📚 UTech Campus Navigation System
Building, Floor & Room Modification Guide

System Architecture Overview

The UTech Directions System uses three main data structures:

1. **Tree Database** - Hierarchical structure (Root → Building → Floor → Room)
2. **Rooms Hash Map** - Quick lookup for room names
3. **Graph Database** - Pathfinding between buildings with coordinates

📦 PART 1: ADDING NEW ENTITIES

A. Adding a New Building with Floors and Rooms

Follow these steps **in order**:

> **STEP 1: Update Tree Database**
>
> File: `treeDatabase.js`
>
> Location: In the `buildTree()` method, after existing buildings
>
> 1. Create the building node
> 2. Create floor(s) for the building
> 3. Add rooms to each floor
> 4. Register each room in the Rooms Hash Map
> 5. Repeat for additional floors if needed

> **Key Points:**
>
> - Building names: `'building99'` (lowercase, no spaces)
> - Floor names: `'floor99a'`, `'floor99b'` etc. (lowercase)
> - Room names: Can include spaces for special types (`'lt 99'`, `'99a3 lab'`)
> - **ALWAYS** register rooms in roomsHashMap using `.toLowerCase()`

> **STEP 2: Add Building to Graph Database**
>
> File: `graphDatabase.js`
>
> Location: In `buildGraph()` method
>
> 1. Create the building graph node with coordinates (x, y pixel positions)
> 2. Add to the building graph hash map using `buildingGraph.set()`
> 3. Create walkway nodes connecting to existing buildings
> 4. Connect the building to the walkway network using `addBidirectionalNeighbor()`

Key Points:

- Use naming convention: `<region>_b<number>`
- Regions: north, south, east, west, central
- Hash map key **MUST** match tree database building name
- Type is either `'building'` or `'walkway'`
- **MUST** connect to existing walkway network for pathfinding to work

STEP 3: Add Building Picture

File: `buildingPicturesOutput.js`

Location: In `getBuildingPicture()` method, buildingPictures object

- Add entry: `'building99': 'assets/buildings/building99.jpg'`
- Key must be lowercase and match tree database name
- Place actual image file in `assets/buildings/` folder
- Supported formats: .jpg, .jpeg, .png

STEP 4: Add Floor Pictures

File: `floorPicturesOutput.js`

Location: In `getFloorPicture()` method, floorPictures object

- Add entries for each floor
- Keys must match tree database floor names **exactly** (case-sensitive!)
- Place actual image files in `assets/floors/` folder

STEP 5: Add Building Vertices for Map Coloring

File: `pathDrawer.js`

Location: In `buildingVerticesHashMap` Map definition

- Define polygon vertices for building outline on map
- List vertices in clockwise or counter-clockwise order
- Coordinates are pixel positions on the campus map image

- Use image editing software to determine precise coordinates

STEP 6: Update Tests (Optional but Recommended)

- `main.test.js` - Add test cases for new rooms
- `pathfinder.test.js` - Test pathfinding to new building
- `pathdrawer.test.js` - Test map rendering with new building

B. Adding a Single Room to an Existing Floor

> **STEP 1: Update Tree Database**
>
> File: `treeDatabase.js`
>
> Location: Find the existing floor's rooms array
>
> - Add new room entry to the array
> - The forEach loop will automatically register it

That's it! The room is now fully integrated since it uses existing floor/building infrastructure.

C. Adding a New Floor to an Existing Building

> **STEP 1: Update Tree Database**
>
> File: `treeDatabase.js`
>
> Location: After existing floors for the building
>
> 1. Create new floor node
> 2. Add to existing building using `building.addChild()`
> 3. Add rooms array
> 4. Register rooms in hash map

> **STEP 2: Add Floor Picture**
>
> File: `floorPicturesOutput.js`
>
> - Add picture entry for new floor

Note: No changes needed to graph database, building pictures, or vertices - these are building-level, not floor-level.

D. Adding Special Room Types

(Lecture Theatres, Labs, Shared Facilities)

Special rooms are handled **exactly the same** as regular rooms, just with descriptive names:

Examples:

- Lecture theatre: `'lt 9'` → "Go to Lecture Theatre 9"
- Laboratory: `'8a2 lab'` → "Go to room 8A2 Lab"
- Shared facility: `'engineering tuck shop'` → "Go to Engineering tuck shop"
- Department office: `'fenc student affair'` → "Go to FENC Student Affairs"

Key Points:

- Room names can contain spaces
- Use descriptive suffixes: 'lab', 'lt', etc.
- Stored in hash map with lowercase keys
- No special handling needed - treated like any other room

🗑️ PART 2: REMOVING ENTITIES

A. Removing a Room

> **STEP 1: Update Tree Database**
>
> File: `treeDatabase.js`
>
> - Simply delete the room entry from the floor's rooms array
> - The room will automatically not be registered in the hash map

That's it! The room is now completely removed from the system.

B. Removing a Floor

> **STEP 1: Update Tree Database**
>
> File: `treeDatabase.js`
>
> - Comment out or delete the entire floor section (floor node, rooms array, forEach loop)

> **STEP 2: Remove Floor Picture**
>
> File: `floorPicturesOutput.js`
>
> - Delete or comment out the floor picture entry
> - Optional: Delete the actual image file from `assets/floors/`

C. Removing a Building

> ⚠️ **CAUTION:** This requires changes to 5 files!

> **STEP 1: Update Tree Database**
>
> File: `treeDatabase.js`
>
> - Comment out or delete the entire building section (building node, all floors, all rooms)

STEP 2: Update Graph Database

File: `graphDatabase.js`

- Remove the building graph node creation
- Remove from hash map entry
- Remove all connections

IMPORTANT: If removing a building disconnects parts of the walkway network, you must create alternative connections!

STEP 3: Remove Building Picture

File: `buildingPicturesOutput.js`

- Delete the building picture entry

STEP 4: Remove Floor Pictures

File: `floorPicturesOutput.js`

- Delete all floor entries for the building

STEP 5: Remove Building Vertices

File: `pathDrawer.js`

- Delete the vertices entry for the building

✏️ PART 3: MODIFYING ENTITIES

A. Renaming a Room

> **STEP 1: Update Tree Database**
>
> File: `treeDatabase.js`
>
> - Change both the room name and direction text in the room array

That's it! The new name is automatically registered in the hash map.

B. Renaming a Floor

> **STEP 1: Update Tree Database**
>
> File: `treeDatabase.js`
>
> - Change the floor node name
> - Update all room entries to use new floor reference

> **STEP 2: Update Floor Picture**
>
> File: `floorPicturesOutput.js`
>
> - Update the key to match new floor name
> - Optional: Rename the actual image file

C. Renaming a Building

This affects multiple files!

> **STEP 1: Update Tree Database**
>
> File: `treeDatabase.js`
>
> - Change building node name

> **STEP 2: Update Graph Database**
>
> File: `graphDatabase.js`
>
> - Update hash map key (CRITICAL!)

> **STEP 3: Update Building Picture**
>
> File: `buildingPicturesOutput.js`
>
> - Update the key to match new building name

> **STEP 4: Update Building Vertices**
>
> File: `pathDrawer.js`
>
> - Update the key to match new building name

D. Changing Building Coordinates (Graph Position)

> **STEP 1: Update Graph Database**
>
> File: `graphDatabase.js`
>
> - Change the x_coor and y_coor values in the GraphNode constructor

> **STEP 2: Update Building Vertices**
>
> File: `pathDrawer.js`
>
> - Update all vertex coordinates to match new position

> **Note:** Both the graph node position and vertices must be updated together!

E. Modifying Walkway Connections

> **Update Graph Database**
>
> File: `graphDatabase.js`

> - **To ADD a connection:** Use `addBidirectionalNeighbor()`
> - **To REMOVE a connection:** Delete the line
> - **To CHANGE a connection:** Delete old line, add new one

🚪 PART 4: SPECIAL CASES

A. Adding a New Gate

Gates are special - they're direct children of root with no floors!

> **STEP 1: Update Tree Database**
>
> File: `treeDatabase.js`
>
> Location: After existing gates, before end of buildTree()
>
> - Create gate node as direct child of root (no floors!)
> - Register in roomsHashMap

> **STEP 2: Update Building/Floor Node Finder**
>
> File: `buildingFloorNodefinder.js`
>
> - Add new gate name to `isGateNode()` method

> **STEP 3: Update Graph Database**
>
> File: `graphDatabase.js`
>
> - Create gate graph node with type 'gate'
> - Add to hash map
> - Connect to walkway network

> **STEP 4: Add Gate Picture**
>
> File: `buildingPicturesOutput.js`
>
> - Add gate picture entry

> **STEP 5: Add Gate Vertices**
>
> File: `pathDrawer.js`

- Add gate vertices (usually small polygon)

B. Modifying the floorGateGround Virtual Node

This shared floor node is created automatically in `buildingFloorNodefinder.js`.

If you need to change its picture:

File: `floorPicturesOutput.js`

- Modify entry: `'floorGateGround': 'assets/floors/new_gate_ground.jpg'`

Note: You should NOT need to modify the virtual node creation logic itself.

✅ PART 5: VALIDATION CHECKLIST

After making any changes, verify:

Tree Database Checklist

- ☐ Building node created and added to root
- ☐ Floor nodes created and added to building
- ☐ Room nodes created and added to floor
- ☐ All rooms registered in roomsHashMap with lowercase keys
- ☐ Naming conventions followed (lowercase, no spaces in building/floor names)

Graph Database Checklist

- ☐ Building graph node created with correct coordinates
- ☐ Building added to hash map with key matching tree database name
- ☐ Building connected to walkway network
- ☐ No disconnected graph sections (all buildings reachable)

Output Components Checklist

- ☐ Building picture added (buildingPicturesOutput.js)
- ☐ Floor picture(s) added (floorPicturesOutput.js)
- ☐ Building vertices added (pathDrawer.js)
- ☐ Actual image files exist in assets/ folders

Special Cases Checklist

- [ ] Gates updated in isGateNode() if adding/removing gates
- [ ] Lecture theatres/labs named with spaces and suffixes
- [ ] Shared facilities added like regular rooms

❌ PART 6: COMMON MISTAKES TO AVOID

1. Mismatch between tree and graph names

Tree Database: `'building99'`

Graph Database: `'building_99'` ❌ WRONG! Key doesn't match

2. Forgetting to register rooms in hash map

Creating room node but not adding: `this.roomsHashMap.set(roomName, roomNode)`

3. Case sensitivity in floor pictures

Tree Database: `'floor1a'` (lowercase)

Floor Pictures: `'floor1A'` ❌ WRONG! Case mismatch

4. Disconnected graph nodes

Creating building graph node but forgetting to connect it to walkway network

5. Using spaces in building/floor names

`'building 99'` ❌ WRONG! No spaces allowed

`'building99'` ✅ CORRECT

🧪 PART 7: TESTING YOUR CHANGES

Manual Testing Steps:

🧪 PART 7: TESTING YOUR CHANGES

Manual Testing Steps:

1. Test Room Lookup

- Enter the new room as "To Room"
- Verify it's found in the system
- Check that error messages are clear if not found

2. Test Pathfinding

- Navigate FROM new building TO existing building
- Navigate FROM existing building TO new building
- Verify path is displayed on map

3. Test Outputs

- Verify building picture displays
- Verify floor picture displays
- Verify floor highlight shows correct floor
- Verify map colors new building correctly

4. Test Edge Cases

- Test with different cases (e.g., "99A1" vs "99a1")
- Test special room types (lecture theatres, labs)
- Test gate navigation if modified

📊 PART 8: QUICK REFERENCE TABLE

Task	Files to Modify	Order Important?
Add Building	1. treeDatabase.js 2. graphDatabase.js 3. buildingPicturesOutput.js 4. floorPicturesOutput.js 5. pathDrawer.js	✅ Must follow this order
Add Floor	1. treeDatabase.js 2. floorPicturesOutput.js	❌ Any order
Add Room	1. treeDatabase.js	N/A - Only one file
Add Gate	1. treeDatabase.js 2. buildingFloorNodefinder.js 3. graphDatabase.js 4. buildingPicturesOutput.js 5. pathDrawer.js	✅ Must follow this order
Remove Room	1. treeDatabase.js	N/A - Only one file
Remove Floor	1. treeDatabase.js 2. floorPicturesOutput.js	❌ Any order
Remove Building	All 5 files: - treeDatabase.js - graphDatabase.js - buildingPicturesOutput.js - floorPicturesOutput.js - pathDrawer.js	⚠️ Any order, but be careful with graph connections
Rename Room	1. treeDatabase.js	N/A - Only one file
Rename Floor	1. treeDatabase.js 2. floorPicturesOutput.js	❌ Any order
Rename Building	All 4 files: - treeDatabase.js	❌ Any order

Task	Files to Modify	Order Important?
	- graphDatabase.js - buildingPicturesOutput.js - pathDrawer.js	
Modify Coordinates	1. graphDatabase.js 2. pathDrawer.js	⚠️ Must update together

💡 SUMMARY & KEY PRINCIPLES

The Core Principle

Tree Database is the source of truth for hierarchical structure (Building → Floor → Room)

Graph Database is the source of truth for pathfinding and coordinates

Always start with Tree Database, then update supporting systems (Graph, Pictures, Vertices) keeping names consistent across all files.

Golden Rules:

1. **Consistency is Critical:** Building/floor/room names must match exactly across all files (respecting case sensitivity where noted)
2. **Hash Map Keys:** Always use lowercase for room registrations in Tree Database
3. **Graph Connectivity:** Every building must be connected to the walkway network for pathfinding to work
4. **Visual Assets:** Don't forget to add actual image files to the assets folders
5. **Special Rooms:** Lecture theatres, labs, and shared facilities are just regular rooms with descriptive names

⚠️ Most Common Pitfalls

- **Name Mismatches:** Graph database key doesn't match tree database name
- **Forgotten Registration:** Creating room node but not adding to roomsHashMap
- **Case Issues:** Floor picture key case doesn't match tree database floor name
- **Disconnected Graphs:** Building exists but isn't connected to walkway network
- **Spacing Errors:** Using spaces in building/floor identifiers (only room names can have spaces)

🎯 Best Practices for Maintenance

- **Start Simple:** When adding a new building, start with one floor and a few rooms to test
- **Test Immediately:** Don't wait until all changes are complete - test after each major step
- **Use Consistent Naming:** Follow the existing naming conventions (building99, floor99a, etc.)
- **Document Changes:** Keep notes about what you modified and why
- **Backup First:** Always backup files before making major changes
- **Check Dependencies:** When removing elements, verify nothing else references them

Need Help?

If you encounter issues:

- Review the validation checklist for your specific task
- Check the common mistakes section
- Verify all file names and keys match exactly
- Test pathfinding between buildings to verify graph connectivity
- Examine existing examples in the codebase for reference

📝 Document Version

UTech Campus Navigation System

Building, Floor & Room Modification Guide

Last Updated: December 2025