International Scientific Journal of Engineering and Management (ISJEM) ISSN: 2583-6129
Volume: 04 Issue: 04 | April – 2025 DOI: 10.55041/ISJEM03013
An International Scholarly || Multidisciplinary || Open Access || Indexing in all major Database & Metadata

# ADDING RECIPES WEBSITE USING DJANGO

**Agashini V Kumar**
**Computer Science and Engineering Jain Deemed to be University Bangalore, Karnataka**
**agashini.v@jainuniversity.ac.in**

**D. Yaswanth Pavan**
Computer Science and Engineering Jain Deemed to be University Bangalore, Karnataka
21btlcn004@jainuniversity.ac.in

**Abhay Gowda**
Computer Science and Engineering Jain Deemed to be University Bangalore, Karnataka
21btrcs029@jainuniversity.ac.in

**Priyansh Sharma**
Computer Science and Engineering Jain Deemed to be University Bangalore, Karnataka
21btlcn006@jainuniversity.ac.in

Abstract— This project is a web-based application that was created with the Django framework and allows users to effectively maintain their favourite recipes. Key features of the application include user registration, login, adding new recipes, changing old ones, removing recipes, and browsing through them. Django's ORM is used to handle the database that contains each recipe's name, description, and optional image. Through the usage of Django's integrated User model, the application guarantees safe user authentication and offers customised recipe management by associating recipes with specific users. The system renders dynamic HTML pages for user login (login.html), recipe submission (receipes.html), updating (update_receipes.html), and registration (register.html) using Django's templating engine. Bootstrap is used to create a responsive, user-friendly user interface, and media handling is utilised to save uploaded recipe photos.

***Keywords: Python, MVC Architecture, Web Applications, Django, Recipe Management, and CRUD***

## I. Introduction

Organising and sharing one's own culinary creations online has grown in popularity in the era of technological convenience. This web-based application, "Adding Recipes Website Using Django," was created to give customers an easy-to-use platform for managing, storing, and interacting with their favourite recipes. The program provides an intuitive user interface for basic recipe administration functions, such as creating, editing, deleting, and uploading images, and was developed with the Django web framework. Through the usage of Django's integrated User model, the platform integrates secure user authentication to provide customised access Every user can create an account, log in, and oversee their own recipe library. The system makes sure that only users who have been verified can carry out specific tasks. The application makes use of the robust Model-View-Template (MVT) architecture of Django. The model specifies how recipe data, such as name, description, image, and related user, should be stored. Business logic, such as processing forms, controlling database queries, and maintaining user sessions, is handled by the Views. In order to provide a contemporary and responsive design, Bootstrap is used by the Templates to define the front-end. A login and registration system, user-friendly forms for contributing recipes with image support, dynamic content management with edit and delete capabilities, and a search engine for rapidly sorting recipes by name are some of the main features. In addition to being a useful recipe management tool

## II. Literature Review

Over time, web-based recipe management systems have undergone substantial development, moving from static webpages and basic desktop apps to dynamic, interactive platforms. Recipe sharing and updating were made more difficult by earlier systems' lack of essential features like user identification, dynamic data processing, and scalable storage. These systems frequently relied on manual maintenance and file-based storage. MVC structures and improved tools for managing user interfaces and backend logic became available to developers with the emergence of contemporary web frameworks like Laravel (PHP), Ruby on Rails, and ASP.NET. MVC designs and improved tools for backend logic and user interface management become available to developers. Many of these frameworks, however, lacked some integrated functionality and required a significant amount of setup. The built-in admin interface, powerful user authentication system, ORM for database interaction, and solid security features against common web threats like XSS, CSRF, and SQL injection make Django, a high-level Python web framework, stand out in this market. Despite offering extensive features like media integration, user interactions, and personalised feeds, large- scale platforms like All Recipes, Tasty, and Yummly are frequently excessively centralised and complicated for users. looking for simplicity and customisation. This project fills the gap by providing a customised, minimalist, and user-centric web environment that may be enhanced in the future. It suggests a lightweight, Django-based recipe management system that concentrates on essential CRUD operations and user account management. The development of the recipe management space is exemplified by a number of commercial programs, including Yummly, Tasty, and All Recipes. User-generated content, multimedia uploads, ratings, comments, social sharing, and recommendation systems are all supported by these feature-rich platforms. Additionally, they are designed to withstand large traffic loads and are backed by sophisticated backend systems that allow for community participation and content personalisation. These systems are powerful and feature-rich, but they are also centralised and monolithic, meaning that individual users who might want to build and maintain their own private or specialised recipe collections have little to no customisation options. By providing a lightweight, Django-based recipe management system that emphasises user liberty, simplicity, and extensibility, this project seeks to close that gap. In addition to basic user management capabilities like registration, login, and user-specific recipe views, the main focus is on implementing CRUD (Create, Read, Update, Delete) activities for recipes. Individual users or local communities can deploy and customise the system's efficient, secure, and maintainable solution by utilising Django's built-in capabilities. In subsequent revisions, more sophisticated features—like image uploads, tagging, search, and sharing—can be added to this basic design to better meet the changing needs of end users.

## III. Methodology

The Agile Development methodology, which stresses iterative development, continuous testing, and continuous improvement throughout the development lifecycle, is used in this project. Agile makes guarantee that work is flexible and responsive to input by dividing it into manageable phases and providing tiny, functional increments. This method works especially well for web development projects like this one, where functional modifications and user experience are crucial. This project's methodology is divided into a number of separate but related stages. Understanding the system's fundamental requirements is the main goal of the first step, requirement analysis. This entails figuring out the essential features, including user registration, login, recipe creation, and editing, as well as defining the target users, who are people who wish to manage and arrange their own recipe collections. The minimum viable product (MVP) is established in light of this research to guarantee that only the most crucial elements are included in the first iteration, allowing for iterative improvement in response to user feedback. Scalability and maintainability are taken into consideration when planning the application's architecture during the System Design process. To guarantee that each user may safely manage their own recipe data, the database schema is constructed using Django's Object- Relational Mapping (ORM), which creates a Recipe model that is directly related to Django's built-in User model. For responsive and aesthetically pleasing frontend design.

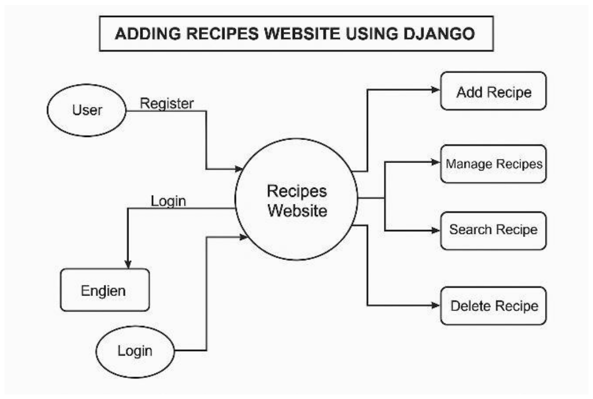


Fig1. Data Flow Diagram

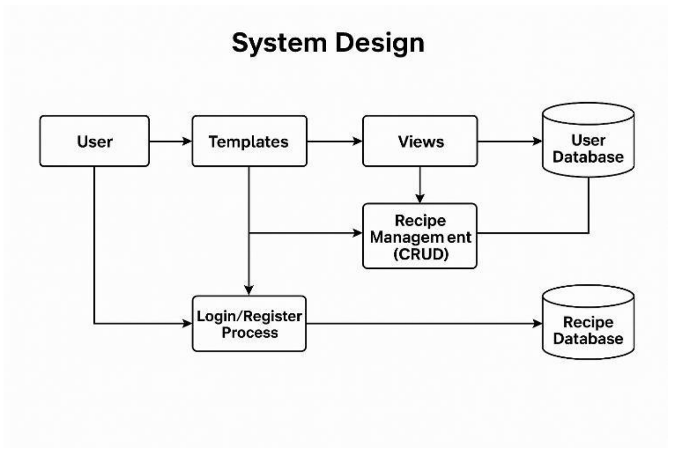


**Fig2. System Design**

Django's templating engine is used in conjunction with Bootstrap to create templates like login.html, register.html, and recipes.html. In order to guarantee that every user action has a matching backend procedure and user interface element, URL routing and view logic are also mapped out during this stage. The actual coding and feature development take place during the implementation phase. Django's powerful built-in system is used to set up user authentication, utilising the authenticate() and login() functions to provide safe access. Users can add new recipes, view existing ones, update entries, and remove recipes they no longer need thanks to the implementation of CRUD (Create, Read, Update, Delete) capability for recipe management. With MEDIA_URL and MEDIA_ROOT set up in Django settings to efficiently manage media content, file upload functionality is incorporated to allow users to attach photographs to their recipes. Each action is handled by a view, which is linked to the appropriate HTML template using Django's URL dispatcher. The testing and debugging stage follows, which guarantees the application's stability and dependability. To make sure they operate as intended, unit tests are run on separate parts and features. Particular attention is paid to user access rights, picture uploads, and form submissions. Users can receive real-time feedback on their actions, such as successful login, invalid input, or errors during form submission, by using Django's messages framework. Iterative changes are performed based on test findings and usability evaluations, and bugs found during testing are swiftly fixed. Finally, the Deployment and Maintenance phase prepares the project for real-world use. Static and media files are properly configured to work in development and production environments. The project structure is reviewed to ensure clean code and scalability. Although the initial version is kept simple, the system is designed with future enhancements in mind. Potential features such as recipe ratings, user profiles, comment sections, recipe categorization, and advanced search capabilities are documented as part of the long-term maintenance and upgrade plan. This modular and flexible approach ensures that the project can evolve over time based on user needs and technological advancements.

The goal of the implementation phase is to use the Django framework to turn the system design into a working web application. To keep data models, application logic, and user interface rendering distinct, Django's Model-View-Template (MVT) architecture was used. Throughout the development process, this architectural strategy encourages modularity, scalability.

There are two main Django applications that make up the project. Static pages like the home, about, and contact sections are managed by the home app, which also offers navigation and basic information. The veggie app, which manages recipe administration and user identification, is where the essential functionality is found. Four essential elements make up Django's structure within each app: frameworks to specify the database structure, Templates are used to create dynamic HTML content, views are used to process user requests and apply business logic, and URL settings are used to direct incoming HTTP requests to the relevant view functions. Django's Object-Relational Mapper (ORM) is used to manage database operations. It allows for smooth backend interaction without the need for raw SQL queries. A foreign key connects the main model, Recipe, to Django's integrated User model, enabling users to control their own recipes. The model supports media uploads using Django's Image Field and contains fields for the recipe name, description, and image. The MEDIA_ROOT and MEDIA_URL parameters in settings.py and uploaded photos are arranged under a certain folder.

Templates are used to create dynamic HTML content, views are used to process user requests and apply business logic, and URL settings are used to direct incoming HTTP requests to the relevant view.

Django's Object-Relational Mapper (ORM) is used to manage database operations. It allows for smooth backend interaction without the need for raw SQL queries. A foreign key connects the main model, Recipe, to Django's integrated User model, enabling users to control their own recipes. The model supports media uploads using Django's Image Field and contains fields for the recipe name, description, and image. The MEDIA_ROOT and MEDIA_URL parameters in settings.py are used to configure the file handling system and uploaded photos are arranged under a certain folder. Templates are used to create dynamic HTML content, views are used to process user requests and apply business logic, and URL settings are used to direct incoming HTTP requests to the relevant view functions.

Django's Object-Relational Mapper (ORM) is used to manage database operations. It allows for smooth backend interaction without the need for raw SQL queries. A foreign key connects the main model, Recipe, to Django's integrated User model, enabling users to control their own recipes. The model supports media uploads using Django's Image Field and contains fields for the recipe name, description, and image. The MEDIA_ROOT and MEDIA_URL parameters in settings.py are used to configure the file handling system and uploaded photos are arranged under a certain folder. Templates are used to create dynamic HTML content, views are used to process user requests and apply business logic, and URL settings are used to direct incoming HTTP requests to the relevant view functions.

Django's Object-Relational Mapper (ORM) is used to manage database operations. It allows for smooth backend interaction without the need for raw SQL queries. A foreign key connects the main model, Recipe, to Django's integrated User model, enabling users to control their own recipes. The model supports media uploads using Django's Image Field and contains fields for the recipe name, description, and image. The MEDIA_ROOT and MEDIA_URL parameters in settings.py are used to configure the file handling system and uploaded photos are arranged under a certain folder.

## IV. Results and Discussions

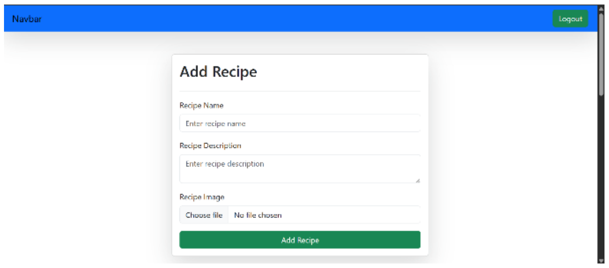
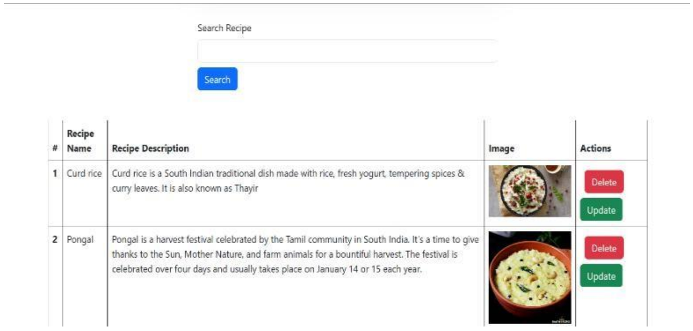


Fig3 . Adding recipe page

The implementation of the Recipe Website using Django successfully achieved its intended functionality. Templates are used to create dynamic HTML content, views are used to process user requests and apply business logic, and URL settings are used to direct incoming HTTP requests.

Django's Object-Relational Mapper (ORM) is used to manage database operations. It allows for smooth backend interaction without the need for raw SQL queries. A foreign key connects the main model, Recipe, to Django's integrated User model, enabling users to control their own recipes. The model supports media uploads using Django's Image Field and contains fields for the recipe name, description, and image. The MEDIA_ROOT and MEDIA_URL parameters in settings.py are used to configure the file handling system and uploaded photos are arranged under a certain folder.



| # | Recipe Name | Recipe Description | Image | Actions |
|---|---|---|---|---|
| 1 | Curd rice | Curd rice is a South Indian traditional dish made with rice, fresh yogurt, tempering spices & curry leaves. It is also known as Thayir | | Delete Update |
| 2 | Pongal | Pongal is a harvest festival celebrated by the Tamil community in South India. It's a time to give thanks to the Sun, Mother Nature, and farm animals for a bountiful harvest. The festival is celebrated over four days and usually takes place on January 14 or 15 each year. | | Delete Update |

Fig4 . list of recipes

Templates are used to create dynamic HTML content, views are used to process user requests and apply business logic, and URL settings are used to direct incoming HTTP requests to the relevant view functions.

Django's Object-Relational Mapper (ORM) is used to manage database operations. It allows for smooth backend interaction without the need for raw SQL queries. A foreign key connects the main model, Recipe, to Django's integrated User model, enabling users to control their own recipes. The model supports media uploads using Django's Image Field and contains fields for the recipe name, description, and image. The MEDIA_ROOT and MEDIA_URL parameters in settings.py are used to configure the file handling system and uploaded photos are arranged under a certain folder. The development of the Recipe Website project provided valuable insights into building a full-stack web application using the Django framework. The implementation successfully demonstrated how Django's integrated components—such as its built-in authentication system, ORM, templating engine, and security features—contribute to rapid and efficient web development. These tools greatly simplified the creation of common functionalities like user login, form handling, file uploads, and CRUD operations.

One of the significant strengths of the project lies in its adherence to modular design principles. By separating concerns using Django's Model- View-Template architecture, the application ensures clarity and maintainability of code. The use of Django's ORM eliminated the need for raw SQL, simplifying database interactions and reducing the likelihood of errors such as SQL injection.

## V. Conclusion and Future work

Templates are used to create dynamic HTML content, views are used to process user requests and apply business logic, and URL settings are used to direct incoming HTTP requests

Django's Object-Relational Mapper (ORM) is used to manage database operations. It allows for smooth backend interaction without the need for raw SQL queries. A foreign key connects the main model, Recipe, to Django's integrated User model, enabling users to control their own recipes. The model supports media uploads using Django's Image Field and contains fields for the recipe name, description, and image. The MEDIA_ROOT and MEDIA_URL parameters in settings.py are used to configure the file handling system and uploaded photos are arranged under a certain folder. Templates are used to create dynamic HTML content, views are used to process user requests and apply business logic, and URL settings are used to direct incoming HTTP requests to the relevant view functions.

Django's Object-Relational Mapper (ORM) is used to manage database operations. It allows for smooth backend interaction without the need for raw SQL queries. A foreign key connects the main model, Recipe, to Django's integrated User model, enabling users to control their own recipes. The model supports media uploads using Django's Image Field and contains fields for the recipe name, description, and image. The MEDIA_ROOT and MEDIA_URL parameters in settings.py are used to configure the file handling system and uploaded photos are arranged under a certain folder.

Templates are used to create dynamic HTML content, views are used to process user requests and apply business logic, and URL settings are used to direct incoming HTTP requests

Django's Object-Relational Mapper (ORM) is used to manage database operations. It allows for smooth backend interaction without the need for raw SQL queries. A foreign key connects the main model, Recipe, to Django's integrated User model, enabling users to control their own recipes. The model supports media uploads using Django's Image Field and contains fields for the recipe name, description, and image. The MEDIA_ROOT and MEDIA_URL parameters in settings.py are used to configure the file handling system and uploaded photos are arranged under a certain folder.

## VI. References

[1] The Definitive Guide to Django: Web Development Done Right, by A. Holovaty and J. Kaplan-Moss, 2nd ed., Berkeley, CA, USA: Apress, 2009.

[2] In February 2013, K. Soni published "A Survey of Web Application Frameworks" in the International Journal of Computer Applications, volume 64, issue 15, pages 1–5.

[3] HTML and CSS: Design and Build Websites by J. Duckett. Wiley, Indianapolis, IN, USA, 2011.

[4] M. Grinberg, Flask Web Development: Create Web Apps with Python, 2nd ed., O'Reilly Media, Sebastopol, CA, USA, 2018.

[5] M. D. Soper, Starting Django: Using Python to Develop and Deploy Web Applications. Apress, Berkeley, CA, USA, 2021.

[6] "Bootstrap 5 Documentation," [Online], Bootstrap Team. https://getbootstrap.com/docs/5.0/ is accessible. [retrieved: 15 April 2025].

[7] "Django Documentation," Django Software Foundation, [Online]. https://docs.djangoproject.com/en/stable/ is accessible. [retrieved: 15 April 2025].

[8] Fielding and Taylor, R. "Principled Design of the Modern Web Architecture," ACM Transactions on Internet Technology, vol. 2, no. 2, May 2002, pp. 115–150.

[9] C. A. Klein, "Comprehending RESTful API Architecture in Contemporary Web Development," IEEE Internet Computing, no. 2, March–April 2019, pp. 84–89.

[10] A. Sharma and M. R. Chauhan, "A Method for Developing Secure Web Applications with the Django Framework," International Journal of Innovative Technology and Exploring Engineering, March 2019, vol. 8, no. 6, pp. 2456–2460.