DESIGN HOME PLANNER ARCHITECT

Project Report

Submitted to the

T.D.M.N.S College of Arts and Science

(Affiliated to the Manonmaniam Sundaranar University, Tirunelveli)

In partial fulfilment of requirements for

The award of the degree of

"Bachelor of Information Technology"

MARIAPPAN .K - 20221291516226

ESAKKIAPPA .E - 20221291516206

SELVAMANI .L - 20221291516242

VELMURUGAN .A - 20221291516246

Supervisor and Guide

M. LENCY MCA.,M.Phill.,

HOD OF Department of Information Technology

T.D.M.N.S College of Arts and Science T.Kallikulam

DEPARTMENT OF INFORMATION TECHNOLOGY

T.D.M.N.S COLLEGE

T.KALLIKULAM-627 113

2024-2025

T.D.M.N.S COLLEGE

T. KALLIKULAM – 627 113

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATEE

Certified that this project **"Online Student Registration System"** Bonafide work done by **MARIAPPAN .K (20221291516226), ESAKKIAPPA .E (20221291516206), SELVAMANI .L (20221291516242), VELMURUGAN .A (20221291516246)** during the academic year 2024-2025 who carried out the project under our supervision, certified further that to the best of our knowledge the work reported here in does not from part of any other project work on the basic of which a degree or award was conferred on an easier occasion on this or any other candidate.

Place: T.Kallikulam
Date:

Head of the Department
Mrs. M. LENCY MCA.,M,Phill.,

project Guide
A. ANTO CELINE GOLDA MCA.,M.Phill.,

External Examiner

1.

2.

ABSTRACT

ABSTRACT

The "**DESIGN HOME PLANNER ARCHITECT** " is an innovative software application designed to assist users in planning, organizing, and visualizing their home spaces effectively. This platform offers a comprehensive set of tools for homeowners, interior designers, and architects to create detailed layouts, manage resources, and bring design ideas to life.

The system enables users to create customizable floor plans, experiment with furniture arrangements, and explore various design styles through an intuitive drag-and- drop interface.

The **"DESIGN HOME PLANNER ARCHITECT "** also incorporates a resource management module to track budgets, materials, and project timelines. This feature ensures that users can align their design aspirations with practical constraints, reducing the likelihood of overspending or delays.

Collaboration tools are built into the system, allowing multiple users to work on the same project simultaneously. Designers and clients can share ideas, provide feedback, and finalize plans efficiently, fostering clear communication throughout the process.

Security is prioritized through user authentication and encrypted data storage, ensuring that personal and project information remains protected. Designed to be scalable and accessible, the application is compatible with desktop and mobile devices, catering to diverse user needs.

By combining powerful design tools, resource management capabilities, and collaboration features, the "DESIGN HOME PLANNER ARCHITECT " simplifies home design projects and empowers users to transform their ideas into reality efficiently and creatively.

ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

Apartac from the efforts of myself, the success of any project depends largely on the encouragement and guidelines of many others. I take this opportunity to express my gratitude to the people who have been instrumental in the successful completion of this project. I would like to show my greatest appreciation to Golda Mam. I can't say thank you enough for his tremendous support and help.I feel motivated and encouraged every.Without his encouragement and guidance this project would not have materialized.

The Guidance and support received from all the members who contributed and who are contributing to this project, was vital for the success of the project, I am grateful for their constant support and help.

NAME

MARIAPPAN .K

ESAKKIAPPA .E

SELVAMANI .L

VELMURUGAN .M

SOFTWARE REQUIRED SPECIFICATION

SOFTWARE REQUIRED SPEFICATION

INTRODUCTION

CHAPTER 1

INTRODUCTION

1.1 ABOUT THE PROJECT

The "**DESIGN HOME PLANNER ARCHITECT** " is ideal for homeowners looking to redesign their spaces, interior designers planning projects for clients, and architects visualizing layouts. It simplifies the design process, enhances collaboration, and provides a comprehensive toolkit for turning ideas into actionable plans.

This system is not only a creative solution but also a practical one, ensuring that users can plan their dream homes efficiently while staying within budget and timeline constraints.

The system includes a robust resource management module to help users track budgets, materials, and project timelines. This feature ensures that designs are feasible and align with financial and time constraints.

Designed for teamwork, the platform allows multiple users to work on the same project. Designers and clients can share ideas, annotate plans, and communicate feedback directly within the application. Virtual designs minimize waste by allowing users to plan precisely, reducing the likelihood of unnecessary purchases or material use. Encourages sustainable design choices by integrating eco-friendly material suggestions. The intuitive interface ensures that even users without technical or design expertise can create professional-looking plans Step-by-step guides and tutorials help users navigate the platform effortlessly.

Users can save and revisit their designs to adapt to future needs, such as renovations or expansions. Scalable layouts accommodate changes in lifestyle, such as growing families or new functionality requirements.

1.2 SYSTEM SECIFICATION

The Software Requirement Specification is a technical specification of requirements for the software product. The goal of software requirements definition is to completely specify the technical requirements for the software products in a concise and unambiguous manner.

1.2.1 HARDWARE SPECIFICATION

Machine	: Intel Core i5 or AMD Ryzen 5 (Quad-core or higher).
Clock Speed	: 500 MHz or higher
System Memory	: 8 GB (16 GB recommended).
Hard Disk Space	: 256 GB SSD (512 GB recommended).

1.2.2 SOFTWARE SPECIFICATION

Operating System	: Windows 10/11, macOS, or Linux (Ubuntu preferred for web development).
IDE/Editor	: Python-compatible IDE.
Framework	: Django (Latest Version).
Programming Languages	: Python – Backend development.

HTML, CSS, JavaScript – Frontend development.

1.2.3 SOFTWARE DESCRIPTION

Python

Python is a high-level, interpreted programming language known for its simplicity and versatility. It is widely used for web development, data analysis, artificial intelligence, and many other fields. Python's key features include:

1. **Ease of Learning and Use**

Python has a clean and readable syntax, making it an excellent choice for beginners and experienced developers.

2. **Extensive Libraries and Frameworks**

Python provides a vast ecosystem of libraries, such as NumPy for scientific computing, Pandas for data analysis, and Flask/Django for web development.

3. **Cross-Platform Support**

Python runs on various operating systems, including Windows, macOS, and Linux, ensuring flexibility in development and deployment.

4. **Dynamic Typing**

Python supports dynamic typing, allowing developers to write less boilerplate code while focusing on functionality.

5. **Community Support**

Python has a large and active community that contributes to its continuous improvement, providing abundant resources, tutorials, and libraries.

DJANGO

Django is a high-level, open-source web framework built using Python. It is designed to enable rapid development and clean, pragmatic design. Django’s primary goal is to simplify web development by providing reusable components and tools. Key features include:

1. **MVC Architecture**

Django follows the Model-View-Controller (MVC) pattern, allowing separation of concerns and modular code structure.

2. Batteries Included

Django comes with pre-built features such as an authentication system, admin interface, ORM for database management, and URL routing, reducing the need for third-party tools.

3. Security

Django provides robust security features, including CSRF protection, SQL injection prevention, XSS protection, and secure password storage.

4. Scalability

Django is scalable and can handle applications ranging from small-scale websites to enterprise-level systems.

5. REST Framework Integration

Django seamlessly integrates with the Django REST Framework (DRF) for building APIs, making it suitable for modern web applications and mobile app backends.

6. Community and Documentation

Django has excellent documentation and a strong community, ensuring developers have ample support during development.

Together, Python and Django form a powerful duo for building dynamic, secure, and scalable web applications efficiently.

Python Django

Python-based web framework Django allows you to create efficient web applications quickly. It is also called batteries included web framework Django because It provides built-in features for everything including Django Admin Interface, default database – SQLlite3, etc.

When you're building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc. Django Python gives you ready-made components to use for rapid development. There any many more benefits of using the Django framework. Let's look at some other reasons why you should learn Python Frameworks in Django.

Why Use Django Framework?

- Excellent documentation and high scalability.
- Used by Top MNCs and Companies, such as Instagram, Disqus, Spotify, Youtube, Bitbucket, Dropbox, etc. and the list is never-ending.
- Web framework Django is easy to learn, rapid development, and Batteries fully included.
- The last but not least reason to learn Django in Python, It has a huge library and features such as Web Scraping, Machine Learning, Image Processing, Scientific Computing, etc. One can integrate all this with web applications and do lots and lots of advanced stuff.

Pre requisites to Learn Django

Web framework Django based on Python. You have good knowledge about Python. Some other concepts you should be familiar with are:

- Understanding of Syntax of Python .
- Understanding of importing and exporting modules is required in the project development phase.
- Understanding Python path concepts t o access the data, images or any kind of data.
- Knowledge of Object Oriented concepts as it reduces the code repetition with classes and objects.
- Knowledge about HTML , CSS , JavaScript are very important. As they are the building block of Web development .
- Knowledge about Data Structures like Tuple and List are important.

Getting Started with Django

In this getting started with Django section, you will learn how to get up and running with Django, a powerful web framework for building dynamic websites and applications. Django makes it easy to organize your project into different apps, each handling specific functionality. You'll begin by creating a project, which will automatically set up everything you need, from a folder structure to basic settings.

Django Views

In Django views are the backbone of handling user requests and rendering responses. There are two primary paradigms for implementing views: Function Based Views (FBVs) and Class Based Views (CBVs). Function Based Views offer simplicity and directness, allowing developers to define views as Python functions. Within this paradigm, common functionalities like creating, listing, displaying details, updating, and deleting objects are implemented as separate functions.

While both paradigms have their merits, the choice between FBVs and CBVs ultimately depends on factors such as project requirements, development preferences, and scalability concerns.

- Function Based Views
- Create View
- List View
- Detail View
- Update View
- Delete View
- Class Based Generic Views Django
- Createview
- ListView
- DetailView
- UpdateView
- DeleteView
- FormView
- Class Based vs Function Based Views

Django URLs

In Django URL patterns serve as a crucial mechanism for directing incoming requests to the appropriate views within your web application. With the flexibility of regular expressions, Django's URL dispatcher allows you to define patterns that match specific URL patterns and route them to corresponding views. When dealing with GET parameters passed through URLs in Django, accessing these parameters within views is straightforward, enabling efficient handling of user inputs and customization of responses.

Django URL patterns

Get parameters passed by urls in Django

- URL Validator in Django
- URL Shortener with Django
- Django URLResolver error
- Django Templates

In Django, URLs play a crucial role in navigating through different views and templates within your web application. When working with Django templates, several key concepts enhance the flexibility and functionality of your frontend. Template filters allow you to manipulate variables displayed in your templates, enabling transformations like date formatting or string manipulation.

- Template Filters
- Template Tags
- variables
- Boolean Operators
- for loop
- if – Django Templates
- Template Inheritance

Django Models

Django Models serve as the backbone of database operations, facilitating seamless management of data. This guide delves into various aspects of Django Models, starting from the fundamental operations of inserting, updating, and deleting data using the Object-Relational Mapping (ORM) provided by Django.

Here, You'll learn how to create a basic app model, initialize migrations, and execute them to synchronize your database schema. Moreover, we delve into built-in field validations, ensuring data integrity and consistency, while also delving into the customization of these validations to suit specific application requirements.

- ORM – Inserting, Updating & Deleting Data
- Basic App Model – Makemigrations and Migrate
- model data types and fields list
- Add the slug field inside Django Model

- Intermediate fields in Django
- Uploading images in Django
- Render Model in Django Admin Interface
- Change Object Display Name using __str__function – Django Models
- Built-in Field Validations – Django Models Custom Field Validations in Django

Django Forms

To start, you can create a form using Django Forms by defining a class that inherits from Django's forms.Form class. Within this class, you can specify the fields you want to include in your form using various field types provided by Django, such as CharField, IntegerField, EmailField, etc. Once you've defined your form, you can render HTML forms in Django using both GET and POST methods. Django's built-in template tags and filters make it easy to render forms in your HTML templates while ensuring security and CSRF protection.

Django Forms offer a wide range of field types to cater to different data types and validation requirements. Additionally, you can customize the appearance and behavior of form fields by using form field custom widgets, allowing you to enhance user experience and tailor forms to your specific needs.

Django architecture

Django is based on MVT (Model-View-Template) architecture. MVT is a software design pattern for developing a web application. MVT Structure has the following three parts – Model: Model is going to act as the interface of your data. It is responsible for maintaining data. It is the logical data structure behind the entire application and is represented by a database (generally relational databases such as MySQL, Postgres). View: A Django View is a Python function or class that receives a web request, processes it according to defined logic (often involving interaction with Models), and returns a web response. It acts as the interface between the Model (data) and the Template (presentation), handling the flow of data and determining how to respond to different requests within a web application. Template: A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

CHAPTER 2

SYSTEM STUDY

2.1 EXISTING SYSTEM

In the absence of a robust platform like the **DESIGN HOME PLANNER ARCHITECT** , the current processes for home planning and design are often inefficient, fragmented, and time consuming.

2.1.1 DISADVANTAGES

These systems are characterized by the following limitations:

1. **Manual Design Processes**:
    - Home designs are often created manually using paper sketches or generic design software that lacks specific home planning features.
    - This approach is prone to errors, lacks precision, and consumes significant time.
2. **Limited Visualization**:
    - Existing tools or methods provide basic 2D layouts, with little to no ability to visualize designs in realistic 3D views or augmented reality (AR).
    - Clients and homeowners often struggle to understand how a design will look in reality.
3. **Scattered Resources**:
    - Users must rely on multiple platforms for different aspects of planning, such as budget tracking, material sourcing, and furniture selection.

- This disjointed approach leads to inefficiencies and increases the likelihood of miscommunication.

4. **Lack of Collaboration**:

   - Designers, architects, and homeowners often use emails or other generic tools for sharing feedback and updates, which can be cumbersome and unstructured.

   - Collaborative changes are not reflected in real-time, leading to delays and misunderstandings.

5. **No Integration with E-Commerce**:

   - Users manually browse furniture and materials from various stores, making comparison and purchasing processes time-consuming.

2.2 PROPOSED SYSTEM

The proposed **DESIGN HOME PLANNER ARCHITECT** system is a comprehensive, all-in-one solution designed to address the inefficiencies of the existing system.

2.2.1 FEATURES

It introduces several innovative features and benefits:

1. **Automated Design Tools**:

   - Users can create precise, customizable 2D and 3D layouts with a drag-and-drop interface.

   - Pre-built templates and automated tools simplify the process for both beginners and professionals.

2. **Advanced Visualization**:

    - Realistic 3D renderings and AR functionality enable users to visualize their designs in real-world settings.

    - This improves decision-making and reduces uncertainties about the final output.

3. **Centralized Resource Management**:

    - A unified platform manages all aspects of home planning, including budgeting, material tracking, and timeline scheduling.

    - Integration with online suppliers allows seamless browsing and procurement of materials and furniture.

4. **Real-Time Collaboration**:

    - The system supports multi-user collaboration, allowing designers, architects, and homeowners to work on the same project simultaneously.

    - Feedback and updates are synchronized instantly, enhancing communication and reducing delays.

5. **E-Commerce Integration**:

    - Direct links to furniture and material suppliers enable users to browse, compare, and purchase items without leaving the platform.

6. **Enhanced User Experience**:

    - Intuitive interfaces ensure that users of all skill levels can easily use the system.

    - Security features like encrypted storage and role-based access controls protect sensitive data.

CHAPTER 3

SYSTEM DESIGN AND DEVELOPMENT

3.1 INPUT DESIGN

Objectives of Input Design

- Ensure user-friendly form inputs for quick data entry.
- Implement validation to avoid spam or incomplete submissions.
- Provide real-time error messages for incorrect inputs.

Input Forms

1. Login Form Fields:

Fields:

- Email / Username
- Password
- Login Button

2. UserRegistration Form

Fields:

- Full Name
- Email ID
- Password
- Confirm Password
- Register Button

3. QuestionPosting Form

Fields:

- Question Title
- Description
- Tags (Dropdown Selection)
- Submit Button

4. Answer Submission Form

Fields:

- Answer Content (Text Editor)
- Submit Button

5. Search& Filter Form

Fields:

- Search Query
- Tags / Categories (Dropdown)
- Sort By (Recent / Popular)

3.2 OUTPUT DESIGN

Objectives of Output Design

- Ensure clear and structured presentation of information.
- Provide real-time updates on new questions and answers.
- Implement **pagination** for better readability.

Output Screens

1. **User Dashboard**

   Displays:

   - Welcome Message
   - Total Questions Posted
   - Latest Activity

2. **Question List Page**

   Displays:

   - List of Questions with Titles & Tags
   - Number of Answers
   - Upvote Count

3. **Question Detail Page**

   Displays:

   - Full Question Content

- All Submitted Answers
- Upvote / Downvote Buttons

4. **Admin Dashboard**
   Displays:
   - Total Users, Questions, and Answers
   - List of Reported Questions
   - System Activity Logs

5. **User Profile Page**
   Displays:
   - User Details
   - List of Questions Posted
   - Reputation Score

3.3 DATABASE DESIGN

The most important consideration in designing the database is how information will be used. The main objectives of designing a database are:

DATA INTEGRATION

In a database, information from several files are coordinated, accessed and operated upon as through it is in a single file. Logically, the information are centralized, physically, the data may be located on different devices, connected through data communication facilities.

DATA INTEGRITY

Data integrity means storing all data in one place only and how each application to access it. This approach results in more consistent information, one update being sufficient to achieve a new record status for all applications, which use it. This leads to less data redundancy; data items need not be duplicated; a reduction in the direct access storage requirement.

DATA INDEPENDENCE

Data independence is the insulation of application programs from changing aspects of physical data organization. This objective seeks to allow changes in the content and organization of physical data without reprogramming of applications and to allow modifications to application programs without reorganizing the physical data.

The tables needed for each module were designed and the specification of each and every column was given based on the records and details collected during record specification of the system study.

Table: Users

Column name	Data type	Constraints
Id	INT(PK)	Auto Increment
Name	VARCHAR	Not Null
Email	VARCHAR	Unique,Not Null
Password	VARCHAR	Not Null
Role	ENUM('User','Admin')	Default 'User'

Table: Questions

Column Name	Data Type	Constraints
id	INT (PK)	Auto Incremnet
user_id	INT (FK)	References Users(id)
description	TEXT	Not Null
tags	VARCHAR	Not Null
Created_at	TIMESTAMP	DefaultCURRENT_TIMESTAMP

Table: Answers

Column Name	Data Type	Constraints
id	INT (PK)	Auto Increment
question_id	INT (FK)	Reference Questions(id)
user_id___	INT (FK)	Reference Users(id)
content	TEXT	Not Null
upvotes	INT	Default 0
Created_at	TIMESTAMP	DefaultCURRENT_TIMESTAMP

3.4 SYSTEM DEVELOPMENT

3.4.1 DESCRIPTION OF MODULES

1. **User Authentication Module Description:**
    - Manages user login, logout, and registration.
    - Ensures secure authentication with password hashing.
    - Role-based access control (User/Admin).

Functions:

- register_user(): Handles user signup.
- login_user(): Verifies credentials and starts a session.
- logout_user(): Ends the user session.

2. **User Dashboard Module Description:**
    - Displays user-specific content, including posted questions and answers.
    - Shows statistics like total questions asked, total answers given.

Functions:

- get_user_dashboard(user_id): Fetches user details and activity summary.

- display_user_questions(user_id): Lists all questions posted by the user.
- display_user_answers(user_id): Lists all answers submitted by the user.

3. Question Posting Module Description:

- Allows users to post new questions related to mechanics.
- Supports tagging for better categorization.
- Validates input before storing in the database.

Functions:

- create_question(user_id, title, description, tags): Saves a new question.
- validate_question_input(title, description): Ensures valid input.
- list_recent_questions(): Retrieves latest questions.

4. Answer Submission Module Description:

- Enables users to submit answers to existing questions.
- Supports text formatting and real-time updates.

Functions:

- submit_answer(user_id, question_id, content): Saves an answer.
- fetch_answers(question_id): Retrieves all answers for a question.
- validate_answer(content): Checks for inappropriate content.

5. Search & Filter Module Description:

- Allows users to search for questions based on keywords and tags.
- Provides sorting options such as "Most Answered" and "Newest."

Functions:

- search_questions(keyword): Returns questions matching the keyword.
- filter_questions_by_tag(tag): Fetches questions with a specific tag.
- sort_questions(order): Sorts questions by popularity, date, etc.

6. Admin Panel Module Description:

- Provides admin controls for managing users, questions, and reports.

- Displays overall system statistics.

Functions:

- get_all_users(): Fetches all registered users.
- flag_question(question_id): Marks a question as inappropriate.
- delete_user(user_id): Removes a user from the system.

7. User Profile Module Description:

- Displays user details, reputation, and activity.
- Allows users to update their profile information.

Functions:

- fetch_user_profile(user_id): Retrieves user details.
- update_profile(user_id, new_data): Saves updated user info.
- get_user_reputation(user_id): Returns reputation score.

8. Notifications Module Description:

- Sends alerts for new answers, upvotes, and admin messages.

Functions:

- notify_user(user_id, message): Sends a notification.
- fetch_notifications(user_id): Displays unread notifications.

CHAPTER 4

TESTING AND IMPLEMENTATION

SYSTEM TESTING

After the source code has been completed, documented as related data structures. Completion of the project has to undergo testing and validation where there is subtitle and definite attempt to get errors. The project developer treats lightly, designing and execution of the project test that will demonstrates that the program works rather than uncovering errors, unfortunately errors will be present and if the project developer doesn't found errors, the user will find out.

The project developer is always responsible for testing the individual units i.e. modules of the program. In many cases, developer should conduct the integration testing i.e. the testing step that leads to the construction of the complete program structure. This project has undergone the following testing procedures to ensure its correctness.

- **UNIT TESTING**
- **INTEGRATION TESTING**

UNIT TESTING

In unit testing, user has to test the programs making up the system. For this reason, Unit testing sometimes called as Program testing. The software units in a system are the modules and routines that are assembled and integrated to perform a specific function.

In this project, all modules are tested individually like given all the fields and can be updated for all criteria.

INTEGRATION TESTING

Integration testing is a systematic technique for constructing the program structure. While at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested modules and build a program structure that has been dictated by design

In this integration testing its done using the main module and based on the type of integration testing the subordinate tables and other criteria along with their path, is replaced one at a time with actual modules. In the project, after integrating the all modules like complaint and service update are tested with their integration and that could integrated and manipulated with respect to the to and from in between modules.

SYSTEM IMPLEMENTATION

When the initial design was done for the system, the client was consulted for the acceptance of the design so that further proceedings of the system development can be carried on. After the development of the system a demonstration was given to them about the working of system. The aim of the system illustration was to identify any malfunction of the system.After the management of the system was approved the system implemented in the concern, initially the system was run parallel with existing manual system. The system has been tested with live data and has proved to be error free and user friendly. Implementation is the process of converting a new or revised system design into an operational one when the initial design was done by the system; a demonstration was given to the end user about the working system. This process is uses to verify and identify any logical mess working of the system by feeding various combinations of test data.

After the approval of the system by both end user and management the system was implemented. A product software implementation method is a blueprint to get users and/or organizations running with a specific software product. The method is a set of rules and views to cope with the most common issues that occur when implementing a software product: business alignment from organizational view and acceptance from the human view.

The implementation of product software, as the final link in the deployment chain of software production, is in a financial perspective of a major issue. The Implementation methodology includes four phases - Discovery, System Development, User Acceptance Testing and Production Rollout. It’s easy to be overwhelmed by slick marketing presentations, particularly when the sales force is talking about things that most people don’t completely understand. Showmanship gets in the way of real capabilities. Unless the review team is judging each vendor against the same list of needs, with the

same understanding of the significance of each rating, “likeability” can win over capability.

These implementation phases are designed to provide clients with a seamless transition from an existing electronic or paper-based system to Sigmund while ensuring all aspects of a client’s operations are accounted for by the software. The Sigmund project team, comprised of individuals with clinical, billing and operations experience, is equipped with the skills and tools to manage the entire process from system requirements gathering to deployment. Sigmund provides various levels of support, depending on client needs, including clientside Project Management.

The Discovery Phase is preceded by a Project Kick-off Work-session that includes application demonstrations, completion and review of requirements and configuration questionnaires, identification of key client documentation, as well as consultation on possible process re-engineering needs

.

CHAPTER 5

CONCLUSION

The Design Home Planner Architect system simplifies and enhances the home planning process by eliminating the need for extensive manual drafting and calculations. With minimal user input, the system efficiently generates accurate 2D architectural layouts, making it accessible to users with little to no technical expertise.

By integrating automated blueprint generation, Vastu compliance, and precise engineering-style drawings, the application ensures error-free and efficient home design. The system significantly reduces planning time while improving accuracy in room placements, structural alignment, and space utilization.

The user-friendly interface and AI-driven features allow individuals to create home layouts effortlessly, without requiring prior architectural knowledge. The system has been thoroughly tested to ensure seamless performance, quick processing, and reliable output.

Additionally, the application enables quick modifications and customizations, allowing users to explore different design possibilities with ease. This system serves as an essential tool for homeowners, architects, and designers, streamlining the planning process while ensuring a well-structured and visually appealing home layout.

FUTURE ENHANCEMENTS

Future of this project can be easily updated. To achieve the benefits that expected from the user must understand the overall system and they must be able to carry out their specific tasks effectively. The successful implementation depends upon the right people at the right time.

The project provides a best assistance in the systematic regime. It allows adding up the following facilities in future,

BIBLOGRAPHY

TEXT BOOKS:

1. **"Django for Beginners"** – *William S. Vincent*

2. **"Python and Django Full Stack Web Developer Bootcamp"** – *Jose Portilla (Course-*

3. **"Django 4 By Example"** – *Antonio Mele*

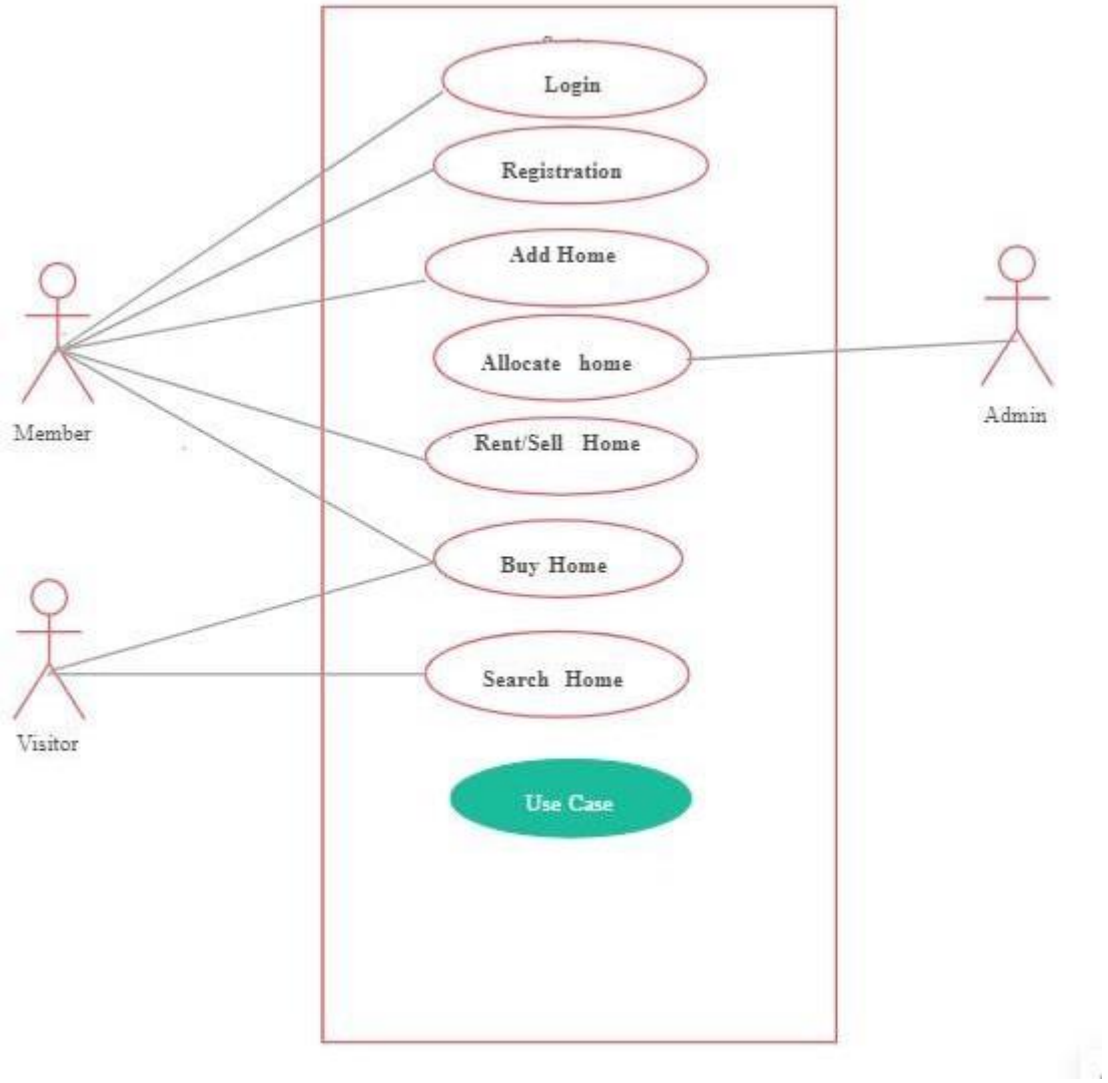
4. **"Two Scoops of Django"** – *Audrey Roy Greenfeld & Daniel Roy Greenfeld*

WEB REFERENCES

- http://msdn.microsoft .com
- http://www.vbdotnetheaven .com
- http://www.codeproject .com
- http://www.programmersheaven .com

APPENDICES

A.USE CASE DIAGRAM



C.SAMPLE CODING

Urls.py

from django.contrib

import admin from django.urls import path, include from accounts import urls as accounts_urls from core import urls as core_urls urlpatterns = [path('admin/', admin.site.urls), path('accounts/', include(accounts_urls)), path('', include(core_urls))]

]

Settings.py

import os

BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))

SECRET_KEY = "7ebtto=fj6$l*sila5o(@%xjo$m0677_1rh#t^55q#f5g5q2#0"

DEBUG = True

ALLOWED_H
OSTS = [] #
Application
definition

INSTALLED_APPS = [
'django.contrib.admin',
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.messages',
'django.contrib.staticfiles',
'accounts',
‘core’

]

MIDDLEWARE = [
'django.middleware.security.SecurityMiddleware',

```
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'stackabundant.urls'

TEMPLATES = [
    {
        'BACKEND':  'django.template.backends.django.DjangoTemplates',                'DIRS':
[os.path.join(BASE_DIR, 'templates'), ],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'stackabundant.wsgi.application'


# Database
# https://docs.djangoproject.com/en/3.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

```
# Password validation
#   https://docs.djangoproject.com/en/3.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
   {
      'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',

}
,
{
      'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
   },
   {
      'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',

}
,
{
      'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
   },
]


# Internationalization
# https://docs.djangoproject.com/en/3.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'
USE_I18N  =  True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
#   https://docs.djangoproject.com/en/3.0/howto/static-files/
```

STATIC_URL = '/static/'

STATICFILES_DIRS = [os.path.join(BASE_DIR, 'static'),]

LOGIN_REDIRECT_URL = 'core:home'

LOGOUT_REDIRECT_URL = 'core:home'

#LOGOUT_REDIRECT_URL = '/'

DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

Models.py

from django.contrib import auth from django.db import models

class User(auth.models.User, auth.models.PermissionsMixin):

def __str__(self):

return f"@{self.username}" **views.py** from django.urls import reverse_lazy from django.views.generic import CreateView from django.contrib.auth import logout from django.shortcuts import redirect from . import forms def logout_view(request):

logout(request) return redirect('/') class SignUp(CreateView):

form_class = forms.UserCreateForm success_url = reverse_lazy("accounts:login") template_name = "accounts/signup.html"

Forms.py

from django.contrib.auth import get_user_model from django.contrib.auth.forms import UserCreationForm class UserCreateForm(UserCreationForm): class Meta: fields = ("username", "email", "password1", "password2") model = get_user_model() def __init__(self, *args, **kwargs):

super().__init__(*args, **kwargs)

self.fields["username"].label = "Display name"

self.fields["email"].label = "Email address"

self.fields['password1'].help_text = None

self.fields['password2'].help_text = None

self.fields['email'].help_text = None

self.fields['username'].help_text = None

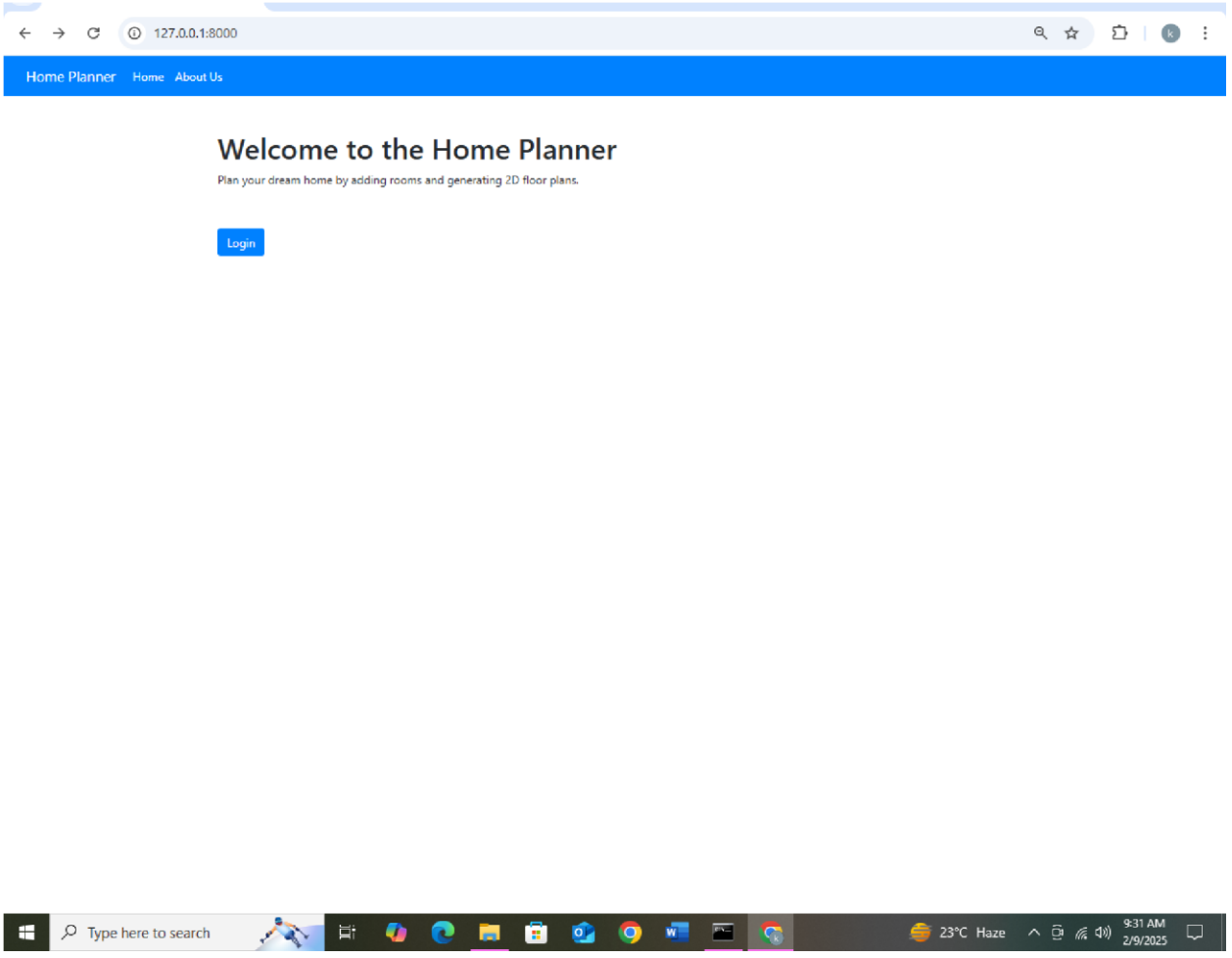
Admin.py

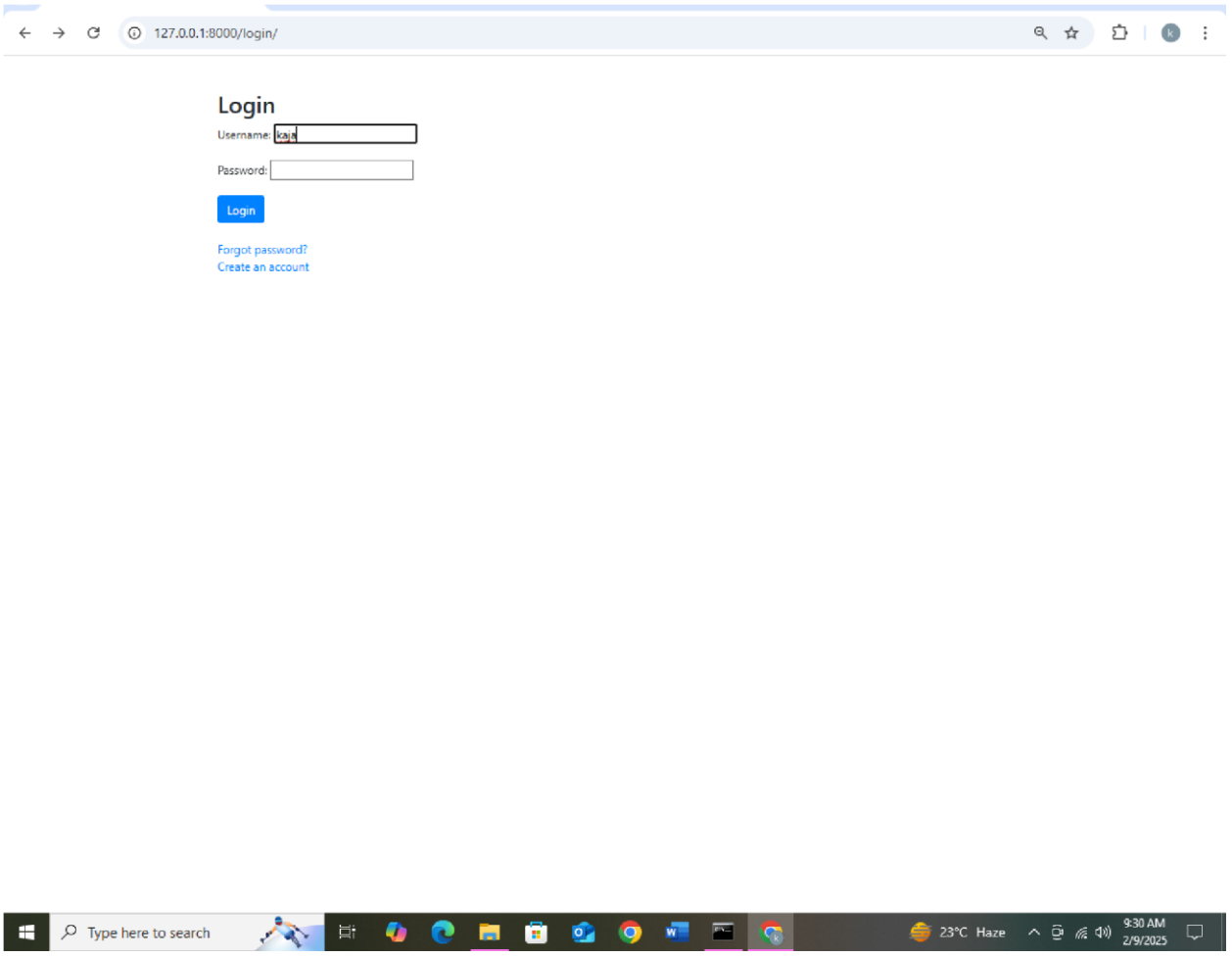
```
from django.contrib import admin

from . models import FloorPlan                    # Register your models here.
admin.site.register(FloorPlan)
```

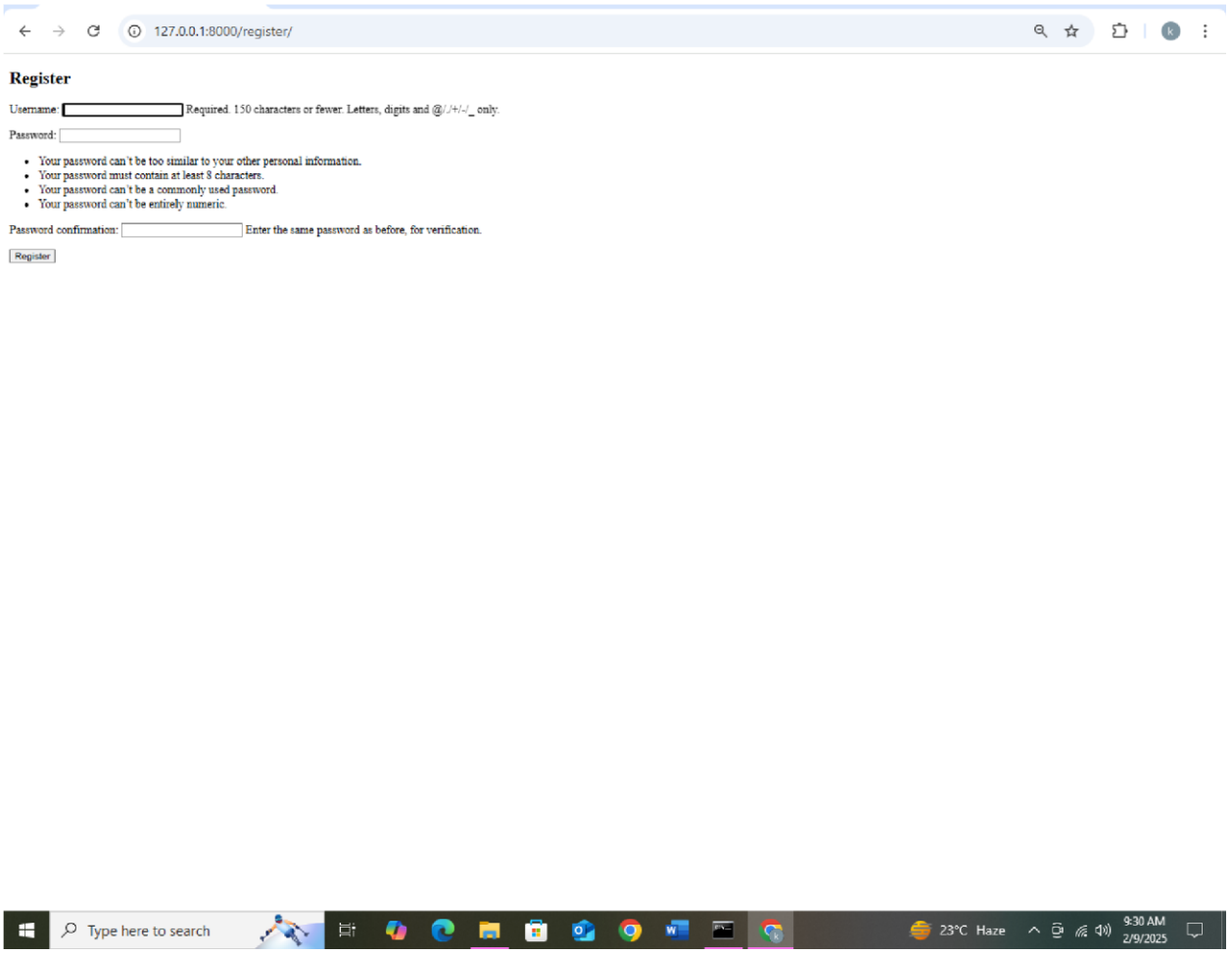
Utils.py

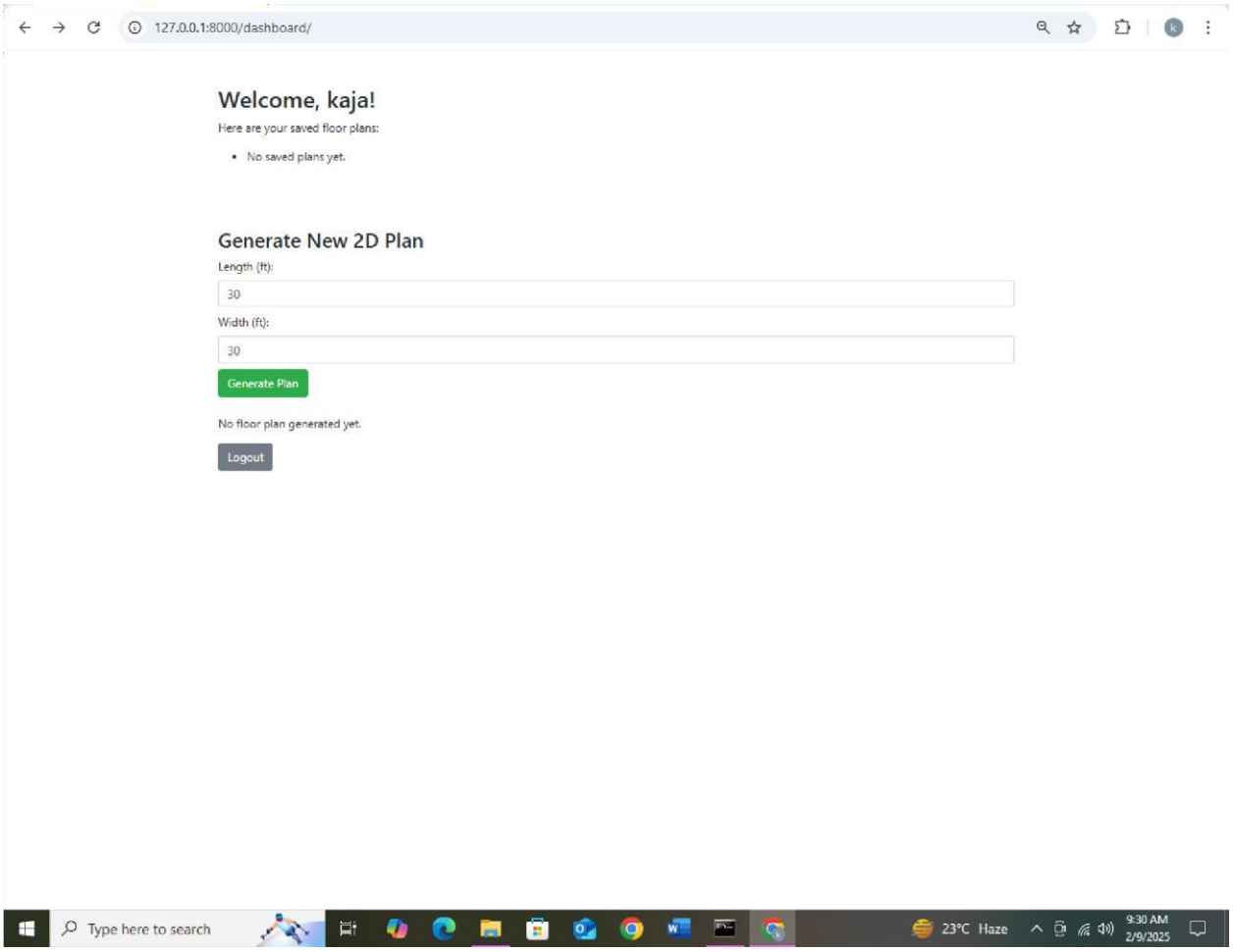
```
def get_building_materials():
    # Example function to get building
materials     return [
        {"name": "Wood", "price": "$10 per sqft"},
        {"name": "Steel", "price": "$20 per sqft"},
        {"name": "Concrete", "price": "$15 per sqft"},
    ]

def generate_2d_design(num_rooms, square_feet, room_type, width, length):
    # Example function to generate a simple 2D design
    design = {
        "num_rooms": num_rooms,
        "square_feet": square_feet,
        "room_type": room_type,
        "dimensions": {"width": width, "length": length},
    }     return design
```

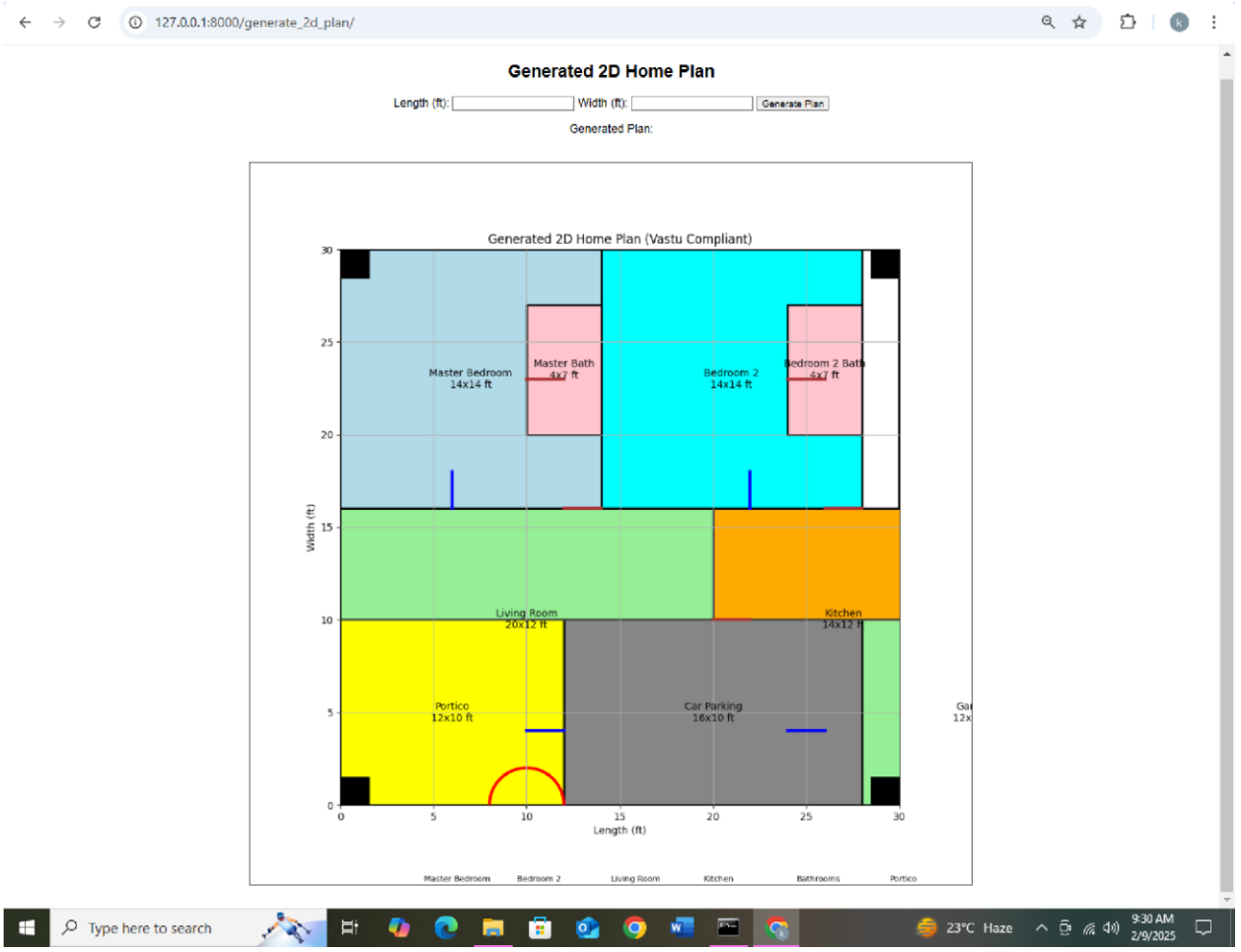
Sample output




127.0.0.1:8000/login/
Login
Username:
Password:
Login
Forgot password?
Create an account
Type here to search
23°C Haze
9:30 AM
2/9/2025


127.0.0.1:8000/register/
Register
Username: Required. 150 characters or fewer. Letters, digits and @/./+/-/_ only.
Password:
Your password can't be too similar to your other personal information.
Your password must contain at least 8 characters.
Your password can't be a commonly used password.
Your password can't be entirely numeric.
Password confirmation: Enter the same password as before, for verification.
Register
Type here to search
23°C Haze
9:30 AM
2/9/2025


127.0.0.1:8000/dashboard/
Welcome, kaja!
Here are your saved floor plans:
No saved plans yet.
Generate New 2D Plan
Length (ft):
30
Width (ft):
30
Generate Plan
No floor plan generated yet.
Logout
Type here to search
23°C Haze
9:30 AM
2/9/2025


127.0.0.1:8000/generate_2d_plan/
Generated 2D Home Plan
Length (ft):
Width (ft):
Generate Plan
Generated Plan:
Generated 2D Home Plan (Vastu Compliant)
Master Bedroom
14x14 ft
Master Bath
4x7 ft
Bedroom 2
14x14 ft
Bedroom 2 Bath
4x7 ft
Living Room
20x12 ft
Kitchen
14x12 ft
Portico
12x10 ft
Car Parking
16x10 ft
Width (ft)
Length (ft)
Master Bedroom
Bedroom 2
Living Room
Kitchen
Bathrooms
Portico
Type here to search
23°C Haze
9:30 AM
2/9/2025