# **BotBrain: Campus Navigation System**

**Abstract** Modern university campuses are complex environments with numerous buildings, facilities, and interconnected pathways.Boasting a 126-acre campus, Chankaya University itself is rapidly expanding its campus and this often causes students,especially newcomers, often struggle to navigate efficiently between different locations, leading to time wastage and missed appointments. Traditional static maps fail to provide dynamic routing solutions or comparative analysis of different path options.

This project focuses on developing "BotBrain," an intelligent campus navigator that leverages fundamental AI search algorithms to provide efficient pathfinding solutions for Chanakya University campus. The system demonstrates practical implementation of search strategies learned in artificial intelligence coursework while solving a genuine problem faced by the student community.

## **I. Problem Statement**

Students at Chanakya University face several navigation-related challenges:

* Inefficient Route Planning: Students often take longer routes between locations due to unfamiliarity with optimal paths
* This is especially because a large chunk of the campus is still under construction.
* Lack of Dynamic Navigation: Static campus maps don't provide real-time route optimization or alternative path suggestions
* Limited Campus Information Access: Students struggle to find information about building services, operating hours, and facility availability
* Time Management Issues: Poor navigation leads to delays and missed classes or appointments

## **II. Scope**

* Minimum 12 campus locations including Academic Blocks (A, B, C), Library, Administration Building, Main Hostel, Canteen, Sports Complex, Medical Center, Main Gate, Student Center, and Auditorium
* Path connections with accurate walking distances and basic constraints (one-way paths, variable speeds) using
* Potentially, in the future, we may explore an upgrade to the system to display real time crowd density

## **III. Tech Requirements Stack**

**A. Frontend (User Interface Layer)**

* React.js – SPA (Single Page Application) framework for interactive chatbot UI.
* HTML5 / CSS3 – structure & styling.
* JavaScript – dynamic functionality.

**B. Backend (Application Layer)**

* Python Flask – lightweight web framework to handle requests & responses.
* RESTful API Endpoints:  
  + navigate → takes source & destination, returns path (BFS/DFS/UCS/A\*).
  + info → fetches building/FAQ details from database.
  + /ompare → runs algorithms & returns performance table.
* Search Algorithms: BFS, DFS, UCS, A\* (implemented in Python).
* NLP-Natural language query interpretation for locations and commands

**C. Google Maps API**

* FOR VISUALISATION ONLY
* Maps JavaScript API → to display routes in React.
* Geocoding API → map building names → coordinates.

**D. DevOps / Deployment**

* GitHub/GitLab → version control & repo hosting.

## **IV. Functional Requirements**

**A. Campus Graph Representation**

* Model the campus with at least 12 buildings (nodes).
* Define weighted paths (edges) with distances in meters.
* Support constraints (e.g., one-way paths, different walking speeds).

**B. Search Algorithm Implementation**

* Implement BFS, DFS, UCS, A\* in Python.
* Each algorithm should return a path, total distance, and estimated walking time.
* Provide step-by-step trace of nodes explored.

**C. User Queries**

* Input source and destination locations (via frontend).
* Select search algorithm to use.
* Display the found path with total distance + time.

**D. Frontend (React UI)**

* Simple form to select source, destination, and algorithm.
* Display result in text (path + distance).
* Show path highlighted on Google Maps API campus visualization.

**E. Backend (Flask API)**

* Expose REST endpoints for:  
  + navigate → Input: source, destination, algorithm → Output: path + distance.
  + compare → Compare algorithms for 2–3 test cases.
* In-memory data only (graph + building info stored in Python).

**F. Building Information**

* Store building info (e.g., library hours, canteen menu) in a Python dictionary or JSON file.
* Return info when path is displayed.

**G. Algorithm Comparison**

* Record number of nodes explored by each algorithm.
* Generate a comparison table in frontend or backend response.

**WEEK TWO:**

### **Defined Modules and Features**

1. **User Interface (Chatbot Interface)**
   * Simple web interface built using React.js.
   * Allows students to input queries such as “Find path from Library to Hostel.”
   * Displays path result, distance, and building info.
2. **NLP & Bot Engine (Understanding Queries)**
   * Lightweight query handling in Flask.
   * Matches user input to source and destination buildings.
   * Selects the algorithm to use (BFS/DFS/UCS/A\*).
3. **Navigation / Pathfinding (Google Maps API / Campus Map)**
   * Implements search algorithms (BFS, DFS, UCS, A\*).
   * Works on a graph of campus buildings and paths.
   * Uses Google Maps API for map visualization.
4. **Database (Campus Details, FAQs, Staff Info)**
   * Stores building details such as opening hours and facilities.
   * Contains FAQs for quick responses.
   * Holds staff contact information and campus events.

**For this project, I am NOT using a database as the software will not hold user history. SO, I am switching out the Database Design section with:**

**Data Storage Layer (In-Memory / JSON)**

* Stores campus building info (e.g., opening hours, canteen menu, staff details).
* Paths and distances stored in Python graph structure.
* Easy to extend later to a real database if needed

