CS 492 - Senior Design Project

Detailed Design Report

T2432 - Yes Chef!

İsmail Barış Sunar - 22003479

Mert Emre Yamalı - 22102932

Mert Terkuran - 22101645

Serhan Turan - 22001880

Ulaş Keskin - 22101766

Supervisor: Uğur Doğrusöz

Innovation Expert: Muhammed Naci Dalkıran

Table of Contents

1. Introduction

1.1. Purpose of the System

Yes Chef! is designed to revolutionize the cooking experience by providing an engaging, interactive, and user-friendly platform for home cooks of all skill levels. The system addresses common challenges related to meal planning, ingredient tracking, and recipe discovery by incorporating gamification and personalized recommendations.

The primary objectives of Yes Chef! include:

- **Simplifying Meal Preparation:** The system offers users curated recipe recommendations based on their available ingredients, reducing the time spent on meal planning.
- **Enhancing User Engagement through Gamification:** By integrating achievements, points, and rewards, Yes Chef! transforms cooking into an enjoyable and rewarding activity rather than a chore.
- **Integrating a Voice Assistant for Hands-Free Cooking:** Yes Chef! includes a voice assistant feature that allows users to navigate through recipes, set timers, and receive cooking instructions without needing to touch their devices, enhancing convenience and usability in the kitchen.
- **Optimizing Ingredient Usage:** The ingredient tracking feature helps users manage their pantry effectively, minimizing food waste and encouraging efficient grocery shopping.
- **Encouraging Social Interaction and Community Building:** The platform fosters a sense of community by enabling users to share recipes, interact with others, and discover new culinary ideas.
- **Ensuring Accessibility and Ease of Use:** With an intuitive UI/UX design, the application ensures that users can easily navigate its features, regardless of their technical proficiency.

By combining these features, Yes Chef! aims to create a comprehensive, interactive, and sustainable cooking solution that caters to modern lifestyle needs while making cooking an engaging and enjoyable process.

1.2. Design Goals

1.2.1. 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.

1.2.2. 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.

1.2.3. 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.

1.2.4. Aesthetics

- UI should make users desire food like colors from McDonald's.
- UI also should create a healthy ambiance for users. They should think the app helps them.

1.2.5. 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.

1.2.6. 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.

1.3. Definitions, Acronyms, and Abbreviations

We did not use any definition, acronym, or abbreviation in this report.

1.4. Overview

Cooking and meal planning is an essential part of daily life, but for a lot of people it can be a very overwhelming, time-consuming and an overall dreadful task, especially when you have to decide what to cook with your available ingredients every day. Furthermore, meal planning, managing groceries, discovering good recipes, and even the overall process of cooking is very tedious to many. Research indicates that only 10% of people love cooking while 45% of people hate it [1]. This general trend pushes people to resort to unhealthy takeout options. Yes Chef! aims to solve these challenges by providing a gamified cooking application that tackles all of these issues by integrating a recommendation system, ingredient tracking, and a reward system to enhance the process of cooking.

Yes Chef! is a mobile cooking application that allows users to share recipes, track ingredients, receive personalized recipe recommendations, and collect points for interaction with the app. The application also provides a platform for users to be able to interact with each other via social interaction features, allowing people with similar culinary preferences to form communities and share experiences. The gamification aspect of the application will also reward users with badges, achievements, and points to further make cooking more enjoyable and make it feel like a game rather than a task.

Yes Chef! stands out from other traditional recipe apps by leveraging its sophisticated gamification features. Unlike the standard traditional recipe apps which fail to make cooking interesting and approach it more like a task that should be done as efficiently and as fast as possible, Yes Chef! uses its architecture to provide users with the ability to cook the best meal they can while still having fun. Also, unlike traditional recipe apps which provide only static recipes based on queries, Yes Chef! can dynamically suggest meals based on users' preferences and available ingredient options, making the app feel even more enticing.

With Yes Chef!, we aim to revolutionize home cooking, filling gaps that other recipe apps have failed to fill by making cooking more accessible, engaging, and most importantly enjoyable while ensuring users maximize their ingredient use and discover new dishes effortlessly.

This report provides a detailed analysis of YesChef's system architecture, functional and non-functional requirements, subsystem decomposition, and design principles. The following sections will compare existing solutions, explain how YesChef differentiates itself, and provide a deep dive into the technical implementation, including APIs, database models, and integration strategies. Furthermore, the report will discuss various engineering considerations, including public health, safety, security, and sustainability aspects. Finally, test cases and system validation will be outlined to ensure that YesChef meets performance, security, and usability requirements.

2. Current Software Architecture

Many widely known cooking applications and websites are available for users. We analyzed them and found a huge potential. We came up with Yes Chef. We are implementing new features that our competitors do not have. There are essential features for this area and those are recipe guides, calorie information, a blog, and a forum.

- We have all the features above except the forum because we do not want users to interact with each other.
- We have gamified recipe guides to make cooking more fun and more accessible. We added a voice assistant because hands get dirty and wet while cooking, so users cannot constantly interact with their device using their hands.
- Yes Chef! has a gamification engine for creating premium recipes and sharing them with all users. It also has basic recipes as a blog.
- Users can specify what ingredients they have, and the application suggests recipes they can cook with their own ingredients.
- The application has a detailed calorie tracker for each recipe and ingredient (if possible). Calorie and nutrition information is available.

3. Proposed Software Architecture

3.1. Overview

YesChef! is built with a robust technology stack to ensure scalability, performance, and seamless user experience. The backend is developed using .NET, providing a structured and efficient API layer that handles business logic, authentication, and data processing. Firebase is integrated for authentication, push notifications, and media storage, allowing real-time interactions and secure user management. MongoDB is used as the primary database for storing user-generated content, including recipes, pantry items, and interactions, offering flexibility in data modeling and rapid read/write operations. React Native is used to build the interface that is native to mobile applications.

Future development includes additional backend enhancements that will focus on refining API performance, implementing caching strategies, and optimizing data synchronization between services. As the platform grows, architectural improvements such as containerization with Docker and microservices-based deployment strategies will be explored to ensure scalability and maintainability.

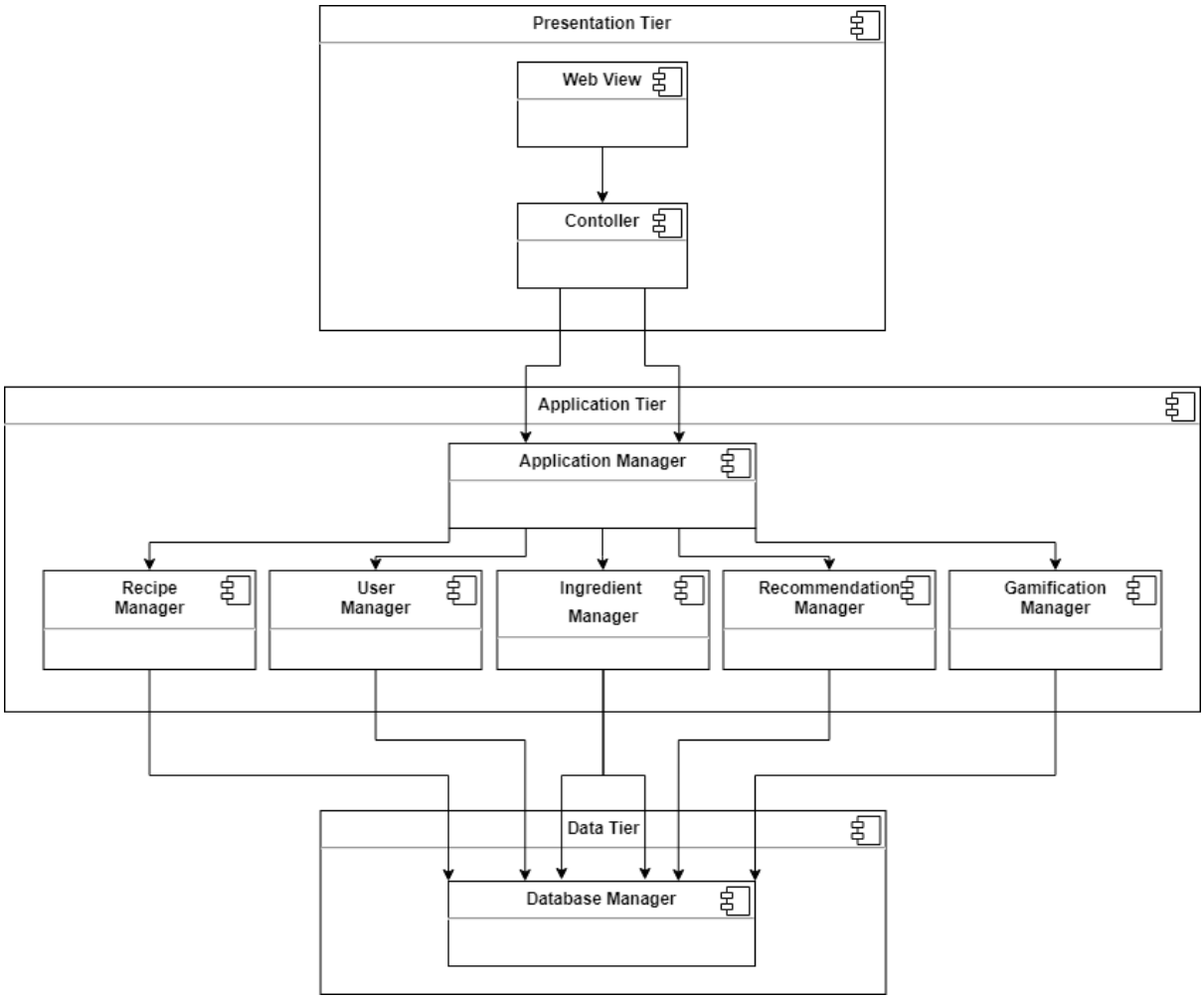

Figure 1: Subsystem Decomposition

3.2. Hardware/Software Mapping

Since Yes Chef! does not include any IoT devices or dedicated hardware, it leverages cloud infrastructure and mobile device capabilities for a seamless, scalable, and efficient user experience.

3.2.1. Server-Side Infrastructure (Backend & Database)

- **Cloud Provider:** Backend is hosted on AWS, Firebase is used for authentication and media management.
- **Backend Server:**
  - .NET Core running on a Linux-based virtual machine/container.
  - Cloud-based virtual machines or Kubernetes clusters with CPU and memory for efficient API request handling.
- **Database Server:**
  - MongoDB Collections for data management.
  - Cloud-hosted database services or dedicated database instances with SSD storage for high-speed read/write operations.

3.2.2. Client-Side Infrastructure (Mobile Application)

- **Devices:** Smartphones or tablets running IOS and Android.
- **Software:**
  - React Native for cross-platform compatibility.
  - Firebase SDK for authentication, push notifications, and media storage.
  - SQLite or AsyncStorage for caching user data and offline access.
- **Hardware Mapping:** Mobile devices with minimum hardware requirements:
  - CPU: ARM-based processors (Apple-A series, Qualcomm Snapdragon, or MediaTek)
  - RAM: Minimum 2GB for smooth app performance.
  - Storage: At least 100MB required for app installation, additional storage is needed for caching images, user preferences, and offline data.

3.2.3. DevOps & Deployment Infrastructure

- **Continuous Integration/Continuous Development(CI/CD):**
  - GitHub actions, AWS, Firebase App Distribution for automated builds and deployment.
  - Containerization with Docker for scalable backend development.
- **Hosting & API Gateway:**
  - Software: .NET APIs deployed via AWS Lambda, or a dedicated VPS.
  - Hardware: Cloud-based instances with auto-scaling capabilities for effective user traffic handling.

3.3. Persistent Data Management

Our project requires storing our recipes and users. We have images and videos for recipes. Each recipe could have many multimedia items. Therefore, we will store multimedia items in Firebase and use MongoDB for basic storage. MongoDB is our primary database. Our basic items and URLs' of multimedia items will be stored in MongoDB.

3.4. Access Control and Security

We have login and signup functionalities. They are sufficient for this project. We could add mail verification if we decide that it is a necessary element. Also, the privacy of our users is crucial for us. We encrypt their important data fields and do not share their data with 3rd party applications.

4. Subsystem Services

4.1. UI/UX Service

Functionality: Main goal of the UI/UX systems in our application is to provide an intuitive and user-friendly interface.

Components: React Native

Integration: Connects to User Management Service for authentication. Fetches and displays data from Recipe Management Service, Ingredient Tracking Service, Gamification & Achievement Service, Recommendation Engine, Social Interaction & Community Engagement Service, Push Notifications & Alerts Service, Analytics & Insights Service, Admin & Moderation Service.

4.2. User Management Service

Functionality: The role of the User Management Service is to handle authentication and user roles.

Components: Authentication token based login.

Integration: Fetches from and stores data to User Database.

4.3. Recipe Management Service

Functionality: The role of Recipe Management Service is to allow users to create, edit, delete and save recipes.

Components: Recipe CRUD operations.

Integration: Fetched data and stores recipes into Recipe Database. Works with Recommendation Engine for personalized suggestions.

4.4. Ingredient Tracking Service

Functionality: The role of the Ingredient Tracking Service is to both manage available ingredients in the users pantry and also filter recipes with selected ingredients.

Components: Ingredient CRUD operations.

Integration: Fetches data and stores ingredients in the Ingredient Inventory. Works with Recommendation Engine for personalized suggestions.

4.5. Gamification & Achievement Service

Functionality: The role of Gamification & Achievement Service is to implement a reward system with points and badges to further increase the gamification aspect of the application as well as improve user experience.

Components: .NET Web API.

Integration: Updates the UI, works with Points & Achievements to store data.

4.6. Recommendation Engine

Functionality: The role of the recommendation engine is to recommend recipes with available ingredients and user preferences.

Components: .NET API Endpoints.

Integration: Works with Ingredient Inventory, User Database, and Recipe Database to find suitable recipes.

4.7. Social Interaction & Community Engagement Service

Functionality: The role of Social Interaction & Community Engagement Service is to allow and enable users to interact with each other.

Components: .NET Web API CRUD operations.

Integration: Works with User Database.

4.8. Push Notifications & Alerts Service

Functionality: Sends appropriate notifications and alerts. These could be ingredient expiration alerts, new recipe alerts, comments, achievements etc.

Components: .NET Background Services to trigger reminders.

Integration: Works with User Database, Recipe Database, Ingredient Inventory, Points & Achievements Data, and Recommendation Data to fetch appropriate changes in the database.

4.9. Analytics & Insights Service

Functionality: The role of Analytics & Insights Service is to track user behavior, recipe popularity, and engagement trends to pick correct recipes for recommendation.

Components: .NET Web API for custom analytics queries.

Integration: Works with User Database, Recipe Database, and Recommendation Data to find correct trends.

4.10. Admin & Moderation Service

Functionality: The role of Admin & Moderation Service is to provide a panel where content in the application can be overlooked and managed.

Components: .NET Web API Admin Dashboard.

Integration: Works with User Database, Recipe Database, Ingredient Inventory, Points & Achievements Data, and Recommendation Data to be able to track every data in the app.

4.11. User Database

Functionality: The user database is responsible for storing and managing all sorts of user-related data.

Components: MongoDB Collection, Firebase Storage.

Integration: Has relations with Recipe Database, Ingredient Inventory, Points & Achievements Data.

4.12. Recipe Database

Functionality: The Recipe Database is responsible for storing all sorts of recipes.

Components: MongoDB Collection, Firebase Storage.

Integration: Has relations with Ingredient Inventory.

4.13. Ingredient Inventory

Functionality: The Ingredient Inventory is responsible for storing and managing ingredients.

Components: MongoDB Collection, Firebase Storage.

Integration: Has relations with Recipe Database, and User Database.

4.14. Points & Achievements Data

Functionality: The Points & Achievements Data is responsible for tracking individual points, achievements and badges of users.

Components: MongoDB Collection.

Integration: Has relations with User Database.

4.15. Recommendation Data

Functionality: The Recommendation Data is responsible for processing user preferences, cooking history, and available ingredients to provide personalized recipe suggestions.

Components: MongoDB Collection.

4.16. Logging & Analytics

Functionality: The role of Logging & Analytics is to track system performance, user activity, and application errors for monitoring and debugging.

Components: MongoDB Collection, .NET Background Services, Google Analytics for Firebase.

Integration: Works with Analytics & Insights Service to generate reports on user engagement, with Admin & Moderation Service to monitor flagged content and suspicious activity, and with Push Notifications & Alerts Service to send alerts when critical system issues occur.

5. Test Cases

5.1. Functional Test Cases

Test ID	F001	Category	Functional	Severity	Critical
Objective	Verify a user can register with valid inputs.				
Steps	1. Navigate to the registration page. 2. Fill out the registration form with valid inputs 3. Click the register button				
Expected results	● Registration is confirmed and the user is redirected to the homepage. ● New user added to database				
Results	Empty for this report.				

Test ID	F002	Category	Functional	Severity	Critical
Objective	Verify a user can’t register with invalid inputs.				
Steps	1. Navigate the registration page 2. Fill the registration form with empty or invalid input. 3. Click the register button.				
Expected results	● Registration is unsuccessful and the error message pops up.				
Results	Empty for this report.				

Test ID	F003	Category	Functional	Severity	Critical
Objective	Verify a user can login with valid inputs.				
Steps	1. Navigate to login page 2. Enter valid credentials of existing user to the login form 3. Click login				
Expected results	● Login is successful and the user is redirected to the homepage.				
Results	Empty for this report.				

Test ID	F004	Category	Functional	Severity	Critical
Objective	Verify a user can’t login with invalid inputs.				
Steps	1. Navigate to login page 2. Enter invalid or empty inputs to login form 3. Click login				
Expected results	● Login is unsuccessful, error message pops up.				
Results	Empty for this report.				

Test ID	F005	Category	Functional	Severity	Critical
Objective	Verify a user can edit his profile with valid inputs.				
Steps	1. Navigate to the profile page. 2. Click “Edit the Profile” button 3. Enter non-empty inputs to the edit profile form. 4. Click confirm profile button				

Expected results	● Profile edit is successful, and the user is redirected to his profile page. ● The user is updated in the database according to the changes.
Results	Empty for this report.

Test ID	F006	Category	Functional	Severity	Critical
Objective	Verify a user can't edit his profile with invalid inputs.				
Steps	1. Navigate to the profile page. 2. Click edit the profile button 3. Enter empty or invalid inputs to the form. 4. Click confirm profile button				
Expected results	● Profile edit is unsuccessful, error message pops up to the user				
Results	Empty for this report.				

Test ID	F007	Category	Functional	Severity	Major
Objective	Check if the search bar in the recipes menu gives a matched result when the user enters a nonempty string.				
Steps	1. Navigate to the recipes page. 2. Click the search bar 3. Enter a non empty text in search bar field 4. Click enter				
Expected results	● Only matched results are shown to the user. If there are no matches, a 'No matches found' message is displayed.				
Results	Empty for this report.				

Test ID	F008	Category	Functional	Severity	Major
Objective	Check the search bar in the recipes menu gives an error message if the user enters empty input.				
Steps	1. Navigate to the recipes page. 2. Click the search bar 3. Enter an empty text in search bar field 4. Click enter				
Expected results	● “Please enter a text” message shown to the user.				
Results	Empty for this report				

Test ID	F009	Category	Functional	Severity	Major
Objective	Check if the user can see the desired recipe’s info page.				
Steps	1. Navigate the recipes page 2. Select desired recipe 3. Click the recipe				
Expected results	● App calls spoonacular api to get information about the recipe. The information page which includes total time, image and ingredients is shown to the user.				
Results	Empty for this report				

Test ID	F010	Category	Functional	Severity	Major
Objective	Check if the user can view a recipe.				
Steps	1. Navigate the recipes page 2. Select desired recipe 3. Click the recipe 4. Click “start” button				
Expected results	● The user is redirected to the recipe view, starting from Step 1.				

Results	Empty for this report

Test ID	F011	Category	Functional	Severity	Major
Objective	Check if the recipe steps with a timer can show the remaining time for the step.				
Steps	1. Navigate the recipes page 2. Select desired recipe 3. Click the recipe 4. Click “start” button 5. User comes to a step that includes timer.				
Expected results	● A timer pop-up appears on the rightmost side of the page.				
Results	Empty for this report				

Test ID	F012	Category	Functional	Severity	Major
Objective	Check when the timer completes, it warns the user.				
Steps	1. Navigate the recipes page 2. Select desired recipe 3. Click the recipe 4. Click “start” button 5. During the recipe viewing one of the timers reaches at 00:00.				
Expected results	● An alarm sound plays, and the pop-up enlarges on the page for the user to close it.				
Results	Empty for this report				

Test ID	F013	Category	Functional	Severity	Major
Objective	Check when the user says yes chef, the user is redirected to the next step of the recipe.				
Steps	1. Navigate the recipes page 2. Select desired recipe 3. Click the recipe 4. Click “start” button 5. During the recipe viewing the user says “Yes Chef”.				
Expected results	● Text to speech api handles the text to speech transformation. It gives a text. In the backend it is checked that the user says “Yes Chef”, and the user is redirected to the next step of the recipe.				
Results	Empty for this report				

Test ID	F014	Category	Functional	Severity	Minor
Objective	Check if the upvoted recipe’s upvote count is updated.				
Steps	1. Navigate the recipes page 2. Select desired recipe 3. Click the recipe 4. Click the upvote button				
Expected results	● In the database upvote count of selected recipe should be incremented by one.				
Results	Empty for this report				

Test ID	F015	Category	Functional	Severity	Minor
Objective	Check the database if the blog is posted with valid input fields.				
Steps	1. Navigate to the blog menu. 2. Click create a blog button 3. Fill out the title and text fields with valid and non-empty inputs 4. Click the submit the blog button.				
Expected results	● New blog is created in the database.				
Results	Empty for this report				

Test ID	F016	Category	Functional	Severity	Major
Objective	Check the response if the blog is not posted with invalid input fields.				
Steps	1. Navigate to the blog menu. 2. Click create a blog button 3. Fill out the title and text fields with invalid or empty inputs 4. Click the submit the blog button.				
Expected results	● An error message that says “Invalid text or title fields” is shown to the user.				
Results	Empty for this report				

Test ID	F017	Category	Functional	Severity	Minor
Objective	Check the database if the blog is updated with valid and non-empty input fields.				
Steps	1. Navigate to the blog menu. 2. Click my blogs button. 3. Click the blogpost which is going to be updated 4. Make desired changes in blog posts with valid inputs				
Expected results	● In the database selected blogpost is updated.				
Results	Empty for this report				

Test ID	F018	Category	Functional	Severity	Major
Objective	Check the response if the input field of one step is valid when creating a recipe.				
Steps	1. Navigate to the recipes menu. 2. Click create a recipe 3. Click the add step button. 4. Choose the type of step as text. 5. Enter a valid text 6. Click done				
Expected results	● “Step is created” message is shown to the user. ● Step is added to the json file.				
Results	Empty for this report				

Test ID	F019	Category	Functional	Severity	Major
Objective	Check image, gif and video import when creating a recipe.				
Steps	1. Navigate to the recipes menu. 2. Click create a recipe 3. Click the add step button. 4. Choose the type of step as visual. 5. From your gallery select image, gif or video with certain sizes. 6. Click import				
Expected results	● Path of the image should be added to the json file of recipe.				
Results	Empty for this report				

Test ID	F020	Category	Functional	Severity	Major
Objective	Check the response if the valid recipe is added to the database.				
Steps	1. Navigate to the recipes menu. 2. Click create a recipe 3. Add steps to the recipe 4. Click complete recipe button				
Expected results	● New row for the recipe should be added to the database.				
Results	Empty for this report				

Test ID	F021	Category	Functional	Severity	Minor
Objective	Check the response when a recipe is shared				
Steps	1. Navigate to the recipes menu. 2. Select a recipe 3. Click the recipe 4. In the information page click share button 5. Choose one of your contacts.				
Expected results	● Receiver should get a message that redirects to the recipe.				
Results	Empty for this report				

Test ID	F022	Category	Functional	Severity	Major
Objective	Check the response when a user wants to buy a subscription plan and enters valid bank information.				
Steps	1. Navigate to the subscription plans menu. 2. Select which type of plan to buy (asian, mexican, middle east, etc.) 3. Enter credit card information 4. Click buy				
Expected results	● Api calls are made to the paying services. ● Money is withdrawn from users bank account ● In the database plan is added to the user's subscription plans.				
Results	Empty for this report				

Test ID	F023	Category	Functional	Severity	Major
Objective	Check the response when a user wants to buy a subscription plan and enters invalid bank information.				
Steps	1. Navigate to the subscription plans menu. 2. Select which type of plan to buy (asian, mexican, middle east, etc.) 3. Enter credit card information 4. Click buy				
Expected results	● App shows an error message that says “Invalid credit card information”.				
Results	Empty for this report				

5.2. Non-functional Test Cases

Test ID	NF001	Category	Performance	Severity	Critical
Objective	Ensure the application launches and displays the home screen quickly under typical operating conditions.				
Steps	1. Launch the application on a standard device. 2. Use a stopwatch or automated timer to measure the duration from launch initiation to full home screen display. 3. Repeat the test across different network conditions and device models. 4. Record the load times for each iteration.				
Expected results	● The home screen must load within 2 seconds in at least 95% of all iterations.				
Results	Empty for this report				

Test ID	NF002	Category	Performance	Severity	Major
Objective	Verify that multimedia (images and videos) uploads during recipe creation complete within acceptable time limits.				
Steps	1. Initiate a recipe creation process and add multiple multimedia files. 2. Monitor the upload process for each file under standard network conditions. 3. Record upload durations for different file sizes and formats. 4. Analyze the results against the target performance threshold.				
Expected results	● All multimedia uploads should complete within 2 seconds for 95% of attempts.				
Results	Empty for this report				

Test ID	NF003	Category	Performance	Severity	Critical
Objective	Confirm that voice recognition processes and responds to user commands within an acceptable time frame.				
Steps	1. Start a voice-activated recipe session. 2. Issue the command “Yes Chef!” using a standard microphone setup. 3. Measure the time taken for the system to process and respond to the command. 4. Repeat the test in different ambient noise conditions and document the response times.				
Expected results	● The system must respond and advance to the next step within 5 seconds in the majority of cases.				
Results	Empty for this report				

Test ID	NF004	Category	Security	Severity	Critical
Objective	Validate that all sensitive user data is encrypted both during transmission and while stored on the device/server.				
Steps	1. Review network communication during login and payment transactions using network monitoring tools. 2. Verify that no sensitive data (passwords, credit card details) is transmitted in plain text. 3. Inspect data storage logs or backups to confirm that stored sensitive data is encrypted as per ISO/IEC 27001 standards. 4. Document any deviations from the encryption protocols.				
Expected results	● No sensitive data should be visible in plain text during transmission or in storage.				
Results	Empty for this report				

Test ID	NF005	Category	Stability	Severity	Major
Objective	Ensure that the application remains stable and does not crash or degrade in performance during extended use.				
Steps	1. Run the application continuously for at least 4 hours while performing typical user actions (navigation, recipe creation, multimedia uploads). 2. Monitor system logs and resource usage (memory, CPU) throughout the session and identify and record any crashes, performance slowdowns, or error messages. 3. Test on multiple devices and OS versions to check consistency.				
Expected results	● The application should operate continuously for the test duration without crashing and without significant performance degradation.				
Results	Empty for this report				

Test ID	NF006	Category	Compatibility	Severity	Major
Objective	Confirm that the application works consistently across various devices and OS versions.				
Steps	1. Select a representative set of devices (different models, screen sizes, OS versions such as Android 8.0+ and iOS 12+). 2. Execute a suite of core functionalities (login, recipe creation, multimedia upload, voice recognition) on each device. 3. Document any inconsistencies or errors encountered on specific devices or OS versions.				
Expected results	● The application must exhibit consistent behavior and performance across all tested devices and OS versions.				
Results	Empty for this report				

Test ID	NF007	Category	Compliance	Severity	Critical
Objective	Ensure that the application fully complies with GDPR and other relevant data protection regulations.				
Steps	1. Review data collection, storage, and processing procedures as implemented in the application. 2. Verify that all user data is handled in accordance with GDPR requirements (e.g., user consent, data minimization, right to access/erase). 3. Check that privacy policies and user agreements are clearly presented and accessible within the app. 4. Document any non-compliant processes or potential privacy concerns.				
Expected results	● The application should comply with all relevant data protection regulations, with no data handling process falling outside these guidelines.				

Results	Empty for this report

Test ID	NF008	Category	Supportability	Severity	Major
Objective	Confirm that error logging and monitoring systems capture all critical events to facilitate prompt troubleshooting and maintenance.				
Steps	1. Simulate common error scenarios (network disconnection, invalid input, failed payment transaction). 2. Review the application's error logs and monitoring dashboard for each simulated scenario. 3. Check that detailed error messages and event logs include timestamps, error codes, and relevant contextual information. 4. Evaluate if the logs provide sufficient information for troubleshooting.				
Expected results	• All errors should be logged with detailed and actionable information, and critical security events should trigger immediate alerts.				
Results	Empty for this report				

Test ID	NF009	Category	Performance	Severity	Major
Objective	Ensure efficient resource usage (memory, CPU, battery) during typical application operations.				
Steps	1. Monitor resource usage while performing a series of core functionalities (launch, multimedia upload, voice commands, navigation). 2. Use device profiling tools to record CPU, memory, and battery consumption over time. 3. Compare these metrics against acceptable baseline thresholds for similar applications.				

	4. Identify any operations that cause abnormal resource spikes.
Expected results	• Resource usage should remain within acceptable limits with no significant drain on battery life or device performance.
Results	Empty for this report

Test ID	NF010	Category	Reliability	Severity	Critical
Objective	Verify that the system can reliably back up data and restore it accurately in case of system failure.				
Steps	1. Simulate a system or database failure scenario. 2. Initiate the backup restoration process using the most recent backup. 3. Verify that all user data, recipes, and settings are restored correctly and completely. 4. Repeat the test to ensure consistent recovery performance.				
Expected results	• The system must restore all data accurately and resume normal operation with minimal downtime.				
Results	Empty for this report				

Test ID	NF011	Category	Performance	Severity	Critical
Objective	Stress test the application under simulated high concurrent user load.				
Steps	1. Simulate multiple concurrent user sessions (e.g., 500 users) using a stress tool. 2. Monitor system response and overall performance metrics.				

	3. Record any delays, crashes, or performance degradations.
Expected results	• The system should maintain acceptable performance and responsiveness with no critical failures under high load.
Results	Empty for this report

Test ID	NF012	Category	Security	Severity	Major
Objective	Validate that user sessions timeout correctly after inactivity.				
Steps	1. Log in to the application. 2. Leave the session inactive for the defined timeout period. 3. Attempt to perform any user action after timeout. 4. Verify that the system prompts for re-authentication.				
Expected results	• The session expires as specified, and the user must log in again.				
Results	Empty for this report				

Test ID	NF013	Category	Performance	Severity	Major
Objective	Ensure the system enforces API rate limits to prevent abuse.				
Steps	1. Rapidly send a high volume of API requests to the server. 2. Check if requests beyond the limit are rejected or throttled.				

Expected results	• The system should enforce rate limits, rejecting excessive requests with proper error messages.
Results	Empty for this report

Test ID	NF014	Category	Data Integrity	Severity	Critical
Objective	Verify data consistency across different modules after simultaneous operations.				
Steps	1. Perform data updates in one module (e.g., user profile update). 2. Trigger related operations in connected modules (e.g., recipe attribution updates). 3. Retrieve data from each module concurrently. 4. Compare the data to ensure uniformity. 5. Document any discrepancies observed.				
Expected results	• Data should be consistent and accurately synchronized across all modules.				
Results	Empty for this report				

Test ID	NF015	Category	Supportability	Severity	Major
Objective	Assess the performance of error logging under normal and high-error conditions.				
Steps	1. Simulate both normal operations and error conditions (e.g., network errors, invalid inputs). 2. Monitor the logging system for completeness and timeliness. 3. Validate that logs contain accurate timestamps and error details.				

Expected results	• All errors are logged efficiently and comprehensively for troubleshooting.
Results	Empty for this report

Test ID	NF016	Category	Performance	Severity	Major
Objective	Evaluate the UI responsiveness under slow network conditions.				
Steps	1. Simulate slow network speeds using network throttling tools. 2. Navigate through the application's UI and perform key actions. 3. Record any delays or unresponsive UI elements. 4. Analyze the impact on user experience.				
Expected results	• Despite slower network speeds, the UI should remain responsive without significant delays.				
Results	Empty for this report				

Test ID	NF017	Category	Stability	Severity	Major
Objective	Monitor device temperature to check for excessive heating during prolonged use.				
Steps	1. Operate the application continuously for an extended period (e.g., 6 hours). 2. Monitor device temperature and performance metrics.				
Expected results	• The device should not overheat, and performance should remain stable throughout the test duration.				

Results	Empty for this report

Test ID	NF018	Category	Reliability	Severity	Critical
Objective	Detect potential memory leaks during continuous application operation.				
Steps	1. Run the application continuously while executing various functionalities repeatedly. 2. Monitor memory usage using profiling tools. 3. Note any gradual increase in memory usage over time. 4. Trigger garbage collection manually if possible and observe changes. 5. Compare memory usage against baseline metrics.				
Expected results	● Memory usage should remain stable over time without significant increases indicative of leaks.				
Results	Empty for this report				

Test ID	NF019	Category	Perfomance	Severity	Major
Objective	Measure the battery consumption of the application under continuous operation.				
Steps	1. Use a standardized test device fully charged. 2. Run the application continuously while executing typical user interactions. 3. Record battery level at regular intervals.				

Expected results	• Battery consumption should align with acceptable thresholds for similar applications, ensuring prolonged usability.
Results	Empty for this report

Test ID	NF020	Category	Reliability	Severity	Critical
Objective	Validate system recovery time after a simulated failure.				
Steps	1. Intentionally simulate a critical component failure (e.g., server crash). 2. Initiate the recovery process manually or automatically. 3. Measure the downtime duration until normal operation resumes. 4. Verify that recovery procedures restore full functionality.				
Expected results	• The system must recover and resume operations within the predetermined acceptable downtime.				
Results	Empty for this report				

Test ID	NF021	Category	Integration	Severity	Major
Objective	Verify the reliability of third-party integrations (e.g., payment gateways, voice APIs).				
Steps	1. Execute key functionalities involving third-party services (e.g., payment processing, voice command processing). 2. Document any discrepancies or failures in the integration.				

Expected results	• All third-party integrations should work seamlessly and reliably, with error handling in place for failures.
Results	Empty for this report

Test ID	NF022	Category	Performance	Severity	Major
Objective	Assess the response reliability of cloud services under varying loads.				
Steps	1. Perform multiple cloud-based operations (e.g., data retrieval, multimedia upload) under normal and high-load conditions. 2. Record response times and error rates. 3. Compare results with baseline expectations.				
Expected results	• Cloud services should consistently respond within acceptable limits, even under high-load scenarios.				
Results	Empty for this report				

Test ID	NF023	Category	Usability	Severity	Major
Objective	Verify that the application is accessible via screen readers and other assistive technologies.				
Steps	1. Enable a screen reader on a supported device. 2. Navigate through the application using only keyboard shortcuts and screen reader commands. 3. Check that all UI elements are properly labeled and accessible.				

	4. Document any areas that fail to provide adequate accessibility cues.
Expected results	● The application should be fully accessible, with all interactive elements properly announced by the screen reader.
Results	Empty for this report

Test ID	NF024	Category	Internationalization	Severity	Major
Objective	Ensure proper localization and internationalization support for multiple languages.				
Steps	1. Change the device language settings to several different languages. 2. Launch the application and navigate through various sections. 3. Verify that all text elements, date formats, and currency symbols are correctly localized. 4. Check for any untranslated strings or layout issues. 5. Record observations and compare with expected localized content.				
Expected results	● The application must accurately display localized content for all supported languages without layout or formatting issues.				
Results	Empty for this report				

Test ID	NF025	Category	Reliability	Severity	Critical
Objective	Verify that the system recovers gracefully from interrupted transactions.				

Steps	1. Initiate a transaction (e.g., payment or subscription purchase) and forcibly interrupt the process (e.g., disconnect network). 2. Resume the transaction after reconnection and observe the system's behavior.
Expected results	• The application should properly handle the interruption, allowing the transaction to be resumed or safely rolled back without data corruption.
Results	Empty for this report

Test ID	NF026	Category	Security	Severity	Critical
Objective	Verify the system's resistance to common injection attacks and ensure robust input sanitization.				
Steps	1. Attempt SQL injection attacks on all input fields using common payloads. 2. Inject cross-site scripting (XSS) payloads into comment and feedback fields. 3. Monitor the system's responses to confirm that malicious inputs are sanitized and rejected.				
Expected results	• The system must neutralize any malicious payloads, ensuring no unauthorized data access or execution of injected scripts.				
Results	Empty for this report				

Test ID	NF027	Category	Compliance	Severity	Major
Objective	Validate the application's adherence to international accessibility and usability standards.				

Steps	1. Execute manual accessibility testing using standard evaluation tools. 2. Conduct user testing sessions with participants who use assistive technologies. 3. Check for compliance with WCAG 2.1 Level AA guidelines. 4. Document any non-compliance issues and propose corrective actions.
Expected results	● The application should meet or exceed WCAG 2.1 Level AA standards, ensuring full accessibility and usability for all users.
Results	Empty for this report

6. Consideration of Various Factors in Engineering Design

6.1. Constraints

6.1.1. Implementation Constraints

- MongoDB and Firebase are used for the effective handling of relations and their feature-rich design.
- A custom engine for recipe input will be developed to make it easier for users to use the core features of the application.
- React Native is used as the primary front-end library for the application.
- The Google Cloud services for speech-to-text and text-to-speech will be used for the implementation of voice-recognition and vocalized recipe instructions for better interaction and engagement.
- Cloud hosting via Amazon AWS is used for a seamless production/development cycle and UX.
- Cross-platform compatibility is implemented by developing native apps using React Native.
- Git is used as version control.
- Jira is used for project management.

6.1.2. Economic Constraints

- We will use the Spoonacular API to retrieve nutritional data of various foods and recipes. The API has an "academic access" that allows up to 5000 API requests per day and is 10$/month.
- The Google Cloud service for speech-to-text costs $0.024 per minute
- The Google Cloud service for text-to-speech is free for the first 1 million characters per month and an additional 0.0000016$ per character (16$ per 1 million characters) after the initial million.
- The databases, libraries, and CI/CD tools that we will use are all free.

- Our preferred cloud hosting service provider, Amazon, has a free tier available for students.

6.1.3. Health Constraints

- Recipes will all have a comprehensive ingredient list with appropriate allergen labels to prevent unforeseeable allergic reactions.
- Possible substitutions for allergenic foods will be provided.
- Accurate nutritional values of recipes will be provided so that users can make informed choices.

6.1.4. Safety Constraints

- Safety tips for dangerous tasks such as cutting, heating up an oven, etc. will be implemented.

6.1.5. Ethical Constraints

- Personal user data such as dietary preference, available ingredients, and more sensitive information such as e-mail addresses, credit card information etc. will be securely stored and encrypted to ensure protection of privacy.
- The aforementioned user data will not be shared with other third-party applications without users' consent.
- Advertising and recommendations within the app will follow standard codes of ethics as to not be manipulative or misleading
- To ensure the integrity of the project, the "Professional Responsibilities" section under the ACM Code of Ethics and Professional Conduct [2] will be followed by all members.
- The subscription service within the application will be clear and transparent about all information, including pricing and additional benefits received. Users will be notified prior to a possible auto-renewal charge, and canceling a subscription will be made easy.

6.2. Standards

- All developers will follow the ACM Code of Ethics and Professional Conduct [2] to ensure a professional and responsible lifecycle for the project.
- Code quality will adhere to the SOLID principles [3] 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 [4].
- The payment process for the subscription system will comply with Payment Card Industry Data Security Standards [5] to ensure secure transactions.
- UML 2.5.1 will be used for modeling.
- IEEE format will be used for citations in the documents.

7. Teamwork Details

7.1. Contributing and Functioning Effectively on the Team

From the start of the project, our team placed significant importance on the fair contribution of team members. All of us contributed significantly to improving Yes Chef!

Mert Yamalı worked on the backend side of the project and researched and communicated with relevant services for us to use relevant API's such as Spoonacular. Moreover, the implementation of Firebase in our project was his responsibility.

Serhan Turan also worked on the backend, designing the database and structure on MongoDB for our project. Text-to-speech features are also his responsibility.

Ulaş Keskin worked on the front end of our project, implementing social media-like features such as profile page, feed, etc.

Ismail Barış Sunar worked on designing the game engine as well as the recipe viewing experience.

Mert Terkuran worked on designing the game engine, the recipe viewing experience, and the overall design of the application.

7.2. Helping Create a Collaborative and Inclusive Environment

To create a collaborative environment, we took some steps to ensure that each person took on responsibilities that they were interested in. This meant that they would be more incentivized to improve their part in the project and come up with improvements. We use communication channels such as WhatsApp, zoom, jira, etc., to keep in touch and track progress, and each member has their say in the project's development.

7.3. Taking the Lead Role and Sharing Leadership on the Team

Because we assign parts of our project to interested members, all of us are leaders in our domains.

Mert Yamalı took the initiative and set up the Jira Account and is responsible for assigning and tracking deadlines and deliverables and the backend design due to his experience.

Serhan Turan is responsible for documentation and monitoring the project deliverables.

Ulaş Keskin took a lead role in the social media aspect of our project and is responsible for designing the user interaction aspect of our application.

İsmail Barış Sunar took a lead role in designing the game engine and made significant decisions as to the UX of the engine itself.

Mert Terkuran took a lead role in designing the application, from its features to its architecture, framework, and design in almost every development aspect.

8. Glossary

ED: Eating Disorder.
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.
UX: User experience.
CI/CD: Continuous Integration/Continuous Delivery.

9. References

[1] E. Yoon, “The grocery industry confronts a new problem: Only 10% of Americans love cooking,” Harvard Business Review, https://hbr.org/2017/09/the-grocery-industry-confronts-a-new-problem-only-10-of-americans-love-cooking
[2] 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
[3] “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-o f-object-oriented-design (accessed Mar. 8, 2024).
[4] “ISO/IEC 27001:2022,” ISO, https://www.iso.org/standard/27001 (accessed Mar. 8, 2025).
[5] “Standards,” PCI Security Standards Council, https://www.pcisecuritystandards.org/standards/ (accessed Mar. 8, 2025).