
50th year
HACETTEPE UNIVERSITY
www.hacettepe.edu.tr
To the leading edge... Toward being the best...

BBM384 Software Engineering Laboratory

Hungry Users Food Delivery System (HU-FDS)

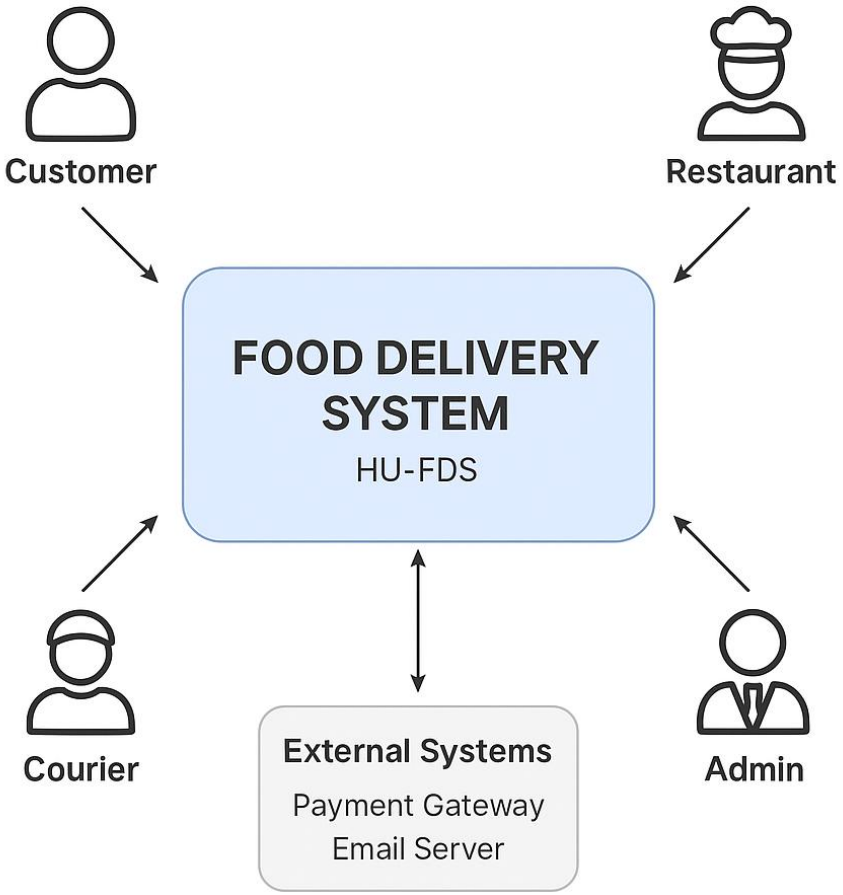
Application Type: Web-based

<Team28 & Name>

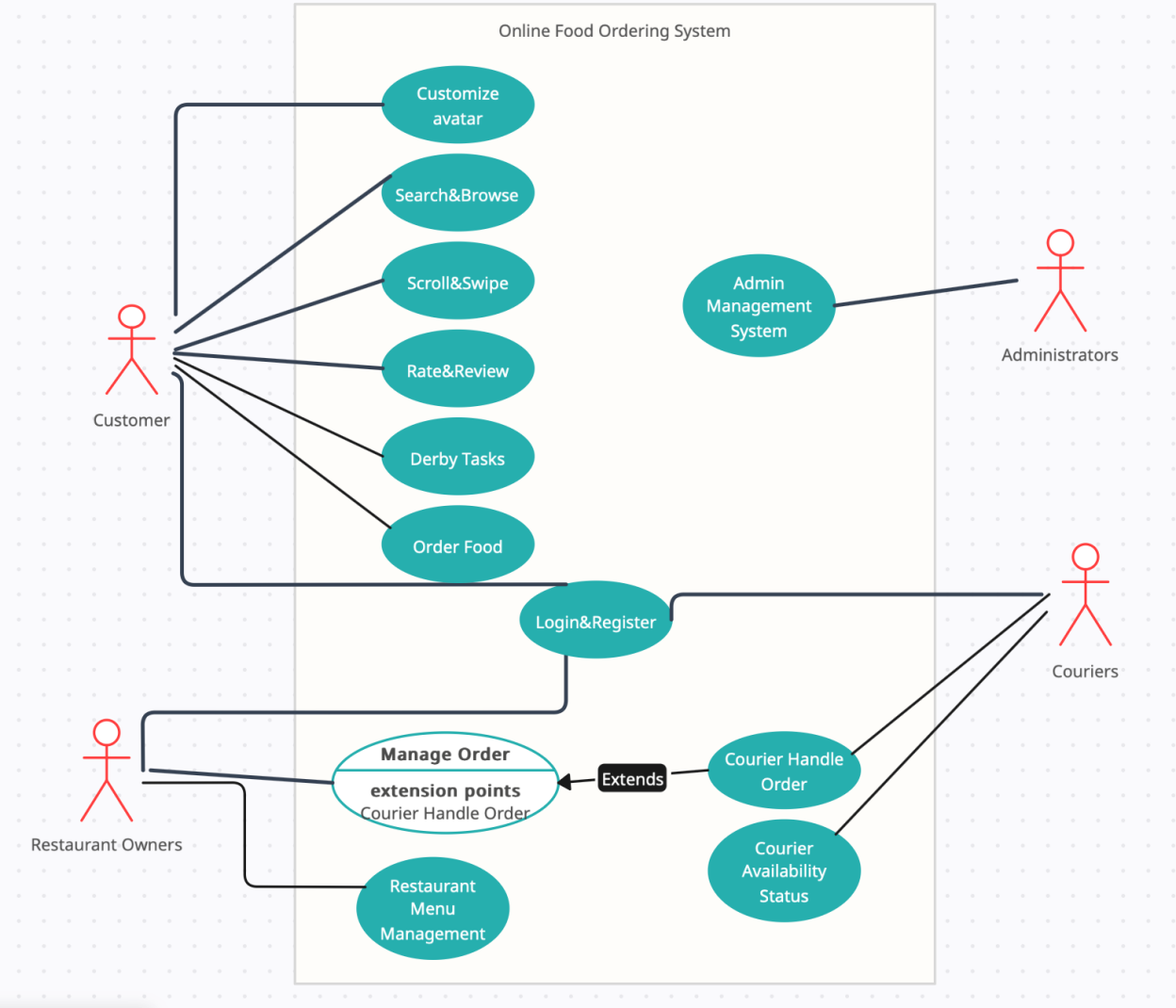
Team Members:

Specialized Team Role	Student No	Student Name	Student Surname
Software Project Manager	2210356067	Gökdeniz	Şimşek
Software Analyst	2210356079	Cansu	Aslan
Software Architect	2220356055	Mert Can	Köseoğlu
Software Configuration Manager		Yavuz Selim	Çakır
Software Tester	2210356066	Zeynep Nisa	Karataş

Proiect Context



Functional Software Requirements



Non-Functional Software Requirements

✅ Performance

- 🕑 **Response Time**: < 2s
- 📈 **Throughput**: 10 orders/min
- 👥 **Scalability**: 100 concurrent users

🔁 Reliability & Availability

- ✅ 99.5% uptime
- 💾 Data recovery in < 15 mins
- ❌ Error rate < 1%

🔐 Security

- 🔑 JWT-based authentication
- 🔒 AES-256 encrypted data
- 👥 Role-based access (RBAC)

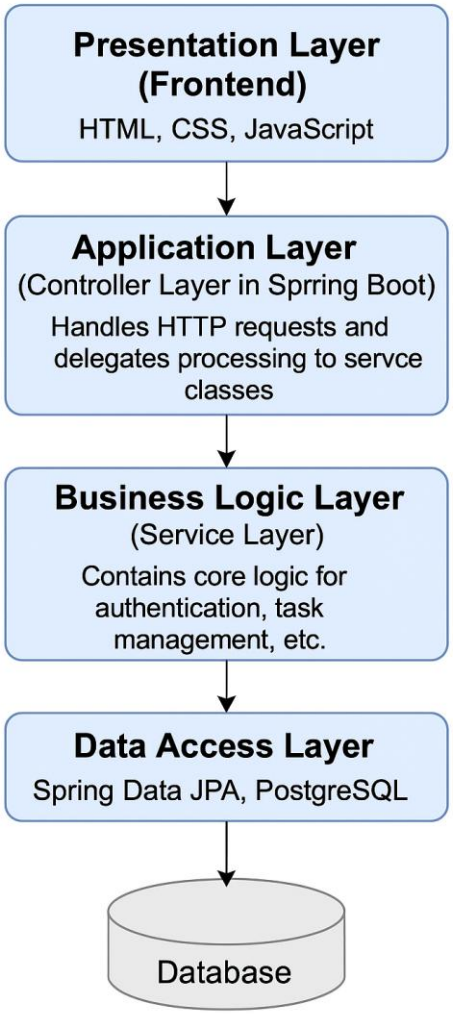
⚙️ Maintainability & Portability

- 🦎 Critical bugs fixed in < 24h
- 🌐 Runs on all major browsers

🧠 Usability

- 👤 First order in < 3 mins
- ✔️ 90% task success rate

Software Architecture



Technological Background

🧱 Backend – Java Spring Boot

- `Spring Web` → RESTful API endpoints
- `Spring Security + JWT` → User authentication & authorization
- `Spring Data JPA` → Database operations with PostgreSQL
- `Maven` → Dependency & build management

🖥️ Frontend – HTML / JavaScript

- `Vanilla JavaScript` → Dynamic client-side behavior
- `Fetch API` → REST API calls
- `Font Awesome` → UI icons
- `CSS` → Responsive styling

🛠️ Development Tools

- `Visual Studio Code` → Code editing
- `Postman` → API testing
- `Git & GitLab` → Version control and collaboration

🗄️ Database

- `PostgreSQL` → Relational data storage
- `TablePlus` → GUI for managing the database

Final Project Demo

- <Run the most recent version of your product.>

Work Allocation

In our HUDFS-28 food delivery system project, we adopted a **collaborative team-based approach** to development. Rather than working in isolation, we made a conscious effort to involve every team member in most phases of the project life cycle. This includes:

- **Requirement Gathering and Planning:** We held joint discussions to outline features, set milestones, and define user roles (customer, restaurant, courier, admin).
- **Design and Architecture:** We contributed together to the UI/UX layout and overall system architecture, making key decisions collectively.
- **Development Tasks:** While we each had areas of focus—such as backend, frontend, or database management—we frequently worked together on debugging, testing, and integrating components.
- **Testing and Troubleshooting:** Bugs and issues were resolved as a team, with everyone reviewing and improving each other's code.
- **Presentation and Documentation:** Content was prepared cooperatively to ensure consistency and full coverage of the system features.

Overall, our team dynamic was one of **mutual support and shared responsibility**, ensuring the success of each module as part of a unified system.

Work Done By Team Member:

Mert Can Köseoğlu - 2220356055

As the **backend developer** of our food delivery system, I was responsible for designing, implementing, and maintaining the server-side logic and communication between the frontend and database.

In **Demo 1**, I deployed our backend application to Render using our team's GitLab repository. This involved configuring the project structure to meet Render's deployment requirements, modifying the database connection settings to match Render's environment, and ensuring secure communication between the frontend and backend through proper CORS (Cross-Origin Resource Sharing) configuration. I also created and configured a Dockerfile to containerize our application and ensure consistency during deployment. This initial phase allowed us to demonstrate the early functionality of our system online, although we encountered some delays due to deployment complexity.

During **Demo 2**, we transitioned our backend to a local environment for improved performance and easier debugging. I replaced the hosted database with a cloud PostgreSQL instance from Neon Dashboard and updated the application.properties configuration to reflect this change. To improve system security, I integrated **JWT** (JSON Web Token) **authentication**, enabling token-based login and access control for users. I implemented restaurant login and registration APIs, and created full CRUD operations for products. Additionally, I developed APIs to list products and menus on the homepage and menus on the explore page. I added endpoints to like and dislike menu items, and integrated like/dislike counts directly into the explore view. Admin functionalities were also expanded: I created APIs to list all system users (customers, couriers, restaurants) and enabled the admin to ban users via delete operations. I also developed endpoints to allow restaurants to add and delete addresses, ensuring accurate delivery information.

In the **final phase**, I focused on implementing more advanced backend features directly impacting user experience and interaction. I developed a complete cart system that allows customers to add items and update quantities. I added avatar management capabilities: admins can create avatars, and customers can select their avatar, which is displayed on their profile along with their current level and accumulated points. I built an API allowing customers to assign from available derby tasks and unassign them if desired. The most complex logic I implemented was in the order placement system. When a customer places an order, the system checks whether it is linked to an assigned derby task; if so, it updates the task's progress, and if completed, awards the task's bonus points to the customer. Each order also grants 10 base points. I created APIs to list all orders for every user role. I designed a multi-role order status update system: customers can cancel their orders, restaurants can mark orders as prepared or ready, and couriers can update statuses to delivering or delivered. These functionalities complete the transactional aspects of our system.

Work Done By Team Member:
Gökdeniz Şimşek - 2210356067

As the **backend developer** of our food delivery system, I was responsible for designing the system architecture and implementing the backend functionalities that connect the frontend with the database.

In **Demo 1**, I started by designing and implementing the **project structure** and creating the **entity classes** along with their respective **repositories**. I helped my teammate Mertcan with the **initial database setup and testing**. During this phase, I ensured the foundational components were in place and ready for further development.

In **Demo 2**, I developed the complete **registration and login system** for both **customers and couriers**, including account deletion support. I implemented **menu management functionalities** for restaurants, including **create, update, and delete** operations. I designed and implemented the **customer task (Derby Task) system**, where the **admin can define tasks** and assign them to customers. I also developed the logic for displaying these tasks in the customer's profile. Furthermore, I added features that allow customers to **update their profile information**, and **add or delete multiple addresses and credit cards**, all managed through separate endpoints.

In the **final phase**, I introduced several advanced features. I enabled **editing of previously added addresses**, and implemented a system that reflects **restaurant availability** (open/closed) and **product/menu stock status**, integrating it across all order-related processes. I developed **courier profile update functionality** and implemented logic to manage their **availability status**, making them unavailable upon order assignment and available again upon delivery. I implemented the **cart clearing mechanism after successful order placement**. I designed and integrated a **coupon system** that automatically assigns a coupon to a customer when their level increases based on accumulated points. Coupons can be applied at checkout and are automatically removed upon use. Additionally, I implemented a **rating and review system** where customers can leave feedback for restaurants based on their orders. These reviews are displayed on restaurant profiles and visible across relevant customer-facing pages.
This structured progression from core architecture to complex user-facing features ensured a scalable and user-friendly backend system.

Work Done By Team Member:

Zeynep Nisa Karataş - 2210356066

As the Frontend Developer of our team, I was responsible for creating responsive, user-friendly interfaces using HTML, CSS, and JavaScript. I collaborated closely with the backend team to integrate RESTful APIs, focusing on clean UI/UX, role-based interaction, and seamless navigation for customers, restaurants, couriers, and admins.

In **Demo 1**: In the first phase of the project, I focused on building the frontend for the Admin Management System. I designed and implemented user-friendly interfaces that allowed the admin to view, manage, and control access to all platform users—restaurants, couriers, and customers. This included developing listing views with clean layouts, actionable buttons for banning or deleting accounts. The goal was to provide the admin with full visibility and control in an intuitive and efficient interface.

In **Demo 2**: During the second demo, my responsibilities expanded to cover multiple critical features. I worked extensively on the Derby Task Management system, where I implemented dynamic forms for admins to create and assign tasks to customers, both individually and collectively. In parallel, I developed full CRUD functionality for restaurant menus and products, allowing restaurant owners to manage their offerings with ease. I also contributed to the authentication system, building role-based login and register interfaces for customers, couriers, restaurants, and admins and connecting authentication with backend.Additionally, I took charge of developing the profile management interface. This included structured input forms for updating personal details like name and phone number, as well as modular sections for managing addresses—allowing users to add, edit, and remove their delivery locations with validation support. On the customer side, I designed an interactive shopping cart that supported add and delete operations. This cart was tightly coupled with the ordering system, ensuring that users could only place orders from a single restaurant, as required by business logic.

In **Demo 3**: In the final demo, my work centered on enhancing the customer-facing features that drive engagement and satisfaction. I developed an avatar customization system, allowing users to select or change their visual representation from a set of predefined avatars. I also built the rate and review module, where customers could submit reviews and see restaurant feedback in a well-structured format featuring dynamic star ratings and customer comments.Furthermore, I completed the frontend for the ordering flow, ensuring that users could finalize their carts and track order statuses from placement through delivery, with real-time visual updates. Finally, I worked on the gamification elements of the platform—specifically the levels and points system. I created a clear and engaging display that showcased user progress, levels, and earned points, contributing to a more rewarding and interactive experience such as their coupons that they can use for each level for discount.

Work Done By Team Member:

Cansu Aslan - 2210356079

Demo 1:In the initial phase of the project, I was responsible for designing the core interfaces for all three primary user roles—Restaurant, Customer, and Courier. I developed intuitive, visually cohesive layouts that established the foundation for role-specific interactions and workflows. Each interface was tailored to its user type, ensuring a smooth and user-friendly experience that aligned with our platform's goals.

Demo 2:During the second milestone, I enhanced the homepage by implementing advanced search and filtering functionality, allowing users to quickly find menus and products based on category, price, and nutritional values. I also made significant improvements to the homepage layout, optimizing it for both usability and aesthetic appeal.One of my key contributions in this phase was the development of the Explore Page, a dynamic, visually appealing section designed to promote restaurant discovery and drive customer engagement. This page became a cornerstone of the user experience, helping customers navigate offerings in a modern, scrollable, and category-based interface.

Demo 3:In the final phase, my work focused on operational interactivity and visual refinement across the entire application. On the restaurant side, I implemented status controls that allowed restaurant owners to open or close their businesses dynamically. I also enabled restaurants to update order statuses (e.g., Pending, Waiting, Ready) which directly impacted courier workflows.I resolved multiple functional bugs, including fixing add and update address features, ensuring users could manage delivery details reliably.For the Customer role, I introduced the Add to Cart and Purchase Flow functionalities—key components that completed the ordering process. I also refined the Explore Page logic, improving category handling and interactivity.On the Courier side, I finalized pages that allowed couriers to view and manage their assigned orders. I added functionality for couriers to update their availability and change delivery statuses, streamlining the delivery workflow.Finally, I conducted a comprehensive visual pass, refining the aesthetic consistency, layout spacing, color harmony, and button styling across all pages to ensure a polished, professional UI/UX across the platform.

Self-Evaluation of Project Development

Mert Can Köseoğlu - 2220356055

Do I consider our project successful?

Yes, I believe our project was highly successful due to the following reasons:

- We completed all the **planned core features** and integrated them into a functioning platform.
- The backend was **stable and secure**, supporting authentication, database management, and smooth API communication.
- Team collaboration was excellent; we resolved issues quickly and made joint decisions that improved overall quality.

What would I do differently if starting over?

- I would **avoid attempting full backend deployment** in the first demo. It consumed valuable time and added unnecessary complexity at that stage.
- Instead, I would have **used local backend hosting** from the beginning, which we later switched to in Demo 2, allowing smoother testing and development.

Self-Evaluation of Project Development

Gökdeniz Şimşek - 2210356067

Do I consider our project successful?

- I think our project is working very well and meeting the requirements. Our project is good and stable.

What would I do differently if starting over?

- I don't think there's any need to make any changes. It's just that although we're very well-planned, it could have been a little more planned.

Self-Evaluation of Project Development

Zeynep Nisa Karataş - 2210356066

Do I consider our project successful?

Yes, I believe our project was successful because:

- All core frontend features were completed and worked across all user roles.
- Interfaces were responsive, user-friendly, and functioned smoothly.
- We had strong collaboration and seamless backend-frontend integration.

What would I do differently if starting over?

- I would build reusable UI components earlier to avoid repetition. I'd use mock APIs from the start to develop independently of backend delays. I'd set up role-based navigation earlier for cleaner structure.

Self-Evaluation of Project Development

Cansu Aslan - 2210356079

Do I consider our project successful?

- I think our project was successful. We delivered a complete, functional platform with a smooth and intuitive experience across user roles. The frontend was responsive, visually consistent, and well-integrated with the backend. Teamwork was strong, and the project was enjoyable to work on.

What would I do differently if starting over?

- If starting over, I'd focus on better structure from the beginning—building reusable components earlier, using mock data to reduce dependencies, and planning layout and navigation more proactively for long-term clarity.

Thank You!

- Any questions..?

References

OpenAI's ChatGPT – for help with logic, structure, and debugging tips

Java Spring Boot Documentation – official documentation for Spring Boot

JavaScript Documentation (MDN Web Docs) – comprehensive reference for JavaScript language features

JavaScript JWT Guide (JWT.io) – introduction and guide to JSON Web Tokens

Stack Overflow – for troubleshooting and resolving backend development challenges