Great! Let's dive into Phase 1 of your project plan. Below is a detailed breakdown of the steps, including what you need to do, how to do it, and what the expected outcomes should be.

---

### \*\*Phase 1: Project Setup and Backend Development\*\*

---

### \*\*1.1 Set Up Your Development Environment\*\*

\*\*Objective:\*\* Establish a clean and organized project structure, set up version control, and prepare your Python environment.

#### \*\*Steps:\*\*

1. \*\*Create a Project Directory:\*\*

- \*\*Task:\*\*

- On your local machine, create a new directory for your project. This will be the root folder where all your code, assets, and documentation will reside.

- \*\*Example:\*\*

- Name it something meaningful like `options\_calculator\_project`.

2. \*\*Initialize Git Repository:\*\*

- \*\*Task:\*\*

- Inside your project directory, initialize a Git repository to keep track of your code changes.

- \*\*Command:\*\*

```bash

git init

```

- \*\*Outcome:\*\*

- A `.git` folder is created in your project directory, and you can now track changes using Git.

3. \*\*Set Up a Python Virtual Environment:\*\*

- \*\*Task:\*\*

- Create a virtual environment to manage your project’s dependencies. This ensures that your project is isolated and doesn’t interfere with your system Python or other projects.

- \*\*Command:\*\*

```bash

python3 -m venv venv

```

- Activate the virtual environment:

- \*\*For macOS/Linux:\*\*

```bash

source venv/bin/activate

```

- \*\*For Windows:\*\*

```bash

venv\Scripts\activate

```

4. \*\*Install Necessary Python Packages:\*\*

- \*\*Task:\*\*

- Install Flask (or Django if you prefer), NumPy, SciPy, and any other libraries you’ll need later for options pricing and backend logic.

- \*\*Command:\*\*

```bash

pip install flask numpy scipy

```

- \*\*Outcome:\*\*

- Your virtual environment is now set up with the essential packages.

5. \*\*Create Project Structure:\*\*

- \*\*Task:\*\*

- Organize your project by creating directories for different components like frontend, backend, and assets.

- \*\*Example Structure:\*\*

```

options\_calculator\_project/

├── venv/ # Virtual environment

├── backend/ # Python code for backend

│ ├── app.py # Main Flask application

│ ├── models/ # Option pricing models

├── frontend/ # HTML, CSS, JS files

│ ├── index.html # Main frontend page

│ ├── static/ # Static assets like CSS, JS, images

│ │ ├── css/

│ │ ├── js/

├── README.md # Project documentation

├── .gitignore # Files to ignore in Git

```

6. \*\*Create Initial README and .gitignore:\*\*

- \*\*Task:\*\*

- Write a basic `README.md` to describe your project’s purpose and structure.

- Create a `.gitignore` file to prevent unnecessary files from being committed to your Git repository (e.g., `venv/`, `\_\_pycache\_\_/`).

- \*\*Example .gitignore:\*\*

```

venv/

\_\_pycache\_\_/

\*.pyc

```

7. \*\*First Commit:\*\*

- \*\*Task:\*\*

- Add all your files to Git and make your first commit.

- \*\*Command:\*\*

```bash

git add .

git commit -m "Initial project setup with directory structure and virtual environment"

```

#### \*\*Outcome:\*\*

- Your development environment is fully set up, with a well-organized project structure, version control initialized, and a Python virtual environment ready to go.

---

### \*\*1.2 Choose and Set Up a Web Framework\*\*

\*\*Objective:\*\* Set up a basic Flask (or Django) application that can serve a simple HTML page.

#### \*\*Steps:\*\*

1. \*\*Set Up Flask (or Django):\*\*

- \*\*Flask:\*\*

- \*\*Task:\*\*

- Inside the `backend/` directory, create a file called `app.py`.

- \*\*Basic Flask Application:\*\*

```python

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

return render\_template('index.html')

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

```

- \*\*Outcome:\*\*

- This code creates a simple Flask app that serves `index.html` from your frontend directory.

- \*\*Django (if preferred):\*\*

- \*\*Task:\*\*

- Install Django:

```bash

pip install django

```

- Start a new Django project:

```bash

django-admin startproject options\_calculator\_project

```

- Follow Django’s typical structure and create an app within the project to handle the backend logic.

2. \*\*Configure Template and Static Folder:\*\*

- \*\*Flask:\*\*

- \*\*Task:\*\*

- Update the Flask app configuration to recognize where your frontend files are located.

- \*\*Code Example:\*\*

```python

app = Flask(\_\_name\_\_, template\_folder="../frontend", static\_folder="../frontend/static")

```

3. \*\*Run the Application:\*\*

- \*\*Task:\*\*

- Start your Flask server to ensure everything is working.

- \*\*Command:\*\*

```bash

python backend/app.py

```

- \*\*Outcome:\*\*

- The server should start, and you should be able to navigate to `http://127.0.0.1:5000/` in your browser and see a simple web page (even if it’s just a placeholder for now).

#### \*\*Outcome:\*\*

- A basic Flask (or Django) application is running locally, serving an HTML page.

---

### \*\*1.3 Integrate Plotly.js on the Frontend\*\*

\*\*Objective:\*\* Set up Plotly.js and ensure it works with your basic Flask/Django application.

#### \*\*Steps:\*\*

1. \*\*Download or Link Plotly.js:\*\*

- \*\*Task:\*\*

- You can either download Plotly.js or include it via a CDN link in your `index.html`.

- \*\*Example (CDN Link):\*\*

```html

<script src="https://cdn.plot.ly/plotly-latest.min.js"></script>

```

2. \*\*Create a Basic HTML Page:\*\*

- \*\*Task:\*\*

- In the `frontend/` directory, create an `index.html` file if you haven’t already.

- \*\*Code Example:\*\*

```html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Options Calculator</title>

</head>

<body>

<h1>Options Calculator</h1>

<div id="plotly-chart"></div>

<script src="https://cdn.plot.ly/plotly-latest.min.js"></script>

<script>

var data = [

{

x: [1, 2, 3, 4, 5],

y: [1, 2, 4, 8, 16],

type: 'scatter'

}

];

Plotly.newPlot('plotly-chart', data);

</script>

</body>

</html>

```

3. \*\*Test the Plotly.js Integration:\*\*

- \*\*Task:\*\*

- Start your Flask/Django server and navigate to `http://127.0.0.1:5000/` to check if the Plotly.js chart renders correctly.

- \*\*Outcome:\*\*

- A simple scatter plot should be visible on your webpage, confirming that Plotly.js is correctly integrated.

#### \*\*Outcome:\*\*

- Plotly.js is successfully integrated and rendering a basic chart on your HTML page.

---

### \*\*1.4 First Commit and Review\*\*

\*\*Objective:\*\* Commit all changes and review your progress.

#### \*\*Steps:\*\*

1. \*\*Stage and Commit Changes:\*\*

- \*\*Task:\*\*

- Add all new files and changes to Git and commit them.

- \*\*Command:\*\*

```bash

git add .

git commit -m "Set up Flask backend and integrated Plotly.js on frontend"

```

2. \*\*Review and Document:\*\*

- \*\*Task:\*\*

- Review what you’ve accomplished in this phase.

- Update your `README.md` to include a brief description of the setup process and current status.

3. \*\*Push to Remote Repository:\*\*

- \*\*Task:\*\*

- If you have a remote repository (e.g., on GitHub), push your commits to keep a backup and share your progress.

- \*\*Command:\*\*

```bash

git push origin main

```

#### \*\*Outcome:\*\*

- All changes are committed, documented, and pushed to a remote repository if applicable. The foundation of your project is now set up and ready for further development.

---

### \*\*End of Phase 1\*\*

\*\*Final Outcome:\*\*

- By the end of Phase 1, you have a clean and organized project structure, a running Python backend, and a basic frontend with Plotly.js integrated. You're now ready to start building out the core features of your options calculator in the next phases.

---

This