Bilkent University
Department of Computer Engineering

Senior Design Project

T2432

Yes Chef!

Analysis and Requirement Report

22003479, İsmail Barış Sunar, baris.sunar@ug.bilkent.edu.tr

22102932, Mert Emre Yamalı, emre.yamali@ug.bilkent.edu.tr

22101645, Mert Terkuran, mert.terkuran@ug.bilkent.edu.tr

22001880, Serhan Turan, serhan.turan@ug.bilkent.edu.tr

22101766, Ulaş Keskin, ulas.keskin@ug.bilkent.edu.tr

Supervisor: Uğur Doğrusöz

Course Instructors: Mert Bıçakçı, Atakan Erdem

16-12-2024

This report is submitted to the Department of Computer Engineering of Bilkent University in partial fulfillment of the requirements of the Senior Design Project course CS491/2.

Table Of Contents

1. Introduction

Planning and cooking meals at home, managing ingredients, and successfully following a recipe can all often be very tedious and overwhelming, especially for those who must fit these aspects into their already busy schedules. Although these experiences attached to maintaining a healthy nutritional aspect in life seem to be a relatively global problem, a proper and maintainable solution has yet to be created, and applicable modern approaches such as the use of gamification to enhance these experiences remain underutilized. Our project's purpose is to revolutionize home cooking with "Yes Chef!", a gamified cooking app explicitly designed to make all processes attached to cooking less daunting and more efficient. The app will be able to maintain an extensive database filled with recipes for all people, handle these recipes to create personalized suggestions based on preferences, gamify the entire process of following the recipe to make it easier, and overall enhance the entire process of cooking. With features like ingredient management, interactive cooking, recipe sharing, and subscriptions, "Yes Chef!" will turn home cooking into a fresh and enjoyable experience.

2. Current System

This section is intentionally left empty as Yes Chef! does not build upon or directly compare to any existing system.

3. Proposed System

3.1. Overview

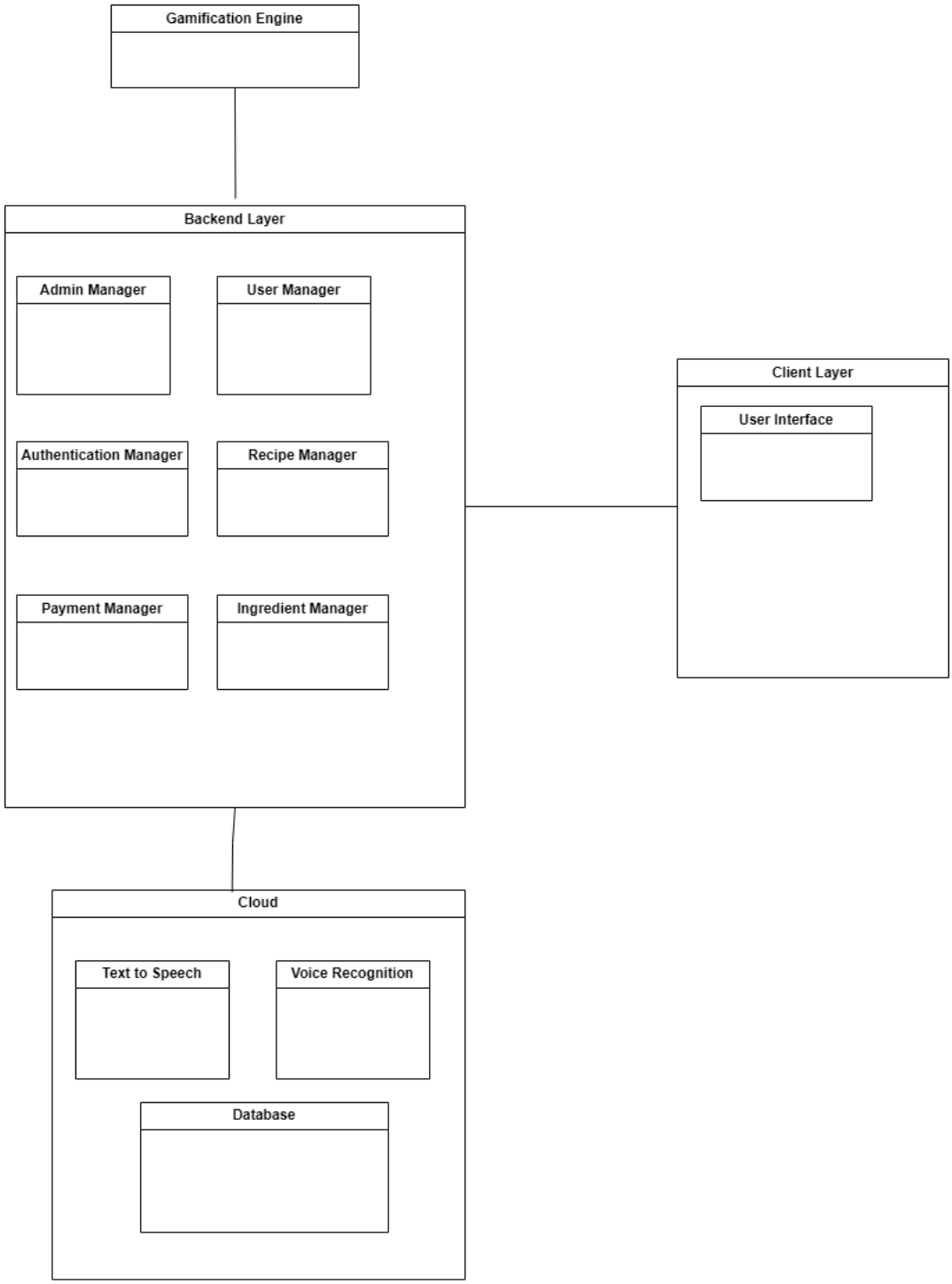


Fig. 1: Proposed System Architecture

3.1.1. Client Layer

This layer is responsible for presenting the information and interactions in a visually appealing way to the user. It serves as the primary way for users to access and interact with the core features of the application. This layer

handles rendering interfaces, managing input, and ensuring efficient communication between the user and the backend.

3.1.1.1. User Interface

The user interface is the sublayer of the client layer that is dedicated to handling visual representation and interaction between the user and the backend. This layer ensures that the content of the application is displayed in a visually appealing and accessible way.

3.1.2. Backend Layer

3.1.2.1. Admin Manager

This service provides developers the ability to modify the database and check the errors to improve the project.

3.1.2.2. User Manager

It handles profile settings such as changing name, profile photos, etc. This service handles the user's relationship with other features.

3.1.2.3. Authentication Manager

It handles functionalities such as registering, log in/out, email confirmation, and password changes.

3.1.2.4. Recipe Manager

The recipe manager handles functionalities regarding recipes like adding, removing, and editing recipe details.

3.1.2.5. Payment Manager

The payment manager controls any sort of in-app purchases, making sure safety protocols are followed, and the right amount of transactions are made for both ends.

3.1.2.6. Ingredient Manager

Ingredient Manager handles ingredient variables operations such as quantities, correct units, attributes like freshness, allergens, nutrition values, and costs.

3.1.3. Cloud

3.1.3.1. Text To Speech

This service handles the voice assistant part. It reads what to do in this step of the recipe. We are using Google Cloud's service.

3.1.3.2. Voice Recognition

This service handles the voice input feature. It iterates to the next step when it hears "Yes Chef!". This way, users can interact with the application by speaking. We are using Google Cloud's service.

3.1.3.3. Database

This part contains the whole system database. All users, recipes, ingredients, etc. It is uploaded to the cloud.

3.1.4. Gamification Engine

This engine allows administrators and users to create gamified recipes and store them in the backend. It contains the tools necessary to upload multimedia like audio, video snippets, and events such as timers to the recipe presentation. After creation, the multimedia and event hierarchy are stored inside the database inside the backend layer.

3.2. Functional Requirements

3.2.1. User Profile Management

- Users must be able to sign up using email, or phone number.
- Users must be able to change their passwords.
- Users must be able to create and edit their profiles (username etc.).
- Users must be able to specify dietary restrictions such as vegan or any allergens.
- Users must be able to view and manage (delete, etc.) their uploaded recipes.
- Users must be able to view and modify their profile visibility.

3.2.2. Recipe Creation and Shared Content

- Users must be able to save their recipes in both text and gamified recipe presentation format.
- Users must be able to add multimedia to their recipes.
- Users must be able to share the recipes within the app.

3.2.3. Social Interaction

- Users must be able to follow other users and see their recipes.
- Users must be able to rate recipes by other users.
- Users must be able to see the ratings of other users on recipes.

3.2.4. Ingredient Management

- Users must be able to input their available ingredients into the application.
- Users must be able to see all ingredients and their missing ingredients for any specific recipe.
- Users must be able to see any substitute ingredient in a recipe if any exists.

3.2.5. Price Information

- Users must be able to see a reasonable average cost of any ingredient from specific stores.

3.2.6. Notifications

- Users must be able to be alerted for expiring ingredients or ingredients that the user is running low on.
- Users must be able to be notified of ratings and reviews on their recipes.
- Users must be able to modify which notifications they want to receive.

3.2.7. Dietary and Nutritional Information

- Users must be able to see the calorie count and nutritional breakdown of gamified recipes.
- Users must be able to see allergen alerts specific to them based on user settings.

3.2.8. Settings and Feedback

- Users must be able to switch between imperial and metric units.
- Users must be able to submit errors or new feature requests.

3.2.9. Monetization Features

- Users must be able to sell their gamified recipes individually or in packs.
- Users must be able to subscribe to premium recipe packs.

3.2.10. Gamification and Organization Features

- Users must be able to create weekly meal plans using the recipes on the application.
- Users must be able to use voice recognition while using the gamified recipes.

3.3. Nonfunctional Requirements

3.3.1. Usability

- The user experience will be compact and easy to use. Users should understand at first glance how to achieve their needs in the application. Universal symbols will be used in UI to achieve simplicity in user experience.
- The user should be able to reach the desired pages easily.

3.3.2. Reliability

- The application should handle failures without any data loss.
- The app should perform without failure in 99 percent of use cases during a month.

3.3.3. Performance

- Photos should be uploaded at most 2 seconds in 95 percent of events.
- Speech recognition and answering back to the user should be less than 5 seconds.

3.3.4. Supportability

- The project's design should be made so that it should be easy to add new features to the system later. New features will be added without changing fundamental parts of the code base.
- The system should be able to recognize errors and what caused them.
- The same codebase should work for all systems operating under the same OS.

3.3.5. Security

- ISO/IEC 27001 standards will be followed.

- The sensitive data of users will be encrypted.
- User data will not be shared with third-party applications.

3.4. Pseudo Requirements

This section is intentionally left empty as we do not incorporate any predetermined or implied requirements that fall outside the formal scope of this project.

3.5. System Models

3.5.1. Scenarios

3.5.1.1. Login

Actors: User

Entry conditions:

➔ The user has an existing account.

➔ The user launches the application or navigates to the login page.

Event flow:

1. The user enters a username and password into the login form.
2. The user clicks the "Login" button.
3. The system verifies the credentials.
4. If the credentials are correct, the user is granted access to the application's main page.

Alternate event flows:

2.1 If the username or password fields are left empty, the system prompts the user to fill in the missing fields.

3.1 If the credentials are incorrect, the system displays an error message and allows the user to try again.

3.2 If multiple failed attempts occur, the system may ask the user to complete additional verification or prompt the user to reset the password.

Exit conditions:

➔ The user is successfully logged into the application.

➔ The user cancels the login attempt.

3.5.1.2. Registration

Actors: User

Entry conditions:

- ➔ The user navigates to the registration page or selects an option to sign up for a new account.

Event flow:

1. The user enters required information into the registration form.
2. The user reviews terms of service and privacy policy and checks the box to accept them if required.
3. The user clicks the "Register" or "Sign Up" button.
4. The system validates the provided information.
5. If the information is valid, the system creates a new user account.
6. The system may send a verification email to the user's provided email address.
7. The user, after receiving the verification email, clicks on the verification link.
8. The system confirms account verification and grants full access to the user's new account.

Alternate event flows:

4.1 If the email is already in use, the system displays a message requesting the user to choose another email or log in if they own it.

4.2 If the password does not meet complexity requirements, the system displays a message instructing the user on password criteria.

4.3 If the user declines terms of service, the registration process cannot proceed.

4.4 If the user's verification email link expires or is not received, the user can request a new verification email.

Exit conditions:

- ➔ The user successfully creates and verifies a new account.
- ➔ The user cancels the registration process by navigating away or clicking a "Cancel" button.

3.5.1.3. Creating a recipe

Actors: User

Entry conditions:

- ➔ The user should be logged in.
- ➔ The user should enter the recipe menu.

Event flow:

5. The user selects the number of steps for the recipe.
6. The user selects photos of his food that will be used in the recipe.
7. The user assigns photos to the corresponding steps.
8. The user enters text at each step.
9. The user set timers to each step.

Alternate event flows:

1.1 Number of steps is above the selected limit. The app asks the user to choose a smaller number.

2.1 If photos are not in compatible type, users will be asked to select new photos.

Exit conditions:

- ➔ The user cancels the recipe reaction process and exits.
- ➔ The user successfully creates the recipe.

3.5.1.4. Viewing a recipe

Actors: User

Entry conditions:

- ➔ The user should be logged in.
- ➔ The user selects a recipe that is shared.

Event flow:

1. The user clicks start button
2. First step is shown to the user and the user is expected to say “Yes Chef!” to proceed.
3. When a steps’ timer is finished the next step is displayed.
4. After each step is shown, step 2 is repeated.
5. When the user completes the last step, the recipe is completed.

Alternate event flows:

2.1 If user doesn’t say “Yes Chef” in a certain time, app gives a warning to user

3.1 Before the timer finishes, the user can stop the timer and look at the recipe more detailed.

Exit conditions:

- ➔ The user completes the recipe.
- ➔ The user can cancel viewing a recipe, click the exit button.

3.5.1.5. Giving upvote to recipe

Actors: User

Entry conditions:

- ➔ The user should be logged in.
- ➔ The user should select a recipe to vote.

Event flow:

1. The user clicks an upvote button.
2. The upvote count increases by one.

Alternate event flows:

2.1 The user can undo upvote.

Exit conditions:

- ➔ The user should click the upvote button.

3.5.1.6. Write a blog

Actors: User

Entry conditions:

- ➔ The user should be logged in.
- ➔ The user should click the create blog button.

Event flow:

1. The user writes a heading for the blog.
2. The user writes the blog's content.
3. The user clicks the publish button.

Alternate event flows:

1.1 Heading might exceed a specified length. The user will be asked to write shorter heading.

2.1 Content of the blog can exceed a certain size. The user will be asked to write shorter text.

Exit conditions:

- ➔ The blogpost is published.
- ➔ Back button is clicked, operation is cancelled.

3.5.1.7. View a blog

Actors: User

Entry conditions:

- ➔ The user should be logged in.
- ➔ The user should select a blog to read.

Event flow:

1. The user clicks the blog that he wants to read.
2. App shows blogs to the user.
3. The user can read the blog.

Alternate event flows:

2.1 If an error happens in accessing the blogpost, the system redirects user to the search blogs page.

Exit conditions:

- ➔ The user closes the blog.

3.5.1.8. Creating a recipe (Admin specialized with more options)

Actors: Admin

Entry conditions:

- ➔ The admin opens the admin page.

Event flow:

1. The admin should select a request from users to create a specialized recipe.
2. The admin creates nodes for each step in the recipe.
3. The admin puts photos to desired nodes.
4. The admin writes text to each node.

5. The admin sets timers to the nodes.
6. The admin sends the recipe to the user.

Alternate event flows:

1.1 Admin can reject a request from a user if it is not compatible with the application's rules.

Exit conditions:

➔ The admin cancels the creating recipe and exits without finishing the recipe.

➔ The admin successfully creates the recipe and sends it to the user.

3.5.1.9. Creating a subscription plan

Actors: User

Entry conditions:

➔ The user should be logged in.

➔ The user should enter the subscriptions page.

Event flow:

1. The user selects recipes to include their subscription plan.
2. The user selects a price for the plan.
3. The user names the subscription plan.
4. The user clicks the publish button.
5. The user can share subscription plans on social media platforms.

Alternate event flows:

2.1 Price of the plan should be between selected intervals. The user will be asked to enter value in interval

Exit conditions:

➔ The subscription plan is created.

3.5.1.10. Joining a subscription plan

Actors: User

Entry conditions:

➔ The user should be logged in.

➔ The user should select a subscription plan to join.

Event flow:

1. The user clicks plan to join.
2. The user selects a payment method.
3. The user enters credit card information.
4. Purchase is done, and the user joins the plan.
5. The user can see recipes included in the plan.

Alternate event flows:

4.1 Payment couldn't be made, and the user will be asked to provide another payment information.

Exit conditions:

➔ The user successfully joins the subscription plan.

➔ The user cancels the purchase.

3.5.1.11. Search Recipes

Actors: User

Entry conditions:

➔ The user should be logged in.

Event flow:

1. The user clicks the search bar.
2. The user writes the name of the recipe he wants to see.
3. App shows matching recipes with the text the user provided.

Alternate event flows:

3.1 If the app couldn't find any matching recipes, some warning will be shown to the user.

Exit conditions:

➔ The user finds the recipe he wants.

3.5.1.12. Search recipe categories

Actors: User

Entry conditions:

➔ The user is logged in.

➔ The user navigates to the recipe categories section or search interface.

Event flow:

1. The user enters a search term or selects a category filter

2. The system displays a list of matching categories or recipes under the chosen category.
3. The user scrolls through the results and selects a category to explore.
4. The system shows subcategories or associated recipes.

Alternate event flows:

1.1 If the search term does not produce results, the system suggests similar categories or prompts the user to alter the search criteria.

Exit conditions:

- ➔ The user clicks away from the categories page.
- ➔ The user selects a category and proceeds to view or prepare a recipe from it.

3.5.1.13. Share recipes

Actors: User

Entry conditions:

- ➔ The user is logged in.
- ➔ The user has at least one complete recipe saved in their profile.

Event flow:

1. The user selects a recipe from their saved recipes.
2. The user clicks the “Share” button.
3. The system provides sharing options
4. The user chooses the desired sharing method and confirms.
5. The system sends the shared link or content to the chosen destination.

Alternate event flows:

3.1 If certain platforms are not supported, the system will inform the user and suggest alternative sharing methods.

4.1 If the user’s friend is not found in the system, the system prompts the user to enter an email or another contact method.

Exit conditions:

- ➔ The user completes sharing the recipe.
- ➔ The user cancels the sharing process and returns to the previous screen.

3.5.1.14. Meal planning

Actors: User

Entry conditions:

- ➔ The user is logged in.
- ➔ The user accesses the meal planning section from the main menu.

Event flow:

1. The user chooses a time frame for the meal plan
2. The user selects meals for each day from their saved recipes, recommended recipes, or by searching categories.
3. The system organizes the selected recipes into a calendar view.
4. The user reviews and confirms the meal plan.

Alternate event flows:

2.1 If the user does not have saved recipes, the system suggests popular or trending recipes to add to the plan.

3.1 If the user wants to modify the order or remove a meal, they can do so before finalizing.

Exit conditions:

➔ The user confirms and saves the meal plan.

➔ The user cancels meal planning and returns to the main menu.

3.5.1.15. Trending recipes

Actors: User

Entry conditions:

➔ The user is logged in.

➔ The user navigates to the trending recipes section.

Event flow:

1. The system displays a list of currently trending recipes, often curated by popularity, ratings, or recent interest.
2. The user browses through the trending list.
3. The user selects a trending recipe to view details.
4. The user can choose to save the recipe, begin cooking instructions, or share it.

Alternate event flows:

2.1 If there are no trending recipes available (e.g., data fetch error or no recent activity), the system suggests popular categories or recommended recipes.

3.1 If the user has dietary restrictions, the system can filter the trending recipes accordingly.

Exit conditions:

➔ The user exits the trending section and returns to the main menu.

➔ The user selects a recipe and proceeds with viewing or preparing it.

3.5.2. Use Case Model

The scenarios above are modeled in the use case diagram.

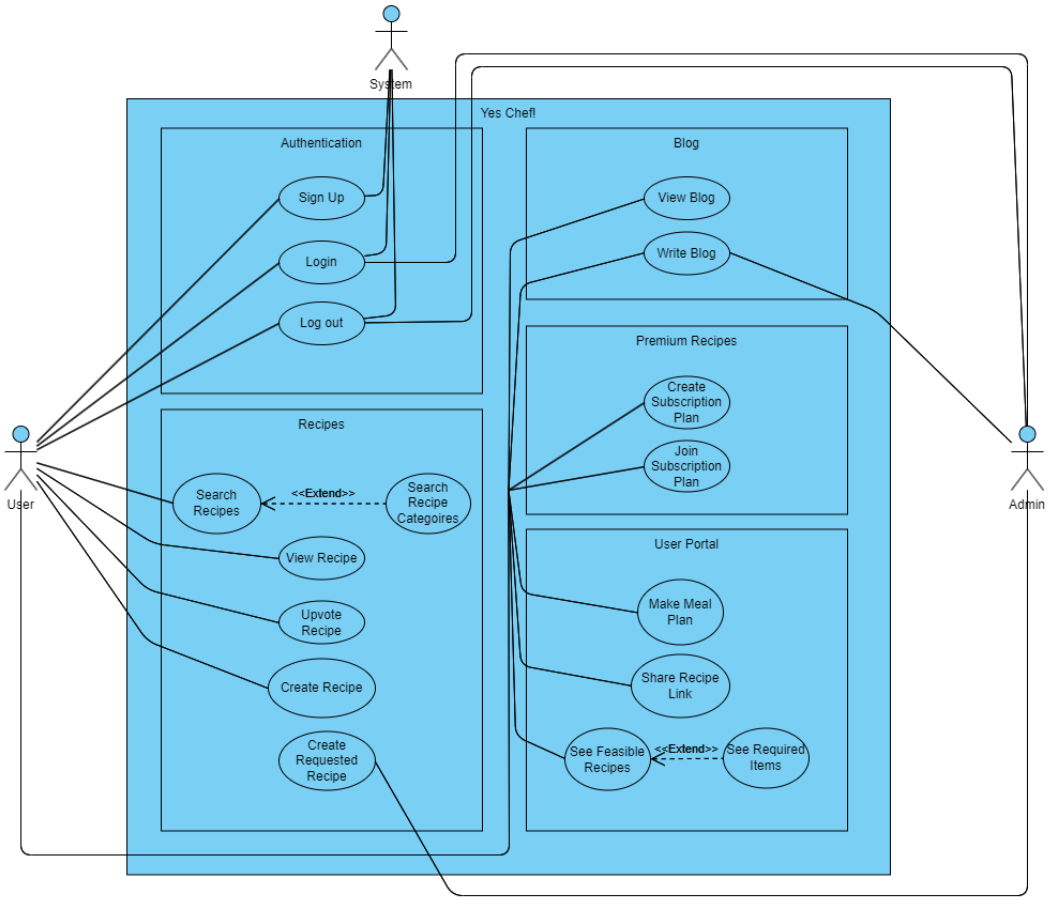


Fig. 2: Use Case Diagram

3.5.3. Object and Class Model

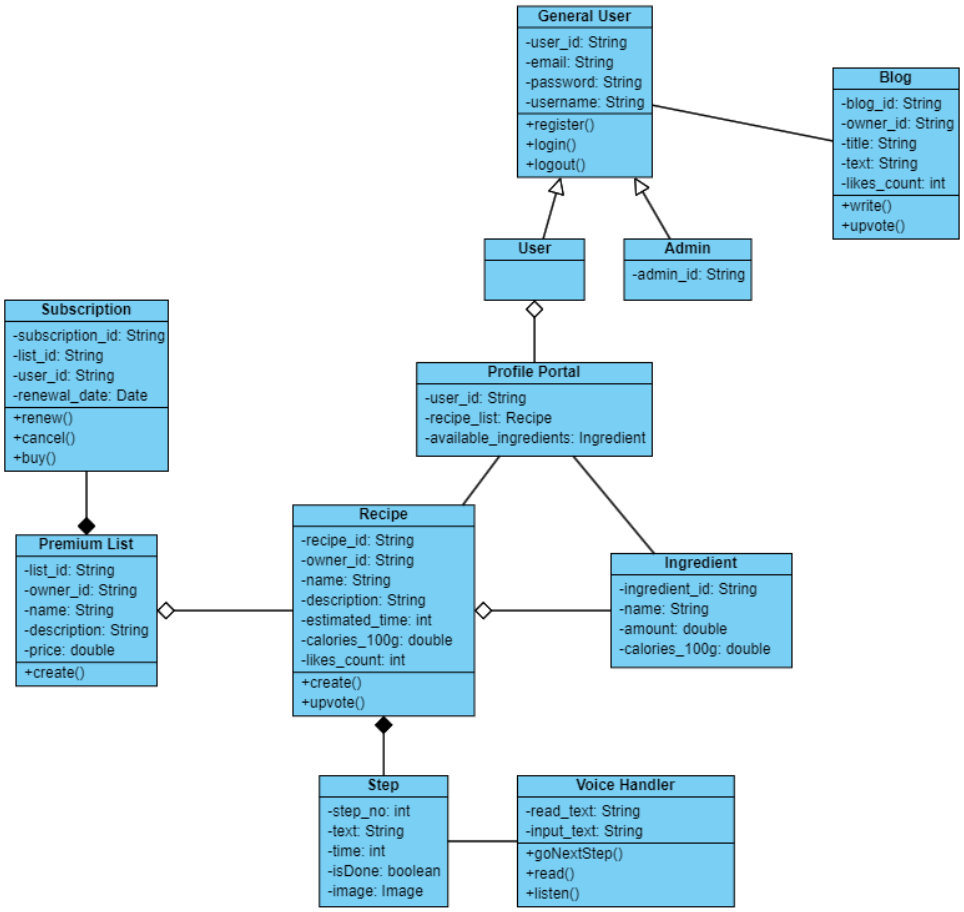


Fig. 3: Class Diagram

We are using Google Cloud Service for voice handling. The voice handler listens for input to proceed to the next step and it reads what to do in each step.

3.5.4. Dynamic Models

3.5.4.1. Activity Diagrams

3.5.4.1.1. Ingredient Management

A user can add or remove ingredients in their personal profile.

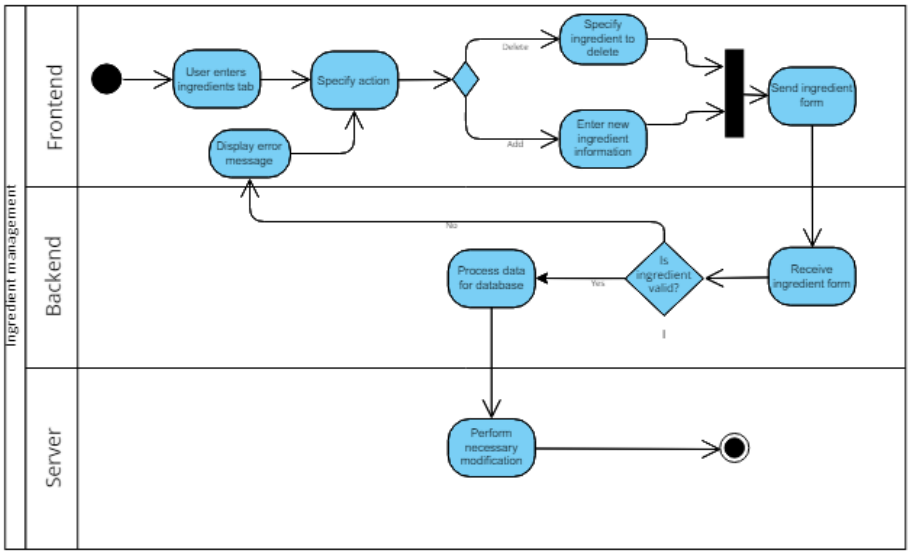


Fig. 4: Ingredient Management State Diagram

3.5.4.1.2. Recipe Input

A user enters recipe information into the app. The backend and the database do proper operations to securely add the recipe to the app.

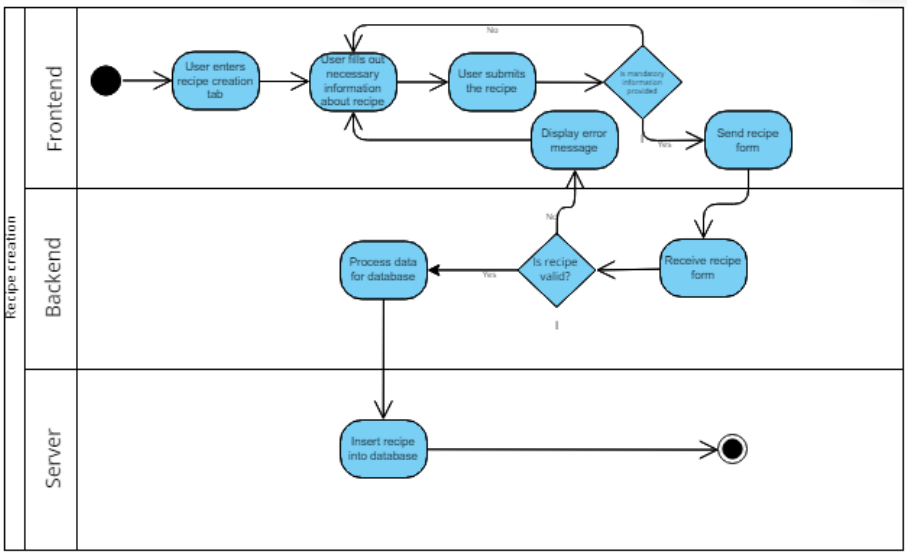


Fig. 5: Recipe Creation State Diagram

3.5.4.2. State Diagrams

3.5.4.2.1. Recipe Following States

Proper and seamless recipe guidance should be implemented. The app will ensure this by going step by step.

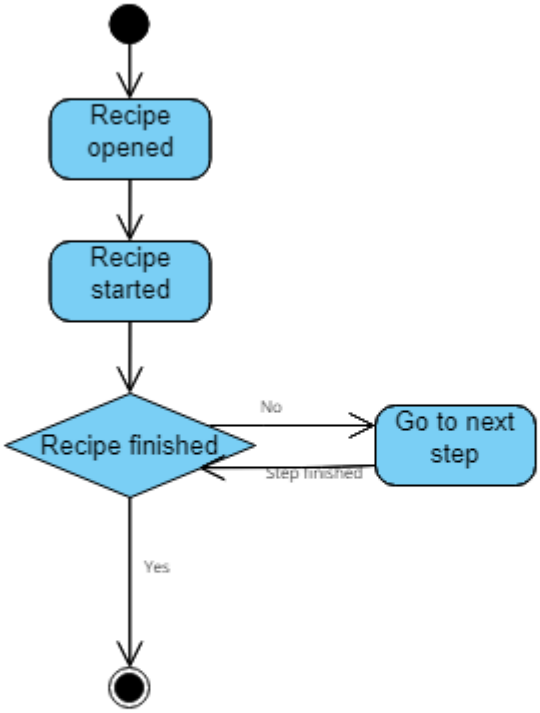
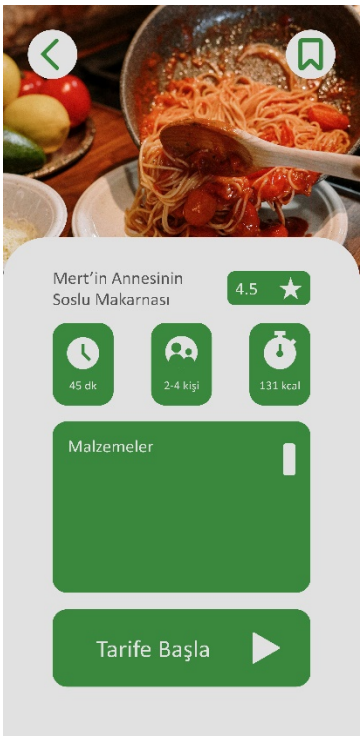
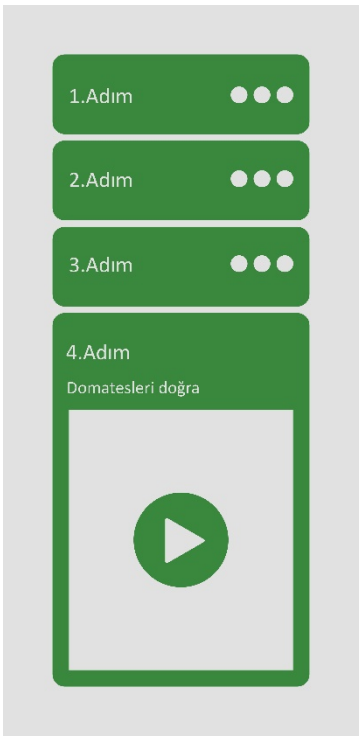


Fig. 6: Recipe Following State Diagram

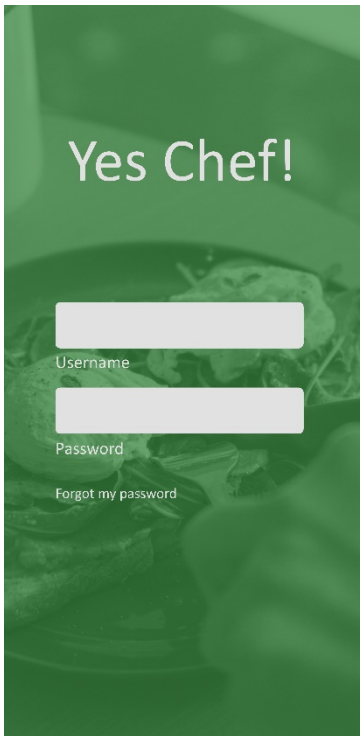
3.5.5. User Interface - Navigational Paths and Screen Mock-ups



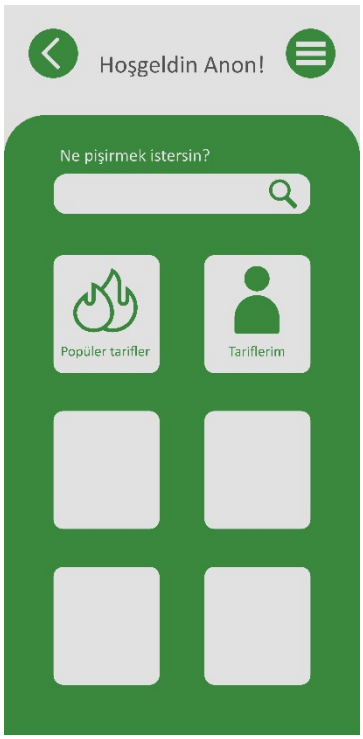
Recipe Details Page



Recipe Steps Page



Login Page



Main Menu Page

4. Other Analysis Elements

4.1. Considerations of Various Factors in Engineering Design

In this section, many aspects that may affect the Yes Chef! design will be discussed.

4.1.1. Constraints

The development of Yes Chef! involves several constraints that must be considered to ensure its feasibility, usability, and effectiveness. These constraints are categorized as follows:

Technical Constraints

- **Minimum Hardware and Software Specifications**:
  - The app must function efficiently on standard smartphones and tablets with mid-range specifications, including:
    - Minimum RAM: 2 GB.
    - Operating Systems: Android 8.0+ and iOS 12+.
- **Interoperability**:
  - The app must integrate with other platforms such as social media (for sharing recipes).
- **Extensibility**:
  - The design must allow for future feature expansions. These expansions may include different subscription services, more detailed recipes, UI improvements etc.

Functional Constraints

- **Usability**:
  - The interface must cater to a diverse audience, such as individuals with limited technical expertise.
- **Accessibility**:
  - The app must comply with accessibility standards, such as WCAG[1], to accommodate all types of users.

Economic Constraints

- **Cost**:
  - The app must be affordable for end-users, with a freemium model offering basic features for free and a subscription service for advanced functionalities.
- **Marketability**:
  - The app must appeal to a global audience with diverse cooking habits and preferences.

Cultural and Social Constraints

- **Globalization**:
  - Recipes and features must cater to various cultural cuisines and dietary restrictions (e.g., vegan, halal, kosher).

- **Social Factors**:
    - Features such as sharing and reviewing must promote positive interactions and avoid unhealthy competition.

Environmental Constraints

- **Sustainability**:
    - Promote sustainable cooking practices by suggesting recipes that minimize food waste and highlight eco-friendly options.

Policy and Legal Constraints

- **Data Privacy Regulations**:
    - The app must comply with data protection laws such as GDPR[2].
- **Copyright Compliance**:
    - Ensure recipes and images shared by users do not infringe copyright laws.

Schedule Constraints

- **Development Timeline**:
    - The project must adhere to a strict timeline to meet key milestones, including prototype delivery, testing, and launch.

	Effect Level (0-10)	Effect
Public Health	9	With a higher incline to home cooking, people will be eating healthier and more balanced meals.
Public Safety	8	Food safety will be ensured by reminding users to check labels for expiration dates, including proper cooking techniques and necessary safety guidelines for more dangerous recipes.
Public Welfare	3	By promoting a community-driven medium for recipe sharing, the social well-

		being of users will be enhanced.
Global factors	6	By providing multi-language support, people all over the globe will be able to use the app.
Cultural Factors	5	By creating a respectful and inclusive medium for recipe sharing, people will be able to experience meals of other cultures.
Social Factors	4	A lot of engagement between users will happen via recipe sharing, and reviews.
Environmental Factors	8	Minimizing food waste and reducing fast food consumption will lead to an eco-friendlier approach to nutrition.
Economic Factors	7	With a mix of the freemium and the premium subscription models, the app will remain feasible while also remaining affordable.

4.1.2. Standards

The development of Yes Chef! involves several standards that must be considered to ensure its practicality, accessibility, and security. These standards are categorized as follows:

- All developers will follow the ACM Code of Ethics and Professional Conduct [3] to ensure a professional and responsible lifecycle for the project.
- Code quality will adhere to the SOLID principles [4] in object-oriented designs. Also, an external code formatter tool Prettier will be used to ensure maintainable and understandable code.

- The protection of user privacy will be ensured by following the ISO/IEC 27001 standard for information security management systems [5].
- The payment process for the subscription system will comply with Payment Card Industry Data Security Standards [6] to ensure secure transactions.
- UML 2.5.1 will be used for modeling.
- IEEE format will be used for citations in the documents.

4.2. Risks and Alternatives

YesChef! functions by several connected systems that are all working simultaneously. The gamification engine works by gathering relevant event nodes and timing information from the database. If the database response is slow or it fails either fully or partially, the experience will suffer greatly. The voice recognition module serves the purpose of hands free recipe following and it contains a risk of not functioning , due to connections with Google voice to text not functioning or other reasons. This will hamper the experience greatly and the only fallback to this issue is the “Next” button requiring physical touch.

Risk	Likelihood	Effect	Alternative
Game Engine fails to compile the relevant recipe data	Medium	Recipe cannot be loaded.	Storing common nodes prematurely and only requesting additional nodes and media from the database.
Voice recognition may fail	High	Touch-free functionality does not function	Several attempts for recognition after the first failure and a next button pop up after a specific time to progress through the recipe.

4.3. Project Plan

Assignment	Title	Members
A1	CS491 Reports	All Members
A2	CS492 Reports	All Members
A3	CS491 Demo	All Members

A4	CS492 Demo	All Members
A5	Development of the Game engine	Mert Terkuran İsmail Barış Sunar
A6	Development of the application Layout	All Members
A7	Integration between the backend and frontend	All Members
A8	Database Design and AWS integration	Serhan Turan Ulaş Keskin Mert Yamalı
A9	Voice Recognition Integration	All Members
A10	IOS support	Serhan Turan Ulaş Keskin Mert Yamalı
A11	Recipe Library Creation	All Members

Detailed Assignments:

A1: CS 491 Reports	All Members
Objective: Constructing required project reports.	
Tasks: 1. Project Specification Report: 1.1. Determining the project topic, contacting a supervisor and the innovation expert. 2. Analysis and Requirement Report: 2.1. Formalize specifications for the YesChef! Project.	

A2: CS 492 Reports	All Members
Objective: Constructing required project reports.	
Tasks: 1. Detailed Design Report: 1.1. Constructing the detailed design of the YesChef! Project. 2. Finalized Project Report: 2.1. Create the finalized report of all functionalities and properties of the finished project.	

A3:CS 491 Demo	All Members
Objective: Showing off the progress made in the development of the project to supervisors.	
Tasks: 1. Creating a prototype: Implement a snapshot of the final product and show the critical parts of the app experience.	

A4:CS 492 Demo	All Members
Objective: Showing off the progress made in the development of the project to supervisors.	
Tasks: 1. Show the finalized project to supervisors.	

A5: Development of the game engine.	Mert Terkuran İsmail Barış Sunar
Objective: Creating the game engine to be used to create gamified recipes.	
Tasks: 1. Create the engine to output usable data that can be decoded by the app to create gamified recipes. 2. Simplify the creation process to allow users to create gamified recipes of their own.	

A6: Development of the application Layout	All Members
Objective: Creating the UI and design of the application.	
Tasks: 1. Design an intuitive UI and simple navigation for ease of use. 2. Code modularly and allow implemented modules to interface with future features and systems by doing the infrastructure work early.	

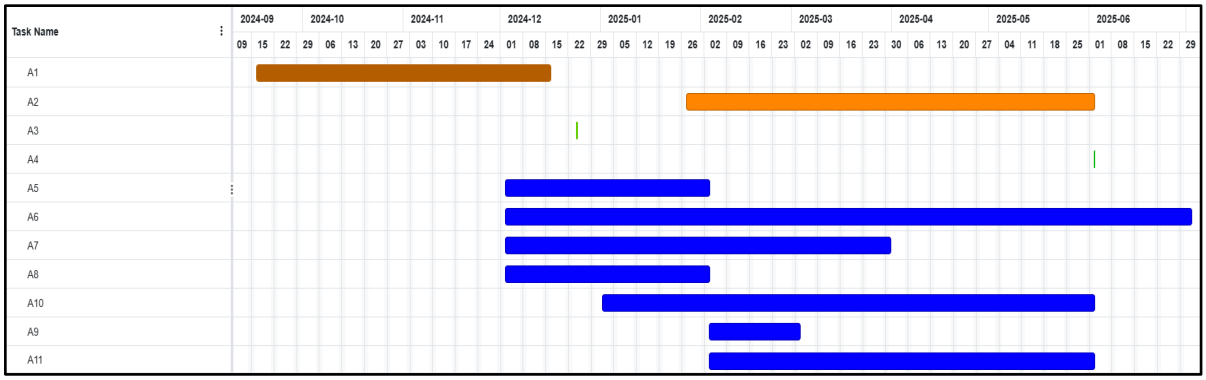
A7: Integration between the backend and frontend	All Members
Objective: Set up endpoints for Get and Post operations.	
Tasks: 1. Plan and implement required endpoints for data access early so that backend integration can be simplified.	

A8: Database Design and AWS integration	Serhan Turan Ulaş Keskin Mert Yamalı
Objective: Designing and utilizing the database.	
Tasks: 1. Connect the AWS service to the backend and set up the communication between the frontend and the backend.	
A9: Voice Recognition Integration	All Members
Objective: Set up Google Voice Recognition inside the game engine.	
Tasks: 1. Integrate Google API's and their functionality into the recipe process.	
A10: IOS Support	Serhan Turan Ulaş Keskin Mert Yamalı
Objective: Assure cross-platform functionality.	
Tasks: 1. Consistently check and ensure implemented features work the same across platforms.	
A11: Recipe Library Creation	All Members
Objective: Expand the gamified recipe library.	

Tasks:

1. Create gamified recipes using the game engine to enrich the app library.

Gantt Chart:



4.4. Ensuring Proper Teamwork

- Deadlines and assignments are decided by the group, and daily updates are shared between members.
- Although each member works in different sections, members have specific areas that they are responsible for.
- Work division is decided democratically and reasonably to be as fair as possible.

4.5. Ethics and Professional Responsibilities

- The source code is privately accessible by the group on GitHub. The private code should not be shared by third parties until the end of the project.
- Frameworks used, such as Expo, Nativewind, Capacitor, etc., should be given credit, and their licenses should be abided by.

4.6. Planning for New Knowledge and Learning Strategies

We are using react-native to provide cross-platform support. All of the group members have experience using React for web development; however, none of us have worked on mobile development. For this reason, we are using the Expo framework. We are using the official react-native and expo guides as well as Chatgpt and several YouTube tutorials for training. However, the expo framework is a rapidly changing system, and our tutorials inevitably have deprecated information. To this end, we are creating simple guides for tricky processes so that similar problems do not cause problems for other team members. Moreover, we are working with AWS, which we do not have any experience with. For this reason, we have joined the AWS Educate program by Amazon for guides. For all of these training processes, team members were assigned specialty areas so that they have more knowledge of a problem scenario than other members.

5. Glossary

Gamification: Application of elements of game playing to other areas of activity.

Multimedia: Using more than one media medium, such as video, movies, or audio for communication.

UI / UX: User interface / User experience.

WCAG: Web Content Accessibility Guidelines.

Freemium: A business model type where basic features of a service are offered to users at no cost, while more premium features are offered behind a paywall.

GDPR: General Data Protection Regulation

6. References

[1] "Web content accessibility guidelines (WCAG) 2.1," W3C, https://www.w3.org/TR/WCAG21/ (accessed Dec. 14, 2024).

[2] "GDPR," General Data Protection Regulation (GDPR), https://gdpr-info.eu/ (accessed Dec. 14, 2024).

[3] The code affirms an obligation of computing professionals to use their skills for the benefit of society. (n.d.). Retrieved from https://www.acm.org/code-of-ethics

[4] "Solid: The first 5 principles of object oriented design," DigitalOcean, https://www.digitalocean.com/community/conceptual-articles/s-o-l-i-d-the-first-five-principles-of-object-oriented-design (accessed Nov. 18, 2024).

[5] "ISO/IEC 27001:2022," ISO, https://www.iso.org/standard/27001 (accessed Nov. 18, 2024).

[6] "Standards," PCI Security Standards Council, https://www.pcisecuritystandards.org/standards/ (accessed Nov. 18, 2024).