Post Research Plan

# Website:

* Recipes: The website will contain recipes for a few popular dishes. The user can query the chatbot about recipes. They can upload recipes of their own and ask for alternatives for certain ingredients which the chatbot will provide.
  + It should be easy to make and find content
  + Allows for users to upload documents (PDF, docx, xlsx, txt, …)
    - Recipes
    - Meal plans
    - Shopping lists
    - Digital receipts
    - Summarize long recipes
  + Many different questions a user may ask
  + Acts as an example use case for the chatbot

## Front End:

* HTML, CSS, JS

## Back End:

* Python with Flask
  + Should allow us to build a simple website to test the Chatbot
  + Allows for usage of Jinja2 templates to quickly make webpages
  + Python will be able to interact with SQL databases easily
* AWS Server
  + Reasonably cheap
  + Flexible

# API:

* Uses ChatGPT to process input text
  + ChatGPT API keys are unique to each website
    - So that each website has their own unique model
    - This avoids issues with the cost of running a ChatGPT API model
* Flask in Python
  + The same as the website so should speed up development
  + Python gives us access to many libraries to help us parse different document types
* SQL database with MariaDB
  + We are all familiar with using these databases and it should allow for a scalable solution
  + Use Amazon RDS (Relational Database Service) provided by AWS to create a MariaDB instance. RDS simplifies database management tasks such as backups, scaling, and maintenance.
* AWS Server (separate to the website)
  + [As above]

## Workflow:

1. Accepts input string
   * The calling website is also identified in order to select the correct model and processing.
2. Pre-processes the string
   * Rejects irrelevant questions
   * Parses documents
   * Checks database
3. Sends to ChatGPT via LangChain
   * [This may be skipped if input is rejected]
4. Post-processing
   * Updating the database
5. Returns output string to API caller