**Spotifood**

**Executive summary**

**By**

**Mr. Kittinut  Saengsri 592115003**

**Mr. Sirisak  Wongwal 592115031**

**Department of Software Engineering**

**College of Arts, Media, and Technology**

**Chiang Mai University**

**Project Advisor**

**Dr. Passakorn  Phannachitta**

**Executive summary**

**Proposal**

In this progress, I had meetings with the project advisor, discussing to define

the scope of the project, the alternative solutions, development plan, deliverables, and

the schedule of the project.

**Progress I**

In this progress, I established documents including project plan, software

requirement specification, software design document, test plan, test record, and

traceability record. I also implemented software covering two features in this progress,

which are

Feature #1: Authentication

Feature #2: Content CRUD

Feature #3: User management

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Name: BuffQ** | | | |
| **Create By:** | **Date** | **Reporting Progress:** | |
| SW, KT | 29/07/2019 | Progress 1 | |
| **Project Overall Status: 50%** | | | |
| **Milestone Deliverables performance reporting** | | | |
| **Milestone** | **Due Date** | **%Complete** | **Deliverable Status** |
| M1 Proposal | 18/07/2019 | 100% | On Schedule |
| M2 Progress 1 | 30/07/2019 | 89% | On Schedule |

\*SW : Sirisak Wongwal

\*KT : Kittnut saengsri

**Features in progress #1**

|  |  |  |
| --- | --- | --- |
| **Features** | **Name** | **Status** |
| 1 | Authentication | Complete |
| 2 | Content CRUD | Incomplete |
| 3 | User management | Incomplete |

**Summary Report**

|  |  |  |
| --- | --- | --- |
| **Document** | **Version** | **Status** |
| Spotifood\_Proposal\_V.3.0 | V.3.0 | 100% Complete |
| Spotifood\_ProjectPlan\_V.1.0 | V.1.0 | 100% Complete |
| Spotifood\_SRS\_V.1.0 | V.1.0 | 100% Complete |
| Spotifood\_SoftwareDesign\_V.1.0 | V.1.0 | 100% Complete |
| Spotifood\_TestPlan\_V.1.0 | V.1.0 | 100% Complete |
| Spotifood\_TestRecord\_V.1.0 | V.1.0 | 100% Complete |
| Spotifood\_Traceability\_V.1.0 | V.1.0 | 100% Complete |
| Spotifood\_ES\_V.1.0 | V.1.0 | 100% Complete |
| Coding | V.1.0 | 75% Complete |

Table of Contents

[**Summary Report** 3](#_Toc15338312)

Project manager plan

[Chapter 1 | Introduction 11](#_Toc15338313)

[1.1 Identification 11](#_Toc15338314)

[1.2 Project Overview 11](#_Toc15338315)

[1.2.1 Purpose 11](#_Toc15338316)

[1.2.2 Scope 11](#_Toc15338317)

[1.3 Acronyms and Definitions 12](#_Toc15338318)

[1.3.1 Acronyms 12](#_Toc15338319)

[1.3.2 Definitions 12](#_Toc15338320)

[Chapter 2 | Infrastructure 14](#_Toc15338321)

[2.1 Development Tools 14](#_Toc15338322)

[2.2 Hardware and Material Resources 14](#_Toc15338323)

[Chapter 3 | Management Procedures 15](#_Toc15338324)

[3.1 Project Team Structures 15](#_Toc15338325)

[3.2 Monitoring and Controlling Mechanisms 15](#_Toc15338326)

[3.2.1 Project Meeting 15](#_Toc15338327)

[Chapter 4 | Quality Standard 16](#_Toc15338328)

[4.1 ISO 29110 for Very Small Entity (VSE) 16](#_Toc15338329)

[4.1.1 Project Management (PM) Process 16](#_Toc15338330)

[4.1.2 Software Implementation (SI) Process 17](#_Toc15338331)

[Chapter 5 | Quality Planning 19](#_Toc15338332)

[5.1 Review/Responsibility 19](#_Toc15338333)

[5.2 Testing 20](#_Toc15338334)

[Chapter 6 | Estimated Duration of Tasks 21](#_Toc15338335)

[6.1 Schedule & Milestone 21](#_Toc15338336)

[Chapter 7 | Software Configuration Management 23](#_Toc15338337)

[7.1 Naming Convention 23](#_Toc15338338)

[7.2 Change Management 23](#_Toc15338339)

[7.3 Project Repository 23](#_Toc15338340)

[Chapter 8 | Risk Management 24](#_Toc15338341)

[8.1 Risk Identification and Solutions 24](#_Toc15338342)

Software requirement specification

[Chapter 1 | Introduction 27](#_Toc15338343)

[1.1 Identification 27](#_Toc15338344)

[1.2 Project Overview 27](#_Toc15338345)

[1.2.1 Purpose 27](#_Toc15338346)

[1.2.2 Scope 27](#_Toc15338347)

[1.3 Work Product to  be developed 27](#_Toc15338348)

[1.3.1 Deliverable 28](#_Toc15338349)

[1.4 Acronyms and Definitions 28](#_Toc15338350)

[1.4.1 Acronyms 28](#_Toc15338351)

[CHAPTER 2 | Overall Description 29](#_Toc15338352)

[2.1 Product Perspective 29](#_Toc15338353)

[2.2 Product Function 29](#_Toc15338354)

[2.3 User Characteristics 29](#_Toc15338355)

[CHAPTER 3 | USER REQUIREMENT SPECIFICATION 30](#_Toc15338356)

[CHAPTER 4 | SOFTWARE REQUIREMENT SPECIFICATION 31](#_Toc15338357)

Software design document

[Chapter 1 | Introduction 75](#_Toc15338358)

[1.1 Identification 75](#_Toc15338359)

[1.2 Project Overview 75](#_Toc15338360)

[1.2.1 Purpose 75](#_Toc15338361)

[1.2.2 Scope 75](#_Toc15338362)

[1.3 Work Product to be developed 76](#_Toc15338363)

[1.3.1 Deliverable 76](#_Toc15338364)

[1.4 Acronyms and Definitions 76](#_Toc15338365)

[1.4.1 Acronyms 76](#_Toc15338366)

[CHAPTER 2 | SOFTWARE ARCHITECTURE 77](#_Toc15338367)

[CHAPTER 3 | DETAIL DESIGN 78](#_Toc15338368)

[3.1 CLASS DIAGRAM 78](#_Toc15338369)

[3.2 CLASS DIAGRAM DESCRIPTION 79](#_Toc15338370)

[3.2.1 Spotifood Application Class Description 79](#_Toc15338371)

[Chapter 4 | Sequence Diagram 95](#_Toc15338372)

[Chapter 5 | User Interface 100](#_Toc15338373)

Test record

[Chapter 1 | Introduction 106](#_Toc15338374)

[1.1 Identification 106](#_Toc15338375)

[1.2 Project Overview 106](#_Toc15338376)

[1.2.1 Purpose 106](#_Toc15338377)

[1.2.2 Scope 106](#_Toc15338378)

[1.3 Acronyms and Definitions 107](#_Toc15338379)

[1.3.1 Acronyms 107](#_Toc15338380)

[Chapter 2 | Test Procedure 108](#_Toc15338381)

[2.1 Test Objectives 108](#_Toc15338382)

[2.2 Test Duration Progress Date and Duration 108](#_Toc15338383)

[2.3 Test Responsibility Item Responsibility 108](#_Toc15338384)

[2.4 Test Strategy Spotifood performed testing strategy by the following steps: 108](#_Toc15338385)

[2.5 Result of Testing The results are separated into two parts: 109](#_Toc15338386)

[Chapter 3 | Unit Test 110](#_Toc15338387)

[Chapter 4 | Appendix A 120](#_Toc15338388)

Traceability record

[CHAPTER 1 | INTRODUCTION 123](#_Toc15338389)

[1.1 Purpose 123](#_Toc15338390)

[1.2 Acronyms and Definitions 123](#_Toc15338391)

[1.2.1 Acronyms 123](#_Toc15338392)

[CHAPTER 2 | TRACEABILITY RECORD TABLE 124](#_Toc15338393)

[2.1 URS-SRS-UC-UI-SD-AD 124](#_Toc15338394)

[CHAPTER 3 | APPENDIX 128](#_Toc15338395)

[3.1 User Requirement 128](#_Toc15338396)

[3.2 System Requirement 128](#_Toc15338397)

[3.3 Use Case 132](#_Toc15338398)

[3.4 Activity Diagram 132](#_Toc15338399)

[3.5 Sequence Diagram 133](#_Toc15338400)

[3.6 User Interface 134](#_Toc15338401)

**Spotifood**

**Project Manager Plan**

**By**

**Mr. Kittinut Saengsri 592115003**

**Mr. Sirisak Wongwal 592115031**

**Department of Software Engineering**

**College of Arts, Media, and Technology**

**Chiang Mai University**

**Project Advisor**

**Dr. Passakorn Phannachina**

Document History

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Document name | Version | Status | Date | Viewable | Editable | Responsible |
| Spotifood\_ProjectPlan\_v0.1 | - Add chapter one  - Add chapter two  - Add chapter three  - Add chapter four  - Add chapter five  - Add chapter six  - Add chapter seven  - Add chapter eight | Draft | 07/10/2019 | SW, KS, PP | SW, KS | SW, KS |
| Spotifood\_ProjectPlan\_v1.0 | - Update chapter one  - Update chapter two  - Update chapter three  - Update chapter four  - Update chapter five  - Update chapter six  - Update chapter seven  - Update chapter eight | Draft | 07/29/2019 | SW, KS, PP | SW, KS | SW, KS |

\*SW = Sirisak Wongwal

\*KS = Kittinut Saengsri

\*PP = Dr.Passakorn Phannachitta

# Chapter 1 | Introduction

## 1.1 Identification

This Project Management Plan is the document for planning, scheduling activities and evaluating overall of the project so that it could complete successfully despite all the risks. It provides detailed scheduled, assigned task and identified risks of the project.

## 1.2 Project Overview

The Spotifood application will guide people find their new dish and help people who are interested in cooking and people who have cooking skill by suggesting the dishes from a word that people search. The AI system will suggest a dish from the personality of people. For example, The people regularly search steamed fish with herbs dish. The AI system will analyze the ingredient tag of dish that is related to the healthy tag. Then, the AI will suggest the people in the type of healthy dishes.

The suggest dish of AI will help those people found new dishes and challenging in cooking every day that makes people not disconnected and bored to cooking. Moreover, we will be a middleman that brings trader of ingredient to people who can’t find the ingredients in any food. The Spotifood help people to buy an ingredient. And the Spotifood will suggest a dish that is similar to personality or suggests from the behavior of people.

### 1.2.1 Purpose

The main purpose of developing the Spotifood is to help the people who are interested in cooking and who already have skills not bored with cooking or disconnected to prepare the food.

### 1.2.2 Scope

There are six major parts in Spotifood.

Feature #1: Authentication

Feature #2: Content CRUD

Feature #3: User management

Feature #4: Similar dishes identification

Feature #5: Personalization

Feature #6: ADS ingredient

## 1.3 Acronyms and Definitions

### 1.3.1 Acronyms

SRS Software Requirement Specification

SDD Software Design Document

OS Operating System

VSE Very Small Entity

PM Project Management

SI Software Implementation

IID Iterative and Incremental Development

SCI Software Configuration Item

### 1.3.2 Definitions

|  |  |
| --- | --- |
| Name | Definition |
| Acceptance test | Test activities for sample checks to verify that the system (or product, solution) has the right quality for deployment or usage. Often acceptance test is done by the customer. |
| Feature | Transformation of input parameters to output parameters based on a specified algorithm. It describes the functionality of the product in the language of the product. Used for requirements analysis, design, coding, testing or maintenance. |
| IEEE | Institute for Electrical and Electronics Engineers. Biggest global interest group for engineers of different branches and computer scientists. |
| Plan | A documented series of tasks requires meeting an objective, typically including the associated schedule, budget, resources, organizational description and work breakdown structure. |
| Project Management | The application of knowledge, skills, tools, and techniques to project activities in order to meet or exceed stakeholder needs and expectations from a project |
| Project Plan | A formal, approved document used to guide both project execution and project control. The primary uses of the project plan are to document planning assumptions and decisions, to facilitate communication among stakeholders, and document approved scope, cost, and schedule baseline. |
| Risk | An uncertain event or condition that, if it occurs, has a positiveornegativeeffectontheproject’sobjectives.Itisa functionoftheprobabilityofoccurrenceofagiventhreat’s occurrence |
| Risk Management | The systematic application of management policies, procedures  and practices to the tasks of identifying, analyzing, evaluating, treating and monitoring risk. |

|  |  |
| --- | --- |
| Name | Definition |
| Traceability | The ability to trace the history, application or location of an item or activity, or work products or activities, by means of recorded identification. The establishment and maintenance of relationships between such items. Horizontal traceability describes the relationship between work products of the same type (e.g., customer requirements). Vertical traceability describes the relationship between work products which build upon each other or are derived from each other (e.g., from customer requirements to qualification test cases). Bidirectional traceability allows to directly following relationships in both directions |
| Unit test | A test of individual programs or modules in order to remove design or programming errors. |

# Chapter 2 | Infrastructure

## 2.1 Development Tools

* Visual code studio
* Postman
* MongoDB Server

## 2.2 Hardware and Material Resources

* Computer
* Acer Aspire vx15
* Processor: Intel(R) Core(™) i7-7700 HQ @2.80GHz
* Memory: 16GB RAM
* OS: Window 10 Pro x64
* Acer Notebook NITRO AN515-52
* Processor: Intel(R) Core i5-8300H 2.3 GHz up to 4.0 GHz
* Memory: 8GB RAM
* OS: Window 10 Home

# Chapter 3 | Management Procedures

## 3.1 Project Team Structures

|  |  |
| --- | --- |
| Particaipants | Activities |
| Kittinut Saengsri  Sirisak Wongwal | Feasibility Study |
| Project Proposal |
| Project Requirement |
| Project Plan |
| Project Design |
| Implementation |
| Testing |

## 3.2 Monitoring and Controlling Mechanisms

### 3.2.1 Project Meeting

|  |  |
| --- | --- |
| Participants | Roles |
| Mr. Kittinut Saengsri | Development team member |
| Mr. Sirisak Wongwal | Development team member |
| Dr. Passakorn Phannachitta | Project advisor |

# Chapter 4 | Quality Standard

## 4.1 ISO 29110 for Very Small Entity (VSE)

ISO 29110 is the Software Life Cycle Profiles and Guidelines for Very Small Entities (VSEs) standards and technical reports are targeted at Very Small Entities (VSEs). A Very Small Entity (VSE) is an enterprise, organization, department or project having up to 25 people. ISO 29110 concerns on the project management process and software implementation process.

### 4.1.1 Project Management (PM) Process

* **Purpose**

ISO 29110 is the Software Life Cycle Profiles and Guidelines for Very Small Entities (VSEs) standards and technical reports are targeted at Very Small Entities (VSEs). A Very Small Entity (VSE) is an enterprise, organization, department or project having up to 25 people. ISO 29110 concerns on the project management process and software implementation process.

* **Objective**

**PM.O1.** The Project Plan for the execution of the project is developed according to the Statement of Work and validated with the Customer. The tasks and resources necessary to complete the work sized and estimated.

**PM.O2.** Progress of the project is monitored against the Project Plan and recorded in the Progress Status Record. Corrections to remediate problems and deviations from the plan are taken when project targets are not achieved. Appropriate treatment is taken to correct or avoid the impact of risk. Closure of the project is performed to get the Customer acceptance documented in the Acceptance Record.

**PM.O3.** The Change Requests are addressed through their reception and analysis. Changes to software requirements are evaluated for cost, schedule, and technical impact.

**PM.O4.** Review meetings with the Work Team and the Customer are held. Agreements are registered and tracked.

**PM.O5.** Risks are identified as they develop and during the conduct of the project.

**PM.O6.** A software Version Control Strategy is developed. Items of Software Configuration are identified, defined and baseline. Modifications and releases of the items are controlled and made available to the Customer and Work Team including the storage, handling, and delivery of the items.

**PM.O7.** Software Quality Assurance is performed to provide assurance that work products and processes comply with the Project Plan and Requirements Specification.

* **Activities**

**PM.1** Project Planning

**PM.2** Project Plan Execution

**PM.3** Project Assessment and Control

**PM.4** Project Closure

### 4.1.2 Software Implementation (SI) Process

* Purpose

The purpose of the Software Implementation process is the systematic performance of the analysis, design, construction, integration and tests activities for new or modified software products according to the specified requirements.

* Objectives

**SI.O1.** Tasks of the activities are performed through the accomplishment of the current Project Plan.

**SI.O2.** Software requirements are defined, analyzed for correctness and testability, approved by the Customer, baseline and communicated.

**SI.O3.** Software architectural and detailed design is developed and baseline. It describes the software items and internal and external interfaces of them. Consistency and traceability to software requirements are established.

**SI.O4.** Software components defined by the design are produced. Unit tests are defined and performed to verify the consistency with requirements and the design. Traceability to the requirements and design are established.

**SI.O5.** Software is produced performing integration of software components and verified using Test Cases and Test Procedures. Results are recorded at the Test Report. Defects are corrected and consistency and traceability to Software Design are established.

**SI.O6.** A Software Configuration that meets the Requirements Specification as agreed to with the Customer, which includes user, operation and maintenance documentation is integrated, baseline and stored at the Project Repository. Needs for changes to the Software Configuration are detected and related Change Requests are initiated.

**SI.O7.** Verification and Validation tasks of all required work products are performed using the defined criteria to achieve consistency among output and input products in each activity. Defects are identified, and corrected; records are stored in the Verification/Validation Results.

* Activities

**SI.1** Software Implementation Initiation

**SI.2** Software Requirements Analysis

**SI.3** Software Architectural and Detailed Design

**SI.4** Software Construction

**SI.5** Software Integration and Tests

**SI.6** Product Delivery

# Chapter 5 | Quality Planning

## 5.1 Review/Responsibility

|  |  |  |  |
| --- | --- | --- | --- |
| Stage Exit Review | | | |
| No. | Stage | Review Item | Responsibility |
| 1 | Project Planning | Project Plan | Sirisak, Kittinut |
| 2 | Requirement  Specification | Software Requirement  Specification | Sirisak, Kittinut |
| 3 | Architecture and  Detailed Design | Software Design Document | Sirisak, Kittinut |
| 4 | Implementation | Code | Sirisak, Kittinut |
| 5 | Software Testing | Test Plan | Sirisak, Kittinut |
| 6 | Software Testing | Test Record | Sirisak, Kittinut |
| 7 | Project Monitoring  and Control | Traceability | Sirisak, Kittinut |

## 5.2 Testing

|  |  |  |
| --- | --- | --- |
| Stage Exit Review | | |
| No. | Test | Responsibility |
| 1 | Unit Testing | Sirisak, Kittinut |
| 2 | System Testing | Sirisak, Kittinut |

# Chapter 6 | Estimated Duration of Tasks

## 6.1 Schedule & Milestone

|  |  |  |  |
| --- | --- | --- | --- |
| Milestone | Task | Milestone Criteria | Planned date |
| 1 | Proposal | -Topic defined | 21 May, 2019 |
| 2 | Proposal | -Proposal reviewed  -Proposal submitted  -Proposal presentation | 18 June, 2019 |
| 3 | Progress 1 | -Software requirement specification  -Feature#1 (Authentication)  -Feature#2 (Post dish)  -Feature#3 (View dish)  -Feature#4 (Search)  -Feature Design  -Test plan  -Feature implement  -Feature test  -Progress report submit  -Progress report presentation | 12 July, 2019 |
| 4 | Progress 2 | -Software requirement specification  -Feature#5 (Suggest dish)  -Feature#7 (Link to shop)  -Feature Design  -Test plan  -Feature implement  -Feature test  -Progress report submit  -Progress report presentation | 22 August, 2019 |
| 5 | Show pro | Review  -Feature#1 (Authentication)  -Feature#2 (Post dish)  -Feature#3 (View dish)  -Feature#4 (Search)  -Feature#5 (Suggest dish)  -Feature#7 (Link to shop) | Sep, 2019 |

|  |  |  |
| --- | --- | --- |
| Task and Estimated Duration | | |
| No. | Phase | Estimated Duration (Days) |
| 1. | Proposal | 28 |
| 2. | Progress1 | 25 |
| 3. | Progress2 | 60 |
| 4. | Progress3 | 41 |
| Total | | 154 |

# Chapter 7 | Software Configuration Management

## 7.1 Naming Convention

For the filename format that we use for all documents is:

Spotifood-[File Name]\_v[Version].[File Type]

* File Name - This part will depend on the substance of that file.
* Version - This part is the version of the file. The version number will be in this format“[Mainversion].[Subversion]”
* File Type - This part is a type of file.

(I.E. Spotifood--ProjectPlan\_v1.0.docx)

## 7.2 Change Management

Change management manages all of the changes in the project during the development process. All of the change requests will be recorded into the change record document.

The procedures for managing changes are:

1. Discuss with an advisor about the change.

2. Record the change information to change the document.

3. Send the change request to an advisor.

3.1 Request accepted: change document and software follow the change information.

3.2 Request rejected: continue on and find an alternative solution

## 7.3 Project Repository

This project uses“GitHub” to manage the version of documents and software. It can be used to store and share code or binary files for software development projects.

The repository's directories will be created as following;

# Chapter 8 | Risk Management

## 8.1 Risk Identification and Solutions

|  |  |  |
| --- | --- | --- |
| No. | Risk statement | Solution |
| 1 | The developer \lacks necessary skills for project development | - Study from textbook or online resources |
| 2 | The requirements might change. | - Make change request and discuss with an advisor to reprioritize the changed requirements |
| 3 | The process flow might not keep up with the project schedule. | - Start tasks execution before their schedule.  - Prioritize tasks and to do more important tasks first |

**Spotifood**

**Software Requirement Specification**

**By**

**Mr. Kittinut  Saengsri 592115003**

**Mr. Sirisak  Wongwal 592115031**

**Department of Software Engineering**

**College of Arts, Media, and Technology**

**Chiang Mai University**

**Project Advisor**

**Dr. Passakorn  Phannachitta**

Document History

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Document name | Version | Status | Date | Viewable | Editable | Responsible |
| Spotifood\_SRS\_v0.1 | - Add chapter one  - Add chapter two  - Add chapter three  - Add chapter four | Draft | 07/10/2019 | SW, KS, PP | SW, KS | SW, KS |
| Spotifood\_SRS\_v1.0 | - Update chapter one  - Update chapter two  - Update chapter three  - Update chapter four | Draft | 07/29/2019 | SW, KS, PP | SW, KS | SW, KS |

\*SW = Sirisak Wongwal

\*KS = Kittinut Saengsri

\*PP = Dr.Passakorn Phannachitta

# Chapter 1 | Introduction

## 1.1 Identification

This Project Management Plan is the document for planning, scheduling activities and evaluating overall of the project so that it could complete successfully despite all the risks. It provides detailed scheduled, assigned task and identified risks of the project.

## 1.2 Project Overview

The Spotifood application will guide people find their new dish and help people who are interested in cooking and people who have cooking skill by suggesting the dishes from a word that people search. The AI system will suggest a dish from the personality of people. For example, The people regularly search steamed fish with herbs dish. The AI system will analyze the ingredient tag of dish that is related to the healthy tag. Then, the AI will suggest the people in the type of healthy dishes.

The suggest dish of AI will help those people found new dishes and challenging in cooking every day that makes people not disconnected and bored to cooking. Moreover, we will be a middleman that brings trader of ingredient to people who can’t find the ingredients in any food. The Spotifood help people to buy an ingredient. And the Spotifood will suggest a dish that is similar to personality or suggests from the behavior of people.

### 1.2.1 Purpose

The main purpose of developing the Spotifood is to help the people who are interested in cooking and who already have skills not bored with cooking or disconnected to prepare the food.

### 1.2.2 Scope

There are six major parts in Spotifood.

Feature #1: Authentication

Feature #2: Content CRUD

Feature #3: User management

Feature #4: Similar dishes identification

Feature #5: Personalization

Feature #6: ADS ingredient

## 1.3 Work Product to  be developed

### 1.3.1 Deliverable

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Deriverable/Release | Media | Copies | Date |
| 1 | Project proposal  - Spotifood\_proposal3 | Hard copy | 3 | 19/06/2019 |
| 2 | Progress report I  - Project management plan version 1.0  - Software requirement specification version 1.0  - Software design document version 1.0  - Test plan version 1.0  - Traceability record version 1.0  - Software version 0.1 | Hard copy | 3 | 29/07/2019 |

## 1.4 Acronyms and Definitions

### 1.4.1 Acronyms

UC Use case

URS User Requirement Specification

SRS Software Requirement Specification

CRUD Create-Read-Update-Delete

AI Artificial Intelligence

ADS Advertising

ID Identification Data

# CHAPTER 2 | Overall Description

## 2.1 Product Perspective

Spotifood contain two parts. First is web application and second is AI. Web application use for interaction. Then AI will retrieve data from web application and send back to web application again.

## 2.2 Product Function

Spotifood have five major parts.

* Authentication system

This feature is used for preventing the unwanted user to access some feature in the system.

* Content CRUD

User input ingredients and how-to-cook step in this form. Then the user shares their dish into the app page.And Feature is used to pin dishes(Dish that user interest and pin it) and user’s history dishes(Dish that user view it already).

* User management

This feature is used to show the dish details including ingredients and hot-to-cook step. And used to edit their personal information.

* Similar dish identification

Registered user search by name of the dish and show the dishes that similar to registered user searched.

* Personalization

This feature is used for suggestion a dish by type of food and word of dish that frequently searched.

* ADS ingredient

This function is used to help a registered user find ingredients that registered user can’t cook or main ingredients to use in the dishes. By using a sponsored ranking example, the shop that sold the ingredients is most popular or pay a lot to the system will show first. The shop info comes from website that we can contact shop allow to show.

## 2.3 User Characteristics

There are three types of users in Spotifood application: general, registered user, admin.

General users are the users of the web application. Which can use a few functions. That is register and sign in.

Registered users are the users of the web application. Which can use most of the functions. That is sign in, create post, edit profile, delete post, delete profile, and view dish.

Admins are the administrator of the web application. Admin can manage post and manage user.

# CHAPTER 3 | USER REQUIREMENT SPECIFICATION

**Progress 1**

**Feature#1:** Authentication system

USER: General user, Registered user

URS01- User can register to the system by input email, password, name, date of birth, and gender.

URS02 - User can sign in to the system by input email and password.

URS03 - User can sign out of the system by sign out nav bar.

USER: Admin

URS04 - Admin can sign in into the system by input email and password.

URS05 - Admin can sign out of the system by sign out nav bar.

**Feature#2**: Content CRUD

USER: Registered user

URS06 - User can post the dish by input name of dishes, ingredients, how-to-cook of the dish, and thumbnail photos of the dish.

URS07 - User can delete the post of the dish.

URS08 - User can view a pin dish.

URS09 - User can view a history dish.

URS10 - User can select topic or type of food that user interested.

URS15 - User can edit their post.

**Feature#3**: User management

USER: Registered user

URS11- User can view a dish that user want.

URS12 - User can update their name, about, email and password.

URS13- User or upload and update their profile picture.

URS14- User can update their personality of interesting food.

URS16 - User can view their profile page.

# CHAPTER 4 | SOFTWARE REQUIREMENT SPECIFICATION

**Use case**

**UC01:** User can register to the system.

**UC02:** User can sign in to the system.

**UC03:** User can sign out of the system.

**UC04:** Admin can sign in to the system.

**UC05:** Admin can sign out of the system.

**UC06:** User can update their profile.

**UC07:** User can update their personality of interesting food.

**UC08:** Edit profile picture.

**UC09:** User can post the dish.

**UC10:** User can delete the posted dish.

**UC11:** User can like dish.

**UC12:** User can unlike dish.

**UC13:** User can view like dish

**UC14:** User can view history.

**UC15:** User can select topic.

**UC16:** User can view dish.

**UC17:** User can edit their post.

**UC18:** User can view profile page.

**Feature#1:** Authentication system

**URS01** - User can register to the system by input email, password,name,date of birth, and gender.

SRS-01: The system shall provide the user interface for receiving an email, password,  name, and gender.

SRS-02: The system shall provide an error message “This field can not empty” when the textbox to empty.

SRS-03: The system shall provide a button to register.

SRS-04: The system shall validate the email, password, name, and gender of the user.

SRS-05: The system shall provide an error message “This [input] is incorrect format.” if input incorrect format of email, password, name, and gender.

SRS-06: The system shall provide an error message “This email already exists.” if the input email is already in database.

SRS-07: The system shall display a message “New account is successfully created. Please Sign In”.

SRS-66: The system shall update data in the database.

**URS02** - User can sign in to the system by input email and password.

SRS-08: The system shall provide the user interface for receiving email and password.

SRS-09: The system shall provide a button to sign in.

SRS-10: The system shall validate the email and password.

SRS-11: The system shall provide an error message “Incorrect password” if email is exist but password not match.

SRS-12: The system shall provide an error message “This email does not exist” if email does not exist in the database.

SRS-13: The system redirects user to homepage.

**URS03** - User can sign out of the system by sign out button.

SRS-14: The system shall provide a button to sign out.

SRS-15: The system shall be logged out.

SRS-16: The system redirects user to the home page.

**URS04** - Admin can sign in into the system by input email and password.

SRS-17: The system shall provide a user interface for receiving email and password.

SRS-18: The system shall validate the email and password.

SRS-19: The system shall provide an error message “Incorrect password” if email is exist but password not match.

SRS-20: The system shall provide an error message “This email does not exist” if email does not exist in the database.

SRS-21: The system redirects admin to homepage.

**URS05** - Admin can sign out of the system by sign out nav bar.

SRS-22: The system shall provide a button to sign out.

SRS-23: The system shall be logged out.

SRS-24: The system redirects admin to the home page.

**Feature #2**: Content CRUD

**URS06** - User can post the dish by input name of dishes, ingredients, how-to-cook of the dish, and thumbnail photos of the dish.

SRS-25: The system shall provide the user interface for receiving data of the dish including :

- Name of dish

- Ingredients

- How-to-cook of the dish

- Upload thumbnail photos of food

SRS-26: The system shall provide a button to post.

SRS-27: The system shall validate the name of the dish, ingredients, and how-to-cook of the dish, and thumbnail photos of the dish.

SRS-28: The system shall provide an error message “This [input] field can not empty” when the textbox to empty.

SRS-29: The system shall provide an error message “File size should be less than 150kb” when file size is more than 150kb.

SRS-30: The system redirect registered user to the user’s profile.

SRS-66: The system shall update data in the database.

**URS07** - User can delete the post of the dish.

SRS-31: The system shall provide the delete button for the user.

SRS-32: The system shall provide the confirm button for the user after click delete button.

SRS-33: The system shall delete the dish when the user clicks the delete button and confirm to delete the dish.

SRS-34: The system redirect registered user to homepage.

SRS-35: The system shall alert message “Are you sure to delete the post?”.

SRS-66: The system shall update data in the database.

**URS08** - User can view a pin dish.

SRS-36: The system shall provide the pin button for the user.

SRS-37: The system shall cancel the pin dish when the user clicks a pin button again.

SRS-38: The system shall update the data into the database after the user clicks pin button.

SRS-39: The system shall provide the user interface to show a pin dish.

SRS-65: The system will retrieve data in database.

**URS09** - User can view a history dish.

SRS-40: The system shall provide the history dish button to show a history dish.

SRS-41: The system shall redirect to history dish page when the user clicks the history button.

SRS-65: The system will retrieve data in database.

**URS10** - User can select topic or type of food that user interested.

SRS-42: The system shall provide the user interface for receiving a type of food that user interested.

SRS-65: The system will retrieve data in database.

**URS11** - User can view a dish that user want.

SRS-67: The registered user click a name of the dish.

SRS-65: The system will retrieve data in database.

SRS-68: The system shall redirect user to view dish page.

**Feature#3**: User management

**URS12** - User can update their name, about, email and password.

SRS-46: The system shall provide the user interface for editing data of the dish including :

- Name

- About

- Email

- Password

SRS-47: The system shall provide the edit button to the user.

SRS-48: The system shall validate name , about , email and password.

SRS-49: The system saves the new data of user management to the database after the user input all data validation is passed.

SRS-50: The system shall provide an error message “This [input] field can not empty” when the textbox to empty.

SRS-51: The system shall provide an error message “A valid email is required” when the textbox is invalid format. (ex. dummy123@hotmail.com)

SRS-52: The system shall provide an error message “Email is taken!”. if input email is already in database.

SRS-66: The system shall update data in the database.

SRS-53: The system redirect registered user to profile page.

**URS13** - User or upload and update their profile picture.

SRS-59:  The system shall provide the user interface for upload profile picture.

SRS-60: The system shall provide the upload button to the user.

SRS-61: The system shall provide an error message “File size should be less than 150kb” when file size is more than 100kb.

SRS-66: The system shall update data in the database.

SRS-62: The system redirects user to user’s profile page.

**URS14** - User can update their personality of interesting food.

SRS-54: The system shall provide the user interface for personality of interesting food.

SRS-55: User can select tag that user want to update on personality. And user click selected tag again to unselect.

SRS-56: The system shall provide a submit button to the user.

SRS-57: The system saves the new data of User management to the database.

SRS-58: The system redirects user to the homepage.

SRS-66: The system shall update data in the database.

**URS15** - User can edit their post.

SRS-25: The system shall provide the user interface for receiving data of the dish including :

- Name of dish

- Ingredients

- How-to-cook of the dish

- Upload thumbnail photos of food

SRS-51: The system shall provide edit button.

SRS-28: The system shall provide an error message “This [input] field can not empty” when the textbox to empty.

SRS-45: The system shall provide an error message “File size should be less than 150kb” when file size is more than 150kb.

SRS-32: The system redirect registered user to the user’s profile.

SRS-66: The system shall update data in the database.

**URS16** - User can view their profile page.

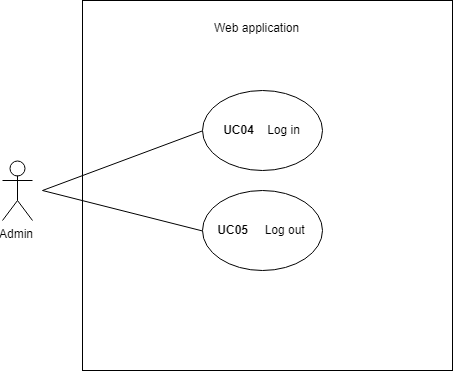
SRS-63: The system shall provide profile nav bar on the web application.

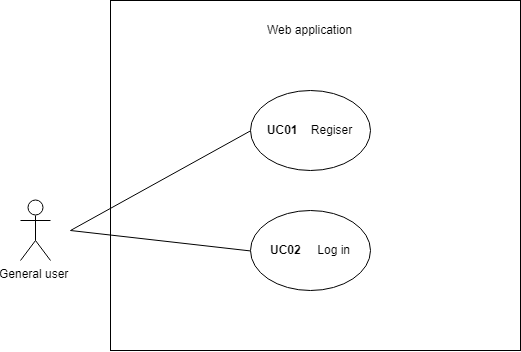
SRS-64: The system shall redirect user to their profile page.

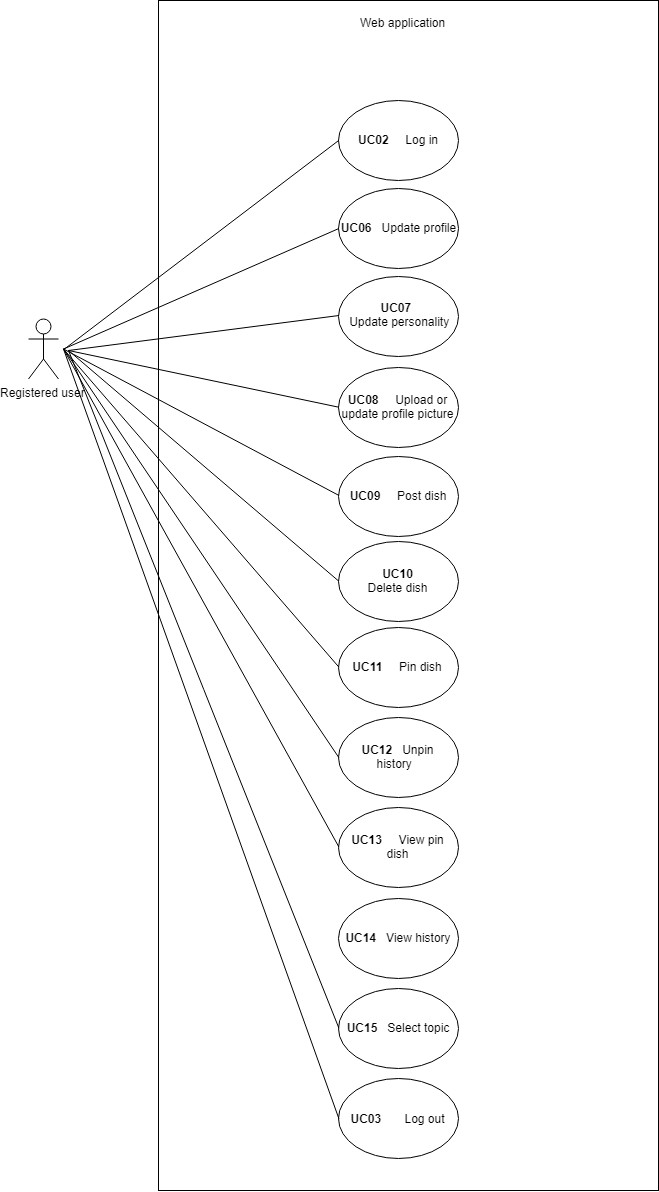
SRS-65: The system will retrieve data in database.

**Requirements Specifications**

Use case diagram







**Use case Description**

**Feature#1:** Authentication system

Description: This feature is used for preventing the unwanted user to access some feature in the system.

Actor: Admin, Registered system

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC01 | | |
| Use Case Name: | Register | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/05/2019 | Last Revision Date: | 07/20/2019 |
| Actor: | General user | | |
| Description: | General user can register to have their own account. | | |
| Trigger: | General user clicks the register nav bar on the web application. | | |
| Precondition: | General user open Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| Email | String | - This field can not empty.  - Have to input correct format. | temabc123@hotmail.com |
| Password | String | - This field can not empty.  - Have to input correct format. | passcode123 |
| Name | String | - This field can not empty.  - Have to input correct format. | Kittinut |
| Gender | String | - This field can not empty.  - Have to input correct format. | Male |
| DateOfBirth | integer | - This field can not empty | 01/01/1111 |
| Post Condition: | General user has an account for sign in. | | |
| Normal Flow: | 1. General user clicks the register bar. 2. The system will redirect the general user to register page. 3. General user input name. 4. General user input date of birth. 5. General user input gender. 6. General user input email. 7. General user input password. 8. General user click submit button. 9. The system validates the input. 10. The system will save account to the database.[A1][A2][A3] 11. The system shall display a message “New account is successfully created. Please Sign In”. | | |
| Alternative Flow: | [A1] If the text field is empty.   1. The system shall provide an error message “This [input] field can not empty”. 2. Back to normal flow 3.   [A2] If the email is incorrect format.   1. The system shall provide an error message “Input a valid email”. 2. Back to normal flow 6.   [A3] If the email already exists.   1. The system shall provide an error message “This email is taken!”. 2. Back to normal flow 6. | | |
| Exception Flow: | - | | |
| Assumption: | 1. General user must connect with the internet. | | |

Related Requirements

**URS01** - User can register to the system by input email, password,name,date of birth, and gender.

SRS-01: The system shall provide the user interface for receiving an email, password,  name, and gender.

SRS-02: The system shall provide an error message “This field can not empty” when the textbox to empty.

SRS-03: The system shall provide a button to register.

SRS-04: The system shall validate the email, password, name, and gender of the user.

SRS-05: The system shall provide an error message “This [input] is incorrect format.” if input incorrect format of email, password, name, and gender.

SRS-06: The system shall provide an error message “This email already exists.” if the input email is already in database.

SRS-07: The system shall display a message “New account is successfully created. Please Sign In”.

SRS-66: The system shall update data in the database.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC02 | | |
| Use Case Name: | User sign in | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/05/2019 | Last Revision Date: | 07/20/2019 |
| Actor: | Registered user | | |
| Description: | The registered user can sign in into the system with their own account. | | |
| Trigger: | The registered user click the sign in nav bar on the web application. | | |
| Precondition: | The registered user open Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| Email | String | - This field can not empty.  - Have to input correct format. | temabc123@hotmail.com |
| Password | String | - This field can not empty.  - Have to input correct format. | passcode123 |
| Post Condition: | The registered user sign in into the system. | | |
| Normal Flow: | 1. Registered user click sign in nav bar. 2. The system shall redirect registered user to the sign in page. 3. The registered user input email. 4. The registered user input password. 5. The registered user input click the sign in button. 6. The system validates the account. [A1][A2] 7. The system redirect registered user to the homepage. | | |
| Alternative Flow: | [A1] If the email does not exist in the database.   1. The system shall provide a message “This email does not exist” 2. Go back to normal flow 4.   [A2] If the password does not correct.   1. The system shall provide a message “Incorrect password” 2. Go back to normal flow 4. | | |
| Exception Flow: | - | | |
| Assumption: | 1. Registered user must connect to the internet. | | |

Related Requirements

**URS02** - User can sign in to the system by input email and password.

SRS-08: The system shall provide the user interface for receiving email and password.

SRS-09: The system shall provide a button to sign in.

SRS-10: The system shall validate the email and password.

SRS-11: The system shall provide an error message “Incorrect password” if email is exist but password not match.

SRS-12: The system shall provide an error message “This email does not exist” if email does not exist in the database.

SRS-13: The system redirects user to homepage.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC03 | | |
| Use Case Name: | User sign out | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/05/2019 | Last Revision Date: | 07/20/2019 |
| Actor: | Registered user | | |
| Description: | The registered user can sign out of the system. | | |
| Trigger: | The registered user click the sign out nav bar on the web application. | | |
| Precondition: | The registered user must sign in to the Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| - | - | - | - |
| Post Condition: | The registered user sign out of the system. | | |
| Normal Flow: | 1. The registered user clicks the sign out button. 2. The system redirect registered user to the home page. | | |
| Alternative Flow: | - | | |
| Exception Flow: | - | | |
| Assumption: | 1. Registered user must connect to the internet. | | |

Related Requirements

**URS03** - User can sign out of the system by sign out nav bar.

SRS-14: The system shall provide a button to sign out.

SRS-15: The system shall be logged out.

SRS-16: The system redirects user to the home page.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC04 | | |
| Use Case Name: | Admin sign in | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/05/2019 | Last Revision Date: | 07/05/2019 |
| Actor: | Admin | | |
| Description: | The admin can sign in into the system with their own account. | | |
| Trigger: | The admin click the sign in nav bar on the web application. | | |
| Precondition: | The admin open Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| Email | String | - This field can not empty.  - Have to input correct format. | temabc123@hotmail.com |
| Password | String | - This field can not empty.  - Have to input correct format. | passcode123 |
| Post Condition: | The admin sign in into the system. | | |
| Normal Flow: | 1. Admin click sign in nav bar. 2. The system shall redirect admin to the sign in page. 3. The admin input email. 4. The admin input password. 5. The admin input click the sign in button. 6. The system validates the account. [A1][A2] 7. The system redirect admin to the homepage. | | |
| Alternative Flow: | [A1] If the email does not exist in the database.   1. The system shall provide a message “This email does not exist” 2. Go back to normal flow 4.   [A2] If the password does not correct.   1. The system shall provide a message “Incorrect password” 2. Go back to normal flow 4. | | |
| Exception Flow: | - | | |
| Assumption: | 1. The admin must connect to the internet. | | |

Related Requirements

**URS04** - Admin can sign in into the system by input email and password.

SRS-17: The system shall provide a user interface for receiving email and password.

SRS-18: The system shall validate the email and password.

SRS-19: The system shall provide an error message “Incorrect password” if email is exist but password not match.

SRS-20: The system shall provide an error message “This email does not exist” if email does not exist in the database.

SRS-21: The system redirect admin to homepage.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC05 | | |
| Use Case Name: | Admin sign out | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/05/2019 | Last Revision Date: | 07/05/2019 |
| Actor: | Admin | | |
| Description: | The admin can sign out of the system. | | |
| Trigger: | The admin  click the sign out nav bar on the web application. | | |
| Precondition: | The admin must sign in to the Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| - | - | - | - |
| Post Condition: | The admin sign out of the system. | | |
| Normal Flow: | 1. The admin click sign out nav bar. 2. The system redirect admin to the sign in page. | | |
| Alternative Flow: | - | | |
| Exception Flow: | - | | |
| Assumption: | 1. Registered user must connect to the internet. | | |

Related Requirements

**URS05** - Admin can sign out of the system by sign out nav bar.

SRS-22: The system shall provide a button to sign out.

SRS-23: The system shall be logged out.

SRS-24: The system redirects admin to the home page.

**Feature#2**: Content CRUD

Description:  User input ingredients and how-to-cook step in this feature. Then the user shares their dish into the app page. And feature is used to pin dishes (Dish that user interest and pin it) and user’s history dishes (Dish that user view it already).

Actor: Registered user

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC09 | | |
| Use Case Name: | Post dish | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/05/2019 | Last Revision Date: | 07/05/2019 |
| Actor: | Registered user | | |
| Description: | User can post the dish by input name of dishes, ingredients, and how-to-cook of the dish, tag of the dish, upload their insert photo of each step and thumbnail photos of the dish. | | |
| Trigger: | The registered user clicks the create post nav bar on the web application. | | |
| Precondition: | The registered user must sign in to the Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| nameOfDish | String | This field can not empty. | Omelet |
| ingredient | String | This field can not empty. | Egg, Olive oil, Fish sauce |
| howToCook | String | This field can not empty. | 1. Break the egg and put it into a bowl.  2. Add fish sauce 1 tablespoon.  3. Mix it together  4. Put olive oil into the pan and use medium heat.  5. Put a mixed egg into the pan.  6. Wait until the below side looks cooked then turn it back.  7. Wait until all side of omelet is looking perfect.  8. Put it into the dish and ready to eat. |
| Post Condition: | A new dish saves into the database. | | |
| Normal Flow: | 1. The registered user clicks create post nav bar. 2. The system redirect registered user to the post dish page. 3. The registered user input the name of the dish. 4. The registered user input ingredients. 5. The registered user input how-to-cook of the dish. 6. The registered user uploads thumbnail photos of food.[A1] 7. The registered user clicks create post button. 8. The system will check the text box. [A2] 9. The system updates the database. 10. The system redirect registered user to the posted dish view page. | | |
| Alternative Flow: | [A1] If the photo size is more than 150 kb.   1. The system displays a message “File size should be less than 150kb”. 2. Go to normal flow 7.   [A2] If the textbox is empty.   1. The system displays a message “This [input] field can not empty”. 2. Go to normal flow 3. | | |
| Exception Flow: | - | | |
| Assumption: | 1. Registered user must connect to the internet. | | |

Related Requirements

**URS06** - User can post the dish by input name of dishes, ingredients, how-to-cook of the dish, and thumbnail photos of the dish.

SRS-25: The system shall provide the user interface for receiving data of the dish including :

- Name of dish

- Ingredients

- How-to-cook of the dish

- Upload thumbnail photos of food

SRS-26: The system shall provide a button to post.

SRS-27: The system shall validate the name of the dish, ingredients, and how-to-cook of the dish, and thumbnail photos of the dish.

SRS-28: The system shall provide an error message “This [input] field can not empty” when the textbox to empty.

SRS-29: The system shall provide an error message “File size should be less than 150kb” when file size is more than 150kb.

SRS-30: The system redirect registered user to the user’s profile.

SRS-66: The system shall update data in the database.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC10 | | |
| Use Case Name: | Delete dish | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/06/2019 | Last Revision Date: | 07/06/2019 |
| Actor: | Registered user | | |
| Description: | The registered user uses this function to delete their own posts. | | |
| Trigger: | The registered user clicks name of their own post. | | |
| Precondition: | The registered user must sign in to the Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| - | - | - | - |
| Post Condition: | A selected dish delete from the database. | | |
| Normal Flow: | 1. The registered user clicks name of their own post. 2. The registered user click delete post button. 3. The system shall display a message “Are you sure to delete the post?”[A1][A2] 4. The system update data in the database. 5. The system redirect registered user to the homepage. | | |
| Alternative Flow: | [A1] If registered user clicks the Yes button.   1. Go to normal flow 3.   [A2] If registered user click No button.   1. Go to normal flow 5. | | |
| Exception Flow: | - | | |
| Assumption: | 1. Registered user must connect to the internet. | | |

Related Requirements

**URS07** - User can delete the post of the dish.

SRS-31: The system shall provide the delete button for the user.

SRS-32: The system shall provide the confirm button for the user after click delete button.

SRS-33: The system shall delete the dish when the user clicks the delete button and confirm to delete the dish.

SRS-34: The system redirect registered user to homepage.

SRS-35: The system shall alert message “Are you sure to delete the post?”.

SRS-66: The system shall update data in the database.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC11 | | |
| Use Case Name: | Like dish | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/06/2019 | Last Revision Date: | 07/06/2019 |
| Actor: | Registered user | | |
| Description: | The registered user can pin the dish that they like. | | |
| Trigger: | The registered user clicks name of dish to view dish. | | |
| Precondition: | The registered user must sign in to the Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| - | - | - | - |
| Post Condition: | The registered user will have a new dish in their pin dish view. | | |
| Normal Flow: | 1. The registered user click name of the dish. 2. The system redirect user to view dish page. 3. The registered user clicks like button on the view dish page. 4. The system will update data in the database. | | |
| Alternative Flow: | - | | |
| Exception Flow: | - | | |
| Assumption: | 1. Registered user must connect to the internet. | | |

Related Requirements

**URS08** - User can view a like dish.

SRS-36: The system shall provide the like button for the user.

SRS-37: The system shall unlike the dish when the user clicks a like button again.

SRS-38: The system shall update the data into the database after the user clicks pin button.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC12 | | |
| Use Case Name: | Unlike dish | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/06/2019 | Last Revision Date: | 07/06/2019 |
| Actor: | Registered user | | |
| Description: | The registered user can unlike the dish that they do not like it anymore. | | |
| Trigger: | The registered user clicks name of dish to view dish. | | |
| Precondition: | The registered user must sign in to the Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| - | - | - | - |
| Post Condition: | The registered user will remove the selected dish in their view like dish. | | |
| Normal Flow: | 1. The registered user click name of the dish. 2. The system redirect user to view dish page. 3. The registered user clicks like button to unlike on the view dish page. 4. The system will update data in the database. | | |
| Alternative Flow: | - | | |
| Exception Flow: | - | | |
| Assumption: | 1. Registered user must connect to the internet. | | |

Related Requirements

**URS08** - User can view a like dish.

SRS-36: The system shall provide the like button for the user.

SRS-37: The system shall unlike the dish when the user clicks a like button again.

SRS-38: The system shall update the data into the database after the user clicks pin button.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC13 | | |
| Use Case Name: | View like dish | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/06/2019 | Last Revision Date: | 07/06/2019 |
| Actor: | Registered user | | |
| Description: | The registered user can view like dish on the Spotifood web application. | | |
| Trigger: | The registered user clicks view like dish nav bar. | | |
| Precondition: | The registered user must sign in to the Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| - | - | - | - |
| Post Condition: | The registered user will see all the dish that they pin. | | |
| Normal Flow: | 1. The registered user clicks view pin dish button. 2. The system retrieve data in database. 3. The system redirects the registered user to view pin dish page. | | |
| Alternative Flow: | - | | |
| Exception Flow: | - | | |
| Assumption: | 1. Registered user must connect to the internet. | | |

Related Requirements

**URS08** - User can view a pin dish.

SRS-39: The system shall provide the user interface to show a pin dish.

SRS-65: The system will retrieve data in database.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC14 | | |
| Use Case Name: | View history | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/06/2019 | Last Revision Date: | 07/06/2019 |
| Actor: | Registered user | | |
| Description: | The registered user can view the history of their account. Only the dish that they ever click in the past. | | |
| Trigger: | The registered user clicks history nav bar. | | |
| Precondition: | The registered user must sign in to the Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| - | - | - | - |
| Post Condition: | The registered user will see all the dish that they ever click on it. | | |
| Normal Flow: | 1. The registered user clicks history nav bar. 2. The system retrieve data in database. 3. The system will redirect registered user to history page. | | |
| Alternative Flow: | - | | |
| Exception Flow: | - | | |
| Assumption: | 1. Registered user must connect to the internet. | | |

Related Requirements

**URS09** - User can view a history dish.

SRS-40: The system shall provide the history dish button to show a history dish.

SRS-41: The system shall redirect to history dish page when the user clicks the history button.

SRS-65: The system will retrieve data in database.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC15 | | |
| Use Case Name: | Select topic | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/06/2019 | Last Revision Date: | 07/06/2019 |
| Actor: | Registered user | | |
| Description: | The registered user can select the type of food that user interest. And system will show the dish that tag the same as that type of food. | | |
| Trigger: | The registered user clicks select topic nav bar. | | |
| Precondition: | The registered user must sign in to the Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| - | - | - | - |
| Post Condition: | The registered user will see all the dishes have the same tag as a topic that they select. | | |
| Normal Flow: | 1. The registered user click topic nav bar. 2. The system redirects the user to select topic page. 3. The registered user click topic that they want. 4. The system retrieve data in the database. 5. The system shows all dishes that have the tag same as a selected topic. | | |
| Alternative Flow: | - | | |
| Exception Flow: | - | | |
| Assumption: | 1. Registered user must connect to the internet. | | |

Related Requirements

**URS10** - User can select topic or type of food that user interested.

SRS-42: The system shall provide the user interface for receiving a type of food that user interested.

SRS-65: The system will retrieve data in database.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC17 | | |
| Use Case Name: | Edit post | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/10/2019 | Last Revision Date: | 07/10/2019 |
| Actor: | Registered user | | |
| Description: | User can edit their post the dish by input name of dishes, ingredients, and how-to-cook of the dish, tag of the dish, upload their insert photo of each step and thumbnail photos of the dish. | | |
| Trigger: | The registered user clicks the edit post button on the web application. | | |
| Precondition: | The registered user must sign in to the Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| nameOfDish | String | This field can not empty. | Omelet |
| ingredient | String | This field can not empty. | Egg, Olive oil, Fish sauce |
| howToCook | String | This field can not empty. | 1. Break the egg and put it into a bowl.  2. Add fish sauce 1 tablespoon.  3. Mix it together  4. Put olive oil into the pan and use medium heat.  5. Put a mixed egg into the pan.  6. Wait until the below side looks cooked then turn it back.  7. Wait until all side of omelet is looking perfect.  8. Put it into the dish and ready to eat. |
| Post Condition: | A new dish saves into the database. | | |
| Normal Flow: | 1. The registered user clicks edit post button. 2. The system redirect registered user to the edit dish page. 3. The registered user input name of the dish. 4. The registered user input ingredients. 5. The registered user input how-to-cook of the dish. 6. The registered user uploads thumbnail photos of food.[A1][A2][A3] 7. The registered user clicks the edit button. 8. The system will check the text box. [A3] 9. The system updates the database. 10. The system redirect registered user to user’s profile page. | | |
| Alternative Flow: | [A1] If the photo does not exist.   1. The system displays a message “Select a file to upload photo.”. 2. Go to normal flow 7.   [A2] If the photo size is more than 150 kb.   1. The system displays a message “Photo size is more than 150 kb.”. 2. Go to normal flow 7.   [A3] If the textbox is empty.   1. The system displays a message “This [input] field can not empty”. 2. Go to normal flow 3. | | |
| Exception Flow: | - | | |
| Assumption: | 1. Registered user must connect to the internet. | | |

Related Requirements

**URS15** - User can edit their post.

SRS-25: The system shall provide the user interface for receiving data of the dish including :

- Name of dish

- Ingredients

- How-to-cook of the dish

- Upload thumbnail photos of food

SRS-51: The system shall provide edit button.

SRS-28: The system shall provide an error message “This [input] field can not empty” when the textbox to empty.

SRS-45: The system shall provide an error message “File size should be less than 150kb” when file size is more than 150kb.

SRS-32: The system redirect registered user to the user’s profile.

SRS-66: The system shall update data in the database.

**Feature#3**: User management

Description: This feature is used to post the dish details including ingredients and how-to-cook step. And used to edit their personal information.

Actor: Registered system

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC06 | | |
| Use Case Name: | Edit profile | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/05/2019 | Last Revision Date: | 07/21/2019 |
| Actor: | Registered user | | |
| Description: | The registered user can update their own profile. Update email, password, name, gender, and profile picture. | | |
| Trigger: | The registered user click the edit profile button on view profile page. | | |
| Precondition: | The registered user must sign in to the Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| Email | String | - This field can not empty.  - Have to input correct format. | temabc123@hotmail.com |
| Password | String | - This field can not empty.  - Have to input more than 6 characters. | passcode123 |
| name | String | - This field can not empty. | Kittinut |
| About | String | - | Hi. I love sushi. |
| Post Condition: | Profile of registered user has been updated. | | |
| Normal Flow: | 1. Registered user clicks edit profile button. 2. The system redirect registered user to the edit profile page. 3. Registered user input name. 4. Registered user input about. 5. Registered user input email. 6. Registered user input password. 7. Registered user clicks the update button. 8. The system will update the profile in the database.[A1][A2][A3] 9. The system will redirect the registered user to the homepage. | | |
| Alternative Flow: | [A1] If the text field is empty.   1. The system shall provide an error message “This [input] field can not empty”. 2. Back to normal flow 4.   [A2] If the input is incorrect format.   1. The system shall provide an error message “A valid email is required”. 2. Back to normal flow 4.   [A3] If the email already exists.   1. The system shall provide an error message “Email is taken!”. 2. Back to normal flow 4. | | |
| Exception Flow: | - | | |
| Assumption: | 1. Registered user must connect to the internet. | | |

Related Requirements

**URS12** - User can update their name, about, email and password.

SRS-46: The system shall provide the user interface for editing data of the dish including :

- Name

- About

- Email

- Password

SRS-47: The system shall provide the edit button to the user.

SRS-48: The system shall validate name , about , email and password.

SRS-49: The system saves the new data of user management to the database after the user input all data validation is passed.

SRS-50: The system shall provide an error message “This [input] field can not empty” when the textbox to empty.

SRS-51: The system shall provide an error message “A valid email is required” when the textbox is invalid format. (ex. dummy123@hotmail.com)

SRS-52: The system shall provide an error message “Email is taken!”. if input email is already in database.

SRS-66: The system shall update data in the database.

SRS-53: The system redirect registered user to profile page.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC07 | | |
| Use Case Name: | Edit personality | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/07/2019 | Last Revision Date: | 07/08/2019 |
| Actor: | Registered user | | |
| Description: | The registered user can update their personality of interesting food. | | |
| Trigger: | The registered user click the personality nav bar on the web application. | | |
| Precondition: | The registered user must sign in to the Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| - | - | - | - |
| Post Condition: | Personality of registered user has been updated. | | |
| Normal Flow: | 1. The registered user click the personality nav bar. 2. The system shall provide user interface for personality of interesting food. 3. User click tag that user want to update on personality of food. [A1] 4. The registered user click submit button. [A2] 5. The system saves the new data of user management to the database. 6. The system redirects user to the homepage. | | |
| Alternative Flow: | [A1] If registered user click tag again.   1. Tag will unselect. 2. Go to normal flow 3.    [A2] If registered user click cancel button.   1. Go to normal flow 2. | | |
| Exception Flow: | - | | |
| Assumption: | 1. Registered user must connect to the internet. | | |

Related Requirements

**URS14** - User can update their personality of interesting food.

SRS-54: The system shall provide the user interface for personality of interesting food.

SRS-55: User can select tag that user want to update on personality. And user click selected tag again to unselect.

SRS-56: The system shall provide a submit button to the user.

SRS-57: The system saves the new data of User management to the database.

SRS-58: The system redirects user to the homepage.

SRS-66: The system shall update data in the database.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC08 | | |
| Use Case Name: | Edit profile picture. | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/05/2019 | Last Revision Date: | 07/05/2019 |
| Actor: | Registered user | | |
| Description: | The registered user can upload their profile picture on profile page. | | |
| Trigger: | The registered user click the profile nav bar on the web application. | | |
| Precondition: | The registered user or admin must sign in to the Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| - | - | - | - |
| Post Condition: | Profile picture of registered user has been updated. | | |
| Normal Flow: | 1. Registered user clicks the profile nav bar. 2. Registered user clicks the edit button. 3. Registered user click select file button. 4. The system validates file. [A1] 5. Registered user click update button. 6. The system update the database. 7. The system redirects user to user’s profile page. | | |
| Alternative Flow: | [A1] If picture file is more than 100kb.   1. The system shall provide an error message “File size should be less than 100kb”. 2. Go to normal flow 2. | | |
| Exception Flow: | - | | |
| Assumption: | 1. Registered user must connect to the internet. | | |

Related Requirements

**URS13** - User or upload and update their profile picture.

SRS-59:  The system shall provide the user interface for upload profile picture.

SRS-60: The system shall provide the upload button to the user.

SRS-61: The system shall provide an error message “File size should be less than 150kb” when file size is more than 100kb.

SRS-66: The system shall update data in the database.

SRS-62: The system redirects user to user’s profile page.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC16 | | |
| Use Case Name: | View dish. | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/08/2019 | Last Revision Date: | 07/08/2019 |
| Actor: | Registered user | | |
| Description: | The registered user can view dish. | | |
| Trigger: | The registered user click on name of dish. | | |
| Precondition: | The registered user must sign in to the Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| - | - | - | - |
| Post Condition: | User will redirect to selected view dish. | | |
| Normal Flow: | 1. Registered user click on any dish. 2. The system will retrieve data in database. 3. The system redirects user to view page of the dish. | | |
| Alternative Flow: | - | | |
| Exception Flow: | - | | |
| Assumption: | 1. Registered user must connect to the internet. | | |

Related Requirements

**URS11** - User can view a dish that user want.

SRS-67: The registered user click a name of the dish.

SRS-65: The system will retrieve data in database.

SRS-68: The system shall redirect user to view dish page.

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case ID: | UC18 | | |
| Use Case Name: | View profile. | | |
| Create By: | Kittinut Saengsri | Last Update By: | Kittinut Saengsri |
| Date Create: | 07/08/2019 | Last Revision Date: | 07/22/2019 |
| Actor: | Registered user | | |
| Description: | The registered user can view their profile page. | | |
| Trigger: | The registered user click on profile nav bar. | | |
| Precondition: | The registered user must sign in to the Spotifood web application. | | |
| Use Case Input Specification | | | |
| Input | Type | Constrain | Example |
| - | - | - | - |
| Post Condition: | Registered will redirect to their profile page. | | |
| Normal Flow: | 1. Registered user click on profile nav bar. 2. The system retrieve data in database. 3. The system redirect user to their profile page. | | |
| Alternative Flow: | - | | |
| Exception Flow: | - | | |
| Assumption: | 1. Registered user must connect to the internet. | | |

Related Requirements

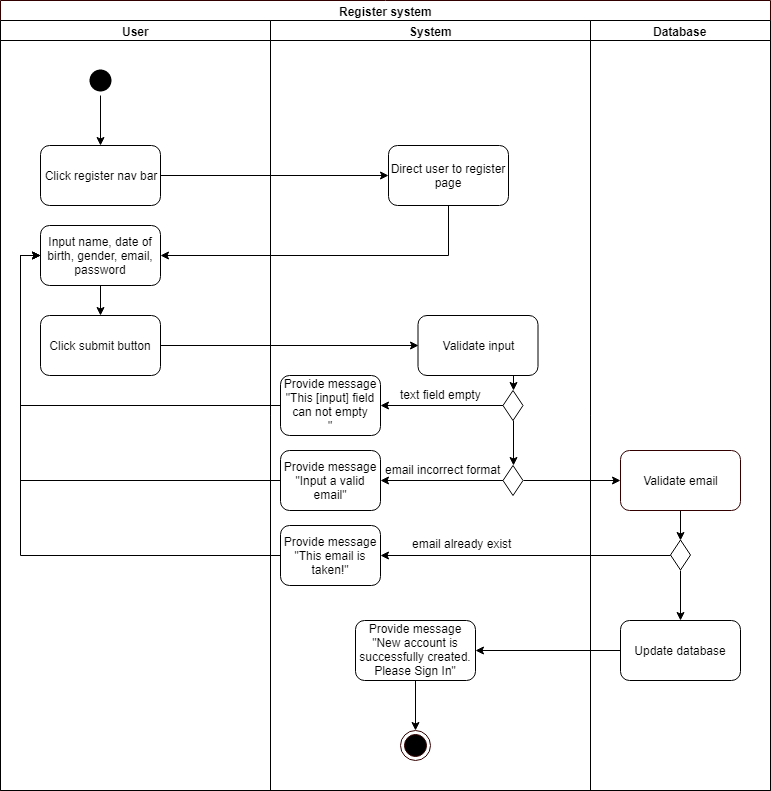
**URS16** - User can view their profile page.

SRS-63: The system shall provide profile nav bar on the web application.

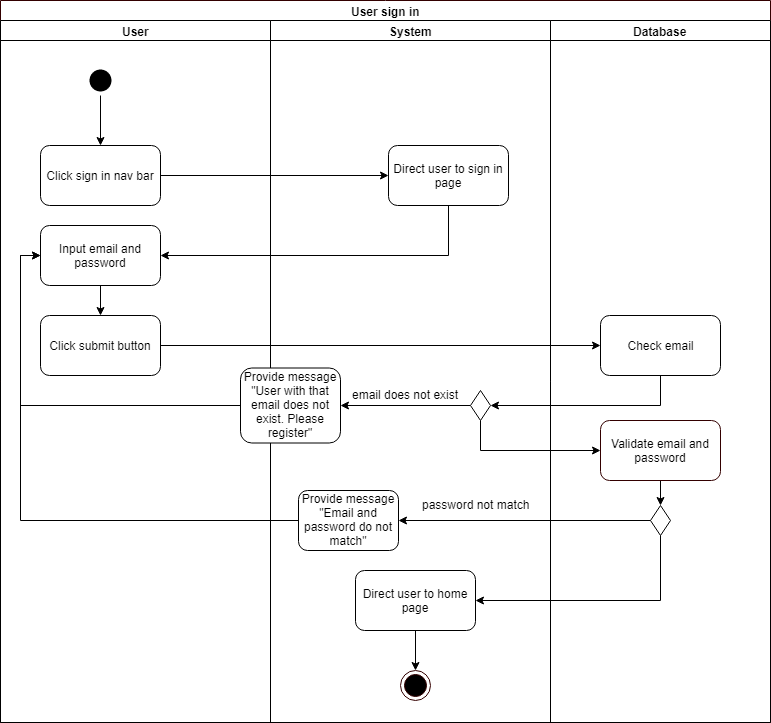
SRS-64: The system shall redirect user to their profile page.

SRS-65: The system will retrieve data in database.

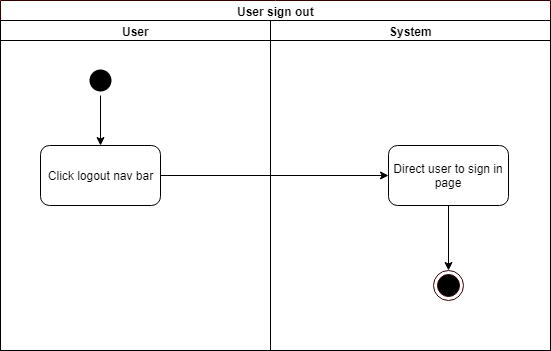
Activity diagram 1: AC01



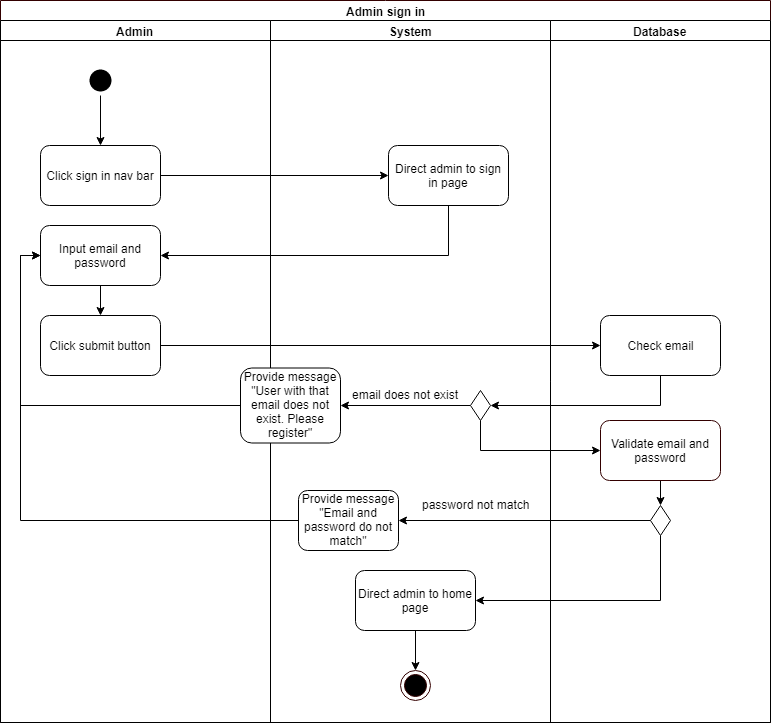
Activity diagram 2: AC02



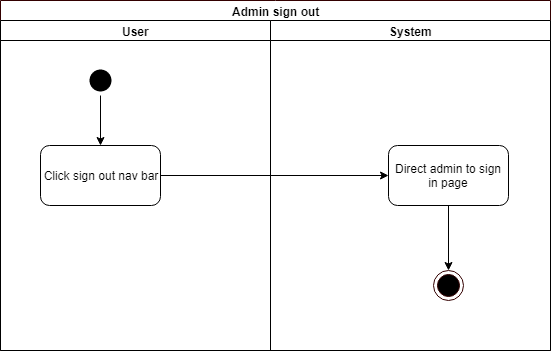
Activity diagram 3: AC03



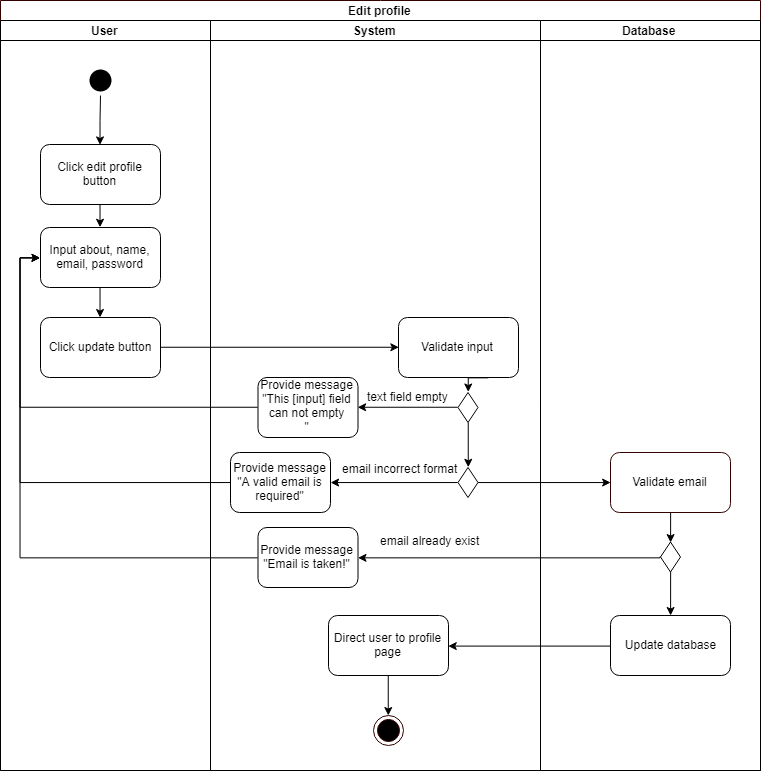
Activity diagram 4: AC04



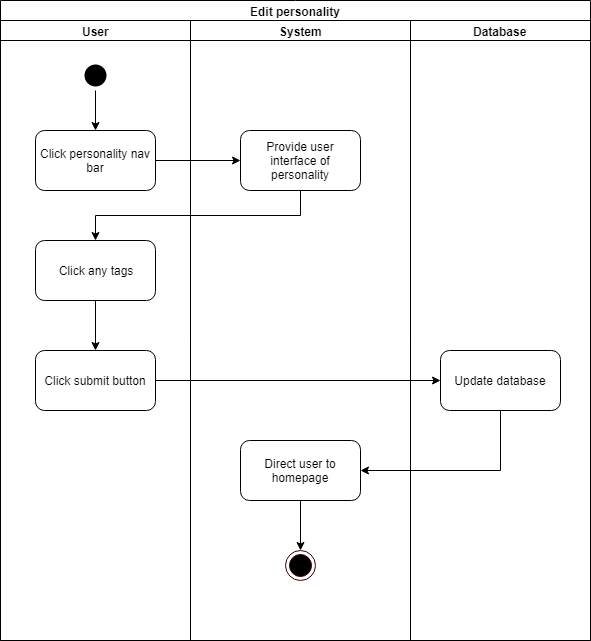
Activity diagram 5: AC05



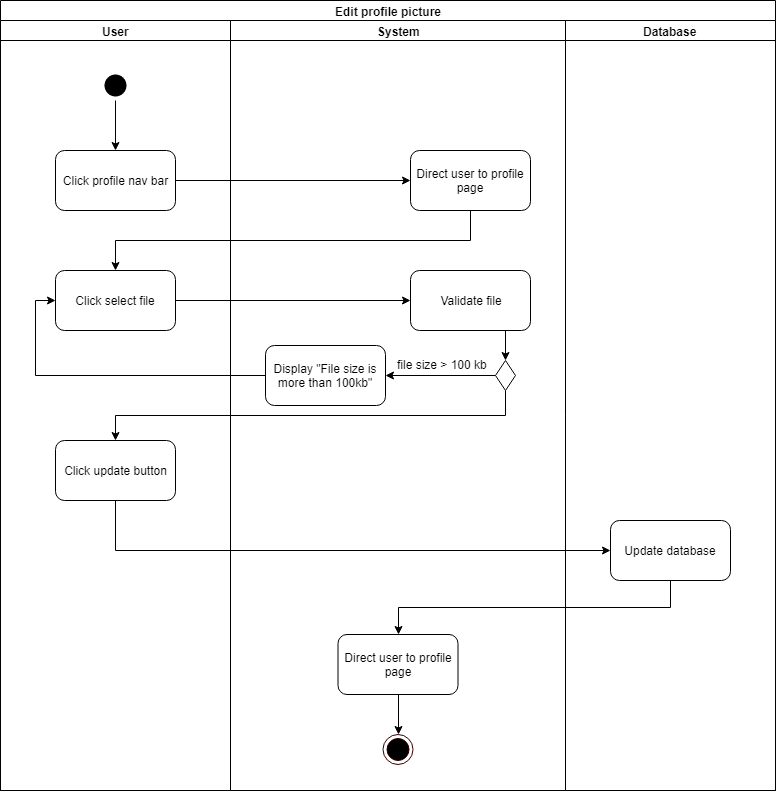
Activity diagram 6: AC06



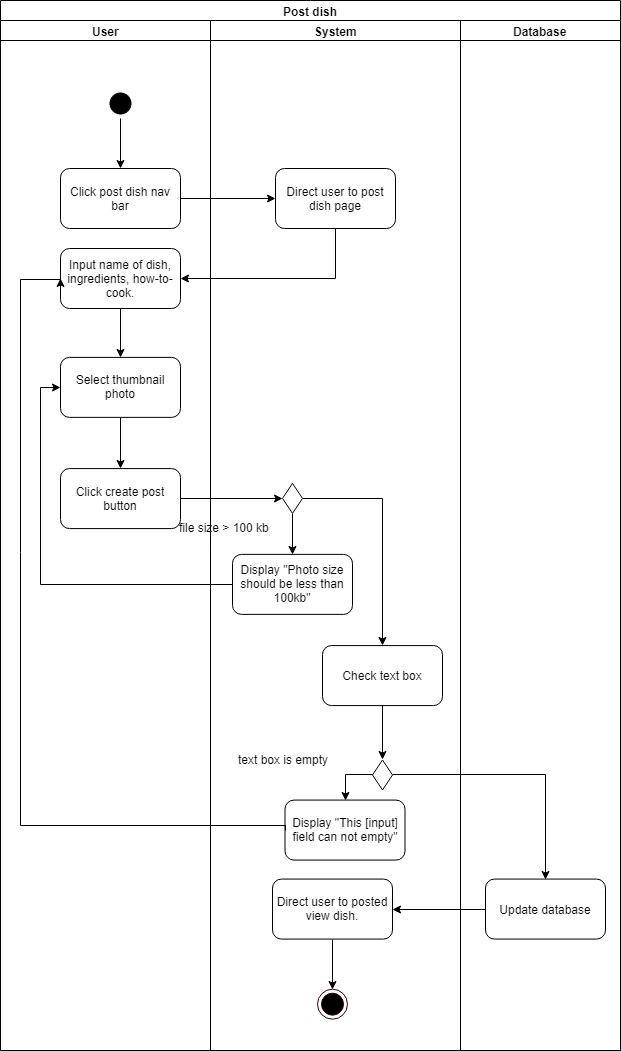
Activity diagram 7: AC07



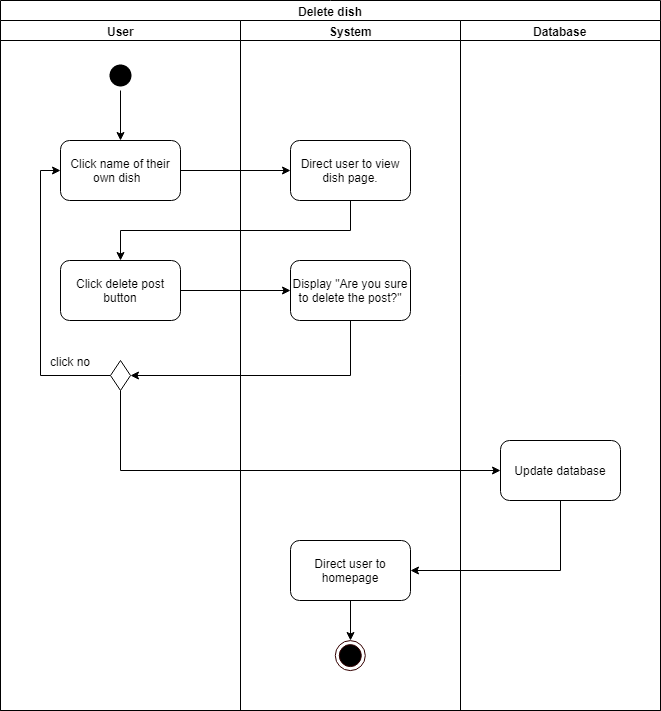
Activity diagram 8: AC08



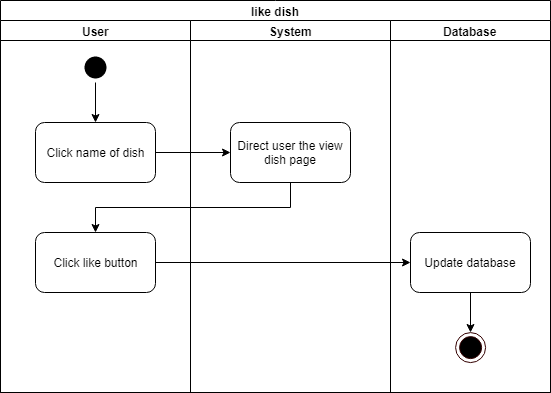
Activity diagram 9: AC09



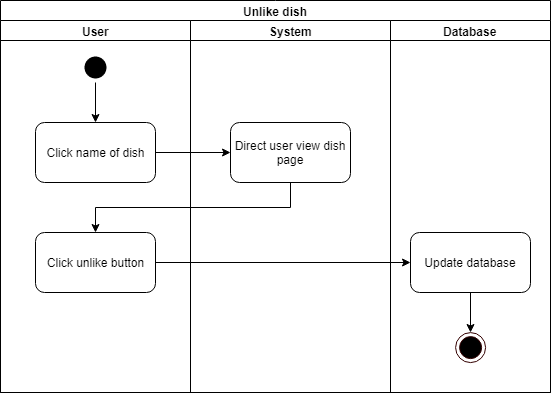
Activity diagram 10: AC10



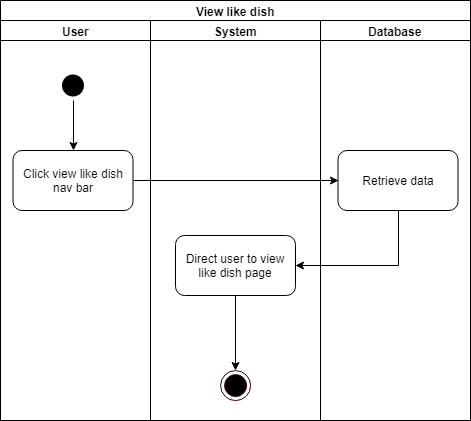
Activity diagram 11: AC11



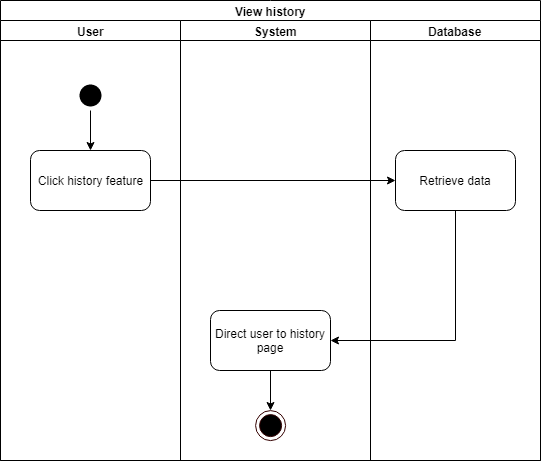
Activity diagram 12: AC12



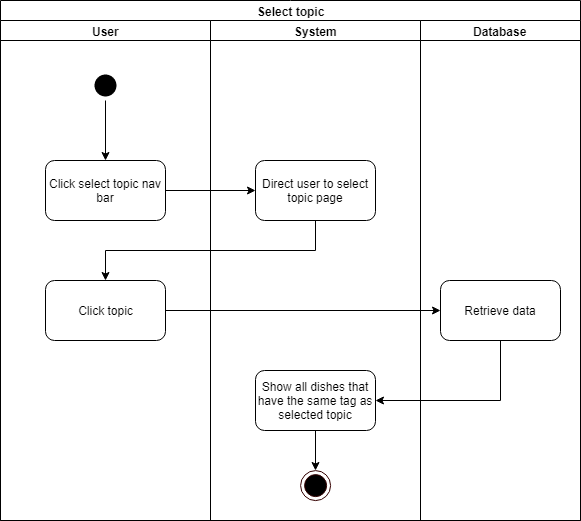
Activity diagram 13: AC13



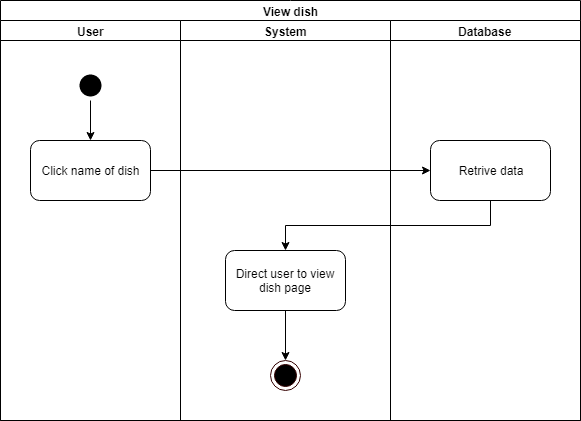
Activity diagram 14: AC14



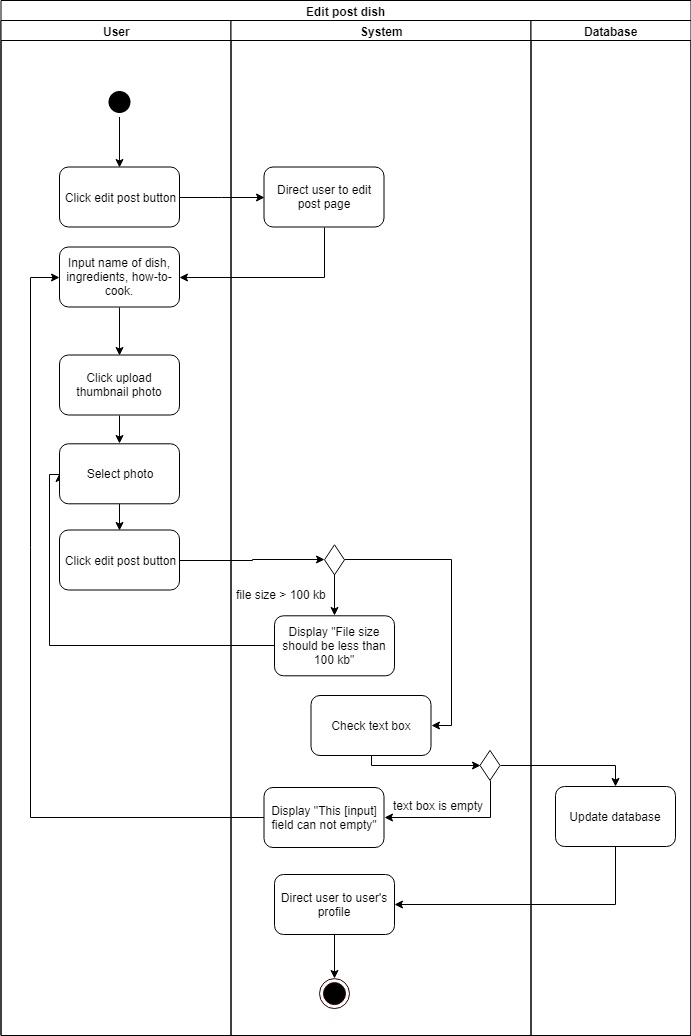
Activity diagram 15: AC15



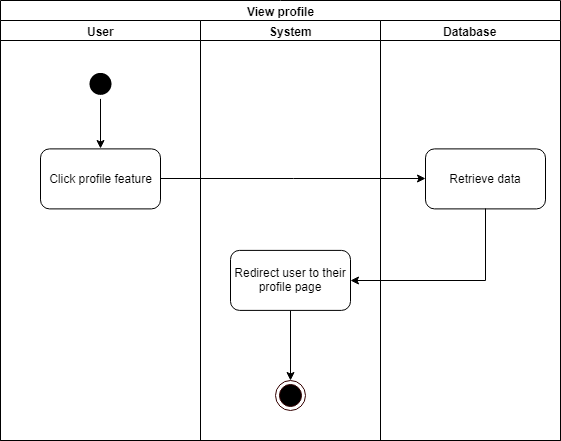
Activity diagram 16: AC16



Activity diagram 17: AC17



Activity diagram 18: AC18



**Spotifood**

**Software design document**

**By**

**Mr. Kittinut Saengsri 592115003**

**Mr. Sirisak Wongwal 592115031**

**Department of Software Engineering**

**College of Arts, Media, and Technology**

**Chiang Mai University**

**Project Advisor**

**Dr. Passakorn Phannachitt**

Document History

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Document name | Version | Status | Date | Viewable | Editable | Responsible |
| Spotifood\_SoftwareDesign\_v0.1 | - Add Chapter one  - Add chapter two  - Add chapter three  - Add chapter four  - Add chapter five | Draft | 07/10/2019 | SW, KS, PP | SW, KS | SW, KS |
| Spotifood\_SoftwareDesign\_v1.0 | - Add Chapter one  - Add chapter two  - Add chapter three  - Add chapter four  - Add chapter five | Draft | 07/29/2019 | SW, KS, PP | SW, KS | SW, KS |

\*SW = Sirisak Wongwal

\*KS = Kittinut Saengsri

\*PP = Dr.Passakorn Phannachitta

# Chapter 1 | Introduction

## 1.1 Identification

This Project Management Plan is the document for planning, scheduling activities and evaluating overall of the project so that it could complete successfully despite all the risks. It provides detailed scheduled, assigned task and identified risks of the project.

## 1.2 Project Overview

The Spotifood application will guide people find their new dish and help people who are interested in cooking and people who have cooking skill by suggesting the dishes from a word that people search. The AI system will suggest a dish from the personality of people. For example, The people regularly search steamed fish with herbs dish. The AI system will analyze the ingredient tag of dish that is related to the healthy tag. Then, the AI will suggest the people in the type of healthy dishes.

The suggest dish of AI will help those people found new dishes and challenging in cooking every day that makes people not disconnected and bored to cooking. Moreover, we will be a middleman that brings trader of ingredient to people who can’t find the ingredients in any food. The Spotifood help people to buy an ingredient. And the Spotifood will suggest a dish that is similar to personality or suggests from the behavior of people.

### 1.2.1 Purpose

The main purpose of developing the Spotifood is to help the people who are interested in cooking and who already have skills not bored with cooking or disconnected to prepare the food.

### 1.2.2 Scope

There are six major parts in Spotifood.

Feature #1: Authentication

Feature #2: Content CRUD

Feature #3: User management

Feature #4: Similar dishes identification

Feature #5: Personalization

Feature #6: ADS ingredient

## 1.3 Work Product to be developed

### 1.3.1 Deliverable

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Deliverable/Release | Media | Copies | Date |
| 1 | Project proposal  - Spotifood\_proposal3 | Hard copy | 3 | 19/06/2019 |
| 2 | Progress report I  - Project management plan version 1.0  - Software requirement specification version 1.0  - Software design document version 1.0  - Test plan version 1.0  - Traceability record version 1.0  - Software version 0.1 | Hard copy | 3 | 27/07/2019 |

## 1.4 Acronyms and Definitions

### 1.4.1 Acronyms

UC Use case

URS User Requirement Specification

SRS Software Requirement Specification

CRUD Create-Read-Update-Delete

AI Artificial Intelligence

ADS Advertising

ID Identification Data

# CHAPTER 2 | SOFTWARE ARCHITECTURE

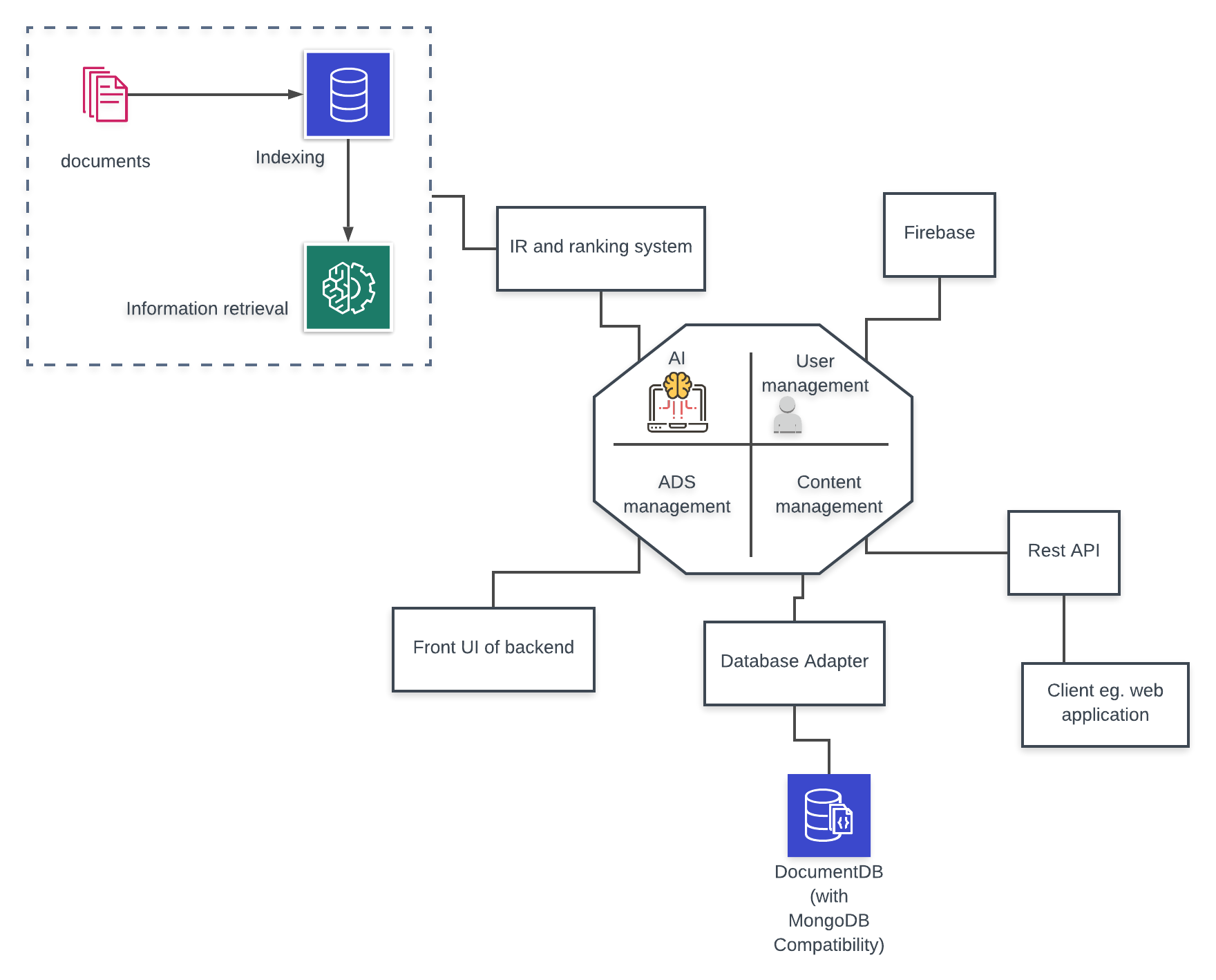


Figure 01: Spotifood System Architecture Diagram

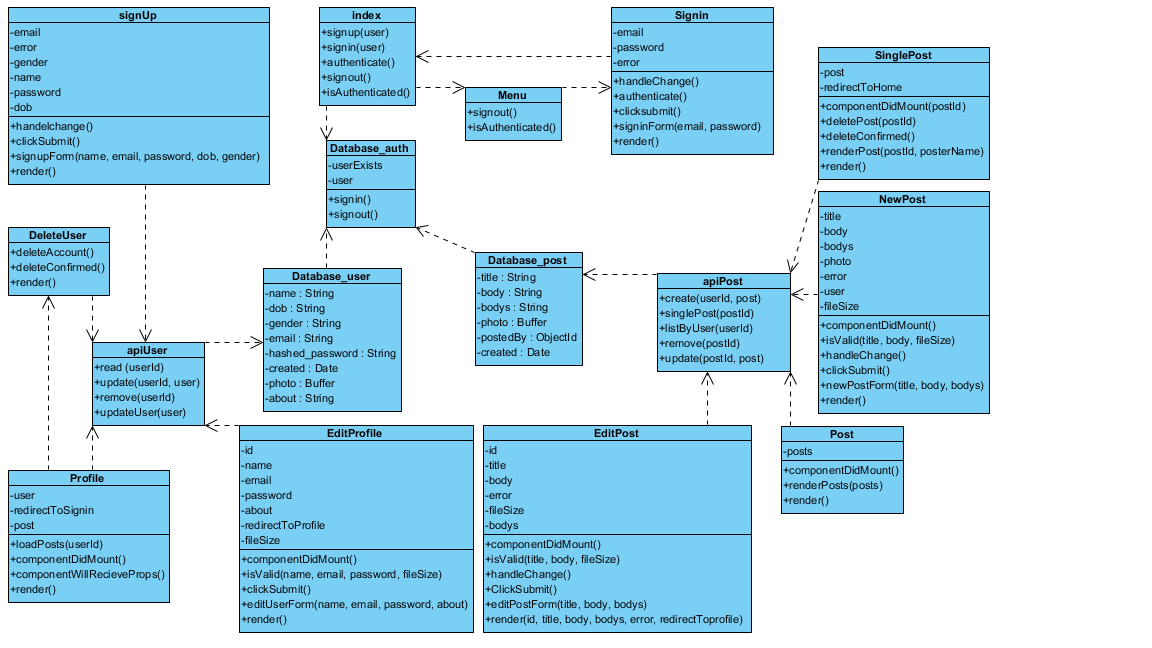
There are four subsystem of system architecture

1. AI (Artificial intelligence)
2. User management
3. ADS management (Advertisement management)
4. Content management

Firstly content management , The user interact with the system through web application developed using React, Node.js, And backend service of spotifood system using MongoDB. Secondly user management, Is the system backend for user authentication is handled with Firebase services because We use to collect only username and password. Thirdly ADS management , Is the font UI for admin to interact with the system through web application developed using React, Node.js. Finally AI or artificial intelligence , This part is the search for items that are stored in the form that is most relevant to the person from their search using IR ( information retrieval) to incidence matrix and inverted index

# CHAPTER 3 | DETAIL DESIGN

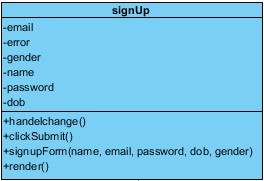
## 3.1 CLASS DIAGRAM



## 3.2 CLASS DIAGRAM DESCRIPTION

### 3.2.1 Spotifood Application Class Description

**CD-01 : signUp**



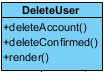
**Attributes**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | email | Email of the user. | String |
| 2 | error | Contain message of each error parameter. | String |
| 3 | gender | Gender of the user. | String |
| 4 | name | Name of the user. | String |
| 5 | password | Password of the user. | String |
| 6 | dob | Date of birth of the user. | String |

**Method**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | handleChange | Set state of error message and set event from user name value. | function(event) |
| 2 | clickSubmit | Send all parameter from signupForm to the database. | function(event) |
| 3 | signupForm | Display signup form to the user for handling input. | Return name , email , password , dob , gender |
| 4 | render | Alert error and success message to the user. | Return error message and success message |

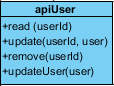
**CD-02 : DeleteUser**



**Method:**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | deleteAccount | Delete account the user. | function(event) |
| 2 | deleteConfirmed | Alert confirm box to the user before delete. | function(event) |
| 3 | render | Display the delete profile button. | Return Delete profile button |

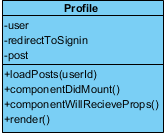
**CD-03 : apiUser**



**Method**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | read | Get the user json file from database by user id. | Param: userId  Return: - |
| 2 | update | Put or update the user json file to the database by user id. | Param: userId  Return: - |
| 3 | remove | Delete the user json file from the database by user id. | Param: userId  Return: - |
| 4 | updateUser | Put or update the user json file to the database and show old information by user id. |  |

**CD-04 : Profile**



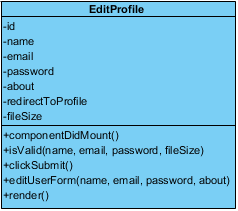
**Attributes**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | user | Data of the user from database. | String |
| 2 | redirectToSignin | Use to redirect to sign-in page. | boolean |
| 3 | post | Data of the post from user. | Array |

**Method**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | loadPosts | Display post from user. | Param: userId  Return: posts.data |
| 2 | componentDidMount | Receive post of the user from user id. | function(event) |
| 3 | componentWillReceiveProps | Receive all post of the user from user id. | function(event) |
| 4 | render | Display information of user and allow user to click create post button and edit profile button. | Return: user , posts. |

**CD-05 : EditProfile**



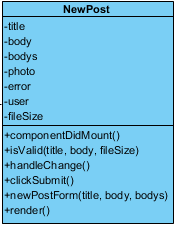
**Attributes**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | id | Unique identifier of the stadium. | String |
| 2 | name | Name of the user. | String |
| 3 | password | Password of the user | String |
| 4 | about | About user can handling in about field. | String |
| 5 | redirectToProfile | Redirect to profile page. | boolean |
| 6 | fileSize | File size of photo of the user. | int |

**Method**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | isValid | Verify all format of parameters and send the error message to render method. | Param: name, email , password, fileSize  Return: - |
| 2 | componentDidMount | Receive post of the user from user id. | function(event) |
| 3 | clickSubmit | Put the information of user update to the database. | function(event) |
| 4 | editUserForm | Display information of user and allow user to handling edit information. | Param: name, email,  Password, fileSize  Return: - |
| 5 | render | Display information of user and allow user to create post band edit profile. | Return: name, email , password , about |

**CD-06 : NewPost**



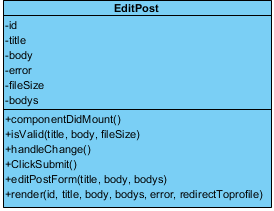
**Attributes**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | photo | Unique identifier of the stadium. | String |
| 2 | title | Name of the dish. | String |
| 3 | body | Ingredient of the dish. | String |
| 4 | bodys | Step of cook of the dish. | String |
| 5 | error | Contain message of each error parameter. | String |
| 6 | fileSize | File size of thumbnail photo of the post. | int |
| 7 | user | Information of user in database | Array |

**Method**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | isValid | Verify all format of parameters and send the error message to render method. | Param: title, body, fileSize  Return: - |
| 2 | handleChange | Receive post of the user from user id. | function(event) |
| 3 | clickSubmit | Create the information of post to the database. | function(event) |
| 4 | newPostForm | Display information of post and allow user to handling create information. | Param: title, body, bodys  Return: - |
| 5 | render | Display information of create a new post. | Return: title, body |

**CD-07 : EditPost**



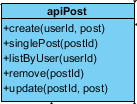
**Attributes**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | id | Unique identifier of the stadium. | String |
| 2 | title | Name of the dish. | String |
| 3 | body | Ingredient of the dish. | String |
| 4 | bodys | Step of cook of the dish. | String |
| 5 | error | Contain message of each error parameter. | String |
| 6 | fileSize | File size of thumbnail photo of the post. | int |

**Method**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | isValid | Verify all format of parameters and send the error message to render method. | Param: title, body, fileSize  Return: - |
| 2 | handleChange | Receive post of the user from post id. | function(event) |
| 3 | clickSubmit | Put the information of post update to the database. | function(event) |
| 4 | editPostForm | Display information of post and allow user to handling edit information. | Param: title, body, bodys  Return: - |
| 5 | render | Display information of post. | Return: title, body, bodys |

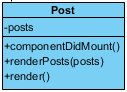
**CD-08 : apiPost**



**Method**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | create | Post or create the post of user to the database by userId. | Param: userId, post  Return: - |
| 2 | singlePost | Get or receive post information from the database by postId. | Param: postId  Return: - |
| 3 | listByUser | Get or receive all post information of post update to the database by userId. | Param: userId  Return: - |
| 4 | remove | Delete a post from the database by postId. | Param: postId  Return: - |
| 5 | update | Put or update the post of user to the database by postId. | Param: postId , post  Return: - |

**CD-09 : apiPost**



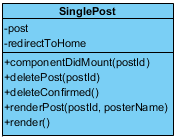
**Attributes**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | posts | All of the posts in the database | Array |

**Method**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | componentDidMount | Receive information of all posts from the database. | function(event) |
| 2 | renderPosts | Receive all of the posts to display and send to render function. | Param: posts  Return: - |
| 3 | render | Display all of the posts. | Return: posts |

**CD-10 : SinglePost**



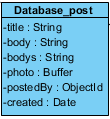
**Attributes**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | posts | All of the posts in the database | Array |
| 2 | redirectToHome | Redirect to home page. | boolean |

**Method**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | componentDidMount | Receive information of post from the database by postId. | Param: postId  Return: - |
| 2 | deletePost | Remove the post from the database by postId and send to deleteConfirmed | Param: postId  Return: - |
| 3 | deleteConfirmed | Alert delete confirm box to the user. | function(event) |
| 4 | renderPost | Receive all of the posts to display , provide Edit and Delete post button send to render function. | Param: postId, postName  Return: - |
| 5 | render | Display single page of the post. | Return: post |

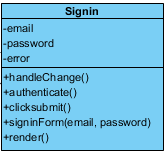
**CD-11 : Database\_post**



**Attributes**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | photo | Thumbnail photo of the post. | Buffer |
| 2 | title | Name of the dish. | String |
| 3 | body | Ingredient of the dish. | String |
| 4 | bodys | Step of cook of the dish. | String |
| 5 | postedBy | Name of user from object id by user. | ObjectId |
| 6 | created | Date of user created the post. | Date |

**CD-12 : Signin**



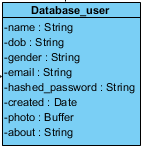
**Attributes**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | email | Email of the user | String |
| 2 | password | Password of the user | String |
| 3 | error | Contain message of each error parameter. | String |

**Method**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | handleChange | Set state of error message and set event from user name value. | function(event) |
| 2 | authenticate | Authenticate the user to sign in to the web application. | function(event) |
| 3 | clicksubmit | Send email and password to the database. | function(event) |
| 4 | signinForm | Display the sign in form to the user for handling sign in. | Param: email, password  Return: - |
| 5 | render | Display error message and redirect to home page. | Return: email, password |

**CD-13 : Database\_user**



**Attributes**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | name | Thumbnail photo of the post. | Buffer |
| 2 | dob | Name of the dish. | String |
| 3 | gender | Ingredient of the dish. | String |
| 4 | email | Step of cook of the dish. | String |
| 5 | hashed\_password | Name of user from object id by user. | ObjectId |
| 6 | created | Date of user created the post. | Date |
| 7 | photo | Photo profile of the user. | Buffer |
| 8 | about | Bio about the user. | String |

**CD-14 : Database\_auth**



**Attributes**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | userExists | The user that has in the database. | String |
| 2 | user | The new user in the database. | String |

**Method**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | signin | Sign in to the spotifood application and check email , password in the database then access to the spotifood application. | function(event) |
| 2 | signout | Sign out the user to the spotifood application. | function(event) |

**CD-15 : Database\_Menu**



**Method**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | isAuthenticated | Authenticated the user who can access the function in spotifood application. | function(event) |
| 2 | signout | Sign out the user to the spotifood application. | function(event) |

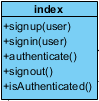
**CD-16 : Database\_Menu**



**Method**:

|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | isAuthenticated | Authenticated the user who can access the function in spotifood application. | function(event) |
| 2 | signout | Sign out the user to the spotifood application. | function(event) |

**CD-17 : Database\_Menu**

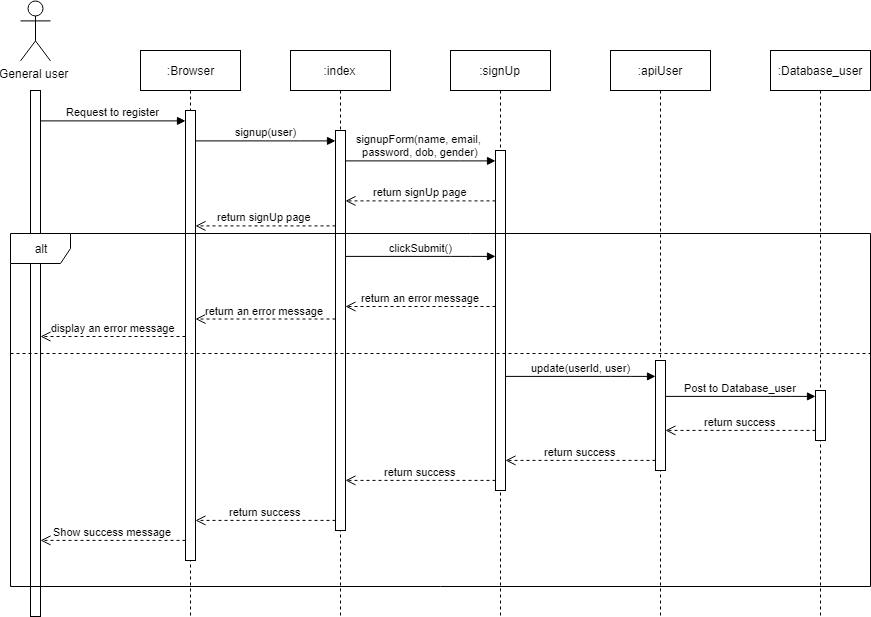


**Method**:

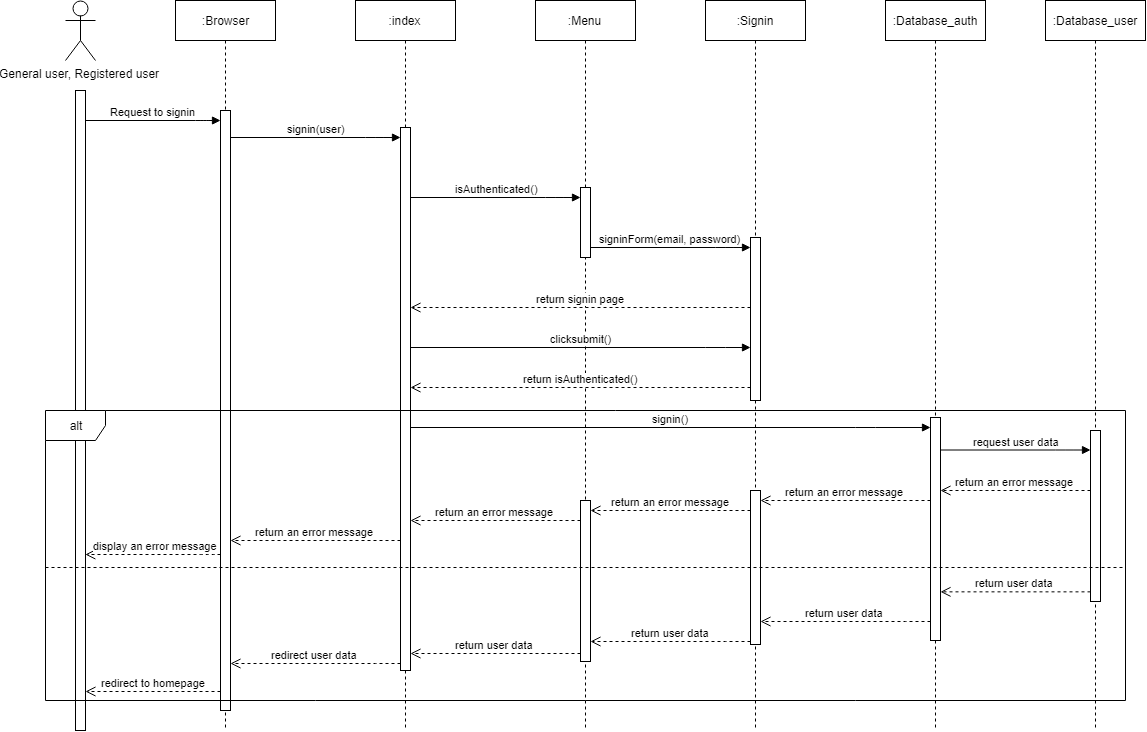
|  |  |  |  |
| --- | --- | --- | --- |
| ID | NAME | Description | Type |
| 1 | signup | Authenticated the user who can access the function in spotifood application and send to authenticate function by post jwt token to json file in database. | Param: user  Return: - |
| 2 | signout | Sign out the user to the spotifood application. | function(event) |
| 3 | authenticate | Set jwt token to json file in database. | function(event) |
| 4 | signin | Post the user into the database of spotifood application by jwt token to json file in database. | Param: user  Return: - |
| 5 | isAuthenticated | Authenticated the user who can access the function in spotifood application by get jwt token to json file. | function(event) |

# Chapter 4 | Sequence Diagram

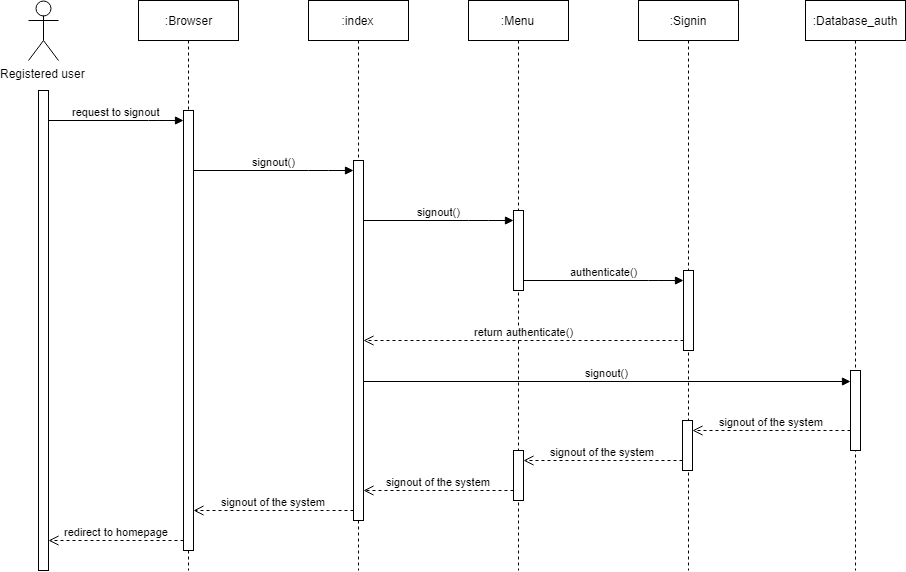
SD-01 Register



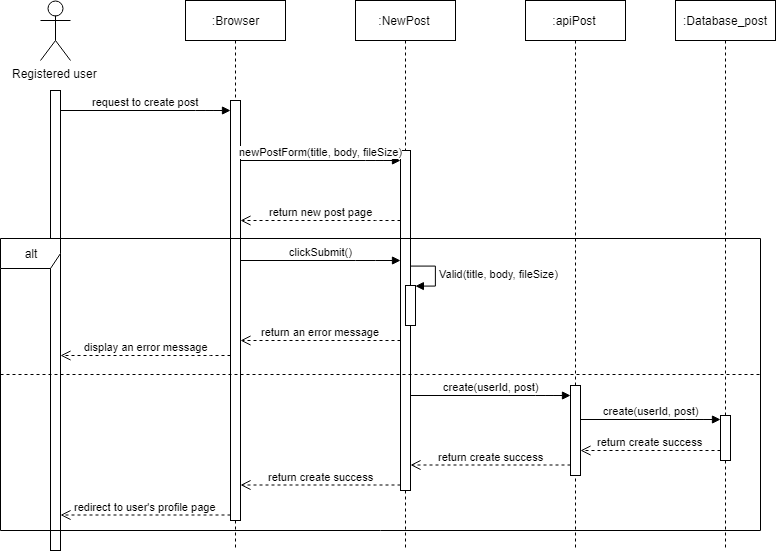
SD-02 Sign in



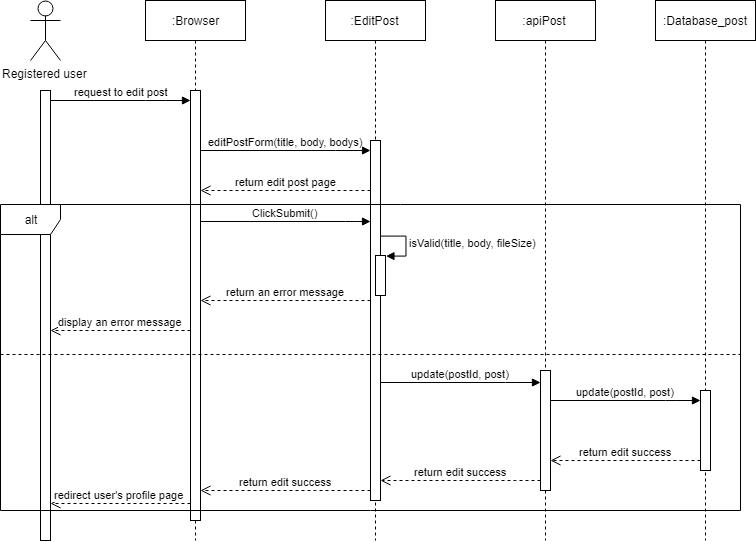
SD-03 Sign out



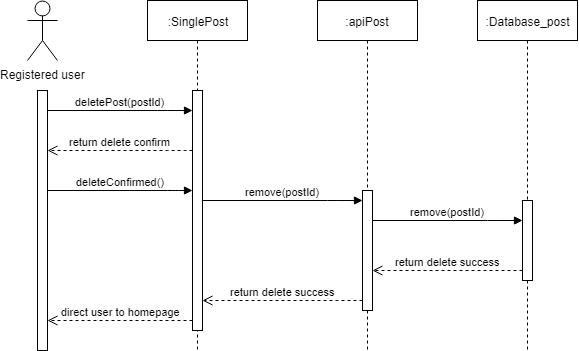
SD-04 Create post



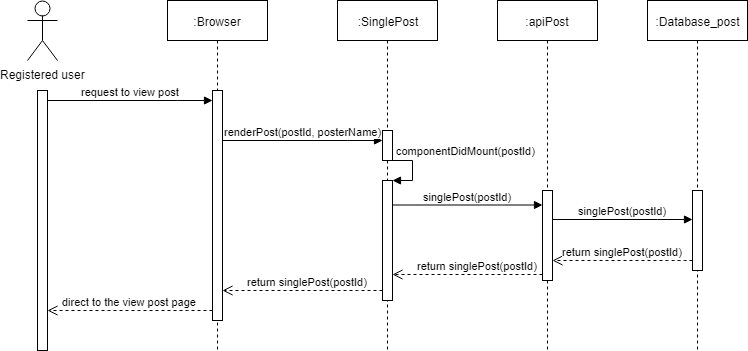
SD-05 Edit post



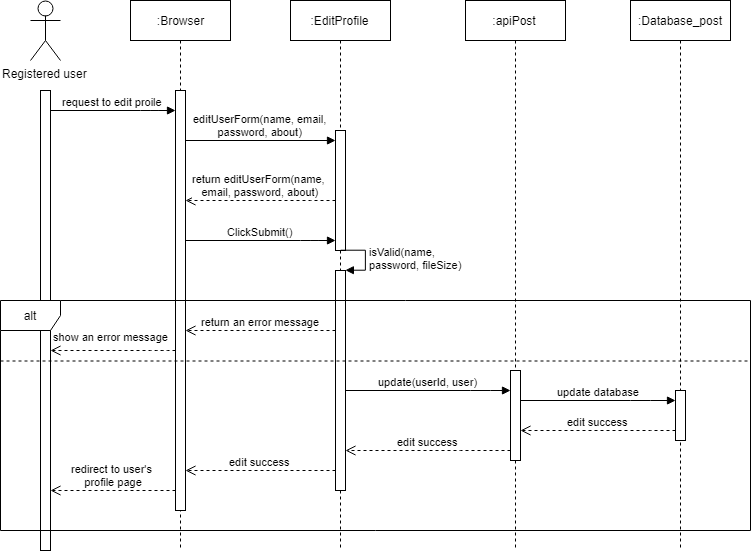
SD-06 Delete post



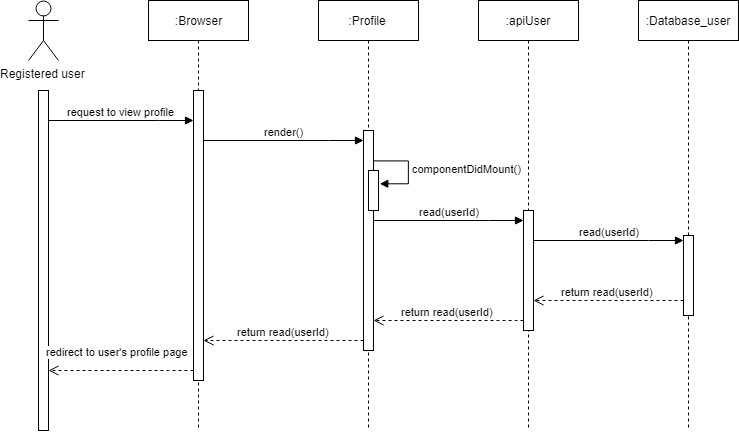
SD-07 View post



SD-08 Edit profile

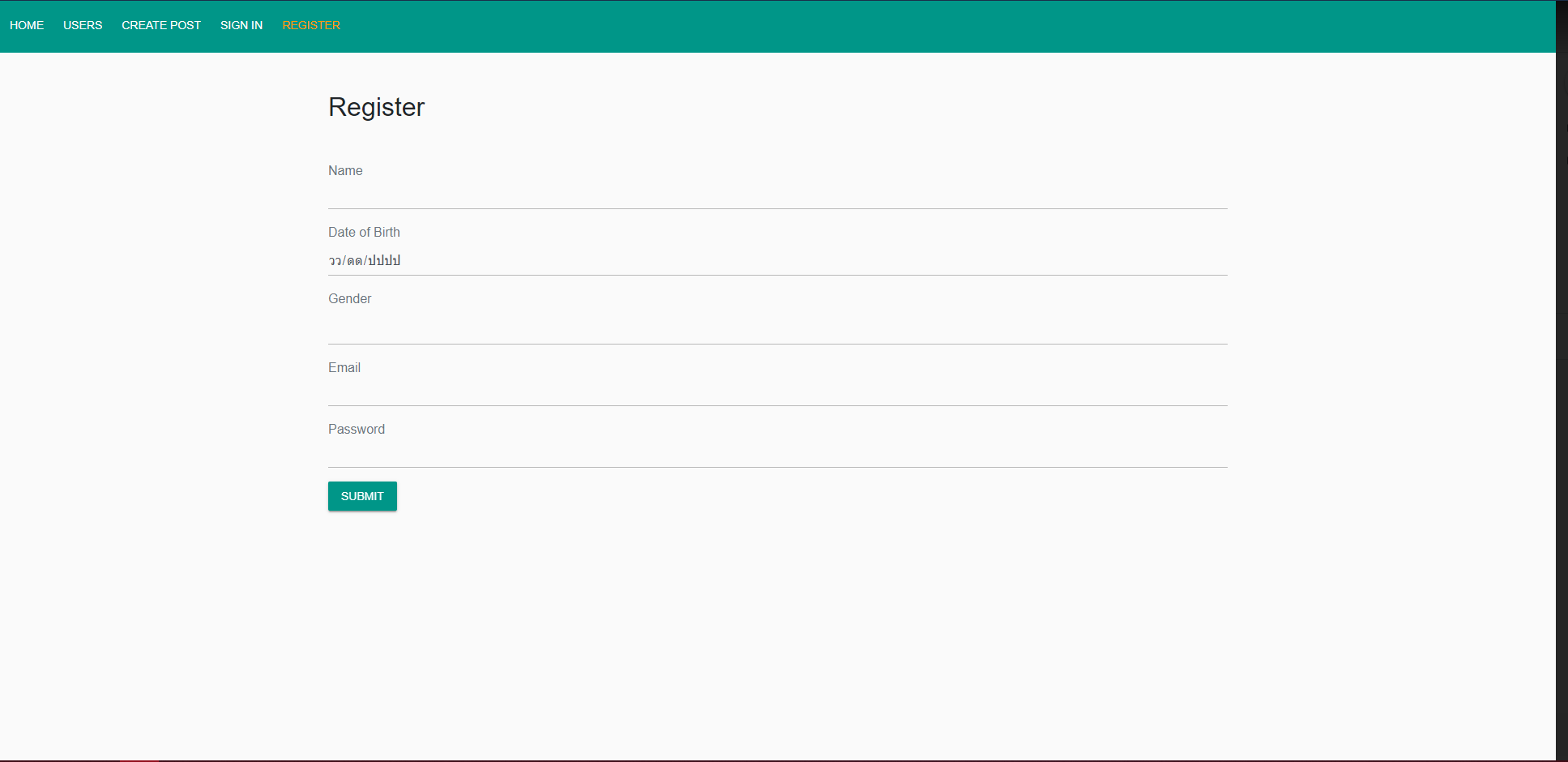


SD-09 View profile

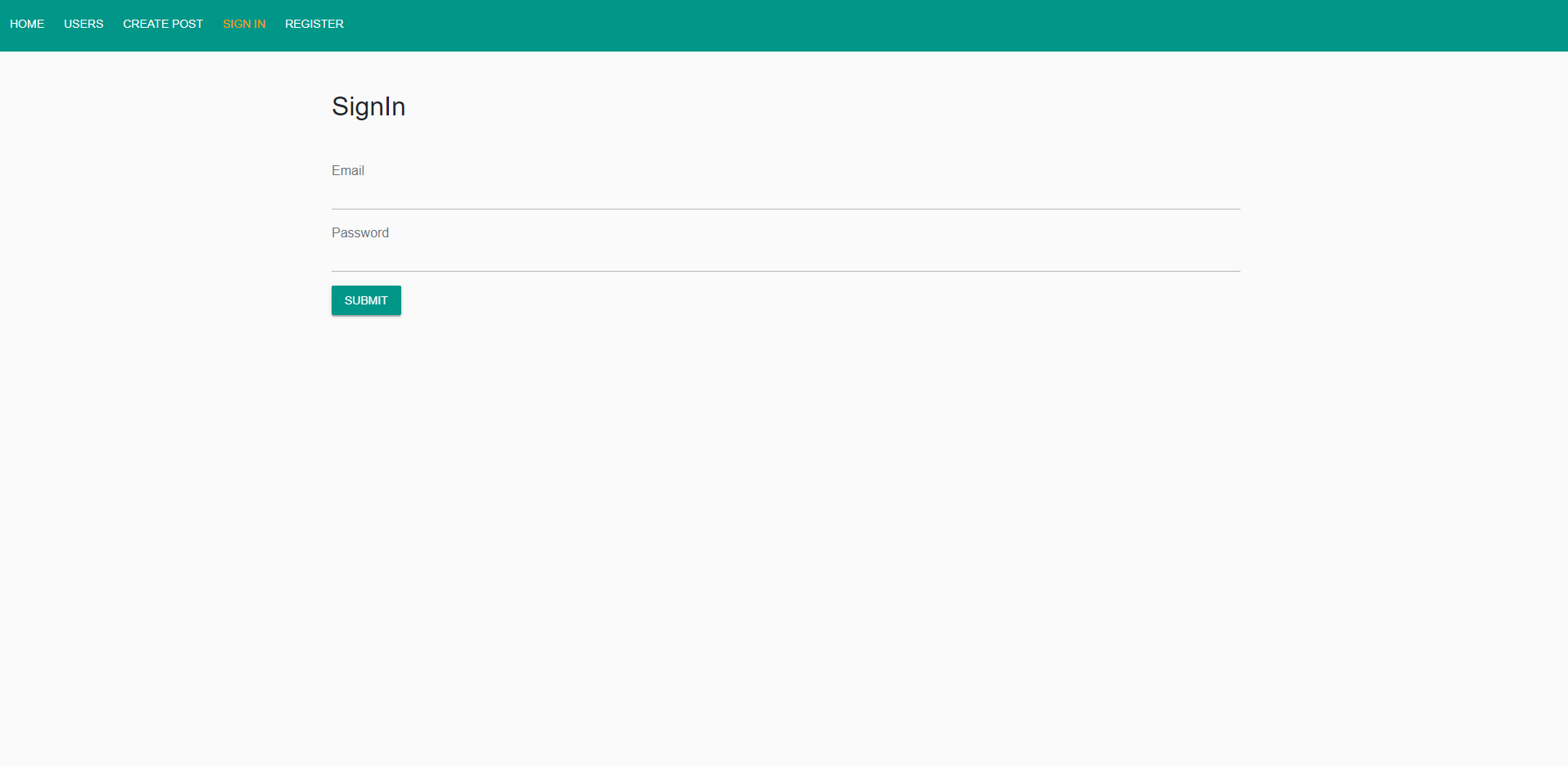


# Chapter 5 | User Interface

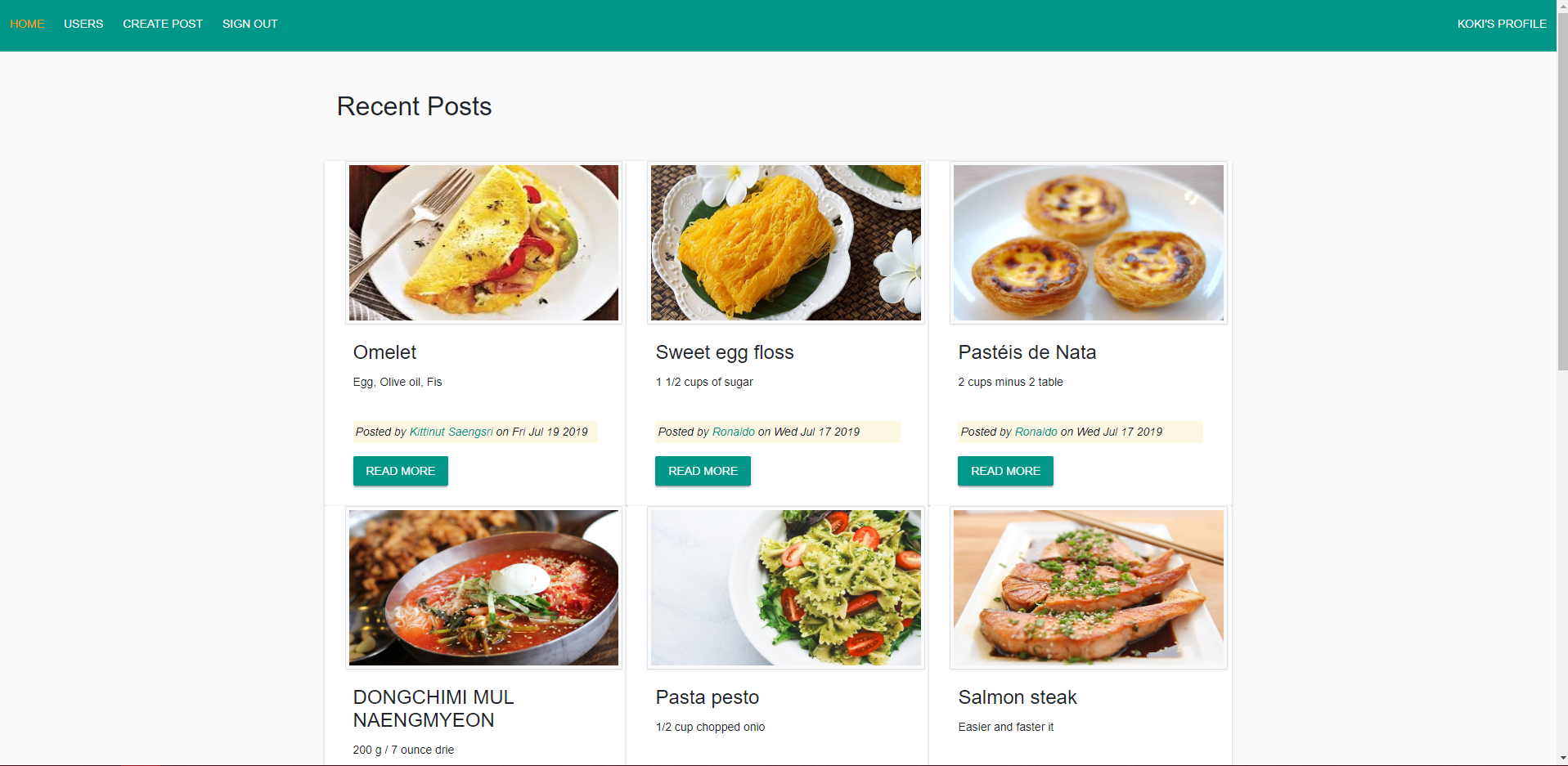
UI-01: Register page



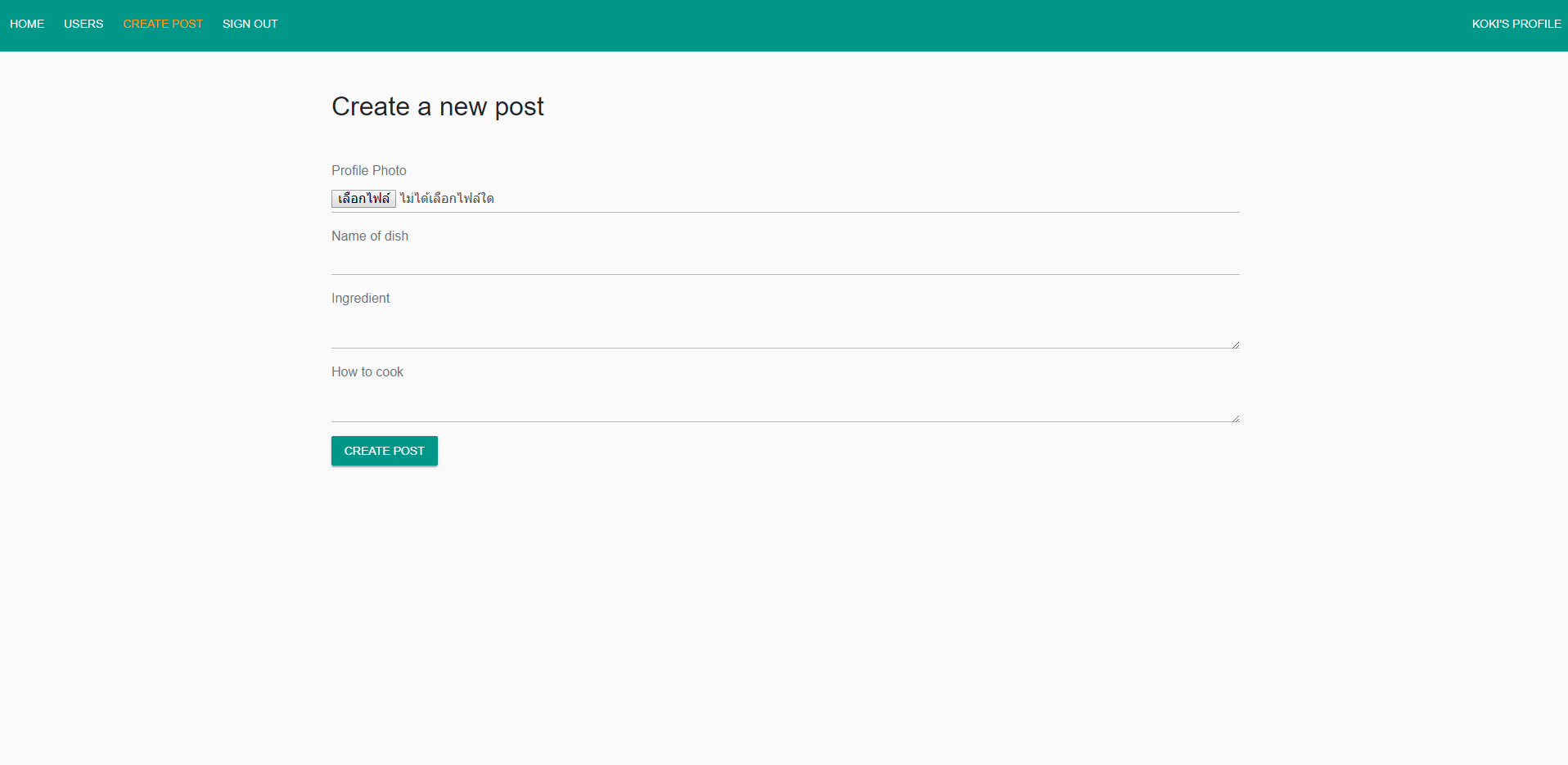
UI-02: Sign in page



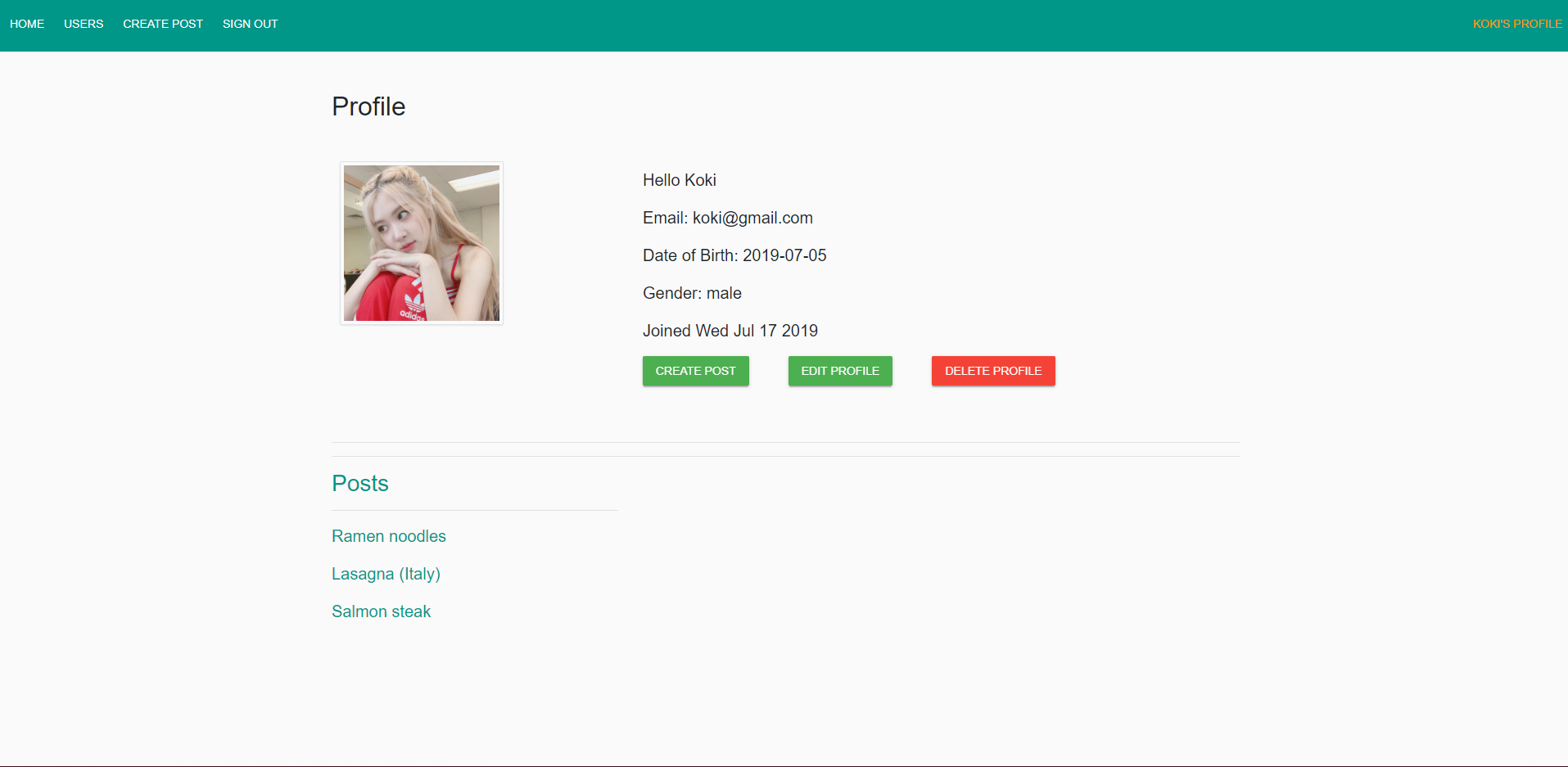
UI-03: Home page



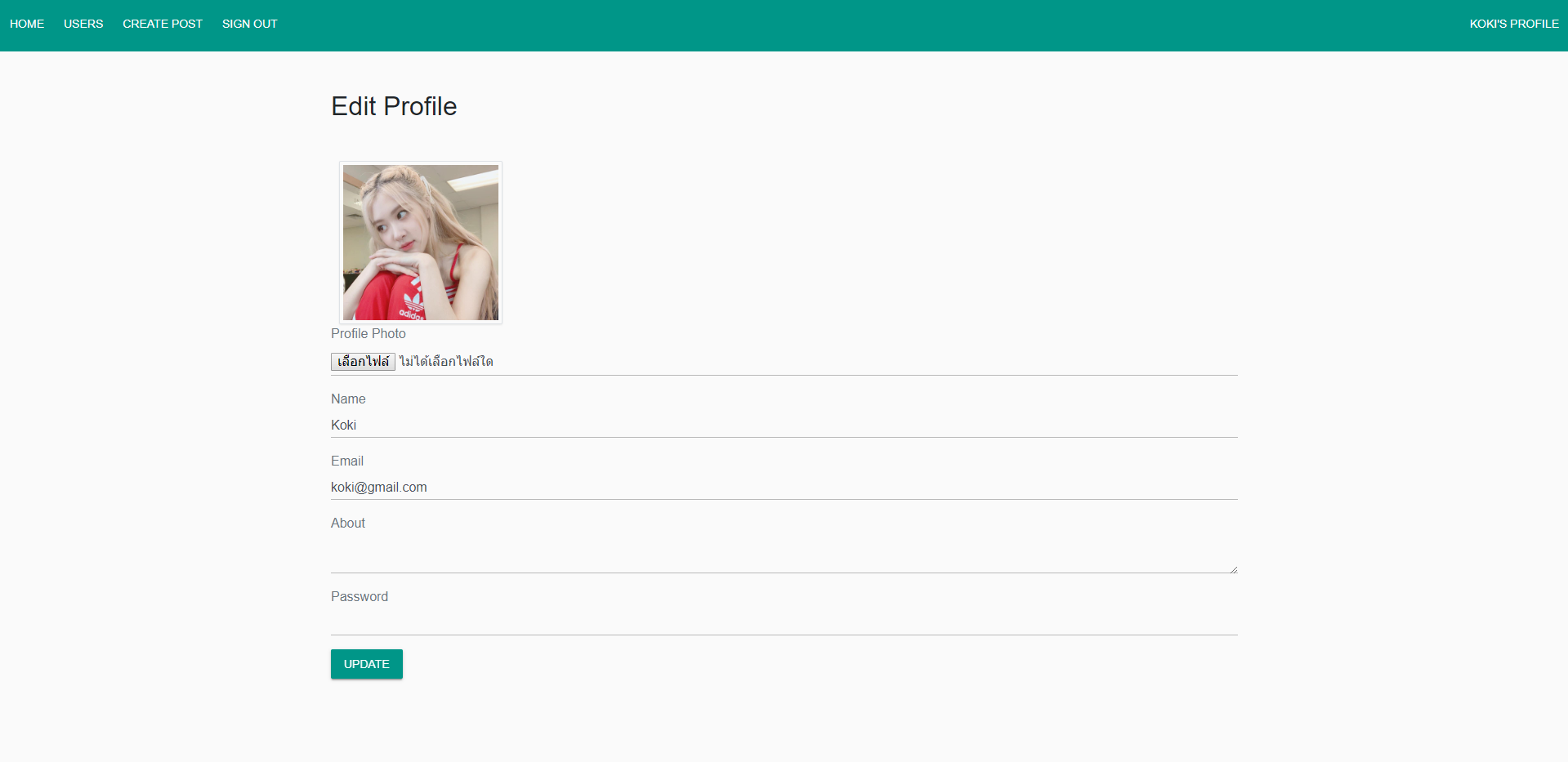
UI-04: Create new post page



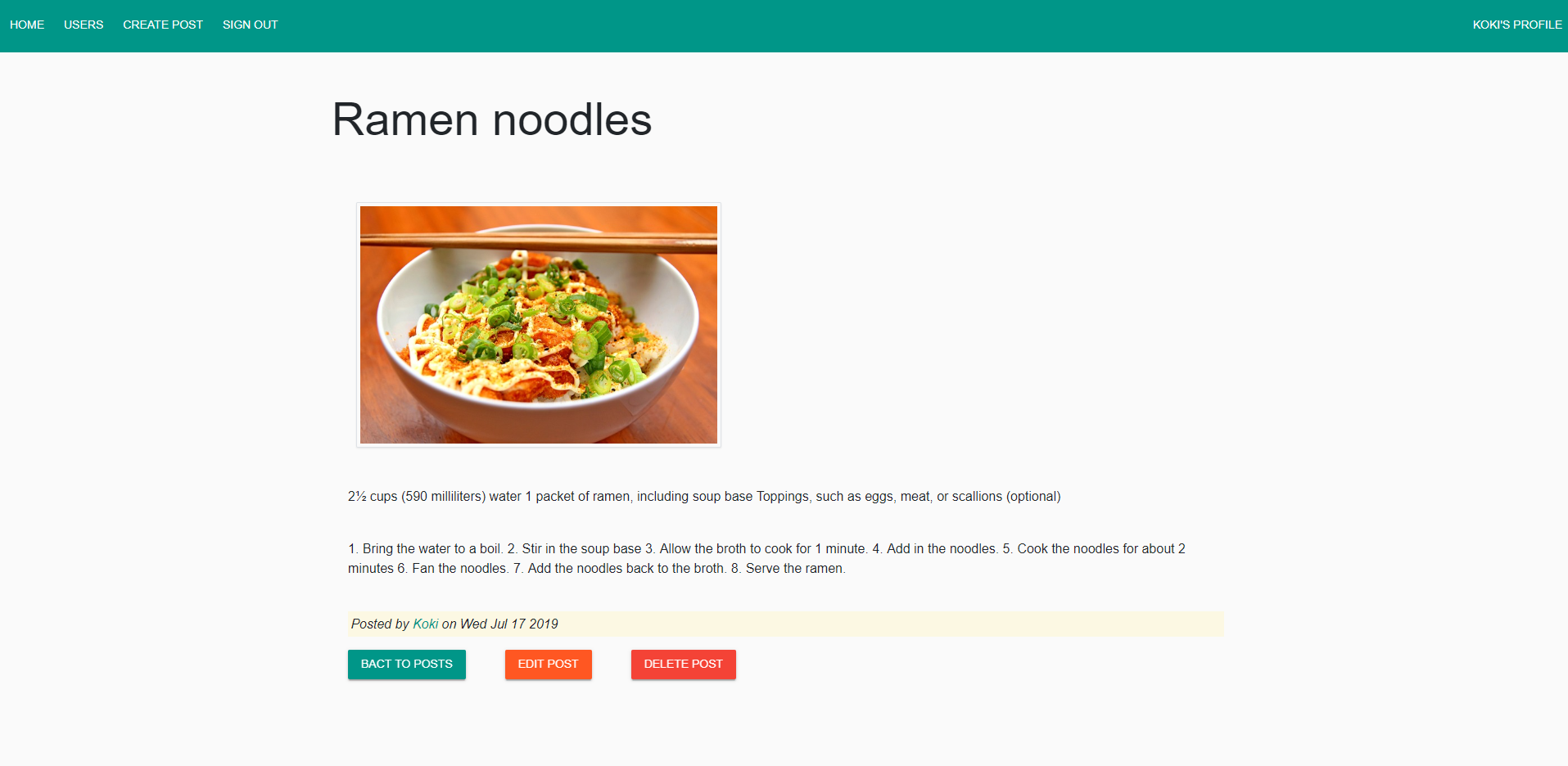
UI-05: Profile page



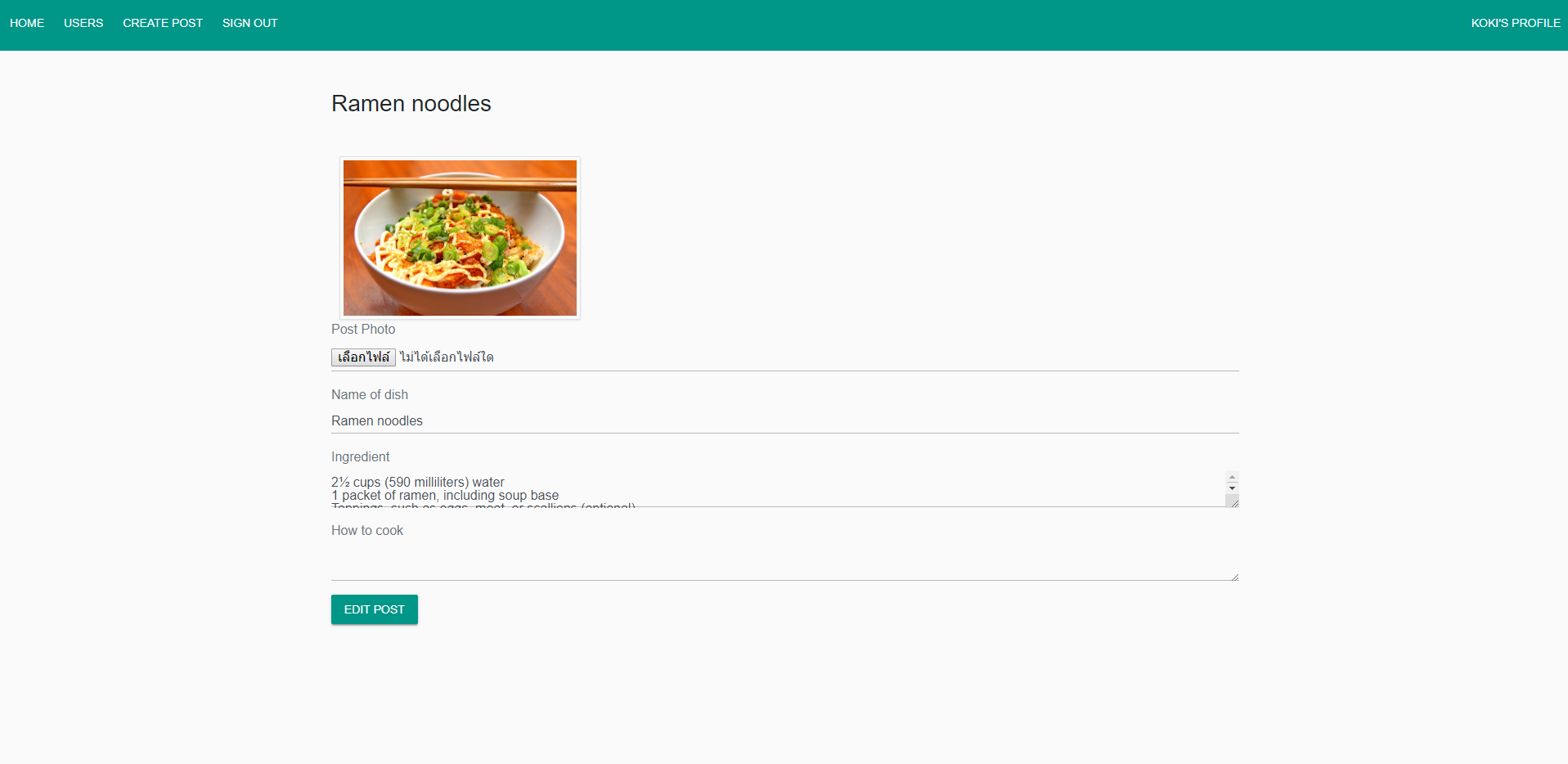
UI-06: Edit profile page



UI-07: The single dish post page



UI-08: Edit dish page



**Spotifood**

**Test record**

**By**

**Mr. Kittinut Saengsri 592115003**

**Mr. Sirisak Wongwal 592115031**

**Department of Software Engineering**

**College of Arts, Media, and Technology**

**Chiang Mai University**

**Project Advisor**

**Dr. Passakorn Phannachitta**

Document History

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Document name | Version | Status | Date | Viewable | Editable | Responsible |
| Spotifood\_Testrecord\_v0.1 | - Add chapter one  - Add chapter two  - Add chapter three  - Add chapter four | Draft | 07/10/2019 | SW, KS, PP | SW, KS | SW, KS |
| Spotifood\_Testrecord\_v1.0 | - Update chapter one  - Update chapter two  - Update chapter three  - Update chapter four | Draft | 07/29/2019 | SW, KS, PP | SW, KS | SW, KS |

\*SW = Sirisak Wongwal

\*KS = Kittinut Saengsri

\*PP = Dr.Passakorn Phannachitta

# Chapter 1 | Introduction

## 1.1 Identification

This Project Management Plan is the document for planning, scheduling activities and evaluating overall of the project so that it could complete successfully despite all the risks. It provides detailed scheduled, assigned task and identified risks of the project.

## 1.2 Project Overview

The Spotifood application will guide people find their new dish and help people who are interested in cooking and people who have cooking skill by suggesting the dishes from a word that people search. The AI system will suggest a dish from the personality of people. For example, The people regularly search steamed fish with herbs dish. The AI system will analyze the ingredient tag of dish that is related to the healthy tag. Then, the AI will suggest the people in the type of healthy dishes.

The suggest dish of AI will help those people found new dishes and challenging in cooking every day that makes people not disconnected and bored to cooking. Moreover, we will be a middleman that brings trader of ingredient to people who can’t find the ingredients in any food. The Spotifood help people to buy an ingredient. And the Spotifood will suggest a dish that is similar to personality or suggests from the behavior of people.

### 1.2.1 Purpose

The main purpose of developing the Spotifood is to help the people who are interested in cooking and who already have skills not bored with cooking or disconnected to prepare the food.

### 1.2.2 Scope

There are six major parts in Spotifood.

Feature #1: Authentication

Feature #2: Content CRUD

Feature #3: User management

Feature #4: Similar dishes identification

Feature #5: Personalization

Feature #6: ADS ingrediente

## 1.3 Acronyms and Definitions

### 1.3.1 Acronyms

UC Use case

URS User Requirement Specification

SRS Software Requirement Specification

SDD Software Design Document

OS Operating System

VSE Very Small Entity

PM Project Management

SI Software Implementation

IID Iterative and Incremental Development

SCI Software Configuration Item

CRUD Create-Read-Update-Delete

# Chapter 2 | Test Procedure

## 2.1 Test Objectives

The objectives of Spotifood System are:

1. Identify the faults – Finding the fault which caused the failure.

2. Correct and remove the faults – Fixing errors and remove faults.

3. Ensure that all functionalities are implemented correctly.

## 2.2 Test Duration Progress Date and Duration

Progress I Perform date: 07/10/2019 Duration days: 07/29/2019

## 2.3 Test Responsibility Item Responsibility

Unit test SW,KS

Record unit test SW,KS

System test SW,KS

Record of system test SW,KS

## 2.4 Test Strategy Spotifood performed testing strategy by the following steps:

1. Design test cases

2. Prepare test data

3. Determine the expected result

4. Perform testing

5. Record all results

## 2.5 Result of Testing The results are separated into two parts:

1. Actual result: The actual results after testing

2. Pass/ Fail result:

a. Pass: The actual result which match with the expected result.

b. Fail: The actual result which does not match with the expected result.

# Chapter 3 | Unit Test

**Unit Test Case ID:** UT01

**Test Method**

signupForm(name: String, email: String, password: String, dob: String, gender: String)

**Test by:** Sirisak Wongwal

**Test case**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected Return** | **P/F** |
| **General user** |
| 1 | All parameters valid form in the database. | {email: “[zmacktem2@hotmail.com](mailto:zmacktem@hotmail.com)”, password: “temhere123”, name: “tem”,  dob: “19/09/1997”,  gender: “male”} | “Register successfully created. Please Sign In” | P |
| 2 | The account has already in database. | {email: “[zmacktem@hotmail.com](mailto:zmacktem@hotmail.com)”, password: “temhere123”, name: “tem”,  dob: “19/09/1997”,  gender: “male”} | “Email is taken” | P |
| 3 | The password field is empty. | {email: “[zmacktem2@hotmail.com](mailto:zmacktem@hotmail.com)”, password: “temhere123”, name: “”,  dob: “19/09/1997”,  gender: “male”} | “Password is required” | P |
| 4 | The password field is invalid form. | {email: “[zmacktem2@hotmail.com](mailto:zmacktem@hotmail.com)”, password: “tem”,  name: “tem”,  dob: “19/09/1997”,  gender: “male”} | “Password must contain at least 6 characters” | P |

**Unit Test Case ID:** UT02

**Test Method**

editUserForm(name: String, email: String, password: String, about: String)

**Test by:** Kittinut Saengsri

**Test case**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected Return** | **P/F** |
| **Registered user** |
| 1 | All parameters valid form in the database and information in the database is changed. | {email: “[zmacktem2@hotmail.com](mailto:zmacktem@hotmail.com)”, password: “temhere123”, name: “tem”,  about: “Hello Tem”} | Redirect to user’s profile page | P |
| 2 | The account is not correct format. | {email: “[zmacktem2hotmail.com](mailto:zmacktem@hotmail.com)”, password: “temhere123”,  name: “tem”,  about: “Hello Tem”} | “Email must contain @” | P |
| 3 | The password field is not match with email. | {email: “[zmacktem2@hotmail.com](mailto:zmacktem@hotmail.com)”, password: “tem”,  name: “tem”,  about: “Hello Tem”} | “Password must contain at least 6 characters” | P |
| 4 | The email field is empty. | {email: “”,  password: “temhere123”, name: “tem”,  about: “Hello Tem”} | “This email field can not empty” | P |

**Unit Test Case ID:** UT03

**Test Method**

editPostForm(title: String, body: String, bodys: String)

**Test by:** Kittinut Saengsri

**Test case**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected Return** | **P/F** |
| **Registered user** |
| 1 | All parameters valid form in the database and information in the database is changed. | {title:”Omelet”,  body:”1. Egg 2. Olive oil”, bodys: “1. Crack an egg into the bowl. 2. Mix egg together 3. Put it into the pan. 4. Flip when it looks good. 5. Then it ready to serve.”} | Redirect the user to the user’s profile page. | P |
| 2 | The post form is not correct format title is empty. | {title:””,  body:”1. Egg 2. Olive oil”, bodys: “1. Crack an egg into the bowl. 2. Mix egg together 3. Put it into the pan. 4. Flip when it looks good. 5. Then it ready to serve.”} | “All fields are required” | P |
| 3 | The post form is not correct format body is empty. | {title:”Omelet”,  body:””,  bodys: “1. Crack an egg into the bowl. 2. Mix egg together 3. Put it into the pan. 4. Flip when it looks good. 5. Then it ready to serve.”} | “All fields are required” | P |
| 4 | The post form is not correct format bodys is empty. | {title:”Omelet”,  body:”1. Egg 2. Olive oil”, bodys: “”} | “All fields are required” | P |

**Unit Test Case ID:** UT04

**Test Method**

newPostForm(title: String, body: String, bodys: String)

**Test by:** Kittinut Saengsri

**Test case**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected Return** | **P/F** |
| **Registered user** |
| 1 | All parameters valid form in the database and post information in the database. | {title:”Omelet”,  body:”1. Egg 2. Olive oil”, bodys: “1. Crack an egg into the bowl. 2. Mix egg together 3. Put it into the pan. 4. Flip when it looks good. 5. Then it ready to serve.”} | Redirect the user to the user’s profile page. | P |
| 2 | The post form is not correct format title is empty. | {title:””,  body:”1. Egg 2. Olive oil”, bodys: “1. Crack an egg into the bowl. 2. Mix egg together 3. Put it into the pan. 4. Flip when it looks good. 5. Then it ready to serve.”} | “All fields are required” | P |
| 3 | The post form is not correct format body is empty. | {title:”Omelet”,  body:””,  bodys: “1. Crack an egg into the bowl. 2. Mix egg together 3. Put it into the pan. 4. Flip when it looks good. 5. Then it ready to serve.”} | “All fields are required” | P |
| 4 | The post form is not correct format bodys is empty. | {title:”Omelet”,  body:”1. Egg 2. Olive oil”, bodys: “”} | “All fields are required” | P |

**Unit Test Case ID:** UT05

**Test method**

signinForm(email: String, password: String)

**Test by:** Kittinut Saengsri

**Test case**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Description** | **Input** | | **Expected Return** | **P/F** |
| **email** | **password** |
| 1 | The email and password are corrected in the database. | “zmack\_tem@hotmail.com” | “temhere123” | Redirect the user to the homepage. | P |
| 2 | The email and password do not exist in the database. | “zmack\_temhotmail.com” | “temhere123” | “User with that email does not exist. Please register” | P |
| 3 | The email and password do not exist in the database. | “” | “temhere123” | “User with that email does not exist. Please register” | P |
| 4 | The email and password do not exist in the database. | “zmacktem@hotmail.com” | “temhere123456” | “Email and password do not match” | P |

**Unit Test Case ID:** UT06

**Test method**

handleChange(name: String, fileSize: int)

**Test by:** Sirisak Wongwal

**Test case**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Description** | **Input** | | **Expected Return** | **P/F** |
| **name** | **fileSize** |
| 1 | Set user data name and file size of photo. | “tem” | “0” | true | P |
| 2 | Set user data name and file size but not authorization . | “” | “0” | false | P |

**Unit Test Case ID:** UT07

**Test method**

read(userId: String, token: hashed\_Password)

**Test by:** Sirisak Wongwal

**Test case**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Description** | **Input** | | **Expected Return** | **P/F** |
| **userId** | **hashed\_Password** |
| 1 | User ID and token match. | “5d208546d6f6f23ee441c2fd” | “qiyh4XPJGsOZ2MEAyLkfWqeQ  ” | “redirectToSign: true” | P |
| 2 | User ID empty and token not empty. | “” | “qiyh4XPJGsOZ2MEAyLkfWqeQ” | “redirectToSign: false” | P |
| 3 | User ID not empty and token empty. | “5d208546d6f6f23ee441c2fd” | “” | “redirectToSign: false” | P |

**Unit Test Case ID:** UT08

**Test method**

remove(userId, token)

**Test by:** Sirisak Wongwal

**Test case**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Description** | **Input** | | **Expected Return** | **P/F** |
| **userId** | **token** |
| 1 | Remove user in the database. | “5d208546d6f6f23ee441c2fd” | “855c3697d9979e78ac404c4ba2c66533” | Remove in the database. | P |
| 2 | Unauthorized user to remove the user. | “” | “” | Show error unauthorized. | P |

**Unit Test Case ID:** UT09

**Test method**

signinForm(email: String, password: String)

**Test by:** Sirisak Wongwal

**Test case**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Description** | **Input** | | **Expected Return** | **P/F** |
| **email** | **password** |
| 1 | Email and password match. | “[zmacktem@hotmail.com](mailto:zmacktem@hotmail.com)” | “temhere123” | “redirectToReferer: true” | P |
| 2 | Email and password not match. | “[zmacktem@hotmail.com](mailto:zmacktem@hotmail.com)” | “tem” | “redirectToReferer: false” | P |
| 3 | Email not empty and password empty. | “[zmacktem@hotmail.com](mailto:zmacktem@hotmail.com)” | “” | “redirectToReferer: false” | P |
| 4 | Email empty and password not empty. | “” | “temhere123” | “redirectToReferer: false” | P |

**Unit Test Case ID:** UT10

**Test method**

signupForm(name: String, email: String, password: String, dob: String, gender: String)

**Test by:** Sirisak Wongwal

**Test case**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected Return** | **P/F** |
| **signupForm** |
| 1 | Name, email, dob, gender not empty. | name: “Kittinut Saengsri”  email: “[zmack\_tem@hotmail.com](mailto:zmack_tem@hotmail.com)”  dob: “1997/09/19”  gender: “male” | True | P |
| 2 | Name empty. | name: “”  email: “[zmack\_tem@hotmail.com](mailto:zmack_tem@hotmail.com)”  dob: “1997/09/19”  gender: “male” | False | P |
| 3 | Email empty. | name: “Kittinut Saengsri”  email: “”  dob: “1997/09/19”  gender: “male” | False | P |
| 4 | Dob empty. | name: “Kittinut Saengsri”  email: “[zmack\_tem@hotmail.com](mailto:zmack_tem@hotmail.com)”  dob: “”  gender: “male” | False | P |
| 5 | Gender empty. | name: “Kittinut Saengsri”  email: “[zmack\_tem@hotmail.com](mailto:zmack_tem@hotmail.com)”  dob: “1997/09/19”  gender: “” | False | P |

**Unit Test Case ID:** UT11

**Test method**

singlePost(postId)

**Test by:** Sirisak Wongwal

**Test case**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Description** | **Input** | **Expected Return** | **P/F** |
| **postId** |
| 1 | Get post from postId in the database. | “5d319cc13f5b8b0fc4038e07” | Get post | P |
| 2 | Invalid postId to get post in the database. | “555555” | Return false | P |

**Unit Test Case ID:** UT12

**Test method**

update(postId,post)

**Test by:** Kitinut Saengsri

**Test case**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **No.** | **Description** | **Input** | | **Expected Return** | **P/F** |
| **postId** | **post** |
| 1 | Update post information to the database. | “5d319cc13f5b8b0fc4038e07” | “Omelete(japan)” | Return true | P |
| 2 | Invalid postId to update post in the database. | “555555” | **“**Omelete(japan)**”** | Return false | P |

# Chapter 4 | Appendix A

Mock object

|  |  |
| --- | --- |
| **User** | **Return** |
| {email: “[zmack\_tem@hotmail.com](mailto:zmacktem@hotmail.com)”,  password: “temhere123”,  name: “Kittinut Saengsri”,  dob: “19/09/1997”,  gender: “male”,  about: “Hi Kittinut Saengsri”,  hased\_password: “qiyh4XPJGsOZ2MEAyLkfWqeQ” ,  userId: “5d208546d6f6f23ee441c2fd” ,  token: “855c3697d9979e78ac404c4ba2c66533”  } | “Success” |
| “” | null |

Mock object

|  |  |
| --- | --- |
| **Post** | **Return** |
| {title:”Omelet”,  body:”1. Egg 2. Olive oil”, bodys: “1. Crack an egg into the bowl. 2. Mix egg together 3. Put it into the pan. 4. Flip when it looks good. 5. Then it ready to serve.”} | “Success” |
| “” | null |

**Spotifood**

**Traceability Record**

**By**

**Mr. Kittinut Saengsri 592115003**

**Mr. Sirisak Wongwal 592115031**

**Department of Software Engineering**

**College of Arts, Media, and Technology**

**Chiang Mai University**

**Project Advisor**

**Dr. Passakorn Phannachitta**

Document History

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Document name | Version | Status | Date | Viewable | Editable | Responsible |
| Spotifood Traceability | V0.1 docx  - Add Chapter one  - Add chapter two  - Add chapter three | Draft | 07/16/2019 | SW, KS, PP | SW, KS | SW, KS |

\*SW = Sirisak Wongwal

\*KS = Kittinut Saengsri

\*PP = Dr.Passakorn Phannachitta

# CHAPTER 1 | INTRODUCTION

## 1.1 Purpose

The main purpose of developing the Spotifood is to help the people who are interested in cooking and who already have skills not bored with cooking or disconnected to prepare the food.

## 1.2 Acronyms and Definitions

### 1.2.1 Acronyms

SRS System Requirement Specification

URS User Requirement Specification

ID Identification Data

UT Unit Test

SD Sequence Diagram

AC Activity Diagram

UC Use Case

UI User Interface

# CHAPTER 2 | TRACEABILITY RECORD TABLE

## 2.1 URS-SRS-UC-UI-SD-AD

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| No. | URS | SRS | Use case | User interface | Sequence diagram | Activity diagram |
| 1. | URS01 | SRS-01  SRS-02  SRS-03  SRS-04  SRS-05  SRS-06  SRS-07  SRS-66 | UC01 | UI-01 | SD-01 | AC01 |
| 2. | URS02 | SRS-08  SRS-09  SRS-10  SRS-11  SRS-12  SRS-13 | UC02 | UI-02 | SD-02 | AC02 |
| 3. | URS03 | SRS-14  SRS-15  SRS-16 | UC03 | UI-03  UI-04  UI-05  UI-06  UI-07  UI-08 | SD-03 | AC03 |
| 4. | URS04 | SRS-17  SRS-18  SRS-19  SRS-20  SRS-21 | UC04 |  |  | AC04 |
| 5. | URS05 | SRS-22  SRS-23  SRS-24 | UC05 |  |  | AC05 |
| 6. | URS06 | SRS-25  SRS-26  SRS-27  SRS-28  SRS-29  SRS-30  SRS-66 | UC09 | UI-04 | SD-04 | AC09 |
| 7. | URS07 | SRS-31  SRS-32  SRS-33  SRS-34  SRS-35  SRS-66 | UC10 | UI-07 | SD-06 | AC10 |
| 8. | URS08 | SRS-36  SRS-37  SRS-38  SRS-39  SRS-65 | UC11  UC12  UC13 |  |  | AC11  AC12  AC13 |
| 9. | URS09 | SRS-40  SRS-41  SRS-65 | UC14 |  |  | AC14 |
| 10. | URS10 | SRS-42  SRS-65 | UC15 |  |  | AC15 |
| 11. | URS11 | SRS-65  SRS-67  SRS-68 | UC16 | UI-07 | SD-07 | AC16 |
| 12. | URS12 | SRS-46  SRS-47  SRS-48  SRS-49  SRS-50  SRS-51  SRS-52  SRS-53  SRS-66 | UC06 | UI-06 | SD-08 | AC06 |
| 13. | URS13 | SRS-59  SRS-60  SRS-61  SRS-62  SRS-66 | UC08 | UI-06 | SD-10 | AC08 |
| 14. | URS14 | SRS-54  SRS-55  SRS-56  SRS-57  SRS-58  SRS-66 | UC07 |  |  | AC07 |
| 15. | URS15 | SRS-25  SRS-28  SRS-32  SRS-45  SRS-51  SRS-66 | UC17 | UI-08 | SD-05 | AC17 |
| 16. | URS16 | SRS-63  SRS-64  SRS-65 | UC18 | UI-05 | SD-09 | AC18 |

# CHAPTER 3 | APPENDIX

## 3.1 User Requirement

URS01- User can register to the system by input email, password, name, date of birth, and gender.

URS02 - User can sign in to the system by input email and password.

URS03 - User can sign out of the system by sign out nav bar.

URS04 - Admin can sign in into the system by input email and password.

URS05 - Admin can sign out of the system by sign out nav bar.

URS06 - User can post the dish by input name of dishes, ingredients, how-to-cook of the dish, and thumbnail photos of the dish.

URS07 - User can delete the post of the dish.

URS08 - User can view a pin dish.

URS09 - User can view a history dish.

URS10 - User can select topic or type of food that user interested.

URS11- User can view a dish that user want.

URS12 - User can update their name, about, email and password.

URS13- User or upload and update their profile picture.

URS14- User can update their personality of interesting food.

URS15 - User can edit their post.

URS16 - User can view their profile page.

## 3.2 System Requirement

SRS-01: The system shall provide the user interface for receiving an email, password, name, and gender.

SRS-02: The system shall provide an error message “This field can not empty” when the textbox to empty.

SRS-03: The system shall provide a button to register.

SRS-04: The system shall validate the email, password, name, and gender of the user.

SRS-05: The system shall provide an error message “This [input] is incorrect format.” if input incorrect format of email, password, name, and gender.

SRS-06: The system shall provide an error message “This email already exists.” if the input email is already in database.

SRS-07: The system shall display a message “New account is successfully created. Please Sign In”.

SRS-08: The system shall provide the user interface for receiving email and password.

SRS-09: The system shall provide a button to sign in.

SRS-10: The system shall validate the email and password.

SRS-11: The system shall provide an error message “Incorrect password” if email is exist but password not match.

SRS-12: The system shall provide an error message “This email does not exist” if email does not exist in the database.

SRS-13: The system redirects user to homepage.

SRS-14: The system shall provide a button to sign out.

SRS-15: The system shall be logged out.

SRS-16: The system redirects user to the home page.

SRS-17: The system shall provide a user interface for receiving email and password.

SRS-18: The system shall validate the email and password.

SRS-19: The system shall provide an error message “Incorrect password” if email is exist but password not match.

SRS-20: The system shall provide an error message “This email does not exist” if email does not exist in the database.

SRS-21: The system redirects admin to homepage.

SRS-22: The system shall provide a button to sign out.

SRS-23: The system shall be logged out.

SRS-24: The system redirects admin to the home page.

SRS-25: The system shall provide the user interface for receiving data of the dish including :

- Name of dish

- Ingredients

- How-to-cook of the dish

- Upload thumbnail photos of food

SRS-26: The system shall provide a button to post.

SRS-27: The system shall validate the name of the dish, ingredients, and how-to-cook of the dish, and thumbnail photos of the dish.

SRS-28: The system shall provide an error message “This [input] field can not empty” when the textbox to empty.

SRS-29: The system shall provide an error message “File size should be less than 150kb” when file size is more than 150kb.

SRS-30: The system redirect registered user to the user’s profile.

SRS-31: The system shall provide the delete button for the user.

SRS-32: The system shall provide the confirm button for the user after click delete button.

SRS-33: The system shall delete the dish when the user clicks the delete button and confirm to delete the dish.

SRS-34: The system redirect registered user to homepage.

SRS-35: The system shall alert message “Are you sure to delete the post?”.

SRS-36: The system shall provide the pin button for the user.

SRS-37: The system shall cancel the pin dish when the user clicks a pin button again.

SRS-38: The system shall update the data into the database after the user clicks pin button.

SRS-39: The system shall provide the user interface to show a pin dish.

SRS-40: The system shall provide the history dish button to show a history dish.

SRS-41: The system shall redirect to history dish page when the user clicks the history button.

SRS-42: The system shall provide the user interface for receiving a type of food that user interested.

SRS-45: The system shall provide an error message “File size should be less than 150kb” when file size is more than 150kb.

SRS-46: The system shall provide the user interface for editing data of the dish including :

- Name

- About

- Email

- Password

SRS-47: The system shall provide the edit button to the user.

SRS-48: The system shall validate name , about , email and password.

SRS-49: The system saves the new data of user management to the database after the user input all data validation is passed.

SRS-50: The system shall provide an error message “This [input] field can not empty” when the textbox to empty.

SRS-51: The system shall provide an error message “A valid email is required” when the textbox is invalid format. (ex. dummy123@hotmail.com)

SRS-52: The system shall provide an error message “Email is taken!”. if input email is already in database.

SRS-53: The system redirect registered user to profile page.

SRS-54: The system shall provide the user interface for personality of interesting food.

SRS-55: User can select tag that user want to update on personality. And user click selected tag again to unselect.

SRS-56: The system shall provide a submit button to the user.

SRS-57: The system saves the new data of User management to the database.

SRS-58: The system redirects user to the homepage.

SRS-59: The system shall provide the user interface for upload profile picture.

SRS-60: The system shall provide the upload button to the user.

SRS-61: The system shall provide an error message “File size should be less than 150kb” when file size is more than 100kb.

SRS-62: The system redirects user to user’s profile page.

SRS-63: The system shall provide profile nav bar on the web application.

SRS-64: The system shall redirect user to their profile page.

SRS-65: The system will retrieve data in database.

SRS-66: The system shall update data in the database.

SRS-67: The registered user click a name of the dish.

SRS-68: The system shall redirect user to view dish page.

## 3.3 Use Case

UC01: User can register to the system.

UC02: User can log in to the system.

UC03: User can log out of the system.

UC04: Admin can log in to the system.

UC05: Admin can log out of the system.

UC06: User can update their profile.

UC07: User can update their personality of interesting food.

UC08: Edit profile picture.

UC09: User can post the dish.

UC10: User can delete the posted dish.

UC11: User can pin dish.

UC12: User can unpin dish.

UC13: User can view pin dish

UC14: User can view history.

UC15: User can select topic.

UC16: User can view dish.

UC17: User can edit their post.

UC18: User can view profile page.

## 3.4 Activity Diagram

AC01: Register

AC02: User sign in

AC03: User sign out

AC04: Admin sign in

AC05: Admin sign out

AC06: Edit profile

AC07: Edit personality

AC08: Edit profile picture

AC09: Create post

AC10: Delete post

AC11: Like post

AC12: Unlike post

AC13: View like post

AC14: View history

AC15: Select topic

AC16: View dish

AC17: Edit post

AC18: Vew profile

## 3.5 Sequence Diagram

SD-01 Register

SD-02 Sign in

SD-03 Sign out

SD-04 Create post

SD-05 Edit post

SD-06 Delete post

SD-07 View post

SD-08 Edit profile

SD-09 View profile

## 3.6 User Interface

UI-01: Register page

UI-02: Signin page

UI-03: Home page

UI-04: Create new post page

UI-05: Profile page

UI-06: Edit profile page

UI-07: The single dish post page

UI-08: Edit dish page