2019

Hofstra Campus Map Application

DESIGN DOCUMENT

J. ABRAHAM; F. CHEN; X. CHEN; X. HU; W. SANTOS

Contents

1. Introduction

This document gives an outline of the overall System Architecture, highlighting the different parts of the system and illustrating their relationships to each other. It is divided into 3 sections: Phase Process, Communication Overview, and Module System. It is heavily derived from the Requirements Specification v1.2 and Project Planning v1.0 Documents. Please refer to these documents for further information.

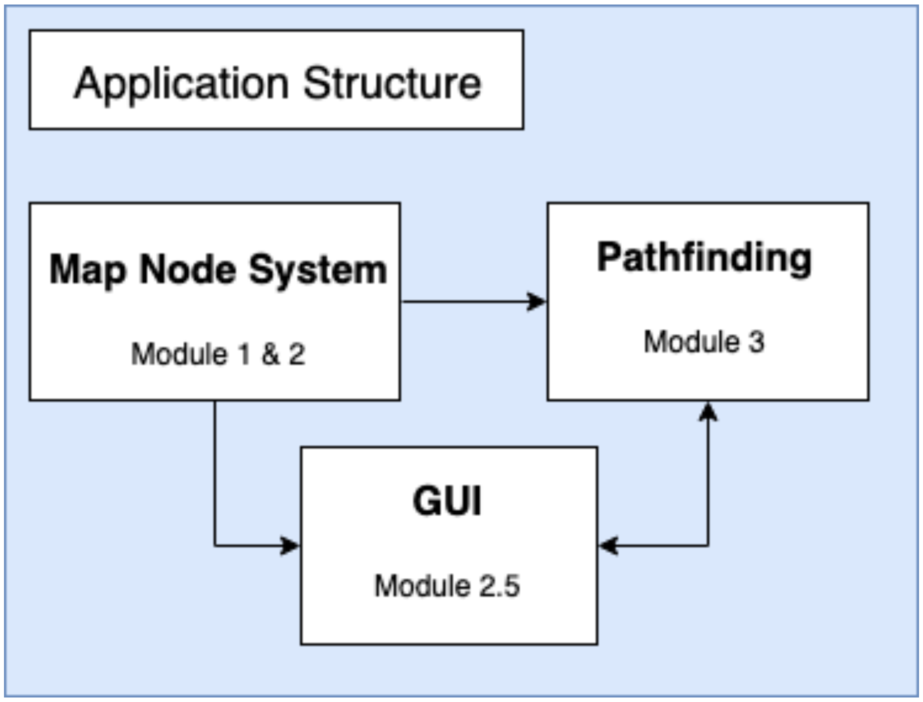


Figure 1: General Application Architecture

2. Communication Overview

Each module takes different inputs and returns different outputs. Figure 1 depicts the high-level structure of the application and the table below represents all the inputs and outputs that each module will take/return. The current implementation will have the client pull data from the database and construct a local copy for the application to work with.

Module 1 Input	Output
• Nodes (Node) • Distances (int)	• Nodes (Node) • Links (Link)
Module 2 Input	**Output**
• .json Data	• Nodes (Node) • Structs (LinkedList<Struct>)
Module 3 Input	**Output**
• Start (Node / NodeSearch) • Goal (Node / NodeSearch)	• Route (LinkedList<NodeSearch>)
Module 5 Input	**Output**
• Node (Node / Struct) • Route (LinkedList<Node>)	• GUI

3. Module System

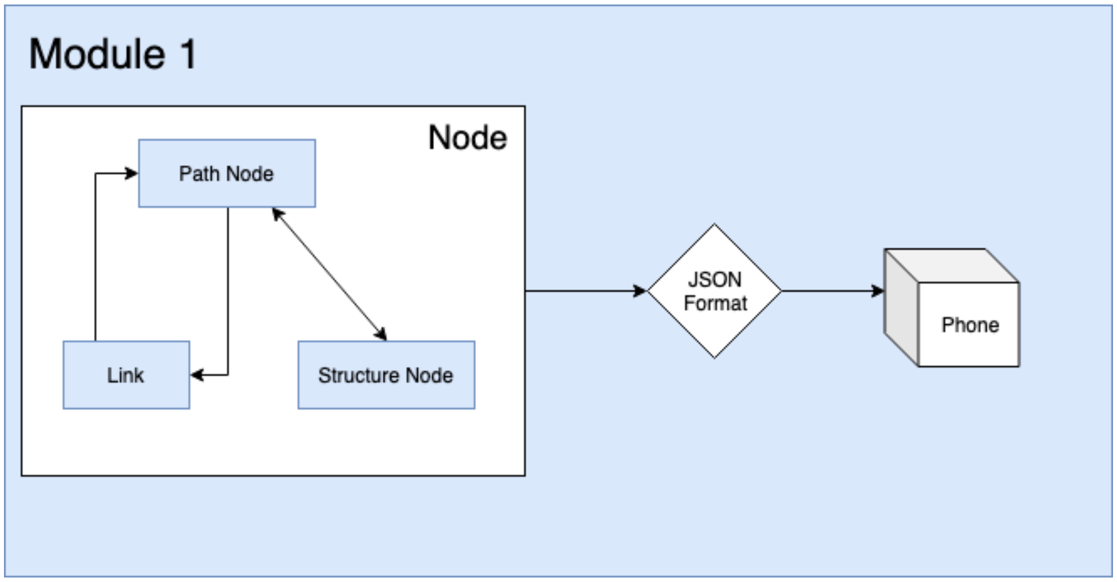
This section discusses the architecture for each Module system. Note that the **Req ID** is the corresponding number from the Requirements Document Ver. 1.2.

Module 1: Node System & Database Setup

Req ID: 4.1

Module 1 may be broken down into three data structures:

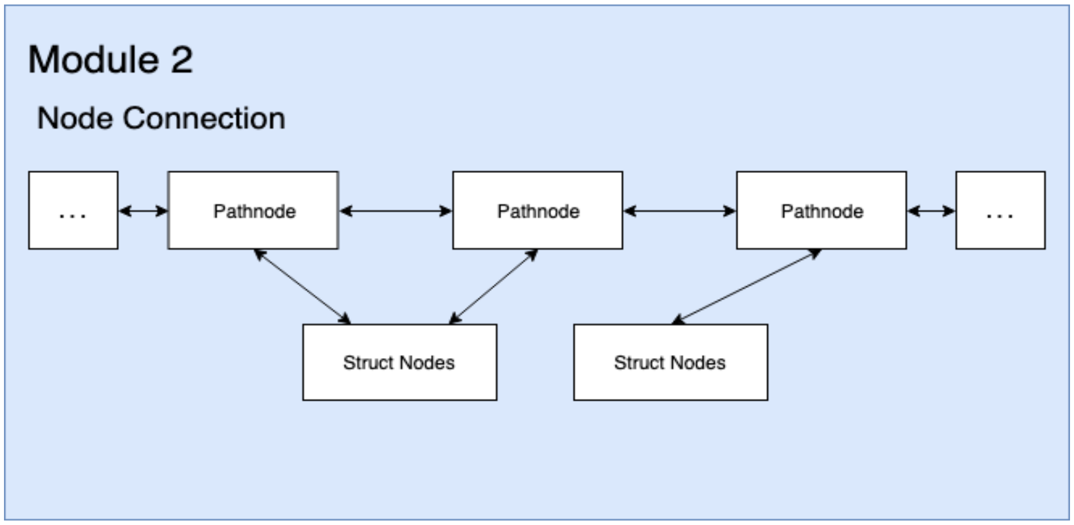
- Path nodes represent distinct positions within the service area.
    - Consist of a unique identifier, a reference to a link, and references to any structure nodes.
- Structure nodes represent distinct buildings, monuments, or locations on campus.
    - Consist of a unique identifier, a list of alternate names, and references to any path nodes.
- Links represent connections between path nodes.
    - Consist of references to two distinct path nodes, and the cost to traverse the link.



Module 2: Map System

Req ID: 3.3, 4.1

When utilized together with data pulled from the database, Module 1 forms a weighted, undirected graph representing the service area; this forms the basis of Module 2. The graph may be traversed from path node to path node by utilizing links. Structure nodes may be directly accessed from connected path nodes, and vice versa.

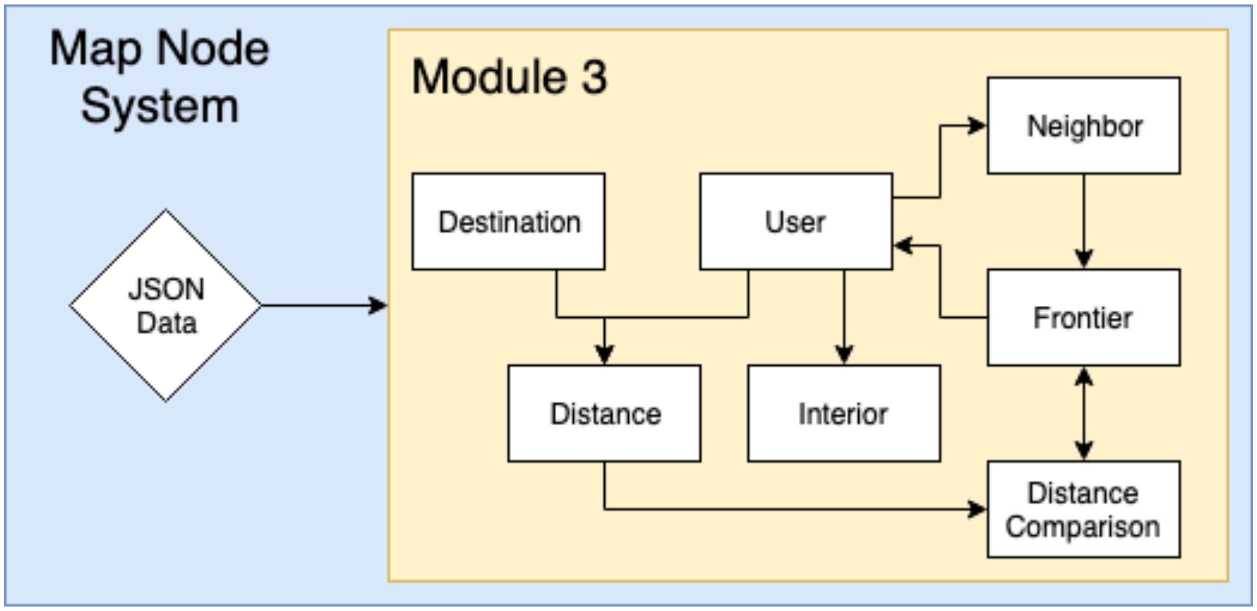


Module 3: Pathfinding Algorithm

Req ID: 3.3, 4.2

Module 3 is mainly focused on the pathfinding algorithm and how to find the shortest route by using the Map System created in Modules 1 & 2. The pathfinding algorithm of choice will be the A* algorithm.

The A* algorithm is an optimal admissible heuristic function. It works well with our established database & format because each node in our database does not have the same values. Each node can have different neighbors and thus, may also have different distance values. As long as the heuristic value is not equal for all nodes, A* is the best algorithm to use given the development team's familiarity with the algorithm as well as its functionality. Below is the diagram of the data flow inside the pathfinding system.



Algorithm Outline

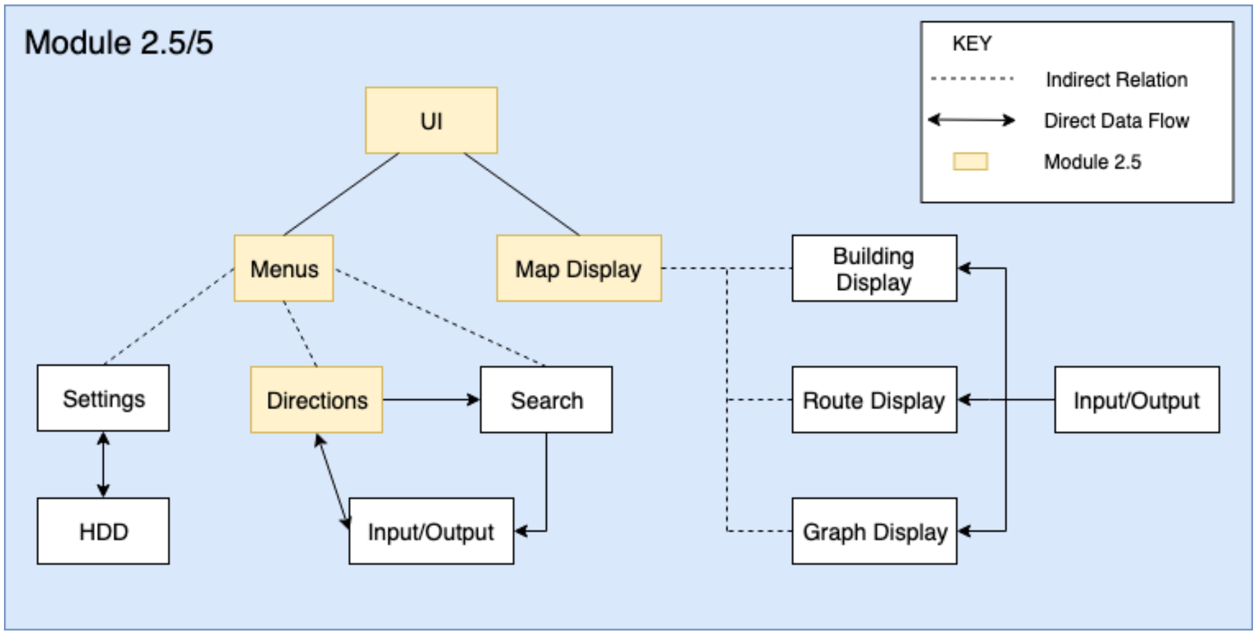
- Take data from Module 1 & 2.
- Create two sets: Frontier and Interior.
  - Frontier is a priority queue.
    - Stores all the neighboring nodes relative to the current node.
  - Interior set stores all the nodes in the optimal path.
- Once the current node is finished processing its neighbor nodes, then the current node is moved to the interior set.
- Heuristic Function (Distance Comparison): $f(x) = g(x) + h(x)$

- This function determines the cost of a node.
    - f(x) = Cost of the node
    - g(x) = Cost so far
        - This is calculated by taking the cost of the current node + the actual distance.
    - h(x) = Estimate cost to destination
        - It is very important to not overestimate the cost.
        - If the cost is higher than the actual cost, the real distance from current node to the destination node, then this function will not be optimal.
        - The closer the estimate to the real cost, the better the heuristic function.
        - For this application, calculating the straight-line distance between two nodes will give a relatively accurate cost estimate without the chance of overestimating the cost.
- Put all neighbor nodes into the Frontier and pop the smallest cost node as the new User node.
    - Since Frontier is a priority queue, this process is O(1).
- Repeat the process until the Destination Node is reached. Then pop all nodes in the Interior set to display the shortest route from the Starting Node to the Destination Node.

Module 2.5/5: GUI

Req ID: 3.1, 3.2, 3.4, 3.5, 4.3

Module 2.5/5 represent the GUI System, taking the data from the Map Node System as well as the user input, and displaying the data accordingly. In the figure below, Module 2.5 is highlighted to show what will appear in Phase I. The non-highlighted represents Phase II, which will come together to comprise Module 5.


Module 2.5/5
KEY
Indirect Relation
Direct Data Flow
Module 2.5
UI
Menus
Map Display
Building Display
Settings
Directions
Search
Route Display
Input/Output
HDD
Input/Output
Graph Display