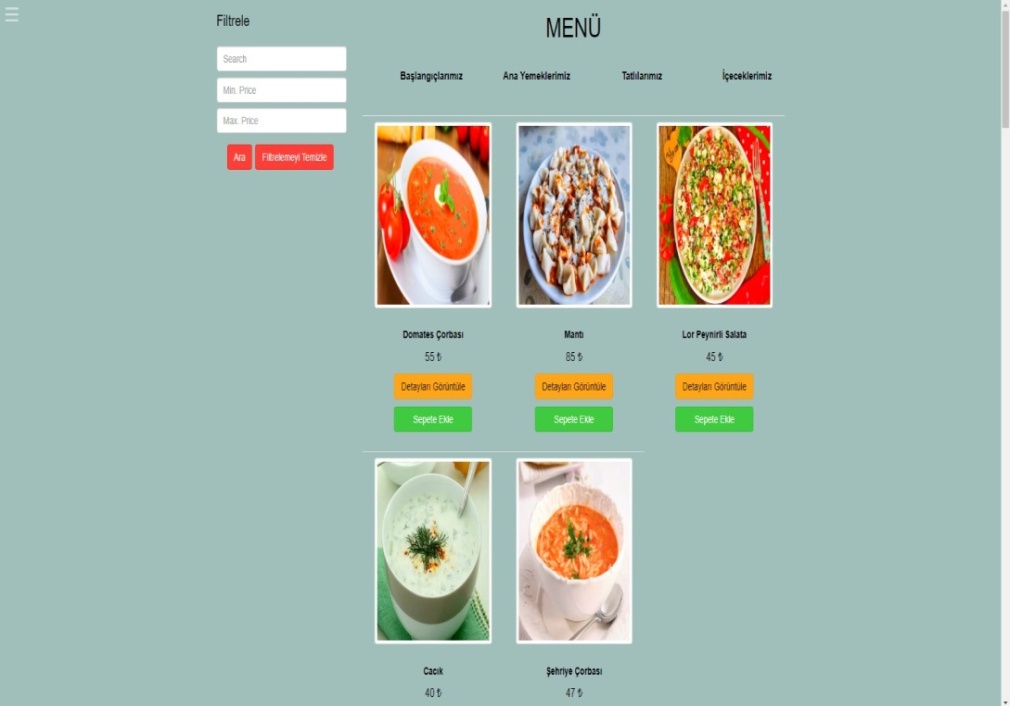
**Serhat GÜLHACI 200101021**

[A INVENTORY MANAGEMENT FOR RESTAURANT]



**Table of Contents**

**1.INTRODUCTION………………………………………………………………………………………………………………:4**

**1.1 Purpose and Scope…………………………………………………………………………………………………….:4**

**1.1.1 The Purpose of the Project…………………………………………………………………………………:4**

**1.1.2 Scope of The Project………………………………………………………………………………………....:4**

**1.2 Goals and Success Criteria………………………………………………………………………………………....:5**

**1.3 Overview………………………………………………………………………………………………………………..….:5**

**1.3.1 System requirements…………………………………………………………………………………………:5**

**1.3.2 Member Requirements…………………………………………………………………………….………..:5**

**1.3.3 Test Items……………………………………………………………………………………………….…………:6**

**1.3.3.1 Features To Be Tested……………………………………………………………………….…..….:6**

**1.3.3.2 Features Not To Be Tested………………………………………………………………..………:6**

**1.3.3.3 Item Pass/Fail Criteria………………………………………………………………………….……:6**

**1.3.4 Software Risk Issues…………………………………………………………………………………….…..:6**

**1.4 Data Models…………………………………………………………………………………………….……….….…..:7**

**1.4.1 E-R Diagram………………………………………………………………………………………………………:7**

**1.4.2 Relational Table…………………………………………………………………………………………….….:8**

**1.5. Software Processes and Process Models……………………………………………….…………………:8**

**1.5.1 Selected 5 Software Process Models and Reasons……………………………………….……:8**

**1.5.2 Decided Software Process Model and Reason……………………………………..……….……:9**

**2. Requirements……………………………………………………………………………………………………….….….:10**

**2.1 Functional Requirements…………………………………………………………………………………….…..:10**

**2.2 Non-Functional Requirements…………………………………………………………………….……….….:10**

**2.2.1. In Terms of Security…………………………………………………………………………………….....:10**

**2.2.2. In Terms of Usability……………………………………………………………………………….…..….:10**

**2.2.3. In Terms of Performance……………………………………………………………………….…..……:11**

**2.2.4. In Terms of Supportability (Maintenance)……………………………………………………...:11**

**2.2.5. In Terms of Constraints (System Constraints):…………………………………….……..……:11**

**3. System Modelling with UML……………………………………………………………………………….……..:12**

**3.1 Structural Models…………………………………………………………………………………………….….….:12**

**3.1.1 Class Diagram……………………………………………………………………………..……………………:12**

**3.1.2 Object Diagram…………………………………………………………………………………………..….:13**

**3.1.3 Component Diagram……………………………………………………………………………….……..:14**

**3.1.4 Deployment Diagram……………………………………………………………………………………..:14**

**3.1.5 Composite Structure Diagram………………………………………………………………………...:15**

**3.1.6 Package Diagram………………………………………………………………………………………..…..:16**

**3.1.7 Data-Flow Diagram………………………………………………………………………………………...:16**

**3.2 Behavioral Models………………………………………………………………………………………………….:17**

**3.2.1 Use-Case Diagram……………………………………………………………………………………….….:17**

**3.2.1 Use-Case Diagram(Developer-Admin)……………………………………………………………...18**

**3.2.2 Activity Diagram for User……………………………………………………………………………..…:19**

**3.2.2 Activity Diagram for Admin/Courier………………………………………………………..……..:20**

**3.2.3 State Machine Diagram…………………………………………………………………………………..:21**

**3.3 Interaction Models……………………………………………………………………………………………….…:22**

**3.3.1 Interaction Overview Diagram…………………………………………………………………..…….:22**

**3.3.2 Sequence Diagram for Admin………………………………………………………………………....:23**

**3.3.2 Sequence Diagram for User………………………………………………………………….………….:24**

**3.3.3 Timing Diagram……………………………………………………………………………………….……….:24**

**3.3.4 Collaboration Diagram……………………………………………………………………………………..:25**

**3.3.5 Communication Diagram…………………………………………………………………………….……:26**

**4. Software Testing……………………………………………………………………………………………………..……:27**

**4.1 Development Testing……………………………………………………………………………………………….:27**

**4.2 Unit Testing………………………………………………………………………………………………………………:28**

**4.2.1 White Box Testing……………………………………………………………………………….……………:28**

**4.3 Integration Testing……………………………………………………………………………………….…………..:29**

**4.3.1 Big-bang……………………………………………………………………………………………………………:30**

**4.3.2 Top-down……………………………………………………………………………………..………………….:31**

**4.3.3 Bottom-up…………………………………………………………………………………………………..……:32**

**4.3.4 Hybrid……………………………………………………………………………………………………………….:33**

**4.4 System Testing…………………………………………………………………………………………………………..:34**

**4.4.1 Black Box Testing…………………………………………………………………………………………….…:34**

**4.5 User Testing/Acceptance Testing………………………………………………………………………………..:34**

**4.5.1 Alpha Testing………………………………………………………………………………………………….…..:34**

**4.5.2 Beta Testing………………………………………………………………………………………………………..:35**

**4.5.3 User Acceptance Testing…………………………………………………………………………………..…:36**

**4.6 Debugging…………………………………………………………………………………………………………………:36**

**4.6.1 Debugging by Brute Force …………………………………………………………………………………:36**

**4.6.2 Debugging by Induction……………………………………………………………………………….……:36**

**4.6.3 Debugging by Deduction……………………………………………………………………………………:37**

**4.6.4 Debugging by Backtracking………………………………………………………………………….……:38**

**5. Identification of Project Risks and Project Risk List………………………………………………..…:39**

**5.1 Identification of Project Risk…………………………………………………………………………………..…:39**

**5.2 Project Risk List…………………………………………………………………………………………………………:39**

**5.3 Planning Risks and Contingencies…………………………………………………………………………..…:40**

**1.INTRODUCTION**

**1.1 Purpose and Scope**

The aim of the project is to help people who want to order home-cooked meals to order cheap and suitable to their taste online.

* + 1. **The Purpose of the Project**

This project was produced specifically for a restaurant and is managed by the admin (restaurant owner). The project contains information about the dishes in the menu (nutritional value, recipe...). At the same time, meal controls or other controls are managed by the admin (user operations, meal addition-deletion, remaining meal information, relationship with the courier, etc.). Users can access detailed information about each dish. At the same time, users can control their accounts (add-delete-edit password, e-mail, address). Users can track the meals they order (food is being prepared, courier is on its way, delivered...). The courier easily performs operations such as delivering the food and setting off on the road from his/her account and presents it to the user and admin information. The project is easy to use and provides detailed information.

* + 1. **Scope of The Project**

The project was specially produced for a restaurant and is open to everyone. The project was produced so that people can order food easily and quickly. There are admin, user and courier roles in the project. The user can place an order practically, the admin can control the entire system, and the courier informs whether the order has been delivered or not.

**1.2 Goals and Success Criteria**

**Goals:**

* Customer happiness
* Protect personal information
* Fast and reliable delivery
* Serving clean and delicious food
* Food suitable for every region
* Ease of Use
* Ensuring calorie control with nutritional values
* Offer a variety of products

**Success Criteria:**

* Appeal to the eye with the beauty of the background.
* A website that users of all levels can easily use and be satisfied with.
* Regular interaction between courier admin and user.
* Easily organize the project to suit every generation.
* Smooth operation of the project.
* Coordinated work among project designers.

**1.3 Overview**

**1.3.1 System requirements:**

* Must be user login
* There should be a registration screen for the user
* Must be admin login
* Must have courier login
* Must have food information
* Must have basket to add dishes
* There should be a payment screen
* There should be an information update screen
* There should be an order tracking system
* User should be able to view past orders
* The manager should be able to see the status of the courier and the order
* Administrator must be able to see and control users

**1.3.2 Member Requirements:**

* User should be able to update personal information
* The user should be able to add and remove food from the cart
* User should be able to view past orders
* There should be a registration screen for the user
* Each member must have an email and password
* Making the authorization

**1.3.3 Test Items:**

**1.3.3.1 Features To Be Tested**

* Login/logout
* Inventory control
* Minimum price in cart
* Password-mail control
* Coupon validity
* Interface
* Record
* Database Security
* Limitations
* CRUD operations

**1.3.3.2 Features Not To Be Tested**

* Overloads
* Hardware errors
* Infrastructure adequacy
* Server capacity

**1.3.3.3 Item Pass/Fail Criteria**

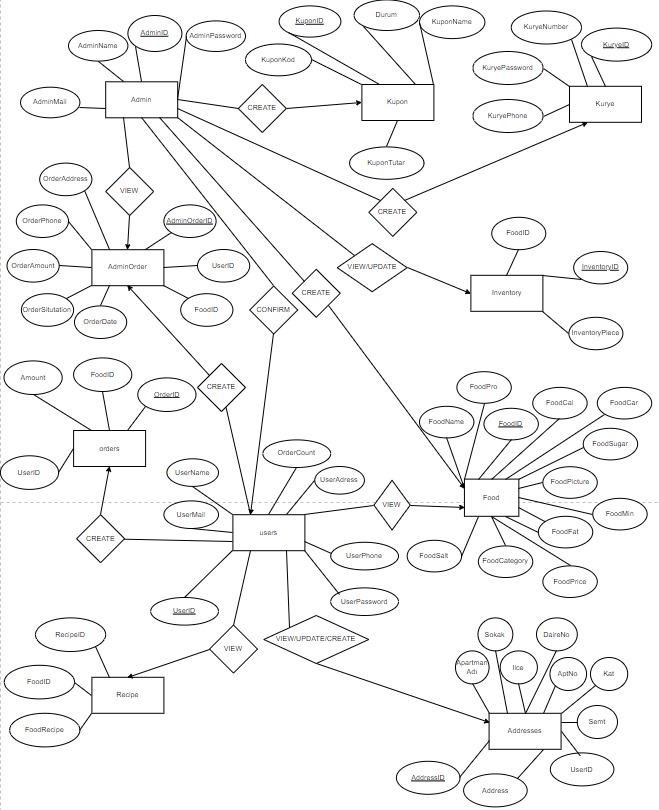
* Have security
* Ordering correctly
* Correct storage of data
* Performing admin operations correctly
* Performing courier operations correctly
* Being able to do the desired situations

**1.3.4 Software Risk Issues:**

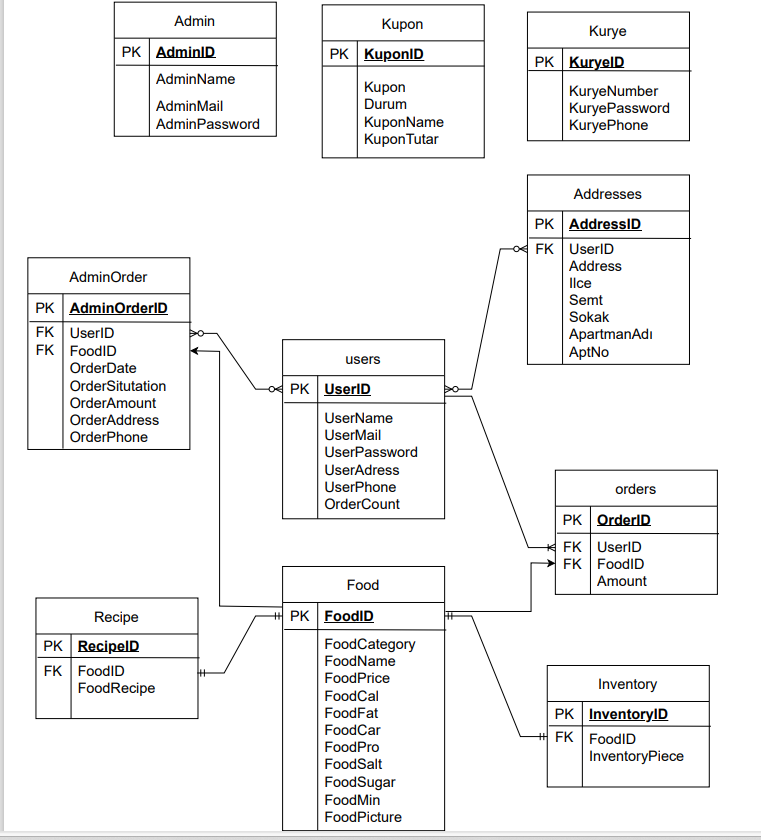
* Complex code
* Logic errors
* Server errors
* Open source code

**1.4 Data Models**

**1.4.1 E-R Diagram**

****

**1.4.2 Relational Tables**

****

**1.5 Software Processes and Process Models**

**1.5.1 Selected 5 Software Process Models and Reasons**

**Incremental Development Model:**Incremental development model breaks the project into smaller parts.Thanks to it,we can quickly create a prototype and then refine these parts and ensuring early delivery. Since we can detect errors early, we can complete our project by avoiding more labor and cost. At the same time, thanks to the prototype we present to the customer, we can make corrections according to customer requests. However, since we have a testing and integrated process for each stage, we cannot adapt to complex projects and increase costs.

**Agile Model:** Thanks to the flexibility it provides, the Agile model helps us quickly adapt to changing needs and make adjustments. In this way, we can provide value to the customer faster by making continuous deliveries in small iterations. Because the Agile model is customer-focused, itencourages customers to be more involved in the process. At the same time, the agile model can strengthen communication skills as it helps us establish interaction between teams. However, when there is disagreement and harmony problems among the team, it creates problems for the project and the customer. If all our requirements are not clear at the beginning of the project and we are constantly making changes to the project, the uncertainty that will arise makes it difficult to control the project. In addition, constantly involving the customer in the project may have negative consequences in terms of time and cost.

**Evolutionary Model:**Evolutionary model is commonly used when the customer wants to start using the core features instead of waiting for the full software.We create a basic prototype to start with and make changes in line with the demands of the users. While the desired changes allow us to better understand user requirements, they may make comprehensive planning difficult and increase uncertainty throughout the entire project process. We involve the customer throughout the project and attach importance to customer satisfaction, but this may be more costly because every change in request increases the cost and makes it difficult to control the project.

**Prototyping Model:** This model is used in cases where customers do not know the project requirements exactly in advance. A prototype of the actual product is developed according to the customer's feedback. Thanks to this feedback, we can understand the project better and detect errors earlier, thus obtaining a user-friendly product. Thanks to the prototype, we create a more comprehensive product and this allows the project to be operational for a longer period of time. It is possible to quickly adapt to changing requirements that arise during the prototype phase. This supports a more flexible and adaptable development process.

**Rapid Application Development:**RAD is a software development model that aims to enable rapid application development. The RAD model allows us to develop software quickly by constantly creating prototypes and basing these prototypes on user feedback. The RAD model offers a flexible approach that can quickly adapt to changing requirements thanks to the reusable component. This helps our project better adapt to a dynamic environment. It strengthens the communication between the project team and the customer, as we work in collaboration with customers. Continuous testing of prototypes and improvement with customer feedback often allows us to achieve a higher quality product.

**1.5.2 Decided Software Process Model and Reason**

**Rapid Application Development:** We chose the RAD model because our project has multiple reusable components and involves minimal planning. In addition, it has powerful techniques and tools so that we can produce more effective projects with more customer participation in a short time.

In the Agile model, customer participation is constantly encouraged, but we only want customers to participate at the prototype stage so we can deliver products faster. There is detailed documentation in the incremental development model. We think that this much documentation will prolong the process. We want to develop the project in short and frequent sprints, but in the evolutionary model, development is generally divided into large and specific stages. We did not choose a protoyping model because using reusable components makes our project easier to understand and faster, and this model does not contain reusable components.

**2. Requirements**

**2.1 Functional Requirements**

* Registration of new customers.
* User logs in.
* User can see the food list.
* Consumed dishes are removed from the list.
* User the dishes you want to order can be added to the cart.
* User can apply discount coupon.
* User can pay for the order by credit card.
* User can update quantity of food in cart
* User can filter foods.
* The user can update his information.
* The user sees the history order.
* The user can review the food details.
* The user logs out.
* Admin logs in.
* Displays the food list with the consumed dishes.
* Admin can edit the details of the foods.
* Admin can add or subtract food.
* Admin can view user details.
* Admin can delete the user.
* Admin can confirm and cancel orders.
* Admin can view past orders.
* Admin logs out.
* The courier enters.
* The courier can report that he received and delivered the food.
* The courier logs out.

**2.2 Non-Functional Requirements**

**2.2.1. In Terms of Security:**

* Open to admin access.
* User will have a password and username
* Passwords are saved encrypted in the database

**2.2.2. In Terms of Usability:**

* Will have a login screen
* The system is available 24/7
* There is a navbar that shortcut
* There is filtering

**2.2.3. In Terms of Performance:**

* Speed
* An understandable screen
* Payment response should be as fast as possible
* Food list screen’s load time should not be more than three second for users
* When the admin and the courier update the order status, the user can see it instantly

**2.2.4. In Terms of Supportability (Maintenance):**

* Planning should be done
* Should be open to updates
* Food list should be updated

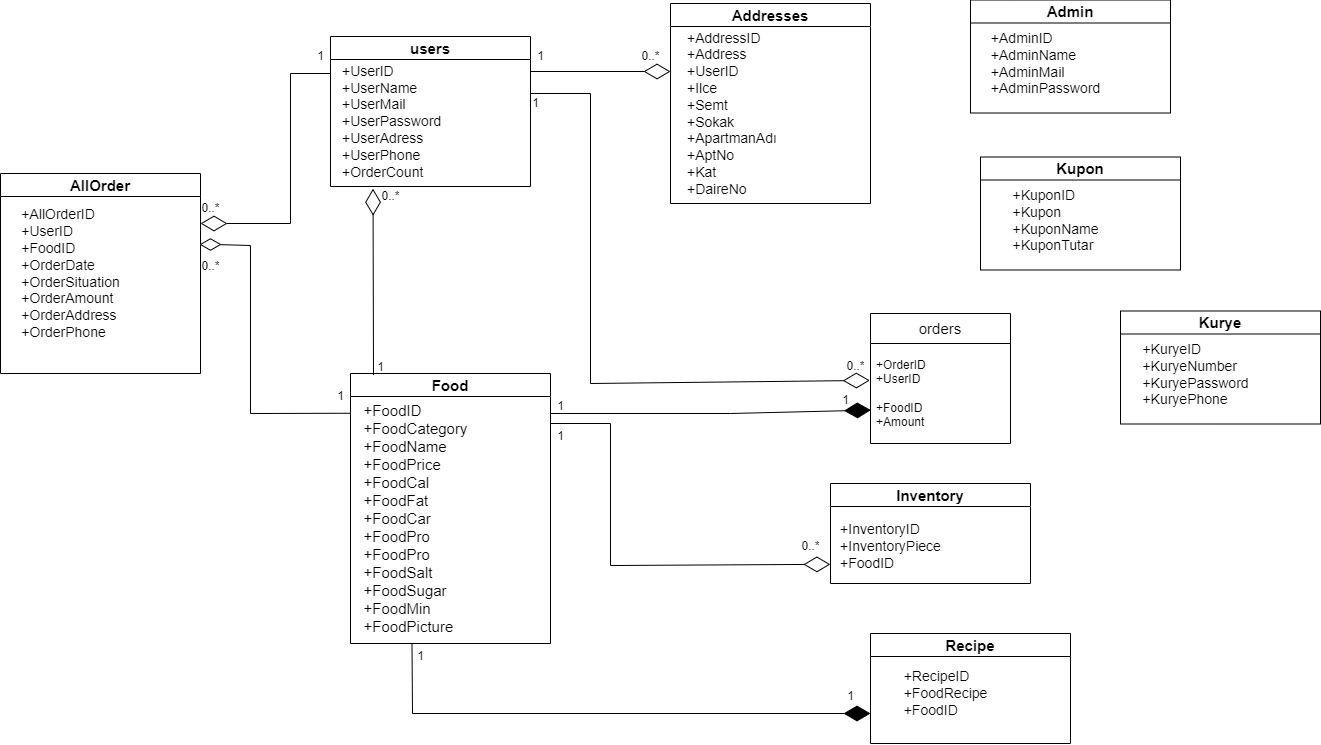
**2.2.5. In Terms of Constraints (System Constraints):**

* Assignments should be made(admin, courier, user)
* Password has to be minimum 4 digits
* Phone number must be 11 digits
* Order total in cart must meet minimum amount
* The same e-mail address cannot be used again when registering
* Information cannot be left blank on the registration screen

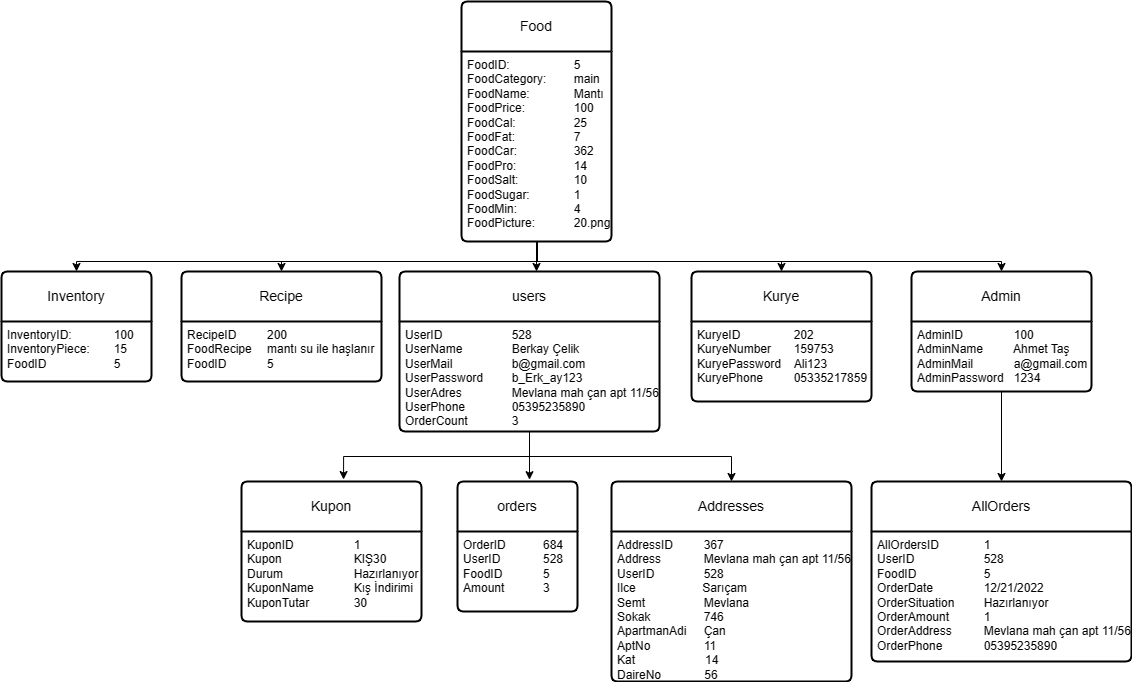
**3. System Modelling with UML**

**3.1 Structural Models**

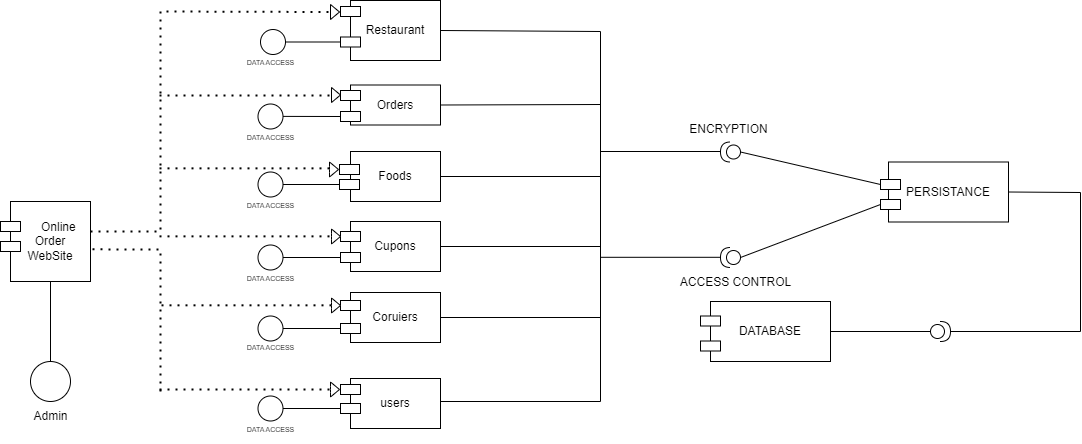
**3.1.1 Class Diagram**



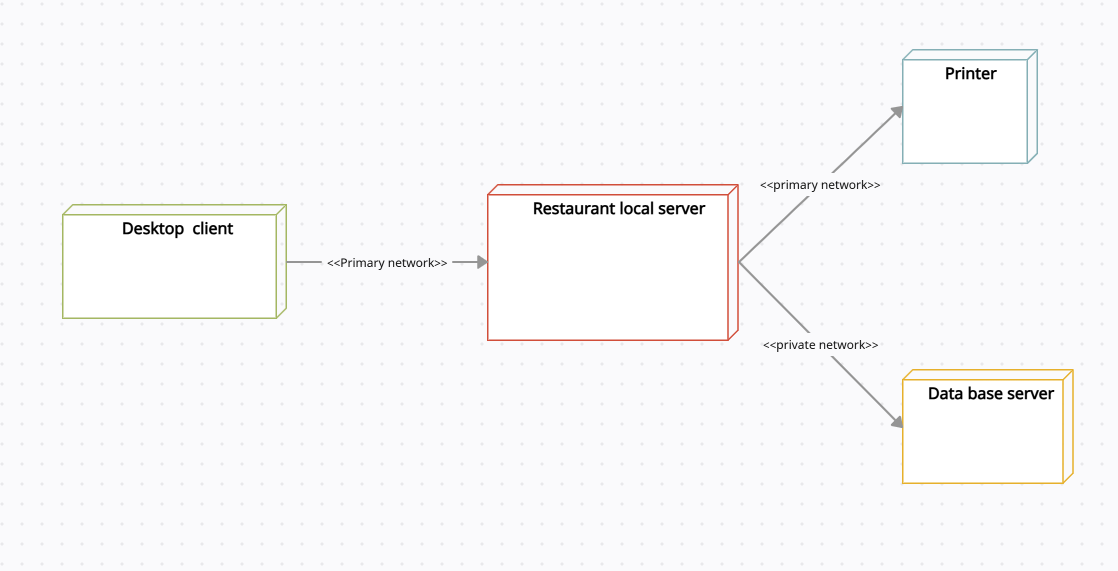
**3.1.2 Object Diagram**

****

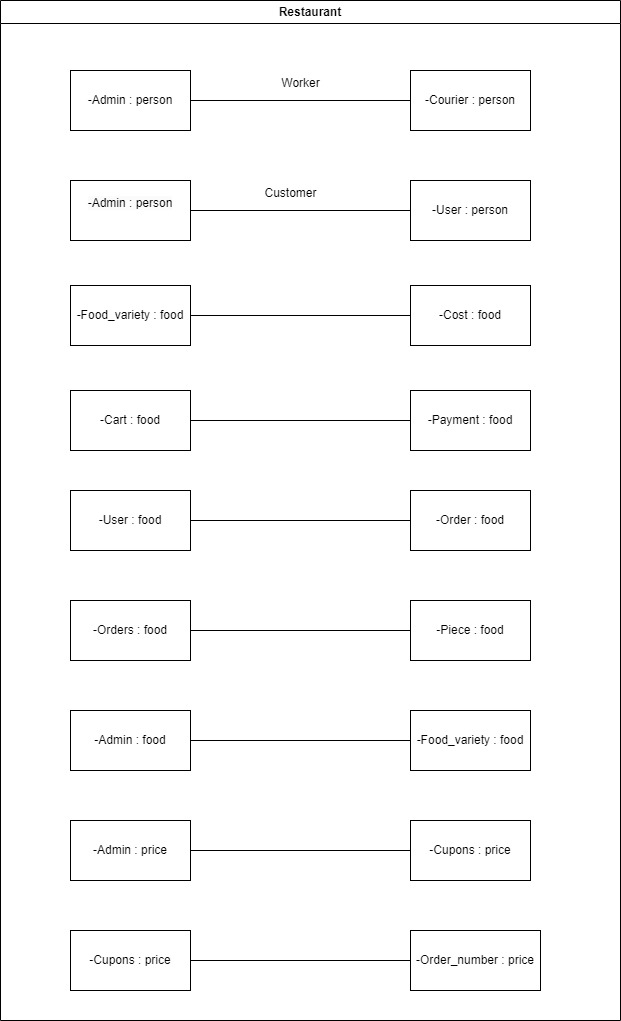
**3.1.3 Component Diagram**



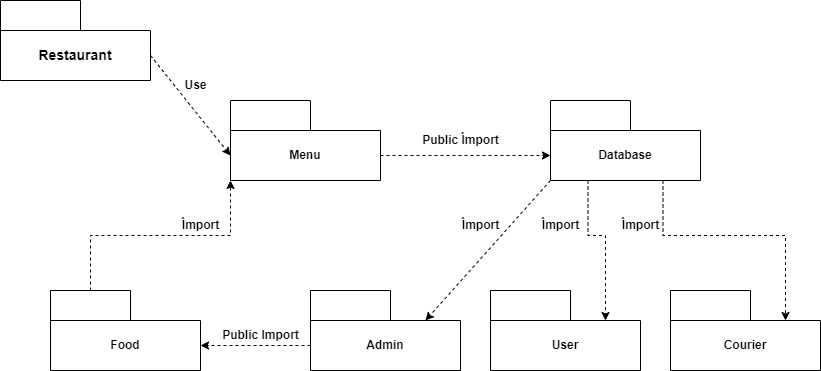
**3.1.4 Deployment Diagram**



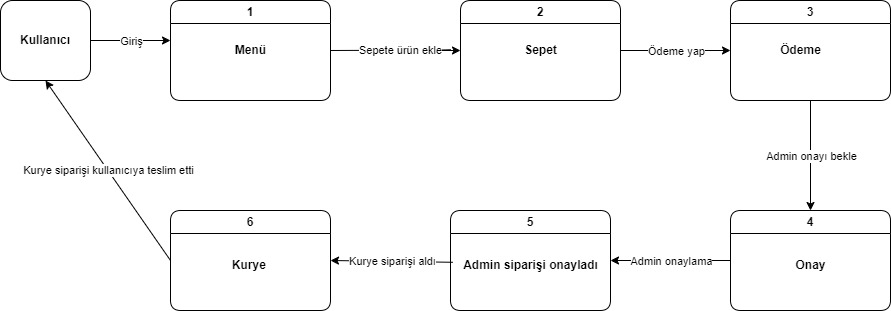
**3.1.5 Composite Structure Diagram**



**3.1.6 Package Diagram**

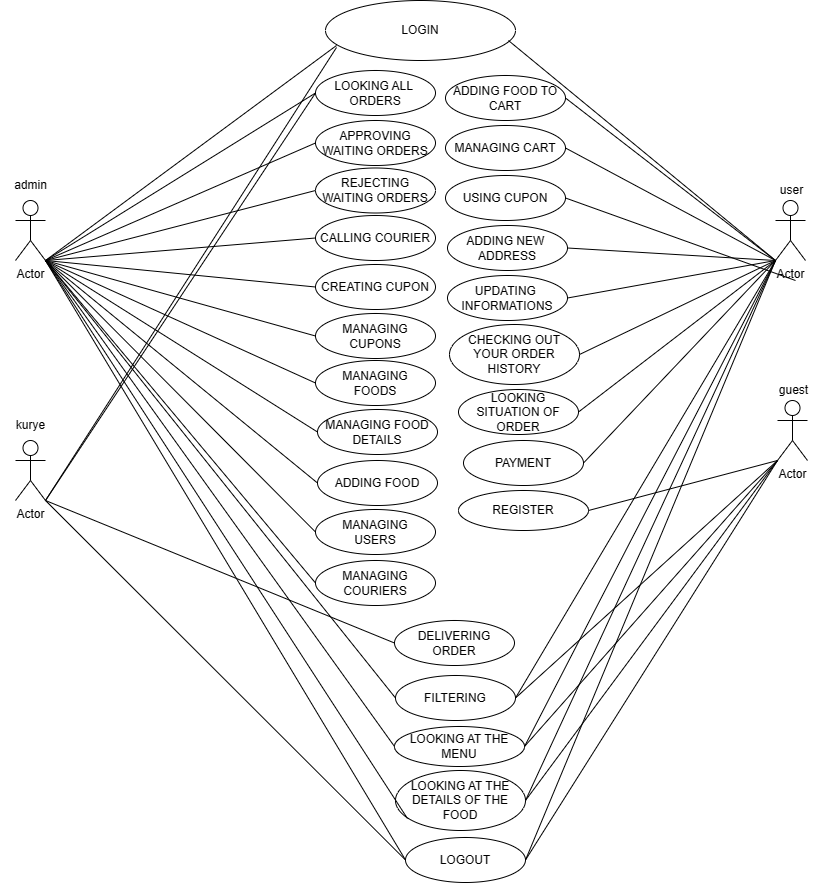


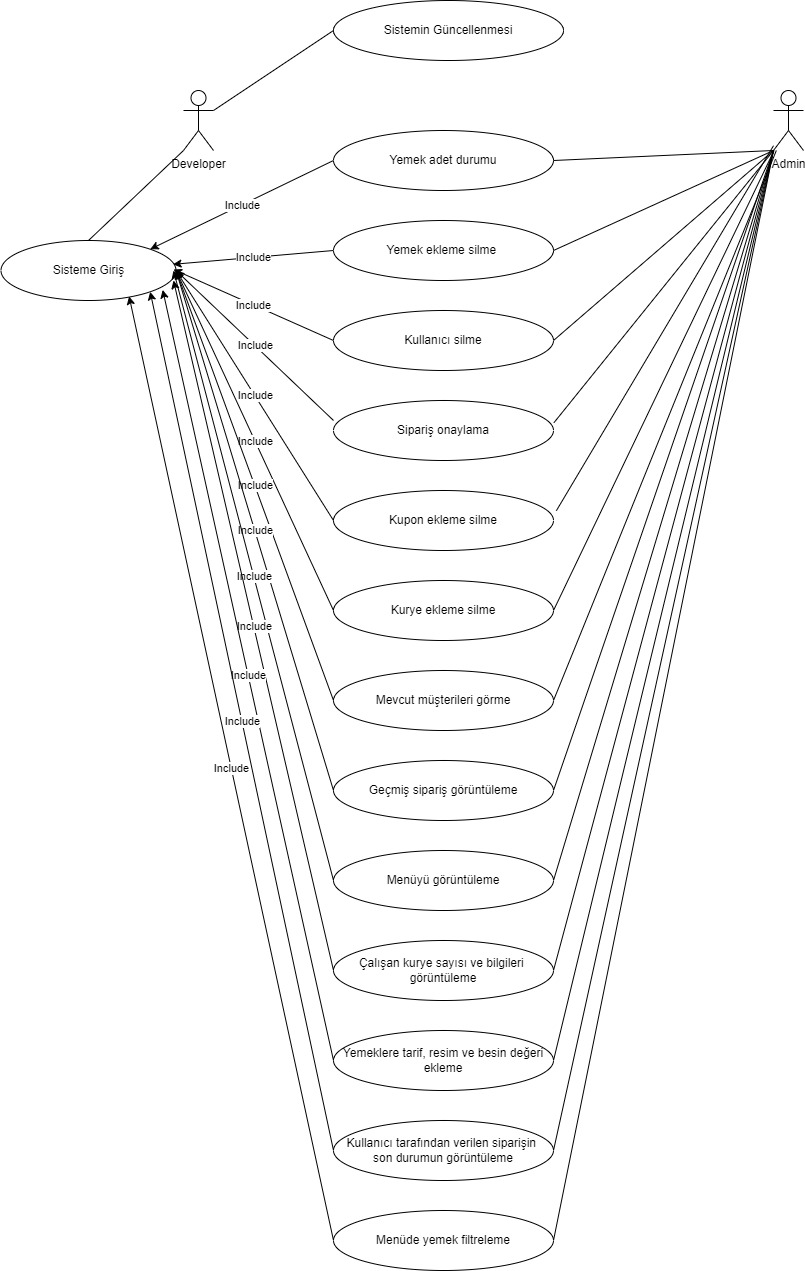
**3.1.7 Data-Flow Diagram**



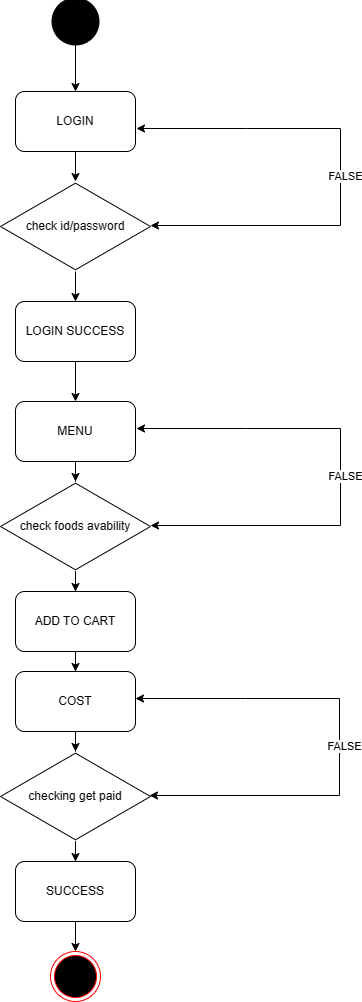
**3.2 Behavioral Models**

**3.2.1 Use-Case Diagram**

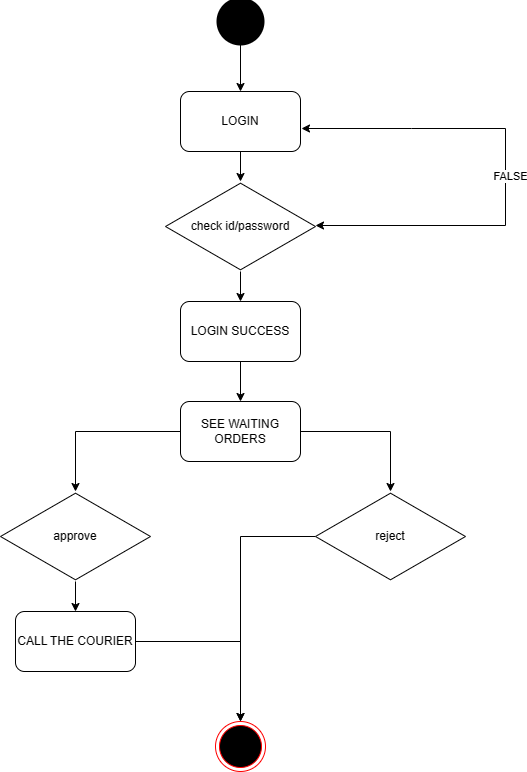


****

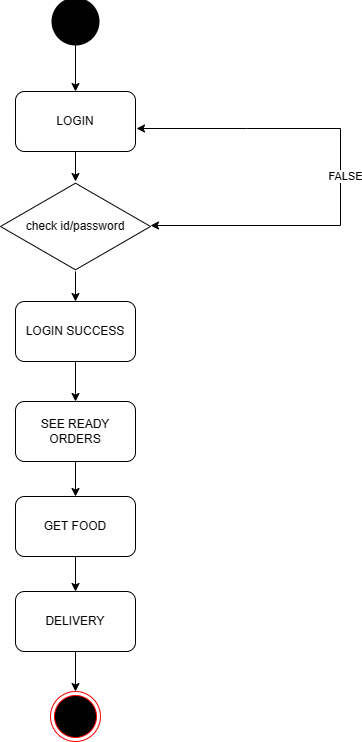
**3.2.2 Activity Diagram for User:**



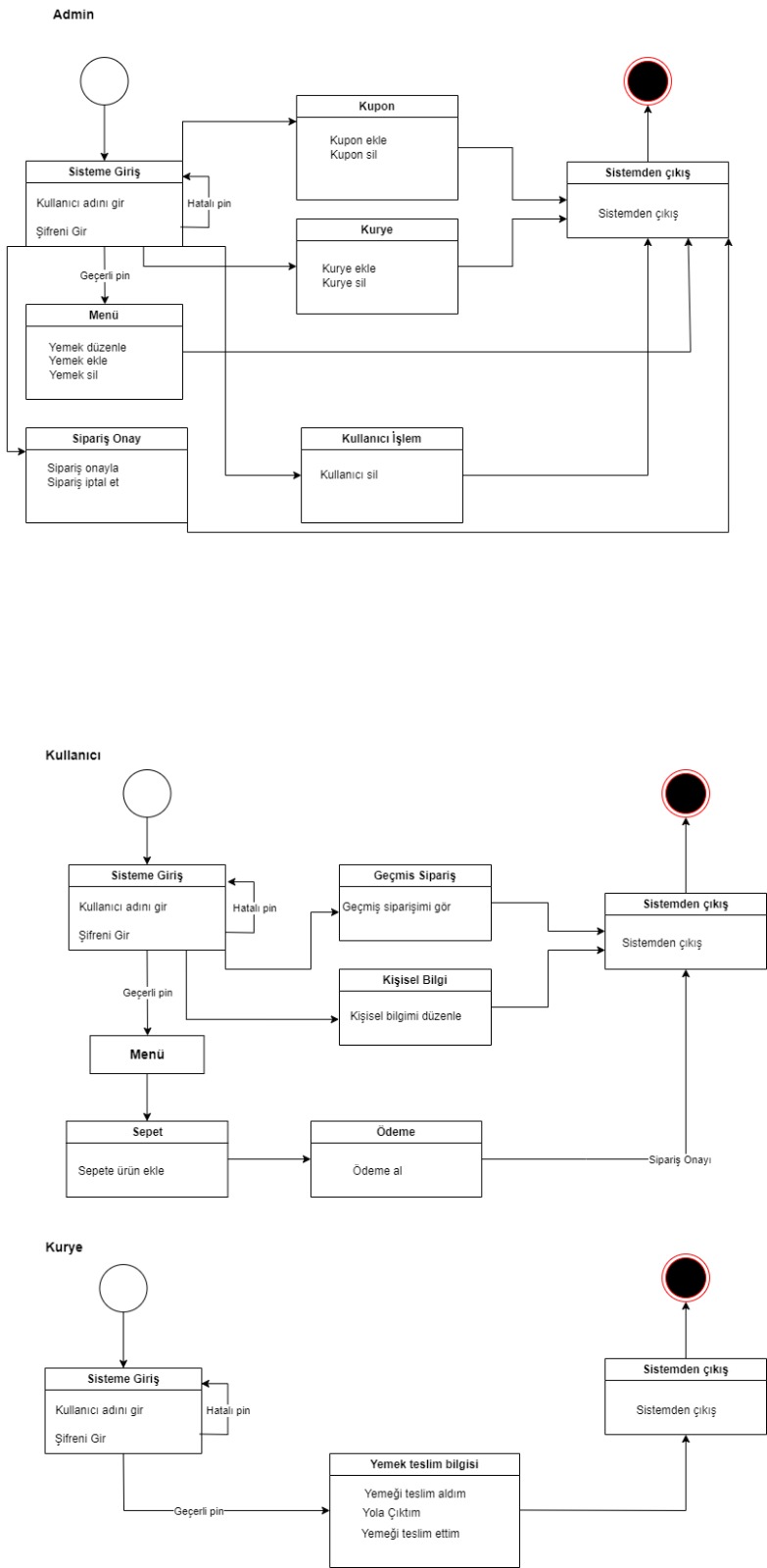
**3.2.2 Activity Diagram for Admin:**



**3.2.2 Activity Diagram for Courier:**

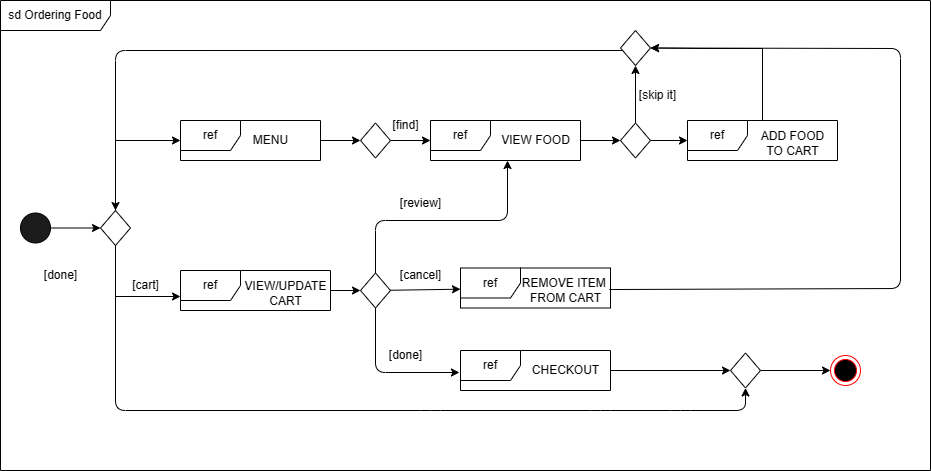


**3.2.3 State Machine Diagram**

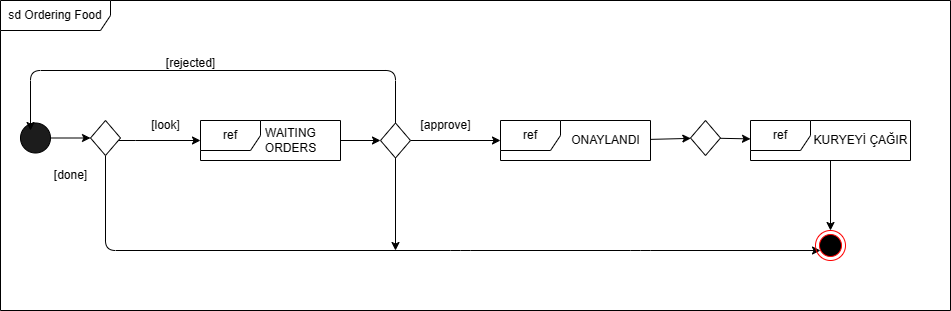


**3.3 Interaction Models**

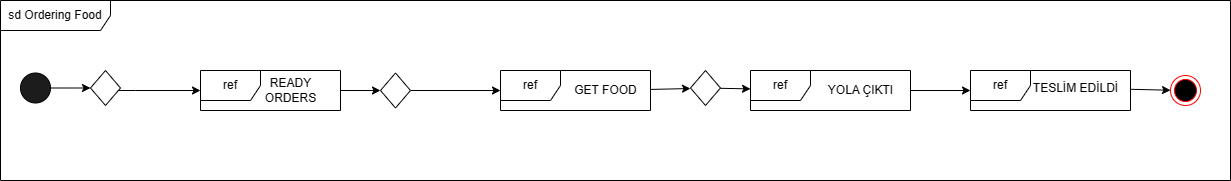
**3.3.1 Interaction Overview Diagram for User:**



**3.3.1 Interaction Overview Diagram for Admin:**

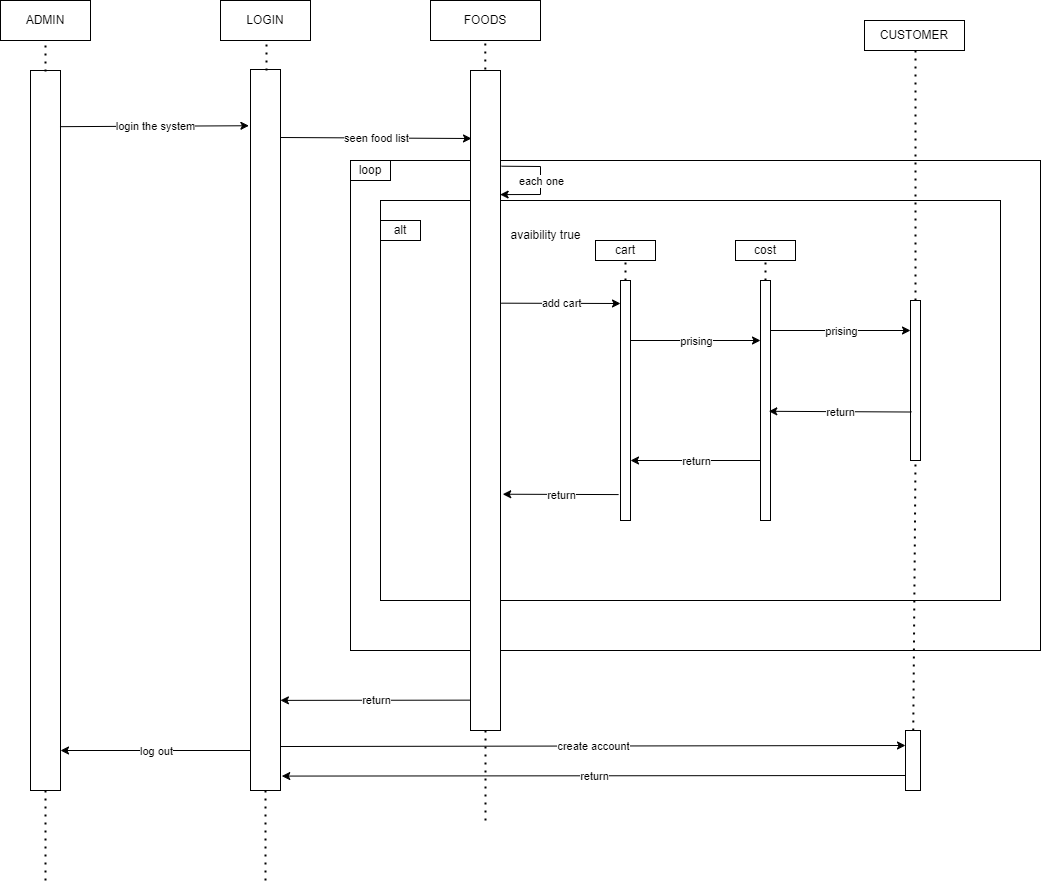


**3.3.1 Interaction Overview Diagram for Courier:**

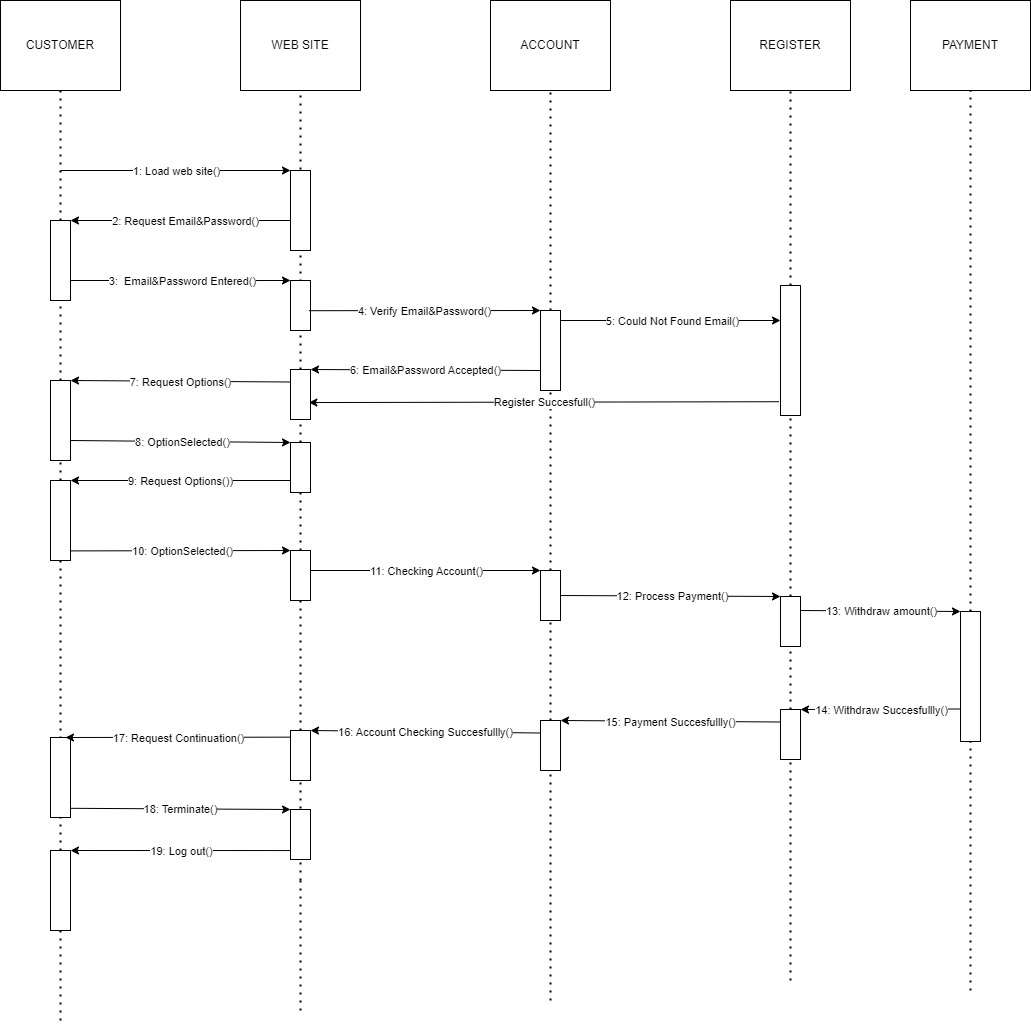


**3.3.2 Sequence Diagram**

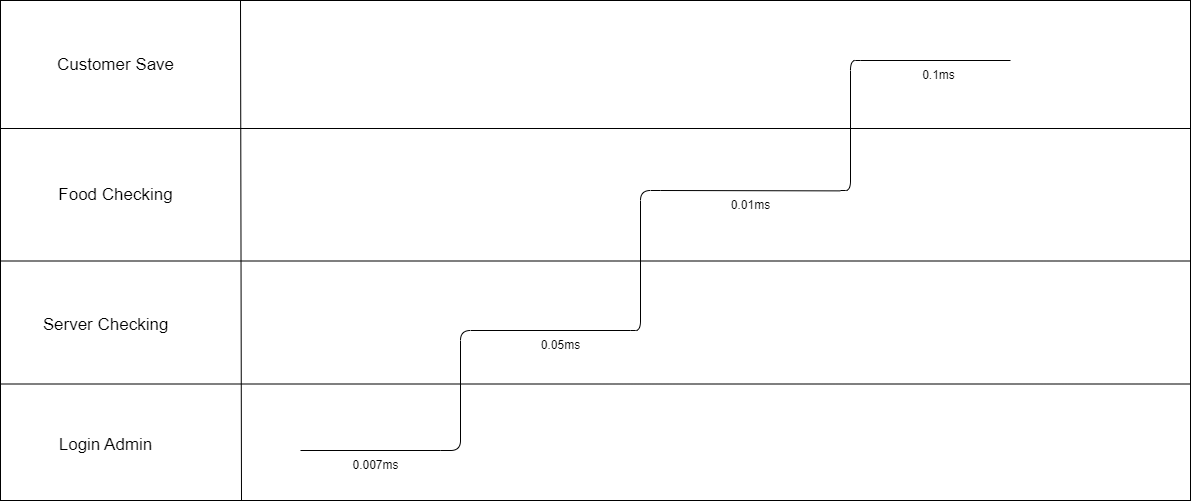
**a)Admin**



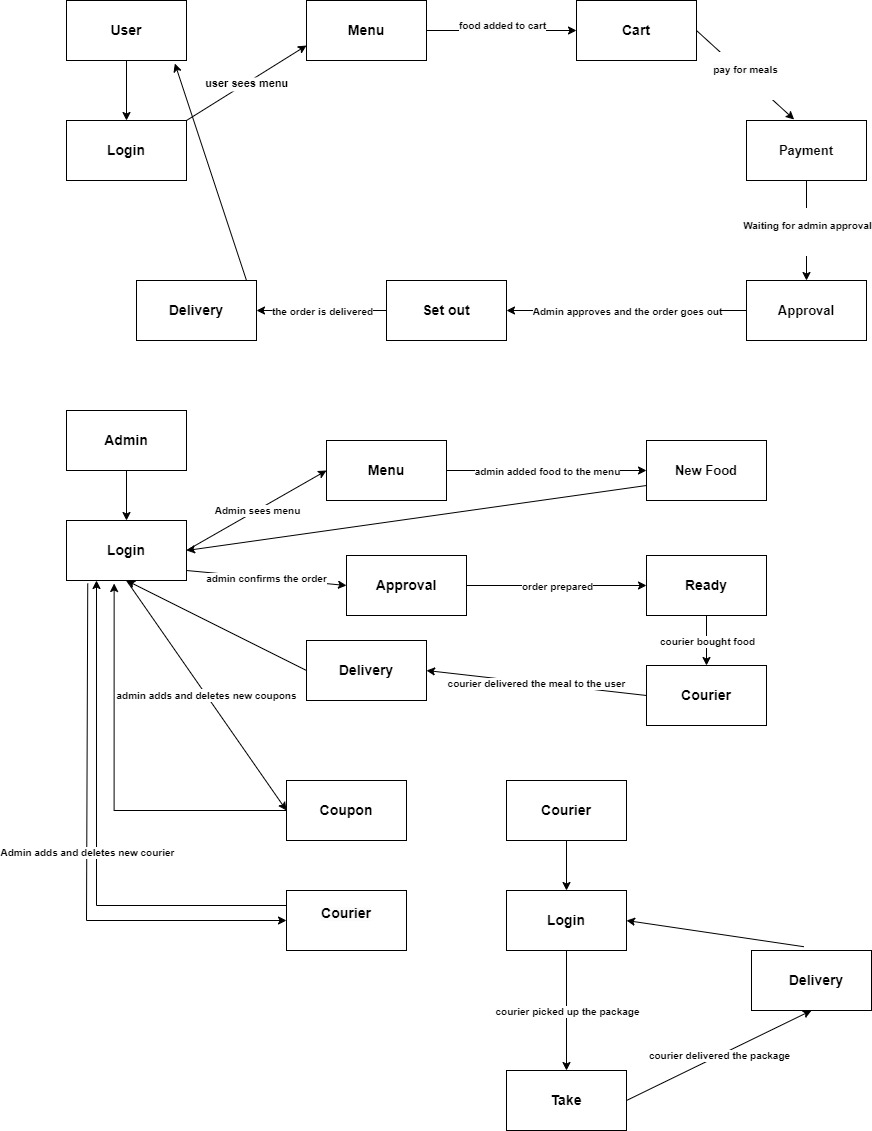
b) User



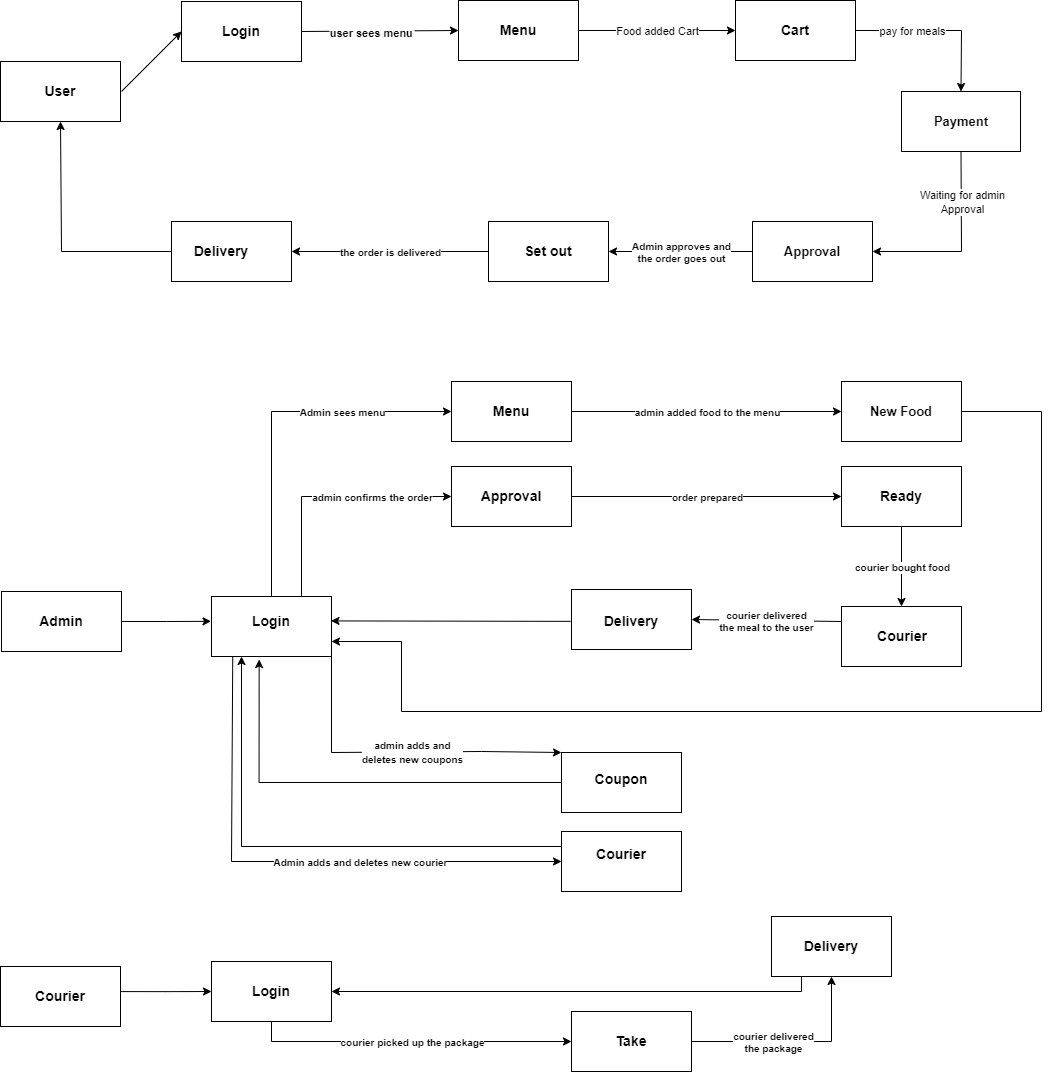
**3.3.3 Timing Diagram**



**3.3.4 Collaboration Diagram**



**3.3.5 Communication Diagram**



**4. Software Testing**

**4.1 Development Testing**

1.Register a user

2. Check if restrictions are checked when registering a user.

3-Check if correct inputs have been entered

4-If there is an error, give an error message

5-Login as a user.

6-Check if the user login is successful

7-Can the user see the menu?

8-Can the user successfully add food to the cart?

9-Can the user see the food details?

10-Can the user change the number of food in the cart?

11-Can the user apply coupons in the cart?

12-Is there a minimum amount in the cart?

13-Can the user change the address to place the order?

14-Does the quantity of the food decrease when a food is ordered?

15-Does the food that is out of stock appear on the menu?

16-Can the user see past orders?

17-Can the user see the status of the order?

18-Can the user update their account information?

19-Adding a new address

20-Can the user see the payment information?

21-Check system uptime.

22- Are there any delays?

23- Log out of the system.

24-Admin adds courier.

25-Can admin edit couriers?

26-Admin adds coupons.

27-Can admin edit coupons?

28-Can admin edit users?

29-Are coupon, courier and user transactions completed successfully in the database?

30-Can admin approve user orders?

31-Admin sends a verification code to see user information.

32-The courier announces that he is on his way and has delivered the food.

**RESULT:**

Each system part must operate reliably and correctly. The user should be able to use the application whenever he wants with the shortest delay. Basic operations should produce correct output and give an error message if there is an error. The system should be ready for any scenario.

**4.2 Unit Testing**

The user has successfully registered and logged in to the system. When the user entered an incorrect password, a warning message was given and password restrictions were made. No errors were observed. The user was able to successfully add the meals to the cart and complete the checkout step reliably. The user was able to view their past orders. The user was also able to successfully update his account information. Admin has successfully logged in to the system. When the admin entered an incorrect password, a warning message was given and password restrictions were made. New dishes can be added. Administrator can view Users and order content. The administrator was able to add courier and coupons without any problems. The courier has successfully logged into the system. When the courier entered the wrong password, a warning message was given and the password was restricted. The Courier successfully delivered the Meal. Warning messages, password restrictions and database information are working properly.

**4.2.1 White Box Testing**

|  |  |  |  |
| --- | --- | --- | --- |
| Items(AdminOperations) | Testing Method | Reasons | Expected |
| Limitations New Admin | White Box Testing | Database not accepted-Error | Error should be received, database should not be accepted |
| Limitations New Admin | Database accepted-Next | Must proceed successfully |
| Limitations Old Admin | Database not accepted-Error | Error should be received, database should not be accepted |
| Limitations Old Admin | Database accepted | Must proceed successfully |
| Save Admin | Saved | Must be saved |

|  |  |  |  |
| --- | --- | --- | --- |
| Items(UserOperations) | Testing Method | Reasons | Expected |
| Limitations New User | White Box Testing | Database not accepted-Error | Error should be received, database should not be accepted |
| Limitations New User | Database accepted-Next | Must proceed successfully |
| Limitations Old User | Database not accepted-Error | Error should be received, database should not be accepted |
| Limitations Old User | Database accepted-Next | Must proceed successfully |
| Save User | Saved | Must be saved |

|  |  |  |  |
| --- | --- | --- | --- |
| Items(CoruierOperations) | Testing Method | Reasons | Expected |
| Limitations New Coruier | White Box Testing | Database not accepted-Error | Error should be received, database should not be accepted |
| Limitations New Coruier | Database accepted-Next | Must proceed successfully |
| Limitations Old Coruier | Database not accepted-Error | Error should be received, database should not be accepted |
| Limitations Old Coruier | Database accepted-Next | Must proceed successfully |
| Save Coruier | Saved | Must be saved |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Position | Version | Date |
| Person1 | Project Manager | 1.0.0 | 25.12.2023 |
| Person2 | Software Engineer | 1.1.0 | 27.12.2023 |

**4.3 Integration Testing**

The user successfully registered in the system (User Management) and then selected an item in the menu and added it to the cart (Menu Management and Order Management).

The user can update the address registered in the system and add a new address. Additionally, the user can change personal information such as password. (User management)

The user has successfully completed the payment step (Order Management and Payment Integration).

User was able to log out (Login management)

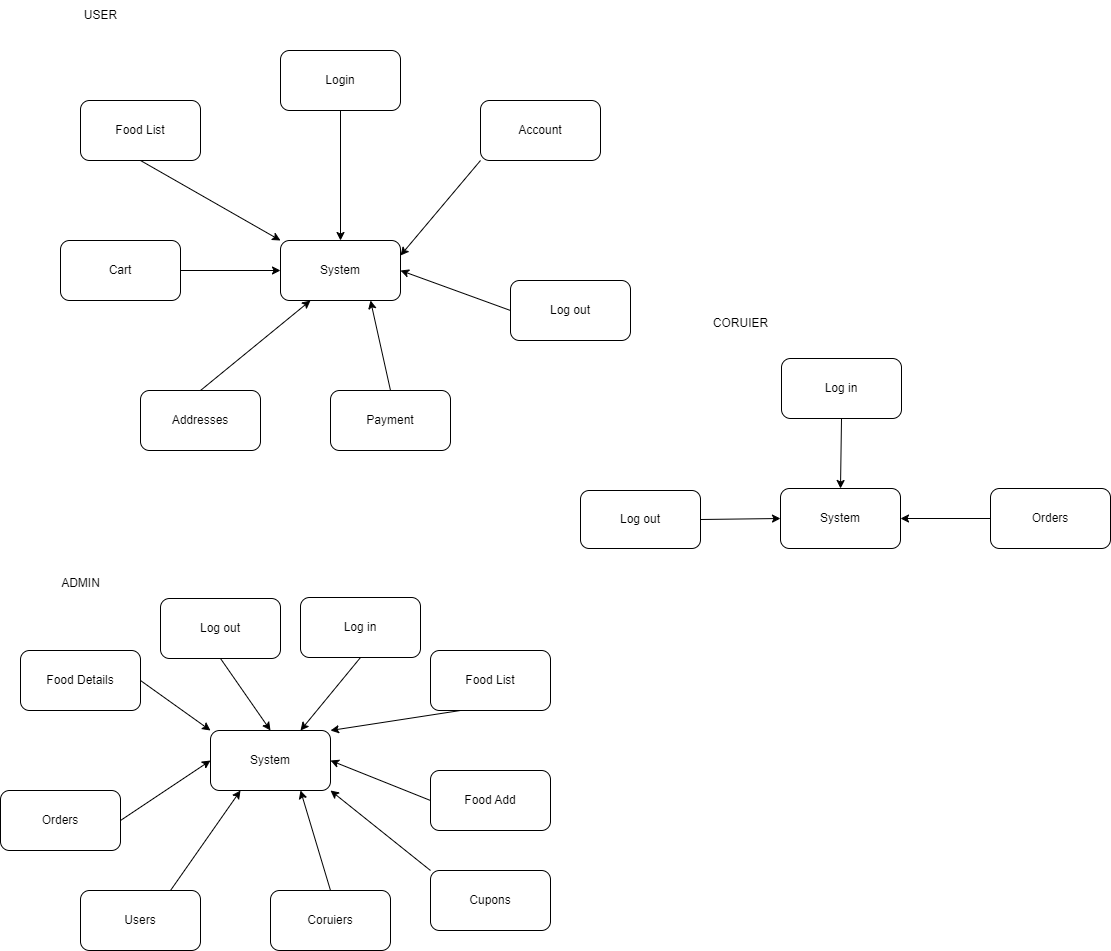
Admin has successfully logged in to the system (Admin management)

Admin can see the available meals and sold out meals. (Menu management)

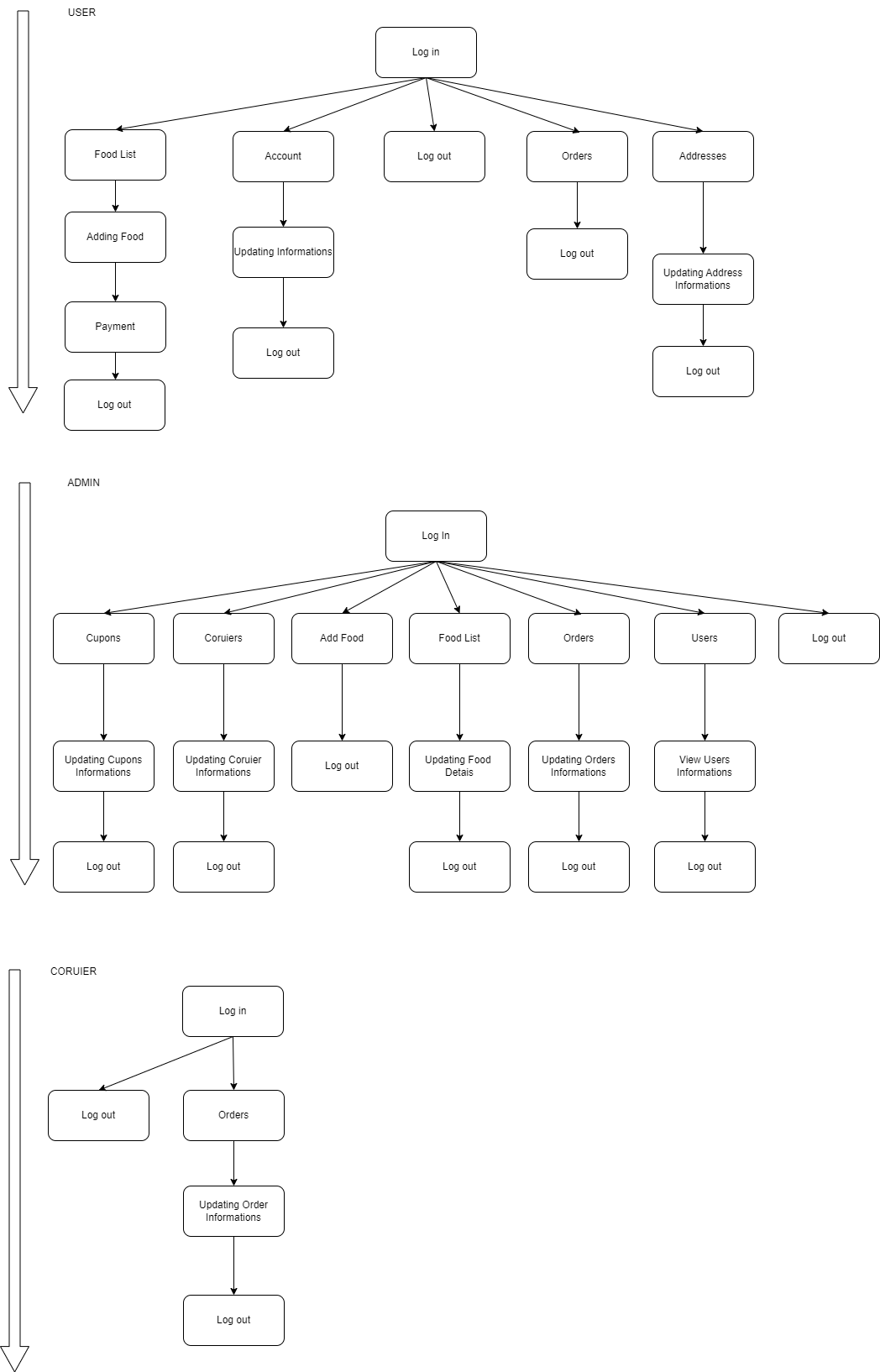
Admin was able to add new dishes, couriers and coupons. (Restaurant and menu management)

Admin was able to log out (Login management)

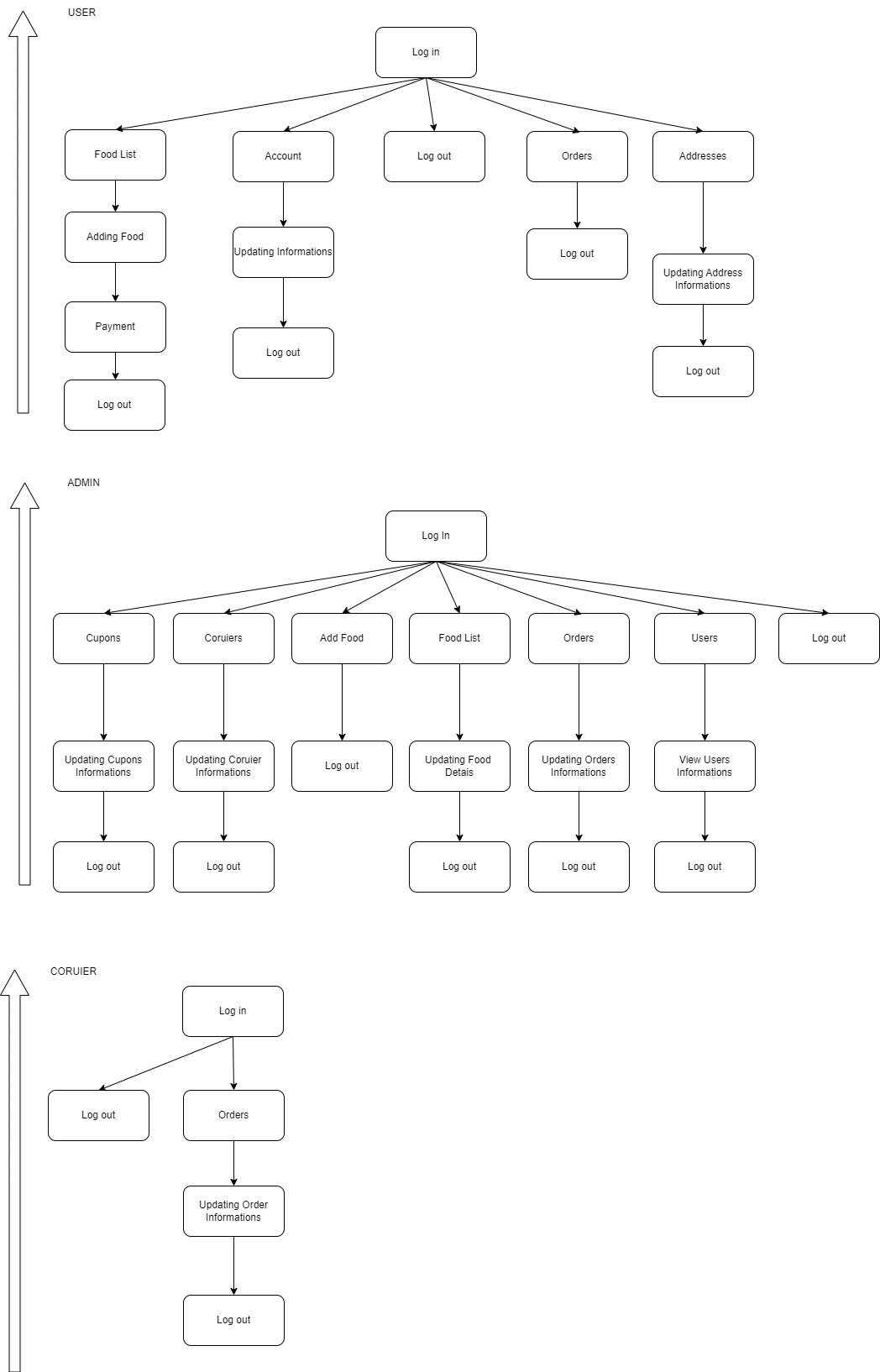
**4.3.1 Big-bang**



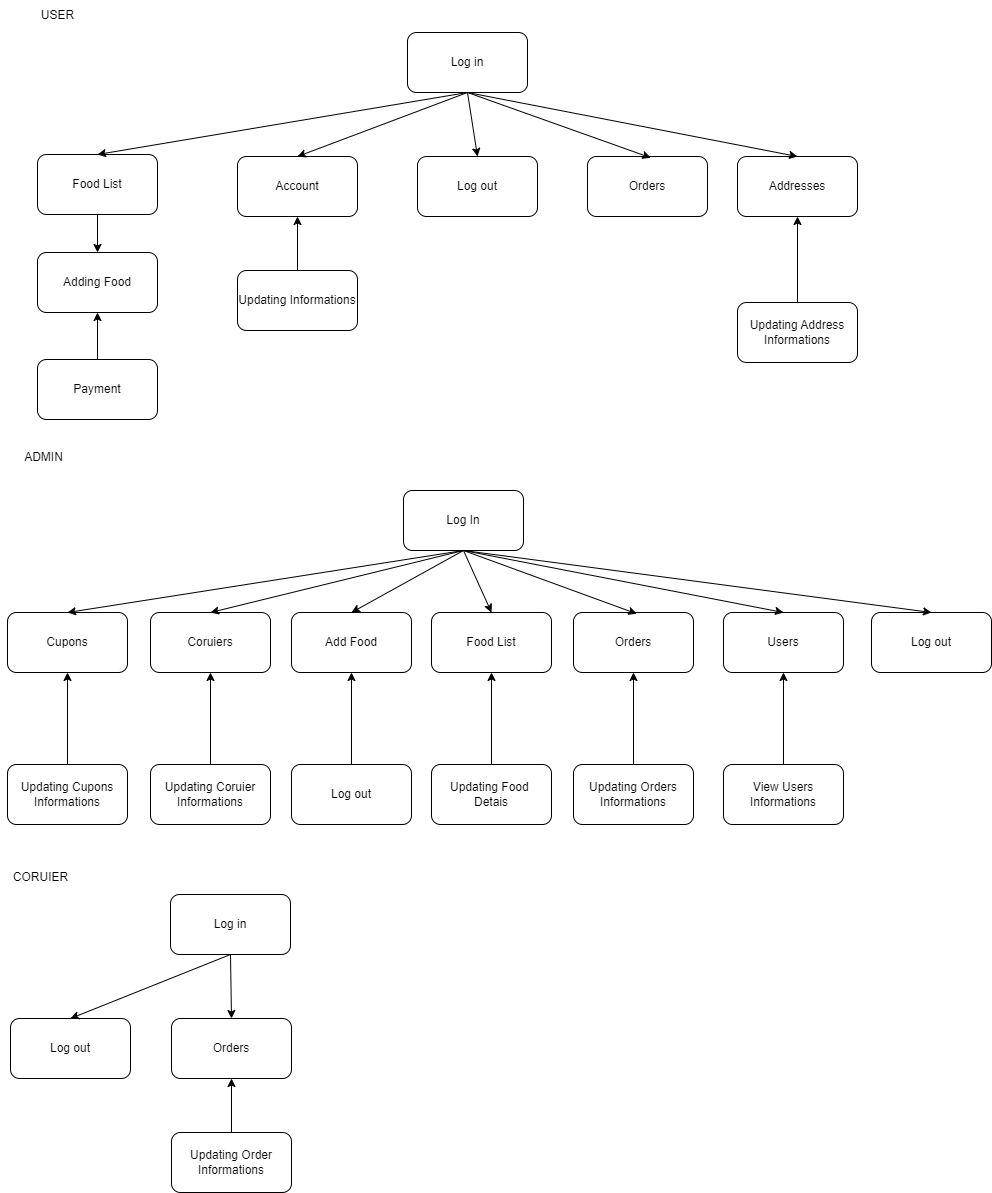
**4.3.2 Top-down**



**4.3.3 Bottom-up**



**4.3.4 Hybrid**



**4.4 System Testing**

**4.4.1 Black Box Testing**

|  |  |  |
| --- | --- | --- |
| **Items** | Testing Method | Reasons |
| Login | Black Box Testing | All users can do |
| Register | All users can do |
| Food List | All users can do |
| Cart | Customer can do |
| Customer Saved Screen | Admin can do |
| Customer Deleted | Admin can do |
| Order Screen | Customer can do |
| Account Informations | Customer can do |
| Payment | Customer can do |
| Addresses | Customer can do |
| Add Food | Admin can do |
| Delete Food | Admin can do |
| Add Cupon | Admin can do |
| Add Coruier | Admin can do |
| Views All Customers | Admin can do |
| Views Past Orders | Customer can do |
| Views All Orders | Admin can do |
| Update Food Details | Admin can do |
| Views Food Details | Admin and Customer can do |

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Position | Version | Date |
| Person2 | Software Engineer | 1.1.0 | 25.12.2023 |

**4.5 User Testing/Acceptance Testing**

**4.5.1 Alpha Testing**

This testing refers to a testing phase done by the developers of the software by the internal team or a select group of internal users. Alpha testing generally aims to detect issues such as bugs, deficiencies and performance issues of the software.

Scenario: User Login and Menu Review

1- The user logs in to the application.

2- The user selects a meal by examining the current menu.

3- The user adds the selected food to the cart.

4- The user checks the cart and proceeds to complete the order.

5- The user enters the payment information and confirms the order.

6- A confirmation screen is shown to the user to verify that the order has been received correctly.

The results of alpha testing can often relate to user feedback and bugs found.

After the alpha testing process is over;

**Errors Found:**

-Users sometimes receive errors on the payment page.

-There are delays when updating cart contents.

**Feedbacks:**

-The user interface is user-friendly, but navigation on some pages can be difficult.

-There are users who think that some dishes in the menu are missing or in the wrong category.

-Specific recommendations to improve application performance.

**Compliments:**

-Menu design and visual aesthetics were appreciated.

-The order tracking system is useful and works accurately.

--The user registration and login process is seamless.

**Solution offers:**

The error on the payment page can be resolved by adding another payment method for payment transactions.

Optimizing database queries for cart update delays may be considered.

**4.5.2 Beta Testing**

Beta testing is user testing in which the software is tested under real-world conditions by a wide range of users.

We activate the website for selected users.

**Testing for General Users:**

-General users test the application on different devices (phone, tablet, computer).

-New user registration and login, accuracy and response times of adding and removing meals, speed and accuracy of updating the cart page, accuracy of meal coupons, order process, payment process, changing and updating account information and adding new addresses.

**User Feedback:**

-The working speed of the cart is too slow:

-There is no e-mail restriction when registering a new user.

-There are some coupons that do not work.

Developers:

-Some adjustments were made to the cart-related codes and the error was corrected.

-Email restriction added for new users.

-A more stable coupon addition page was implemented on the admin side.

**4.5.3 User Acceptance Testing**

After alpha and beta testing, this is the stage where end users decide whether the application will be published or not.

Parts checked for the release phase:

-Easily sign up and log in to the website.

-Not accepting false information.

-Non-members can also view the menu.

-Members can easily add the food to the cart and proceed to the payment screen.

-Easy tracking of the order.

-Viewing old orders.

-Changing account information easily and securely.

**4.6 Debugging**

**4.6.1 Debugging by Brute Force**

While trying to make changes to the food detail page, the problem of not being able to display the food details arose. While debugging, we realized that the error was not in the controller because the FoodDeatils(arg1) controller was not working at all. We typed console.log("error") in various parts of the JavaScript and JSON file and checked which block it was in. After finding the faulty part, we tried to solve the problem with various pieces of code. While doing the last debug, we realized that the data we sent was coming to the controller and the error was solved.

**4.6.2 Debugging by Induction**

**Error Status:**

Users order from the restaurant and frequently encounter failures during the payment process.

Debugging with Induction Steps:

**Step 1: Log Analysis**

By examining the application's log rules, we determined under what conditions this error would occur.

For example, we found errors related to log transactions such as "payment\_failed" or similar.

**Step 2: Examining Similar Errors**

We reviewed logs containing similar payment errors on previous dates.

We tried to determine under what conditions such errors occur and how they are resolved. For example, when the credit card information was incorrect or empty, the payment system would crash when it should have given an "input is empty" warning.

**Step 3: Derive a General Rule**

We noticed that similar errors are based on a general rule. For example, these errors may occur more frequently when the user enters card information incorrectly or does not enter it at all.

**Step 4: Solving the Error**

Based on the general rules we obtained, we made an improvement to solve these errors. For example, a warning message appears when the input is empty or the CVC is more than 3 digits or missing.

**Step 5: Testing and Monitoring**

We tested the fix before applying it and confirmed that similar errors were reduced and the improvements were effective.

**4.6.3 Debugging by Deduction**

**Error Status:**

Users have reported about a specific feature in the app not working properly on a particular device or browser.

Debugging by Deduction Steps:

**Step 1: First Observation**

By reviewing user feedback, we observed that this issue occurs on older browsers or devices.

**Step 2: Inferring the Causes of the Error**

Together with the developers and the testing team, we tried to uncover the causes of the problem. For example:

Possible Cause 1: JavaScript incompatibility in a particular browser.

Possible Reason 2: Incompatibility with the screen resolution of the device used, specifically.

Possible Cause 3: A problem with the browser cache.

**Step 3: Testing Inferences**

We tested the possible causes we deduced. We checked if the error was reproducible by using the features on the problematic browser or device.

**Step 4: Finding the Exact Cause**

Based on the data obtained from the test results, we determined which cause is causing the problem. We found that JavaScript incompatibility in a particular browser is causing the issue.

**Step 5: Solving the Error**

We made appropriate corrections to address the cause we identified. We've made adjustments to the JavaScript code to make it compatible with a particular browser or screen resolution.

**Step 6: Monitoring and Verification**

We monitored performance before and after applying our fixes. We followed user feedback and confirmed that the issue has been resolved.

**4.6.4 Debugging by Backtracking**

**Error Status:**

After users order food from the restaurant, the order status is not updated correctly in the app.

Debugging by Backtracking Steps:

**Step 1: Identifying the Error**

By reviewing user feedback and application logs, we found that there were some errors in order status updates. For example, we retrieved relevant information by searching for an error report such as "order\_status\_not updating" or similar.

**Step 2: First Observation**

We tried to understand what steps users followed to determine when the error occurred. For example, do users experience problems immediately after ordering or during delivery?

**Step 3: Backtracking the Error**

We followed the steps back from the point where the error occurred. For example:

What steps does the system follow when the user completes the order?

What modules or services interact when the order status is updated?

**Step 4: Identifying Potential Points with Backtracking**

During the traceback, we determined that the database queries were problematic.

**Step 5: Detailed Review**

After identifying the problem spot, we took a more detailed look at it. We detected that a database query was incorrect.

**Step 6: Solving the Error**

We made the appropriate corrections to fix the problem spot. We fixed the incorrect database query.

**Step 7: Monitoring and Verification**

We monitored performance before and after applying our fixes. By following user feedback, we confirmed that the order status update issue has been resolved.

**5. Identification of Project Risks and Project Risk List**

**5.1 Identification of Project Risks**

* There may be a mismatch in design expectations between the client and the project team.
* Cost
* Errors and late delivery in delivery time.
* Technical difficulties may arise that may cause website performance issues.
* Security of customer information, payment information and other sensitive data
* There may be difficulties in project management issues such as on-time delivery and budget control.
* Keeping restaurant menus, photos, and other content organized and up-to-date can be challenging.
* Problems may occur with different screen sizes and resolutions.

**5.2 Project Risk List**

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk Factor** | **Risks** | **Reasons** | **Measures** |
| **Mission And Vision** | Wrong information | Trouble in communication | continuous communication |
| **Customer** | Not knowing what you want | Errors in redirects | continuous communication |
| **Design** | Customer does not like the design | Trouble in communication | continuous communication |
| **Financial And Economic** | Cost | wrong cost calculation | Calculation |
| **Payment security** | Security breaches or leaks | Not PCI DSS compliant | during payment |
| **Scope** | Lack of time | error in project plan selection | Calculation |
| **TechnologicalCompatibilityIssues** | Visual and systems problems on different devices | Compatibility issues with older browsers | when entering the website |
| **Technical** | Codding errors | Complex code | Simple code using |
| **Employee** | Mismacth | not doing enough research | Interview |

**5.3 Planning Risks and Contingencies**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Risks** | **Risks Likelihood of Realization** | **Cost Effect** | **Risk Rating by Cost** | **Effect on Duration** | **Risk Rating by Duration** |
| **Payment security** | Low | Low | Mid | - | High |
| **Technological Compatibility Issues** | Low | Low | Mid | - | High |
| **Wrong information** | High | High | High | increases | High |
| **Not knowing what you want** | High | High | High | increases | High |
| **Cost** | Low | Mid | High | - | Low |
| **Lack of time** | High | Mid | Mid | increases | High |
| **Codding errors** | Low | High | High | increases | High |
| **Mismacth** | Low | High | High | increases | High |
| **communication problem** | High | High | High | increases | High |