User Interaction(Front end): The user interacts with a user interface, a web application. They input the ingredients they have, like eggs and milk, and send this information to your API.

Features:

- Must Have::
    - Be able to accept text input from user
    - Http requests to the API that we are building
- Should have
    - Styling
    - Design landing page, make it good
    - User friendly designs
    - Loading screen when it makes request to API
- Could have
    - Mobile app
    - security(make it safe from attacks)
    - Word limits
    - Making sure that whats inputted is related to ingredients(ingredient validation)
    - Drop down menu to automatically fill out ingredient(no spelling errors)
    - Tells the user what ingredients have been selected
- Won’t have
    - Security

API Request: Your server receives a POST request at your recipe endpoint (e.g., /recipes). This request includes the ingredients the user has available, as a list in the request body.

API Response: Once your server has scraped a certain number of websites or found a certain number of recipes, it returns a response to the user. This response includes the list of recommended recipes in the response body.

Features:

- Must have
    - Endpoint for getting recipes (list of ingredients) ex: [milk, eggs, water]
    - Parse the ingredients
- Should have
    -
- Could have
    - Save recipes
        - (Database)
        - schema design

        - users
    - Validation
    - deployment

Won’t have

    - security

Scraping: Your server begins to scrape websites for recipes. For each website, it sends a GET request to the site's URL, parses the returned HTML to extract the recipe data, and checks if the recipe can be made with the user's ingredients.

Recommendation: If the scraped recipe can be made with the user's ingredients, it's added to a list of recommended recipes.

User Interaction: The user interface receives the response from your API and displays the recommended recipes to the user.

Tech Stack:

- Frontend:
    - **(must have)**
        - React
        - Axios
    - **(should have)**
        - Styling:
            - Mui,
            - (tailwind if somebody want to do some coding)
            - Bootstrap
    - **(could have)**
        - Redux (if we have a user data)
- Backend:
    - **(must have)**
        - Flask
    - **(should have)**
        - Gunicorn
    - **(could have)**
        - GraphQL
        - Apollo (if we have db)
        - Nginx (for deployment)

- ML Model
    - Word2vec?
    - Transformers?
- Database: **(could have)**
    - Redis? Postgres?
- CI/CD: **(could have)**
    - Docker
    - K8s

Agenda: (overall project schedule guide)

- Introductions
- Overview of project scope
    - Mainly api (ML)
    - Simple frontend
    - One resource for the backend
        - Pre processing
- Feedback -> brainstorm -> possible changes / direction
    - User stories (define them)
    - What do team members want to create
    - Run through initial presentation
    - Run through revise plan outline
- Check in on interest?
- Admin stuff
    - Scheduling
    - People's interests / skill sets
        - Project manager / project owner?, scrum master
- Start developing tasks -> airtable or trello -> figma designing
    - Setup airtable or github project (kanban board)
        - Storypoints for sprint, look at agile docs
    - Setup figma
        - System design and layout
- Sprint planning
    - Story points, ideal hours for a work week, people's avaliablity
    - template
- Setup repos?
    - Create github organization
    - Outline stack first and folder directories
    - Create-react-app?
    - App.py, requirements.txt
    - Ml models
- Outline cross repo data types:
    - Schema for object request
    - ML training data csv
    - Input for ML

    - Other stuff that pops up

Steps:

1. Data collection(web scraping):(Sayak,
    a. Setup a python environment(newest version of python)
    b. Write a script to scrape
        i. **(Must have)**
            1. Main dishes in all recipes - scrape all the main dishes
                a. Only get ingredients, not measurements and other gunk
            2. Plan out schema to store on CSV(Collaborate with ML team)
            3. Send to .csv
        ii. **(Could haves)**
            1. Other categories of dishes
                a. Get Breakfast, lunch, and dinner data
    c. **End goal:** Extract a large dataset of recipes,including their ingredients
2. Data processing(ML): (Raghavendra, Colder, Aaron)
    a. After collecting we need to clean up the data and structure it for further analysis
        i. Take out unnecessary words and letters
            1. “Fresh chicken breasts” -> “chicken breast”
            2. Roma tomatoes → tomatoes
            3. free-range egg yolks → egg yolks
3. Feature extraction(Word Embedding)(ML/NLP step 1):
    a. Take your parsed ingredients and convert them into a form that can understood by machine learning models
        i. Word2vec and TF-IDF to create word embeddings for ingredients
    b. (Must Have)
        i. Convert the parsed ingredients so that it can be understood by machine learning model (embedded vector)
        ii. Something like one hot encoding (ingredients in a recipe)
        iii. Create word embeddings
        iv. Common ingredients (figure out what number is enough)
4. Build a recommendation system(ML):
    a. (Must Have)
        i. Use cosine similarity to measure euclidean distance between the word embeddings
        ii. Tf-idf - creates higher weights for different ingredient
        iii. Define hyperparameters
        iv. Evaluate our model? Manually
            1. Choosing n amount of best recipes
5. Build API with flask**(Sayak,**

    a. Api accepts post requests where the body contains the users ingredients
        i. (Must Have)
            1. Define routes
            2. Http method handing (flask_restful)
        ii. (Could Have)
            1. Define database schema
        iii. Def get():
            1. If model is loaded (pkl)
            2. Then it run model(request payload)
            3. Result
            4. Return jsonify(result)
    b. Api calls recommendation system to get a list of recommended recipes

6. Create a an app(React, streamlit-python):**(Sayak, Yera, Hisham)**
    a. Create user interface for api
    b. App allows users to input ingredients and see recommended recipes
    c. (Must have)
        i. Designing layout
        ii. Be able to handle user input
        iii. User friendly interface
        iv. Drop down menu for ingredients
        v. Handle the response from ML model and display it well
    d. (**Could have)**
        i. Mobile app
        ii. Chips for visualizing what you imputed
        iii. Animations to create 3d component (component library, MUI, bootstrap, tailwindcss)
        iv. Loading screen

TODO:

- Sprint tasks:
-

-