**Week 1 Documentation — Student Campus Navigator (BotBrain)**

Abstract

- BotBrain is an intelligent assistant to help students navigate Chanakya University's campus.
- It treats the campus as a weighted graph and uses classic search algorithms (BFS, DFS, UCS, A*) to find the best routes.
- The interface is user-friendly and shows the shortest path, total distance, and estimated walking time for navigation questions.

Problem Statement

- New and current students often struggle to find the quickest or most efficient routes across the large campus.
- This project tackles that challenge by modeling the campus and offering guided, data-driven route options.

Objectives

- Model the Chanakya University campus as a weighted graph with at least 12 key locations.
- Implement four search algorithms: BFS, DFS, UCS, and A* for pathfinding.
- Build a friendly user interface where users can enter source and destination.
- Show the calculated path, total distance, and estimated walking time.
- Let users pick their start and end points, choose a search algorithm, and see results visually.
- Provide basic building information (open hours, facilities).
- Demonstrate route comparisons and real-time path suggestions.
- Compare how the different algorithms perform in practice.

Scope

- This project focuses on digital navigation of the Chanakya University campus.
- Core functionality includes pathfinding between 12 locations using four algorithms.
- The agent can be used from a text interface, a GUI, or a web app.

Functional Requirements

- Back-End
  - Model the campus as a graph with 12 locations and weighted paths.
  - Implement BFS, DFS, UCS, and A* search algorithms.
  - Calculate the path, total distance, and walking time.
- Graphical User Interface (GUI)
  - Dropdowns to pick start and destination.
  - Option to select a search algorithm.
  - A “Find Path” button to run the search.
  - Display a visual map and highlight the chosen route.
  - Show the path, distance, and estimated time.
  - Let users click on locations to learn more.
- Interaction
  - Users can select source and destination from campus locations.
  - Users can choose among BFS, DFS, UCS, and A*.
  - Provide the path and distance.
  - Show basic facts about buildings (services, timings).
- Interfaces
  - The tool can be used via command line, GUI, or web app.

Non-Functional Requirements

- The tool should be fast, reliable, and easy to use.
- The campus graph should reflect a real, well-justified campus layout.
- The design should be extendable (e.g., visuals, chatbot, database integration).

Data Requirements

- Campus Graph Data: 12 buildings, defined connections (edges) with distances in meters, and coordinates for each location to support heuristics.

- Building Information: Short descriptions, services, and schedules.
- Overall, an accurate map of walking paths and building details.

Tools and Technologies

- Language: Python (for the agent, algorithms, and backend).
- UI options: PySimpleGUI, Tkinter, or PyQt/PySide if available.
- Real-world maps: Google Maps API or OpenStreetMap for satellite views or overlays, where allowed.
- Documentation/Viz: MS Word, PowerPoint for artifacts.
- Source control: GitHub.

Proposed Deliverables for Week 1

- A formal requirements specification (functional and non-functional).
- A preliminary campus graph model (12+ buildings) with nodes and edges annotated by distance/time.
- A plan to compare BFS, DFS, UCS, and A* (path length, nodes expanded, runtime).
- A UI/UX sketch or mockups (command-line, GUI, and/or web app).
- A data dictionary for building information (services, timings, descriptions).
- A bibliography of references and prior art.

Optional Artifacts (If Requested)

- Draft data dictionary for buildings (fields like id, name, coordinates, services, hours, description).
- Sample graph schema (JSON) with 12+ nodes and representative edge weights.
- Week 1 progress report template in Word or PDF.