Software Architecture:

1.1 Identify and describe the major software components and their functionality at a conceptual level:

- 3D Map Renderer
    - Functionality
        - Renders a 3D representation of the campus using OpenGL ES 3.2. Allows zooming, rotating, and panning of the model.
    - Interface
        - Accepts user inputs (touch and gestures) to manipulate the 3D map and adjusts the camera view accordingly.
    - Data
        - Loads campus data and 3D models from local storage (e.g., OBJ files).
- Pathfinding Module
    - Functionality
        - Calculates the optimal path between two points (user's current location and a selected destination) within a building
    - Interface
        - Communicates with the 3D Map Renderer to visually display the path and ensures real-time updates based on the user's position.
    - Data
        - Uses locally stored node-based maps to compute paths using algorithms like A* or Dijkstra's.
- User Location Tracker
    - Functionality
        - Uses Android's Map SDK or Navigate to determine the user's current position on the campus map.
    - Interface
        - Updates the 3D Map Renderer with the user's position and interacts with the Pathfinding Module to show real-time navigation.
    - Data

        - Temporarily stores location data in memory but does not persist data beyond the app session.

- Local Data Management
    - Functionality
        - Handles storage, retrieval, and updates of the campus map, building layouts, and navigation data on the device.
    - Interface
        - Provides interfaces for the 3D Map Renderer and Pathfinding Module to access necessary files (OBJ models, navigation data).
    - Data
        - Stores 3D models, building layouts, and pathfinding data in local storage.

1.2 Specify the interfaces between components:

- *Android SDK 2.0* has the tools necessary to make a customizable 2D navigation application. As stated on their GitHub repository, "The SDK offers indoor mapping capabilities, allowing developers to integrate detailed maps of indoor spaces into their Android applications. Users can benefit from interactive maps, highlighted routes, and turn-by-turn directions to efficiently navigate through the venue and reach their desired destinations."

1.3 Describe in detail what data your system stores, and how:

- OBJ Models
    - Used by the 3D Map Renderer to display the campus.
- Navigation Nodes
    - Used by the pathfinding module for calculating routes between points.

1.4 If there are particular assumptions underpinning your chosen architecture, identify and describe them:

- All data (maps, 3D models, navigation paths) is static and manually updated when new maps or layouts are available. No real-time updates are required.
- The device must have sufficient storage to locally save the campus map and navigation data.
- The app will run offline, relying entirely on local resources.

1.5 For each of two decisions pertaining to your software architecture, identify and briefly describe an alternative. For each of the two alternatives, discuss its pros and cons compared to your choice:

- Client-Server Model (Rejected)
    - Pros
        - Centralized updates to map data, reduced app size, and real-time data synchronization.
    - Cons
        - Requires server management, increases complexity, and introduces latency due to network dependencies.
- Cloud-Storage-Based Approach (Rejected)
    - Pros
        - Allows easier map updates without a dedicated server, reducing the app's local storage footprint.
    - Cons
        - Requires an internet connection for data retrieval, and syncing mechanisms for offline functionality.

2.1 What packages, classes, or other units of abstraction form these components & 2.2 What are the responsibilities of each of those parts of a component:

- Android
    - Activities:
        - MainActivity: First page of the app contains a list of all campus maps in local storage the user can access. User taps on a tap to open it in the MapActivity.
        - MapActivity: Activity where the 3D renderer is implemented and navigation occurs.
- 3D Map Renderer
    - Classes:
        - MapRenderer: Handles loading OBJ models and rendering the 3D map.
        - Camera: Manages camera movements and controls the zoom, pan, and rotation.
    - Responsibilities:
        - Load OBJ models from local storage.
        - Manage user interactions with the map (zoom, pan, rotate).
- Pathfinding Module
    - Classes:
        - Pathfinder: Implements pathfinding algorithms (e.g., A*).

        - NodeGraph: Represents the node-based graph of navigation points within buildings.
    - Responsibilities:
        - Compute the shortest path between points.
        - Communicate with MapRenderer to visualize the computed path.

- User Location Tracker
    - Classes:
        - LocationManager: Interfaces with Android location services to track user position.
    - Responsibilities:
        - Retrieve the user's real-time location.
        - Update the map view and pathfinding results based on the user's movements.

- Local Data Management
    - Classes:
        - DataLoader: Loads 3D models and navigation data from local storage.
        - FileHandler: Reads and writes files for storing OBJ and JSON data.
    - Responsibilities:
        - Manage access to map and pathfinding data.
        - Load required files at startup and refresh data as needed.

3.1 Coding Guidelines:

- Overall
    - Kotlin-Java Interop Guide
- Java
    - The Android Open Source Project Java code style
        - Guide on linters lint
- Kotlin
    - The Android Kotlin style guide
        - Kotlin klinter
- GitHub
    - Conventional Commits
        - A structure to writing git commit messages that defines an easy set of rules to describe the features, fixes, and breaking changes made in commit messages.

- Useful Cheat Sheet.

4.1 Risk Assessment:

- Map data inaccuracies:
  - Likelihood
    - Low
  - Impact
    - High
  - Mitigation
    - Perform manual validation of campus maps before adding them to the app.
- Performance issues with large 3D models:
  - Likelihood
    - Low
  - Impact
    - Medium
  - Mitigation
    - Optimize 3D models and implement efficient culling techniques to reduce the rendering load.
- Pathfinding algorithm inefficiencies:
  - Likelihood
    - Medium
  - Impact
    - Medium
  - Mitigation
    - Test different algorithms (e.g., A*, Dijkstra's).
- Location tracking inaccuracies:
  - Likelihood
    - Medium
  - Impact
    - Medium
  - Mitigation
    - Use fused location provider APIs to improve accuracy, and implement fallback strategies for location triangulation.
    - Test the conversion of location data to 3D space.
- Limited offline storage space:
  - Likelihood
    - Low
  - Impact
    - Low
  - Mitigation
    - Compress 3D models and navigation data to minimize space requirements.

4.2 Project Schedule:

| Milestone | Task | Estimate | Dependencies |
|---|---|---|---|
| Campus Blueprints | Acquire campus blueprints from the Township of Abington | 1 Week | None |
| 3D Map Rendering | Implement OBJ Model Loader | 1 Week | None |
| | Implement shading model and programs | 1 Week | None |
| | Add zoom/pan/rotate functionality | 1 Week | None |
| Campus Model | Use the campus blueprints to create a 3D model of the first floor of sutherland. | 1 Week | Campus Blueprints |
| | Load and render the model in OpenGL | 1 Week | Campus Model |
| | Visualize path on 3D map | 1 Week | None |
| Pathfinding Algorithm | Programmatically generate a node map file | 2 Weeks | Campus Model |
| | Implement A* algorithm | 2 Weeks | Node file generated |
| | Visualize Path on 3D map | 1 Week | Node file generated |
| User Location Tracking | Implement location tracking | 2 Weeks | None |
| | Integrate location with 3D map | 1 Week | Location tracking complete |

4.3 Team structure:

- Adam Jeffery Zorgo
    - Graphic & UI/UX Designer
- Robert Jajko
    - Graphics Programmer
    - Pathfinding Programmer
- Sarim Kazmi
    - Database Programmer
    - Front-End Developer
- Thomas Patrick McLinden
    - Pathfinding Programmer
    - 3D Modeler

4.4 Documentation Plan:

- User Guide:
    - Explains how to use the app, search for buildings, and follow navigation paths.
- Developer Guide:
    - Describes how to add new map data, update the pathfinding algorithm, and optimize rendering performance.