DoughDoughs Online Ordering System

Computer Science Team Project I: CS491

Group D: Emma Monroy-Rincon, Wesley McElhinny, Michael McCoy, Antoine Gaton

November 3, 2024

Table of Contents

[1. Project Vision and Description 3](#_Toc179736654)

[2. Team Roles 4](#_Toc179736655)

[3. Collaboration Methodology 5](#_Toc179736656)

[4. The Definition of “Done” 5](#_Toc179736657)

[5. Product Design 6](#_Toc179736658)

[6. Sprint 1 Retrospective Summary Report 7](#_Toc179736659)

[7. Sprint 2 Retrospective Summary Report 8](#_Toc179736660)

[8. References 9](#_Toc179736661)

# Project Vision and Description

* **Vision Statement**: DoughDoughs aims to create a delightful pizza ordering experience by providing a modern, user-friendly online platform that reflects our passion for quality food and exceptional service. Our website will serve as a digital hub for customers to explore our menu, enjoy seamless ordering, and take advantage of exclusive promotions—all while fostering a strong connection to our community.
* **Product Description**: The DoughDoughs Online Ordering System is an innovative web platform designed to enhance customer experience at DoughDoughs Pizzeria. This responsive and intuitive website will provide a comprehensive view of our offers, enabling customers to place customized orders with ease.
* **Key Features**
  + **User-Friendly Interface:** The website will be designed with a focus on simplicity and accessibility, ensuring customers can navigate effortlessly to find what they need, whether on a desktop, tablet, or mobile device.
  + **Detailed Menu Presentation:** Customers can explore our full menu, featuring high-quality images and detailed descriptions of each item, along with prices. The site will highlight current discounts and promotions, enticing users to try new flavors.
  + **Customizable Ordering:** Users will have the ability to customize their pizza and other menu items according to personal preferences, allowing them to create their perfect meal.
  + **Real-Time Billing:** The system will automatically calculate and display the final bill based on the selected items, providing transparency and helping customers make informed choices during the ordering process.
  + **Secure Payment Processing:** A secure payment gateway will be integrated to handle transactions smoothly, offering customers various payment options for their convenience.
  + **Managerial Control:** The website will empower the store manager with the capability to update menu items, prices, and promotions in real time, ensuring the information remains current and relevant.
  + **Target Audience:** DoughDoughs Online Ordering System is tailored for pizza enthusiasts of all ages who value convenience, variety, and quality. It caters to local customers seeking a quick and easy way to enjoy delicious pizza, as well as tech-savvy individuals who prefer online ordering.
* **Conclusion**: The DoughDoughs Online Ordering System is designed to elevate the pizza ordering experience, combining user-friendly features with operational efficiency. By focusing on customization, transparency, and security, we will create a platform that not only showcases our love for pizza but also strengthens our connection with our community and enhances customer loyalty.

# Team Roles

* **Product Owner –** *Emma Monroy- Rincon***:** The Product Owner is responsible for managing the product backlog, ensuring resources, prioritizing features, representing stakeholders, budget reporting, creating user reports, clarifying requirements, removing obstacles, assessing performance, adding user stories, and preparing for the next sprint. In the DoughDoughs project, I will focus with my team on creating a delightful pizza ordering experience by coordinating resources, prioritizing features, creating user reports, clarifying requirements, removing obstacles, assessing performance, adding user stories, and preparing for the next sprint. This approach aims to keep the project aligned with its vision and effectively meet the needs of customers while fostering strong community ties.
* **Scrum Master –** *Michael McCory***:** The Scrum Master will facilitate the Scrum process by organizing sprints, running meetings, and ensuring that the team follows Agile practices. He will also work to remove obstacles that might block the team’s progress and maintain efficient communication between team members.
* **Lead Developer -** *Antoine Gaton***:** The lead developer is responsible for overseeing the entire development process of the DoughDoughs web application. The role includes setting technical directions, guiding the development team, and ensuring that the project meets its technical requirements and deadlines. The lead developer also collaborates with other stakeholders, like the product owner and scrum master, to align the technical aspects of the project with business goals.
* **Developer -** *Wesley McElhinny*: The role of the developer will act as a multi-tooled and self-organizing member of the product team. The developer will be responsible for creating and maintaining the GitHub repository. This member will also be accountable for creating the full-stack source code for the DoughDoughs web application. The developer will also be responsible for creating and executing test cases. The developer will also provide development expertise to the product owner and scrum master to develop proper code requirements.

# Collaboration Methodology

* DoughDough’s development team will be utilizing software-based tools for collaboration. The main means of collaboration for design will be using Discord. Discord is going to offer a platform to allow for voice chat, text chat, and document sharing. Within the team Discord server, several channels will be created and ensure proper organization of tasks. As for the application code development, DoughDough’s will be using the GitHub platform for configuration management. Within GitHub, we will run a main branch that will be the accepted code branch, then additional development branches. The development branches will then be merged based on the definition of done. The use of these two tools will allow our team to maintain an agile methodology for collaboration. This is due to the messaging app always being available, instead of just waiting for meetings. We will also be ensuring daily check-ins with our scrum team to communicate what has been done, what will be done and what needs help. GitHub will also allow the team to check off completion requirements of the code before being pushed to production or the main development branch.

# The Definition of “Done”

1. **Requirements Completion:** All functional and non-functional requirements specified in the project documentation are met
2. **Regression Testing:** Any code changes must pass a comprehensive regression test to ensure that existing functionality remains unaffected.
3. **GitHub Project Tracker:** The item must be marked as closed in the GitHub project tracker, indicating it has been reviewed and is no longer active.
4. **Product Owner Acceptance:** The completed task must be accepted and approved by the product owner, confirming that it meets their expectations and criteria for success.

This definition ensures that all aspects of the project are thoroughly completed, tested, and approved before considering a task or feature finished.

# Product Design

* **Use Cases**
  + *User Registration and Authentication*: Users can register for an account or log in with security measures such as password encryption and email verification to protect personal information.
  + *Menu Browsing and Filtering*: Users can browse a detailed menu with descriptions, images, and prices. Filtering options (e.g., vegetarian, gluten-free) enhance usability and allow users to quickly find suitable items.
  + *Customizable Ordering*: Users can customize their pizza by selecting options like size and toppings, with real-time updates to the pricing to reflect changes accurately.
  + *Order Checkout and Payment*: During checkout, users review their order and complete payment securely via integrated services like Stripe or PayPal. They can also access their order history for reordering.
  + A white background with black text

    Description automatically generated*Menu Management (Admin)*: Managers can add, update, or remove items and promotions. Role-based access control restricts menu management features to authorized users only, ensuring security and maintaining data integrity.
* **Class Diagram**
  + This class diagram represents the structure of a system with multiple classes and their relationships for an online ordering application. Here’s a brief overview of each class:
    - User:
      * Attributes: userId, name, email, phoneNumber, address, passwordHash.
      * Methods: browseMenu(), customizeItem(), placeOrder(), makePayment(), register(), login().
    - MenuItem:
      * Attributes: itemId, name, description, price, image.
      * Methods: getItemDetails(), applyCustomization(), updateMenuItem(), removeMenuItem().
    - Order:
      * Attributes: orderId, userId, items (list of MenuItem), totalAmount, orderStatus.
      * Methods: calculateTotal(), applyDiscount(), saveOrder(), trackOrder().
    - Payment:
      * Attributes: paymentId, orderId, paymentStatus, paymentMethod.
      * Methods: processPayment(), verifyPayment().
    - Cart:
      * Attributes: cartId, userId, items (list of MenuItem), subtotal.
      * Methods: addItem(), removeItem(), updateItemQuantity(), calculateSubtotal().
    - StoreManager:
      * Attributes: managerId, name, email.
      * Methods: addMenuItem(), updateMenuItem(), removeMenuItem(), viewSalesReport().
    - SalesReport:
      * Attributes: reportId, date, totalSales, itemsSold (list of MenuItem).
      * A screenshot of a computer

        Description automatically generatedMethods: generateReport(), viewReport().
  + The diagram shows how users can interact with various features like browsing and customizing menu items, managing orders, and making payments. The store manager’s class includes methods for managing the menu and viewing sales reports.
* **Activity Diagram**
  + The activity diagram demonstrates the order process flow:
    - The user logs in and browses the menu.
    - After choosing items, they customize their order and add it to their cart.
    - The system calculates the total price in real-time, then proceeds to checkout.
    - At checkout, the user reviews the order, selects a payment method, and confirms.
    - Upon successful payment, the system displays an order confirmation.

A diagram of a payment method

Description automatically generated

* + This diagram captures the user journey from login to order confirmation, detailing each interaction point with the system.
* High-Level Design
  + The DoughDoughs Online Ordering System’s architecture consists of the following components:
    - Frontend Web Application\*\*: Built with React to provide an interactive and responsive user interface on desktop and mobile devices.
    - Backend Server\*\*: Node.js server managing business logic, data processing, and secure data exchange.
    - \*\*Database\*\*: A relational database (e.g., PostgreSQL) that stores user data, order history, menu items, and transaction records.
    - - \*\*Payment Gateway Integration\*\*: Secure payment handling through Stripe and PayPal, protecting sensitive transaction data.
    - - \*\*Admin Management Portal\*\*: Restricted to managers, this portal allows menu updates, promotions, and order management through role-based access control.
* **Non-Functional Requirements**
  + Security:
    - Use HTTPS for all transactions.
    - Implement input validation to prevent SQL injection.
    - Encrypt sensitive user data.
  + Performance:
    - Optimize loading times for images and data retrieval.
    - Ensure the website is responsive across devices.
  + Usability:
    - Conduct user testing to refine the interface.
    - Provide clear navigation and support options.
  + Scalability:
    - Design the system architecture to accommodate future growth in user base and menu items.

# 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

* Kondura, S. (n.d.). *Role of a Product Owner in a Sprint Review*. Retrieved from Premier Agile: Auth0. (n.d.). *Best practices for user authentication and authorization*. Retrieved from https://auth0.com/docs/best-practices
* Fowler, M. (n.d.). *UML distilled: A brief guide to the standard object modeling language*. ThoughtWorks. Retrieved from https://martinfowler.com/books/umlDistilled.html
* OWASP Foundation. (n.d.). *Top 10 web application security risks*. Retrieved from https://owasp.org/www-project-top-ten/
* Scrum.org. (n.d.). *The basics of Scrum, Agile methodology*. Retrieved from https://www.scrum.org/resources/what-is-scrum
* DigitalOcean. (n.d.). *Database optimization techniques for web applications*. Retrieved from https://www.digitalocean.com/community/tutorials/database-optimization-techniques