Step-by-Step Setup Guide – Endless LLC AI-Driven Budgeting Suite

Capstone Project – CSE 485: Endless Moments LLC (AI-Driven Budgeting Suite: Intelligent Forecasting and Dynamic Budget Management)

This guide explains every step required to set up and run the project locally from scratch — ideal for beginners and teammates who have never worked with Python, FastAPI, or React before.

1. System Requirements

Before starting, make sure your system meets the following requirements:

- Operating System: Windows 10/11, macOS, or Ubuntu Linux

- Minimum 8 GB RAM and stable internet connection

- Around 5 GB free disk space

2. Install Required Tools

Tip: These tools only need to be installed once. If you already have them, skip this step.

1. **Python (version 3.10 or higher)**

   - Visit https://www.python.org/downloads/

   - Download and install Python. During installation, make sure to check the box that says "Add Python to PATH".

   - Verify installation by opening Command Prompt or Terminal and typing:

     python --version

2. **Node.js (version 18 or higher)**

   - Go to https://nodejs.org/ and install the LTS version (recommended).

   - Verify installation:

     node -v

3. **Git (for version control)**

   - Download from https://git-scm.com/downloads

   - Verify installation:

     git --version

4. **VS Code (Code Editor)**

- Download from https://code.visualstudio.com/

- Open VS Code and install the following extensions: Python, Prettier, and React Developer Tools.

3. Downloading the Project

1. Obtain the project ZIP file or clone it from GitHub if applicable.

2. Extract the ZIP to an easy-to-access location, such as Desktop or Documents.

3. You should now have a folder structure like this:

```
CapstoneLatest/
│
├── Budgeting capstone Fall 2025/
│   ├── fpna_backend_final/
│   ├── Front end code/
│   └── database/
│
└── README.md
```

4. Backend Setup (FastAPI)

Info: The backend is the 'brain' of your app. It handles data, logic, and communication with the database.

1. Open VS Code and click **File → Open Folder** → select `fpna_backend_final`.

2. Open a new terminal in VS Code (**View → Terminal**) and run:

```
python -m venv venv

venv\Scripts\activate     (on Windows)

source venv/bin/activate   (on macOS/Linux)
```

Warning: If the terminal says 'python not recognized', restart VS Code or reinstall Python with the 'Add to PATH' option checked.

3. Install dependencies:

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

4. Start the backend server:

uvicorn app_super_v3:api --reload

5. Open your browser and visit http://127.0.0.1:8000/docs to confirm it's running. You should see the Swagger API page.

5. Frontend Setup (React + Vite)

Info: The frontend is what users interact with — the dashboard and interface.

1. Open a new terminal in VS Code and navigate to the frontend folder:

cd "../Front end code"

2. Install dependencies:

npm install

3. Start the frontend development server:

npm run dev

4. When it starts, note the address shown (usually http://localhost:5173). Open it in your browser.

Warning: If you get an error like 'npm not recognized', reinstall Node.js and restart VS Code.

6. Linking Frontend and Backend

The frontend and backend communicate through an API. This project already includes a proxy setup in `vite.config.js` that handles it automatically.

1. Keep your backend running on port 8000 (FastAPI).

2. Keep your frontend running on port 5173 (Vite).

3. In your web app, find the field labeled **API Base URL** (top of the Headcount page). Enter `/api` and click **Refresh**.

4. The table should populate with data from the backend (like Alice's record).

7. Troubleshooting

Tip: Always read the error message carefully — it usually tells you exactly what's missing.

- If you see 'CORS Error': Make sure your Base URL is '/api', not the full http://127.0.0.1:8000.
- If 'port already in use': Stop any previous server (Ctrl + C) or change port numbers in vite.config.js.

- If Swagger doesn't open: Recheck uvicorn command or ensure your virtual environment is active.
- If the frontend shows blank: Reopen terminal → cd to frontend → npm run dev again.

8. Final Demo Checklist

✅ Backend running successfully at http://127.0.0.1:8000/docs
✅ Frontend running at http://localhost:5173
✅ API Base URL set to /api
✅ Clicking Refresh loads roster data (Alice appears)
✅ No CORS or connection errors

9. Notes

This setup uses the original backend code provided by the client without any changes. The frontend proxy (already configured) ensures CORS-free communication. This setup is cross-platform and verified on Windows 11 and macOS.

🧩 Changes Made for API Proxy Setup (to Make Frontend and Backend Work Together)

When this project was first set up, the frontend (React) and backend (FastAPI) ran on two different ports:

- Frontend → http://localhost:5173

- Backend → http://127.0.0.1:8000

Because of this difference, browsers block communication between them for security reasons (a CORS error). To fix this without modifying backend code, we added a simple proxy configuration in Vite.

🔧 What File Was Changed

The change was made in:

Budgeting capstone Fall 2025/Front end code/vite.config.js

🪄 What Was Added

We added this `proxy` section under the `server` block:

```
server: {
  port: 5173,
  host: '0.0.0.0',
  proxy: {
    '/api': {
      target: 'http://127.0.0.1:8000',
      changeOrigin: true,
      rewrite: (path) => path.replace(/^\/api/, ''),
    },
  },
},
```

🧠 Why This Was Needed

Modern browsers protect users by blocking “cross-origin” requests (known as CORS) when an app tries to access another port or domain.
This setup makes your frontend and backend appear as one single application to the browser, avoiding these CORS issues while keeping the backend untouched.

Info: This change does NOT modify the backend logic. It only changes how the frontend routes its requests.

How to Apply It (Step-by-Step)

1. Open VS Code and navigate to the `Front end code` folder.
2. Locate and open the file named `vite.config.js`.
3. Replace its content with the following code:

```
import { defineConfig } from 'vite'
import react from '@vitejs/plugin-react'

export default defineConfig({
  plugins: [react()],
  server: {
    port: 5173,
    host: '0.0.0.0',
    proxy: {
      '/api': {
        target: 'http://127.0.0.1:8000',
        changeOrigin: true,
        rewrite: (path) => path.replace(/^\/api/, ''),
      },
    },
  },
})
```

4. Save the file and restart the frontend server by running:
   npm run dev
5. In the web app, enter `/api` in the API Base URL box and click Refresh.
6. You should now see the roster table filled with backend data (e.g., Alice in Headcount).

Tip: Always keep backend (Uvicorn) running before starting the frontend. Otherwise, the proxy will have nothing to connect to.

Final Result

After adding the proxy setup:
- Backend code remains unchanged

- No CORS or connection errors
- Frontend and backend communicate seamlessly
- Project runs perfectly with two commands:
  uvicorn app_super_v3:api --reload
  npm run dev