Recipe Recommendation System

Test Cases

There are 60 new **test cases** that cover the entire functionality of the recommendation system

Repository Link

Demo Video Link

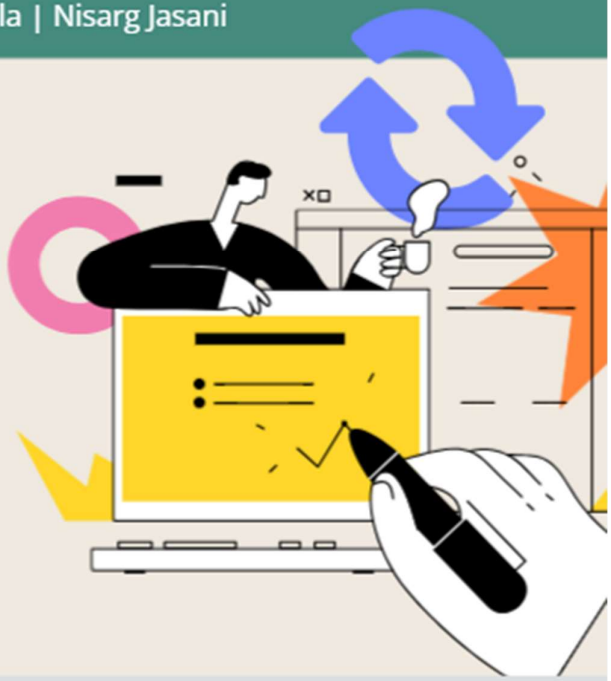
Group 2
Jinish Shah | Nirmit Deliwala | Nisarg Jasani

What is Recipe Recommendation System?

The Recipe Recommendation System is a web-based application designed to simplify meal planning and cooking by providing personalized recipe suggestions based on user preferences, dietary restrictions, and available ingredients.

Why Recipe Recommendation System?

This innovative platform addresses the common challenge of deciding what to cook, especially when considering nutritional needs and ingredient availability.



New Features and Improvements

Personalized Diet Planner

- Implemented an unsupervised machine learning model (KNN) for personalized recipe recommendations.

Historical Meal Tracking

- Users can now bookmark meals they love and easily access them later.
- Track past meals to analyse eating habits and nutritional intake over time.

Backend Optimization & Testing

- Backend updated to handle a larger dataset of 50,000 recipes, ensuring faster load times.
- Added comprehensive test cases to improve reliability and ensure smooth functionality.

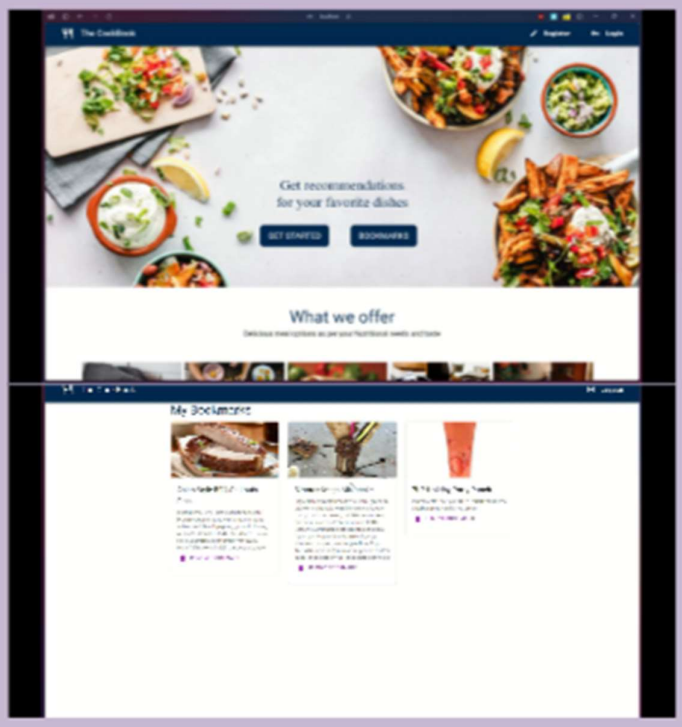
Methodology

1. Frontend
Improved the user interface by removing unused features such as order tracking, which were no longer relevant. The frontend code was cleaned up and restructured to enhance readability and maintainability, providing a smoother user experience.

2. Backend
Migrated the database to MongoDB Atlas, expanding it from a minimal setup to over 50,000 entries.
This ensures faster query times and improved performance. With the new database structure, the system can handle a larger dataset efficiently, supporting user preferences, ingredients, and recipe data.

3. AI-Powered Recommendations
Integrated a machine learning model (K-Nearest Neighbours) into the backend, enabling personalized recipe recommendations.
The system now generates recipe suggestions by analysing user preferences and available ingredients, providing tailored options for each user.
The KNN model searches for closer data points in the expanded recipe database to offer relevant recipes based on user input.

Screenshots



Tech Stack

Future Improvements

- **Community Features**: Introduce a social platform within the app where users can share their own recipes, rate others' recipes, and discuss cooking tips or modifications. Users can create profiles and follow each other for inspiration and collaboration.
- **Ordering Functionality**: This feature will allow users to directly order ingredients or meal kits through the application. Users can select recipes they want to cook, and the system will generate a list of required ingredients. They can then place an order for these ingredients, which will be managed by an administrator interface.
- **AI Chatbot Integration**: An AI-powered chatbot will be integrated into the system to provide real-time assistance to users. The chatbot will help users with recipe suggestions, cooking tips, ingredient substitutions, and general queries related to meal planning and nutrition.
- **Globally Available Dietary Preferences**: Expand the system to cater to a wider range of global dietary preferences and restrictions. This could include options like Halal, Kosher, Vegan, Gluten-Free, and other culturally or religiously significant dietary needs

RECIPE RECCOMENDER SYSTEM

RUBRICS

RUBRICS	
Number of commits	2
Number of commits: by different people	3
Issues reports: there are **many**	3
Issues are being closed	3
Docs: doco generated, format 0t ugly	3
Docs: what: point descriptions of each class/function (in isolation)	2
Docs: how: for common use cases X,Y,Z mini-tutorials showing worked examples on how to do X,Y,Z	2
Docs: why: docs tell a story, motivate the whole thing, deliver a punchline that makes you want to rush out and use the thing	3
Docs: short video, animated, hosted on your repo. That convinces people why they want to work on your code.	3
Use of version control tools	1
Test cases exist	3
Test cases are routinely executed	3
Issues are discussed before they are closed	3
Chat channel: exists	3
Test cases: a large proportion of the issues related to handling failing cases.	2
Evidence that the whole team is using the same tools: everyone can get to all tools and files	3
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 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)	3
Evidence that the members of the team are working across multiple places in the code base	3
Short release cycles	1
The file .gitig0re lists what files should 0t be saved to the repo. See [examples]i(https://github.com/github/gitig0re)	3
The file INSTALL.md lists how to install the code	3
The file LICENSE.md lists rules of usage for this repo	3
The file CODE-OF-CONDUCT.md lists rules of behavior for this repo; e.g. see example	3
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
The file README.md contains all the following	3
Video	3

DOI badge: exists. To get a Digitial Object Indentifier, regiser the project at Ze0do. DOI badges look like this:	
Badges showing your style checkers	3
Badges showing your code formatters.	3
Badges showing your syntax checkers.	3
Badges showing your code coverage tools	3

Total = 85

SOFTWARE EVALUATION METRICS

Questions	**Answer**
	Total Score = 99
What is the name of your software?	**Recipe Recommendation System**
Q1 - What your software does	
Does your website and documentation provide a clear, high-level overview of your software?	**3**
Does your website and documentation clearly describe the type of user who should use your software?	**1**
Do you publish case studies to show how your software has been used by yourself and others?	**0**
Q2 - Your project's and software's identity	
Is the name of your project/software unique?	**0**
Is your project/software name free from trademark violations?	**3**
Q3 - Availability of your software	
Is your software available as a package that can be deployed without building it?	**0**
Is your software available for free?	**3**
Is your source code publicly available to download, either as a downloadable bundle or via access to a source code repository?	**3**
Is your software hosted in an established, third-party repository likeGitHub (https://github.com)	**3**
Q4 - Your software's documentation	
Is your documentation clearly available on your website or within your software?	**3**
Does your documentation include a "quick start" guide, that provides a short overview of how to use your software with some basic examples of use?	**2**
If you provide more extensive documentation, does this provide clear, step-by-step instructions on how to deploy and use your software?	**1**

Do you provide a comprehensive guide to all your software's commands, functions and options?	**2**
If your software can be used as a library, package or service by other software, do you provide comprehensive API documentation?	**0**
Do you store your documentation under revision control with your source code?	**0**
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?	**0**
Q5 - How you support your software	
Does your software describe how a user can get help with using your software?	**3**
Does your website and documentation describe what support, if any, you provide to users and developers?	**3**
Does your project have an e-mail address or forum that is solely for supporting users?	**3**
Are e-mails to your support e-mail address received by more than one person?	**0**
Does your project have a ticketing system to manage bug reports and feature requests?	**0**
Is your project's ticketing system publicly visible to your users, so they can view bug reports and feature requests?	**0**
Q6 - Your software's maintainability	
Is your software's architecture and design modular?	**2**
Does your software use an accepted coding standard or convention?	**3**
Q7 - Open standards and your software	
Does your software allow data to be imported and exported using open data formats?	**0**
Does your software allow communications using open communications protocols?	**0**
Q8 - Your software's portability	
Is your software cross-platform compatible?	**3**
Q9 - Your software and accessibility	
Does your software adhere to appropriate accessibility conventions or standards?	**2**
Does your documentation adhere to appropriate accessibility conventions or standards?	**2**
Q10 - How you manage your source code	
Is your source code stored in a repository under revision control?	**2**
Is each source code release a snapshot of the repository?	**0**
Are releases tagged in the repository?	**0**
Is there a branch of the repository that is always stable? (i.e. tests always pass, code always builds successfully)	**0**

Do you back-up your repository?	**3**
Q11 - Building and installing your software	
Do you provide publicly-available instructions for building your software from the source code?	**3**
Can you build, or package, your software using an automated tool?	**1**
Do you provide publicly-available instructions for deploying your software?	**3**
Does your documentation list all third-party dependencies?	**3**
Does your documentation list the version number for all third-party dependencies?	**3**
Does your software list the web address, and licences for all third-party dependencies and say whether the dependencies are mandatory or optional?	**3**
Can you download dependencies using a dependency management tool or package manager?	**0**
Do you have tests that can be run after your software has been built or deployed to show whether the build or deployment has been successful?	**0**
Q12 - How you test your software	
Do you have an automated test suite for your software?	**0**
Do you have a framework to periodically (e.g. nightly) run your tests on the latest version of the source code?	**0**
Do you use continuous integration, automatically running tests whenever changes are made to your source code?	**0**
Are your test results publicly visible?	**1**
Are all manually-run tests documented?	**3**
Q13 - How you engage with your community	
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?	**0**
Does your website state how many projects and users are associated with your project?	**3**
Do you provide success stories on your website?	**0**
Do you list your important partners and collaborators on your website?	**3**
Do you list your project's publications on your website or link to a resource where these are available?	**0**
Do you list third-party publications that refer to your software on your website or link to a resource where these are available?	**3**
Can users subscribe to 0tifications to changes to your source code repository?	**0**
If your software is developed as an open source project (and, 0t just a project developing open source software), do you have a governance model?	**0**
Q14 - How you manage contributions	
Do you accept contributions (e.g. bug fixes, enhancements, documentation updates, tutorials) from people who are 0t part of your project?	**3**
Do you have a contributions policy?	**0**

Is your contributions' policy publicly available?	**0**
Do contributors keep the copyright/IP of their contributions?	**0**
Q15 - Your software's copyright and licensing	
Does your website and documentation clearly state the copyright owners of your software and documentation?	**3**
Does each of your source code files include a copyright statement?	**3**
Does your website and documentation clearly state the licence of your software?	**3**
Is your software released under an open source licence?	**3**
Is your software released under an OSI-approved open-source licence?	**3**
Does each of your source code files include a licence header?	**0**
Do you have a recommended citation for your software?	**0**
Q16 - Your plans for the future	
Does your website or documentation include a project roadmap (a list of project and development milestones for the next 3, 6 and 12 months)?	**2**
Does your website or documentation describe how your project is funded, and the period over which funding is guaranteed?	**0**
Do you make timely an0uncements of the deprecation of components, APIs, etc.?	**0**