

# **School of Computer Science and Engineering**

# **CSE4015 - Human Computer Interaction**

**Personalized Culinary Companion: A Web Platform for Dynamic Recipe Recommendations, Ingredient-Based Filtering, and User-Generated Recipe Integration**

# Submitted by

# **Aleena 20BCE0200**

# Under the guidance of

# **Professor Swarnalatha P**

**Abstract**

The goal of this project is to develop a comprehensive recipe management system that enables users to efficiently explore, filter, and view detailed information about various dishes. The system is designed to provide a user-friendly interface for searching recipes based on ingredients, cuisines, diets, and other relevant filters. Additionally, users can contribute to the platform by submitting their own recipes, enhancing the diversity of available dishes. The project aims to create a seamless and enjoyable experience for both recipe seekers and contributors.

# **Introduction**

In today's fast-paced work environment, individuals are constantly engrossed in their professional responsibilities, and so, they aspire to swiftly cook their meals without dedicating much time to the decision-making process. The monotony of consuming the same dishes daily can lead to a sense of tedium, thus, there arises a need for a diverse array of accessible recipes. Yet, formulating such recipes presents a challenge of its own, such as modifying meals daily to cater to the diner's preferences while also prioritizing their nutritional needs. Moreover, oversight in stocking the kitchen with essential ingredients is another issue that can surface, which becomes particularly troublesome when individuals desire to prepare a quick meal.

In the realm of culinary exploration and meal preparation, the advent of digital technology has brought forth both challenges and opportunities. With the relentless pace of modern life, individuals seek convenient and efficient ways to access diverse recipes that cater to their culinary preferences, nutritional needs, and the ingredients readily available in their kitchens.

The existing landscape of culinary websites and platforms offers a multitude of recipes, but navigating through this vast sea of content can be a daunting and laborious task. Many existing platforms heavily rely on video content for recipe sharing. While videos provide visual guidance, they can be time-consuming and less efficient for those seeking quick access to ingredient lists and instructions. This has led to the need for a more accessible and text-based alternative, enabling users to quickly grasp information without the hassle of watching entire videos.

To address these challenges and enhance the user experience in culinary exploration, this project introduces a novel web application which serves as a centralized repository of a vast number of recipes. Its core functionality empowers users to effortlessly filter recipes based on the ingredients available at their disposal and their unique nutritional requirements. Additionally, the platform encourages user participation by enabling recipe contributions, fostering a vibrant community of culinary enthusiasts, and therefore promoting the tradition of recipe-sharing. With this innovative solution, we aim to streamline and elevate the culinary journey, offering users a more efficient, accessible, and comprehensive resource for discovering and sharing recipes, all in one centralized digital hub.

**Scope**

The Personalized Culinary Companion will cover the following key features and functionalities:

1. **Recipe Search and Filtering:**
   * Users can efficiently search for recipes based on ingredients, cuisines, diets, and other relevant filters.
   * Advanced filtering options, including ingredients, cuisine types, diet preferences, and sorting preferences.
2. **Recipe Display:**
   * Detailed information for each recipe, encompassing preparation time, servings, weight per serving, dish type, diet type, and cuisine.
   * In-depth nutritional information such as protein, fat, carbohydrates, glycemic index, glycemic load, and nutrition score.
3. **User Contribution:**
   * Users can actively contribute to the platform by submitting their own recipes.
   * Submitted recipes will undergo a review process before being added to the system to ensure quality and accuracy.
4. **User-Friendly Interface:** The system will have an intuitive and user-friendly interface for easy navigation and interaction.

**Objectives**

This HCI project endeavours to address several culinary challenges by creating a dynamic web application with the following objectives:

1. **Recipe Accessibility and Customization:** To provide users with a platform where they can effortlessly input their available ingredients and nutritional requirements, allowing the application to filter and suggest recipes that closely match their criteria.
2. **Effortless Recipe Discovery:** To simplify the process of finding diverse and tailored recipes, thereby minimizing the need to scour various websites for culinary inspiration.
3. **Community Engagement:** To foster an interactive community by enabling users to contribute their own recipes, thereby promoting the cherished tradition of sharing culinary wisdom.

**Requirement Specification**

**Hardware Requirements**

* The system should be accessible on standard personal computers and laptops.
* Minimum system requirements include a modern web browser and a stable internet connection.

**Software Requirements**

1. **Frontend:** Utilize HTML, CSS, and vanilla JavaScript for the user interface.
2. **Backend:** 
   1. Use Node.js for the server-side development.
   2. Implement Express.js framework.
   3. Use MongoDB database system for storing the recipes.
3. **Database:**
   1. Implement the NoSQL database to store recipes, and other relevant information.
   2. Ensure proper indexing and optimization for efficient data retrieval.
4. **Testing:**
   1. Conduct unit testing and integration testing for both frontend and backend components.
   2. Perform user acceptance testing to ensure the system meets user expectations.
5. **Scalability:** Design the system to be scalable to accommodate a growing database of recipes and user contributions.

**Methods Identified**

The development and implementation of the Personalized Culinary Companion involve a variety of methodologies and technologies to ensure a robust and user-friendly platform. The following methods have been identified for the successful execution of the project:

1. **Agile Development Methodology:** Adopting the Agile approach allows for iterative and incremental development, ensuring flexibility to accommodate changes and enhancements throughout the project lifecycle. This methodology is particularly beneficial for a dynamic and evolving system like this.
2. **User-Centered Design (UCD):** Utilizing the principles of User-Centered Design involves engaging potential users throughout the design and development phases. This approach ensures that the system is intuitive, user-friendly, and aligned with the expectations and needs of the target audience.
3. **Full-Stack Development:** Leveraging full-stack development enables the creation of a comprehensive system by addressing both frontend and backend requirements. Technologies such as Node.js, Express.js, and MongoDB on the backend, coupled with HTML, CSS, and JavaScript on the frontend, facilitate a seamless and integrated development process.
4. **Database Management:** Implementing a robust database management system, such as MongoDB, allows for efficient storage and retrieval of recipe data. The NoSQL nature of MongoDB provides flexibility in handling diverse data types and supports scalability as the system grows.
5. **Responsive Web Design (RWD):** Prioritizing Responsive Web Design ensures that the system is accessible and user-friendly across various devices and screen sizes. This method involves designing and developing the interface to adapt seamlessly to desktops, tablets, and mobile devices.
6. **Continuous Integration and Deployment (CI/CD):** Implementing CI/CD pipelines ensures automated testing, integration, and deployment processes. This method enhances the development workflow, accelerates release cycles, and maintains the stability and reliability of the system.

**Process Model**

The Personalized Culinary Companion system follows an iterative and incremental development process, aligning with the Agile methodology. The development lifecycle comprises the following key phases:

1. **Project Planning:**

* Define project scope, objectives, and requirements.
* Establish the development timeline and allocate responsibilities.
* Plan sprints, milestones, and release cycles.

1. **Requirements Gathering:**

* Engage stakeholders to gather detailed requirements.
* Identify user stories, features, and functionalities.
* Prioritize requirements based on business value.

1. **Design Phase:**

* Create wireframes and prototypes for the user interface.
* Define the system architecture, including frontend and backend components.
* Plan the database schema and relationships.

1. **Implementation:**

* Develop the frontend using HTML, CSS, and JavaScript.
* Implement backend logic using Node.js and Express.js.
* Integrate MongoDB for data storage and retrieval.

1. **Testing:**

* Conduct unit testing for individual components.
* Perform integration testing to ensure seamless communication.
* Execute system testing to validate end-to-end functionality.
* Implement automated testing for continuous integration.

1. **User Acceptance Testing (UAT):**

* Collaborate with end-users to validate system functionality.
* Gather feedback on usability, performance, and features.
* Iterate on design and functionality based on UAT results.

1. **Deployment:**

* Package the application for deployment.
* Implement CI/CD pipelines for automated deployment.
* Deploy the system to a production environment.

1. **Monitoring and Optimization:**

* Implement monitoring tools to track system performance.
* Gather analytics on user interactions and preferences.
* Optimize the system based on real-world usage.

1. **Maintenance and Updates:**

* Address bug fixes and issues promptly.
* Release updates and new features based on user feedback.
* Continuously refine and enhance the system.

# **System Architecture – Block Diagram** The system architecture of the Personalized Culinary Companion system can be represented through a block diagram, illustrating the key components and their interactions:

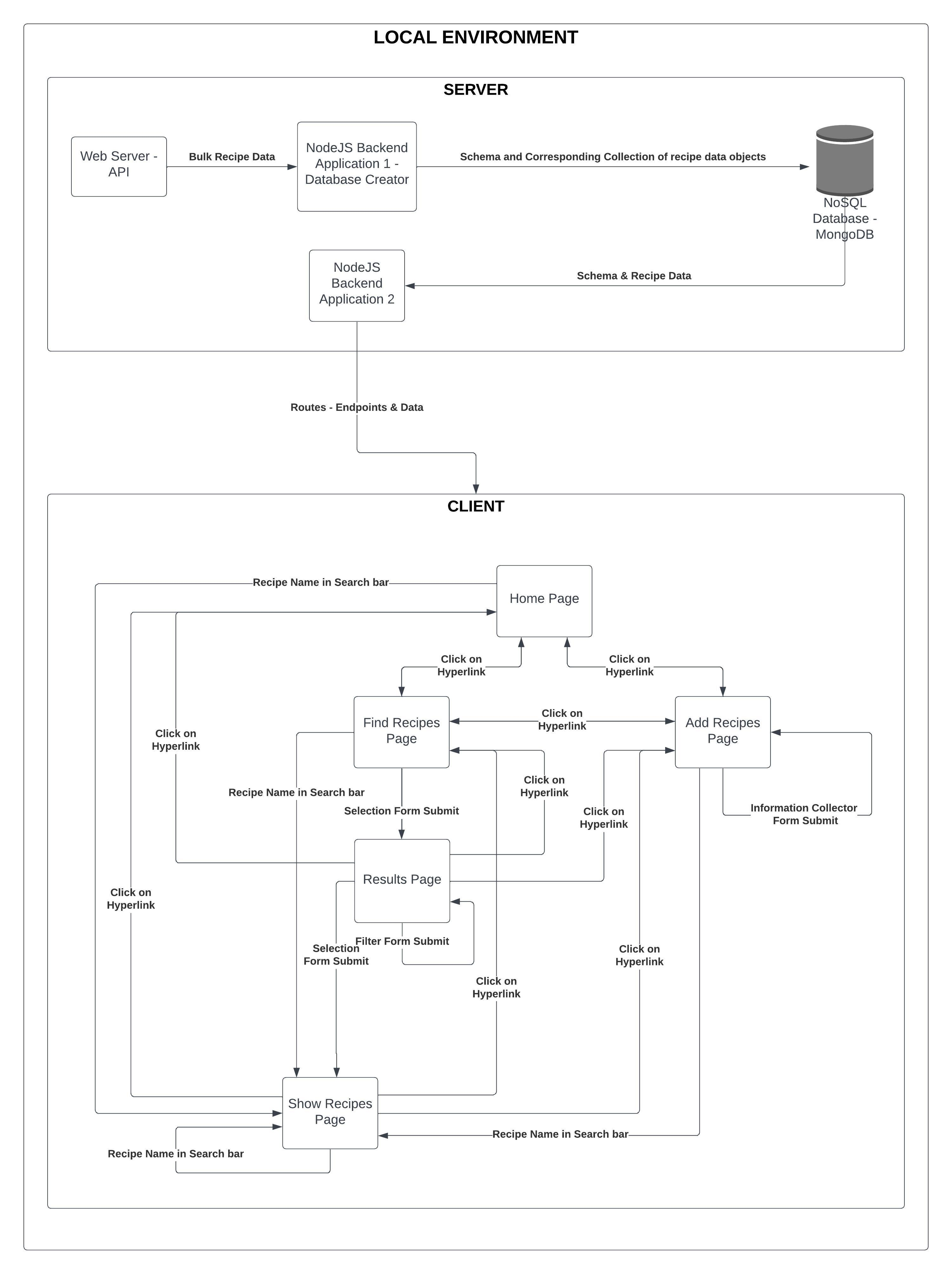


Figure 1: Block Diagram of the System showing the dataflow and interactions between components

**Design Specifications**

**HTA**

Hierarchical Task Analysis is a method used to describe the structure of tasks and subtasks within a system. It breaks down complex tasks into a hierarchy of smaller, more manageable elements.

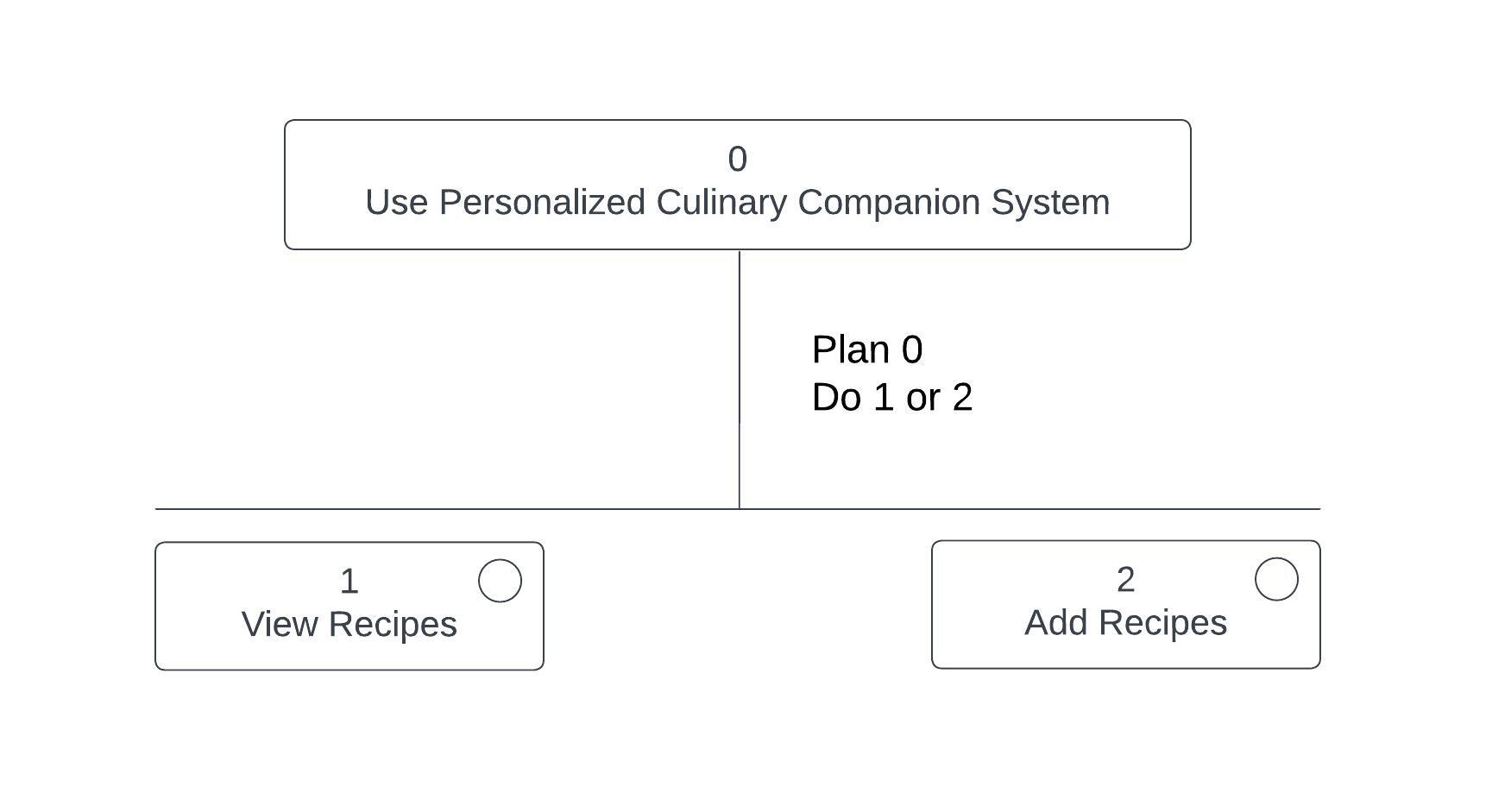


Figure 2: Root tasks of HTA

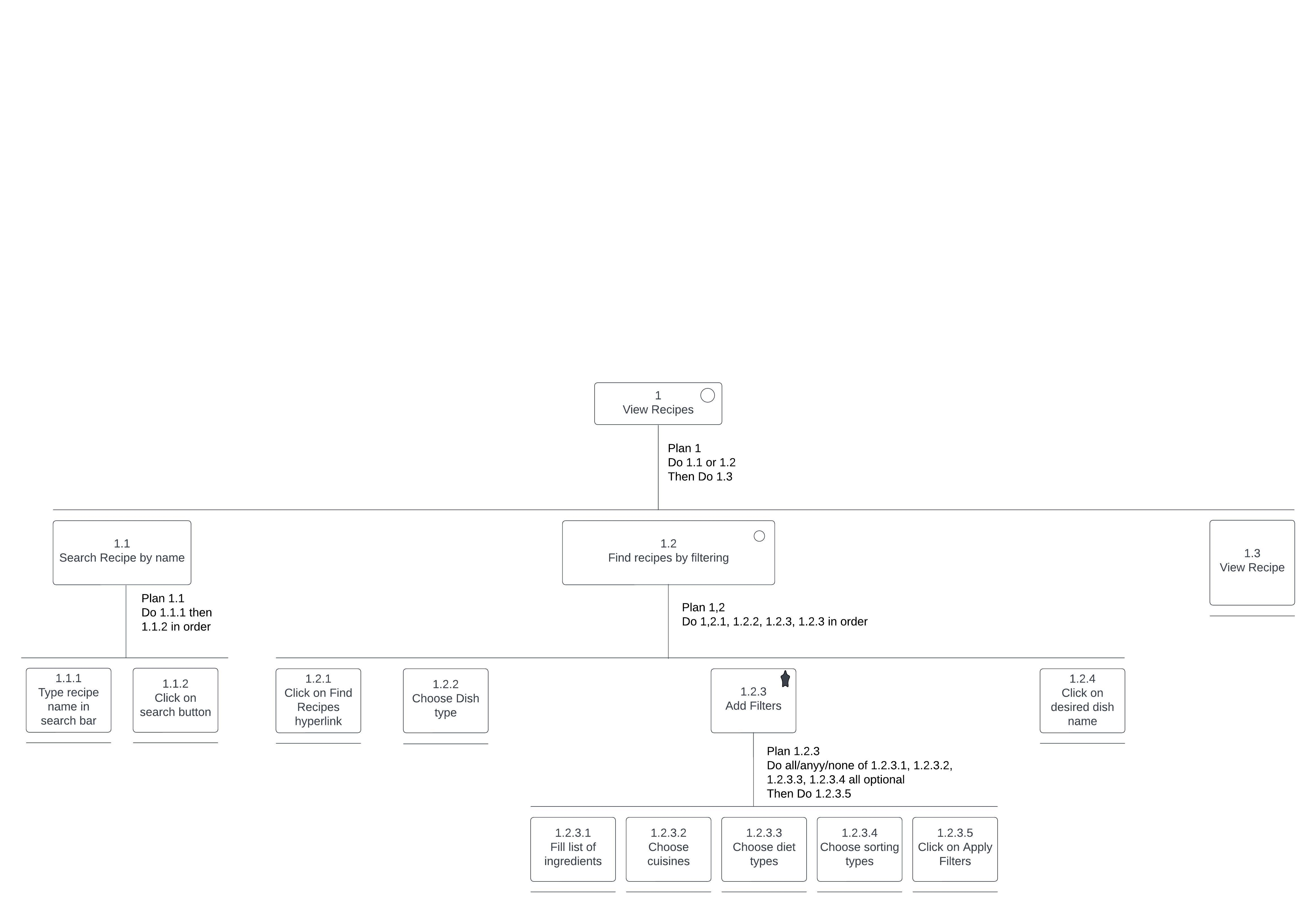


Figure 3: 1st Root Task of HTA

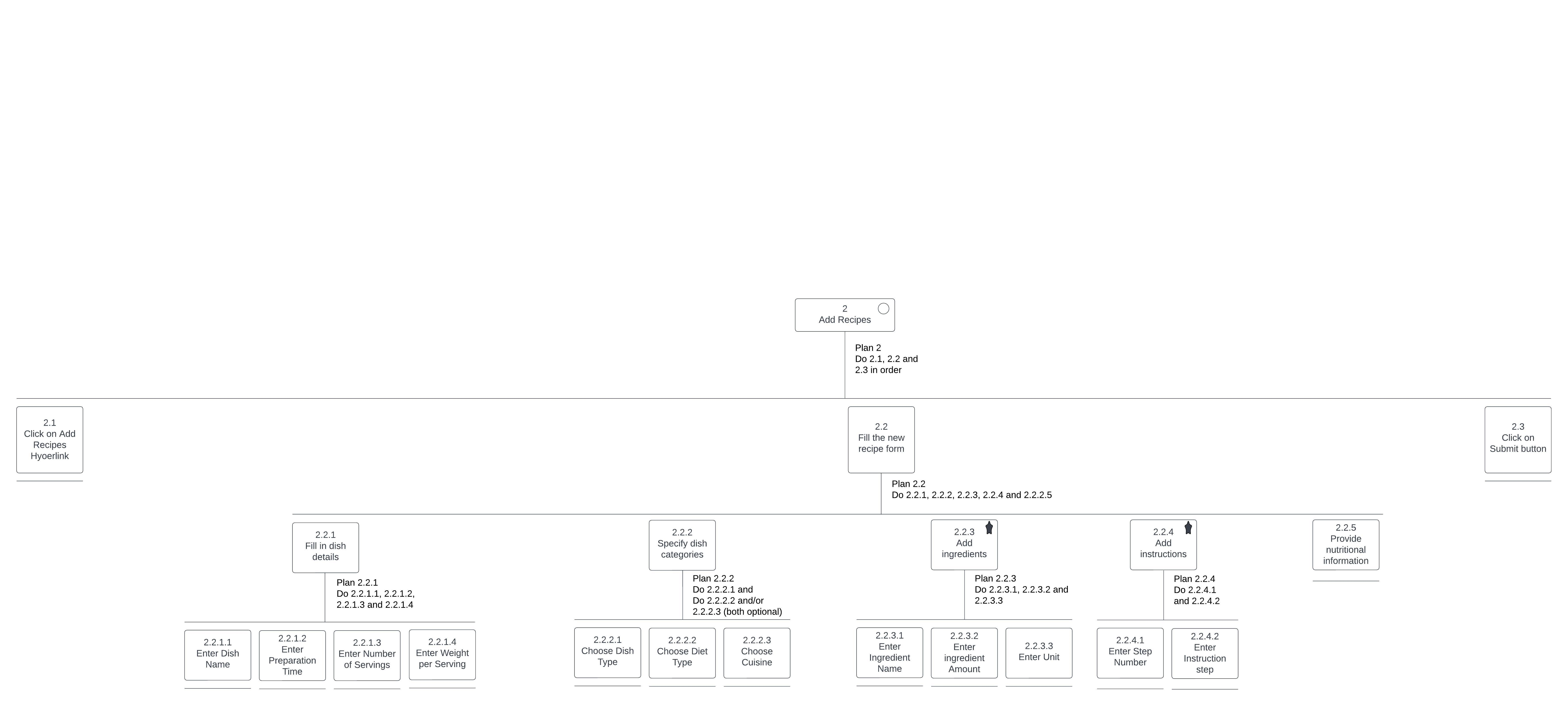


Figure 4: 2nd Root Task of HTA

**Interaction Design**

1. **Identify Goals:** In this step, the primary objectives and purposes of the system are defined.
   1. Goal 1: User-Friendly Recipe Submission
      * Objective: Design an interface that allows users to easily submit their recipes.
      * Prioritize essential information to streamline the submission process.
   2. Goal 2: Efficient Recipe Discovery
      * Objective: Create a robust search functionality for users to find recipes based on various criteria such as cuisine, dish type, and dietary preferences.
2. **Develop Personas:** Understanding the target audience by creating personas helps in tailoring the design to user needs.
   1. Persona 1: Casual Home Cook (Eva)
      * Description: Enjoys cooking as a hobby.
      * Needs: Quick and easy recipe submission, recipe discovery for inspiration.
   2. Persona 2: Health-Conscious User (Alex)
      * Description: Focuses on nutritional content.
      * Needs: Detailed nutritional information, search filters for specific dietary requirements.
3. **Create Scenarios and Storyboarding:** Scenarios and storyboards help visualize how users interact with the system.
   1. Scenario 1: Adding a Recipe (Eva)
      * Eva adds a breakfast recipe quickly using an intuitive form.
   2. Scenario 2: Finding Dinner Ideas (Alex)
      * Alex searches for low-carb dinner recipes with a specific cuisine preference.
4. **Use Case Modelling:** Use cases outline the interactions between users and the system.
   1. Use Case 1: Add New Recipe
      * Actors: Registered Users
      * Main Flow: User fills out recipe details, adds ingredients, and submits.
   2. Use Case 2: Search for Recipe
      * Actors: Registered Users
      * Main Flow: User enters search criteria, browses results, and views a detailed recipe.

**Stakeholders Identification**

Identifying stakeholders is crucial for understanding the perspectives and interests involved in the system.

1. **Users:**
   * Home Cooks (Eva): Interested in easy recipe submission and discovery.
   * Nutrition Enthusiasts (Alex): Look for detailed nutritional information and specific dietary options.
2. **Developers:**
   * Frontend Developers: Responsible for designing the user interface and ensuring a smooth user experience.
   * Backend Developers: Handle data storage, retrieval, and system functionality.
3. **Administrators:**
   * Content Moderators: Review and manage submitted recipes.
   * System Administrators: Oversee system maintenance and ensure smooth operations.
4. **Business Owners:**
   * Stakeholders with Business Interest: Invested in the success and growth of the system.

**Storyboarding**

Storyboarding for Recipe Management System

Scene 1: User Browsing Recipes

1. Setting:

- User opens the Recipe Management System on a laptop.

2. Action:

- User is in the "Browsing State."

- Scans through the recipe catalog.

3. Interaction:

- Clicks on a recipe card.

4. Transition:

- System moves to "Recipe View State."

---

Scene 2: Viewing Recipe Details

1. Setting:

- User is in the "Recipe View State."

2. Action:

- Views detailed information about the selected recipe.

3. Interaction:

- Clicks on the "Show More" button for additional nutritional details.

4. Transition:

- System reveals extra nutritional values.

---

Scene 3: Adding a New Recipe

1. Setting:

- User is in "Browsing State."

2. Action:

- Wants to add a new recipe.

3. Interaction:

- Clicks on "Add New Recipe" in the navigation.

4. Transition:

- System moves to "Adding State."

---

Scene 4: System Feedback

1. Setting:

- User has just submitted a new recipe.

2. Action:

- System processes the submission.

3. Interaction:

- Receives feedback message.

4. Transition:

- System moves back to "Browsing State."

---

Scene 5: Searching for Recipes

1. Setting:

- User is in "Browsing State."

2. Action:

- Wants to find a specific recipe.

3. Interaction:

- Initiates a search using the search bar.

4. Transition:

- System enters "Search State" with search results displayed.

---

These storyboard scenes illustrate the seamless flow of user interactions within the Recipe Management System. The transitions between states ensure a user-friendly experience, allowing users to effortlessly navigate, view, edit, and add recipes while receiving clear feedback from the system.

**Use Case Modelling**

Use Case Modeling for Recipe Management System

Use Case Diagram:

![Use Case Diagram](insert\_image\_link\_here)

Use Cases:

1. Browsing Recipes:

- Actor: User

- Description: Allows the user to browse and view recipes in the system.

2. View Recipe Details:

- Actor: User

- Description: Permits the user to view detailed information about a selected recipe.

3. Edit Recipe:

- Actor: User

- Description: Enables the user to modify the details of an existing recipe.

4. Add New Recipe:

- Actor: User

- Description: Allows the user to add a new recipe to the system.

5. Search for Recipes:

- Actor: User

- Description: Lets the user search for specific recipes based on keywords.

6. Submit Feedback:

- Actor: User

- Description: Allows the user to submit feedback regarding the system.

7. Log In:

- Actor: User

- Description: Permits the user to log in to their account.

8. Log Out:

- Actor: User

- Description: Allows the user to log out of the system.

Use Case Relationships:

- Includes Relationship:

- Edit Recipe includes View Recipe Details:

- Editing a recipe requires viewing its details first.

- Extends Relationship:

- Add New Recipe extends Edit Recipe:

- If the user decides to add a new recipe, it extends the basic editing functionality.

Use Case Descriptions:

1. Browsing Recipes:

- User can scroll through a catalog of recipes.

- User can click on a recipe to view details.

2. View Recipe Details:

- User can see detailed information about a selected recipe.

- User can choose to view more nutritional details.

3. Edit Recipe:

- User can edit various details of an existing recipe.

- Editing includes modifying ingredients, instructions, and nutritional information.

4. Add New Recipe:

- User can add a new recipe to the system.

- Adding includes providing dish name, preparation time, servings, etc.

5. Search for Recipes:

- User can search for recipes using keywords.

- Search results are displayed based on user input.

6. Submit Feedback:

- User can submit feedback about their experience.

- Feedback may include suggestions, issues, or compliments.

7. Log In:

- User can log in using valid credentials.

- Provides access to personalized features.

8. Log Out:

- User can log out, terminating the current session.

- Ensures security and privacy.

This use case model outlines the various interactions users can have with the Recipe Management System, detailing their actions, relationships, and the system's responses.

**Personas**

Below are four personas that represent the diverse user profiles for my Personalized Culinary Companion system, ensuring that it caters to a broad audience with varying cooking preferences, skill levels, and lifestyle choices.

1. **Alex Cunningham - The Busy Professional**
   1. Background: Alex is a 30-year-old marketing executive with a hectic work schedule. Lives in a bustling city and enjoys exploring new cuisines.
   2. Goals:
      1. Seeks quick and easy recipes for busy weekdays.
      2. Interested in diverse recipes to add excitement to meals.
   3. Challenges:
      1. Limited time for elaborate cooking.
      2. Needs recipes that align with a healthy lifestyle.
2. **Eva Rodriguez - The Culinary Enthusiast**
   1. Background: Eva, 28, works remotely as a graphic designer. Enjoys experimenting with flavours and trying gourmet recipes.
   2. Goals:
      1. Finds joy in the cooking process and discovering unique dishes.
      2. Interested in recipes with detailed instructions and rare ingredients.
   3. Challenges:
      1. Struggles to discover new recipes beyond mainstream platforms.
      2. Desires a platform that caters to her adventurous taste buds.
3. **Samir Patel - The Health-Conscious Fitness Freak**
   1. Background: Samir, 35, works in IT and maintains an active lifestyle. Prioritizes healthy eating and follows specific dietary plans.
   2. Goals:
      1. Seeks recipes with precise nutritional information.
      2. Interested in meal plans that align with fitness goals.
   3. Challenges:
      1. Difficulty finding recipes with accurate nutritional data.
      2. Wants a platform that caters to specific dietary needs.
4. **Olivia Johnson - The Novice Cook**
   1. Background: Olivia is a 25-year-old student who recently started cooking. Has a limited culinary skill set but eager to learn.
   2. Goals:
      1. Looks for beginner-friendly recipes with step-by-step guidance.
      2. Interested in building a repertoire of easy-to-cook meals.
   3. Challenges:
      1. Struggles with complex cooking terminology.
      2. Needs a platform that encourages and educates novice cooks.

**State Transition Network**

State Transition Network

The state transition network for the Recipe Management System outlines the various states the system can exist in and the transitions between them. The primary states and transitions are as follows:

States:

1. Idle State:

- Description:

- The system is in a standby state, waiting for user interaction.

- Transitions:

- Triggered by the user accessing the system.

2. Browsing State:

- Description:

- The user is navigating through the recipe catalog.

- Transitions:

- Transition to the "Recipe View State" when a recipe is selected.

- Transition to the "Search State" when the user initiates a search.

3. Search State:

- Description:

- The user is actively searching for specific recipes.

- Transitions:

- Transition to the "Recipe View State" when a search result is selected.

- Return to "Browsing State" when the search is cleared or completed.

4. Recipe View State:

- Description:

- The user is viewing details of a specific recipe.

- Transitions:

- Transition to the "Editing State" when the user chooses to edit the recipe.

- Return to "Browsing State" when the user exits the recipe view.

5. Editing State:

- Description:

- The user is modifying the details of a recipe.

- Transitions:

- Transition to the "Recipe View State" when the editing is completed.

- Return to "Recipe View State" without saving if the user chooses to cancel.

6. Adding State:

- Description:

- The user is adding a new recipe to the system.

- Transitions:

- Transition to "Recipe View State" when the addition is completed.

- Return to "Browsing State" if the user chooses to cancel.

7. Feedback State:

- Description:

- The system displays feedback or confirmation messages.

- Transitions:

- Return to the previous state after the user acknowledges the feedback.

Transitions:

- User Interaction:

- Triggered by user actions like clicks, searches, and edits.

- Directs the system to transition between states based on user input.

- System Events:

- Triggered by backend processes, e.g., data retrieval, updates, and submissions.

- Induces transitions to update the interface based on system responses.

The state transition network ensures a smooth and intuitive user experience, allowing users to seamlessly navigate through the Recipe Management System's features and functionalities.

**Application of Guidelines/Principles**

**Schneideman's Eight Golden Rules**

1. **Strive for Consistency:** The system maintains consistency in design, terminology, and interaction patterns.
2. **Enable Frequent Users to Use Shortcuts:** There are no shortcuts as such but advanced search options cater to users familiar with the system.
3. **Offer Informative Feedback**: The system provides timely feedback on search results and successful recipe submissions.
4. **Design Dialogs to Yield Closure:** The workflow for finding and viewing recipes follows a logical sequence, providing closure.
5. **Offer Simple Error Handling:** Error messages are clear and guide users on resolving issues.
6. **Permit Easy Reversal of Actions:** Users can navigate back and forth in the recipe details view, the applied filters will remain but the contents of the form will be lost.
7. **Support Internal Locus of Control:** Users do not have control over editing and deleting their recipes, so there is limited support for internal locus of control.
8. **Reduce Short-Term Memory Load:** Clear labels, placeholders, visual cues and use of checkboxes in both finding and adding recipes reduce cognitive load.

**Heuristic Evaluation**

A heuristic evaluation was conducted on the system to assess its usability and identify potential usability issues. This evaluation focuses on the two main functionalities: finding recipes and adding recipes.

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **Heuristic Evaluation Criteria** | **Finding Recipes** | **Adding Recipes** |
| 1 | Visibility of System Status | The system provides clear feedback on the status, such as loading indicators during searches and successful submission messages. | Feedback is provided after adding a recipe, indicating success or any errors. |
| 2 | Match Between System and the Real World | The language used for dish types, cuisines, and diet types aligns with users' expectations, enhancing user understanding. | Terminology in the recipe creation process mirrors real-world cooking terms, improving user recognition. |
| 3 | User Control and Freedom | The filters are lost when the user navigates back, making it slightly tedious for them if they had to make changes in their previous selections. However, the user can make additional filter selections after getting the previous filters’ results as well. | The system does not allow users to edit and delete the recipes that they have added, so there’s limited control over their content. |
| 4 | Consistency and Standards | The layout and design are consistent throughout the recipe finding process. | The form elements maintain a consistent structure during recipe addition. |
| 5 | Error Prevention | The system prevents errors by handling empty or invalid searches gracefully. | Mandatory fields are clearly marked, reducing the likelihood of submission errors. |
| 6 | Recognition Rather Than Recall | The search interface displays recipes that match the entered keywords, aiding user recognition. | Users are prompted with clear labels and placeholders during recipe creation, and are provided checkboxes for category names (such as dish types, cuisines, and diet types), minimizing the need for recall. |
| 7 | Flexibility and Efficiency of Use | Advanced search options allow experienced users to quickly refine their queries. For expert users to follow, the filters applied are displayed as query parameters in the URL. | The recipe addition form is streamlined, catering to users of varying expertise. Those who only have basic knowledge of recipe ingredients and instructions are provided instructions to ignore the form sections about nutritional details. |
| 8 | Aesthetic and Minimalist Design | The search results only display the name of the dishes which can be clicked to get the complete dish and recipe details hence, only essential information is presented without unnecessary clutter. | The recipe creation form is very well-organized, avoiding unnecessary elements. |
| 9 | Error Recovery | The system effectively handles errors, but does not provide meaningful feedback and guidance on resolving issues. | In the recipe addition process, if there are errors or missing information, the system guides users on how to correct these issues before submission. |
| 10 | Help and Documentation | There are instructions for the application of each filter, but there is no overall help section or complete documentation available to users. | Users are given instructions about which boxes to necessarily fill and how, but other than that there is no overall help section or complete documentation available to users. |

**KLM Analysis**

**Task 1: Finding Recipes**

KLM Operators:

* K1: Press keys for typing the dish name. (N keystrokes)
* K2: Press keys for applying filters. (N keystrokes)
* P1: Move the mouse to click on suggested dish name.
* P2: Move the mouse to navigate through search bar.
* P3: Move the mouse to choose dish type.
* P4: Move the mouse to go to click on buttons and hyperlinks.
* H1: Move hands between the keyboard and mouse.
* M1: Time taken for decision-making while browsing dish names.
* M2: Time taken for decision-making while choosing dish type.
* M3: Time taken for decision-making during filtering.
* R1: Waiting for system response after applying filters.
* B: Press/Release mouse button.

Task Execution Steps:

1. Observe and click on the search bar (P2+B+B).
2. Type dish name (N keystrokes) (H1+K1).
3. Browse through suggested dish names (Scroll) (H1+P1).
4. Observe and click on suggested dish name (M1+B+B).
5. Click on 'Find Recipes' button (P4+B+B).
6. Browse through dish types (Scroll) (P3).
7. Observe and click on desired dish type (M2+B+B).
8. Apply F filters (F\*N keystrokes) (F\*H1+M3+K2).
9. Click on 'Apply Filters' (H1+P4+B+B+R1).
10. Browse through filtered results (Scroll) (P1).
11. Click on the desired dish name (M1+B+B).

**Task 2: Adding Recipes**

KLM Operators:

* K1: Press keys for filling out the recipe form. (N keystrokes)
* P1: Move the mouse to click checkboxes and buttons.
* H1: Move hands between the keyboard and mouse.
* M1: Time taken for decision-making during data entry.
* R1: Waiting for system response after submitting the recipe.
* B: Press/Release mouse button.

Task Execution Steps:

1. Click on 'Add Recipes' button (P1+B+B).
2. Enter dish name, preparation time, number of servings, weight per serving. (N keystrokes) (H1+K1).
3. Check checkboxes for dish types, cuisines, and diet types (H1+B+B).
4. Enter required ingredient names, amounts, and units (N keystrokes) (H1+M1+K1).
5. Enter step-by-step instructions (N keystrokes) (H1+M1+K1).
6. Enter nutritional details (N keystrokes) (H1+M1+K1).
7. Click 'Submit' (H1+P1+B+B).

**GOMS Analysis**

**Task 1: Finding Recipes**

Goal: Find a recipe efficiently.

Operators:

* O1: Observe the search bar.
* O2: Click on the input box.
* O3: Type dish name ('N' keystrokes).
* O4: Browse through suggested dish names.
* O5: Observe and click on desired dish name.
* O6: Click on the 'Find Recipes' button.
* O7: Observe Dish Type names and choose the desired dish type.
* O8: Browse through results.
* O9: If desired then apply filters (ingredient names, diet types, cuisines, preparation time, nutritional contents).
* O10: Click on 'Apply Filters.'
* O11: Browse through filtered results.
* O12: Click on the desired dish name.

Methods:

* M1: Use the mouse or keyboard for navigation.
* M2: Utilize dropdowns, checkboxes, and text input fields for data entry.
* M3: Navigate through the search bar using arrow keys.

Selection Rules:

* S1: Use the mouse or keyboard based on user preference.

Evaluation:

* Efficiency: GOMS analysis provides insights into potential optimizations for quick and efficient recipe finding.
* User Guidance: The analysis helps design the system to accommodate both novice and power users.

**Task 2: Adding Recipes**

Goal: Add a new recipe efficiently.

Operators:

* O1: Click on 'Add Recipes' button.
* O2: Fill out the recipe form (dish name, preparation time, servings, weight per serving).
* O3: Check dish types, cuisines, and diet types checkboxes.
* O4: Enter required ingredient names, amounts, and units.
* O5: Enter step-by-step instructions.
* O6: Enter nutritional details (if known).
* O7: Click 'Submit.'

Methods:

* M1: Utilize mouse or keyboard for navigation.
* M2: Use checkboxes, text input fields, and dropdowns for data entry.

Selection Rules:

* S1: Provide keyboard shortcuts for users who prefer quick interactions.
* S2: Maintain consistency with the system's overall design.

Evaluation:

* Efficiency: GOMS analysis highlights areas for optimizing the process of adding recipes.
* User Guidance: The analysis contributes to designing an intuitive and user-friendly recipe addition workflow.

**Communication, Collaboration & Groupware**

**Communication Features**

* + - 1. **Real-time Recipe and Ingredient Suggestions:**

The system employs a real-time suggestion mechanism that dynamically displays recipe and ingredient suggestions as users type in the search bar. This feature enhances communication by providing instant feedback and aiding users in finding recipes efficiently.

1. **Ingredient-based Filtering:**

Users have the ability to filter recipes based on specific ingredients. This feature enables users to communicate their preferences indirectly by selecting ingredients they want to include or exclude from their recipes.

1. **Dish Type and Cuisine Filters:**

The system allows users to filter recipes based on dish types and cuisines. Users can communicate their culinary preferences by choosing specific dish types or cuisines, providing a personalized recipe browsing experience.

1. **Filtering by Diet Type:**

Users can filter recipes based on diet types. This feature allows users to communicate their dietary preferences, ensuring that the system provides recipes aligning with their specific nutritional needs.

1. **Sorting Options:**

Users can sort recipes based on various criteria, such as preparation time or nutritional content. This feature enables users to communicate their priorities, whether they are looking for quick recipes or focusing on specific dietary requirements.

1. **Add New Recipe:**

The "Add Recipes" page provides a form for users to submit their recipes to the system. This feature allows users to share their unique recipes with the community, contributing to the overall content and communication within the platform.

**Collaboration Features**

1. **User-Submitted Recipes:**The system encourages collaboration by allowing users to submit their recipes. The "Add Recipes" page provides a form where users can share their culinary creations, fostering a collaborative environment where individuals contribute to the recipe database.
2. **Community-Driven Platform:**The user-submitted recipes, along with the filtering options, contribute to the creation of a community-driven platform. Users actively participate in building the recipe repository, ensuring a collaborative space where individuals share their expertise and experiences.
3. **Personalized Recipe Browsing:**Through various filters like dish types, cuisines, and ingredient-based filtering, users collaboratively shape their recipe browsing experience. The system adapts to user preferences, creating a collaborative platform where the community collectively influences the content available.

**Groupware Features**

1. **Shared Recipe Repository:**The system acts as a shared repository for recipes, serving as a groupware platform where users contribute, access, and benefit from a collective database of culinary creations. This shared repository enhances collaboration and information exchange among users.
2. **Collaborative Filtering:**The system incorporates collaborative filtering mechanisms, allowing users to filter recipes based on dish types, cuisines, and ingredients. This collaborative approach enables users to discover recipes that align with collective preferences, creating a shared experience within the community.
3. **User-Generated Content:**The addition of recipes by users results in a wealth of user-generated content. This groupware feature encourages users to actively contribute to the system, making it a collaborative platform where individuals share their expertise and culinary creations with the broader community.
4. **Dynamic Content Updates:**The real-time updates of recipe suggestions, filtering options, and user-contributed content create a dynamic and collaborative system. Users experience a shared environment where changes made by one user dynamically impact the collective experience of the entire community.
5. **Implicit Collaboration through Search:**The search functionality allows users to implicitly collaborate by exploring recipes based on various criteria. This groupware feature facilitates information sharing and collective decision-making as users navigate through the recipe database.

VALIDATION-USABILITY TESTING, INTERFACE TESTING,USER ACCEPTANCE TESTING : USING ANY OPEN SOURCE TOOL

<https://reflect.run/articles/installing-chromedriver-and-python-selenium/#installing-and-configuring-selenium>

<https://stackoverflow.com/questions/48139676/how-to-get-the-value-of-an-element-in-python-selenium>

<https://selenium-python.readthedocs.io/locating-elements.html>

Certainly! Below are the Selenium test scripts for each of the four scenarios:

### 1. Search for Recipes:

```python

from selenium import webdriver

import time

# Set up the WebDriver (Chrome in this case)

driver = webdriver.Chrome(executable\_path='path/to/chromedriver')

# Navigate to the home page

driver.get('https://your-web-app.com')

# Locate the search bar, enter a recipe name, and click search

search\_bar = driver.find\_element\_by\_id('searchBar')

search\_bar.send\_keys('Pasta')

search\_button = driver.find\_element\_by\_id('searchButton')

search\_button.click()

# Assertions to check if search results are displayed

time.sleep(2) # Wait for results to load

assert "Pasta Recipe" in driver.page\_source

# Close the browser

driver.quit()

```

### 2. Filtering Recipes:

```python

from selenium import webdriver

import time

# Set up the WebDriver (Chrome in this case)

driver = webdriver.Chrome(executable\_path='path/to/chromedriver')

# Navigate to the recipes page

driver.get('https://your-web-app.com/recipes')

# Locate filters, choose options, and apply filters

dish\_type\_filter = driver.find\_element\_by\_id('dishTypeFilter')

dish\_type\_filter.click()

# Choose a dish type, etc.

apply\_filters\_button = driver.find\_element\_by\_id('applyFiltersButton')

apply\_filters\_button.click()

# Assertions to check if filtered results are displayed

time.sleep(2) # Wait for results to load

assert "Filtered Recipe 1" in driver.page\_source

# Close the browser

driver.quit()

```

### 3. Viewing Recipe Details:

```python

from selenium import webdriver

import time

# Set up the WebDriver (Chrome in this case)

driver = webdriver.Chrome(executable\_path='path/to/chromedriver')

# Navigate to a recipe details page

driver.get('https://your-web-app.com/recipe/123') # Replace with an actual recipe ID

# Assertions to check if detailed information is displayed

assert "Recipe Title" in driver.page\_source

assert "Preparation Time" in driver.page\_source

# Close the browser

driver.quit()

```

### 4. Adding a Recipe:

```python

from selenium import webdriver

import time

# Set up the WebDriver (Chrome in this case)

driver = webdriver.Chrome(executable\_path='path/to/chromedriver')

# Navigate to the add recipe page

driver.get('https://your-web-app.com/add-recipe')

# Fill out the recipe form

dish\_name = driver.find\_element\_by\_id('dishName')

dish\_name.send\_keys('New Recipe')

# ... Continue filling out other form fields ...

# Submit the form

submit\_button = driver.find\_element\_by\_id('submitButton')

submit\_button.click()

# Assertions to check if the new recipe is added successfully

time.sleep(2) # Wait for the new recipe to be added

assert "New Recipe" in driver.page\_source

# Close the browser

driver.quit()

```

These scripts cover the basic functionalities of searching, filtering, viewing recipe details, and adding a new recipe using Selenium commands. Remember to replace placeholder URLs and IDs with the actual ones from your web application.

RESULTS AND DISCUSSION-GRAPH GENERATION

CONCLUSION: CONCLUDING WITH GIVING DETAILS OF IMPROVEMENT IN PERFORMANCE AND APPLICABILITY OF THE SYSTEM IN REAL TIME ENVIRONMENT