Project Title : INDIACITYGDP

(A Visualization of Urban Economic Metrics)

Developer

YASHWANTH SAI KASARABADA

Registered Email

yashwanthkasarabada2003@gmail.com

Mentor

DURGA PRASAD

Co-ordinator

MOHAMMAD HUSSAIN

Internship 5.0 (Data Visualization Stack)
NOV 2024 - JAN 2025

Contents

List of Figures

Chapter 1

Introduction

The IndiaCityGDP project provides an interactive platform for visualizing and analyzing critical economic metrics, such as GDP and productivity, for major Indian cities. Leveraging Power BI for advanced visualization and Streamlit for seamless web integration, it empowers policymakers, researchers, and business leaders to make data-driven decisions for sustainable urban development. With accurate data sourced from government reports, users can explore trends, compare city performances, and uncover key insights into urban economic growth. This project lays the foundation for informed decisions in urban planning, policy-making, and investment, with potential for future integration of real-time data and predictive analytics to deepen its impact.

Chapter 2

Project scope

2.1 Scope of the Project

The **IndiaCityGDP** project focuses on the collection, analysis, and visualization of GDP and productivity data for major Indian cities. Using Power BI for interactive dashboards and Streamlit for web-based access, the platform enables users to filter, compare, and export insights, targeting stakeholders such as policymakers, researchers, and urban planners.

2.1.1 Inclusions

- Data collection and analysis from reliable sources like government reports and economic surveys.
- Interactive Power BI dashboards showcasing GDP and productivity data.
- Integration of dashboards with Streamlit for web-based user interaction.
- Filtering and comparison of metrics across cities and time periods.
- Exporting visualized data and insights for offline analysis.
- Comprehensive documentation of methodology, tools, and deployment.

2.1.2 Exclusions

- Real-time data integration and advanced predictive analytics.
- Geospatial analysis and mapping features.
- Multi-language support for the platform interface.
- Economic indicators beyond GDP and productivity (e.g., unemployment, inflation).

2.1.3 Limitations and Constraints

- Dependence on the accuracy and availability of public data sources.
- Aggregated city-level data without micro-level granularity.
- Potential performance issues with larger datasets.
- Reliance on specific technologies like Power BI, Streamlit, and Python libraries.
- Scalability limited by current infrastructure and hosting capacity.

2.1.4 Future Enhancements

- Real-time data integration for up-to-date insights.
- Advanced predictive models to forecast economic trends.
- Geospatial visualizations for better spatial insights.
- Inclusion of additional economic metrics for a comprehensive view.
- Multi-language support to cater to a broader audience.

.

Chapter 3

Requirements

3.1 Requirements for IndiaCityGDP

3.1.1 Functional Requirements

- **Data Processing:** Ingest and clean GDP and productivity data from reliable sources using Python libraries like Pandas and NumPy.
- **Visualization:** Create interactive Power BI dashboards with filters for city, time, and metrics, supporting detailed data exploration.
- **User Interaction:** Provide a Streamlit-based interface for accessing dashboards, filtering data, and exporting insights to CSV or PDF.
- **Documentation:** Include user guides and deployment instructions.

3.1.2 Non-Functional Requirements

- **Performance:** Fast rendering of large datasets and support for multiple users.
- **Scalability:** Expandable to new cities and metrics over time.
- **Usability:** Intuitive navigation and visually appealing dashboards with legends and tooltips.
- **Security:** Basic encryption for sensitive data and secure user access.
- **Compatibility:** Responsive design for both desktop and mobile, compatible with major browsers.
- **Maintainability:** Modular codebase for easy updates and adherence to standard coding practices.

3.1.3 User Stories

- Researchers analyze GDP and productivity trends across cities.
- Policymakers compare economic performance for informed decisions.
- Business leaders identify cities with high economic potential for investments.
- Users filter and export data for offline analysis or presentations.

3.1.4 Use Cases

- **Data Analysis:** Users visualize and compare economic data across cities and time periods.
- **City Comparison:** Stakeholders compare cities' GDP and productivity through side-by-side charts.
- **Interactive Dashboards:** Users explore data by interacting with charts, filters, and detailed views.
- **Navigation:** Users access different platform sections with ease.

Chapter 4

Technology Stack

The **IndiaCityGDP: A Visualization of Urban Economic Metrics** project utilizes a diverse set of technologies, frameworks, and tools to deliver a comprehensive solution. This includes data analysis, visualizations, and the integration of AI-driven features like a chatbot. Below is the detailed technical stack:

4.0.1 Programming Languages

The following programming languages are used to develop the backend logic, data processing, and integrate AI capabilities:

- **Python:** Used for data processing, analysis, and integration of various AI functionalities, including the chatbot. Libraries like Plotly and Pandas facilitate data manipulation and visualization.
- **DAX (Data Analysis Expressions):** Used within Power BI for calculations and aggregations to visualize economic metrics like GDP and productivity.

4.0.2 Frameworks/Libraries

The project relies on multiple frameworks and libraries for various tasks, including data analysis, visualization, and communication:

- **Pandas:** A powerful Python library for data manipulation and analysis, used to clean, transform, and process large datasets.
- **Plotly Express:** A Python graphing library used to create interactive and visually appealing plots and charts, which are integrated into the Power BI dashboard for enhanced data exploration.
- **Smtplib:** A Python library used to send automated email notifications or alerts from the application, such as when a new dataset or report is generated.

- **OpenAI API:** Used to integrate AI capabilities into the project, particularly for developing a chatbot feature that provides users with assistance on using the platform, answering queries, and providing insights on the data.

4.0.3 Databases

The project does not rely on a traditional database management system. Instead, data is sourced externally:

- **CSV/Excel Files:** Economic data, including GDP and productivity metrics, is collected and stored in CSV and Excel formats from public government and economic surveys.

4.0.4 Tools/Platforms

The following tools and platforms have been utilized for the development, deployment, and execution of the project:

- **Microsoft Power BI:** A key tool used for creating dynamic, interactive visualizations and dashboards that allow users to explore the GDP and productivity of cities in India.
- **Streamlit:** A web framework used to build an interactive user interface where Power BI visualizations are embedded, allowing users to engage with the data effortlessly.
- **VS Code/IDE:** Integrated Development Environment (IDE) used for writing Python code, debugging, and managing the project files.
- **OpenAI Platform:** The platform for integrating AI models, such as GPT, for the chatbot functionality that helps users with their queries.
- **Smtp Server:** Used for email functionality, allowing the system to send Feedback.

Chapter 5

System Design

The system design for the **IndiaCityGDP: A Visualization of Urban Economic Metrics** project focuses on creating a scalable, interactive platform for visualizing and analyzing urban economic data, particularly GDP and productivity metrics across Indian cities. The system is designed to provide a seamless user experience by integrating data processing, visualization, and AI-based chatbot assistance into a single platform. The key design decisions, components, and interactions are outlined below.

5.0.1 System Architecture

The system follows a modular design, where each component interacts with other parts of the system in a well-defined manner. The architecture is based on a client-server model, where the front-end (client) communicates with the back-end (server) to request data and render interactive visualizations.

The overall architecture consists of the following components:

- **Frontend (User Interface):**
  - The user interface is built using **Streamlit**, which integrates Power BI visualizations and provides users with an interactive platform to explore the data.
  - The front-end also hosts the AI-based chatbot powered by the **OpenAI API**, assisting users in navigating the platform and answering their queries.
- **Backend (Data Processing):**
  - The backend is built in **Python** and handles the processing of economic data. It performs operations such as data cleaning, transformation, and aggregation using **Pandas**.
  - Data is processed and prepared for visualization in Power BI and sent to the front-end via API calls.

- **Data Sources:**
  - Data is sourced from government reports, economic surveys, and CSV/Excel files, which are updated periodically to keep the visualizations up-to-date.
  - Public APIs may also be used for real-time data fetching when needed.
- **Visualization:**
  - The data is visualized in **Power BI**, with charts, graphs, and other interactive components, such as slicers, to allow users to filter and analyze the data.
- **AI Chatbot:**
  - A chatbot, powered by the **OpenAI API**, assists users in answering questions about the data, providing recommendations, and guiding them through using the platform.
  - The chatbot is integrated into the front-end to provide a conversational interface for querying the system.

5.0.2 Component Interaction

The components of the system interact with each other as follows:

- The user accesses the front-end via a web browser, where the **Streamlit** interface is hosted.
- Upon selecting a city or a metric, the front-end makes API calls to the backend to fetch the corresponding data.
- The backend processes the data (using **Pandas**), aggregates it, and sends it to the **Power BI** platform for visualization.
- The user interacts with the Power BI dashboard to explore the data.
- The AI chatbot, integrated into the front-end, listens to the user's queries and, using **OpenAI's API**, provides intelligent responses related to the data, its analysis, and navigation of the platform.
- If the user requests an email notification (e.g., about new reports), the backend uses **Smtplib** to send the notifications via email.

5.0.3 Design Patterns

The following design patterns are employed in the system:

- **Model-View-Controller (MVC):**
  - The system follows the MVC pattern, where the **Model** represents the data (economic metrics and GDP), the **View** represents the Power BI visualizations, and the **Controller** handles the user input, data processing, and communication between the model and the view.
- **Microservices Architecture (for AI Chatbot):**
  - The AI chatbot is designed as a microservice that communicates with the main system via API calls, ensuring that the chatbot is loosely coupled with the rest of the platform and can be easily updated or replaced.
- **Singleton Pattern (for Chatbot Communication):**
  - The OpenAI API integration follows the Singleton pattern to ensure that only one instance of the communication with the AI service exists, minimizing resource consumption.

5.0.4 Trade-offs and Considerations

During the design phase, several trade-offs were considered:

- **Real-time Data:** While real-time data fetching from APIs would provide up-to-date insights, it would also introduce complexity in terms of rate limits, API reliability, and handling of incomplete data. Therefore, periodic updates from government sources were prioritized.
- **AI Chatbot Complexity:** The chatbot is integrated into the front-end using OpenAI's API, but as it is dependent on external APIs, it can incur latency. However, the benefits of providing intelligent, conversational support outweigh the potential delay.
- **Data Privacy and Security:** Since the platform does not store sensitive user data, no advanced security measures like encryption are implemented. However, API keys and other sensitive information are securely stored using environment variables.

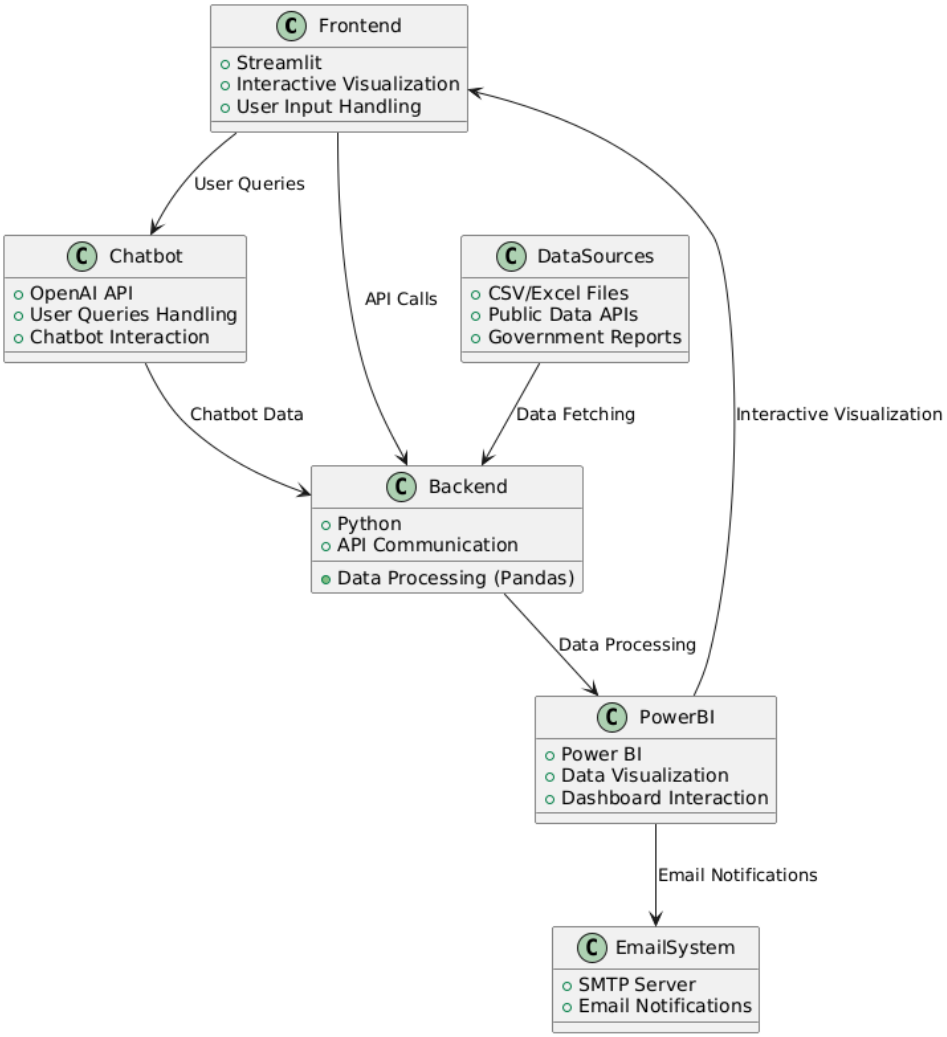

Figure 5.1: UML Diagram

5.0.5 UML Diagram

Below is a high-level UML diagram representing the interactions between various components of the system:

The diagram visualizes the flow of data between the frontend, backend, visualization platform, and external services like the AI chatbot and email notifications.

Chapter 6

Development

The development of the **IndiaCityGDP: A Visualization of Urban Economic Metrics** project involved several technologies and frameworks to achieve a seamless and efficient system for urban economic data visualization.

6.0.1 Technologies and Frameworks

The project leverages a variety of technologies for its front-end, back-end, and data processing needs:

- **Frontend: Streamlit** is used to build the interactive web interface, allowing users to visualize the data in real-time. It integrates with Power BI for data visualization and serves as the medium for the AI chatbot.
- **Backend: Python** is the primary language used for data processing, with **Pandas** and **NumPy** for data manipulation and transformation.
- **Visualization: Power BI** is used to create dynamic and interactive visualizations that allow users to explore and analyze GDP and productivity metrics across cities.
- **Chatbot Integration: OpenAI API** is used to power the chatbot, providing intelligent responses to user queries related to the data.
- **Email System:** The **Smtplib** library is used for sending email notifications to users.

6.0.2 Coding Standards and Best Practices

To ensure maintainable and readable code, several coding standards and best practices were followed:

- **Consistent Naming Conventions:** Clear and descriptive names were used for variables, functions, and classes, adhering to PEP 8 standards for Python.
- **Modular Code Design:** The code was organized into modular components (e.g., data processing, visualization, and chatbot) to ensure scalability and ease of debugging.
- **Documentation:** Detailed inline comments and docstrings were added for all functions and classes to ensure clarity for future developers.
- **Version Control:** Git was used for version control, with regular commits and branching strategies for collaborative development.

6.0.3 Challenges and Solutions

Throughout the development process, several challenges were encountered:

- **Data Inconsistencies:** The data collected from multiple sources (e.g., government reports, APIs) often contained inconsistencies or missing values. This was addressed by applying data cleaning techniques using **Pandas**, such as filling missing values and standardizing data formats.
- **Power BI Integration:** Integrating Power BI with the backend required overcoming issues related to real-time data updates. This challenge was mitigated by automating data extraction and ensuring a smooth flow between the backend and Power BI via API calls.
- **Chatbot Responses:** Fine-tuning the AI chatbot for specific data-related queries proved challenging. By training the model on relevant data and refining its prompts, we ensured the chatbot could respond accurately to user questions.
- **Performance Optimization:** Handling large datasets and rendering interactive visualizations created performance bottlenecks. This was addressed by optimizing data processing algorithms and using efficient libraries like **NumPy** for heavy data computations.

Chapter 7

Testing

The testing phase ensured that the **IndiaCityGDP: A Visualization of Urban Economic Metrics** project functioned as expected, providing accurate data visualization and seamless user interaction. The testing process involved various levels of testing to ensure the system's reliability and performance.

7.0.1 Testing Approach

The testing was performed at multiple stages, including unit testing, integration testing, and system testing:

- **Unit Testing:** Individual components such as data processing functions, chatbot queries, and visualization generation were tested independently using **pytest** to ensure correctness.
- **Integration Testing:** The interaction between the front-end, back-end, and data sources was tested to ensure smooth data flow and proper handling of user inputs and outputs.
- **System Testing:** The entire system was tested end-to-end to ensure that the user could interact with the chatbot, view visualizations, and receive email notifications without errors.

7.0.2 Testing Results

During testing, the following issues were identified and resolved:

- **Data Formatting Errors:** Some data entries had formatting issues which were fixed by cleaning the data using **Pandas** before processing.
- **API Call Failures:** A few API calls failed due to incorrect parameters; these were fixed by reviewing the API documentation and refining the request structure.

- **Performance Bottlenecks:** The application faced performance degradation with large datasets, which was mitigated by optimizing the data processing and visualization logic.

7.0.3 Test Coverage

The testing covered the following key aspects:

- **Functionality:** Ensuring that all features, including data visualization, chatbot interaction, and email notifications, worked as expected.
- **Usability:** Verifying that the application was easy to navigate and user-friendly, especially the Streamlit interface and chatbot interactions.
- **Performance:** Ensuring the system handled large datasets efficiently without significant delays.

Chapter 8

Deployment

Although the **IndiaCityGDP: A Visualization of Urban Economic Metrics** project has not yet been deployed, the following deployment requirements and steps outline how the application can be deployed in different environments.

8.0.1 Deployment Requirements

The deployment of the application requires the following:

- **Web Hosting Platform:** A cloud hosting service such as **Heroku**, **AWS**, or **Google Cloud Platform** is required to deploy the Streamlit front-end and Python back-end.
- **Power BI Service:** A Power BI Pro or Premium subscription to publish and share the visualizations on the web.
- **SMTP Server:** A working email system (e.g., **SendGrid**, **Gmail SMTP**) to enable email notifications.
- **Python Environment:** A Python environment with the necessary libraries, such as **Pandas**, **Plotly**, **Streamlit**, and the **OpenAI API**.
- **Database Access:** If required, a cloud-based database (e.g., **AWS RDS**, **Google Cloud SQL**) for storing historical data and logs.

8.0.2 Deployment Process

The following steps can be followed for deploying the application:

- **Step 1: Prepare the Environment:** Set up the necessary environment, including Python 3.x, required libraries, and any cloud platform tools for deployment.

- **Step 2: Deploy Front-End:** Use **Streamlit** to deploy the user interface on a web hosting platform like **Heroku** or **AWS Elastic Beanstalk**. This will require a 'requirements.txt' file for Python dependencies.
- **Step 3: Deploy Back-End:** Host the back-end Python services on a cloud platform. Ensure API calls between the back-end and front-end are functional.
- **Step 4: Power BI Integration:** Publish the Power BI dashboards to the Power BI Service and integrate them with the front-end.
- **Step 5: Set Up Email Notifications:** Configure the SMTP server for email notifications and ensure it is properly connected to the application for sending updates.
- **Step 6: Monitor and Maintain:** Set up monitoring for the application and ensure the system is scalable and available, with regular updates as needed.

8.0.3 Future Deployment Considerations

For deploying in a production environment, additional considerations include:

- **Security:** Implement security measures like HTTPS for secure communication, proper authentication for the chatbot, and safe handling of user data.
- **Scalability:** Ensure the system can handle increasing data volumes and user interactions by scaling the infrastructure (e.g., using Kubernetes or serverless functions).
- **CI/CD Pipeline:** Set up Continuous Integration/Continuous Deployment (CI/CD) pipelines for automatic testing and deployment of updates.

Chapter 9

User Guide

The **IndiaCityGDP: A Visualization of Urban Economic Metrics** Streamlit application allows users to explore and visualize the GDP and productivity metrics of various Indian cities. Below are the instructions for using the application, including setup and troubleshooting tips.

9.0.1 Setup and Configuration

To run the Streamlit application locally, follow these steps:

1. **Clone the Repository:** Download or clone the project repository from GitHub.
2. **Install Dependencies:** Navigate to the project directory and install the required Python libraries using the following command:

```
pip install -r requirements.txt
```

3. **Run the Application:** Start the Streamlit application by running the following command in the terminal:

```
streamlit run app.py
```

4. **Access the Application:** Open your web browser and go to the provided local URL (usually ‘http://localhost:8501‘) to access the application.

9.0.2 Using the Application

Once the application is running, users can:

- **Select a City:** Use the dropdown menu to select an Indian city for which you want to view GDP and productivity metrics.
- **View Visualizations:** The application will display various charts and graphs that represent the economic metrics of the selected city.
- **Interact with the Chatbot:** Ask the integrated AI chatbot any questions about the data (e.g., "What is the GDP of Mumbai?"). The chatbot will respond with relevant information based on the dataset.
- **Email Notifications:** Users will receive email notifications for any important updates or changes to the data.

9.0.3 Troubleshooting Tips

If you encounter any issues while using the application, try the following:

- **Application Not Loading:** Ensure that all dependencies are correctly installed by running ‘pip install -r requirements.txt‘ again. If the issue persists, check for any missing dependencies or errors in the terminal.
- **Data Not Displaying Correctly:** Verify that the data files are located in the correct directory and are in the expected format (e.g., CSV or Excel).
- **Chatbot Not Responding:** Ensure that the OpenAI API key is correctly configured in the ‘config.py‘ file, and that you have an active internet connection.
- **Email Notifications Not Sending:** Check your email SMTP configuration in the settings and ensure that the email server is reachable.

Chapter 10

Conclusion

The **IndiaCityGDP: A Visualization of Urban Economic Metrics** project successfully integrates data visualization, AI chatbot interaction, and real-time data exploration to provide an insightful platform for analyzing the economic performance of Indian cities. Through the use of powerful tools such as Streamlit, Power BI, and OpenAI, the project not only enables users to explore GDP and productivity metrics but also enhances user engagement with an interactive chatbot. The application serves as a valuable resource for policymakers, business leaders, and researchers to make informed decisions based on reliable urban economic data. While the project has met its initial objectives, future enhancements can focus on adding more cities, improving data accuracy, and expanding the functionality of the chatbot. Overall, this project has provided valuable learning experiences in data visualization, AI integration, and web application development.

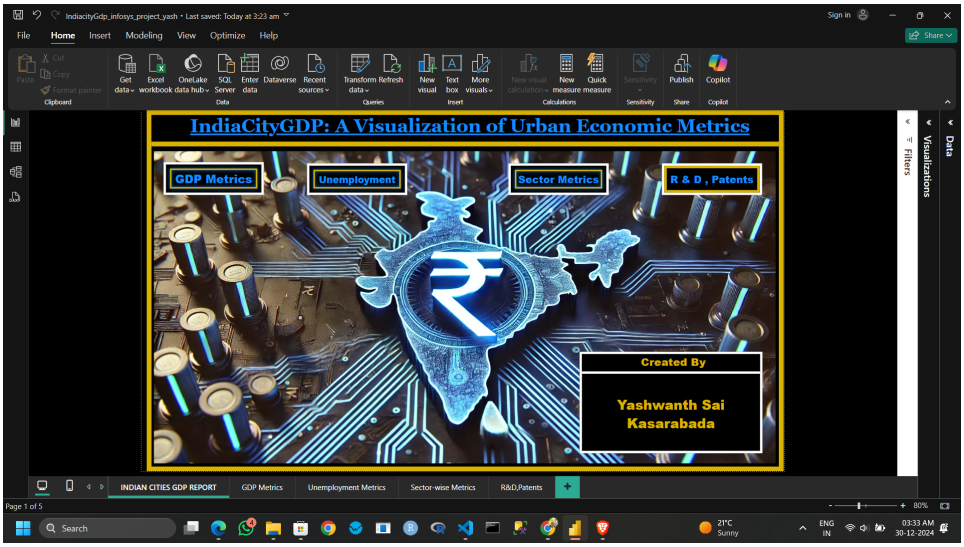

Figure 10.1: Power Bi Report Homepage

Figure 10.2: Streamlit Application Overview page