SCHOOL OF ENGINEERING

INTRODUCTION TO AIML

ASSIGNMENT-1

BOTBRAIN: CAMPUS NAVIGATOR

(FINAL REPORT)

SUBMITTED BY

THANUSHREE G M

24UG00562

'B'- SECTION

BotBrain: Campus navigator

Abstract

This project introduces "BotBrain," a campus navigating system for Chanakya University. BotBrain maps the campus onto a graph with main buildings as nodes and walkable routes as weighty edges. Using search algorithms such as BFS, DFS, UCS, and A*, BotBrain calculates optimal paths between any two campus points efficiently. Developed using Python and Flask for the backend and HTML/CSS for the frontend, the system offers both text and visual instructions by projecting calculated paths onto an actual aerial campus photo. The web interface, which can be interacted with, enables users to pick starting and ending points, select an algorithm, and immediately see the optimal path, distance, and estimated walking time. Testing verifies the validity and usability of the solution, making BotBrain an effective utility for students and visitors navigating the campus landscape.

Introduction

BotBrain is a campus navigator which guides visitors and new students to move around Chanakya University, finding the most efficient route between buildings can be a significant challenge for students and visitors. Recognizing this need, the BotBrain campus navigator was developed to provide a digital solution that simplifies travel within Chanakya University. By representing the campus as a connected graph and using search algorithms such as BFS, DFS, UCS, and A*, BotBrain transforms campus navigation into a fast, accurate, and user-friendly experience. The system's interactive web interface and visual map help users quickly identify optimal paths.

Problem Statement

Navigating a large campus like Chanakya University, which have multiple buildings poses a significant challenge for students, staff, and visitors—particularly newcomers. Manual directions are often inefficient, leading to confusion. This project aims to develop "BotBrain," an intelligent campus navigation system that accurately models the campus as a weighted graph and uses classical search algorithms—BFS, DFS, UCS, and A*—to compute the shortest and most efficient walking routes between any two locations. The system offers an interactive web interface for location selection and algorithm choice, delivering precise route instructions along with visual path overlays on an aerial campus image to simplify navigation and improve user experience.

Objectives

- To digitally map campus buildings and walkable paths as a weighted graph for precise route calculation.

- To implement four search algorithms (BFS, DFS, UCS, A*) allowing flexible and optimal pathfinding options.

- To develop a user-friendly web interface enabling input of start and destination points and selection of preferred algorithms.

- To provide users with detailed textual navigation instructions, walking distance, and estimated travel time.

- To visually display computed routes on a real aerial image of the campus for correct understanding.

- To ensure clarity for future system enhancements such as integration with real-time data and mobile platforms.

Scope

The scope of the BotBrain campus navigator includes providing an accurate and efficient walking route solution for Chanakya University by modeling the campus as a graph with key buildings and paths. It supports multiple search algorithms to find optimal routes, delivers both textual and visual navigation guidance through an interactive web interface, and covers major campus locations. The system is designed for scalability to incorporate features like real walkable paths, dynamic updates, and mobile compatibility in the future. Overall, it aims to assist students, staff, and visitors in navigating the campus with ease while offering a foundation for further smart campus applications.

Literature

Several universities have designed intelligent navigation systems to enable students to navigate more conveniently around campus. The applications usually incorporate digital maps and intelligent pathfinding algorithms to lead the users between buildings. Some further incorporate mobile applications with QR codes. What they all have in common is simplifying navigation and offering helpful information on campus services. Similar to BotBrain, they intend to decrease the stress of being lost in the campus and ease the people to walk around the large campus.

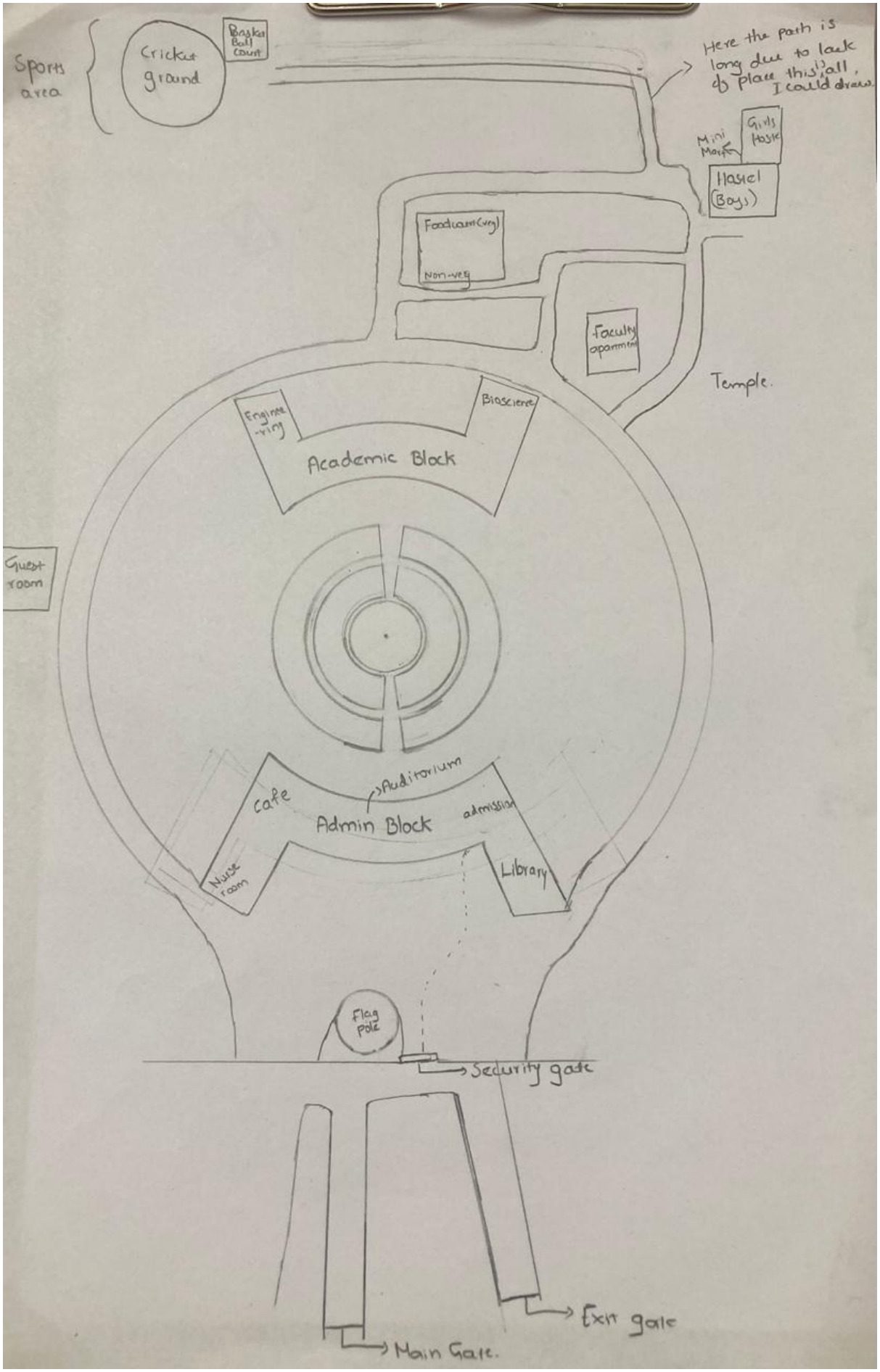
Required documents

1. Campus Entities:

   Here is the list of the key locations/buildings in the campus map for BotBrain

   - Main gate
   - Exit gate
   - Security gate
   - Admin block

- Library
- Medical center
- Cafeteria
- Auditorium
- Academic block
- Food court
- Hostel
- Mart
- Sports area


Sports area
Cricket ground
Basket Ball court
Here the path is long due to lack of place this is all, I could draw.
Mini Mart
Girls Hostel
Hostel (Boys)
Food court (veg)
Non-veg
Faculty apartment
Temple.
Engineering
Bioscience
Academic Block
Guest room
Cafe
Auditorium
Admin Block
admission
Nurse room
Library
Flag pole
Security gate
Exit gate
Main Gate.

2. Physical Connections and distance:
   A dictionary named as campus graph is created which stores the connections between the buildings and the distance between them.
3. Building information:
   This contain some basic information about the buildings. It contains a short description of the building, services available and opening time of the location. These are stored in a dictionary named building info.
4. System Features:

- Models campus as a graph with 12+ key locations and walking distances
- Implements BFS, DFS, UCS, and A* algorithms for flexible pathfinding
- User-friendly web interface for selecting start/destination and algorithm
- Visualizes route on aerial campus image with clear path overlay
- Provides distance and estimated walking time for each route
- Modular backend with Flask API for easy integration and extension

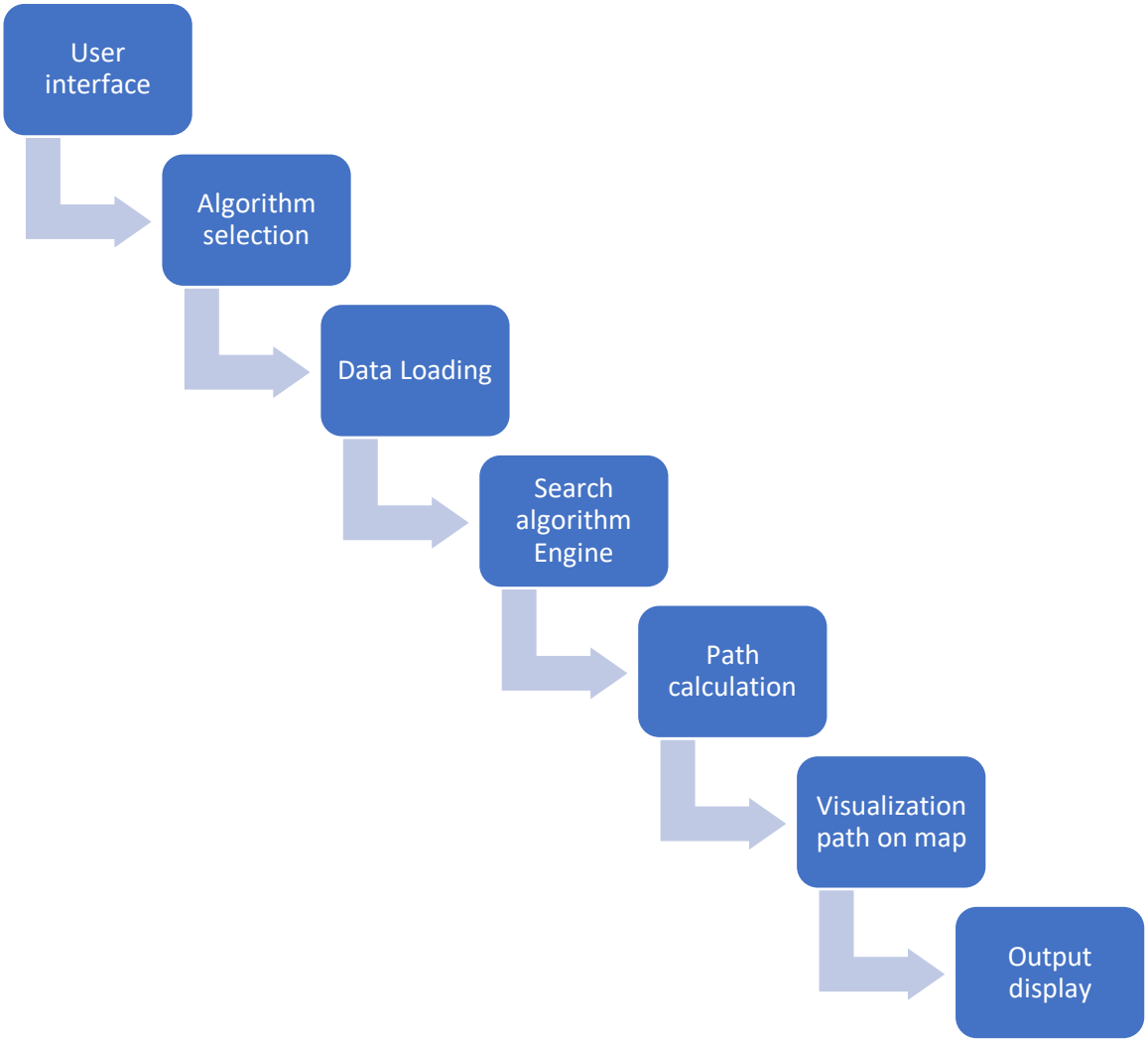
Tools and Technology

The development of the BotBrain campus navigator utilises a combination of programming tools and technologies to build an efficient, interactive, and scalable web-based navigation system for Chanakya University.

- Python: Served as the primary development language, used to implement campus environment modeling and key pathfinding algorithms (BFS, DFS, UCS, and A*). Python's flexibility allowed for modular, clean code for backend functionality.
- Flask: A lightweight Python microframework used to create the RESTful API and serve the web application. Flask enabled backend handling of user requests, algorithm processing, and delivering JSON responses to the frontend.
- HTML & CSS: These web technologies were used to create the user interface, allowing users to input navigation queries and visualize output. The HTML frontend communicates with the Flask backend dynamically.
- Aerial Campus Image: A high-resolution aerial photograph of the Chanakya University campus was integrated and interpret in the interface. This visual reference enhances user understanding by showing computed paths directly overlayed on the real campus layout.

System Design and Architecture

Flowchart



Module and Feature Definitions

User Interface – Takes input of queries/location from the user and shows the result. (i.e., shows the best route.

Navigation – Uses campus graph and some search algorithms like BFS, UCS, A* to find the shortest path between the locations.

Database – stores campus locations and the distance from every location, contains building information and availabities and the connections (paths)

Methodology

Initially, the campus was modeled as a graph with buildings represented as nodes and walkable paths as weighted edges based on actual distances. Four search algorithms—BFS, DFS, UCS, and A*—were also programmed in Python to calculate the shortest distance from a chosen start point to an end point. The backend is built on Flask and receives inputs from the user for the start location and end location, performs the corresponding search algorithm upon the campus graph, and computes the route along with walking distance and estimated time. The frontend user interface, developed with HTML and CSS, enables users to enter their choices and see the output both in textual directions and as a graphical overlay on an aerial campus map. The system was tested to ensure the correctness and usefulness of the calculated paths and the interface to provide a reliable and user-friendly navigation experience.

PEAS ANALYSIS

Performance:
Accurate and effective pathfinding, shortest paths, precise walking distance and time estimations.

Environment:
Chanakya University campus, user queries through web interface

Actuators:
Web interface showing navigation directions and visual map path overlays

Sensors:
User inputs (start and destination points), campus graph data

Development & Implementation

Campus Graph Construction:

The campus environment was modeled as a graph using Python. Each building such as Hostel, Library, Academic Blocks, Auditorium, and Cafeteria is represented as a node, while edges represent walkable paths with associated distances. This graph data is managed and stored in the Campus.py and Building_info.py modules, providing easy access and modification.

Algorithm Implementation:

Four search algorithms—Depth First Search (DFS), Breadth First Search (BFS), Uniform Cost Search (UCS), and A* with a heuristic—were implemented in separate Python scripts (DFS.py, BFS.py, UCS.py, and heuristic.py). These algorithms receive the campus graph and compute the shortest path between two points.

Backend Development:

The Flask-based backend application acts as the bridge between the user interface and the search algorithms. It handles user input (start and destination), selects the appropriate algorithm, executes the pathfinding logic using campus navigator code , and sends results back to the frontend.

Frontend Interface:

Developed using HTML (index.html) and CSS, the frontend presents a user-friendly interface with dropdown menus to select start and end locations and the search algorithm. Upon submission, it displays the found route, distance, estimated walking time, and visually renders the path on a real aerial image (campus_map.jpg) of the campus using node markers and blue connecting lines.

Route Visualization:

The selected route is overlaid on the campus map image, giving users a clear graphical route reference. The visualization enhances textual directions for better navigation comprehension.

Limitations and Future Scope:

The current visualization connects nodes directly, lacking detailed real walkable path curves. I tried to implement API which would show the real walkable paths from start to destination but couldn't make it successful. Some challenges were there during marking the exact nodes to the exact image location. Future work includes developing an API for realistic walkable paths and enhancing UI interactivity.

Demo video

demo video.mp4

Web Interface Pic

Testing and Results

The BotBrain system was tested using multiple start and destination pairs across the campus locations. All four search algorithms—DFS, BFS, UCS, and A*—were run on numerous queries to validate the correctness of the paths found. The results displayed accurate shortest paths with distances and estimated walking times aligning with real campus data. The web interface successfully returned visual paths on the campus map by connecting the nodes, confirming correct integration between backend logic and frontend visualization. [Tried to implement API but that didn't work as expected]

Conclusion

The BotBrain campus navigator successfully achieves the objective of providing efficient and accurate navigation across the Chanakya University campus. By modeling the campus as a graph and implementing multiple classical search algorithms—DFS, BFS, UCS, and A*—the system delivers optimal routes between any two campus locations. The interactive web interface and visual path overlay on a real aerial campus map enhance usability and understanding for users. While the current implementation connects nodes directly, it lays a solid foundation for future improvements such as real walkable path visualization and mobile integration.