# Table of contents

# Overview

This project by **Agnirudra Sil**, **Jaicharan Yeluri**, **Yash Kumar Singh**, is to build a communication system and a navigation system for our college.


**Name**

Jaicharan Yeluri

**Roll No:**

231CV124

**E-mail**

jai.charan944@gmail.com

**Phone**

+91 83108 04824


## Concept

There are two main aspects of this projects:

- **Communication System**: The aim of this project is to establish a single communication system for the entire college, allowing for streamlined, effective communication with all users, with minimal effort. This project is based on a Discord clone project built by Agnirudra Sil during high school. The other team members built upon this foundation to streamline development and bring the idea closer to realization.
- **Maps**: This project focused on creating a local navigation system tailored specifically for our college campus. The goal of this system is to help students, staff, and visitors find the quickest path to any building or location within the campus grounds. By

calculating the shortest route, this system makes it easy for anyone to navigate efficiently from one point to another, saving time and reducing confusion.

## Code and Repository

The code for the projects is available on GitHub:

- **Communication System**: https://github.com/agnirudrasil/avault
- **Maps**: https://github.com/agnirudrasil/geocom
  - The communication system is also available as a GIT submodule in the Maps repository.

All the important files linked in this document are commented with the **Sphynx** or **Doxygen** style comments.

## Data Structures and Algorithms

### Data Structures

A list of data structures used in the projects:

**Linked List**

Used to implement the queue

**Queue**

Used to process multiple messages when several users are posting messages in the same group

**Binary Heap**

Used to store the nodes and their distances from the source node in the shortest path algorithms

**Adjacency List**

Used to represent the graph in the shortest path algorithms

**Graph**

Used to represent the campus layout in the navigation system

## Algorithms

A list of algorithms used in the projects:

**A* Algorithm**

An algorithm used to find the shortest path between two points on the campus. It uses heuristics to improve performance

**Dijkstra's Algorithm**

Another algorithm used to find the shortest path between two points on the campus. It is a special case of A* with a heuristic of zero

# Maps

Contributor: **Agnirudra Sil** and **Yash Kumar Singh**

## Introduction

This project focused on creating a local navigation system tailored specifically for our college campus. The goal of this system is to help students, staff, and visitors find the quickest path to any building or location within the campus grounds. By calculating the shortest route, this system makes it easy for anyone to navigate efficiently from one point to another, saving time and reducing confusion.

This system is especially useful for new students, such as first-year students, who might not be familiar with the layout of the campus yet. Instead of struggling to find classrooms, labs, or other facilities, users can rely on this navigation tool to confidently get where they need to go. Ultimately, this project aims to make our campus more accessible and welcoming, ensuring that everyone—especially newcomers—can move around smoothly and make the most of their time here.

## Design

The various data structures and algorithms are implemented in the C++ programming language. The frontend is an iOS app. To allow interaction between the app and the algorithms, a REST API written in Typescript is used. The Typescript API accesses C++ classes and functions through the WebAssembly module. The C++ code is compiled to WebAssembly using Emscripten. The iOS app communicates with the API to get the shortest path between two locations on the campus.

## Implementation

Two shortest path algorithms have been implemented for this project. The first is the A* algorithm, which is an algorithm that uses heuristics to improve performance. The second is Dijkstra's algorithm, which is a special case of A* with a heuristic of zero.

- **Header File**: https://github.com/agnirudrasil/geocom/blob/main/pathfinding/pathfinding.h

- **Implementation File**: https://github.com/agnirudrasil/geocom/blob/main/pathfinding/pathfinding.tpp

Both the algorithms use a binary heap data structure to store the nodes and their distances from the source node. The binary heap implementation is based on the `heapq` module in Python.

- **Header File**: https://github.com/agnirudrasil/geocom/blob/main/pathfinding/heap.h
- **Implementation File**: https://github.com/agnirudrasil/geocom/blob/main/pathfinding/heap.tpp

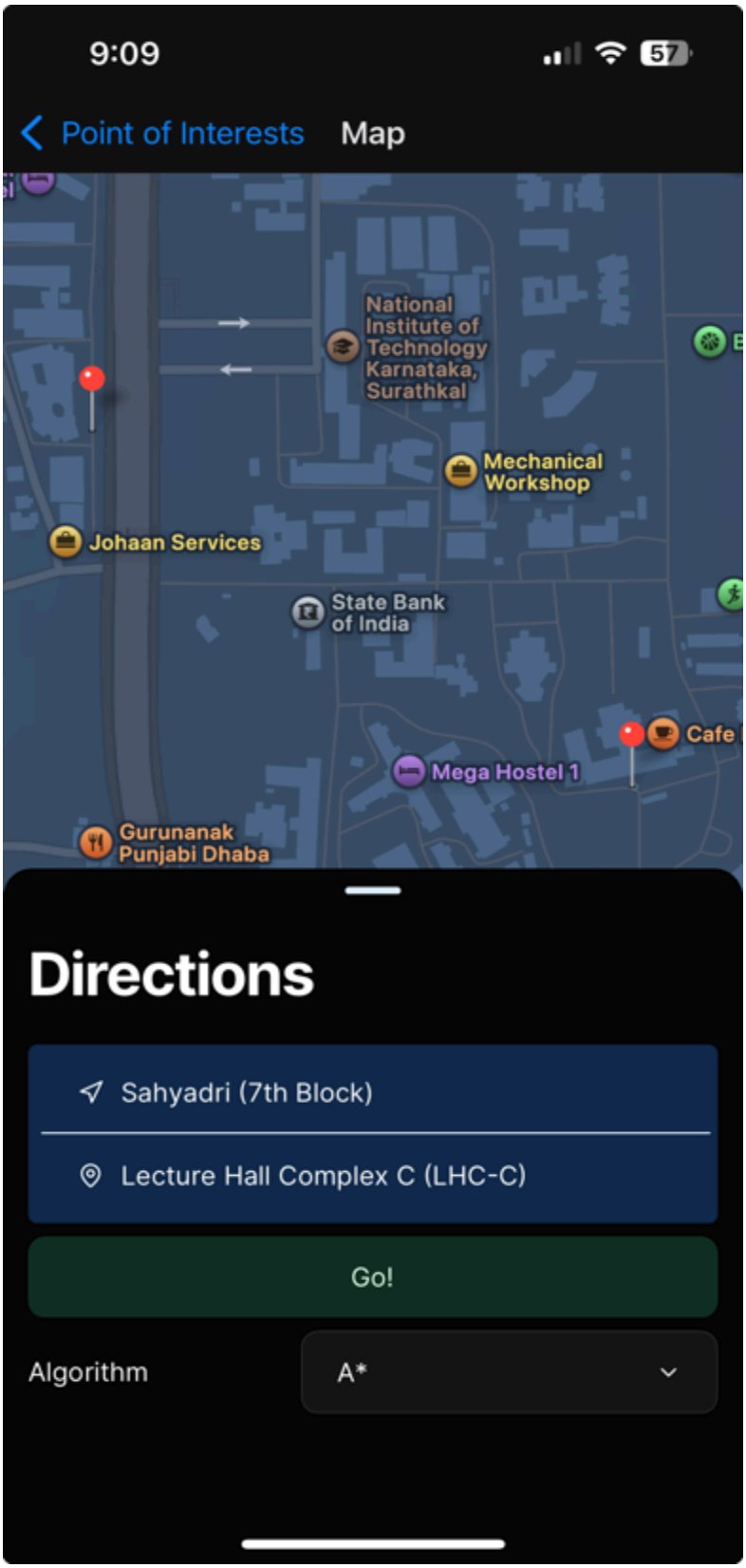
The graph is represented as an adjacency list, where each node has a list of its neighbors.

- **Header File**: https://github.com/agnirudrasil/geocom/blob/main/pathfinding/graph.h
- **Implementation File**: https://github.com/agnirudrasil/geocom/blob/main/pathfinding/graph.tpp

## Execution

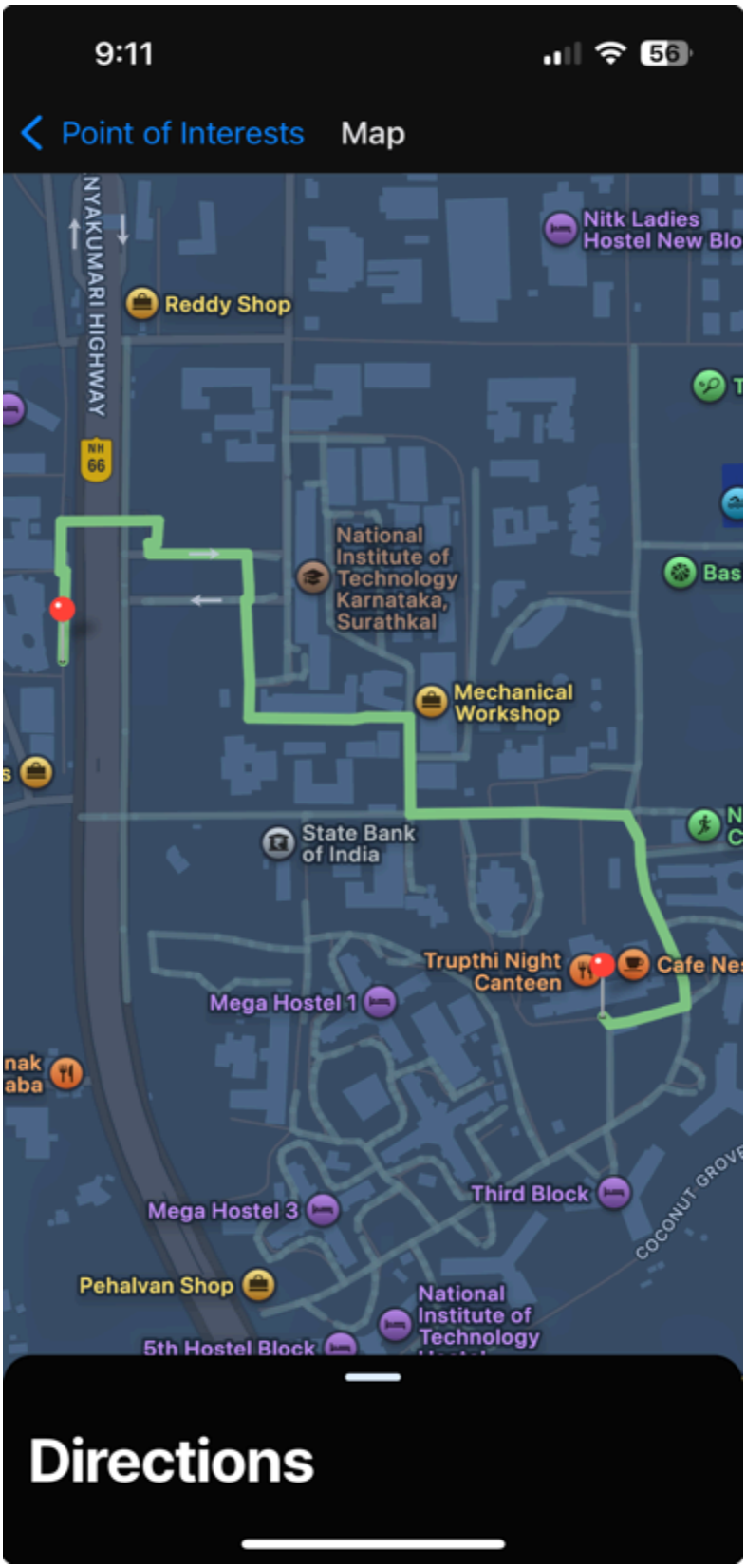
Let us take a realistic situation, Suppose I am a first year student. Being new to the college, I am not very much aware of the campus. Let us say there is an event taking place at LHC-C and I wish to know the quickest path to reach there. Here is where this system would come to aid.

First, Open the app and enter your current location (In my case, block 7 hostel- Sahyadri) then enter the location you wish to reach-destination (LHC-C).




directions

The app would immediately give me the shortest path to reach LHC-C.




shortest_path.png

# Communication System

Contributor: **Jaicharan Yeluri**

## Introduction

My work was to build an effective communication system for our college. However, Implementing the entire communication system from scratch would have been a lengthy process, making it difficult to complete the project within a feasible timeframe. To address this, I used an existing project of my team member, Agnirudra. His project, a Discord messenger clone, closely resembles the communication system we envisioned for our college. By building upon this foundation, we were able to streamline development and bring our idea closer to realization.

The aim of this project is to establish a single communication system for the entire college, allowing for streamlined, effective communication with all users, with minimal effort.

## Design

In the project, I worked on how to process multiple messages when several users are posting messages in the same group. Several users can post messages in the same group at the same time. But we need to reliably display these messages to all clients in the order that they were processed on the server. To achieve this, I implemented a queue data structure using a linked list. Both the linked list and the queue data structure are implemented in Python for easy integration with the existing project.

## Implementation

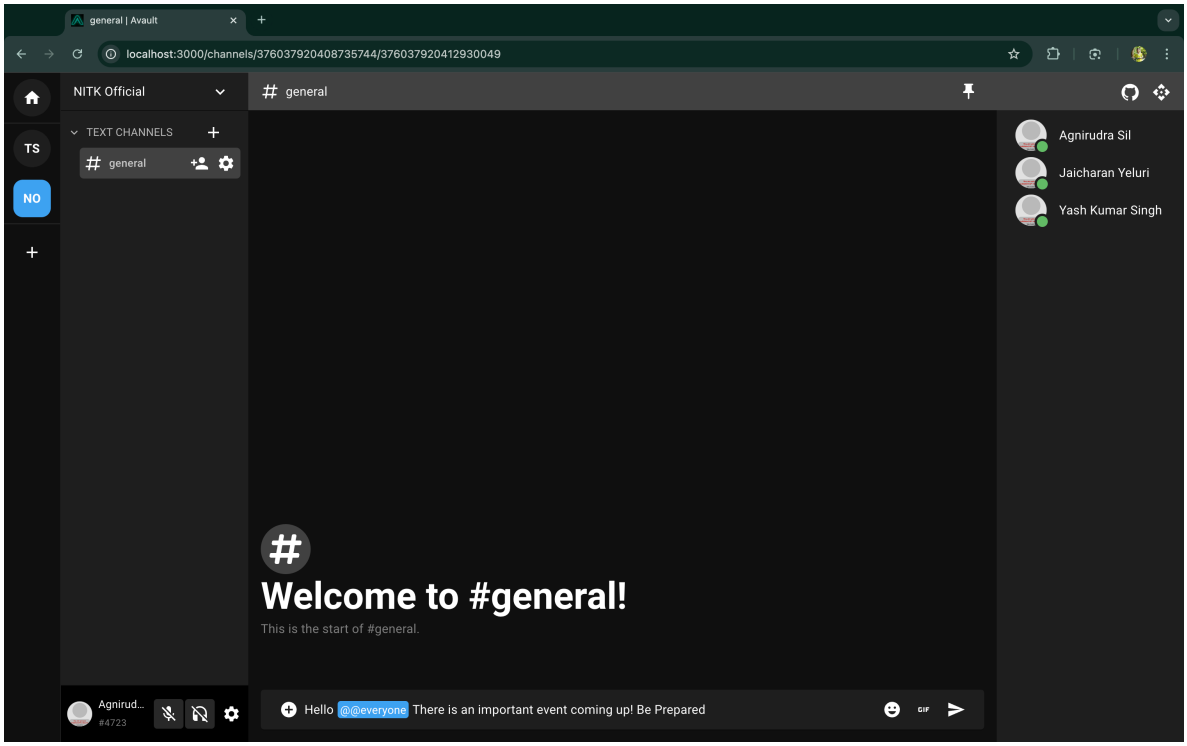
I have implemented linked list and the queue data structure for searching in python in the `linked_list.py` and `queue.py` files respectively. Linked lists were used over dynamic arrays because they allow for constant-time insertions, i.e., $O(1)$ time complexity for deletion at head, and $O(1)$ for insertion at the end, if a pointer to the end of the list is maintained. This is crucial for our project, as we need to process messages quickly and efficiently.

### Github links to the documented codes:

- **Linked lists**: https://github.com/JCode16/avault/blob/ff7407cd5df86d551fe315a41cd330943d97bdd7/backend/app/api/core/linked_list.py
- **Queue**: https://github.com/JCode16/avault/blob/092398f92ebcbae0a097a244a56606b39508bbf4/backend/app/api/core/queue.py

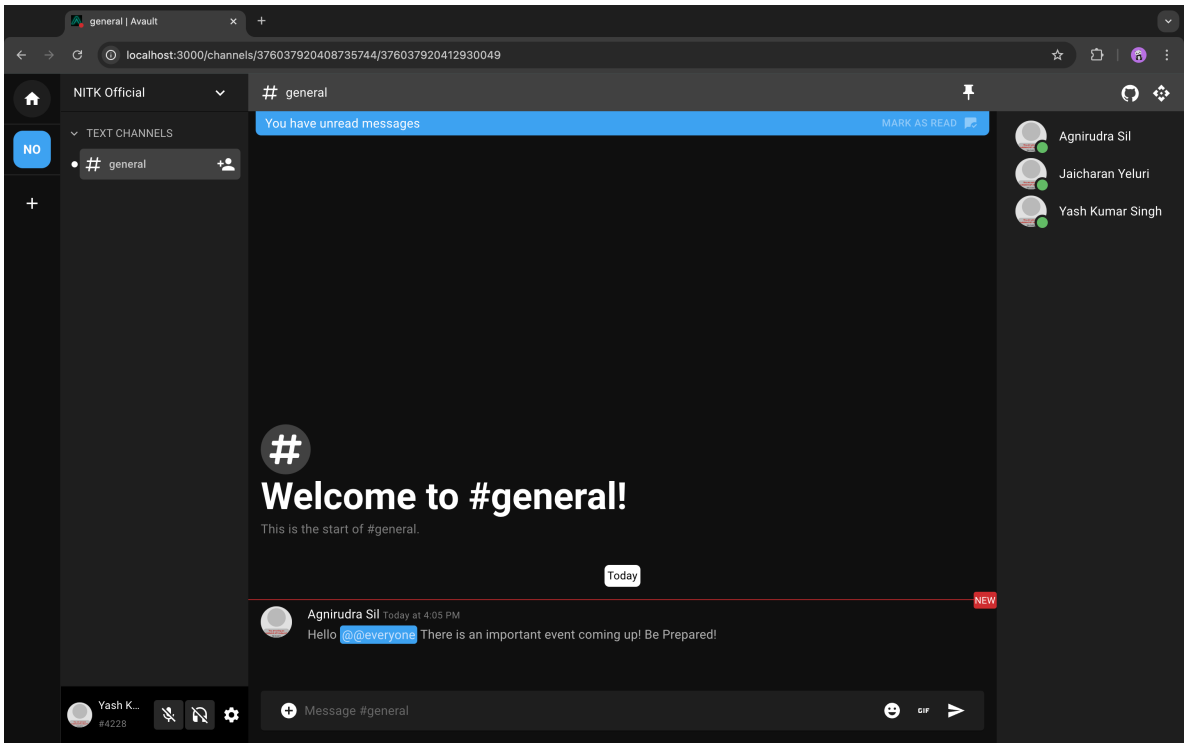
## Execution

Let us take a realistic situation, Suppose I have a group chat with my team members.



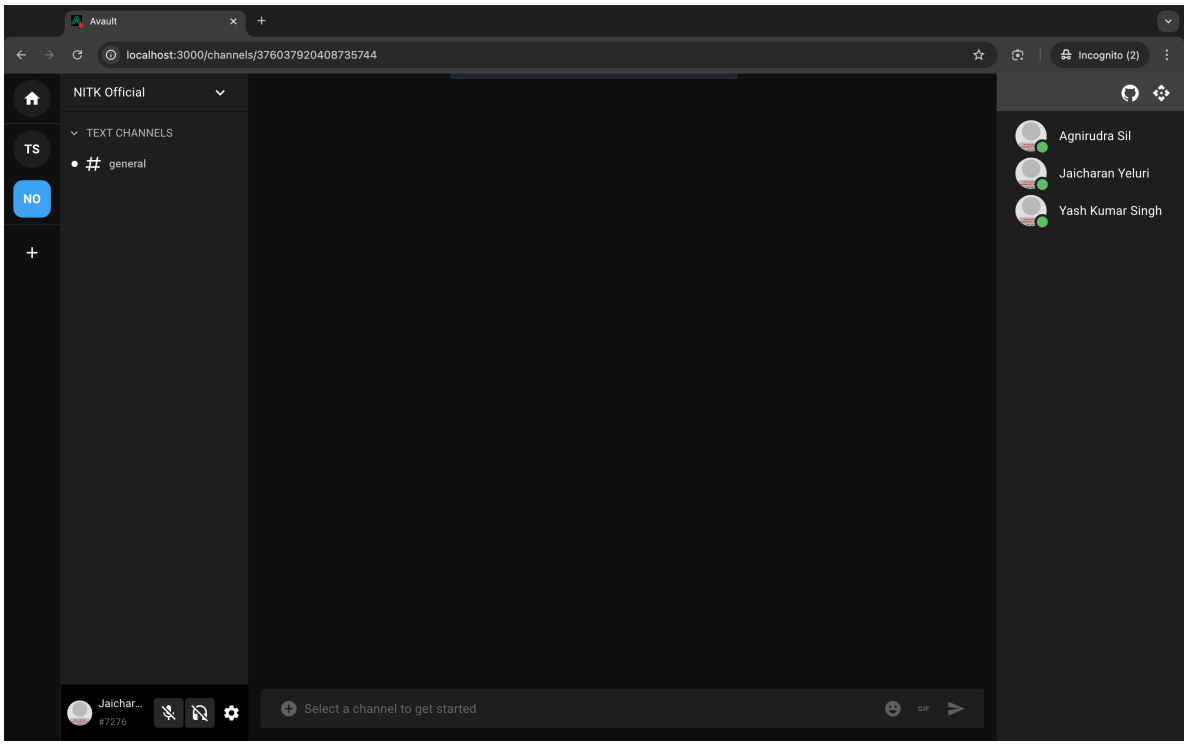

Group Chat

We would be sending different messages at different time intervals. The feature that I implemented now would take all these messages based on a FIFO(first in first out) principle and dequeue them in the chat group




Group Chat

The user who is viewing the channel will be able to see the message immediately after it is sent.




Group Chat

The user will also be notified of the new message in the chat group through a white dot on the chat group icon.