CookBook

group 8

Anish Mulay

Ashwattha Phatak

Akshay Dongare

Check out the website

Check out the repo!

What is better in this version

- Added feature for social sharing using Whatsapp
- Advanced recipe recommendation system, meal planning and grocery list generation using Large Language models
- Personalized recipe suggestions based on dietary preferences
- Dockerized implementations which ensures quick installation and deployment for further work
-

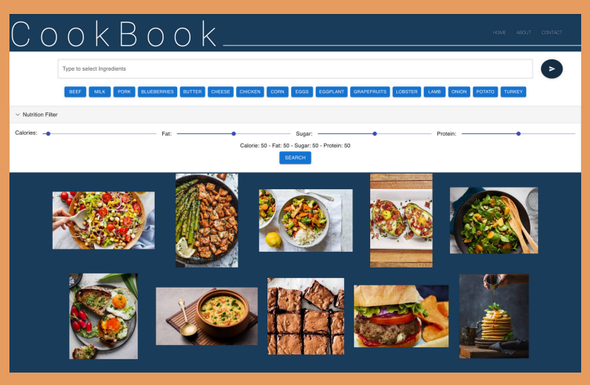
What can be done in the future

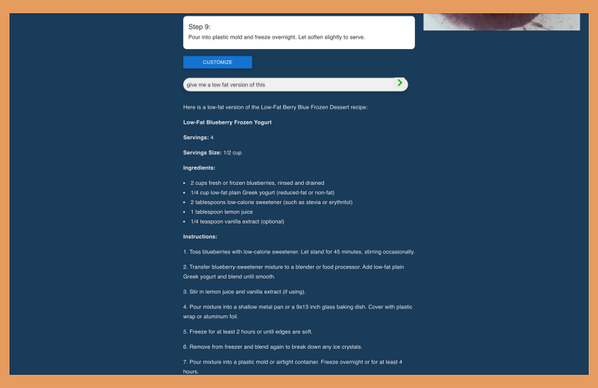
- Add user profiles where users can manage their saved recipes and preferences.
- Add more interactive elements and animations for a better user experience.
- Implement dark mode for better accessibility.
- Implement integrations testing for end-to-end application testing
- Add a feature for users to submit their own recipes and share them with the community.

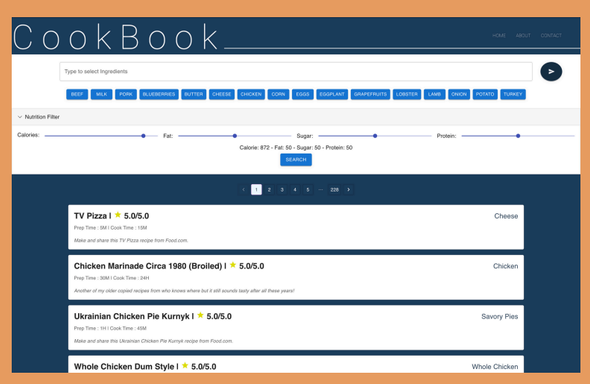
Tech Stack

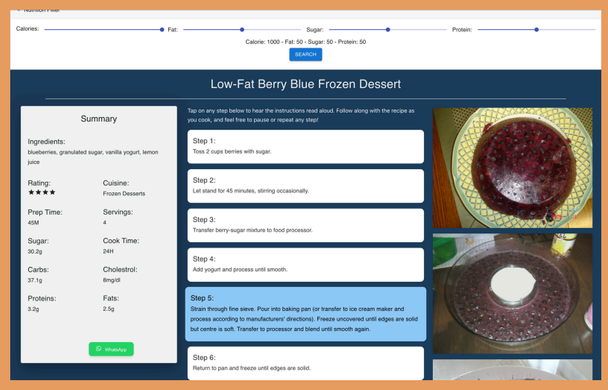
- React + TypeScript
- FastAPI and Python
- MongoDB
- Docker
- GROQ (Llama 3.2)

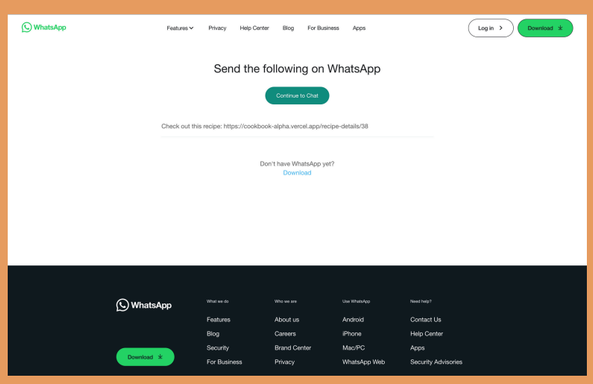
Screen Snaps











Project 2

Group 8

Anish Mulay (amulay2)
Ashwattha Phatak (aaphatak)
Akshay Dongare (adongar)

Github repository
https://github.com/AMAPAD/CookBook

Table

Notes	Score	Evidence
Workload is spread over the whole team	3	Each member worked on separate new features and on their own branches.
Number of commits	3	66 commits
Number of commits per person	3	Akshay Dongare: 32 Anish Mulay: 20 Ashwattha Phatak: 14
Issues reports: there are many	1	We inherited one issue from the previous iteration of the project which we have resolved.
Issues are being closed	3	Evidence in github repository
Docs: doco generated, format not ugly	3	Evidence in github repository
Docs: what: point descriptions of each class/function (in isolation)	3	Evidence in github repository
Docs: how: for common use cases X,Y,Z mini-tutorials showing worked examples on how to do X,Y,Z	1	Instructions on testing the backend and the frontend apps individually can be found in INSTALL.md

Docs: why: docs tell a story, motivate the whole thing, deliver a punchline that makes you want to rush out and use the thing	1	Evidence in README.md
Docs: short video, animated, hosted on your repo. That convinces people why they want to work on your code.	0	
Use of version control tools	3	Evidence in github repository
Test cases exist	3	We added almost 60 new test cases, 20 per member. Evidence in github repository
Test cases are routinely executed	3	Evidence in github repository workflows
Issues are discussed before they are closed	1	Issue #10 has been discussed and closed.
Chat channel: exists	0	
Test cases: a large proportion of the issues related to handling failing cases.	1	Evidence in github repository
Evidence that the whole team is using the same tools: everyone can get to all tools and files	3	Each member used the same environment files. We have not committed these files to the repo.
Evidence that the whole team is using the same tools (e.g. config files in the repo, updated by lots of different people)	3	Evidence in github repository
Evidence that the whole team is using the same tools (e.g. tutor can ask anyone to share screen, they demonstrate the system running on their computer)	1	Containerisation made sharing changes and parallel development easier. Evidence in github repository
Evidence that the members of the team are working across multiple places in the code base	3	Each member has worked both on the frontend and backend. Evidence in commit history

Short release cycles	3	Each new feature had its own branch which allowed for parallel development and faster merging.
The file .gitignore lists what files should not be saved to the repo. See [examples]i(https://github repository.com/github repository/gitignore)	3	Evidence in github repository
The file INSTALL.md lists how to install the code	3	Evidence in github repository
The file LICENSE.md lists rules of usage for this repo	3	Evidence in github repository
The file CODE-OF-CONDUCT.md lists rules of behavior for this repo; e.g. see example	3	Evidence in github repository
The file CONTRIBUTING.md lists coding standards and lots of tips on how to extend the system without screwing things up; e.g. see example	3	Evidence in github repository
The file README.md contains all the following		
Video	3	Evidence in README.md
DOI badge: exists. To get a Digitial Object Indentifier, regiser the project at Zenodo. DOI badges look like this:	3	Evidence in README.md
Badges showing your style checkers	3	Evidence in README.md
Badges showing your code formatters.	3	Evidence in README.md
Badges showing your syntax checkers.	3	Evidence in README.md
Badges showing your code coverage tools	3	Evidence in README.md

Badges showing any other Other automated analysis tools	3	Evidence in README.md
	Total = 81	

Cookbook - Sustainability Evaluation

This software evaluation report is for your software: Cookbook. It is a list of recommendations that are based on the survey questions to which you answered "no".

If no text appears below this paragraph, it means you must already be following all of the recommendations made in our evaluation. That's fantastic! We'd love to hear from you, because your project would make a perfect case study. Please get in touch (info@software.ac.uk)!

Question 1.3: Do you publish case studies to show how your software has been used by yourself and others?

A great way of showing off your software is to write case studies about how yourself, and others, have used it. Case studies can help potential users learn about your software. They also act as a great advert for your software. If you can show happy users benefiting from your software, you are likely to gain more users.

Question 2.1: Is the name of your project/software unique?

You shouldn't choose a project or software name that is shared by another group – especially if they are a competitor – but, sadly, few people spend enough time researching the uniqueness of their project or software names. It is, now, less important to have a domain name that mimics the name of your project or software, because most people will use a web search to find a website rather than manually entering a URL. However, it is important to check, using a web search, that there are no existing domains that include the name of your project or software that are owned by another group, because this can confuse your users.

See our guide on Choosing project and product names (http://software.ac.uk/resources/guides/choosing-project-and-product-names).

Question 3.1: Is your software available as a package that can be deployed without building it?

Building software can be complicated and time-consuming. Providing your software as a package that can be deployed without building it can save users the time and effort of doing this themselves. This can be especially valuable if your users are not software developers.

You should test that your software builds and runs on all the platforms it is meant to support, which means you will already have created packages that can be distributed to your users!

See our guide on Ready for release? A checklist for developers (http://www.software.ac.uk/resources/guides/ready-release).

If you're interested in the consequences of ignoring your users' needs, see our guide on How to frustrate your users, annoy other developers and please lawyers (http://www.software.ac.uk/resources/guides/how-frustrate-your-users-annoy-other-developers-and-please-lawyers).

Question 4.5: Do you provide troubleshooting information that describes the symptoms and step-by-step solutions for problems and error messages?

Troubleshooting information helps users quickly solve problems. It can help to reduce the number of support queries that you have to deal with.

Question 4.8: Do you publish your release history e.g. release data, version numbers, key features of each release etc. on your web site or in your documentation?

A release history allows users to see how your software has evolved. It can provide them with a way to see how active you are in developing and maintaining your software, in terms of new features provided and bugs fixed. Software that is seen to be regularly fixed, updated and extended can be more appealing than software that seems to have stagnated.

Question 5.3: Does your project have an e-mail address or forum that is solely for supporting users?

E-mail is purpose-made for resolving users' problems. The user can provide a good description of their problem and can attach screenshots, log files or other supporting evidence, and it's easy for you to ask for follow-up information, if required. You should always try to have an e-mail address for support queries.

It's best if the support e-mail address is clearly labelled as such e.g. myproject-support@myplace.ac.uk. This makes it easy for users to identify the e-mail address on your website or within your documentation, and it helps you to separate your support queries from all of your other e-mail. However, a personal e-mail address is better than nothing if you don't have the means to provide a dedicated support address.

See our guide on Supporting open source software (http://software.ac.uk/resources/guides/supporting-open-source-software). Its advice applies to supporting closed source software too.

Question 5.6: Is your project's ticketing system publicly visible to your users, so they can view bug reports and feature requests?

An open ticketing system allows your users to see that you are active in fixing bugs and implementing features, and are responsive to your users. This can give them confidence in you and your software and makes them more likely to use it. It also provides your users with a means to see if a problem they have is a known issue, and allow them to check progress on it, or, even, whether their issue has already been addressed and a fix is available.

See our guide on Supporting open source software (http://software.ac.uk/resources/guides/supporting-open-source-software). Its advice applies to supporting closed source software too.

Question 10.5: Do you back-up your repository?

While third-party repositories are very useful, it can be reassuring to know that you have a local back-up of your repository in case the third-party repository goes down for any length of time, or the host ceases to provide the repository hosting service. Distributed revision control repositories such as Git and Mercurial can be easy to back-up locally, just by cloning the repository. Centralised repositories such as Subversion or CVS rely upon the repository hosting service to provide you with the functionality to back-up your repository (see, for example SourceForge Subversion back-ups (http://sourceforge.net/p/forge/documentation/svn/#backups).

Question 12.5: Are all manually-run tests documented?
It may not be possible, or easy, to automate certain tests e.g. testing a browser-based application after it's been deployed. In such cases, you should document the list of steps that are to be done to test the software. Documenting the steps means that the tests can be run by anyone, not just the developer who usually does these tests.

Question 13.1: Does your project have resources (e.g. blog, Twitter, RSS feed, Facebook page, wiki, mailing list) that are regularly updated with information about your software? (e.g. release announcements, publications, workshops, conference presentations)
These resources are all good ways of showing that your project is active. If potential users see frequent posts, especially if they talk about new features and resolved problems, they know that your project is thriving, your software is useful and is under active development. This may encourage them to use your software, knowing that if they run into problems, there may be people who can help, and who they can share experiences with.
These resources also give you a way of getting in touch with your current users, keeping them engaged, and asking for their input or help with problems. Don't know what new feature to implement next? Ask your users whether they think it would be useful.

Question 13.2: Does your website state how many projects and users are associated with your project?
Where you have an active set of users and developers, advertising their existence is not just good for promoting the success and life of your project. If potential users see that there are a large number of users, they know that your project is thriving, your software is useful and is under active development. This may encourage them to use your software, knowing that if they run into problems, there may be people who can help, and who they can share experiences with.

Question 13.3: Do you provide success stories on your website?
A great way of showing off your software is to write case studies about the people who've used it and how they've used it. This helps potential users learn about the software but, more to the point, is a great advert for your software. If you can show happy users benefiting from your software, you are likely to gain more users.

Question 13.5: Do you list your project's publications on your website or link to a resource where these are available?
Listing your software publications provides an academic perspective on the value of your software. It can also help users, and other stakeholders (e.g. current and potential funders) to understand, in detail, how your software contributes to research, what scientific problems it has helped to solve. In addition, these can help to show where your software sits in relation to other software that fulfils a similar need, and what makes yours different, or better.
These publications also gives researchers something to cite when they write their own papers where your software has been used, which is of value to them and also increases your citation count for your papers, which helps you demonstrate your impact!

Question 13.6: Do you list third-party publications that refer to your software on your website or link to a resource where these are available?
Providing a list of third-party publications can show, academically, how the software is used by others, as well as promoting their efforts and successes.

It also gives potential users ideas for how they may choose to use the software, as well as providing assurance that the software can be used by people other than its original developers to achieve something. Having such a list also means you can cite these publications in your own papers, funding proposals and reports to show or justify its value and the impact you have made! As a matter of routine, you should always ask people to cite your software if they've used it in their research for these reasons, and to inform you if they have included such a reference in one of their papers.

Question 15.2: Does each of your source code files include a copyright statement?

It's easy to distribute source code files, and this separates the code from any copyright statement that might be on your web site or in your documentation. To cover this eventuality, and remove any ambiguity about ownership, it's good practice to include a copyright statement with each of your source code files, as a comment, or, if the language permits it, as a string constant.

Question 15.6: Does each of your source code files include a licence header?

It's easy to distribute source code files, and this separates the code from any licence statement that might be on your web site or in your documentation. To cover this eventuality, and remove any ambiguity about what a developer can do with the source code, it's good practice to include a licence statement within each of your source code files, as a comment. This can also help to avoid confusion between source files that may have different licences, particularly if there are a number of third-party dependencies used within your software.

Question 16.2: Does your website or documentation describe how your project is funded, and the period over which funding is guaranteed?

Especially on academic projects, users will view the active lifetime of software to be the duration of the software's project funding. If you want to persuade users that your software will be around in the future, it is a good idea to describe your funding model and the duration over which funding is assured.