* **Capstone Proposal**

**Topic**

Cooking a good meal can be a time-consuming task, but sometimes finding the right recipe can take even more time. With my project, finding a recipe will be quick and easy. It will get recipes from a variety of websites all over the internet, all in one place - without the fluff that recipes online come with, just what you need to know. No more having to sift through sites and recipes just to find out the recipe takes too long or you’re missing an ingredient, everything you need to know about it will be on full display so you can quickly find what you want.

**Core Requirements**

* User can search for recipes using a keyword/term and the system will search multiple other recipe sites and find recipes on that site related to that term.
  + An account is not required to search for recipes
* The system will use a web scraper to scrape a variety of recipe sites to get a recipe’s:
  + Ingredients
  + Instructions
  + Duration
  + Title
  + Any images
* The system will use multiprocessing to search and scrape multiple websites at once so it’s fast and efficient
* The UI will display the recipes found from the search term on a web page using the info provided by the web scraper
  + A link to the site the information is from will also be provided if they want to go to the source of the recipe
* The home page will display random recipes from sites using a list of random search terms, to give ideas to the user if they don’t know what they want to search for.
* User can create an account with a username and password and login afterward using that same username and password
  + Info will be stored in a SQL server database
  + Password will be hashed and salted
* Users can save/bookmark recipes that are found to their account and will be able to see all their saved recipes on another page
  + Will store the recipe’s URL in a SQL database
* All services in the system will be registered through eureka
* A Spring API gateway will be used for all external requests

**Stretch Requirements**

* User can create their own recipes that will be stored in Mongo that will also show up if anyone searches using a term related to it
* Users can choose a certain website they want to search, e.g. only search allrecipes.com for what they are looking for
* Users can rate (out of 5) and comment on recipes to provide feedback

**Tech Stack**

Databases:

* SQL Server – store user information and their saved recipes
* MongoDB Atlas – store user created recipes

Back-End:

* .Net (C#) – A user service will use .Net that will handle the user accounts and activities. I will also use .Net for a recipe service that will handle the creation of recipes on the system.
* Spring (Java) – Spring will be used for the API Gateway that will route all requests from the web client
* Flask (Python) – Flask will be used for the web scraping service as I have the most experience scraping using python and will also be what searches other sites for recipes.

Front-End:

* Web client (Html, CSS, JavaScript) – Basic web client that will have a login and create account page, a page for recipes saved to your account, and the home page where you will also be able to search for recipes

A picture containing diagram

Description automatically generated

**Schedule**

|  |  |
| --- | --- |
| Week 1 | * Research Flask and how to setup a python API as it will be my first time creating a python API * Setup the project structure, services, and their respective databases |
| Week 2 | * Find out how to and then implement searching for recipes on multiple websites using a key word/term in Flask * I plan for this to take a while because I will also be learning flask along the way |
| Week 3 | * Scrape multiples websites and parse the html for the ingredients and instructions of the recipes * Research multiprocessing in python so the scraping is fast and efficient and begin implementing it |
| Week 4 | * Scrape websites and parse html for duration, title, and image * Finish implementing multiprocessing so multiple sites can be scraped at once |
| Week 5 | * Come up with the design scheme of the UI and create the HTML and CSS layouts of all the pages * Home/search page * Login/create account page * Saved recipes page |
| Week 6 | * Connect the UI and scraping service to display random recipes on home page and recipes searched for |
| Week 7 | * Set up the user service * Login/Create account * Save recipes * Connect the UI and user service |
| Week 8 | * Set up services for dockerization and create the dockerfiles * Set up service registration through eureka * Will have to research how to set up eureka in Flask and how to dockerize a python app |
| Week 9 | * Create the API gateway and begin testing to make sure all services are communicating correctly and find any bugs that need fixing |
| Week 10 | * Do any finishing touches and polishing that needs to be done * If I have time find ways to speed up the app more as web scraping can be very slow |