Course: “Project: Software Engineering”

Design and development of a web application

Food Delivery Platform

Course of study: M.Sc. Computer Science

Tutor: Holger Klus

Author: Doniyor Rufatov

Matriculation: 9215551

October 2024

Explanation of Design and Implementation Procedure

Design and Implementation Procedure

The food delivery platform is designed to offer users seamless access to food by allowing them to order food from various restaurants and eateries. Users can enjoy different food options, real-time order tracking, multiple payment methods, and customer reviews to enhance overall food satisfaction.

The system also offers significant benefits to restaurants and eateries. By allowing establishments to rent space exclusively for kitchens, restaurants, and eateries can operate solely in delivery mode, significantly reducing rental costs. This innovative approach increases profitability by reducing overhead costs and allows restaurants to focus on producing high-quality food. Additionally, the platform offers expanded reach to a larger customer base and provides opportunities to increase revenue through an optimized and effective online presence.

Target Group

The primary target group for this platform includes both end-users (individuals and families) who seek a convenient way to order food online and restaurant owners who wish to expand their business through food delivery services. Further segmentation of these target groups can include:

- Busy professionals who prefer ordering takeout instead of cooking.
- Families looking for diverse dining options from the comfort of their homes.
- Restaurant owners and managers aim to lower overhead costs and increase delivery orders.

Chosen Software Development Methodology

We chose the Agile methodology to develop the online food delivery platform. Agile is characterized by iterative development that enables continuous feedback and improvements throughout the project lifecycle. This methodology promotes better collaboration between stakeholders, flexible integration of changing requirements, and rapid delivery of a functional product. Agile's emphasis on short development cycles known as sprints and regular re-evaluation and adaptation aligns well with our goal of delivering a user-centric application that can quickly respond to market needs.

Project Planning:

- **Sprint 1 (Weeks 1-2):** Requirements gathering, project setup, and initial system design.
- **Sprint 2 (Weeks 3-4):** Development of user registration and authentication module.
- **Sprint 3 (Weeks 5-6):** Implementation of restaurant search and filtering features.
- **Sprint 4 (Weeks 7-8):** Development of the food menu display and order placement functionality.

- **Sprint 5 (Weeks 9-10):** Real-time order tracking and restaurant management dashboard.
- **Sprint 6 (Weeks 11-12):** Integration of payment gateway and customer support features.
- **Sprint 7 (Weeks 13-14):** Testing, feedback incorporation, and final refinements.

Project Risks:

- **Technology Risk:** Risk of chosen technology stack not meeting performance expectations.
    - Mitigation: Conduct thorough technology assessments at the beginning of the project
- **Requirement Changes:** Frequent change requests from stakeholders
    - Mitigation: Implement Agile practices to handle evolving requirements.
- **Resource Availability:** Risk of key team members becoming unavailable.
    - Mitigation: Crosstrain team members to ensure redundancy.

Functional and Non-Functional Requirements

Functional

- User registration and authentication
    - User must be able to register using email or social media accounts
- Restaurant search and filter
    - User must be able to search and filter restaurants
- Food menu display
    - Restaurants can list their food items with detailed descriptions, images, and prices
    - Users should be able to customize food items (e.g., extra toppings, side dishes)
- Order placing and status tracking
    - Users can place orders and receive estimated delivery times
    - The system must provide real-time updates on order status (preparation, out for delivery, delivered)
- Restaurant management
    - Provide restaurants with a dashboard to manage orders, update menus, and access sales analytics
    - Allow integration with restaurant’s existing POS systems
- Payment Gateway Integration
    - Secure integration with major payment gateways (e.g., Stripe, PayPal)
    - Support for various payment methods (e.g., credit/debit cards, digital wallets)
- Customer support and Feedback
    - Users should have access to live chat or support tickets for assistance
    - Users can leave feedback and reviews on their orders

Non-Functional

- Security (data protection)
  - Implement SSL/TLS encryption for all data transmissions
  - Ensure compliance with data protection regulations (e.g., GDPR)
- Performance (fast loading time)
  - Ensure average page load time is under 2 seconds
  - Real-time updates must refresh within 5 seconds
- Scalability (ability to handle high user volume)
  - The platform should be able to handle up to 10,000 concurrent users
  - Implement horizontal scaling to add more servers as needed
- Usability (intuitive UI, accessible design)
  - Conduct user testing to ensure an intuitive interface
  - Ensure the platform is WCAG 2.0 compliant for accessibility
- Reliability (99.9% uptime, fault tolerance)
  - Aim for 99.9% uptime
  - Implement redundancy and failover solutions to handle server failures

System Design and Technologies

Scope and context: The system will be a web-based application accessible through multiple devices, including desktops and mobile devices. The main components of the system include users, restaurants, and administrators, along with external systems like payment gateways.

Technologies: Frontend – Bootstrap, Backend – Django, Database – PostgreSQL

The choice to these technologies is driven by their robustness, community support and alignment with agile practices. Bootstrap ensures responsive design making the application accessible from every device, Django facilitates rapid development, and PostgreSQL offers reliability for complex, high-volume transactions.

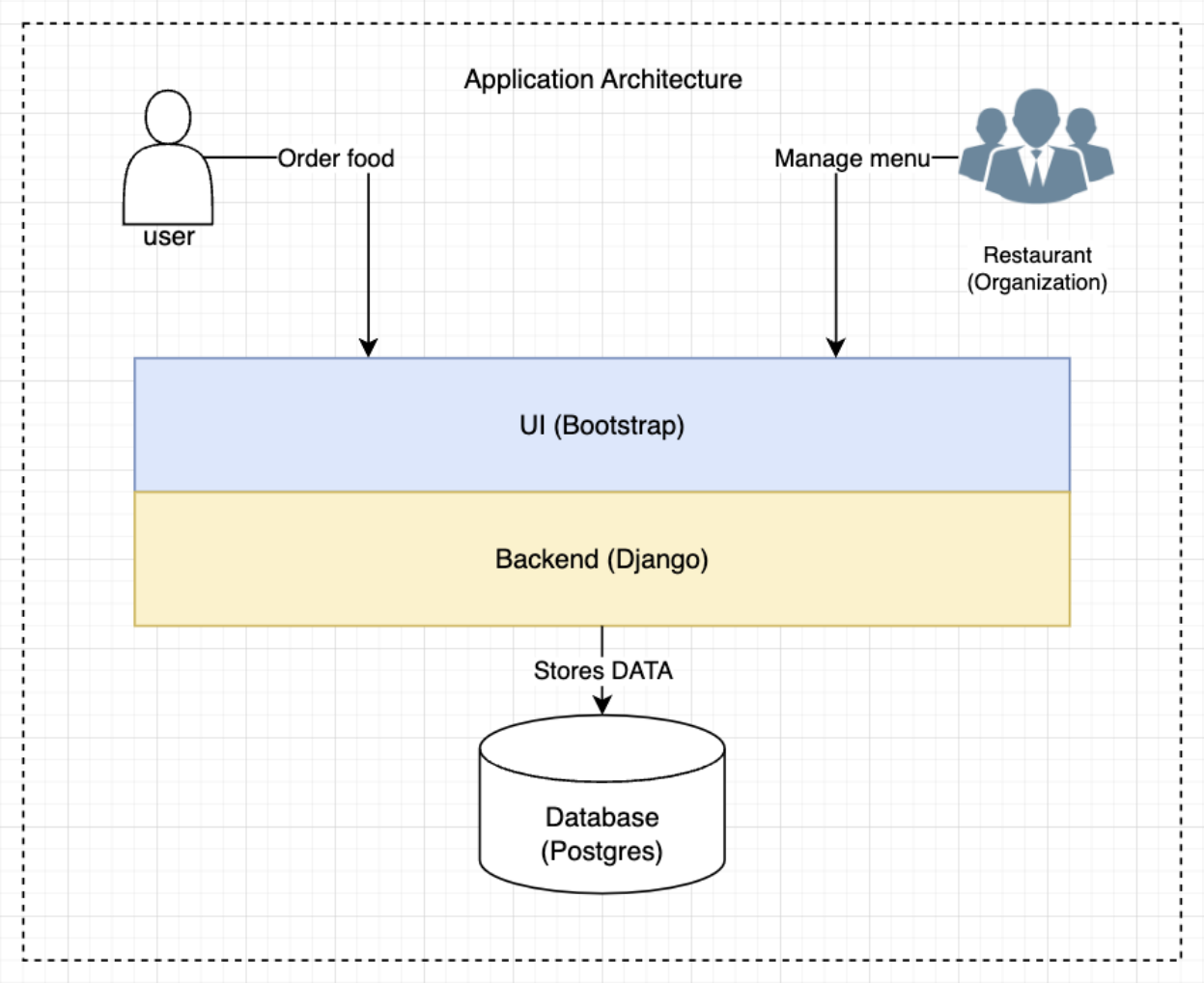

Figure1. Application Architecture

The Application Architecture Diagram provides a high-level overview of the structure of the Online Food Delivery Platform. The architecture is divided into three primary layers:

1. Presentation Layer (Frontend | UI):
    a. **Bootstrap**: This framework is used for creating a responsive and mobile-first user interface. Ensures that users have a seamless and consistent experience across different devices.
2. Business Logic Layer (Backend):
    a. **Django**: A high-level Python web framework that handles the core functionalities of the platform. Facilitates rapid development and follows the DRY (Don't Repeat Yourself) principle, making the platform efficient and scalable.
3. Data Layer:
    a. **PostgreSQL**: An advanced, open-source relational database system that stores data. Provides a reliable and robust database solution capable of handling complex queries and large volumes of data.

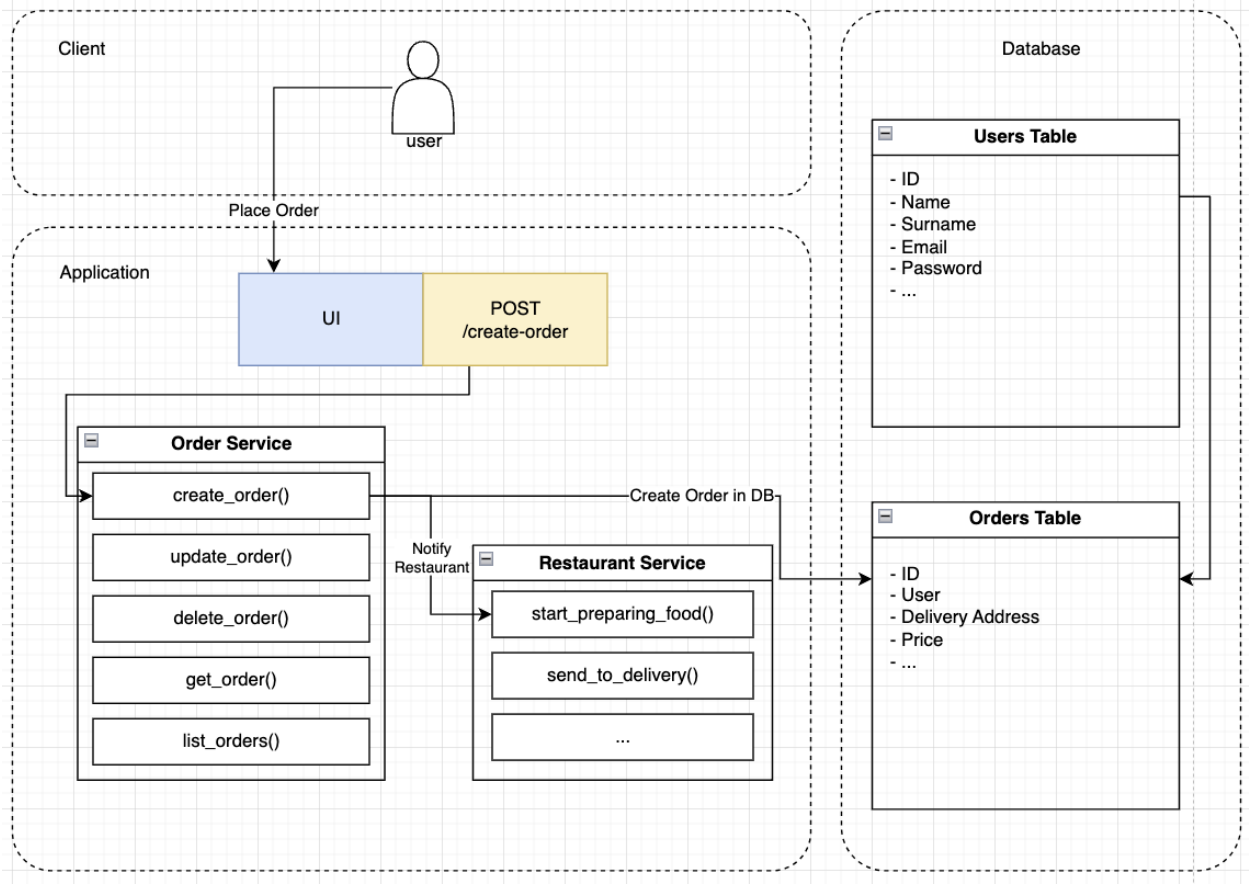

Figure 2. Order Placement Process

Design And Implementation Procedure:

The design and implementation of the online food delivery platform followed a systematic procedure to ensure that all functional and non-functional requirements were met effectively. We used the Agile methodology to facilitate iterative development and continuous feedback. The key steps of the design and implementation procedure included:

Requirement Analysis and Planning:

- Drafted detailed functional and non-functional requirements, ensuring they were measurable and specific.
- Prioritized features and requirements to align with Agile sprints.

Architectural Design:

- Developed an architecture that separates concerns into layers: Frontend (Bootstrap), Backend (Django), and Database (PostgreSQL).
- Created architecture diagrams to understand system structure and component interaction clearly.

Implementation:

- Set up the development environment using Django for backend development, PostgreSQL for database management, and Bootstrap for frontend design.
- Managed dependencies using Poetry for package management. This ensures that all necessary packages and their versions are tracked efficiently.
- Used GitHub for version control to maintain a history of code changes, collaborate effectively, and manage project branches.

Frontend Development:

- Utilized Bootstrap to design a responsive user interface that works seamlessly across different devices.
- Implemented views for user registration, authentication, category/product listings, and order placement.

Backend Development:

- Built the core logic using Django, leveraging its built-in functionalities for rapid development and adherence to the DRY principle.
- Used psycopg2-binary to manage database connections, enabling efficient data storage and retrieval.

Implemented functional requirements

User registration and authentication

- Implemented views for user registration using main Django authentication views

Food menu display

- Added Category and Product list and detail views

Order Placing

- Implemented order creation form
- Added Order list and detail views

Testing of application

To ensure that application works as expected we implemented testing. We used Python pytest library and TestCase classes provided by Django. It helps us to write simple and isolated integration tests, without affecting already existing data.

Implemented Tests:

- **Orders**: Created OrderIntegrationTests class with following tests
    - test_order_create
    - test_order_list
    - test_order_detail

- **Products**: Created ProductIntegrationTests class with following test
    - test_product_list
    - test_product_detail
- **Product Categories**: Created ProductCategoryIntegrationTests class with following tests
    - test_category_list
    - test_category_detail
- **Restaurants**: Created RestaurantIntegrationTests class with following tests
    - test_restaurant_create
    - test_restaurant_admin_create

References and links to externals frameworks, libraries and tools

Django: A high-level Python web framework used to handle backend development

- Official Django Website
- Chosen for its rapid development capabilities, robust security features, and scalability

Bootstrap: A frontend UI framework to ensure responsive and mobile-first design

- Bootstrap Official Webpage
- Provides a consistent design and layout across devices, enhancing user experience

Poetry: A package management tool for Python to manage dependencies

- Poetry Official Website
- Ensures that all project dependencies are managed, and versions are controlled effectively

Pillow: A Python Imaging Library used to render images of products and categories

psycopg-binary: Used to connect Django to the PostgreSQL database

Lessons Learned

The project underscored the importance of detailed planning and iterative development. Agile methodology was crucial in adapting to changing requirements and incorporating feedback throughout the development process. Key lessons learned include:

- **Effective Use of Tools**: Dependency management using Poetry and version control with GitHub provided significant advantages in maintaining consistency and collaboration
- **Importance of Testing**: Automated tests using Django TestCase improved the project's reliability and facilitated early detection of bugs, saving time during final stages
- **System Scalability**: Architectural decisions, such as the choice of Django and PostgreSQL, ensured the platform could handle increased loads and scale efficiently

Glossary of terms

- **User:** an individual who utilizes the platform to order food
- **Restaurant:** a business entity offering food on the platform
- **Sprint:** a set period during which specific work has to be completed and made ready for review
- **User Story:** A brief description of functionality typically written from the user’s perspective.
- **Payment Gateway:** A service that authorizes credit card payments and other forms of transactions.
- **Real-Time Tracking:** A feature that allows users to see the current status of their food delivery in real-time.
- **POS (Point of Sale) System:** A combination of software and hardware that allows businesses to complete sales transactions.