and transaction IDs. This structure allows the system to maintain a clear audit trail for refunds or compliance with financial regulations.

How it relates to the overall project:

• Data Integrity and Relationships: Each major aspect of the ordering process (menu, cart, orders, payments) is captured and linked through primary and foreign keys. This structure ensures that data is stored consistently and that each entity can be accurately referenced without duplication.

• Customization and Promotions: By separating Menu, Pizza, and Promotions, the application can handle both general items and specialized pizza configurations while also allowing for flexible marketing campaigns. This supports user stories that require customized pizza orders or coupon applications at checkout.

• Secure Transactions: Payment and Payment Details are stored in a way that keeps financial information separate from other customer data. This supports compliance with PCI DSS by allowing sensitive data to be accessed and secured appropriately.

• User-Friendly Experience: The Cart class is dedicated to the temporary storage of items before they become part of an Order, reflecting how users add items, review them, and finalize purchases straightforwardly.

• Scalability and Maintenance: By modularizing the system into these distinct classes, new features can be introduced without overhauling the existing data model. Future sprints could add more menu categories, integrate additional payment methods, or enhance order-tracking features.

Overall, this data model underpins the entire online ordering workflow, ensuring that each user story—from menu browsing and cart management to payment processing and order fulfillment—is fully supported. It aligns with the system's sprint plans and backlogs by defining clear data relationships for each functional requirement, making it easier to maintain data integrity, security, and usability across the platform.

Diagram 6: Devon Galloway Wharehouse architecture Diagram

The diagram illustrates a secure and scalable data warehouse architecture for an online pizza ordering system designed to integrate digital and physical sales channels. At the core of the architecture is a Node.js application hosted on AWS Lambda, which serves as the backend for the online ordering platform. This application sends encrypted sales and inventory data using ETL (Extract, Transform, Load) processes to a staging database via HTTPS. Simultaneously, a physical Point of Sale (POS) system in stores also transmits sales and inventory data through its ETL pipeline to a separate staging database. These staging databases temporarily store raw or semi-processed data, allowing for validation, transformation, and cleaning before the data is loaded into the main warehouse.

The centralized data warehouse is built on MongoDB Atlas and includes an analytics node with encryption at rest, IAM roles, and audit logs to ensure security and compliance. This warehouse aggregates data from the online system and POS terminals, synchronizing inventory levels across all platforms. The warehouse also updates the website and POS systems with the most current inventory data, maintaining consistency throughout the business. Firewalls are strategically placed between each component to protect the system from unauthorized access.

For business insights, the data warehouse connects to reporting and analytics tools such as Power BI, Python, and Excel, which generate real-time dashboards, sales analysis, inventory trends, and other business intelligence outputs. Monthly reports, including inventory summaries, sales reports, and profit and loss statements, are generated from this data to aid in strategic decision-making. Overall, this architecture provides a robust and secure foundation for efficiently managing and analyzing sales and inventory data across online and physical pizza ordering channels, ensuring seamless operations and data-driven business growth.

Sprint 1 Retrospective Summary Report:

Things That Went Well:

Things That Could Have Gone Better:

Things That Surprised Us:

Lessons Learned:

Sprint 2 Retrospective Summary Report:

Things That Went Well:

Things That Could Have Gone Better:

Things That Surprised Us:

Lessons Learned:

References:

The 2020 scrum GUIDETM. Scrum Guide | Scrum Guides. (n.d.).

https://scrumguides.org/scrum-guide.html