**­­h**

Contents

[Executive Summary 4](#_Toc167228310)

[1.0 Introduction and Overview 5](#_Toc167228311)

[Problem Statement 5](#_Toc167228312)

[Project Vision and Scope 5](#_Toc167228313)

[Requirements Summary 5](#_Toc167228314)

[Stakeholders and Their Interests 6](#_Toc167228315)

[Expected Costs and Benefits 6](#_Toc167228316)

[Business Benefits 6](#_Toc167228317)

[Business costs 7](#_Toc167228318)

[Constraints 7](#_Toc167228319)

[Recommendation 8](#_Toc167228320)

[Document Overview 8](#_Toc167228321)

[**2.0 System Initiation** 9](#_Toc167228322)

[**3.0 Feasibility Assessment** 15](#_Toc167228323)

[**Introduction** 15](#_Toc167228324)

[Technical Feasibility 15](#_Toc167228325)

[Resource Feasibility 15](#_Toc167228326)

[Schedule Feasibility 15](#_Toc167228327)

[Organizational Feasibility 16](#_Toc167228328)

[Legal and Contractual Feasibility 16](#_Toc167228329)

[Additional Comments 16](#_Toc167228330)

[**Conclusion** 16](#_Toc167228331)

[4.0 Requirements Definition 17](#_Toc167228332)

[**Introduction** 17](#_Toc167228333)

[**Data Requirements** 17](#_Toc167228334)

[Non-functional Requirements 17](#_Toc167228335)

[Functional Requirements 18](#_Toc167228336)

[5.0 Requirements Model 19](#_Toc167228337)

[Introduction 19](#_Toc167228338)

[Use Case Diagram 20](#_Toc167228339)

[Use Case Descriptions 21](#_Toc167228340)

[6.0 System Evolution 46](#_Toc167228341)

[**7.0 Conclusions and Recommendations** 47](#_Toc167228342)

[**Appendices** 47](#_Toc167228343)

[**Glossary** 48](#_Toc167228344)

[**Bibliography** 49](#_Toc167228345)

Executive Summary

In individual and commercial restaurant environments, there is an ongoing difficulty with displaying recipe steps, collecting data on dishes, and correcting recipe information. To solve this problem, a more advanced solution than pen-and-paper is necessary. The application specified in this document so that a chef can collect information about their food preparation, consolidate data over a long period, analyze individual changes that might improve the recipe, and change the recipe as needed, all within the bounds of the application. This application could also be commercially applied so that restaurant chains can leverage their large number of staff and multiple locations to collect this information. An application that optimizes recipes and cooking is necessary to create a product usable by the multitude of chain and individual restaurant businesses.

1. Introduction and Overview

## Problem Statement

When preparing food, tracking cooking metrics is challenging. Cooking metrics are precious for most kitchens, as they allow for the analysis and improvement of recipes. However, ordinary means of tracking these statistics in a kitchen environment is challenging. The paper-and-pencil method is futile, as flour dust and liquids often render paper and other writing materials invalid for long-term use. Standard note-taking applications usually break down when recording or reprinting recipes, as recipes follow a complex and unstandardized format that is difficult to edit. Overall, cooking metrics are difficult to track and are incredibly valuable for any entity that wishes to improve their cooking.

## Project Vision and Scope

This project aims to provide a service that optimizes cooking procedures through data collection. This application’s scope is limited to serving kitchens and cooks in the United States. The project aims to optimize resource usage and product quality during the cooking process.

In response to this vision, this document suggests creating a cross-platform cooking application that optimizes recipes in a casual and business context. Ideally, users will use our application as a recipe reference, quality tracker, and improvement tool. The application should walk users through a specific recipe step-by-step and allow them to record, change, and analyze their recipe. The data tracked should include, but is not limited to, cooking time, output quality, output quantity, recipe adjustments, and recipe reference photos. To ensure recipe optimization, the cooking optimizer should use this data to make graphs and suggestions or improve quality.

## Requirements Summary

* + The application must be easy to use and work across platforms. This requirement is critical because the target audience probably lacks technical ability.
  + A responsive UI (User Interface) is required to display and interact with individual recipes.
  + For larger customers, such as chain restaurants, the cooking optimizer requires a centralized database that customers can report to. This feature is for both profit motive and centralized data collection.
  + So that the users can iteratively improve their recipes, the optimizer is required to allow the cook to note down individual adjustments to recipes. This requirement is critical for recipe improvement, as chefs could note individual changes or slight variations that improve the dish, creating a recipe variation.
  + If a commercial route is chosen, an on-call support team and documentation are required to ensure smooth usage and minimal issues.
  + This application should be able to import, parse, and export recipes from individual sites, cookbooks, and notes. Recipes generally follow similar patterns and notations, so being able to extract recipes from alternate sources into our application is an essential step.
  + The application should be able to store data about yield, success rate, and average cooking time.

## Stakeholders and Their Interests

* One stakeholder is individual household cooks. Because individual chefs are interested in a high-quality product and rarely get the chance to iterate on their recipes, keeping track of past data is critical.
* This company is a primary stakeholder in this project, as it reaps the benefits of an expanding market reach and can offer its technical expertise to other stakeholders. The company will also have a stake in the overall food service industry and with individual consumers.
* A stakeholder to secure is individual restaurant chain executives. Chain restaurants constantly produce food on a commercial scale and are often looking to optimize their dish selections. Because our application can assist with aggregating and visualizing this information, these restaurants should want a stake in our product.
* One stakeholder to avoid conflict with is the FDA (Food and Drug Administration.) As our application intends to give step-by-step instructions on food preparation, health and safety protocols come into play, opening us to liability issues. If our application provides instructions for dishes that include risky ingredients or are dangerous to prepare, there could be issues.
* An unacknowledged stakeholder in this project is application distribution centers. Microsoft, Google, and Apple stores have strict criteria regarding published applications that are challenging to comply with.
* Another set of stakeholders are individual recipe holders and hosting sites. Cookbooks and the recipes in them could fall under copyright protection, and the terms of service of cooking sites often explicitly forbid mass web scrapers and reverse-engineering the site. (*Terms of Service - Dotdash Meredith*, n.d.) However, the application’s goal isn’t to infringe or republish individual recipes but to import and optimize them.

## Expected Costs and Benefits

## Business Benefits

One intangible business benefit from a cooking optimizer is restaurant dependency on our project. Because all relevant data is in our centralized database, restaurants continuously rely on our application for their optimization needs.

A great intangible benefit is the gifted advantage to individual cooks. For scientifically minded chefs, recording every detail about their cooking and what makes one recipe attempt better than the other is difficult to quantify and record. When an individual attempts to improve their cooking, an application to give instructions, record variations, and recognize beneficial patterns is a powerful tool.

Our primary monetary benefit is adoption by food service establishments. When they grow to a specific size, monetizing per location regularly rather than per business becomes possible. The overall gain should cover more than physical server and database costs; it also has the added benefit of tying our growth to the ongoing development of the restaurant industry. With our product, the restaurant will attract more customers. If there are more customers, the restaurant must expand or open another location, increasing our overall profit and creating a larger sample size of dishes to iterate on—progress then snowballs, with better food, better restaurants, and a higher profit for all parties.

## Business costs

If, beyond the minimum viable product, our company set out to maintain a corporate server and database for chain restaurants, the service would be a high ongoing cost associated with the cooking optimizer. The overall cost isn’t easy to estimate, but a short-term test with a singular restaurant should be inexpensive enough to test the overall concept. After the application expands, however, server and database costs increase dramatically.

Another cost to consider is the cost of an ongoing technical support team. Documentation and technical support consume a large part of development time, as code is useless without instructions on how to use it. Documentation is necessary when creating any application, regardless of the design.

An initial cost when creating the application is the development team tasked with designing and implementing these features. Our development teams are well-equipped and well-trained, so looking toward similarly sized projects for cost analysis shouldn’t be difficult.

## Constraints

The primary issue with our application is limited time for user input. Most restaurants, especially high-traffic chains, use a paper or digital order system. Food is prepared according to customer orders, and preparation time constraints are intense. These time constraints dramatically limit the applications' reach, as most food services don’t have the time to use an optimization application. This limits the use of our application to time-insensitive restaurants and high-end chains. However, despite this constraint on certain establishments, the application still has an excellent market. This application still applies to high-end diners, preorder food locations, bakeries, and small businesses.

A significant limitation of this service is the scope reduction to only a United States market. The prospect of international markets is exceptionally enticing for this project. Still, it is not practical to expand into these markets within the minimum viable product because of risks with food regulation and the possibility of localization errors. There is the possibility of expanding into international markets later, but for the initial launch, the target is the United States.

Another tempting possibility that we must constrain is mass data collection. There is an ongoing trend in the technology industry to aggregate and collect as much personal information as possible. With a cooking application, this is exceptionally tempting. If the application can determine the ingredients used, time spent, average time cooking, when a person cooked, and where a person cooked, the advertising possibilities would be limitless. However, attempting this raises several issues. On the legal side, the United States and the European Union are currently tightening their legislation on data collection. Coupling these with increased security risk and high upkeep costs makes exploiting customers nonviable. However, securing a small market of dedicated customers without harvesting advertising data should still provide a valid return on investment.

## Recommendation

If this document is approved, a subset of resources should be allocated to test the concept of a cooking optimization application. These resources include a small development team and basic consumer testing. A development team of 3-4 people for 2-3 months is necessary to create the minimum viable product, following this design document and associated resources. Consumer and field testing is also required to ensure the application can improve dishes in the kitchen environment. Creating a funded cooking optimization application would be the next logical step if testing is successful and the minimum viable product has potential.

## Document Overview

The rest of this document outlines the initiation, assessment, requirements, and recommendations for creating a cooking optimization application. This document intends to rationalize and specify the design details for developing an application that assists with recipes and cooking optimization. The initiation section of this document provides a broad overview and the project initiation request. Afterward, the feasibility section designates potential risks and assesses the overall feasibility of attempting this project. Then, the system requirements outline all necessary components, including data and non-functional and functional necessities for creating this application. Finally, there is the conclusion and recommendation, stating the following steps to pursue the creation of this application.

# **2.0 System Initiation**

**Project Initiation Request (PIR)**  
  
PIR-00000 *[PIR Number to be assigned by the Project Office]* Project Initiation Request (PIR) – Level2 v6.0

Project Name: Cooking Optimizer Student Name: Kyler Veenstra

**This Project Initiation Request (PIR) is to be completed for all requests expected to require over 40 hours of effort or over 4 weeks of total duration. For larger requests requiring over 40 person-days or estimated project costs greater than $5,000, this template is used to assess the product's feasibility and get approval to scope and plan the proposed project.**

**If approved, the Level 2 template (System Proposal: Part 1 and Part 2) must be completed.**

**0. General Project Information**

|  |  |
| --- | --- |
| **Project Name:** | Cooking optimizer |
| **Two Sentence Request Description:** | Statistical and organizational tool for prominent restaurants and individual cooks. Record recipes, cooking success rate, ingredients used, average yield, and other details. |
| **Requested Launch Date(s):** | May 20th 2024 |
| **Department(s) Affected By Project:** | IT department, Security, Billing, UI Design. |
| **Project's Customers:** | Individual cooks or high-end restaurants that desire greater business efficiency. |
| **Date Request Submitted:** | 2024-04-16 |

1. **Project Sponsor and Manager**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Project Sponsor** | |  | **Business Project Manager & Requestor** | |
| **Name:** | Andy Cameron |  | **Name:** | Kyler Veenstra |
| **Title:** | Professor |  | **Title:** | Student |
| **Department:** | Computer Science - SPU |  | **Department:** | Computer Science - SPU |
| **eMail:** | acameron@spu.edu |  | **eMail:** | veenstrak@spu.edu |

1. **Business Problem or Opportunity: The motivation for this request**

*Describe the problem or opportunity that you would like to solve. Include a simple, high-level description of this request's business problems or opportunities. Focus on the problem or opportunity, not the solution. Be sure to include any date or deadline-related dependencies or needs related to the project.*

| Individual cooks and high-end restaurants must work with expensive and scarce resources regularly. For these groups, high-end meals require a significant investment of time, resources, energy, and practice. One way to reduce the risk of failure is by carefully tracking the time, ingredients, revisions, average yield, and other details during the cooking process. However, keeping track of these with a notebook or spreadsheet is difficult in a kitchen environment. Because of this, an application that could keep track of these details could be beneficial. |
| --- |

1. **Justification, Impact, and Importance**

*What is the financial impact and justification for this request? How will the investment of time, resources, and capital be returned to our company? (Please note any contractual or regulatory requirements associated with the request. If you have an NPV, IRR, or ROI calculation, please provide the link(s) in this section.)*

**Assumptions**

|  |
| --- |
| * *Include at least two. Add more rows to each table as needed.* |
| * Product testing is possible with a simple, localized application. If there’s a high adoption rate, a complete project targeting multiple restaurant chains, small businesses, and chefs is worth pursuing. The project can also be removed without significant risk if adoption declines, or the systems prove inefficient. |
| * Since company-owned servers could store critical information, companies that use the product become indefinitely dependent on our product to store information. Recipe and kitchen data are currently unordered, difficult to aggregate, and hard to reference. This disorder makes it difficult to switch to competitors or use other systems if they emerge. |

**Competitive Landscape / Context**

|  |
| --- |
| * *Include at least two* |
| * Household kitchens–Normal kitchens that wish to better analyze their cooking skill or recipe yield could easily use our application. Users could also share a certain number of their personal recipes for free on the platform. |
| * Small Businesses–Small businesses needing to ration their inventory, optimize their recipes and yield, or measure their cooking time could use our application to improve their business. |
| * Restaurant chains–Chain restaurants could, on a larger scale, track their average production rate, ingredients used, and caloric content. To network outside their individual chain, they could pay our business to aggregate and store that data in a database, making it easy for high-level executives to refer to when adjustments are necessary. |
| * Luxury Restaurants–When a customer desires a high-quality product, and expensive ingredients are necessary, statistics and records can reduce the overall risk. |
| * Existing restaurant management software–Most software is concerned with keeping track of orders and delivering to specific customers. As such, we can see this kind of software as a collaborative partner rather than a competitor. |

**Tangible Return, Opportunity, or Value One Time Ongoing**

|  |  |  |
| --- | --- | --- |
| * *Include at least two. Estimate the best you can.* | $ 0 | $ 0 |
| * Restaurant dependency on our product | $ 0 | $ 15/month, per location |
| * Customers paying to display extra recipes | $10/Customer | $0 |

**Intangible Benefits Impact or Value**

|  |  |
| --- | --- |
| * *Include at least two. Estimate the best you can.* | $ 0 |
| * Aggregated data from a variety of users |  |
| * Connections in the food industry |  |
| * Contact at every point along the food preparation process |  |

1. **Product Requirements**

*The Project team will gather detailed requirements once the project is approved. Use this section to articulate the critical solution components to help scope the project's size and complexity. Do not describe how the solution will be implemented; instead, only list the functionality or results you expect to receive when the product is complete/delivered.*

* 1. **Must Haves**

|  |
| --- |
| * + 1. *Include at least two. Add more rows to each table as needed.* |
| * + 1. Import recipes |
| * + 1. This project requires an extensive database of recipes, ingredients, and food selections. |
| * + 1. Desktop / mobile / web application |
| * + 1. A centralized and tested paid server needs to be present for large chains. |
| * + 1. Export recipes |
| * + 1. Step-by-step walkthrough of several recipes |
| * + 1. Responsive cooking time tracker |
| * + 1. Yield tracker. |
| * + 1. Success rate tracker |

* 1. **Could Haves** (Nice to Haves)

|  |
| --- |
| * + 1. *Include at least two.* |
| * + 1. Common substitutes for ingredients listed |
| * + 1. Inventory control / inventory control integration |
| * + 1. Embedded cooking timer |
| * + 1. Share a few personal recipes for free. |

* 1. **Won't Haves** (Don't Do's, aka Out of Scope)

|  |
| --- |
| * + 1. *Include at least two* |
| * + 1. For monetization reasons, this program will not allow users to set up self-hosted servers for data management. |
| * + 1. This system will not integrate with employee management / payroll to remove payment compliance issues. |
| * + 1. This product should not initially integrate with software that purchases ingredients. |
| * + 1. Because of safety issues, this application should not control autonomous appliances. |
| * + 1. The program, at least initially, should not work internationally. |

1. **Project Costs (Operating and Capital: Onetime and Recurring) [Optional]**

*This section is typically fleshed out after the requestor has submitted a PIR and received approval for the initial scoping effort. It captures the effort estimates, capital expenditures, and other costs associated with performing this work and creating the product/solution. If the submitter has thoughts or estimates on what these costs are or suggestions on how they might be estimated, please include those here. Add brief descriptions as needed.* ***Include at least 2 comments on your thinking around these items, even if you don't have specifics yet.***

**Labor Costs**

|  |  |  |  |
| --- | --- | --- | --- |
| **Type** | **Team(s) Affected** | **Low (hrs)** | **High (hrs)** |
| Analysis & Design |  | 0 | 0 |
| Development |  | 0 | 0 |
| Testing and Quality Assurance |  | 0 | 0 |
| Systems Integration |  | 0 | 0 |
| Deployment |  | 0 | 0 |
| Support and Maintenance |  | 0 | 0 |
| Sales and Marketing |  | 0 | 0 |
| **Total** |  | 0 | 0 |

| Comments:Conceptually, this has a simple implementation. The difficulty will be in making a responsive and helpful User Interface that can suit the needs of individuals and companies. The other significant cost is successful advertising, as there will be no adoption without at least a small group of dedicated individuals who generate word-of-mouth advertising. |
| --- |

**Capital Costs** (Equipment, Software, Licenses, …)

|  |  |  |
| --- | --- | --- |
| **Description** | **Quantity** | **Cost ($)** |
| *Item 1* |  | $ 0 |
| *Item 2* |  | $ 0 |
| **Total** |  | $ 0 |

| Comments:The main cost for this product is initial server creation. Software like integrated development environments and user interface design applications we should already have on hand. |
| --- |

**Maintenance Costs** (Costs after the product is live)

|  |  |  |
| --- | --- | --- |
| **Type** | **Hours / Month Low** | **Hours / Month High** |
| System / User Support | 0 | 0 |
| Business / Process Support | 0 | 0 |
| **Total Support & Maintenance** | 0 | 0 |

| Comments:The major cost after publication is server upkeep and security maintenance. Support will also be an additional cost. |
| --- |

# **3.0 Feasibility Assessment**

**Introduction**

The following section is a feasibility analysis for the Cooking Optimizer project. This proposal measures this project's overall feasibility and attempts to give a rough estimate of how feasible the project is overall. All feasibility is rated on the following scale from greatest to least: ideal, highly feasible, feasible, low feasibility, and not feasible. The risk will be quantified by high, medium, and low risk.

## Technical Feasibility

The cooking optimizer is feasible from a pure development standpoint, although there is a medium risk.

*Overall, development carries a low risk.*

* The company has experience with technical projects and database management. This prior experience should make project development more straightforward.
* The solution as described requires minimal innovations or advances and can leverage standard data storage and design patterns. This fact should reduce research costs and make development more linear than your average application.
* A system needs to be developed to import and store recipes. Recipes are a volatile data format with no overarching classification, and creating a standard classification will be difficult.

*The size of the application holds a medium risk.*

* In terms of scale, the cooking optimizer is a relatively medium-sized application compared to other projects.

*Developing a proprietary application has an overall low risk.*

* Our company has experience developing proprietary applications on multiple occasions.
* The company knows the overall structural changes that need to be made when creating a proprietary closed-source application.

*Securing the application holds a high overall risk.*

* Our division currently has no experience with overall access control and management.
* Individuals using our application will need accounts and logins, making us responsible for security standards.

*Cross-platform compatibility holds a low overall risk.*

* This company has experience developing applications for standard operating systems and devices. Therefore, it should be reasonably feasible to create a cross-platform application.

## Resource Feasibility

The cooking optimizer has a low overall feasibility and high risk from a resource standpoint.

*The lack of on-hand resources represents a high overall risk.*

* The company, as of current, has no available servers or databases allocated for a new venture. As this is a significant part of the profit motive, it puts the application in a risky position.
* No on-hand staff are available for technical support, and the company has minimal experience doing as much. Hiring such a team is expensive but straightforward.

## Schedule Feasibility

*The feasibility of the schedule is currently ideal and carries a low overall risk.*

* As of now, no companies or contracts rely on this project.
* If the company entered official contracts, the application would have a strict deadline, changing the overall feasibility and increasing risk.

## Organizational Feasibility

*The organizational structure currently carries ideal feasibility but a medium risk.*

* From an organizational perspective, the project has a medium risk. This company is well-organized to handle such an experiment and has the teams and staff on hand to try such a project.
* Agile frameworks are well-established and would be excellent for tackling this project.
* It is organizationally challenging to cut our losses if risk or development is too difficult.

## Legal and Contractual Feasibility

*From a legal perspective, the project is feasible but carries medium risk.*

* Currently, we are not contractually obligated to develop or create this project. It is therefore possible to cut the project if the risk or cost proves too great.
* Because the optimizer technically maintains instructions on preparing food, our application must conform to national food preparation standards, and advice from the FDA might be required for commercial usage.
* The application might have difficulty with copyright and importing, requiring legal consultation on the overall boundaries of the application.

## Additional Comments

* A crude prototype application is necessary to ensure that this service would be in demand and provides adequate improvements.

**Conclusion**

In its current state, this is a highly feasible project, but it carries a medium risk. To manage the overall risk of this project, rigorous testing, iteration, and review will have to be done to ensure a quality product.

4.0 Requirements Definition

**Introduction**

This document section outlines the complete list of required features and functionality. It overviews several requirements, including the data, functional, and non-functional. The data requirements entail the data that must be stored to make the application practical. The functional and non-functional requirements outline the requirements of application uses, as well as external requirements. Overall, this segment aims to set the baseline requirements for a working product.

**Data Requirements**

1. Recipe storage
   1. The ingredients list of the recipe needs to be stored for reference
   2. The complete original instructions for the recipe need to be preserved.
   3. Partitioning markers between individual recipe instructions need to be made, so the chef can work on multiple parts of the recipe simultaneously.
   4. In case of data corruption and for attribution reasons, the recipe’s source needs to be stored.
   5. A visual identifier, such as a title or image, must be available.
   6. Integrated timer locations need to be stored in the recipe data.
   7. Component and related recipes for items need to be referenced.
   8. Individual recipe notes need to be stored.
      1. The ability to add warning points where the recipe is easy to mess up and notes on avoiding failures is required.
      2. There can be notes in positions where the recipe is viewer-dependent and needs fine adjustment.
2. Recipe variations
   1. To properly prepare for a variation, the application should note changes in recipe ingredients.
   2. The application should be able to note changes in recipe preparation.
   3. Expected changes to the taste should also be recorded.
   4. Preparation time for the recipe should be noted.
3. Recipe preparation data
   1. The recipe’s fail/success rate should be stored to consider opportunity cost.
   2. Yield per recipe is a necessary factor to keep track of.
   3. Storing taste ratings per preparation allows the chef to consider his and other’s tastes.
   4. Logging the differences between preparations will help chefs track when minute changes improve the recipe.
   5. Average preparation time allows a chef to consider the time investment when preparing a meal.
   6. To create visible graphs, record the date the recipe was prepared.
   7. Extra notes are needed for later consideration of the data to determine if the creation process was a statistical outlier.
4. Business data
   1. For businesses, we need to store an individual chef's entire local recipe database and sync it back and forth with the overarching server.
   2. We must store information relevant to company keys, such as hashes and license codes.
   3. The application must store information about the restaurant's location and associated keys to verify its purchase and rightful use.

## Non-functional Requirements

1. Input
   1. Input errors should be removed from the application by design, making entering invalid information very difficult.
   2. Inputting information should be a simple, non-time-consuming process.
2. Installation, maintenance, and data management
   1. The software must be portable and easy to share between devices. Special permissions and rights should be unnecessary.
   2. Hardware requirements for running the product must be minimal.
   3. In return for payment, our business is to do system maintenance, maintain backups, and handle issues.
   4. Backups should be created at relevant intervals and be stored on company servers.
   5. The cooking optimizer must have a high overall uptime based on business hours. Maintenance should only be scheduled during off-hours.
   6. Installation should be relatively painless.
   7. The application must work on mobile and desktop devices.
3. Administration
   1. Purchasers of our business product should be able to access notes and metrics from employees and visualize them.
   2. This tool should be able to report individual employee usage back to the purchaser.
4. Support
   1. Documentation is necessary for upper management. Other instructional tutorials should be available to end users.

An integrated, optional tutorial would be beneficial to teach end-users how to use the application.

## Functional Requirements

1. Basic Functionality
   1. A primary function of the application for restaurants and individual chefs is a recipe walkthrough feature. This feature allows the user, on their device, to step through individual instructions for preparing a dish.
   2. Another required function is to import individual recipes from either books or websites.
   3. Individual cooks should be able to create a “variation” of the recipe based on their personal experience and taste. This variation could include something as simple as instructions for cooking at higher altitudes or as drastic as removing key ingredients for allergen reasons.
   4. Chefs should be able to record when a recipe goes awry, what the problem might have been, and if it tasted better or worse.
   5. Users should be able to record their overall success rate and relevant recipe yield.
   6. The application should keep track of completion time from start to finish.
2. Optional but beneficial functionality.
   1. A cooking timer built into recipes is a handy function to accompany recipe walkthroughs.
   2. A helpful function is an easy substitution and measurement conversion.
   3. The user would benefit from the ability to add individual photos for reference.
3. Data information storage
   1. There is a requirement to store recipe data. This data includes a list of ingredients, pictures, and instructions, with some variations based on taste and conditionals.
   2. The application must store adjustment information, such as taste ratings and recipe variations, both locally and in a cloud database.
4. Data visualization requirements
   1. The user must be able to visualize their overall success rate, production rate, and best work. This visualization is so that the user can optimize their overall cooking strategies.
5. Access control
   1. Accounts should be secured behind a username and password for corporate users.
   2. Unauthorized or individual users should not have any access to databases.
   3. Each corporate customer should have their recipes “siloed,” unavailable for query by other companies without authorization.

5.0 Requirements Model

## Introduction

The following documents show a cooking optimization application's requirements and proposed functionality. Requirements modelling in this document includes a use case diagram and 20 use case descriptions. The use case diagram shows which functionality and usage points align with individual “actors” using our product. Each use case is numbered 1-20 in an arbitrary order. As per the key, blue circles represent food preparation, purple circles represent management use cases, green circles represent recipe modification and creation, and orange circles represent database maintenance done by the company. Each use case description corresponds to the id written in the circle. The use case descriptions specify the various aspects and edge cases that should be considered during implementation.

## Use Case Diagram

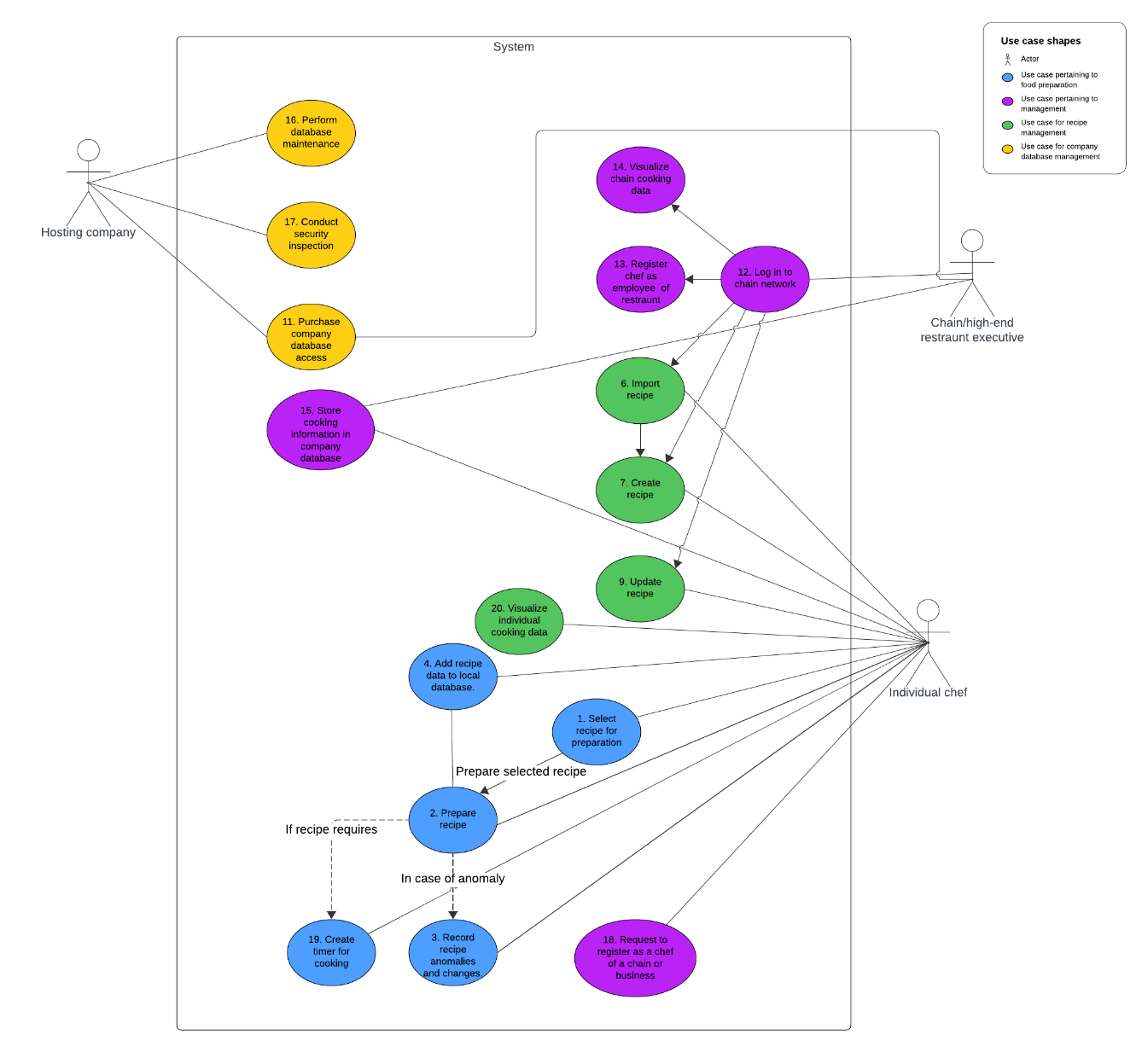


Figure 1 - Use case diagram for cooking optimization application

## Use Case Descriptions

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Select the recipe for preparation | | **ID**: 1 | **Importance**: Must have |
| **Primary Actor**: Individual chef | **Use Case Type**: | | |
| **Supporting Actors: N/A** | | | |
| **Stakeholders and Interests**:  Restaurant executive – If applicable, has a stake in ensuring dishes are created according to customer needs. | | | |
| **Brief Description**: The chef, by order or desire, selects a recipe and variation from the list of recipes to prepare. | | | |
| **Trigger**: A chef decides to prepare a recipe.  **Type** (mark one): X\_\_ External \_\_\_ Temporal | | | |
| **Relationships**:  **Association**:  Individual chefs  **Include**:  Prepare recipe  **Extend**:  Record recipe anomalies and changes  **Generalization**: | | | |
| **The Normal Flow of Events**:  The chef selects a recipe from the local database to prepare. | | | |
| **Sub-flows**:  1. Get the list of recipes from the local database.  2. Select a recipe from the list to prepare.  3. Get a list of the variations of that recipe that can be prepared.  4. Select a variation to prepare. | | | |
| **Alternate/Exceptional Flows**:  If, during step 1, the chef is registered as part of a chain,  1.1 Sync the local database with the corporate database. | | | |
| **Special Requirements:**  Information display  1. The average cooking time for each recipe must be displayed.  2. The title of the recipe should be shown.  3. A display of recipe ingredients should be available to the chef after selection.  Security  1. Authorization should be required to sync to the corporate database. | | | |
| **To do/Issues:**  **N/A** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Prepare recipe | | **ID**: 2 | **Importance**: Must Have |
| **Primary Actor**: Individual chef | **Use Case Type**: | | |
| **Supporting Actors: N/A** | | | |
| **Stakeholders and Interests**:  Recipe executives – If applicable, the executive is interested in ensuring that the dish is created, served, tracked, and improved. | | | |
| **Brief Description**: Guide the chef through the steps to prepare a recipe. | | | |
| **Trigger**: The chef begins to prepare a dish.  **Type** (mark one): \_X\_ External \_\_\_ Temporal | | | |
| **Relationships**:  **Association**:  Individual chef  **Include**:  Add recipe data to a database.  **Extend**:  Record recipe anomalies and changes.  **Generalization**: | | | |
| **The Normal Flow of Events**:  A list of ingredients and the average cooking time is presented to the chef before they begin preparing the food. Once they start, a stopwatch measures the average time needed to prepare the recipe. The application walks them through the recipe step-by-step until the dish is complete. At that time, the stopwatch stops. | | | |
| **Sub-flows**:  1. A list of ingredients and the average cooking time this recipe will take is retrieved from the local database, and shown to the chef.  2. The chef begins the recipe.  3. A stopwatch measures the average time that the recipe takes.  5. The chef is then given the recipe step-by-step, using built-in timers and instructions.  6. The chef finishes the recipe and stops the stopwatch.  7. The quality and yield of the food compared to other preparation instances is noted by the chef, as they desire. | | | |
| **Alternate/Exceptional Flows**:  If, after step 3, the chef takes a break  >3.1 Pause the stopwatch.  If after step 2,  The chef stops making the recipe.  >2.1.1 Ask the chef if they wish to register this as a failure or drop it as a non-issue.  >2.1.2 Record the recipe as a failure in the database if required. Otherwise, drop all data.  The chef makes an error or deviates from the recipe.  >2.2.1 Have a button for the chef to press, allowing them to note the deviation from the recipe.  2.2.2 Continue as before.  If, at step 7, the chef decides not to enter specific information,  7.1 Ensure that the non-recorded information is not displayed in the official statistics. | | | |
| **Special Requirements:**  Information verification   1. Ensure that blank statistics submitted to the database is not counted in official statistics. (e.g., A blank average time doesn’t bring down the overall average completion time) | | | |
| **To do/Issues:**  Integrations   1. Develop an integrated timer into the recipe. 2. Make sure that the chef can take breaks or cancel the recipe. 3. Develop a way for the chef to enter a deviation from the standard recipe. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Record recipe anomalies and changes | | **ID**: 3 | **Importance**: Must Have |
| **Primary Actor**: Individual chef | **Use Case Type**: | | |
| **Supporting Actors: N/A** | | | |
| **Stakeholders and Interests**:  Restaurant executives – If applicable, they want better recipes for their customers. | | | |
| **Brief Description**: By accident or experimentation, the chef notes a change in how the recipe is prepared. | | | |
| **Trigger**: The chef makes a mistake or experiments.  **Type** (mark one): X?\_\_ External \_\_\_ Temporal | | | |
| **Relationships**:  **Association**:  Individual chefs  **Include**:  Add recipe data to the local database.  **Extend**:  **Generalization**: | | | |
| **The Normal Flow of Events**:  The chef, either by accident or on purpose, modifies a recipe. The application allows them to note the change and how it tastes after the recipe is complete. If the chef finds it beneficial, then a variation on the recipe is added. | | | |
| **Sub-flows**:   1. The chef changes the recipe in some way. 2. The chef then hits a button to note what the change was. 3. The recipe instructions continue as usual. 4. The chef is then asked if they wish to create a new recipe variation. 5. The chef writes new instructions for the variation and the steps in which it is changed. 6. The variation is stored and sent to the local database. | | | |
| **Alternate/Exceptional Flows**:  If at step 2, the chef doesn’t have time to note the change,   1. The chef makes a “Quick note,” marking where the recipe changed but not what changed. This function is invaluable when a hot pan is on, or the chef is short on time.   If, at step 4, the chef denies making the variation,   1. All variation comments and data are dropped.   If, at step 5, the chef does not have time to record the variation,   1. The comments are saved for another time temporarily so the chef can record the information later. | | | |
| **Special Requirements:**  Temporary storage   1. The chef needs to be able to store instructions and notes temporarily so that information is not lost. | | | |
| **To do/Issues: N/A** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Add recipe data to the local database | | **ID**: 4 | **Importance**: Must have |
| **Primary Actor**: Individual chef | **Use Case Type**: | | |
| **Supporting Actors: N/A** | | | |
| **Stakeholders and Interests**:  N/A | | | |
| **Brief Description**:  The chef writes information to their local database. | | | |
| **Trigger**:  **Type** (mark one): X\_\_ External \_\_\_ Temporal | | | |
| **Relationships**:  **Association**:  Chef  **Include**:  **Extend**:  **Generalization**:  Prepare recipe. | | | |
| **The Normal Flow of Events**:  After finishing a recipe, the chef writes the data in the local database. | | | |
| **Sub-flows**:  1. The chef finishes a recipe and enters extra data about success rates and variations.  2. The chef hits “complete”  3. Data is written to the local database. | | | |
| **Alternate/Exceptional Flows**:  If, during step 3, there isn’t enough room on the device,  3.1 – Alert the user that their space has run out. Offer to either delete the oldest file or delete the current recipe information. | | | |
| **Special Requirements:**  Data storage  1. Ensure that there is a database that can handle the variations and recipe data. | | | |
| **To do/Issues:**  **N/A** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Import recipe. | | **ID**: 6 | **Importance**: Should have |
| **Primary Actor**: Individual chef, restaurant executive | **Use Case Type**: | | |
| **Supporting Actors:**  N/A | | | |
| **Stakeholders and Interests**:  Recipe sites and authors – Have a stake in ensuring that they receive proper credit and their copyright is not violated. | | | |
| **Brief Description**:  An individual chef or restaurant executive selects a recipe to add to their selection. They scan the cookbook or enter the recipe website link into the application, which converts it to our recipe format. | | | |
| **Trigger**:  **Type** (mark one): \_X\_ External \_\_\_ Temporal | | | |
| **Relationships**:  **Association**:  Restaurant executive  Individual Chef  **Include**:  Create new recipe  **Extend**:  **Generalization**: | | | |
| **The Normal Flow of Events**:  The Executive or Individual chef finds a recipe they wish to import into our application. They scan the recipe, or they submit a link for parsing. Our application finds the critical details in the recipe, including ingredients, steps, and recipe tips. The application then creates a new recipe based on that data. | | | |
| **Sub-flows**:  1. A recipe is found that might be beneficial  2. The recipe is scanned or parsed from its respective media form.  3. The recipe is converted into a new recipe for storage.  4. The new recipe is added to the database. | | | |
| **Alternate/Exceptional Flows**:  If, during step 2, the website or book is under copyright protection,  2.0 Alert the user with a warning to only copy legally available information.  2.1 Show other warnings or even deny the copy if legally prosecutable in that area. | | | |
| **Special Requirements:**  Legal  It might be necessary to ask the user which state they reside in to determine legal jurisdiction. | | | |
| **To do/Issues:**  **1.** Determine what recipes and works fall under copyright protection in areas of the United States. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Create a recipe. | | **ID**: 7 | **Importance**: Must have |
| **Primary Actor**: Chef, Restaurant Executive | **Use Case Type**: | | |
| **Supporting Actors:**  N/A | | | |
| **Stakeholders and Interests**:  FDA – Ensures restaurant safety and food preparation regulation. | | | |
| **Brief Description**:  A chef or executive adds a new recipe to the database. | | | |
| **Trigger**: A new recipe is found to add to the database.  **Type** (mark one): \_X\_ External \_\_\_ Temporal | | | |
| **Relationships**:  **Association**:  Chef  Restaurant executive  **Include**:  **Extend**:  **Generalization**:  Import recipe | | | |
| **The Normal Flow of Events**:  A chef or restaurant executive notices a new recipe for their business. The executive or chef notes the necessary steps, ingredients, tips, and other information. The recipe is then sent to a local database, which can be synced to a company database if needed. | | | |
| **Sub-flows**:  1. A recipe that would benefit the executive or chef is found or made.  2. The chef or executive specifies the recipe's name, steps, ingredients, and expected yield.  3. The recipe is stored in a local database. | | | |
| **Alternate/Exceptional Flows**:  If, during step 3, an executive or employee fills out the recipe, the local database can be synced with the company database. | | | |
| **Special Requirements:**  N/A | | | |
| **To do/Issues:**  Data   * 1. A format for recipe data needs to be created. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Update recipe. | | **ID**: 9 | **Importance**: Must have |
| **Primary Actor**: Individual Chef, Restaurant Executive | **Use Case Type**: | | |
| **Supporting Actors: N/A** | | | |
| **Stakeholders and Interests**:  FDA – Wants to ensure food safety requirements are met  Restaurant executives – want food to be of high quality  Customers – Desire high-quality and safe food. | | | |
| **Brief Description**: A recipe in the local or company database is updated. | | | |
| **Trigger**: The executive or chef decides to update a recipe.  **Type** (mark one): \_X\_ External \_\_\_ Temporal | | | |
| **Relationships**:  **Association**:  Restaurant executive  Individual chef  **Include**:  **Extend**:  **Generalization**: | | | |
| **The Normal Flow of Events**:  A recipe is selected for updating by the executive or chef. They decide to change ingredients or preparation steps and to invalidate previous statistics. They also determine if they want to keep the old version as a variant. | | | |
| **Sub-flows**:   1. The executive or chef decides to update their local or company recipe database. 2. The user in question specifies the updates they wish to make, including ingredients and preparation steps. 3. The user decides whether to remove local or company-stored details based on preparation time and yield. 4. The user then decides whether to keep the old version and its variants. 5. The database is updated accordingly. | | | |
| **Alternate/Exceptional Flows**: N/A | | | |
| **Special Requirements:**  Data   1. The application must be able to refactor recipes into variants of variants. | | | |
| **To do/Issues: N/A** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Purchase company database access | | **ID**: 11 | **Importance**: Should have |
| **Primary Actor**: Restaurant Executive | **Use Case Type**: | | |
| **Supporting Actors: N/A** | | | |
| **Stakeholders and Interests**:  Company – This use case is the main profit motive.  Chefs – Stake in the information provided. | | | |
| **Brief Description**:  The restaurant location purchases a license key for their chain, allowing internet-enabled database access. | | | |
| **Trigger**: The executives decide to purchase a recipe database service.  **Type** (mark one): \_X\_ External \_\_\_ Temporal | | | |
| **Relationships**:  **Association**:  Company  Restaurant executive  **Include**:  **Extend**:  **Generalization**: | | | |
| **The Normal Flow of Events**:  The restaurant executive for the location decides to purchase our recipe database solution. They send an appropriate payment, and we provide them with a license key for employees and a key for executives. | | | |
| **Sub-flows**:   1. The restaurant executive for a location decides to purchase our product. 2. Our company charges an appropriate sum for the service. 3. A license key to sync to our database servers is provided for the executive. 4. A license key for employees to sync with our database servers is provided, and the executive can hand it out to his staff as he sees fit. | | | |
| **Alternate/Exceptional Flows**:  If, after step 4, the executive requires assistance with setup,  4.0 Offer immediate support via a helpline, with the new customer as a priority. | | | |
| **Special Requirements:**  Helpdesk requirements   1. A helpdesk is required to assist chefs and executives using our product. 2. The helpdesk must have a priority queue for new customers – remember, they might still be able to refund the product, and we don’t want them to give up on it early.   Security requirements   1. A secure license key system needs to be in place. 2. Connections to company databases must be verified.   Tax requirements   1. As with any payment, there is tax legislation that needs to be followed. 2. There is a requirement to determine the location’s state of operation so taxes can be collected.   Privacy requirements   1. Private information about the company must be stored and collected with any transaction. | | | |
| **To do/Issues:**  Legal   1. Legal consultation is required to determine data privacy compliance and tax compliance.   Payment   1. Payment information must be secure. 2. The payment process and license keys have to be secure. 3. Determine account requirements for businesses (What information is needed) 4. Determine payment model (Frequency, amount, cost/benefit) | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Log in to the chain network | | **ID**: 12 | **Importance**: Should have |
| **Primary Actor**: Restaurant Executive | **Use Case Type**: | | |
| **Supporting Actors:** | | | |
| **Stakeholders and Interests**:  Our company – has a stake in ensuring our services are accessed securely.  Employee chefs – have a stake in ensuring that their data is appropriately entered and handled. | | | |
| **Brief Description**:  An executive logs in to the employee recipe network to view statistics or register a chef. | | | |
| **Trigger**: The restaurant executive attempts to log in.  **Type** (mark one): \_X\_ External \_\_\_ Temporal | | | |
| **Relationships**:  **Association**:  Restaurant executive  **Include**:  **Extend**:  Visualize chain cooking data  Register chef as an employee of restaurant  Import recipe  Update recipe  Create recipe  **Generalization**: | | | |
| **The Normal Flow of Events**:  The executive decides to log into our service. They enter their credentials and view the available data. | | | |
| **Sub-flows**:   1. The executive attempts to log in to the system by clicking a “business login.” 2. They enter their username and password. 3. They enter a 2FA Authentication. 4. Access is granted. | | | |
| **Alternate/Exceptional Flows**:  If, during step 2, the executive hasn’t signed in on this device before,  2.0 The executive is prompted to enter their license key.  If, during any of these steps, the executive encounters problems,  >0.0 They call the available helpdesk for support.  >0.1 Support is provided, and the problem is fixed. | | | |
| **Special Requirements:**  Helpline   1. There must be an available helpdesk for assistance with logging in.   Security   1. We must securely store emails, passwords, and 2FA secrets. | | | |
| **To do/Issues:**   1. Figure out how to store passwords securely in the database. 2. Only allow the executive access to their section of the database. They should have no access to any other customer database. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Register chef as employee of a restaurant | | **ID**: 13 | **Importance**: Should have |
| **Primary Actor**: Restaurant Executive | **Use Case Type**: | | |
| **Supporting Actors:**  Chef – Enters their company license key | | | |
| **Stakeholders and Interests**:  N/A | | | |
| **Brief Description**:  The restaurant executive adds a chef as an employee of the chain. | | | |
| **Trigger**: An executive marks the chef as part of the chain.  **Type** (mark one): \_X\_ External \_\_\_ Temporal | | | |
| **Relationships**:  **Association**:  Individual chef  **Include**:  **Extend**:  **Generalization**: | | | |
| **The Normal Flow of Events**:  The executive opens the application and views chefs who wish to register with their restaurant. They select one of their employees, and that employee’s local database is set to be synced with the company database. | | | |
| **Sub-flows**:  1. After the executive logs in, they can view chefs who request to register as company chefs and have entered the company license key.  2. The executive then selects a chef they recognize to add.  3. The chef is added, and their database is marked to be synced with the company server. | | | |
| **Alternate/Exceptional Flows**:  If at step 2, the executive cannot find any chefs to add,  2.0 Break the flow. | | | |
| **Special Requirements:**   1. Mark each person trying to register with a name or email address. | | | |
| **To do/Issues:**  **N/A** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Visualize chain cooking data. | | **ID**: 14 | **Importance**: Should have |
| **Primary Actor**: Restaurant executives | **Use Case Type**: | | |
| **Supporting Actors:**  Individual chefs – provide the data to be visualized | | | |
| **Stakeholders and Interests**: | | | |
| **Brief Description**:  The restaurant executive logs into the system, and inspects aggregated cooking data. | | | |
| **Trigger**:  **Type** (mark one): \_X\_ External \_\_\_ Temporal | | | |
| **Relationships**:  **Association**:  Individual chefs.  **Include**:  **Extend**:  **Generalization**: | | | |
| **The Normal Flow of Events**:  After logging in, the executive views charts of recipe data. They can see the average time a dish takes and how many times the dish was prepared over a period of dates. They can then issue changes to recipes or remove recipes from the menu. | | | |
| **Sub-flows**:  1. The executive views charts of recipe data.  2. The executive can then change items on the menu or change the recipe of a dish.  3. After entering price data for ingredients, cost, and salary, the executive can view the profit margins for dishes.  4. They can also view the average time per cooked unit and yield data. | | | |
| **Alternate/Exceptional Flows**:  N/A | | | |
| **Special Requirements:**  Integration   1. The employees should be able to see which items are currently on the menu. | | | |
| **To do/Issues:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Store cooking information in company database | | **ID**: 15 | **Importance**: Should have. |
| **Primary Actor**: Individual chef, company executive | **Use Case Type**: | | |
| **Supporting Actors:** | | | |
| **Stakeholders and Interests**:  Our company - Is interested in recipe data being stored reliably in our systems. | | | |
| **Brief Description**:  The company executive or individual chef syncs their modified recipe information and statistics with the company database. | | | |
| **Trigger**: The executive or individual chef finishes creating data and is logged into a “keyed” account.  **Type** (mark one): \_X\_ External \_\_\_ Temporal | | | |
| **Relationships**:  **Association**:  Restaurant executive  Individual chefs  **Include**:  **Extend**:  **Generalization**: | | | |
| **The Normal Flow of Events**:  After the chef has finished preparing a recipe, or an executive finishes making edits to recipes, the application offers to sync with our managed database. | | | |
| **Sub-flows**:   1. An executive or chef finishes editing or creating recipe data, such as variations, time to complete, or yield. 2. The information is set to the local database. 3. The information is synced to the company database, automatically entering information like time-to-complete and average yield. | | | |
| **Alternate/Exceptional Flows**:  If, during step 3, the chef came up with a variation on a recipe,   1. The variation is synced with the database but is not marked to be sent to all locations. | | | |
| **Special Requirements:** | | | |
| **To do/Issues:**  Security   1. Via the license key, the employees should have an authentication procedure to ensure they are part of the restaurant chain. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Perform database maintenance. | | **ID**: 16 | **Importance**: Should have |
| **Primary Actor**: Hosting company | **Use Case Type**: | | |
| **Supporting Actors:** | | | |
| **Stakeholders and Interests**:  Restaurant executives – Want to ensure their recipe data is safe and secure.  Individual chefs – Wants to make sure that all data is aggregated. | | | |
| **Brief Description**:  A company employee logs in to the databases to perform maintenance or upgrades. | | | |
| **Trigger**: Maintenance is required.  **Type** (mark one): \_\_\_ External \_X\_ Temporal | | | |
| **Relationships**:  **Association**:  Hosting company  **Include**:  **Extend**:  **Generalization**: | | | |
| **The Normal Flow of Events**: | | | |
| **Sub-flows**:   1. A notification will be given to customer companies of scheduled maintenance, and the planned timeframe will be communicated. 2. A backup is performed before maintenance in case of failure. 3. Maintenance begins at two hours past midnight in the UTC-8:00 time zone. This timeframe is set so that all restaurants will hopefully be empty or closed, and maintenance will not affect service. | | | |
| **Alternate/Exceptional Flows**:  If, during step 2, customers send in complaints over service or special occasions,  2.1 Negotiate a new maintenance date that does not interfere with the occasion, if possible. | | | |
| **Special Requirements: N/A** | | | |
| **To do/Issues:**  Maintenance   1. Establish a standard procedure for maintenance. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Conduct security inspection. | | **ID**: 17 | **Importance**: Should have |
| **Primary Actor**: Hosting company | **Use Case Type**: | | |
| **Supporting Actors:** | | | |
| **Stakeholders and Interests**:  Restaurant executives – Want company information secured.  External auditors – Are receiving compensation for services. | | | |
| **Brief Description**: | | | |
| **Trigger**: Regular inspection date comes.  **Type** (mark one): \_\_\_ External \_X\_ Temporal | | | |
| **Relationships**:  **Association**:  Hosting company  **Include**:  **Extend**:  **Generalization**: | | | |
| **The Normal Flow of Events**:  A monthly or annual inspection date comes around. A company auditor or external expert tests the system's security and access control and then reports the results for maintenance or recording. | | | |
| **Sub-flows**:   1. A regularly scheduled inspection date comes. 2. Inspection is conducted either by an independent or company auditor. 3. Security controls and vulnerabilities are tested. 4. Access control is tested. 5. A report is created by the auditor and stored for later action. 6. Any critical security issues are fixed. 7. Non-critical notifications are stored for later maintenance. | | | |
| **Alternate/Exceptional Flows**:  N/A | | | |
| **Special Requirements:** | | | |
| **To do/Issues:**  Security consultants   1. Figure out which consultants and testers to hire or contract.   Legal   1. Find out the best possible testing deadline and regular schedule for security inspections. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Register as a chef of a chain or business | | **ID**: 18 | **Importance**: Should have |
| **Primary Actor**: Individual chef | **Use Case Type**: | | |
| **Supporting Actors:** | | | |
| **Stakeholders and Interests**:  Restaurant executives – Need to register their employees to collect statistics.  Hosting company – Wants chefs to be able to register with restaurant locations. | | | |
| **Brief Description**:  The individual chef enters a license key for their location and attempts to register as a chef at the restaurant location. | | | |
| **Trigger**:  **Type** (mark one): \_\_X External \_\_\_ Temporal | | | |
| **Relationships**:  **Association**:  **Include**:  **Extend**:  **Generalization**: | | | |
| **The Normal Flow of Events**:  The individual chef receives a license key for the restaurant location from the restaurant executive. The chef then enters it into the application, and the application sends a request to the executive to add the chef to the organization. | | | |
| **Sub-flows**:   1. The executive gives the chef of the restaurant the “employee” license key. 2. The key registers the individual chef’s application so the chef can sync with the company server. | | | |
| **Alternate/Exceptional Flows**: | | | |
| **Special Requirements:** | | | |
| **To do/Issues:** | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Create a timer for cooking | | **ID**: 19 | **Importance**: Could have |
| **Primary Actor**: Individual chef | **Use Case Type**: | | |
| **Supporting Actors:** N/A | | | |
| **Stakeholders and Interests**:  Customer – They want their food prepared safely and their dish to remain uncharred. | | | |
| **Brief Description**:  A chef sets an embedded timer for cooking a meal. | | | |
| **Trigger**: The chef sets a timer embedded in the recipe.  **Type** (mark one): \_X\_ External \_\_\_ Temporal | | | |
| **Relationships**:  **Association**:  **Include**:  **Extend**:  **Generalization**: | | | |
| **The Normal Flow of Events**:  As the chef goes through a standard recipe, the cooking time for a particular portion of a dish is specified. The chef clicks on an embedded timer when the dish goes in to cook, then waits for the timer to elapse. | | | |
| **Sub-flows**:   1. The chef reaches a point in a recipe where a timer must be set. 2. An embedded timer is generated at that point in the recipe. 3. The chef lets the timer start. 4. Once the timer elapses, alert the chef. | | | |
| **Alternate/Exceptional Flows**:  If, during step 1, the chef does not wish for timer data to be aggregated into the overall cooking time,  1.1 The chef may select an option to subtract the cooking time from the overarching stopwatch.  If, during step 2, the embedded timer is generated incorrectly  2.1 The chef is given the option to remove or disable the timer entirely from the recipe.  If, during step 3, there are other parts of the recipe that can be completed in the time given,  3.1 Prompt the chef to move to a different subsection of the recipe simultaneously or ignore the opportunity.  If during step 4, the chef is in a noise-sensitive area, or do-not-disturb is enabled,  4.1 Consider an alternative method to alert the chef, such as text messages, vibrations, or flashing lights. | | | |
| **Special Requirements:**  Organization   1. Recipes require breakpoints between different recipe components, so the chef can jump around while a timer runs. 2. The chef must find it easy to switch between components while preparing food. | | | |
| **To do/Issues:**  Recipe data storage   1. The recipe requires a sort of “branching path” to determine when separate components are being worked on to implement parts of this functionality. | | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Use Case Name**: Visualize individual cooking data | | **ID**: 20 | **Importance**: Must have |
| **Primary Actor**: Individual chef | **Use Case Type**: | | |
| **Supporting Actors:** N/A | | | |
| **Stakeholders and Interests**: N/A | | | |
| **Brief Description**:  A chef visualizes his cooking data for individual recipes. | | | |
| **Trigger**:  **Type** (mark one): \_\_X External \_\_\_ Temporal | | | |
| **Relationships**:  **Association**:  **Include**:  **Extend**:  **Generalization**: | | | |
| **The Normal Flow of Events**:  The individual chef decides to review his personal aggregated data on individual recipes. He can view statistics per recipe. The statistics include charts of yields, average cooking time, average taste, and the general resource cost. | | | |
| **Sub-flows**:   1. The chef at the main menu presses an option to view recipe statistics and adjust. 2. The chef is shown a list of recipes and is allowed to select one. 3. After selecting the recipe, they are greeted with a list of statistics, including when the recipe was last created, the average cooking time, a graph of cooking time, and average yield. | | | |
| **Alternate/Exceptional Flows**:  If, during step #1, the chef decides to adjust the recipe,  1.0 Continue to use the case “Update recipe”. | | | |
| **Special Requirements:** | | | |
| **To do/Issues:**  Visualization   1. Consider the best method with which to visualize cooking data. | | | |

6.0 System Evolution

As the project stands now, it is worth noting that the initial minimum viable product is only required to optimize recipes for an individual user. Functionalities such as corporate database management and employee connection should be considered when creating the minimum viable product, but they are not strictly required. So, the obvious next step in the evolution would be to add this functionality.

The best-case scenario for the evolution of this product is to become a sort of source control for cooking projects. If one thinks about source control in the context of code, cooking and software iterations hold similar patterns, albeit with wildly different formats. Both require variety in hardware, with code needing variations in computers and cooks having variations in ovens. Testing for code and recipes is slow and complicated, and the widespread correction of instruction sets takes substantial effort. Food preparation, however, is at a disadvantage because iteration is much slower. Improvement suggestions can’t easily be shared, communication of alternative preparation methods is often limited by proximity, and updating a recipe for a large group is very difficult. So, why not utilize an application to share improvements? This application aims to improve cooking for an individual user: Can the same data be used to improve the recipe for all chefs?

# **7.0 Conclusions and Recommendations**

This document has outlined a full proposal for creating a cooking optimization application. It covers the complete requirements for the iteration of the cooking process, creating recipe data, and interlinking with a related business, if necessary. As it is currently designed, entering recipe data into the application and walking through a specialized and modified recipe step-by-step is straightforward. When mild changes or improvements are discovered, the chef can note and correct them. This application allows a chef to branch off and improve the original recipe. The cooking optimization application is meant to cover the full range of cases so a chef can improve, optimize, and iterate on their cooking process, allowing for every detail and change to be noted for later reference.

In response to this document, the recommendation is to create and test a basic prototype to ensure the idea has potential. If the concept holds and dishes become greatly optimized, then expansion into the restaurant sector as described is possible.

# **Appendices**

# **Glossary**

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Closed source | A term meaning that the code for an application is not released to the public; it remains the property of its owner. |
| Cross-Platform Application | An application that can work across fundamental application bases, such as Linux, Windows, and macOS. |
| Proprietary | Something that is used, produced, or marketed under exclusive legal right of the inventor or maker. (*Definition of Proprietary*, n.d., sec. 2) |
| United States Food and Drug Administration (FDA) | The federal agency responsible for the regulation of food, drugs, medical devices, and other products |
| User Interface (UI) | The presentation layer of an application – the interface that the user interacts with. |

# **Bibliography**

*Adobe photoshop drawing tool*. (n.d.). Adobe Photoshop. https://www.adobe.com/products/photoshop.html

*Blender Open-source modelling and animation tool*. (n.d.). Blender.Org. https://www.blender.org/

Commissioner, O. of the. (2024, April 29). *U. S. Food and drug administration*. FDA. https://www.fda.gov/

*Definition of proprietary*. (n.d.). Retrieved May 5, 2024, from https://www.merriam-webster.com/dictionary/proprietary

Larman, C. (n.d.). *Applying UML and patterns: An introduction to object-oriented analysis and design and iterative development* (3rd ed). Prentice Hall PTR, c2005.

*Purpose-built databases on aws | amazon web services*. (n.d.). Amazon Web Services, Inc. Retrieved May 5, 2024, from https://aws.amazon.com/products/databases/

*Terms of service—Dotdash meredith*. (n.d.). Retrieved May 5, 2024, from https://www.dotdashmeredith.com/brands-termsofservice

Veenstra, K. (n.d.). *Lecture Notes*