# **1** **Executive Summary - “Delectable”**

**Website that allows users to share their cooking recipes with the public and family/friends. Users will be able to post photos and videos to their feeds and attach descriptions to them containing recipes. When posted publicly, users will be able to rate recipes on a 5 star scale, with 5 stars being the highest. Users will be able to look up recipes; where posts with high ratings will be displayed as popular recipes and recently created posts will be displayed as new recipes. Recipes can be categorized by specific ingredients, nationalities, holidays, diets, time of day. Recipes will show nutritional information. Users will be able to create a shopping list.**

# **2** **Competitive analysis**

(Y)=yes

(N)=no

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Website name | Sharing recipe public/private to family/friends | Rating recipe after trying at home | Categorize recipes by (season, type, etc.) | Recipe feeds  (popular, new, trending) | Upload personal recipes to page |
| Delectable | Y | Y | Y | Y | Y |
| Yummly | N | Y | Y | Y | N  (pay to play) |
| Tasty | N | N | Y | Y | N |
| Instagram | Y | N | N | N | Y |
| Pinterest | Y | N | Y | N | Y |
| Facebook | Y | N | N | N | Y |

In the field of online recipes many websites contain useful blogs, articles or even copies of recipes but lack structure or the ability to upload recipes and share with friends. In our website “Delectable” the focus is sharing recipes amongst friends and family to keep people connected during social distancing. The app will allow a user to upload personal recipes and apply them to categories such as food type, seasonal or by meal. The recipe is saved to the user page for rating if public availability. A feed is available for users to view continuously updated with recipes. The website differs from Yummly and Tasty in the fact a user can upload their own recipes and allow for friends to rate. The difference from Instagram and Facebook is a dedicated web page for recipes allowing for category search. To remain competitive with websites Tasty and Yummly a shopping list feature will be saved to user profile so the user can refer to which items need to be purchased to make recipes. Delectable will be a go-to source of recipes for family and friends to reference when gathering for a meal.

<https://www.yummly.com/>

<https://tasty.co/>

<https://www.instagram.com/>

<https://www.facebook.com/>

https://www.pinterest.com/

# **3** **Data definition**

This section serves as the “dictionary” of your document. It defines main terms, data structures and “items” or “*entities*” *at high or logical (not implementation) level* (e.g. name, meaning, usage, and NOT how the data is stored in memory) so it is easier to refer to them in the document. Focus on key terms (main data elements, actors, types of users etc.) specific for your application and not on general well know terms. These terms and their names *must be used consistently* from then on in all documents, user interface, in naming software components and database elements etc. In later milestones, you will add more implementation details for each item. You will later expand this section with more details.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Meaning | Usage | Comments |
| User | actor | Use case scenarios | A person who is signed up in the system has an account |
| Guest | actor | Use case scenarios | A person who is using the system, but is not signed in. |
| Account | data | Use case scenarios | A collection of user’s information |
| Recipe | data | Use case scenarios | Main data unit in the system, contains instructions and media of the recipes |
| Review | data | Use case scenarios | Stores users comments about recipes |
| Cart | data | Use case scenarios | Contains items added by user for buying |
| Category | data | Use case scenarios | Each recipe has keys (categories) assigned to it |
| Recommended | data | Use case scenarios | List of highly rated recipes |
| New | data | Use case scenarios | Recently added recipes |
| Search | service | User service | Searches recipes by the name or ingredients |
| Filter | service | User service | Filters recipes by the category |
| Add | service | User service | Allows users to add new recipes |
| Favorite | service | User service | A list of recipes that user wish to save |
| Sign up | service | User service | Allows user to create an account |
| Sign in | service | User service | Allows user to enter the system |
| Sign out | service | User service | Allows user to leave the system |
| Remove | service | User service | Delete recipe from favorite list |
| Media | service | User service | Allows user to add photos or videos to the recipe or review |
| Home | User interface | User interface | Default page for users |
| Website | User interface | User interface | Collection of all the pages in the system |
| Profile | User interface | User interface | A page containing user information |
| Delectable | Website name | User interface | Name of the system |

# **4** **Overview, scenarios and use cases**

This section describes the project overview (in much more details) and likelihood usage scenarios of your product from end users’ perspectives. Focus only on main use cases. Simple text format is OK and preferable – tell us a story about who and how the application is used. Focus on WHAT users do, their skill level, not on HOW the system is implemented. You can expand use cases provided in high level document in future milestones.

Use Case: **Find**

Users or guests use the system’s search service to find recipes they like. Does not require the user to be signed in.

Use Case: **Add a recipe**

Users add a recipe through the system’s add service.Requires the user to be signed in.

Use Case: **Review and Rate**

Users add their rating and comments related to the particular recipe.Requires the user to be signed in.

Use Case: **Favorite**

Users add recipes to their “Favorite” list.Requires the user to be signed in.

# **5** **Initial list of high-level functional requirements**

This refers to the high-level functionality that you plan to develop to the best of your knowledge at this point. Focus on WHAT and not HOW. Keep the users in mind. Develop these functions to be consistent with use cases and requirements above. Number each requirement and use these numbers consistently from now on. For each functionality use 1-5 line description.



# 

# 

# 

# 

# 

# 

# 

# **6** **List of non-functional requirements**

For example, performance, usability, accessibility, expected load, security requirements, storage, availability, fault tolerance etc. Number each. When possible, try to quantify these quality attributes.

1. **Security-** The website should be publicly accessible to browse and view recipes easily. The recipes should be viewable in categories or as an updated feed. An account and login will be required to upload personal recipes and view private recipes from friends.
   1. Login and access levels
   2. Access permissions assigned by account holder for each upload.
   3. Account inactivity should automatically sign a user out after a period of two hours.
   4. Attempt to adhere to OWASP Application Security Verification Standard.
2. **Performance-** Delectable must compete with a large market of websites and must have a competitive response time. A response time over three seconds begins losing the user's attention. A response time of three seconds should be the goal of Delectable with a load
   1. The most important aspect of performance is how the website behaves in peak use hours. A response time under three seconds when used by up to 500 users simultaneously will ensure the user a smooth experience and prevent system crashes. This estimate is calculated with an estimate of 2000 users in a day/ 12 hours a day with a peak use time around typical American meal times with a multiplier of 5 to account for peak usage.
   2. Upload performance will be improved with the use of templates to input recipes.
   3. Website will be tested using Lighthouse Performance Score to judge performance on the user end.
3. **Accessibility-** For ease of use and accessibility for all the website should be available through major web browsers and supported to run on mobile. There is a standard used in industry, Web Content Accessibility Guidelines which will be referenced when building the website to make it available for anyone to use. When the Web meets this goal, it is accessible to people with a diverse range of hearing, movement, sight, and cognitive ability.
   1. Alternative text for images allows for auto screen readers to help visually impared individual to read a description of the photo.
   2. The ability to navigate a webpage without the user of a mouse allows for users with fine motor skill disabilities to use the features available to most others.
   3. Transcripts for audio files provide assistance to those who are hearing impared and makes a webpage more accessible.
   4. The ability to change font sizes of text entries on Delectable will enhance the accessibility for those with poor eyesight
4. **Maintainability-** To maintain a functional website after publication the design needs to follow strict guidelines.
   1. Understandability, meaning the code is structured and thoroughly documented for future reference. Conformance to Enterprise Architecture standard.
   2. Conforming to Technical design standards, a simple design is always best practice to ensure consistent performance.
   3. Conforming to coding standards and best practices will allow other future software developers to change and reference code segments efficiently.

<https://web.dev/performance-scoring/>

<https://www.perforce.com/blog/alm/what-are-non-functional-requirements-examples>

<https://www.w3.org/WAI/WCAG21/quickref/?showtechniques=121>

<https://owasp.org/www-project-application-security-verification-standard/>

https://www.webperformance.com/load-testing-tools/blog/2011/02/load-testing-basics-how-many-concurrent-users-is-enough/

**7** **High-level system architecture**

Lists of main software products, tools, languages and systems to be used, list of core APIs available at this point, supported browsers etc. You also have to decide on which frameworks you will use if any. These provide both user interface, as well as cross-platform and cross browser layout/css. All external code you plan to use must be listed along with their license

**Languages: HTML, CSS, Javascript, Python,**

**Frameworks: React, Bootstrap, Django**

**Supported browsers: Mozilla Firefox, Microsoft edge, Opera, Google Chrome, Safari**

**Software products: Microsoft Word, Whatsapp, Google docs, Zoom**

**Tools: Visual Studio Code, Github, Jira, WinSCP**

**Systems: Linux, Windows, Mac**

**API’s: Nutritionix, Google My Business API**

# **8** **Team**

List student group names, name of Scrum master, product owner and initial roles for each member

Zachary Astree (Web Developer)

Jakhongir Bekchanov (Scrum Master)

Hunter Grant (Software Developer)

Jonathan Laine (Product Owner)

Abdullah Abumazen (Software Developer)

# **9** **Checklist**

For each item below you must answer with only one of the following: DONE, ON TRACK (meaning it will be done on time, and no issues perceived) or ISSUE (you have some problems, and then define what is the problem with 1-3 lines). Reflect these items in your Jira project space:

a) Team decided on basic means of communications

DONE

b) Team found a time slot to meet outside of the class

ON TRACK

c) Front and back end team leads chosen

DONE

d) Github master chosen

DONE

e) Team ready and able to use the chosen back and front-end frameworks

ON TRACK

f) Skills of each team member defined and known to all

DONE

g) Team lead ensured that all team members read the final M1 and agree/understand it before submission

DONE

# **10** **Tasks before submission**

Teams must collaborate in creating M1 document by having working M1 document on their team GitHub repository (similar to managing code) so all team members can access it. Added advantage of doing it this way is that it builds teamwork and communication. We recommend having a folder for project documentation on team’s GitHub where milestones and other similar files can be kept.

# **11** **Submission**

Each team submits one single word document with all the above required sections to Canvas by the due date. Must have a title page to your document, including:

a) Course Title and term: CEN 4010 Principles of Software Engineering, Semester and Year

b) Document name: Milestone 1 Project Proposal and High-level description

c) Your team name, and project name (you can use the name you chose for your team)

d) Team number (I will assign you one)

e) Names of students (team lead first) with names and emails

f) Documentation Date

g) History table (revisions dates) (Note: you will update this document based on instructors’ feedback so this is important)