# VISVESVARAYA TECHNOLOGICAL UNIVERSITY

# Jnana Sangama, Belgavi-590018

### AN INTERNSHIP REPORT

### [Subject Code: 21INT82]

### ON

**“****SKILL ENHANCEMENT COURSE AND IMPLEMENTATION OF INTERACTIVE CHART GENERATOR”**

***Submitted in partial fulfillment of the requirement for award of degree***

***of***

**BACHELOR OF ENGINEERING**

**in**

**COMPUTER SCIENCE AND ENGINEERING**

#### By

#### RUPAM BHATTACHARYYA 1EP21CS089

**Under the guidance of**

**Mrs. Manimegalai A**

**Asst. Professor**

**Dept. of CSE, EPCET**



**2024-2025**



# CERTIFICATE

This is to certify that the Internship (21INT82) entitled “SKILL ENHANCEMENT COURSE AND IMPLEMENTATION OF INTERACTIVE CHART GENERATOR” is a bonafide work carried out by **Rupam Bhattacharyya[1EP21CS089]**, in partial fulfillment of the requirements of BACHELOR OF ENGINEERING in COMPUTER SCIENCE AND ENGINEERING in VISVESVARAYA TECHNOLOGICAL UNIVERSITY, Belgaum, during the year 2024-2025. It is certified that corrections/suggestions recommended have been incorporated in the Internship report.





|  |  |  |
| --- | --- | --- |
| **(Signature of the Guide)**  **Prof. Manimegalai A**  **Asst. Prof., Dept. of CSE,**  **EPCET, Bengaluru** | **(Signature of the HOD)**  **Dr. I. Manimozhi**  **HOD, Dept. of CSE,**  **EPCET, Bengaluru** | **(Signature of the Principal)**  **Dr. Mrityunjaya V Latte**  **Professor & Principal**  **EPCET, Bengaluru** |

### Examiners:

**Examiner 1 Examiner 2**

### Name: Name:

**Signature with date: Signature with date:**

**ACKNOWLEDGEMENT**

First and foremost, I would like to express my sincere regards and thanks to **Management** of **East Point Group of Institutions, Bengaluru** for providing me an opportunity to work on this Internship.

I am ineffably indebted to **Dr. S Prakash, Senior Vice President**, **East Point Group of Institutions,** for his conscientious guidance and encouragement to accomplish this Internship.

I would like to express my humble and sincere thanks to **Dr. Mrityunjaya V Latte, Principal, East Point College of Engineering and Technology** for his suggestions that motivated me for the successful completion of my internship.

I would like to express my heartfelt thanks to **Dr. I Manimozhi**, **Professor and Head of Department** of Computer Science and Engineering, EPCET for her valuable advice and suggestions to do my best in this Internship.

I am obliged to guide **Mrs. Manimegalai A**, Associate Professor, Dept. of CSE, Internship Coordinators **Prof. Madhushree**, Assistant Professor, Dept. of CSE and **Prof. Nithyananda C R,** Associate Professor, Dept. of CSE who have rendered valuable assistance and guidance for the Internship.

I would like to thank my **Parents** and **Friends** for their support and encouragement during the course of my internship. Finally, I offer my regards to all the **Faculty members of the CSE Department** and all those who supported me in any respect during the Internship.

**Name: RUPAM BHATTACHARYYA USN: 1EP21CS089**

**Abstract**

This report documents the skill enhancement courses initiated by VTU University as an innovative alternative to traditional internships. The initiative aimed to equip students with hands-on experience and theoretical insights in modern technological domains, ensuring they are industry-ready upon course completion. The program covered five key areas: C Language, Data Structures with C, React, Python Libraries for Data Science, and App Development using Kotlin.

Each course was meticulously designed to address both foundational knowledge and advanced concepts, ensuring a comprehensive understanding of the subject matter. The C Language and Data Structures courses laid the groundwork for efficient programming and algorithmic thinking, while React focused on creating dynamic, responsive web interfaces. The Python Libraries for Data Science course empowered students with tools for data analysis, visualization, and machine learning, crucial for decision-making in data-driven industries. Lastly, the App Development course introduced modern mobile application development practices using Kotlin, a language favoured for Android platforms.

The "Interactive Chart Generator Using React and Python" is a web-based application designed to simplify the visualization of data through customizable charts. This tool enables users to input data manually, specify categories and values, and dynamically generate various chart types, including bar, pie, line, and scatter plots. Leveraging a user-friendly interface, the application allows for seamless data entry and editing, with options to add or remove rows as needed.

The generated charts are displayed in real-time, and users can download them in high-quality image format for offline use. The application utilizes React.js for its responsive front-end and Axios for secure API communication with the server. It also incorporates CSRF token-based protection to ensure data security during user interactions. Designed for researchers, educators, and professionals, this tool eliminates the complexities of traditional chart creation, offering an efficient and intuitive solution for data visualization needs.

# 

# CONTENTS

**Chapter no. Description Page no.**

1. Company Profile 1
2. Skill Enhancement Courses 2
3. Courses in Detail 3
   1. [C Programming Language 3](#_TOC_250007)
   2. Data Structure Using C 6
   3. Python in Data Science 9
   4. React in Web Development 12
   5. Kotlin in App Development 15
4. Implementation 17
5. Result 22
6. Conclusion 24

[References 2](#_bookmark0)5

|  |  |  |
| --- | --- | --- |
| **List of Figures** | | |
| **Fig. no.** | **Description** | **Page no.** |
| 3.1 | Execution Flow in C | 3 |
| 3.2 | File Handling in C | 4 |
| 3.3 | File Operations | 5 |
| 3.4 | Sequence flow | 5 |
| 3.5 | Hash function in Data Structure | 8 |
| 3.6 | Data Collection | 9 |
| 3.7 | Data Visualization | 10 |
| 3.8 | Python Libraries for Data Science | 11 |
| 3.9 | React Structure | 12 |
| 3.10 | React Flow Diagram | 13 |
| 3.11 | File Structure in React | 14 |
| 3.12 | Types of Activities | 15 |
| 3.13 | Activity Lifecycle Diagram | 16 |
| 3.14 | Basic UI of Android | 16 |
| 4.1 | Manual Data Form | 18 |
| 4.2 | App.js File | 18 |
| 4.3 | Running Server in Localhost | 19 |
| 4.4 | Views.py | 20 |
| 4.5 | App URL | 20 |
| 4.6 | Project URL | 20 |
| 4.7 | Running Django Server | 21 |
| 5.1 | React App Front Page | 22 |
| 5.2 | List of Charts | 22 |
| 5.3 | Pie Chart with Given Data | 23 |
| 5.4 | Django Backend Server | 23 |

# Chapter 1

# COMPANY PROFILE

Visvesvaraya Technological University (VTU), established in 1998 and headquartered in Belagavi, Karnataka, is a leading institution in the field of technical education in India. Named after Sir M. Visvesvaraya, one of India’s foremost engineers and statesmen, the university is committed to fostering innovation, research, and academic excellence. VTU caters to over 200 affiliated colleges, serving thousands of students across undergraduate, postgraduate, and doctoral programs in engineering, technology, and management.

Recognized for its robust curriculum and emphasis on practical learning, VTU has consistently adapted to the evolving needs of the industry. The university offers a diverse range of programs that combine theoretical foundations with hands-on training, ensuring students are well-prepared for real-world challenges. In line with its mission to bridge the gap between academia and industry, VTU has introduced skill enhancement courses as an alternative to traditional internships, providing students with an opportunity to gain practical expertise in cutting-edge technologies.

VTU’s infrastructure includes state-of-the-art laboratories, research centers, and digital resources, supporting a vibrant learning environment. The university’s collaborations with industry leaders and government organizations further enhance its educational offerings, providing students with access to internships, workshops, and live projects.

With a strong focus on innovation and sustainability, VTU continues to contribute significantly to technological advancement and societal development. The skill enhancement courses reflect VTU’s commitment to nurturing talent and preparing students for leadership roles in a rapidly changing global landscape.

**Chapter 2**

**SKILL ENHANCEMENT COURSES**

Skill enhancement courses are specialized programs designed to provide students with practical, industry-relevant knowledge and hands-on experience in various technical domains. These courses, introduced by VTU University as an alternative to traditional internships, aim to bridge the gap between academic learning and real-world applications, ensuring that students are well-equipped for professional challenges.

**2.1 Importance of Skill Enhancement Courses:**

In today’s competitive and technology-driven world, possessing theoretical knowledge alone is insufficient. Employers seek candidates with practical expertise and problem-solving abilities. Skill enhancement courses address this need by focusing on experiential learning, allowing students to apply their knowledge to real-life scenarios. These programs are particularly important for students who lack access to conventional internship opportunities, ensuring they still gain valuable exposure to industry practices.

**2.2 Benefits of Skill Enhancement Courses:**

1. **Practical Knowledge**: These courses emphasize hands-on training in key areas such as programming, data science, web development, and mobile app development, enhancing technical proficiency.
2. **Industry Readiness**: By simulating real-world challenges through projects and assignments, students gain the confidence and skills required for professional roles.
3. **Flexibility**: Offered as an alternative to internships, these courses provide a structured learning experience within a flexible timeframe.
4. **Enhanced Employability**: Completing these courses adds value to students' resumes, making them more attractive to potential employers.
5. **Skill Development**: Students acquire both foundational and advanced technical skills, improving their problem-solving abilities and innovation potential.

**Chapter 3**

**COURSES IN DETAIL**

### 3.1 C Language: A Comprehensive Overview

#### 3.1.1 Introduction to C Language

C is a versatile, high-performance programming language developed by Dennis Ritchie in 1972. Known for its simplicity and flexibility, C remains a foundational language in computer science, widely used for system programming, developing operating systems, embedded systems, and application development.

**3.1.2 Features of C Language**

1. **Simplicity**: Easy to understand and implement.
2. **Portability**: Code can be executed on different machines with minimal changes.
3. **Structured Language**: Programs can be divided into functions, improving readability.
4. **Efficiency**: Offers fast and efficient execution.

#### 3.1.3 Syntax and Structure of a C Program

#### Figure 3.1: Execution Flow in C Programming

#### 3.1.5 Pointers:

Pointers store the memory address of a variable.

**Syntax:**

data\_type \*pointer\_name;

#### 3.1.6 Structures:

#### Structures group related variables of different types under one name. Syntax:

struct structure\_name {

data\_type member1;

data\_type member2;

};

#### 3.1.7 File Handling:

**File Handling in C**

File handling in C is a mechanism that allows programs to read from and write to files stored on a storage device. It provides a way to persist data beyond the runtime of a program and is essential for tasks such as data logging, configuration management, and structured data processing.

The standard input/output library (stdio.h) in C provides functions for file handling. Here are the main concepts and functions involved in file handling in C:

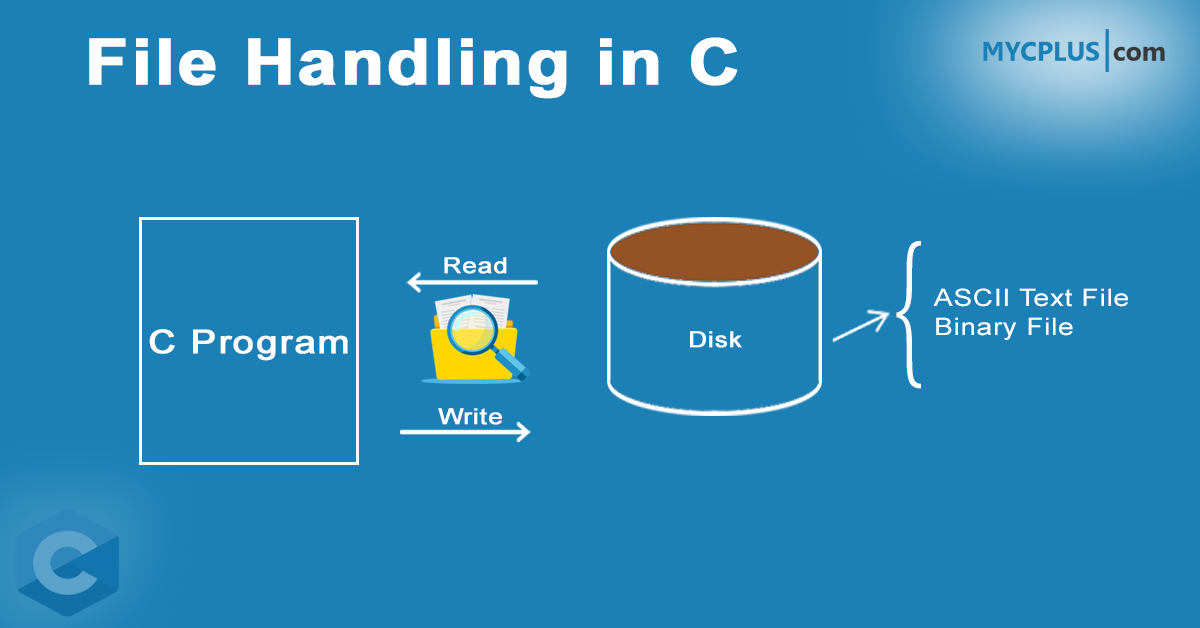


Figure 3.2: File Handling in C

**File Operations**

C supports the following basic file operations:

1. **Creating a file**: Creating a new file for storing data.
2. **Opening a file**: Associating a file with a stream.
3. **Reading from a file**: Extracting data from a file.
4. **Writing to a file**: Storing data into a file.
5. **Closing a file**: Releasing resources associated with the file.

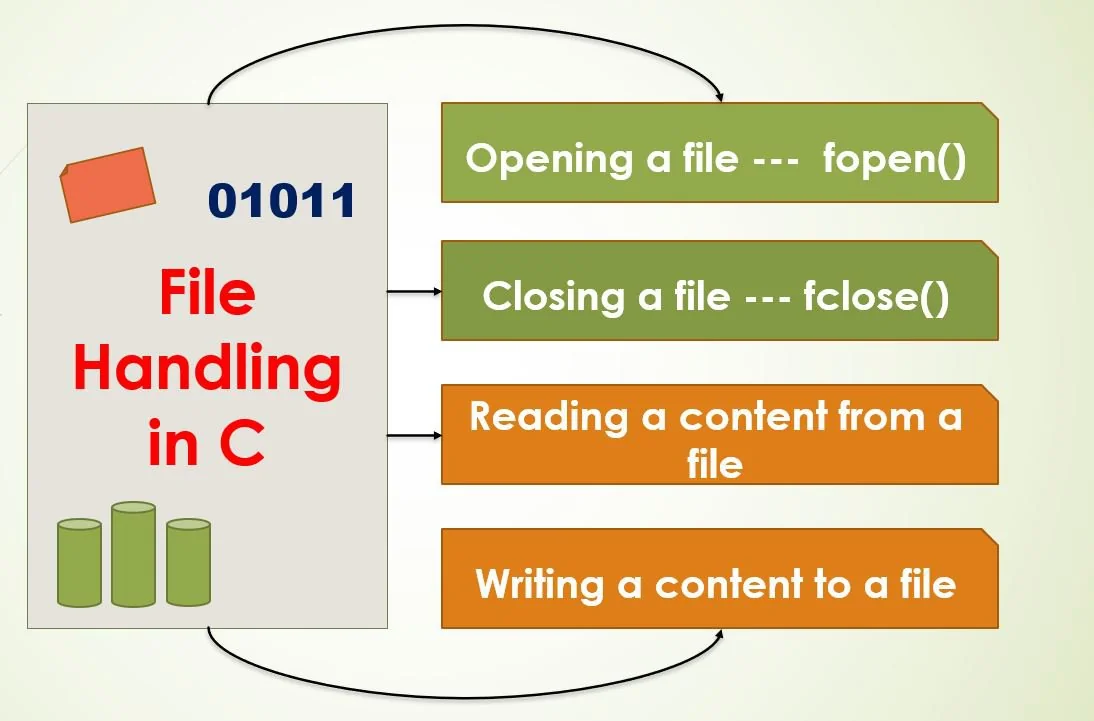
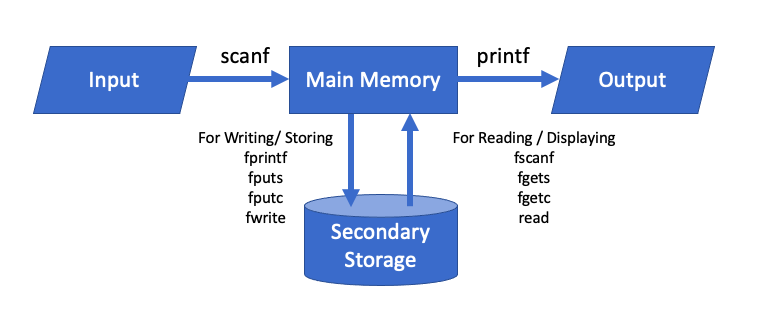


Figure 3.3: File operations

**File Modes**

When opening a file, the mode specifies the type of operations that can be performed. Common file modes include:

* "r": Open for reading. The file must exist.
* "w": Open for writing. If the file exists, its contents are overwritten. If not, a new file is created.
* "a": Open for appending. Data is written at the end of the file.
* "r+": Open for both reading and writing. The file must exist.
* "w+": Open for reading and writing. The file is created or overwritten.

Figure 3.4: Execution Flow

**3.2 Data Structure using C:**

Data structures are essential concepts in computer science that organize and manage data efficiently for various operations. In C programming, data structures are used to store, organize, and manipulate data in ways that optimize performance and resource usage. Understanding these structures is critical to creating efficient algorithms and programs. This article explores the main types of data structures in C, their uses, and their significance in software development.

### 3.2.1. ****Types of Data Structures****

#### 1. ****Primitive Data Structures****

Primitive data structures are the most basic types of data structures. These are the building blocks for creating other complex structures. In C, the most common primitive data types include:

* **Integer**: Stores whole numbers (positive or negative).
* **Float**: Stores numbers with fractional parts.
* **Character**: Stores single characters.
* **Double**: Stores double-precision floating-point numbers.
* **Boolean**: Represents true or false values.

#### 2. ****Linear Data Structures****

Linear data structures are those where data elements are arranged in a sequential manner. Each element has a unique predecessor and successor, making these structures ideal for situations where the order of elements is important.

* **Array**: An array is a collection of elements of the same data type stored in contiguous memory locations. It allows random access to elements using indices. Arrays are simple and efficient, but they have a fixed size, which can be a limitation in dynamic scenarios.
* **Linked List**: Unlike arrays, linked lists are collections of elements called nodes, where each node contains data and a pointer to the next node in the sequence. This structure allows dynamic memory allocation and is useful for scenarios where the size of the data set is not known in advance. Linked lists are categorized into:
  + **Singly Linked List**: Each node points to the next node.
  + **Doubly Linked List**: Each node contains two pointers, one pointing to the next node and another pointing to the previous node.
  + **Circular Linked List**: The last node points back to the first node, making the list circular.
* **Stack**: A stack follows the Last In First Out (LIFO) principle, meaning the last element added to the stack is the first one to be removed. Stacks are used in function calls, expression evaluation, and backtracking algorithms.
* **Queue**: A queue follows the First In First Out (FIFO) principle, where the first element added to the queue is the first one to be removed. Queues are useful in scenarios like scheduling tasks, managing requests in web servers, and handling interrupts.

#### 3. ****Non-Linear Data Structures****

Non-linear data structures allow elements to be connected in a hierarchical or interconnected manner, rather than a sequence. These structures are more complex and often used in cases where relationships between data points need to be represented.

* **Tree**: A tree is a hierarchical structure consisting of nodes, where each node has a value and a list of references (children) to other nodes. The top node is called the root, and nodes with no children are called leaves. Trees are widely used in representing hierarchical data, such as file systems, organizational structures, and databases. Some common types of trees include:
  + **Binary Tree**: Each node has at most two children.
  + **Binary Search Tree (BST)**: A binary tree where each node’s left subtree contains values less than the node, and the right subtree contains values greater than the node.
  + **AVL Tree**: A self-balancing binary search tree.
  + **Heap**: A complete binary tree used to implement priority queues.
* **Graph**: A graph consists of nodes (vertices) connected by edges. Graphs can represent many complex relationships, such as social networks, road maps, and internet routing. Graphs can be:
  + **Directed Graph (Digraph)**: Where edges have a direction, pointing from one node to another.
  + **Undirected Graph**: Where edges do not have a direction, meaning the relationship between nodes is bidirectional.
  + **Weighted Graph**: Where edges have weights representing the cost of traversing between nodes.

Graphs are fundamental in algorithms like shortest pathfinding, network flow, and web page ranking (PageRank).

#### 4. ****Hash-Based Data Structures****

Hash-based data structures rely on hash functions to map data elements to specific locations in an array. These structures are efficient and widely used in computer science for tasks requiring fast data retrieval. The most common example of a hash-based data structure is the hash table. A hash table organizes and stores data in the form of key-value pairs, where the key is passed through a hash function to determine the index or location within the underlying array where the corresponding value will be stored.

The efficiency of hash tables lies in their ability to perform operations like searching, insertion, and deletion with average time complexity of O(1), making them an ideal choice for applications requiring quick access to data. Collisions, situations where multiple keys hash to the same location, are resolved using techniques like chaining (linked lists) or open addressing (probing). These strategies ensure that the hash table maintains its performance characteristics even under non-ideal conditions.

Hash tables are widely used in various domains, including database indexing, caching, symbol tables in compilers, and implementing sets and dictionaries in programming. Their versatility and speed make them one of the most important and foundational data structures in computing.

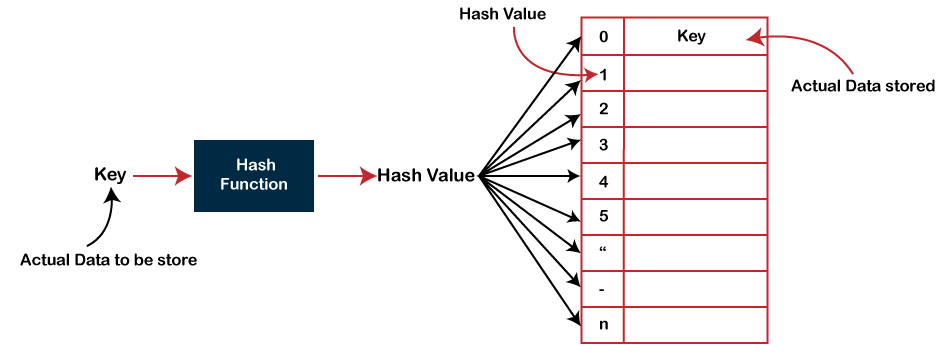


Figure 3.5: Hash Function in Data Structure

**3.3 Python for Data Science**

Python has become one of the most popular programming languages for data science due to its simplicity, flexibility, and extensive ecosystem of libraries and frameworks. It is widely used in various domains such as data analysis, machine learning, artificial intelligence, and scientific computing. This article explores the reasons behind Python's success in data science, its key features, and how it facilitates data-driven decision-making and innovation in various industries.

### 3.3.1. ****Python's Role in Data Science****

Data science involves extracting meaningful insights from large volumes of structured or unstructured data. Python plays a crucial role in each step of the data science process, including data collection, cleaning, exploration, visualization, modeling, and deployment.

#### 1. ****Data Collection and Importing****

Python provides various libraries to connect to databases, APIs, and other data sources to collect data. Common tools include:



#### Figure 3.6: Data Collection

#### 2. ****Data Cleaning and Preprocessing****

Data cleaning is one of the most important and time-consuming tasks in data science. Raw data often contains inconsistencies, missing values, outliers, and irrelevant features that need to be addressed before further analysis. Python provides a range of tools for data preprocessing, such as:

* **Pandas**: Offers methods to clean and preprocess data, like handling missing values (NaN), filtering outliers, transforming columns, and merging datasets.
* **NumPy**: A library that offers array manipulation and mathematical functions for numerical data. It is used for handling missing or invalid values and performing operations on large datasets.
* **Scikit-learn**: A machine learning library that includes tools for feature scaling, encoding categorical variables, and splitting datasets into training and test sets.

By using these libraries, data scientists can clean and transform raw data into a format suitable for analysis and modeling.

#### 3. ****Data Exploration and Visualization****

#### Figure 3.7: Data Visualization

#### 5. ****Model Evaluation and Tuning****

After training a machine learning model, evaluating its performance is crucial to ensure that it generalizes well to new, unseen data. Python provides several tools for evaluating and tuning models:

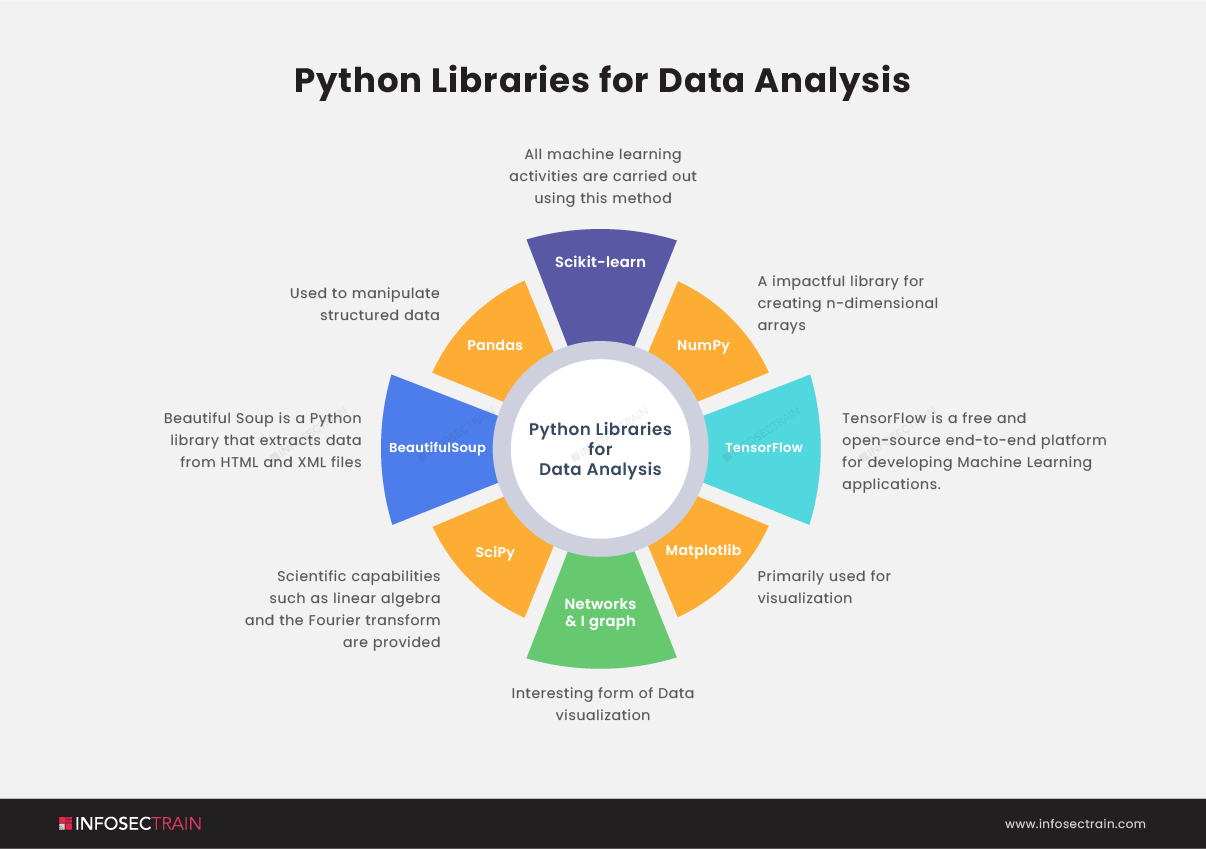
* **Scikit-learn**: Offers metrics such as accuracy, precision, recall, F1 score, confusion matrix, and ROC curve to assess model performance.
* **Cross-validation**: A technique that helps assess how the model will perform on different datasets, minimizing the risk of overfitting.

By evaluating models and adjusting their parameters, data scientists ensure that their models perform optimally and are ready for deployment.

#### 6. ****Deployment and Automation****

Once the model has been trained and evaluated, the next step is deploying it into a production environment where it can provide real-time predictions. Python offers several tools and frameworks for deployment and automation:

* **Flask and Django**: Lightweight web frameworks for building APIs that can serve machine learning models in production.
* **FastAPI**: A modern web framework for building high-performance APIs with Python, often used to deploy machine learning models as part of a microservice architecture.

****These tools help data scientists deploy models and automate workflows, enabling businesses to integrate machine learning into their operations and make data-driven decisions in real-time.

**F**igure 3.8: Python Libraries for Data Science

**3.4 React in Web Development:**

React is one of the most popular and widely used JavaScript libraries for building user interfaces (UIs) in modern web development. Created and maintained by Facebook, React allows developers to create fast, dynamic, and scalable single-page applications (SPAs) with ease. Its component-based architecture and efficient rendering system have made it a go-to choice for developers across the globe. In this article, we will dive deeper into React’s key features, benefits, and its growing importance in the field of web development.

### 3.4.1. ****What is React?****

React is a declarative, efficient, and flexible JavaScript library that focuses on building user interfaces. It allows developers to break down complex UIs into smaller, reusable components. React’s core principle revolves around the idea of building applications using components that manage their own state and interact with each other through props. React is particularly known for its use in creating single-page applications (SPAs), where the UI dynamically updates without the need for reloading the entire page.

### 3.4.2. ****Key Features of React****

#### ****1. Component-Based Architecture****

One of the central concepts in React is the **component-based architecture**. In React, everything is a component, from small UI elements like buttons and input fields to entire sections of a webpage. A component is a self-contained unit of code that defines both the structure and behaviour of a part of the UI.

#### 

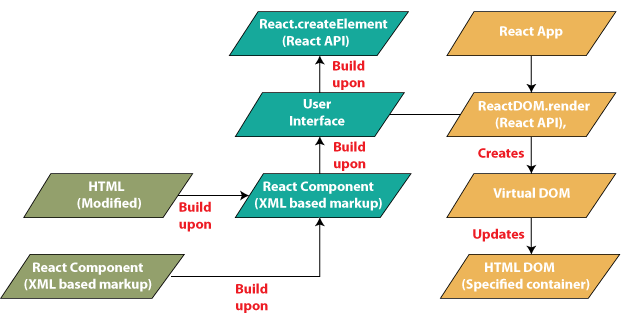
#### **Figure 3.9: React Structure**

#### ****2. JSX (JavaScript XML)****

React uses **JSX** (JavaScript XML), which is a syntax extension for JavaScript. JSX allows developers to write HTML-like code directly within JavaScript. This makes the code more readable and intuitive by combining HTML structure with JavaScript functionality in one place.

Here is an example of JSX: const element = <h1>Hello, World!</h1>;

#### ****3. Virtual DOM****

One of React’s most significant innovations is the **virtual DOM**. The Virtual DOM is a lightweight in-memory representation of the actual DOM (Document Object Model). React uses the Virtual DOM to optimize the process of updating the actual DOM.

### Figure 3.10: Flow Diagram

### 3.4.3. ****Benefits of Using React****

#### ****1. Performance Optimization****

React’s Virtual DOM is a game-changer in terms of performance. By only updating the parts of the DOM that have changed, React minimizes the number of expensive DOM manipulations. This makes React applications highly performant, especially when compared to traditional JavaScript methods, where each change to the UI requires a full re-render of the DOM.

#### ****2. Reusability and Maintainability****

#### React’s component-based architecture encourages the creation of reusable components, which can be shared across different parts of an application. This promotes ****code reusability**** and ****modularity****, reducing duplication and making the codebase more maintainable.

#### **Figure 3.11: File Structure in React**

### 3.4.4. ****React's Role in Modern Web Development****

React has become a fundamental technology for building modern web applications. It is especially suited for creating dynamic, responsive, and interactive user interfaces. React is widely used in building **single-page applications (SPAs)**, where the content is updated dynamically without requiring full-page reloads.

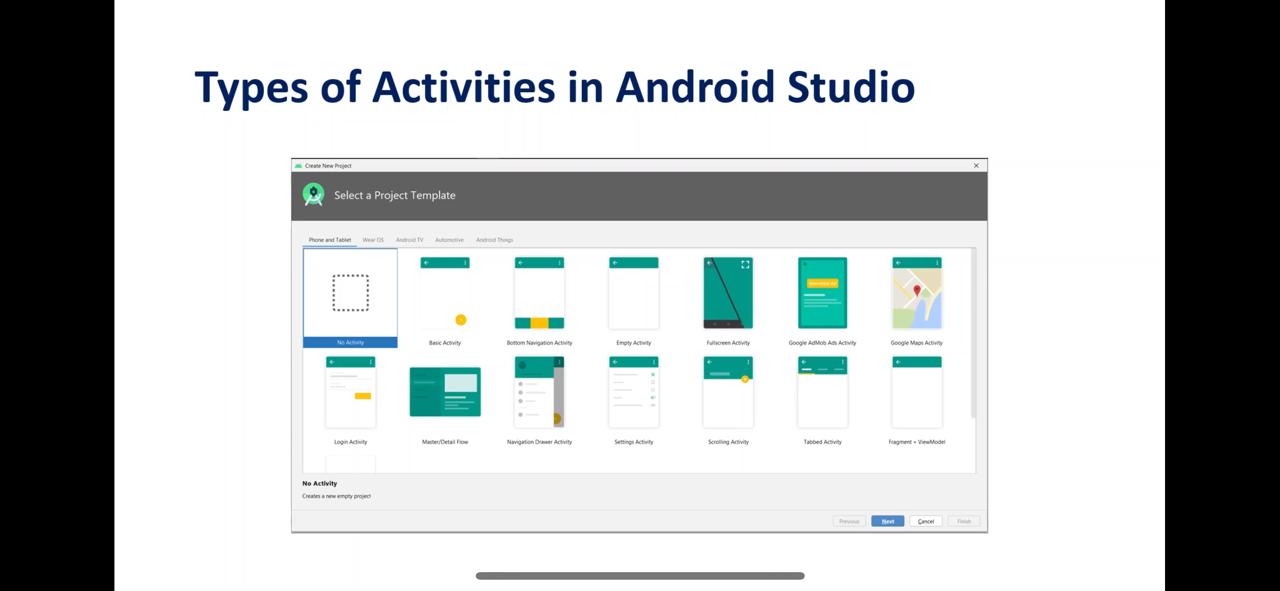
React also integrates seamlessly with backend technologies, enabling developers to build end-to-end solutions. It can easily work with APIs (like REST or GraphQL) to fetch and update data, allowing developers to create data-driven applications.

In the context of modern web development, React is often paired with other tools like **Node.js** for backend development, **GraphQL** for efficient data querying, and **Webpack** for bundling and optimizing assets.

**3.5 Kotlin in App Development:**

Kotlin has emerged as a powerful programming language for app development, particularly in the Android ecosystem. Developed by JetBrains and officially supported by Google, Kotlin addresses many limitations of Java, offering a modern, concise, and safe approach to building applications. Its combination of advanced features, developer-friendly syntax, and interoperability with Java has made it a preferred choice for both new and legacy projects.

#### ****3.5.1. Key Features of Kotlin****

**Null Safety:** Kotlin enhances application reliability by introducing null safety, a feature that minimizes null pointer exceptions (NPEs). It distinguishes between nullable and non-nullable types, ensuring that potential null values are explicitly handled during development. This feature reduces runtime crashes, making applications more robust.

**Figure 3.12: Types of Activities**

**Data Classes and Extension Functions:** Kotlin introduces data classes that automatically generate methods like equals, hashCode, and toString, significantly reducing boilerplate code. Extension functions allow developers to add new functionality to existing classes without modifying their source, fostering modularity and reuse.

**Sealed Classes for State Management:** Sealed classes provide a structured way to represent restricted hierarchies, enabling developers to handle application states effectively. This feature is especially useful for modelling UI states or results from operations like API calls.

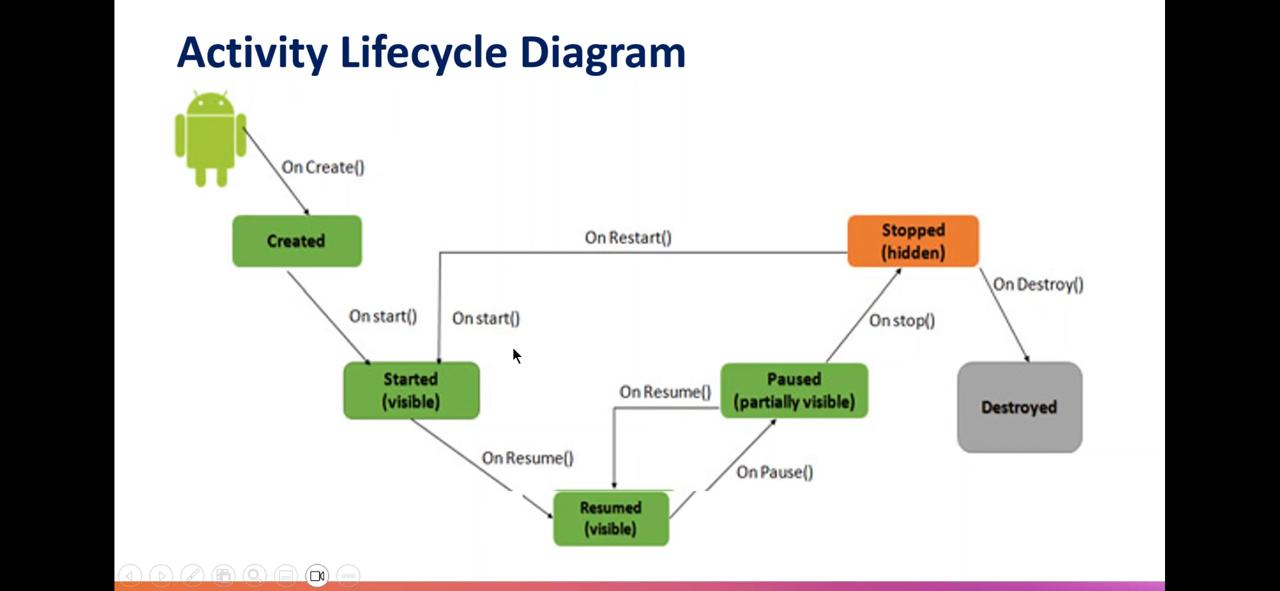
****

Figure 3.13: Activity Lifecycle Diagram

**Kotlin Multiplatform Development:** With Kotlin Multiplatform, developers can share common code across platforms like Android, iOS, and the web while maintaining platform-specific UIs. This reduces development time and ensures consistency across applications.

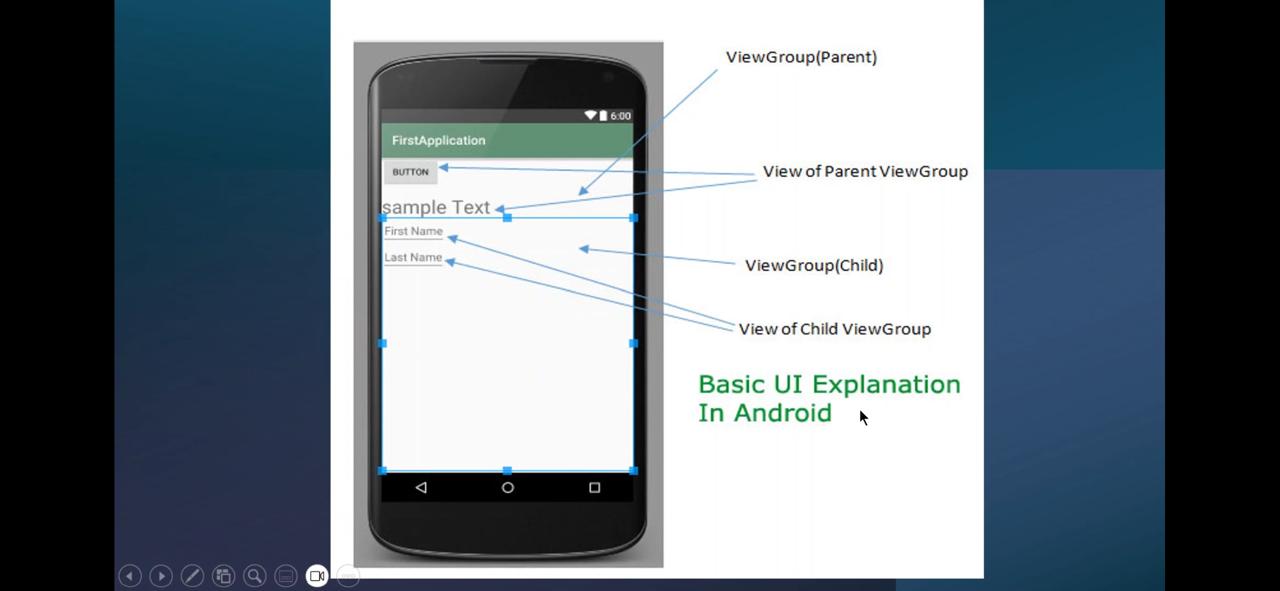


Figure 3.14: Basic UI of Android

**Chapter 4**

**IMPLEMENTATION**

his document provides a step-by-step implementation guide for building the "Interactive Chart Generator with Manual Data Input" project. The project is developed using React.js for the frontend and Python Django for the backend. It allows users to input data manually, generate charts dynamically, and download them as image files. Here’s a detailed breakdown of the implementation:

**4.1. Frontend Implementation**

**4.1.1 Setting Up the React Application**

1. Initialize a new React project:

npx create-react-app chart-generator

cd chart-generator

1. Install necessary dependencies:

npm install axios

1. Structure the project directory as follows:

src/

|-- components/

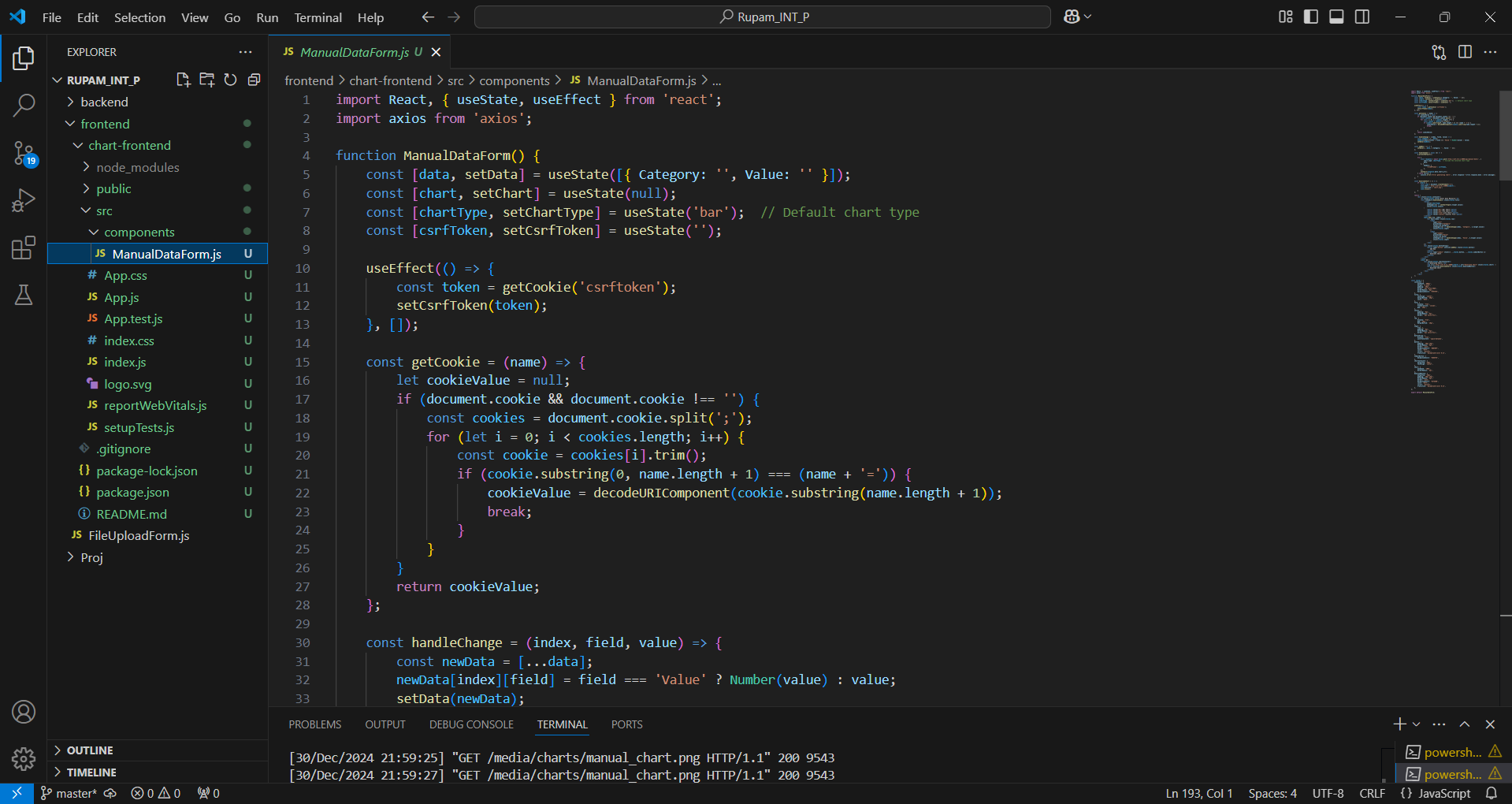
| |-- ManualDataForm.js

|-- App.js

|-- index.js

**4.1.2 Creating the ManualDataForm Component**

The ManualDataForm component manages user inputs, communicates with the backend API, and displays the generated chart.

Figure 4.1: ManualDataForm

**4.1.3 Including the Components Inside the App**

Now include the component named ManualDataForm inside the App.js file.

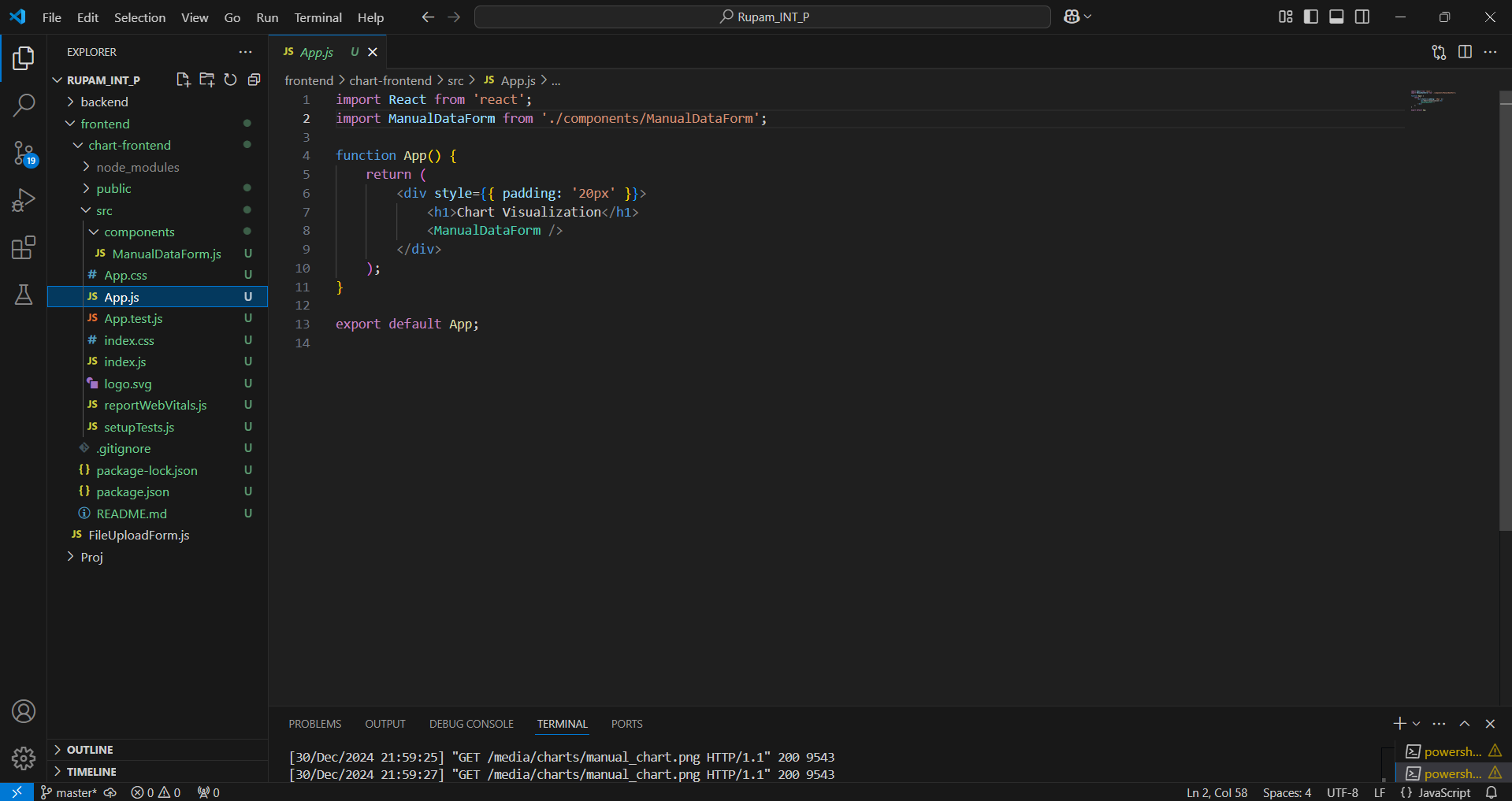
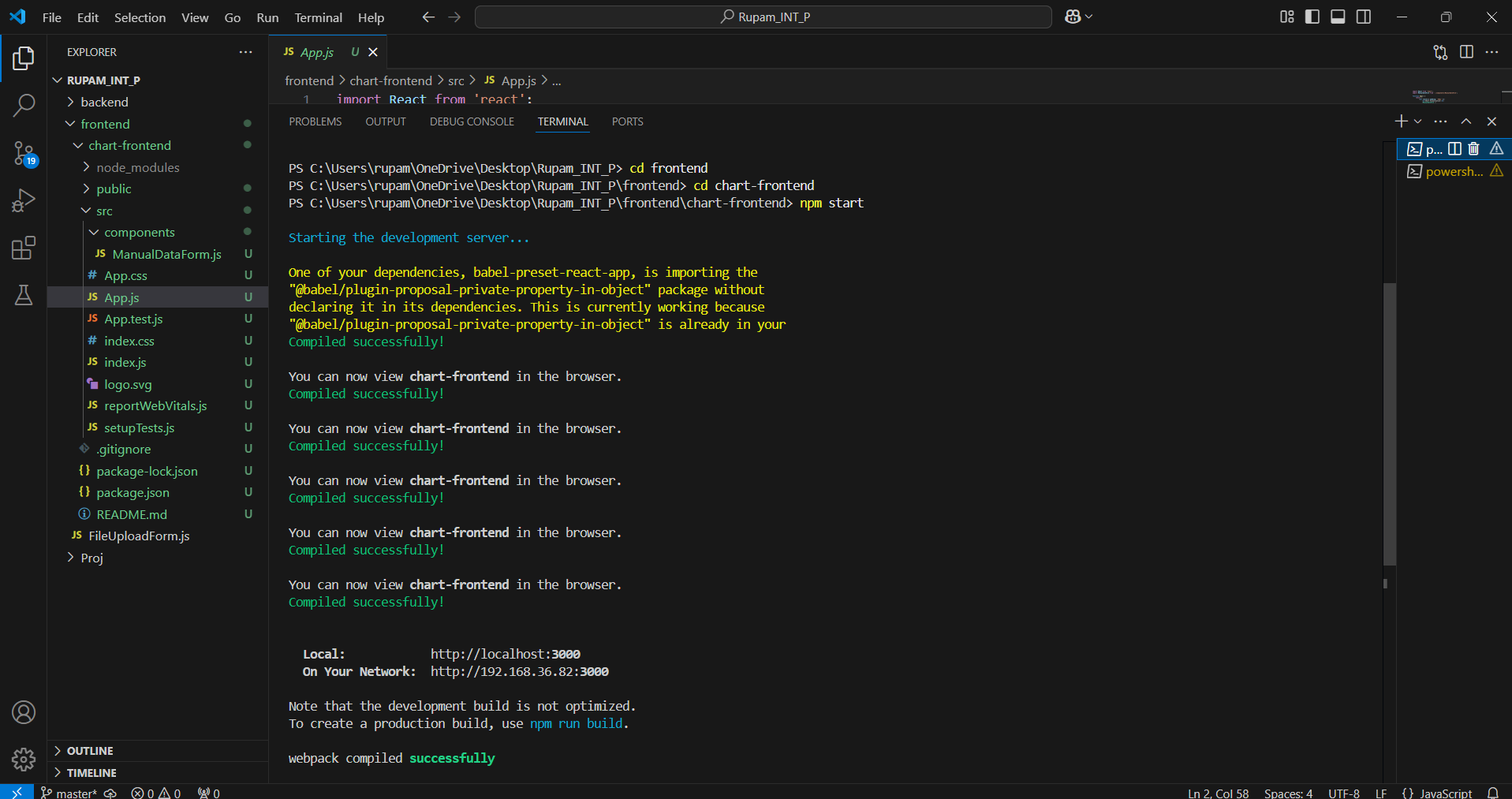


Figure 4.2: App.js file

**4.1.4 Run the React Server:**

****Now run the react app in the localhost by executing the following command inside the terminal:

npm start

Figure 4.3: Run the Server in Localhost

**4.2 Backend Implementation**

**4.2.1 Setting Up Django Backend**

1. Create a Django project:

django-admin startproject chart\_backend

cd chart\_backend

1. Create a Django app:

python manage.py startapp api

1. Install necessary libraries:

pip install matplotlib django-cors-headers

1. Add api and corsheaders to INSTALLED\_APPS in settings.py.
2. Configure CORS:

INSTALLED\_APPS = [

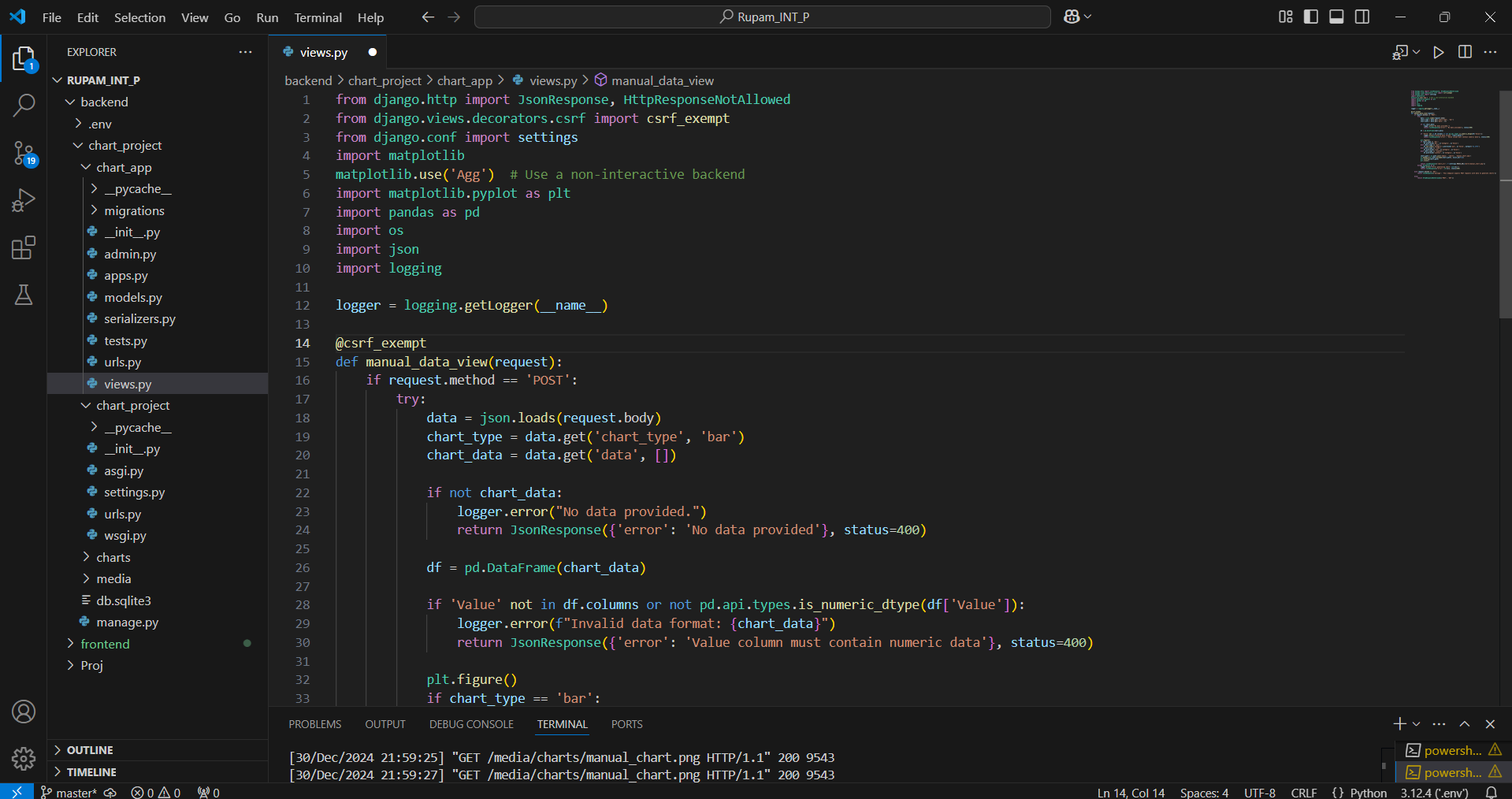
...

'corsheaders',

‘api’, ]

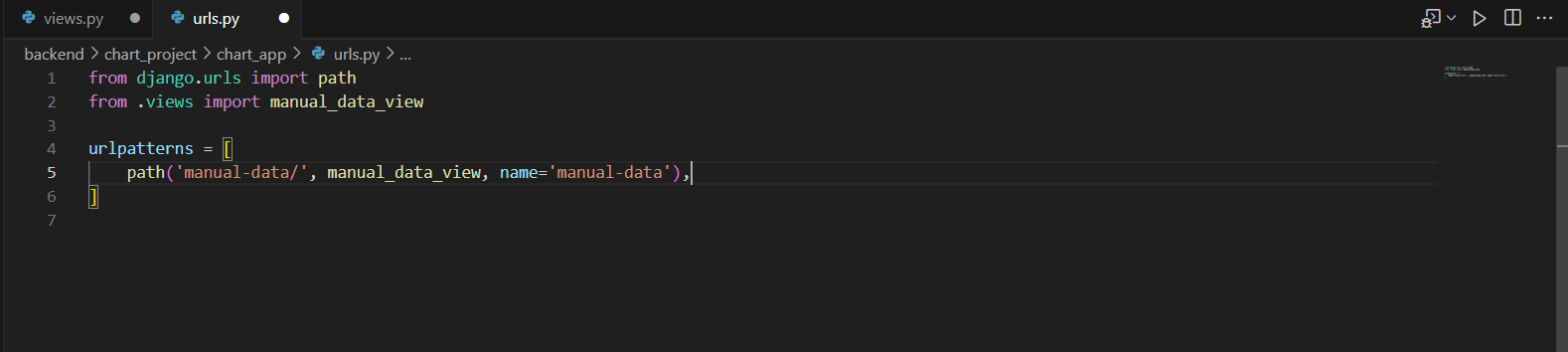
**4.2.2 Implementing API Endpoints**

Create a view in api/views.py to handle data and generate charts:

Figure 4.4: Views.py

**4.2.3 Setting Up URLs**

**In App URL (api/urls.py):**

Figure 4.5: App Url

**In Project URL (chart\_backend/urls.py):**

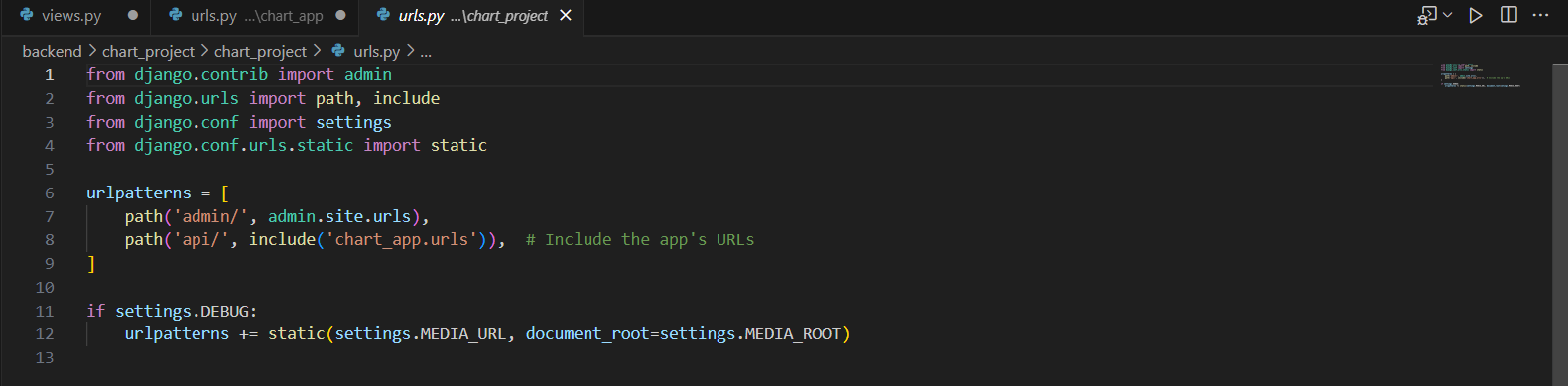


Figure 4.6: Project Url

**4.2.4 Run the Django server:**

python manage.py runserver

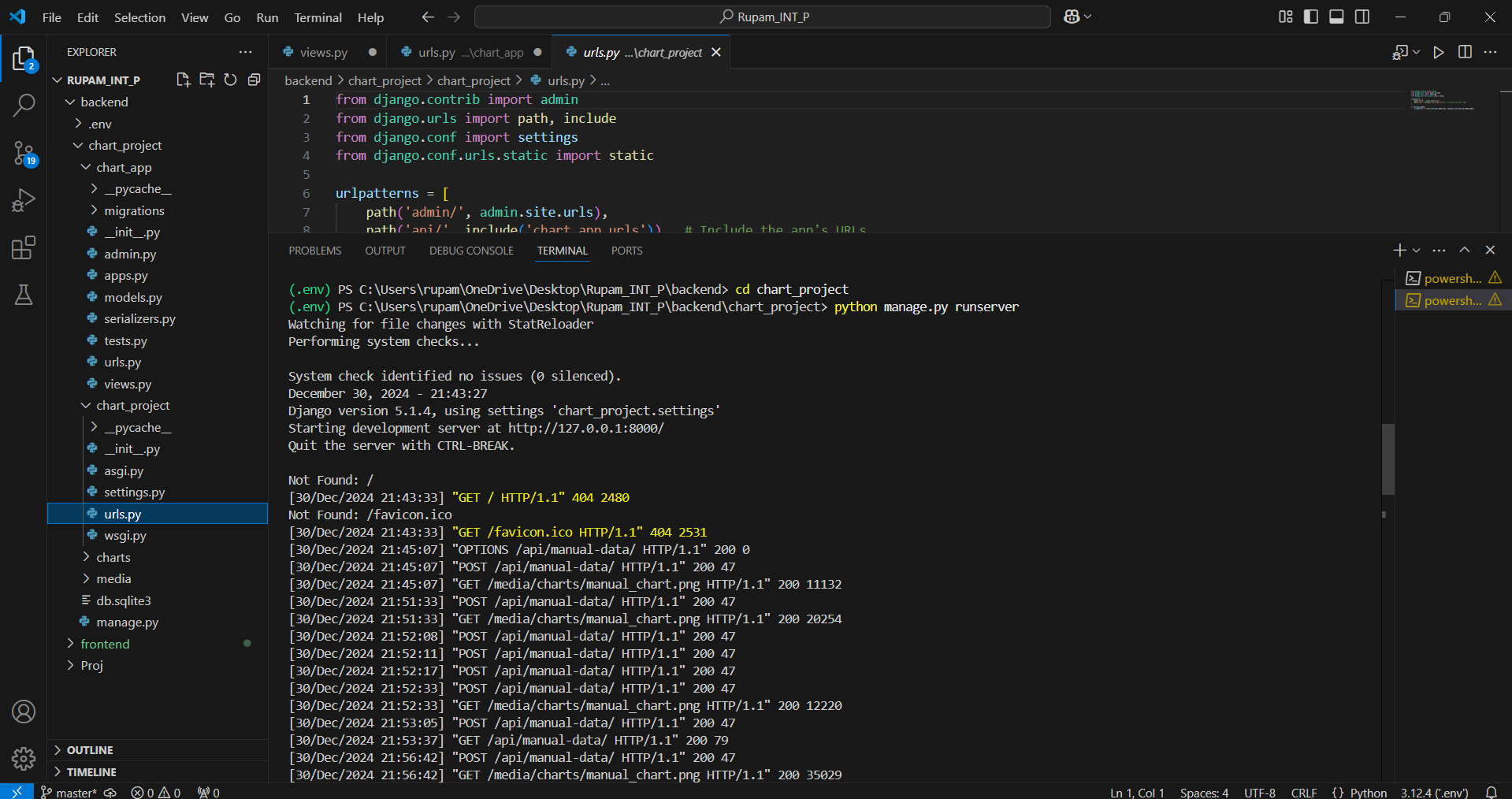
****

Figure 4.7: Run the Django Server

**Chapter 5**

**RESULT**

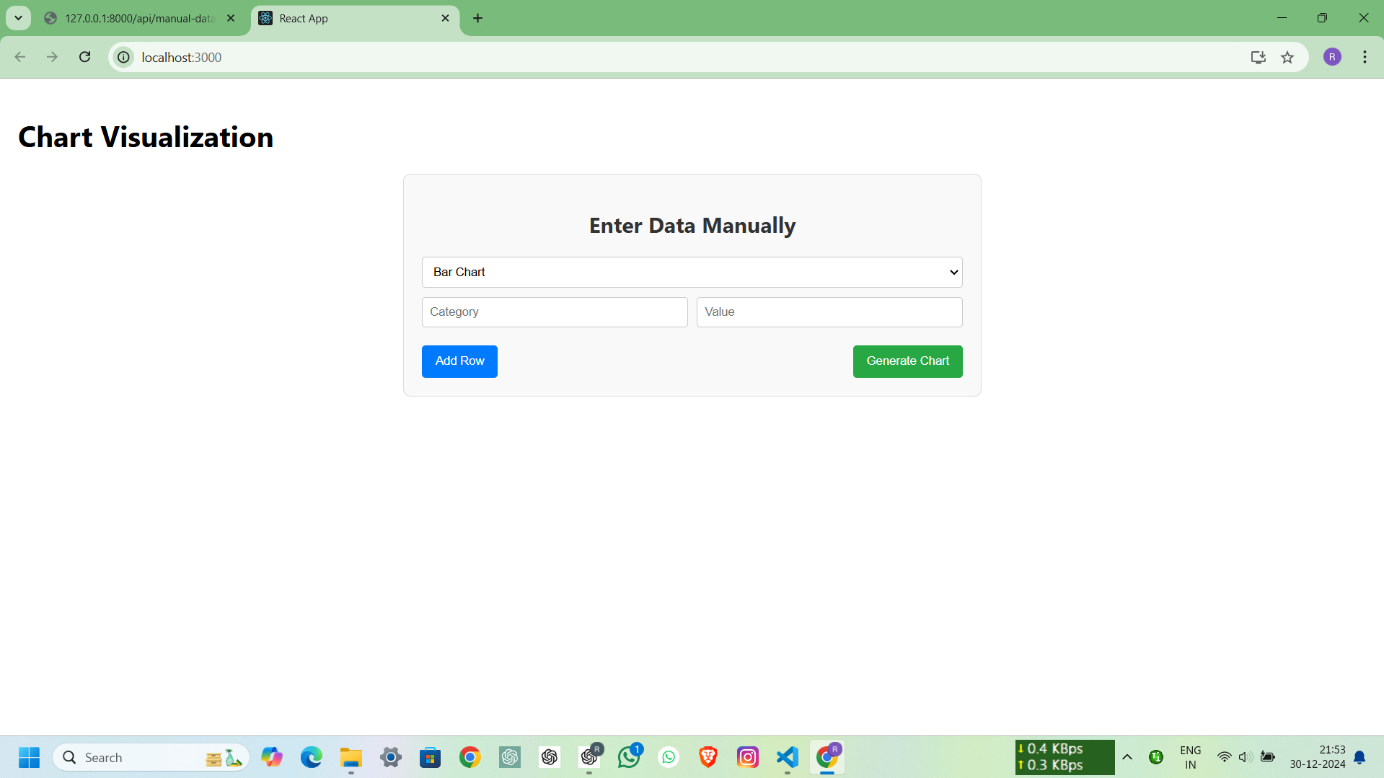
****

Figure 5.1: React App Front Page

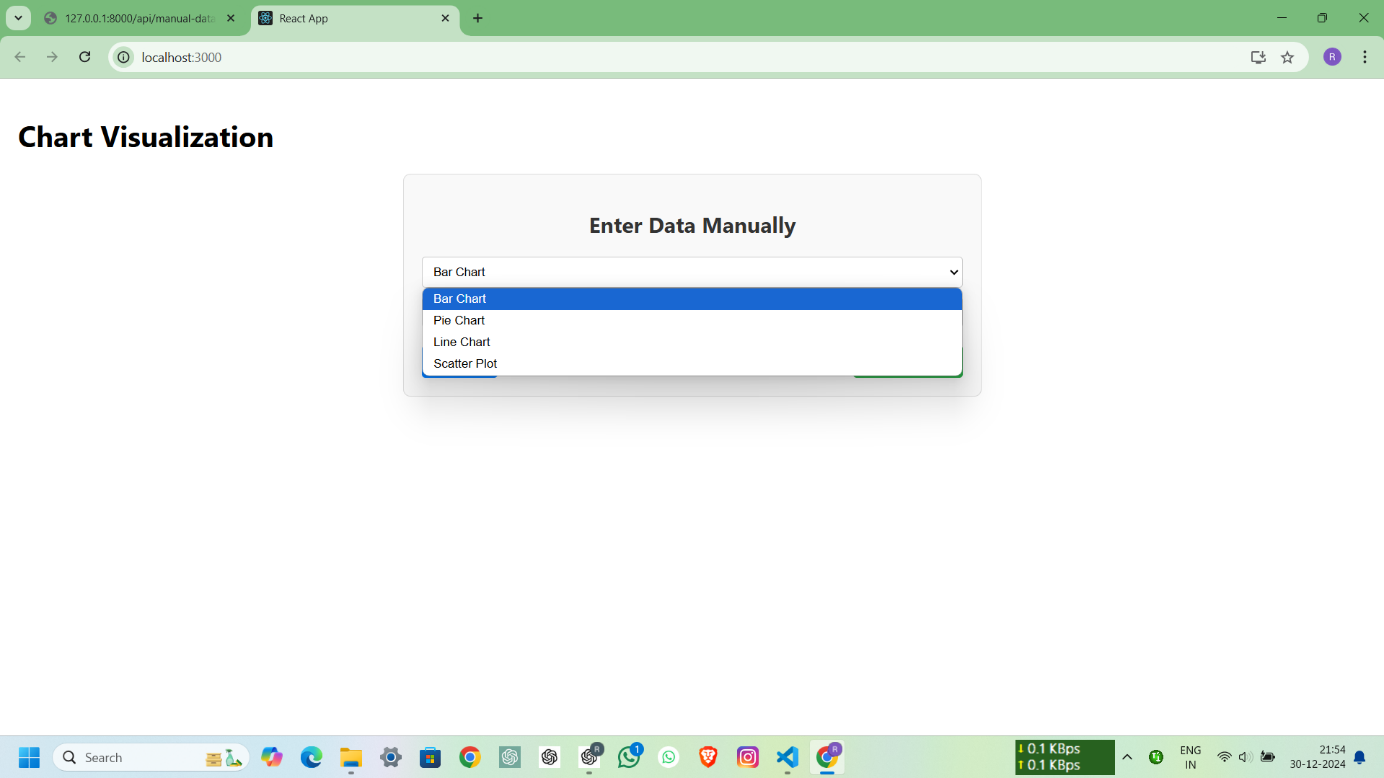


Figure 5.2: List of Charts

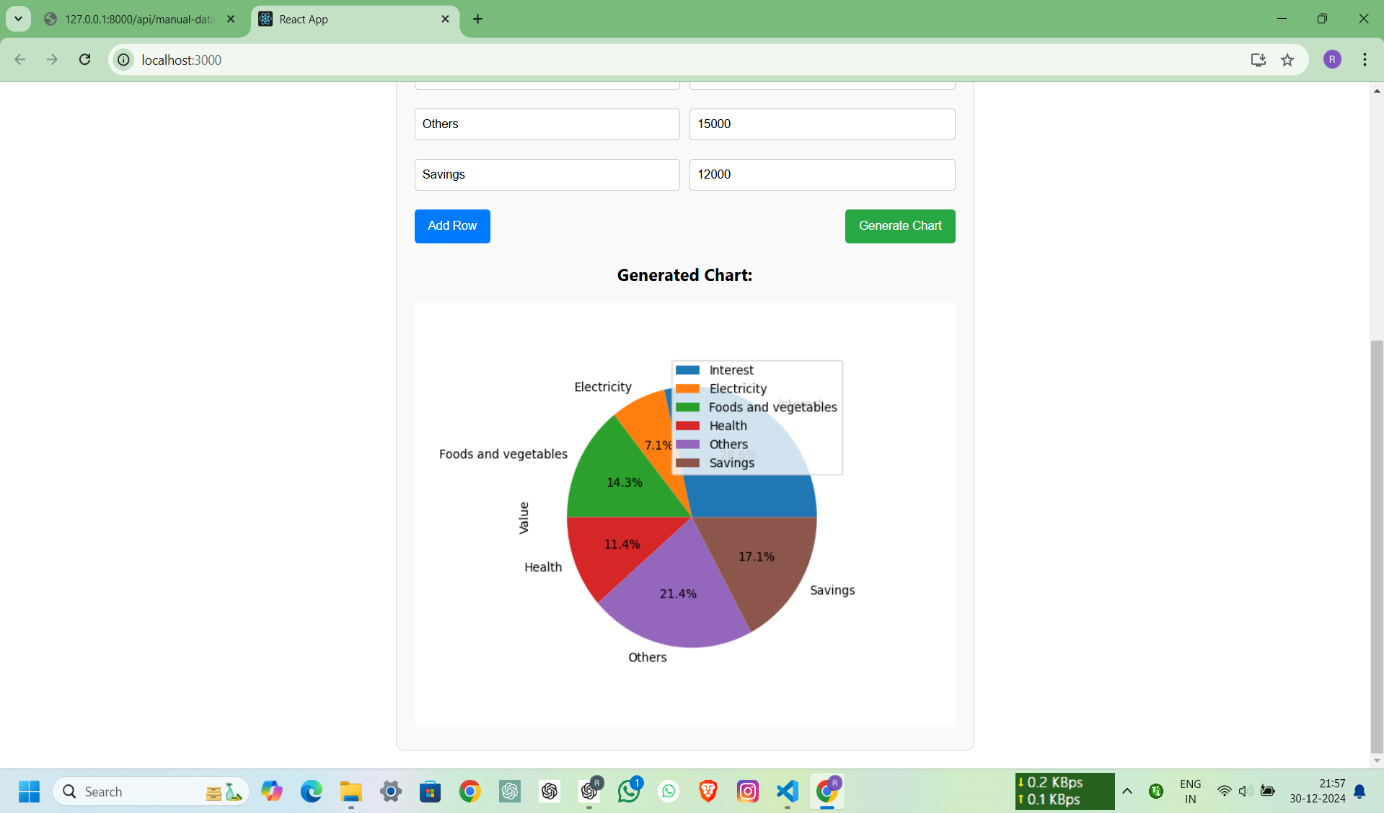
****

Figure 5.3: Pie Chart with Given Data

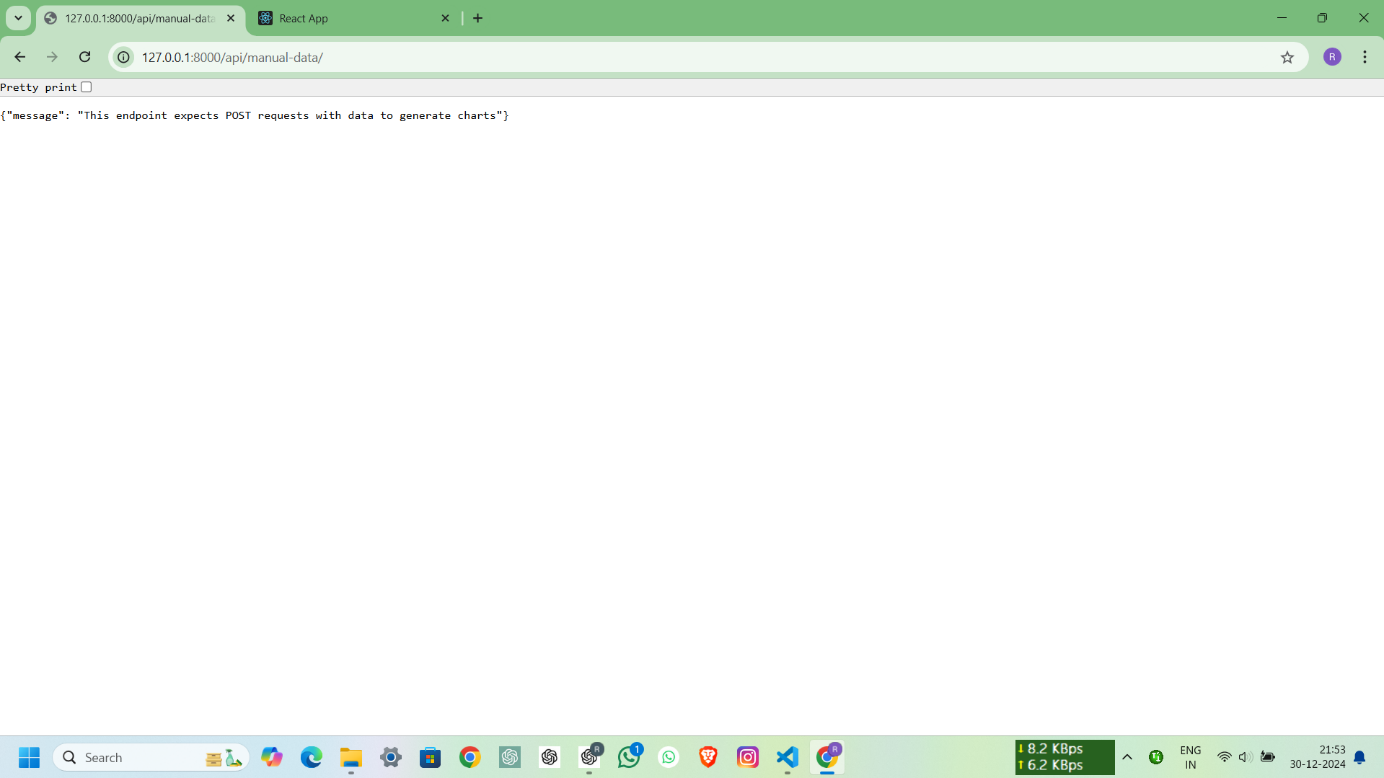


Figure 5.4: Django Backend Server

**Chapter 6**

**CONCLUSION**

The skill enhancement courses introduced by VTU University are a transformative initiative aimed at bridging the gap between theoretical knowledge and practical application. By focusing on modern technologies and essential programming skills, these courses equip students with the tools needed to succeed in today’s competitive job market.

Each course, whether on foundational languages like C or advanced topics such as React, Python libraries for data science, and Kotlin-based app development, is carefully structured to build technical proficiency and foster problem-solving abilities. These programs not only prepare students to meet industry demands but also instil confidence in tackling real-world challenges through hands-on projects and practical assignments.

The emphasis on experiential learning ensures that students are not only job-ready but also capable of innovating and adapting to emerging technologies. Furthermore, the flexibility and accessibility of these courses make them a viable alternative to traditional internships, catering to a diverse range of learners.

In summary, the skill enhancement courses reflect VTU’s commitment to academic excellence and career-oriented education. By investing in such initiatives, the university empowers its students to excel in their careers and contribute meaningfully to the technological advancements of the future.

This implementation of the project provides a comprehensive solution for creating, visualizing, and downloading charts based on user input. Further enhancements, such as user authentication and additional chart customization options, can be added for scalability.

**REFERENCES**

1. **VTU Online Skill Enhancement Course.**
   1. URL: https://online.vtu.ac.in/category/courses/Skill-Enhancement-Course
2. **React.js Documentation**
   1. React: A JavaScript library for building user interfaces.
   2. URL: <https://reactjs.org/docs/getting-started.html>
3. **Axios Documentation**
   1. Promise-based HTTP client for the browser and Node.js.
   2. URL: <https://axios-http.com/docs/intro>
4. **Django Documentation**
   1. Django: The web framework for perfectionists with deadlines.
   2. URL: <https://docs.djangoproject.com/en/stable/>
5. **Matplotlib Documentation**
   1. Comprehensive library for creating static, animated, and interactive visualizations in Python.
   2. URL: <https://matplotlib.org/stable/contents.html>
6. **Django REST Framework**
   1. A powerful and flexible toolkit for building Web APIs in Django.
   2. URL: <https://www.django-rest-framework.org/>
7. **CORS Headers Documentation**
   1. Django app for handling Cross-Origin Resource Sharing (CORS).
   2. URL: <https://github.com/adamchainz/django-cors-headers>
8. **CSRF Protection in Django**
   1. Guide to Cross-Site Request Forgery protection in Django.
   2. URL: <https://docs.djangoproject.com/en/stable/ref/csrf/>