Local Tourist APP USING Django [Go_Native]

Go_Native is more than just an app; it's a platform that brings communities together through shared experiences. Uncover the best-kept secrets of your neighborhood, from quaint cafes to breath-taking viewpoints. Easily add new locations, share stunning photos, and become a local tourism ambassador. Whether you're a seasoned explorer or a curious newcomer, **Go_Native** is your key to unlocking the full potential of your city.

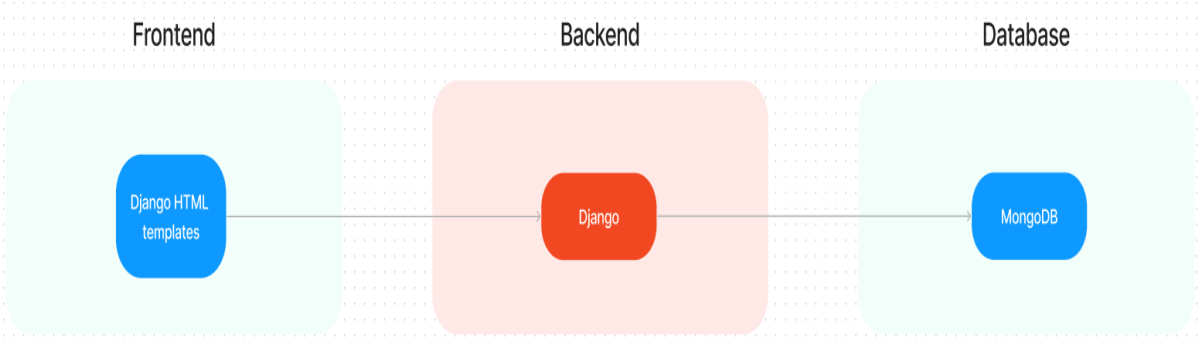
Scenario based use case:

Imagine you're a passionate foodie living in Karad. You've discovered a small, family-owned biryani place tucked away in an unassuming corner of the city. The food is exceptional, but the place is unknown to most. With **Go_Native**, you can easily add this hidden gem to the app, complete with mouth-watering photos of the biryani. You can even share a brief description of the place, its ambiance, and your personal recommendation.

Other food lovers in the city can then discover this hidden treasure through the app, contributing to its growing popularity. By sharing your local find, you've not only helped fellow foodies but also supported a small local business.

This is just one example of how Go_Native can transform the way you experience and share your city.

TECHNICAL ARCHITECTURE



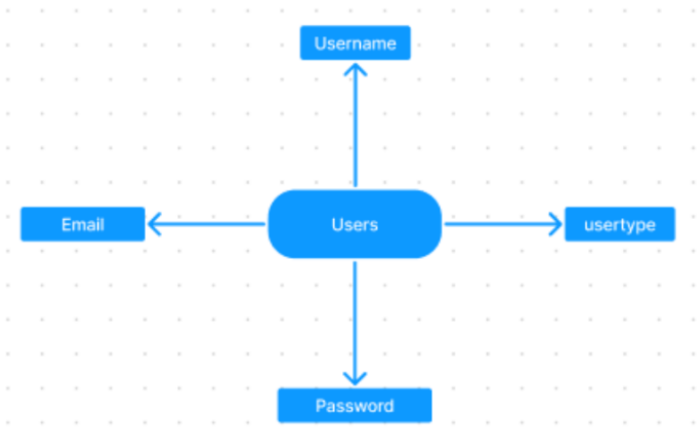
The technical architecture of our Local Tourist app follows a client-server model, where the frontend serves as the client and the backend acts as the server. The frontend utilizes the bootstrap library to establish real-time and better UI experience for any user whether it is admin, doctor and ordinary user working on it.

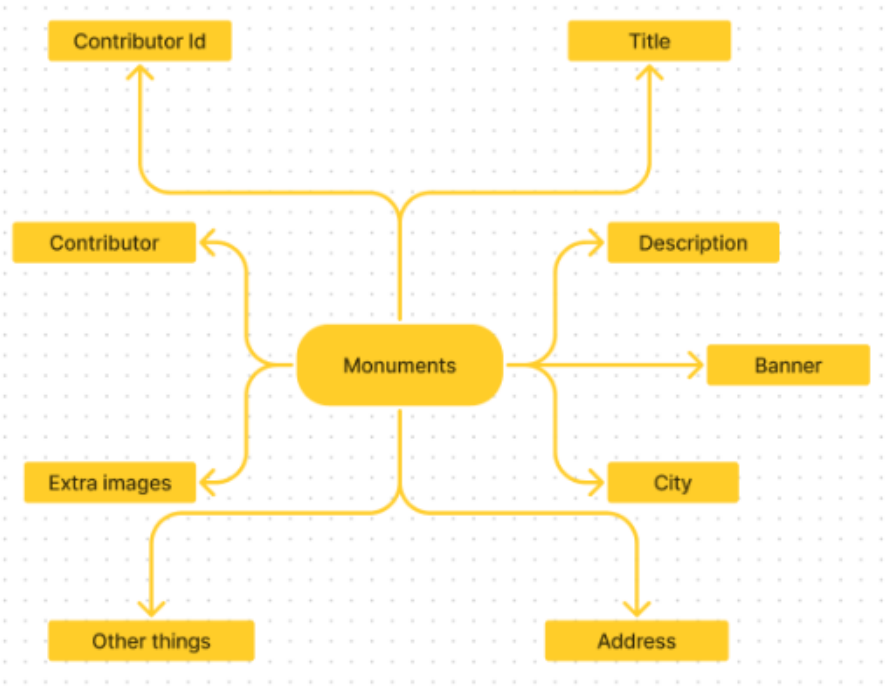
On the backend side, we employ Django framework to handle the server-side logic and communication.

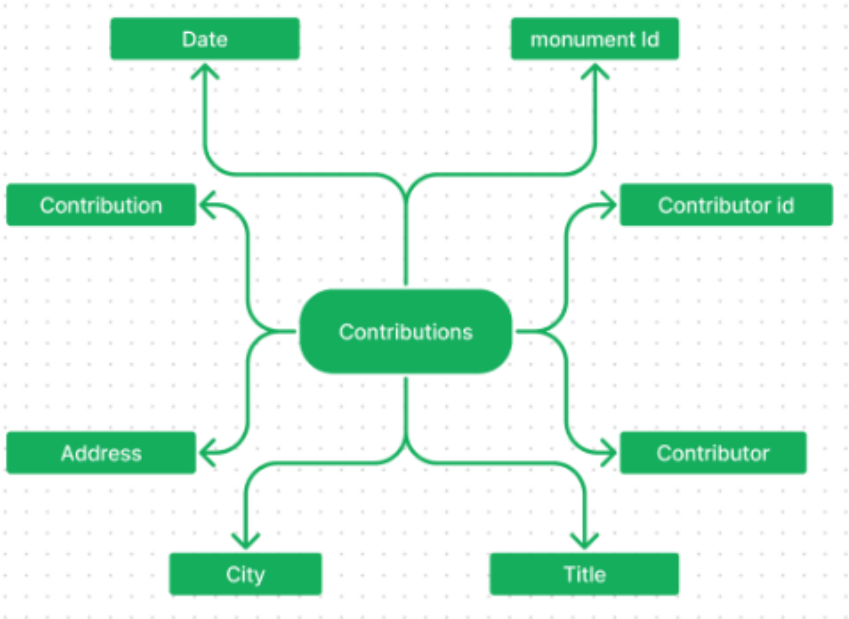
For data storage and retrieval, our backend relies on MongoDB. MongoDB allows for efficient and scalable storage of user data, etc. It ensures reliable and quick access to the necessary information.

Together, the frontend and backend components, along with Django and MongoDB, form a comprehensive technical architecture for our House rent app. This architecture enables real-time communication, efficient data exchange, and seamless integration, ensuring a smooth and immersive booking a property and many more experience for all users.

ER DIAGRAM

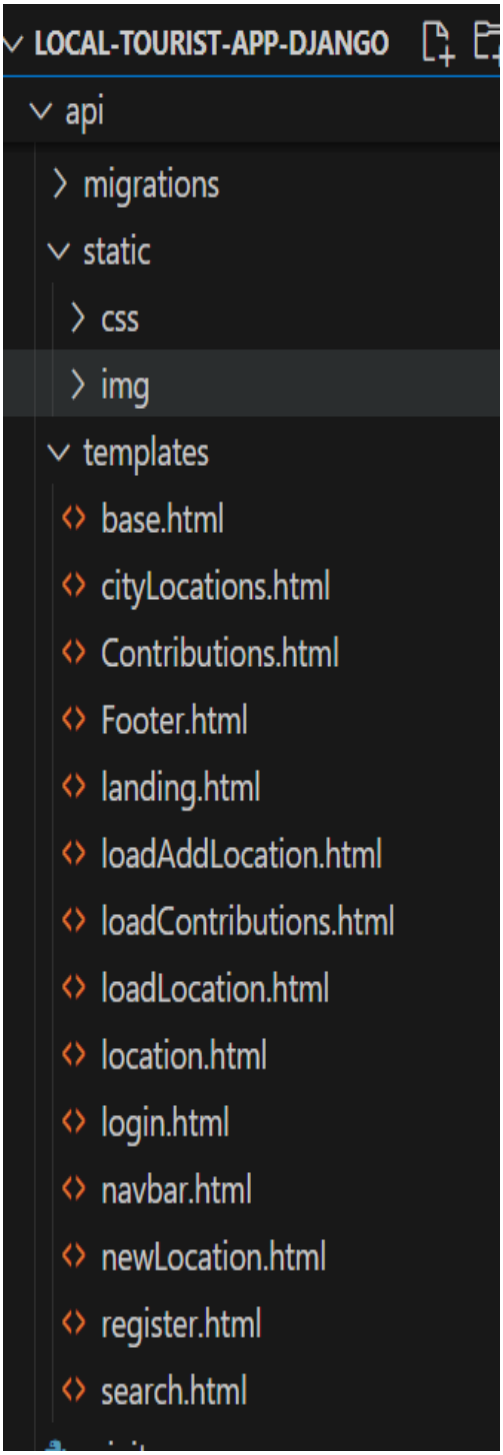


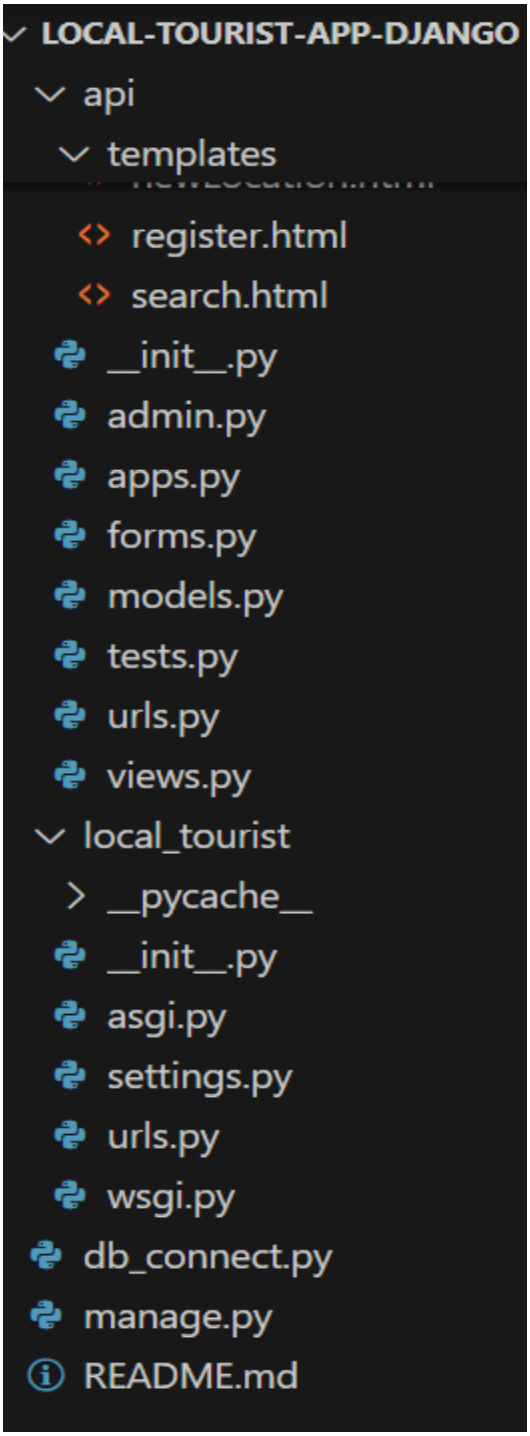




The user schema has basic details such as email and user name along with the password. We use this data for the authentication in the application. The monuments schema contains all the data about the monuments. It has details such as title, description, address, etc., The contributions schema contains the data about the contributions made by users.

PROJECT STRUCTURE





The first image shows the frontend content. It has a well-organized structure with dedicated folders for the API, static files (CSS and images), and templates (admin, and user). The API folder likely contains the code for the application's functionalities. The static folder stores the application's static files, such as Cascading Style Sheets (CSS) used for styling the user interface and images that may be displayed throughout the application. Finally, the templates folder contains the HTML templates that define the

application's layout and user interface. These templates are used to dynamically generate the web pages that users will see.

The second image shows details about the server login. It follows a well-defined structure separating the core application logic from the overall project configuration. The application code resides in the Local Tourist directory, containing models, views, and templates specific to the Local Tourist functionality. The main project directory Local Tourist project-wide settings, URL patterns, and management tools. This separation promotes maintainability and scalability as the project grows.

Features:

Users:

- **Create and Manage Campaigns:** Easily launch fundraising campaigns for any cause, customizing campaign pages with compelling stories, images, and clear donation goals.
- **Donate and Support:** Contribute to campaigns that resonate with you and track your giving history.
- **Stay Informed:** Receive updates and progress reports on supported campaigns to measure impact.

Connect and Share: Engage with other users, share campaigns, and build a community around your cause.

Pre-requisites

Here are the key prerequisites for developing a full-stack application using Django, MongoDB:

Django: Setting Up Your Project

Django is a high-level Python framework for building web applications. It streamlines the development process by providing a robust structure and handling common web development tasks.

Prerequisites:

- Python (version 3.6 or later recommended)
- pip (Python package installer)

Installation:

- Verify Python Installation: Open a terminal or command prompt and type python --version. If Python is installed, you'll see the version number.
- Install pip: If you don't have pip, install it using get-pip.py from https://bootstrap.pypa.io/get-pip.py

Creating a Django Project:

- Open a terminal: Navigate to your desired project directory.
- Create a project: Use the following command, replacing mysite with your project name:

```
django-admin startproject mysite
```

Running the Development Server:

- Navigate to the project directory: Use cd mysite.
- Start the server: Run python manage.py runserver.

Access Your Application:

Open http://127.0.0.1:8000/ in your web browser. You should see the Django welcome page, indicating a successful setup.

?**MongoDB:**

MongoDB is a flexible and scalable NoSQL database that stores data in a JSON-like format. It provides high performance, horizontal scalability, and seamless integration with Node.js, making it ideal for handling large amounts of structured and unstructured data.

Set up a MongoDB database to store your application's data.

Download: https://www.mongodb.com/try/download/community

Installation instructions: https://docs.mongodb.com/manual/installation/

?**HTML, CSS, and JavaScript**: Basic knowledge of HTML for creating the structure of your app, CSS for styling, and JavaScript for client-side interactivity is essential.

?**Database Connectivity**: Use a MongoDB driver or an Object-Document Mapping (ODM) library like Mongoose to connect your Django server with the MongoDB database and perform CRUD (Create, Read, Update, Delete) operations. To Connect the Database add the dependency in the pom.xml file

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
```

</dependency>

?**Development Environment**: Choose a code editor or Integrated Development Environment (IDE) that suits your preferences, such as Visual Studio Code, Sublime Text, or WebStorm.

• Visual Studio Code: Download from https://code.visualstudio.com/download

To run the existing Local Tourist App downloaded from Google Drive, Follow below steps:

Download the code from the drive link provided below.

https://drive.google.com/drive/folders/1ESQWdJO-bh9w3a_Y8J2WGe_3S-fNvvto?usp=sharing

Then, open the project in a suitable ide or code editor and run the application using the run button provided.

• The Local Tourist app will be accessible at http://127.0.0.1:8000/

You have successfully installed and set up the Train Booking application on your local machine. You can now proceed with further customization, development, and testing as needed.

Application flow

The project has 3 types of users – Renter and Owner and other will be Admin which takes care to all the user. The roles and responsibilities of these two types of users can be inferred from the API endpoints defined in the code. Here is a summary:

User:

1. Account Creation and Login: Users create an account with basic information (name, email, password) and log in to access the platform.
2. Campaign Discovery: Users can browse through various active fundraising campaigns based on categories, causes, or search terms.
3. Donation: Users can donate to any campaign they choose, with options for one-time or recurring donations.

Admin:

1. Platform Management: Admins have full control over the platform, including user management, campaign moderation, and financial management.
2. User Management: Admins approve or reject user registrations, manage user roles and permissions, and handle user inquiries.
3. Campaign Oversight: Admins review and approve/reject created campaigns to ensure they comply with platform guidelines.

Project Flow

Let's start with the project development with the help of the given activities.

Project setup and configuration

-

1. **Milestone 1: Project Setup and Configuration:**

Create project folders and files:

1.

Now, firstly create the Django project with the below command:

```
django-admin startproject project_name
```

Also, create an app to manage the rest operations with the command given below:

```
django-admin startapp app_name
```

2. **Install required tools and software:**

For the Django application to function well, we use the libraries mentioned in the prerequisites. Those libraries include

- Django
- MongoDB
- Corsheader
- Rest_framework

Database development

Milestone 3: Database Development:

- Set up a MongoDB database either locally or using a cloud-based MongoDB service like MongoDB Atlas.
- Create a database and define the necessary collections for users, properties and applications.
- Also let’s see the detailed description for the schemas used in the database.

1. Users:

The User model captures essential information for identification, including names, email addresses (acting as a unique identifier). Additionally, the unique identifier is very helpful to map the Local Tourists to the particular user profile.

2. Monuments:

The Monuments model encompasses data related to tourist attractions. It ihttps://drive.google.com/file/d/19vIOUFEv5B7ZSfLh3hw-bkrsWyRQzIgI/view?usp=sharinncludes essential information such as the monument's name, location, description, images, and user-generated content like ratings and reviews. Additional details about the monument's categorization, amenities, operating hours, and entry fees are also captured within this model.

3. Contributions:

The Contribution model tracks user-generated content related to monuments. It stores information about who added new locations, uploaded images, or submitted reviews and ratings. This model establishes a connection between users and their contributions, allowing for attribution, moderation, and potential incentives.

Frontend development

- **Milestone 4: Frontend Development:**
    - **Setup Frontend Application:**

Bringing the Local Tourist application to life involves a three-step development process. First, a solid foundation is built using template of Django. This includes creating the initial application structure, and organizing the project files for efficient development. Next, the user interface (UI) comes to life.

Design UI components:

Crafting a seamless user experience in Local Tourist App requires designing UI components. This involves creating individual elements like buttons, forms, and menus. We'll then define their arrangement and

visual style (layout and styling) for a cohesive look. Finally, navigation elements will be implemented to ensure users can easily move between different functionalities within the app.

- **Implement frontend logic:**

In the final leg of the frontend development, we'll bridge the gap between the visual interface and the underlying data. It involves the below stages.

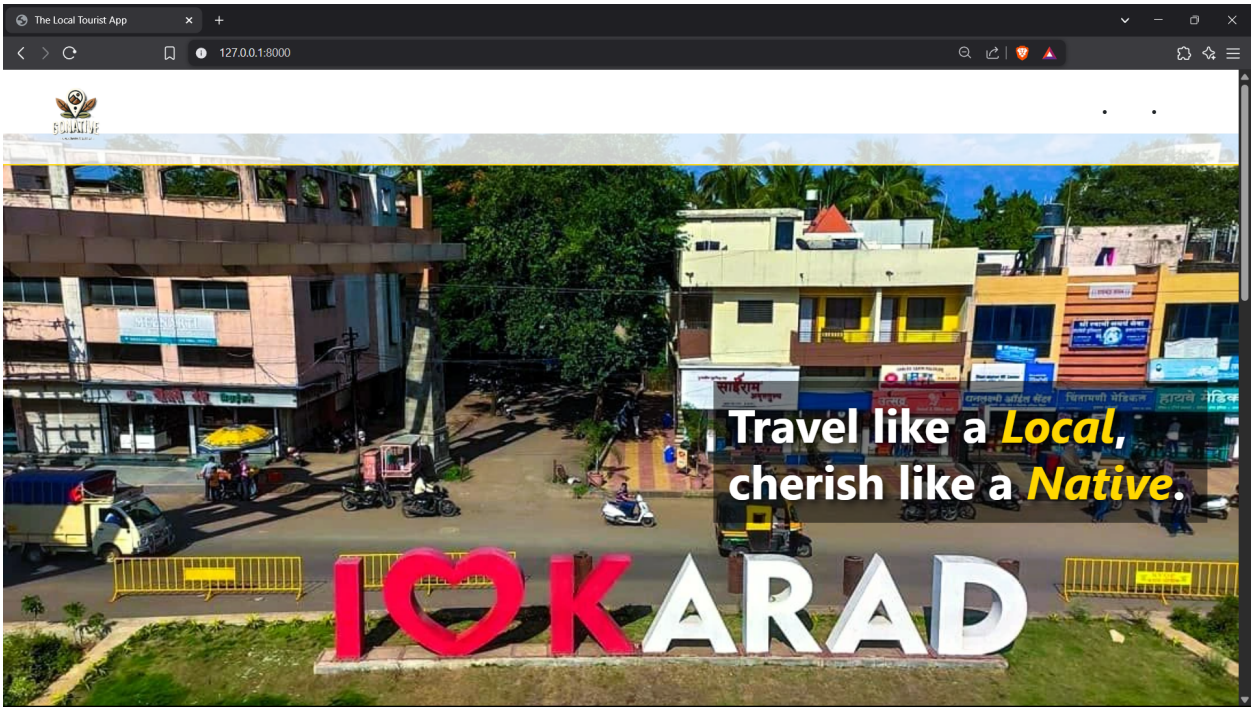
- Integration with API endpoints.
- Implement data binding.

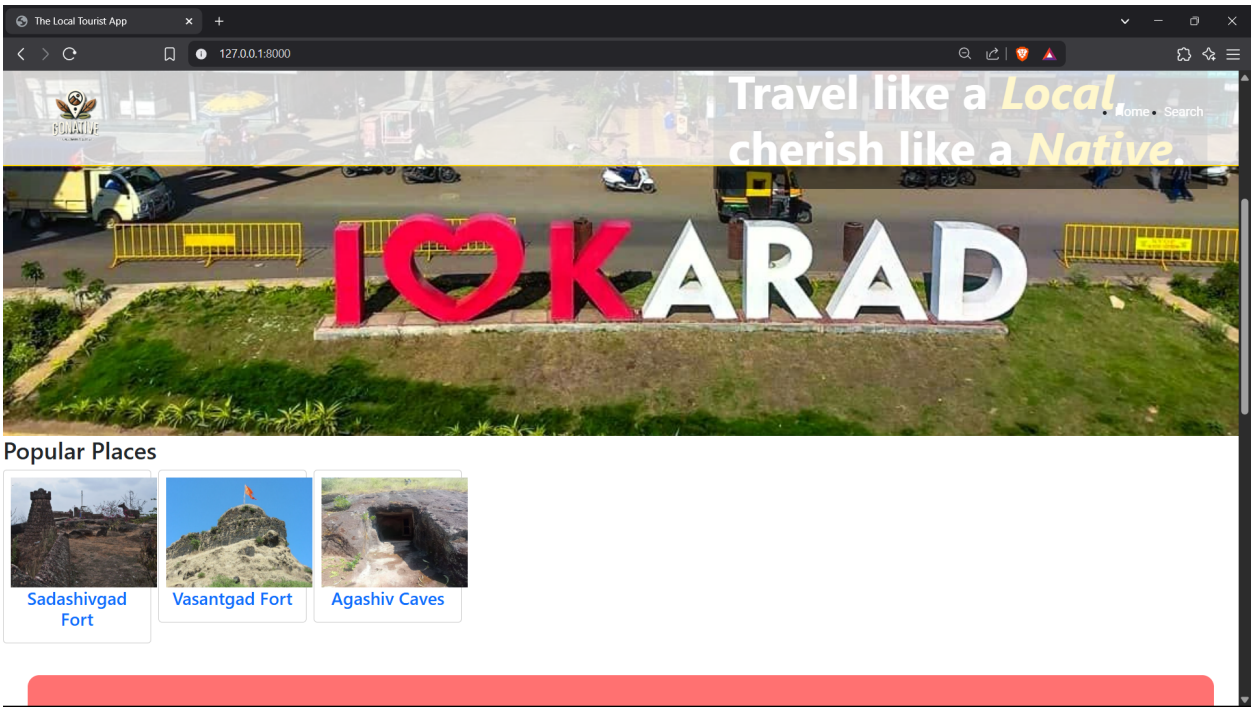
Project Implementation

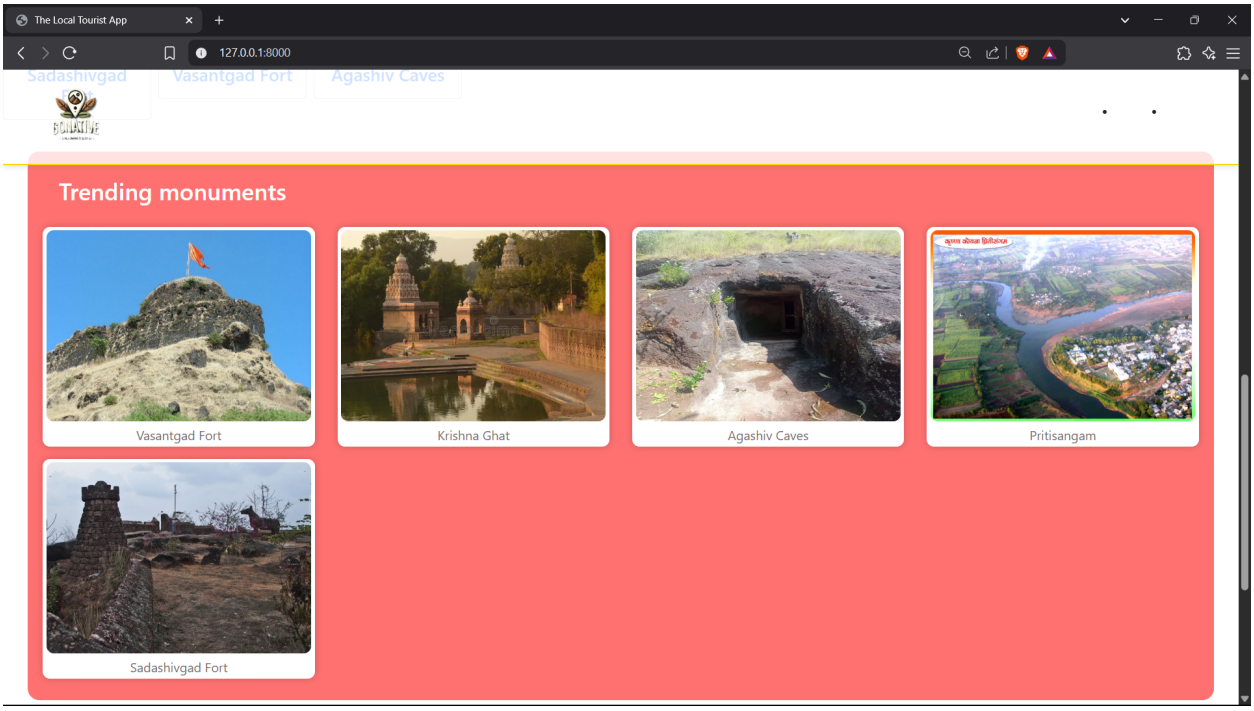
Milestone 5: Project Implementation:

On completing the development part, we then run the application one last time to verify all the functionalities and look for any bugs in it. The user interface of the application looks a bit like the images provided below.

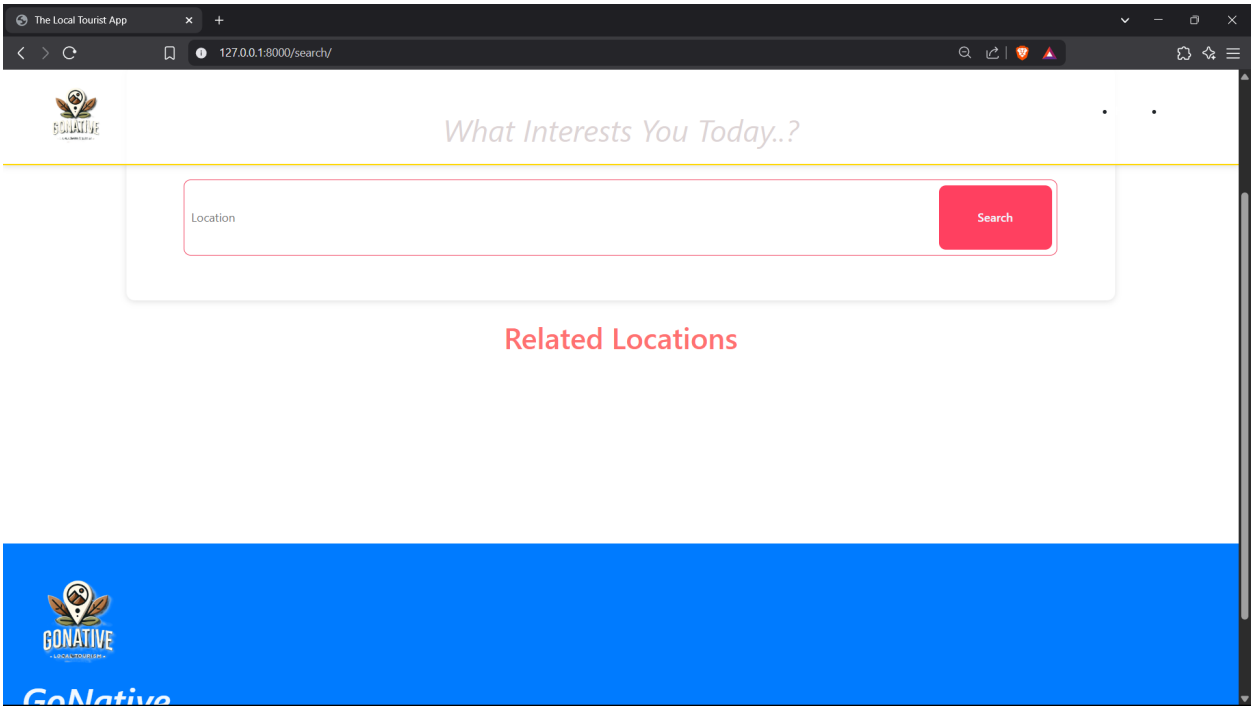
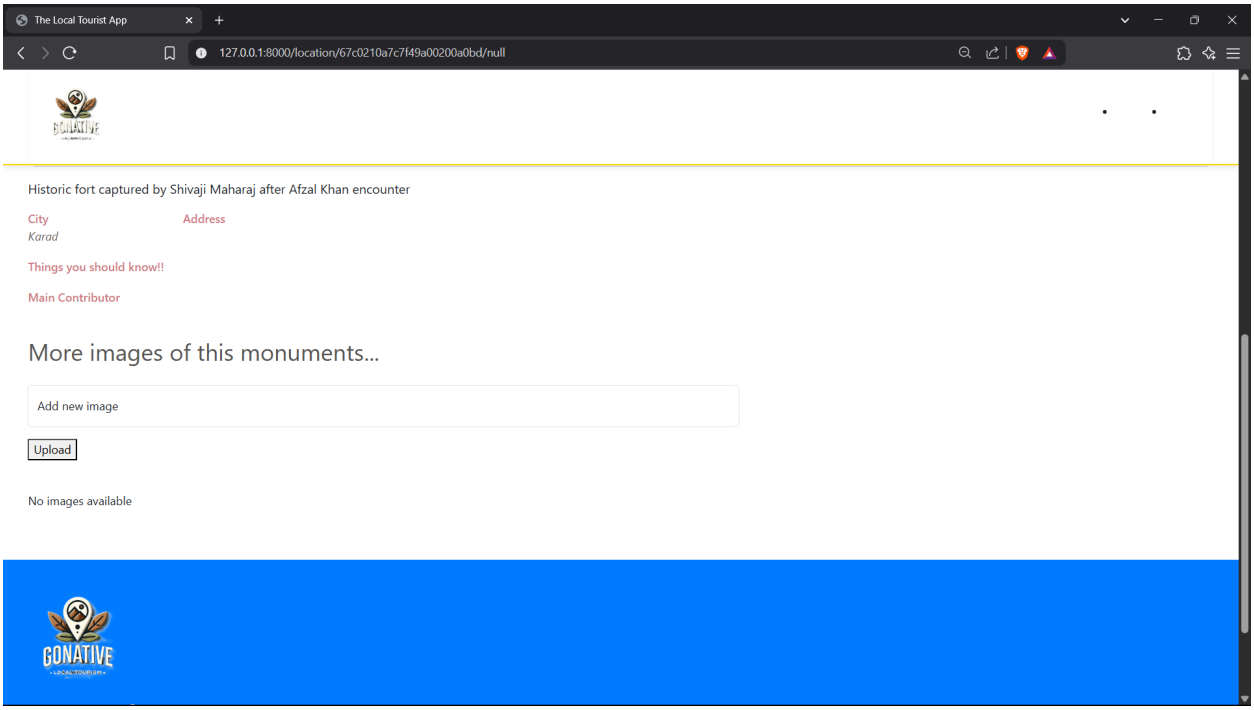
- **Landing page:**




The Local Tourist App
127.0.0.1:8000
Travel like a Local,
cherish like a Native.
Home
Search
I KARAD
Popular Places
Sadashivgad Fort
Vasantgad Fort
Agashiv Caves


The Local Tourist App
127.0.0.1:8000
Trending monuments
Vasantgad Fort
Krishna Ghat
Agashiv Caves
Pritisangam
Sadashivgad Fort

- **Search:**



- **Location:**