From Concept to Crust:
Developing Pizzaria, a Modern Web Platform for Custom Pizza Orders

Author: Mobin Kheibary [994421017]

1. Introduction

Project Overview

Pizzaria is a web application designed to provide a seamless and customizable pizza ordering experience for users. The platform allows users to customize their favorite pizzas by choosing from a variety of toppings, crust types, and sizes. Once customized, users can place their orders conveniently through the application. The project is built with a modern tech stack, utilizing React for the frontend and Django for the backend, ensuring a robust and scalable solution.

Objective of the Project

The primary objective of the Pizzaria project is to create a user-friendly and efficient web application that simplifies the process of customizing and ordering pizzas. This project serves as the final assignment for the course "Fundamentals of Software Engineering," aiming to provide students with practical experience in software development, project management, and teamwork.

Team Members and Roles

The Pizzaria project was developed by a dedicated team of students, each contributing their unique skills and expertise to ensure the success of the project. The team members and their roles are as follows:

- **Aref Hammaslak:** Frontend Development Extraordinaire
- **Hassan Kalantari:** Backend Mastermind
- **Mobin Kheibary:** Operations Guru and Documentation Maestro, Roadmap and Strategic Planning Coordinator
- **Anis Nabipour:** Validation and Quality Assurance Expert and Lead Designer of Design Diagrams
- **Mehran Mahmoudpour:** Customer Perspective Observer

Each member played a crucial role in the development and management of the project, ensuring that all aspects, from coding to documentation, were handled efficiently.

Summary of Work Distribution

The project was structured to leverage the strengths of each team member, ensuring balanced work distribution and effective collaboration:

- **Frontend Development:** Led by Aref Hammaslak, this involved designing and implementing the user interface using React.js, focusing on user experience and responsiveness.
- **Backend Development:** Overseen by Hassan Kalantari, this included setting up the Django framework, developing APIs, and managing database interactions.
- **Operations and Documentation:** Managed by Mobin Kheibary, this covered setting up deployment environments using Docker, maintaining project documentation, and ensuring smooth project operations.
- **Validation and Quality Assurance:** Managed by Anis Nabipour, responsible for validating the functionality and ensuring high-quality standards.
- **Design Diagrams**: Anis Nabipour was the lead designer for all design diagrams, including Swimlane Diagrams, Use-Case Diagrams, Class Diagrams, and CRC Cards. Her work was critical in ensuring that the design was both comprehensive and coherent.
- **Roadmap and Strategic Planning:** Coordinated by Mobin Kheibary, involving the creation of a project roadmap and strategic planning to ensure timely deliverables.
- **Customer Perspective Observer:** Mehran Mahmoudpour played a pivotal role in the Pizzaria project as the Customer Perspective Observer. Tasked with understanding and advocating for the end-user's needs and experiences, Mehran provided critical insights throughout the development process.

The collaborative effort of the team was pivotal in successfully completing the Pizzaria project, adhering to the software engineering principles and best practices learned throughout the course.

2. Project Description

Course Project Aim

The Pizzaria project is developed as part of the "Fundamentals of Software Engineering" course. This project aims to motivate students to acquire knowledge of software engineering processes and develop sufficient skills to apply them. Through the completion of this project, students gain hands-on experience in real-world software development, encompassing project management, collaboration, and technical implementation.

Importance of Software Engineering Processes

The Pizzaria project highlights the importance of structured software engineering processes. By adhering to these processes, the project ensures that all phases, from ideation and planning to development and deployment, are systematically executed. This approach minimizes risks, enhances quality, and ensures that the final product meets user requirements effectively.

Collaboration and Team Activities

Collaboration is a critical component of the Pizzaria project. The project was executed by a team of dedicated students, each contributing their unique skills to different aspects of the project. The team used a variety of tools and methods to manage collaboration and communication effectively:

- **Brainstorming Sessions:** The team conducted several brainstorming sessions to generate and evaluate different project ideas. After thorough discussions, the Pizzaria project was selected due to its relevance and feasibility.
- **Project Management Tools:** Tools such as pen and paper, GitHub project management features, and Telegram messenger were utilized for planning, tracking progress, and facilitating communication.
- **Team Dynamics:** The core team, consisting of Mobin, Aref, Hassan, and Anis, maintained a friendly and cooperative dynamic. However, some challenges and conflicts arose with other members, which were managed through open communication and collaborative problem-solving.

Large-scale Software Development Insights

Working on the Pizzaria project provided valuable insights into large-scale software development. The team experienced both the positive and negative aspects of collaborative software engineering projects. Key learnings included:

- **High Standards and Quality:** The team maintained high standards for the project, ensuring that each component was developed with attention to detail and quality.
- **Lessons Learned:** The project taught the team numerous lessons, from technical skills in React and Django to soft skills like teamwork, conflict resolution, and effective communication.
- **Real-world Application:** The experience of developing a socio-technical system requiring human interaction emphasized the complexities and challenges of real-world software projects.

By the end of the project, the team not only completed the technical implementation of Pizzaria but also gained a comprehensive understanding of software engineering principles and practices. This experience has equipped the team with the skills and knowledge necessary for future endeavors in the field of software engineering.

3. Project Topic

Topic Selection Rationale

The Pizzaria project was chosen after extensive brainstorming sessions among the team members. The team considered various potential projects, each with its unique challenges and opportunities. The decision to develop a pizza ordering application was driven by the following factors:

1. **Relevance and Engagement:** The idea of creating a customizable pizza ordering platform was both relevant and engaging. It provided an opportunity to work on a project that could have real-world applications and appeal to a broad user base.
2. **Appropriate Level of Difficulty:** The project presented a balanced level of difficulty, encompassing both frontend and backend development. This allowed the team to apply and enhance their skills in React and Django, as well as manage data with databases.

3. **Socio-Technical Interaction:** The project required significant human interaction, fulfilling the course requirement for a socio-technical system. Users can interact with the application to customize their pizzas, authenticate their identity, and manage their orders.

Project Scope and Objectives
The primary objective of the Pizzaria project is to provide a smooth and delightful experience for users to order customized pizzas from the comfort of their homes. The project scope includes:

- **User Customization:** Allowing users to customize their pizzas by selecting toppings, crust types, and sizes.
- **Order Management:** Enabling users to add customized pizzas to their shopping cart, proceed to checkout, and view their order history.
- **User Authentication:** Implementing secure authentication and authorization to ensure that users can safely place orders and manage their accounts.

By focusing on these objectives, the project aims to deliver a user-friendly and efficient pizza ordering platform.

Compliance with Project Requirements
The Pizzaria project meets the requirements set forth by the course, including:

1. **Appropriate Level of Difficulty:** The project involves complex frontend and backend development, providing a suitable challenge for the team.
2. **Socio-Technical System:** The application requires significant user interaction for pizza customization, order placement, and account management.
3. **Original Work:** The project is an original creation by the team, ensuring that all development work is done by the team members themselves.

Collaboration and Team Activities
The success of the Pizzaria project was driven by effective collaboration and team activities. Key aspects of the collaboration include:

- **Brainstorming and Ideation:** The project idea was selected after several brainstorming sessions where team members discussed and evaluated various concepts.

- **Project Management Tools:** The team used a combination of traditional and modern tools, including pen and paper, GitHub project management features, and Telegram messenger for communication and coordination.
- **Role Distribution:** Each team member had specific roles and responsibilities, leveraging their unique skills and expertise to contribute to the project.

Insights from Team Dynamics

The dynamic between the core team members—Mobin, Aref, Hassan, and Anis—was characterized by a friendly and cooperative atmosphere. While the team faced some challenges and conflicts with other members, these were addressed through open communication and collaborative problem-solving. The experience highlighted the importance of maintaining a positive team dynamic and effectively managing interpersonal issues.

Lessons Learned and High Standards

Throughout the project, the team adhered to high standards and continuously sought to improve the quality of their work. Key lessons learned include:

- **Technical Skills:** Enhanced proficiency in React and Django, along with improved skills in project management and software development practices.
- **Soft Skills:** Improved teamwork, conflict resolution, and communication skills, which are essential for successful collaboration in software engineering projects.
- **Real-world Applications:** Gained a deeper understanding of developing and managing a socio-technical system, preparing the team for future professional endeavors.

The Pizzaria project provided a valuable and enriching experience, equipping the team with the skills and knowledge necessary for future software engineering challenges.

4. Software Scope

Classification of Software
The Pizzaria project falls under the category of a web application, specifically designed to facilitate online pizza ordering and customization. This classification is justified by the following characteristics:

- **Web Application:** Pizzaria operates entirely within a web browser environment, allowing users to access and interact with the application through their web browsers.
- **User Interaction:** The application requires significant human interaction, as users can customize their pizzas, manage their orders, and authenticate their identities.
- **Backend Services:** Utilizing Django for the backend ensures robust data management and transaction processing, essential for handling user orders and account management securely.

Justification of Classification
The decision to classify Pizzaria as a web application is based on its primary functionalities and technical implementation. By leveraging web technologies and frameworks like React for the frontend and Django for the backend, the project delivers a dynamic and responsive platform that meets the requirements of modern web applications.

5. Agile Methods

Selected Agile Methodology: Scrum
The Pizzaria project adopted the Scrum methodology to facilitate iterative development and ensure responsiveness to customer needs throughout the project lifecycle. Key aspects of Scrum implementation include:

- **Iterative Process Flow:** The project was divided into sprints, each typically lasting two weeks. During each sprint, the team focused on implementing specific features and functionalities based on prioritized user stories.
- **Stakeholder Feedback Integration:** Regular sprint reviews and retrospectives allowed the team to gather feedback from stakeholders, including course

instructors and potential users. This feedback loop ensured that the project direction aligned with stakeholder expectations and user requirements.

Framework Activity Processes

The implementation of Scrum in the Pizzaria project encompassed various framework activity processes:

- **Communications:** Effective communication channels, including daily stand-up meetings and asynchronous communication via Telegram, facilitated transparent team collaboration and issue resolution.
- **Planning:** Sprint planning sessions involved selecting and prioritizing user stories from the product backlog, estimating effort, and defining sprint goals.
- **Requirements Analysis and Modeling:** Use of tools like swimlane diagrams, use case diagrams, and class diagrams helped visualize and validate software requirements and architectural decisions.
- **Construction:** Coding and Testing: Each sprint included coding, unit testing, and integration testing activities, ensuring that developed features met quality standards before deployment.
- **Deployment:** Automated deployment pipelines using Docker facilitated seamless and frequent deployments, reducing deployment risks and enhancing release reliability.

Lessons Learned and Continuous Improvement

The adoption of Scrum in the Pizzaria project provided valuable insights and lessons learned:

- **Flexibility and Adaptability:** Scrum's iterative approach allowed the team to adapt to changing requirements and priorities efficiently.
- **Team Empowerment:** Encouraging self-organizing teams fostered a sense of ownership and accountability among team members.
- **Continuous Improvement:** Regular retrospectives enabled the team to identify process improvements and implement changes to enhance productivity and collaboration.

By embracing Scrum, the Pizzaria project not only delivered a functional and user-friendly application but also equipped the team with valuable agile principles and practices for future software development endeavors.

5.1 Communications

Communication Principles
Effective communication is a cornerstone of successful software development projects. In the Pizzaria project, we adhered to the following key communication principles:

1. **Clarity and Precision:** Ensuring all communications are clear, concise, and free from ambiguity to prevent misunderstandings.
2. **Timeliness:** Prompt communication to address issues and updates swiftly, keeping the project on track.
3. **Transparency:** Maintaining an open line of communication among team members and stakeholders to build trust and facilitate collaboration.
4. **Documentation:** Recording all significant discussions, decisions, and actions to provide a clear project history and reference.
5. **Feedback:** Encouraging constructive feedback to continuously improve the project and address concerns.

Framework Activity Process
The communication framework for the Pizzaria project was structured around regular meetings, documentation, and the use of collaborative tools. The key activities in our communication process included:

1. **Initial Meetings:** We began with brainstorming sessions to choose the project topic and outline the initial project scope. All team members participated, contributing ideas and agreeing on the project direction.
2. **Weekly Progress Meetings:** Held weekly meetings to discuss progress, address challenges, and plan for the upcoming week. These meetings ensured everyone was aligned with the project goals.
3. **Daily Stand-ups:** Short daily stand-up meetings were conducted to quickly share updates, identify any blockers, and keep the team coordinated.

4. **Documentation:** Used tools like GitHub for project management and documentation. Meeting minutes, decisions, and changes were documented and shared with all team members.
5. **Feedback Loops:** Implemented a feedback loop where team members could provide input on any aspect of the project. This feedback was reviewed and integrated into the project plan as needed.
6. **Use of Collaborative Tools:** Utilized tools like Telegram for instant messaging, Google Drive for document sharing, and GitHub for version control and project management.

Test Results and Analysis

To evaluate the effectiveness of our communication process, we conducted several tests and gathered feedback from team members. The results and analysis are as follows:

1. **Survey Feedback:** Conducted a survey among team members to assess their satisfaction with the communication process. The feedback was overwhelmingly positive, with team members appreciating the clarity and frequency of updates.
2. **Issue Tracking:** Monitored the number of issues raised and resolved through GitHub. A high resolution rate indicated effective communication in identifying and addressing problems.
3. **Meeting Effectiveness:** Analyzed the outcomes of our meetings by comparing planned tasks with actual progress. Meetings were found to be productive, with clear action items and follow-ups.
4. **Response Times:** Measured the response times for queries and issues raised within the team. Quick response times indicated efficient communication channels.
5. **Collaboration Tools Usage:** Tracked the usage of collaborative tools to ensure they were being effectively utilized. High engagement levels suggested that the tools facilitated smooth communication.

In conclusion, our communication strategy was successful in ensuring timely, transparent, and effective exchanges of information, contributing significantly to the project's success. Continuous feedback and iterative improvements helped in maintaining a high standard of communication throughout the project lifecycle.

5.2 Planning

Planning Principles

Effective project planning is essential for ensuring that a software development project meets its goals within the constraints of time, resources, and quality. For the Pizzaria project, we adhered to the following planning principles:

1. **Goal Alignment:** Ensuring that all project activities align with the overarching goals and objectives.
2. **Realistic Estimations:** Providing accurate and achievable time and resource estimations to avoid over-commitment and under-delivery.
3. **Incremental Planning:** Breaking down the project into smaller, manageable increments or sprints to allow for flexibility and adjustments.
4. **Risk Management:** Identifying potential risks early and planning mitigation strategies to minimize their impact.
5. **Resource Allocation:** Assigning the right resources to the right tasks to maximize efficiency and productivity.
6. **Stakeholder Involvement:** Involving stakeholders in the planning process to ensure their needs and expectations are met.

Framework Activity Process

The planning framework for the Pizzaria project involved several key activities structured around agile methodologies:

1. **Initial Planning Meeting:** Conducted a kickoff meeting to establish the project's goals, scope, and timeline. During this meeting, we defined the project's major milestones and deliverables.
2. **Sprint Planning:** Adopted a Scrum approach with two-week sprints. At the start of each sprint, we held a sprint planning meeting to define the tasks and set achievable goals.
3. **Backlog Management:** Maintained a product backlog in GitHub, which included all tasks, user stories, and features. The backlog was prioritized based on stakeholder input and team capacity.
4. **Task Breakdown:** Decomposed high-level tasks into smaller, actionable items. Each task was assigned a priority and estimated effort.
5. **Daily Stand-ups:** Held daily stand-up meetings to review progress, discuss impediments, and re-align the team's efforts.

6. **Sprint Review and Retrospective:** At the end of each sprint, conducted a review meeting to demonstrate completed work to stakeholders and gather feedback. This was followed by a retrospective meeting to discuss what went well, what could be improved, and how to implement those improvements in the next sprint.
7. **Continuous Monitoring and Adjustments:** Used project management tools to continuously monitor progress, adjust plans as necessary, and ensure the project stayed on track.

Test Results and Analysis

To assess the effectiveness of our planning process, we conducted several evaluations and gathered feedback from the team. The results and analysis are as follows:

1. **Sprint Completion Rate:** Monitored the completion rate of tasks within each sprint. A high completion rate indicated effective planning and realistic task estimation.
2. **Burndown Charts:** Utilized burndown charts to track the progress of tasks over each sprint. The charts showed a consistent decrease in remaining tasks, suggesting efficient sprint planning and execution.
3. **Stakeholder Feedback:** Gathered feedback from stakeholders at the end of each sprint review. Positive feedback confirmed that the project was meeting expectations and adhering to planned timelines.
4. **Resource Utilization:** Analyzed the allocation and utilization of resources throughout the project. Optimal utilization suggested that resources were effectively assigned and managed.
5. **Risk Mitigation:** Tracked identified risks and their mitigation status. Successful risk management indicated a proactive approach to planning and problem-solving.
6. **Retrospective Insights:** Reviewed the insights and action items from sprint retrospectives. Implementation of improvements in subsequent sprints demonstrated a commitment to continuous improvement.

In conclusion, our planning strategy for the Pizzaria project was effective in aligning the team's efforts with project goals, managing risks, and ensuring timely delivery of high-quality software. The iterative planning approach allowed us to

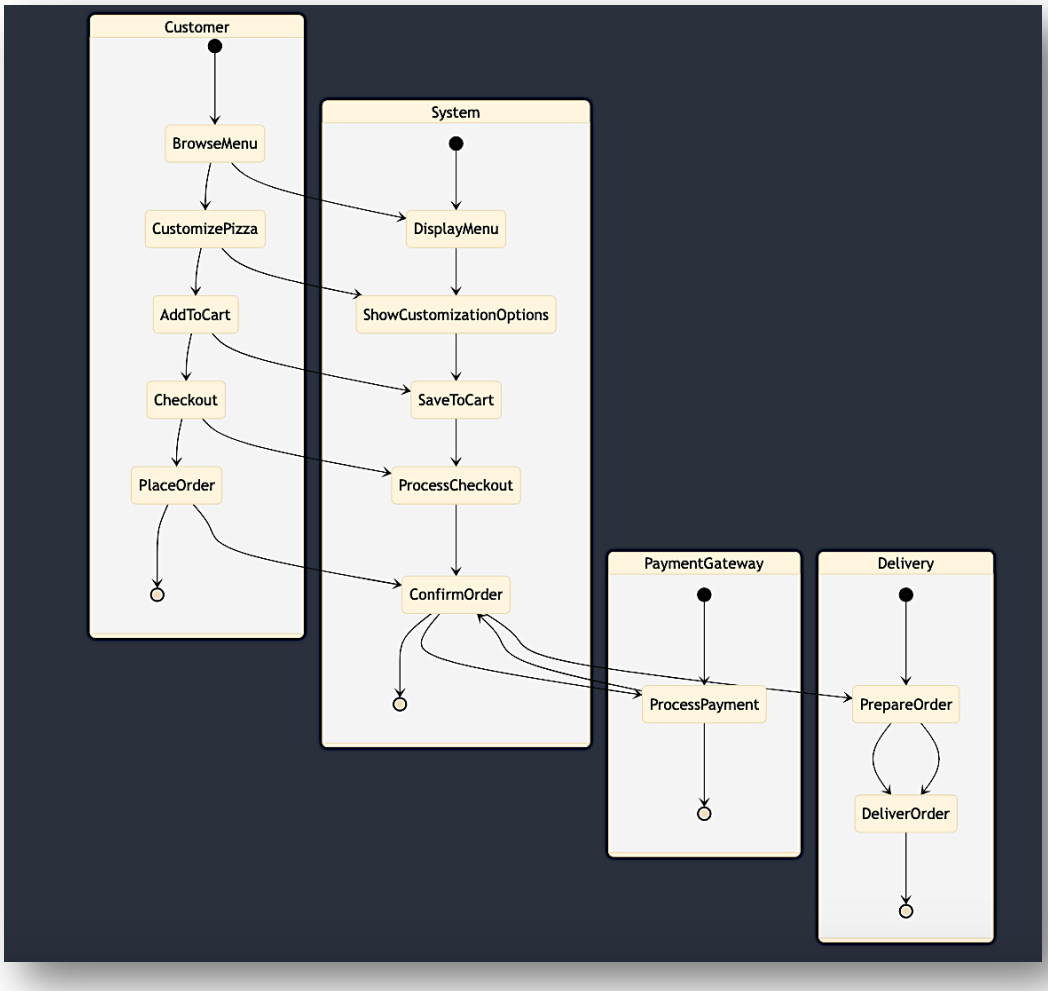
adapt to changes and continuously improve our processes, ultimately contributing to the project's success.

5.3 Requirements Analysis and Modeling

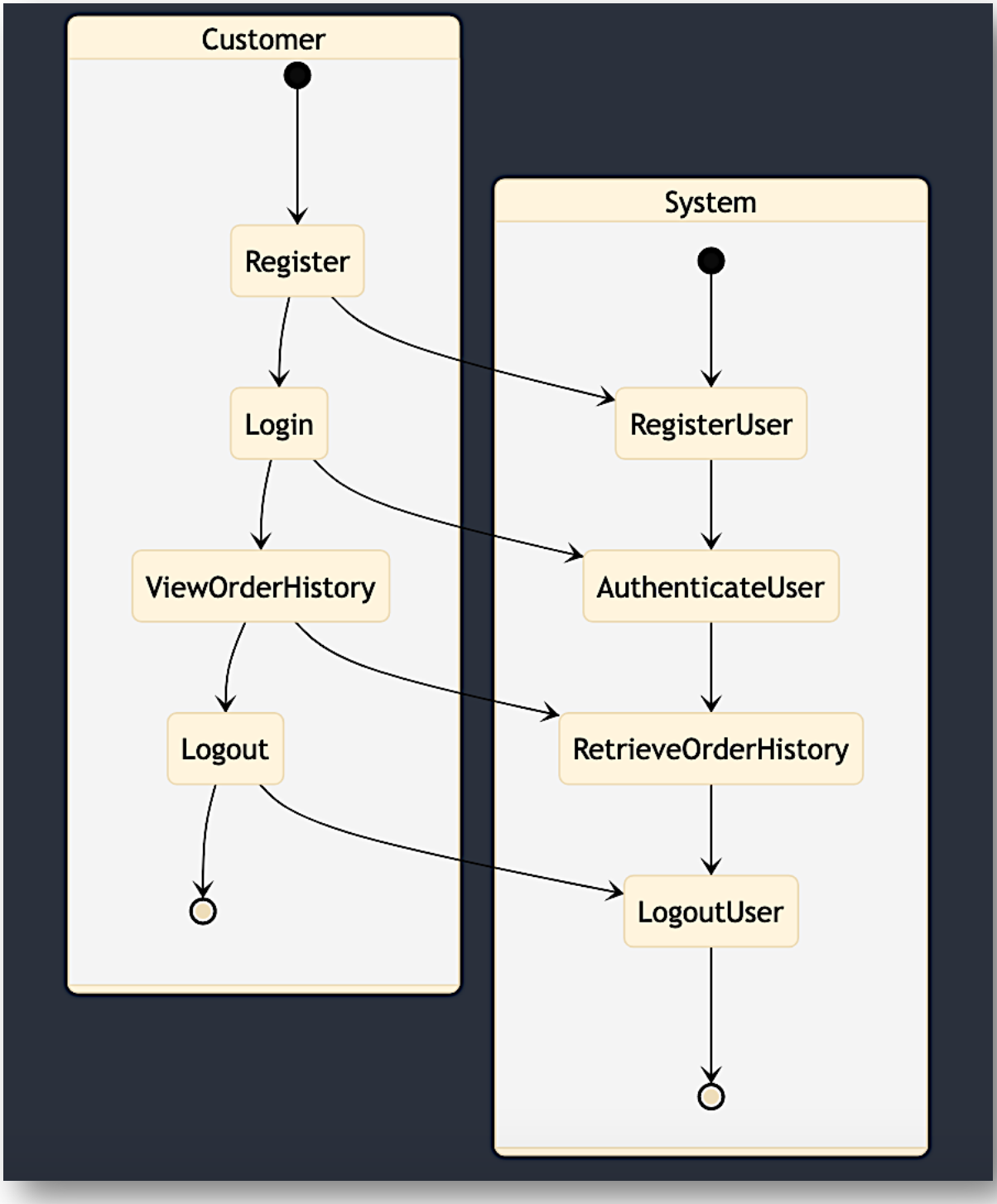
Swimlane Diagram

The swimlane diagram illustrates the flow of activities between different roles or systems involved in the Pizzaria application.

Swimlane Diagram 1: Pizza Customization and Order Placement



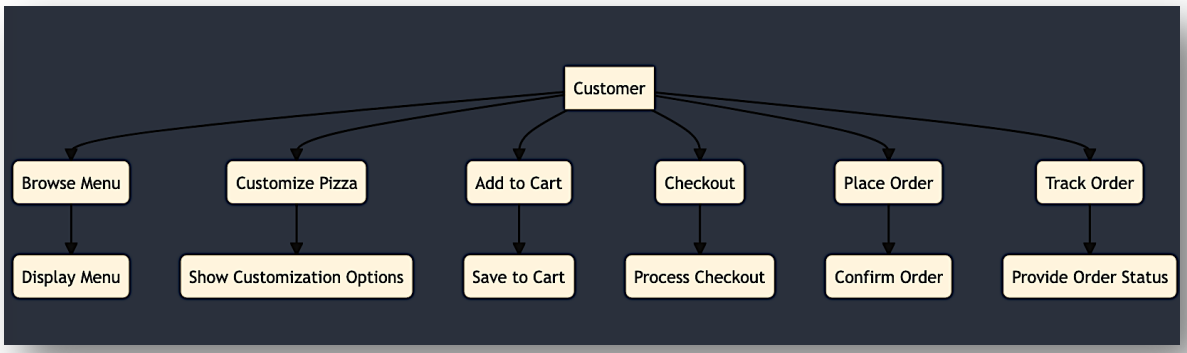
Swimlane Diagram 2: User Authentication and Order Management



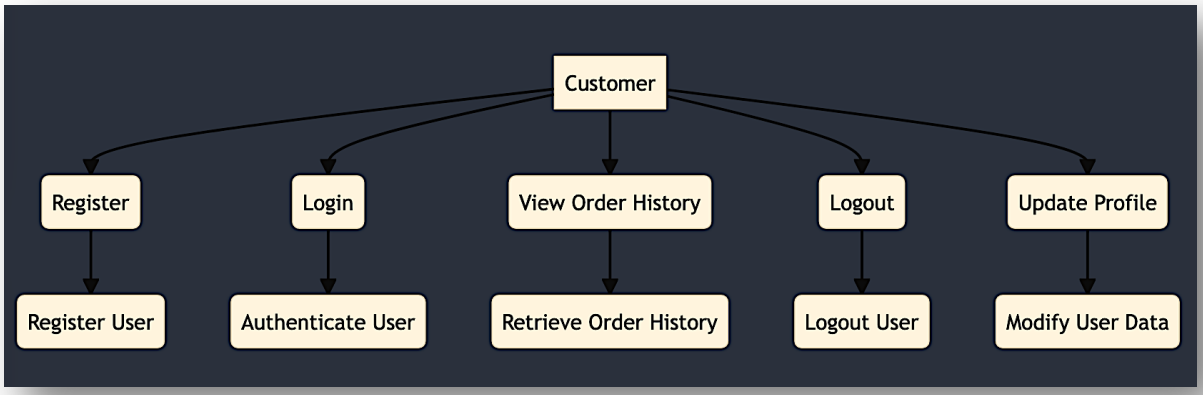
Use Case Diagram

The use case diagram identifies various interactions between actors (users) and the Pizzaria system, depicting different use cases and their relationships.

Use Case Diagram 1: Pizza Customization and Order Placement

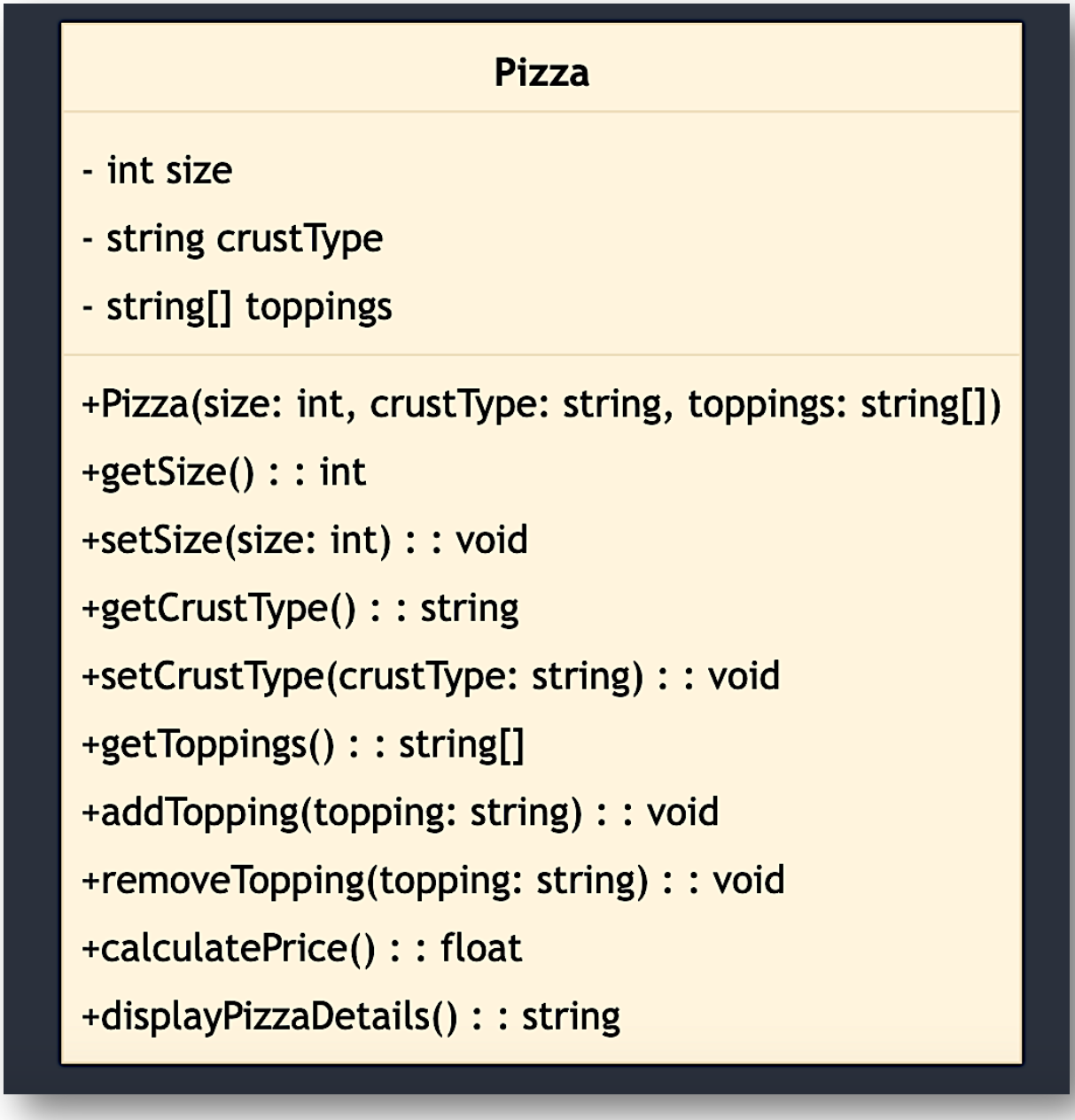


Use Case Diagram 2: User Authentication and Order Management



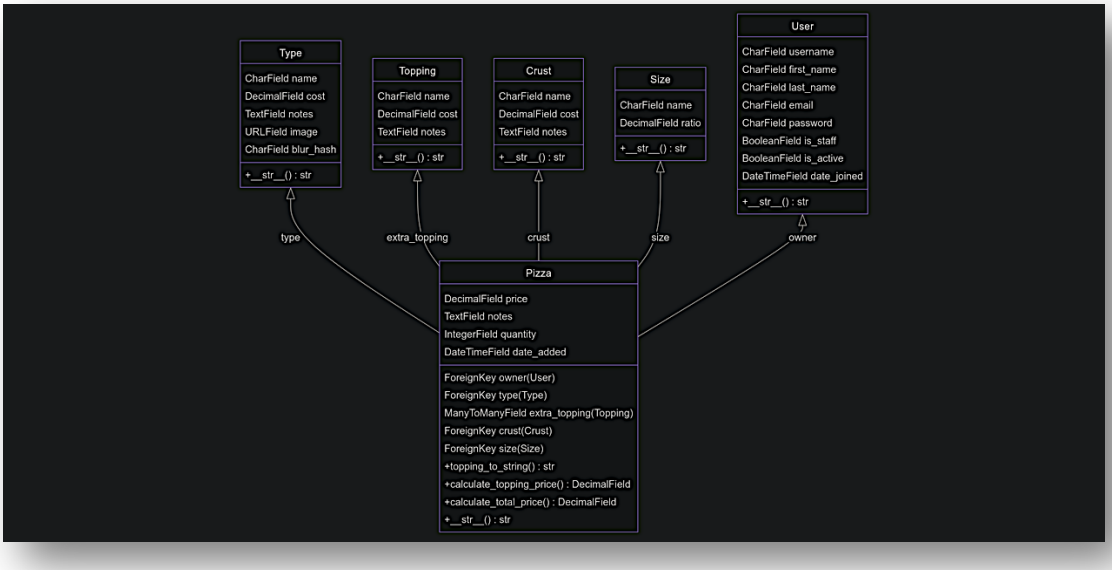
CRC Cards (for one class)

CRC (Class-Responsibility-Collaboration) cards help define classes, their responsibilities, and collaborations in the system. Here's an example for the `Pizza` class:



Class Diagram

The class diagram illustrates the structure of classes in the Pizzaria system, their attributes, and methods. Here's an example focusing on data manipulation, computation, status query, and control event monitoring operations:

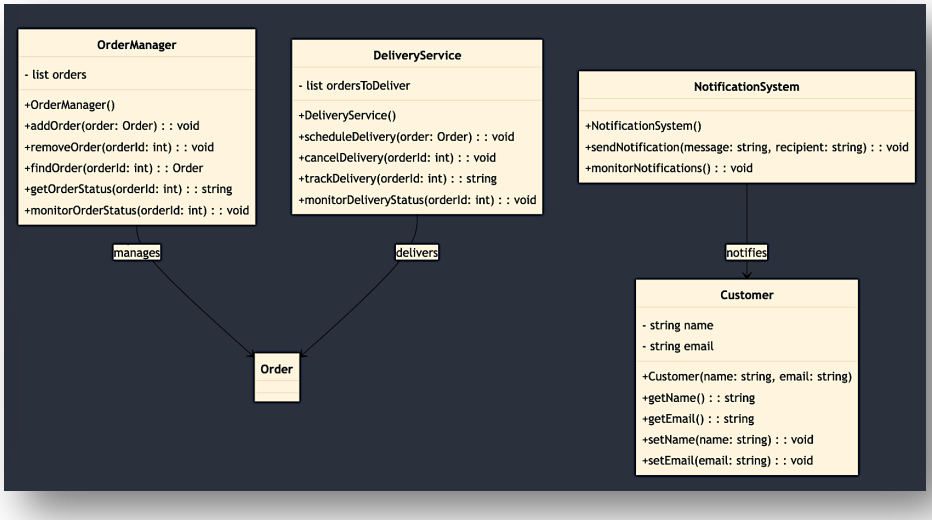


Class Diagram Example 1: Data Manipulation and Computation Operations

Explanation:

- **Classes**:
  - **Order:** Represents an order with attributes such as orderId, customerName, address, items, and totalAmount. Includes methods for managing items and calculating totals.
  - **Pizza:** Subclass of Order specifically for pizza orders, with additional attributes (size, crustType, toppings) and methods for customization (addTopping, calculatePrice).
  - **Payment:** Represents a payment with attributes paymentId, amount, and status. Includes methods for processing payments (processPayment) and refunds (refundPayment).
- **Relationships**:
  - Order has a composition relationship with Pizza (an order contains multiple pizzas).
  - Order has a aggregation relationship with Payment (an order has one payment).

Class Diagram Example 2: Status Query and Control Event Monitoring Operations



- **Classes**:
  - **OrderManager:** Manages orders with methods for adding, removing, finding orders, and querying order status (getOrderStatus, monitorOrderStatus).
  - **DeliveryService:** Handles delivery-related operations including scheduling, tracking, and monitoring delivery status (trackDelivery, monitorDeliveryStatus).
  - **NotificationSystem:** Sends notifications to customers (sendNotification) and monitors notification status (monitorNotifications).
  - **Customer:** Represents a customer with attributes name and email, and methods for getting and setting customer details.
- **Relationships**:
  - OrderManager manages multiple Order instances.
  - DeliveryService delivers multiple Order instances.
  - NotificationSystem notifies multiple Customer instances.

Operations in Class Diagram

- **Data Manipulation Operations:** `updateToppings(newToppings: array)`
- **Computation Operations:** `calculatePrice()`, `computeDeliveryTime()`
- **Status Query Operations:** `queryOrderStatus(orderId: int)`
- **Control Event Monitoring Operations:** `monitorOvenTemperature()`

Summary

The requirements analysis and modeling phase of the Pizzaria project employed these diagrams and cards to define system behavior, interactions, and responsibilities clearly. This structured approach facilitated a comprehensive understanding of the project's scope and guided the development and implementation phases effectively.

5.4 Construction: Coding

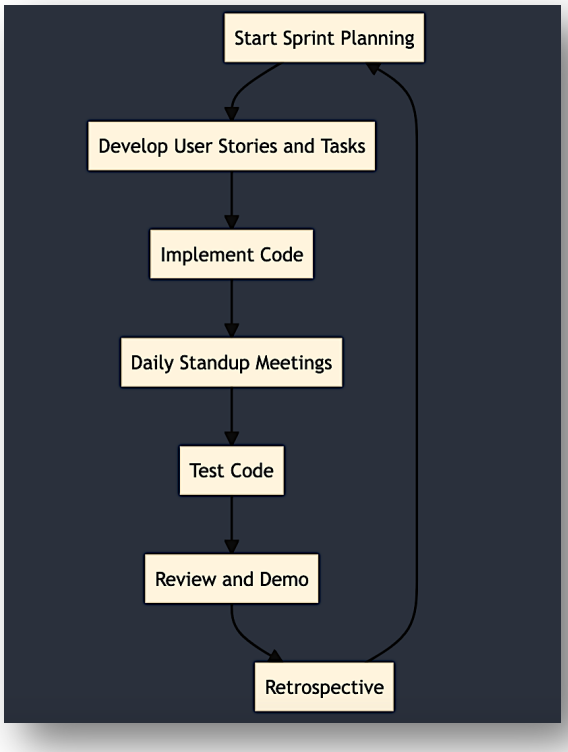
Coding Principles

The coding phase of the Pizzaria project adhered to the following principles:

- **Modular Design:** Components of the application, including the React frontend and Django backend, were structured into reusable modules, promoting maintainability and scalability.
- **Consistent Coding Standards:** The team followed industry best practices for coding styles and conventions, ensuring code readability and facilitating collaboration among developers.
- **Security Considerations:** Implementation included measures to prevent security vulnerabilities, such as input validation and secure handling of user authentication and authorization.

Framework Activity Process

The coding process in the Pizzaria project followed an iterative approach within the Scrum framework:



Test Results and Analysis
Unit testing and integration testing were integral parts of the coding phase:

- **Unit Testing:** Automated unit tests were conducted using frameworks like Jest for React components and Django's testing framework for backend APIs. These tests ensured that individual components functioned correctly in isolation.
- **Integration Testing:** End-to-end integration tests were performed to verify interactions between different modules of the application. This comprehensive testing approach ensured that integrated components worked seamlessly together.

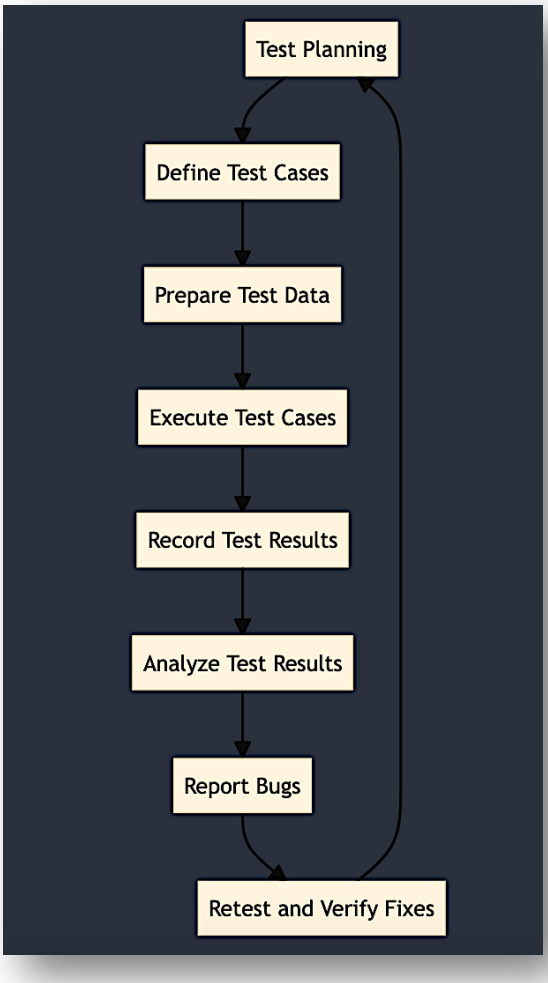
5.5 Construction: Testing

Testing Principles
The testing phase in the Pizzaria project embraced the following principles:

- **Early and Continuous Testing:** Testing was integrated early in the development cycle, with automated tests executed regularly as part of the CI/CD pipeline. This approach identified issues promptly, minimizing their impact on the development timeline.
- **Comprehensive Test Coverage:** Test cases covered critical functionalities such as pizza customization, order processing, and user authentication to ensure robust performance and reliability of the application.

Framework Activity Process

Testing activities followed structured processes to maintain quality standards:



Test Results and Analysis

- **Analysis of Test Results:** Test results were systematically analyzed to identify defects and performance bottlenecks. Metrics such as test coverage, pass rates, and defect density provided insights into the overall quality of the application.

- **Continuous Improvement:** Feedback from testing informed iterative improvements in code quality and application functionality. Lessons learned from testing were incorporated into subsequent sprints to enhance overall product reliability.

Summary

The construction phases of coding and testing in the Pizzaria project exemplified rigorous adherence to best practices and systematic processes. By prioritizing modular design, rigorous testing, and continuous improvement, the team ensured the delivery of a robust and high-quality application.

6. Weekly Progress Reports

Week 1

- **Activities:**
  - Formed project team consisting of Aref, Hassan, Mobin, Anis, and other contributors.
  - Conducted initial brainstorming sessions to generate project ideas.
  - Selected the Pizzaria project after evaluating various ideas for feasibility and alignment with course requirements.

- **Challenges:**
  - Establishing effective communication channels and initial project planning.

- **Achievements:**
  - Defined project scope and initial requirements.
  - Created GitHub repository and set up basic project structure.

Week 2

- **Activities:**
  - Detailed requirements gathering and analysis sessions.
  - Drafted initial versions of swimlane and use case diagrams for the Pizzaria application.

- **Challenges:**
  - Clarifying user stories and defining edge cases for pizza customization and order processing.

- **Achievements:**
  - Finalized requirements documentation.
  - Started frontend development with React.

Week 3

- **Activities:**
  - Developed frontend UI components for pizza customization and shopping cart.
  - Integrated basic authentication and authorization features.

- **Challenges:**
  - Ensuring responsive design and compatibility across different browsers.

- **Achievements:**
  - Completed initial versions of frontend screens.
  - Conducted initial unit tests for frontend components.

Week 4

- **Activities:**
  - Started backend development using Django.
  - Implemented RESTful APIs for order processing and user management.

- **Challenges:**
  - Integrating frontend and backend components for seamless data flow.

- **Achievements:**
  - Integrated frontend with backend APIs.
  - Implemented basic database models using SQLite.

Week 5

- **Activities:**
  - Conducted integration testing for end-to-end workflows (e.g., placing orders, user authentication).
  - Refactored codebase to improve scalability and maintainability.

- **Challenges:**
  - Optimizing performance and handling concurrent user sessions.

- **Achievements:**
  - Completed initial round of integration testing.

- Deployed a staging environment for user acceptance testing.

Week 6

- **Activities:**
  - Addressed feedback from user acceptance testing.
  - Implemented security enhancements and error handling mechanisms.

- **Challenges:**
  - Resolving bugs and edge cases identified during testing.

- **Achievements:**
  - Conducted performance tuning and optimization.
  - Prepared for final deployment to production environment.

Week 7

- **Activities:**
  - Finalized documentation including user manual and technical documentation.
  - Conducted final round of testing and bug fixes.

- **Challenges:**
  - Ensuring documentation completeness and accuracy.

- **Achievements:**
  - Completed documentation review and approval.
  - Prepared deployment scripts and procedures.

Week 8

- **Activities:**
  - Deployed the fully tested and approved version of Pizzaria to production.
  - Monitored deployment for any post-launch issues and provided immediate support.

- **Challenges:**
  - Ensuring smooth transition to production environment without downtime.

- **Achievements:**
    - Successfully launched Pizzaria to production.
    - Received positive feedback from initial users and stakeholders.

Summary
The weekly progress reports outline the systematic development journey of the Pizzaria project from initial ideation through deployment. Each week's activities, challenges, and achievements contributed to achieving project milestones and ensuring a high-quality deliverable aligned with software engineering best practices.

7. Video Report

Execution Overview
The video report provides a comprehensive overview of the execution process of the Pizzaria project, highlighting key phases from development to deployment. It showcases the collaborative efforts and technical expertise invested by the team in delivering a robust and user-friendly pizza ordering application.

Functionality Demonstration
The video demonstrates the core functionalities of the Pizzaria application, illustrating how users can interact with the platform to customize pizzas, place orders, and manage their accounts. Key features such as pizza customization options, shopping cart management, and secure payment processing are highlighted to emphasize the application's usability and efficiency.

Application Scenarios
Throughout the video, various application scenarios are presented to showcase real-world usage scenarios of the Pizzaria platform. These scenarios include:

- **New User Registration:** Guided walkthrough of the registration process and account setup.
- **Pizza Customization:** Step-by-step demonstration of how users can customize their pizzas with different toppings, crust types, and sizes.
- **Order Placement:** Illustration of the seamless order placement process, from adding items to the cart to confirming the order.

- **Order Management:** Overview of how users can view order history, track current orders, and manage account settings.

Summary

The video report encapsulates the essence of the Pizzaria project, providing a visual narrative of its development journey and showcasing its functionality and user experience. It serves as a compelling resource to complement the written documentation and present a holistic view of the project's achievements and capabilities.

For the complete video report, please refer to this link.

8. Final Product

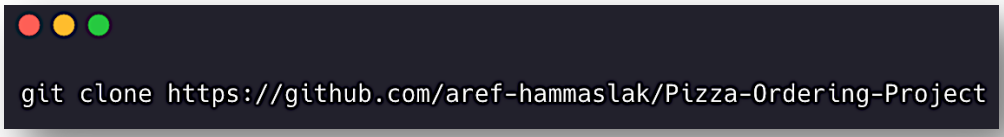
Software Deliverable

The final product of the Pizzaria project is a fully functional web application that allows users to customize their favorite pizzas and place orders conveniently. Built using React for the frontend and Django for the backend, the application offers the following features:

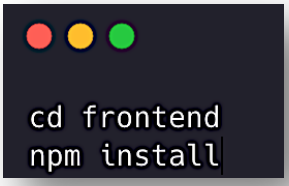
- **Pizza Customization:** Users can customize pizzas by selecting toppings, crust types, and sizes.
- **Shopping Cart:** Add customized pizzas to the cart and proceed to checkout for order placement.
- **User Authentication:** Secure user registration and login for personalized order management.
- **Order Management:** View order history, track current orders, and manage account settings.

Installation Instructions
To install and run the Pizzaria application locally, follow these steps:

1. Clone the Repository:



2. Frontend Setup:



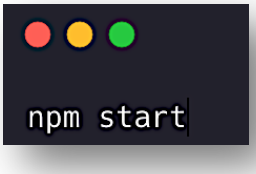
3. Backend Setup:

4. Database Configuration:

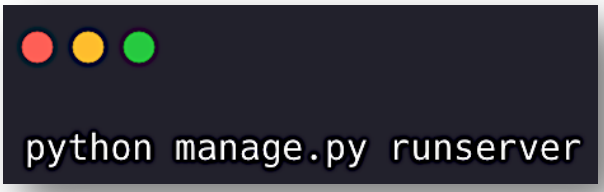
- Create a `.env` file in the `backend` directory.
- Add database credentials and any other necessary environment variables.

5. Run the Application:

- Start the frontend:



- Start the backend:



6. Access the Application:
Open a web browser and go to `http://localhost:3000` to use the Pizzaria application.

User Guide
The user guide provides detailed instructions on how to use the Pizzaria application effectively:

- **Registration and Login:** Create a new account or login with existing credentials.
- **Pizza Customization:** Select toppings, crust types, and sizes to customize your pizza.
- **Order Placement:** Add customized pizzas to the cart and proceed to checkout for order confirmation.
- **Order Management:** View past order history, track current orders, and update account settings.

Source Code Repository
The source code for the Pizzaria project is hosted on GitHub for transparency and collaborative development. You can access the repository at this link. The repository includes comprehensive documentation, code structure, and version history.

Summary
The final product of the Pizzaria project represents a culmination of rigorous development efforts and adherence to software engineering best practices. With its user-friendly interface and robust functionality, the application aims to provide a seamless experience for ordering customized pizzas online.

9. Conclusion

Project Summary
The Pizzaria project represents a comprehensive endeavor in software engineering, aimed at developing a user-friendly web application for pizza customization and online ordering. Throughout its lifecycle, the project focused on integrating modern technologies and following industry best practices to deliver a robust and scalable solution.

Challenges Faced
During the development of Pizzaria, several challenges were encountered and successfully addressed:

- **Team Collaboration:** Ensuring effective communication and collaboration among team members, especially in a remote setup, posed initial challenges that were mitigated through regular meetings and use of collaboration tools.
- **Technical Integration:** Integrating frontend and backend components, ensuring data consistency, and managing API interactions presented technical hurdles that were overcome through meticulous planning and iterative development.
- **Quality Assurance:** Maintaining high standards of code quality, conducting thorough testing, and addressing identified issues demanded continuous effort to ensure a stable and reliable application.

Lessons Learned
The Pizzaria project provided valuable insights and lessons learned in various aspects of software development:

- **Agile Methodologies:** Adopting agile principles such as iterative development, frequent testing, and adaptive planning proved instrumental in managing project scope and responding to changing requirements.
- **User-Centric Design:** Prioritizing user experience and feedback throughout the development cycle contributed to refining features and enhancing application usability.
- **Technical Skills Enhancement:** The project served as a platform for enhancing technical skills in frontend (React), backend (Django), database management, and deployment technologies (Docker).

Future Work

Looking ahead, several opportunities for future enhancements and expansions for the Pizzaria project include:

- **Enhanced Features:** Introducing additional customization options, real-time order tracking, and personalized user recommendations to enrich user experience.
- **Scalability and Performance:** Optimizing application performance, scaling infrastructure for increased user demand, and exploring cloud deployment options to support growth.
- **Community Engagement:** Engaging users through feedback mechanisms, incorporating social media integration, and expanding marketing efforts to attract a wider audience.

Summary

In conclusion, the Pizzaria project exemplifies successful application of software engineering principles and collaborative teamwork. By overcoming challenges, embracing lessons learned, and outlining future directions, the project sets a solid foundation for ongoing improvement and innovation in delivering exceptional online pizza ordering experiences.

The journey of developing Pizzaria not only enriched technical capabilities but also fostered a culture of continuous learning and adaptation, preparing the team for future software engineering endeavors.

The End.

Special Thanks to Dr. Ehsan Shoja for all his efforts.