



# Interactive Urban Navigation System for Icherisheher

Advisor: Stephen Kaisler

Students: Huru Algayeva, Abbas Aliyev

# What If Google Maps Led You *Straight Into a Wall?*

01

## Irregular geometry

- Streets don't follow modern grid.

02

## Incomplete labeling

- Many paths aren't even recognized by mapping tools.

03

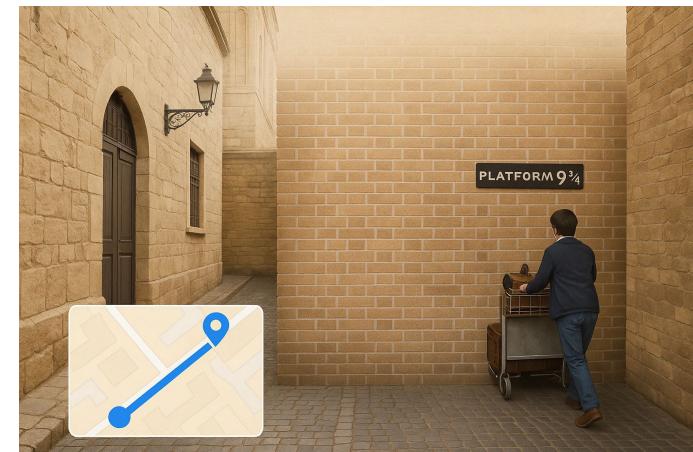
## Conflicting digital data

- APIs provide partial or mismatched coverage.

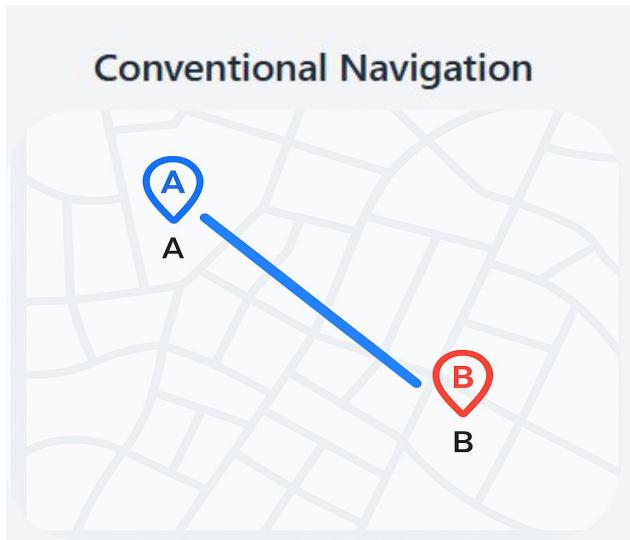
04

## Physical constraints

- Narrow alleys, staircases, and stone barriers disrupt routing logic.



# Beyond Simple Navigation: The Challenge of Historical Districts



- ✗ Ignores cultural significance
- ✗ Treats all paths equally
- ✗ Lacks historical context

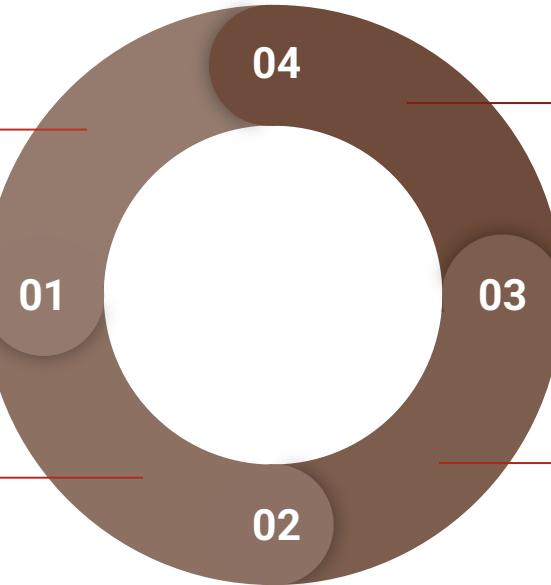
- ✓ Narrow, winding lanes
- ✓ Rich historical landmarks
- ✓ Cultural significance varies by path

# Research Questions

How can we design a **navigation system** that accurately interprets historical urban layouts like Icherisheher?



What are the **limitations of existing maps** in handling culturally and spatially complex environments?



How can **routing algorithms** be designed and integrated to account for **not only distance and time, but also historical and cultural context**?

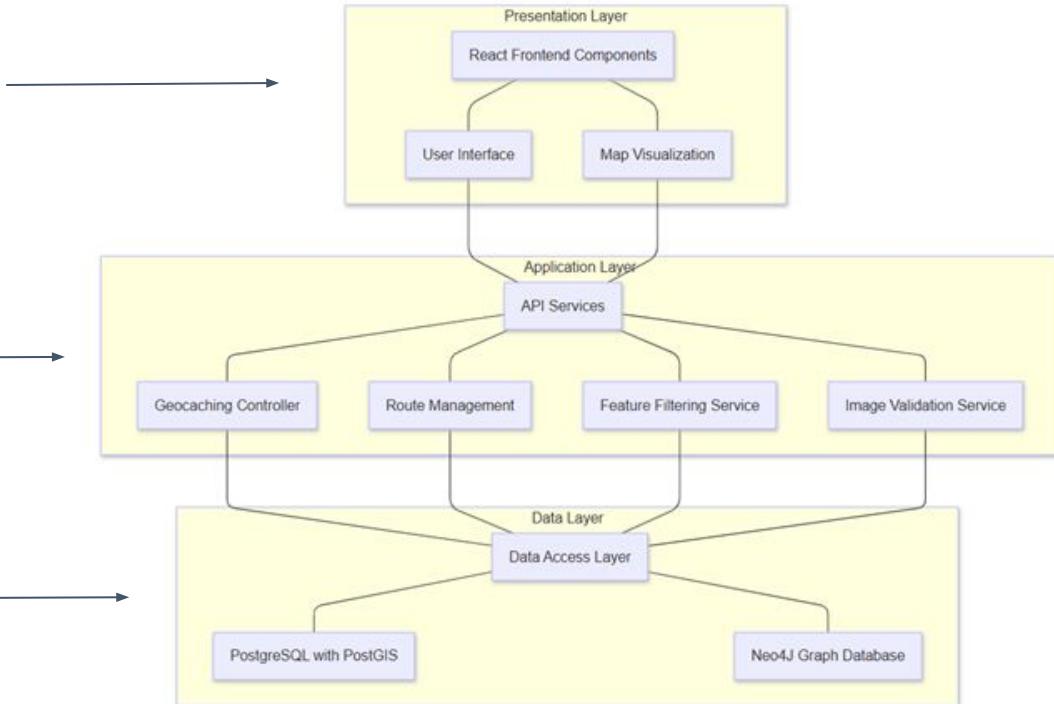


Can computer vision be used effectively to **validate user geocaches** through image?



# High Level Architecture

**Web-based interface** for user interaction and map display.

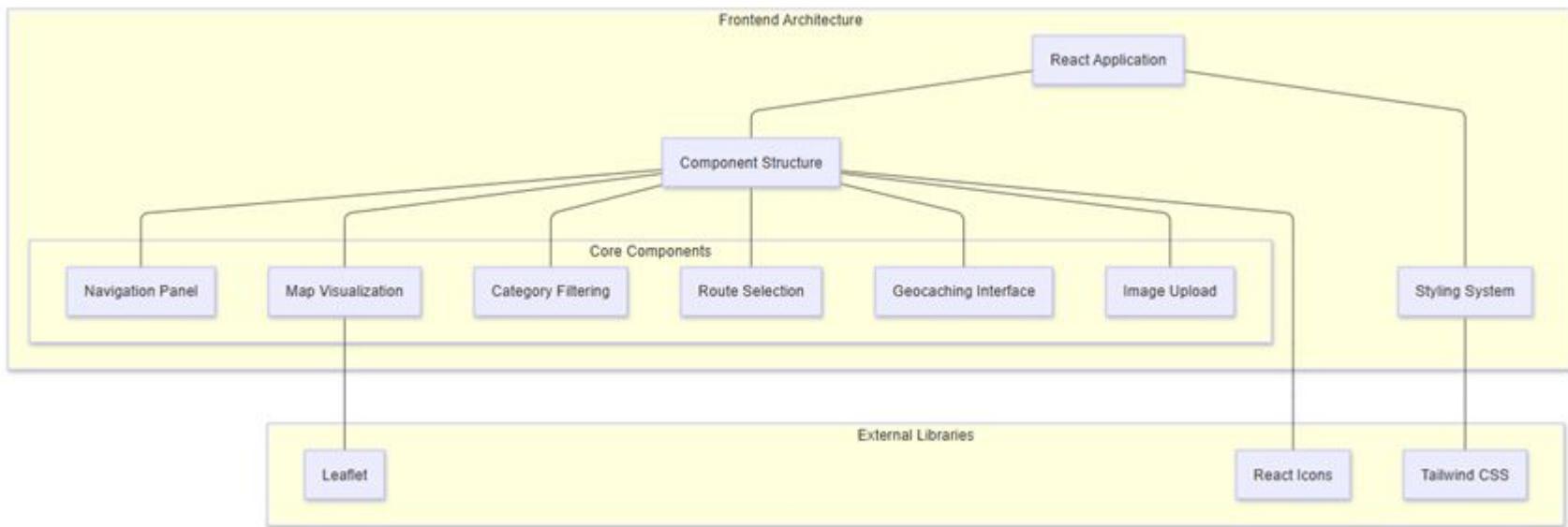


Modular services for **routing, filtering, image validation, and geocaching**.

Dual-database logic — relational (PostgreSQL + PostGIS) for **geocaching and filtering** and graph-based (Neo4j) for **route relationships**.

# Frontend Architecture

- React-based frontend with reusable components
- Leaflet used for map rendering and route overlays
- Interfaces for **route selection, feature filtering, geocaching, and image upload**
- Tailwind CSS and React Icons provide fast styling and visual clarity

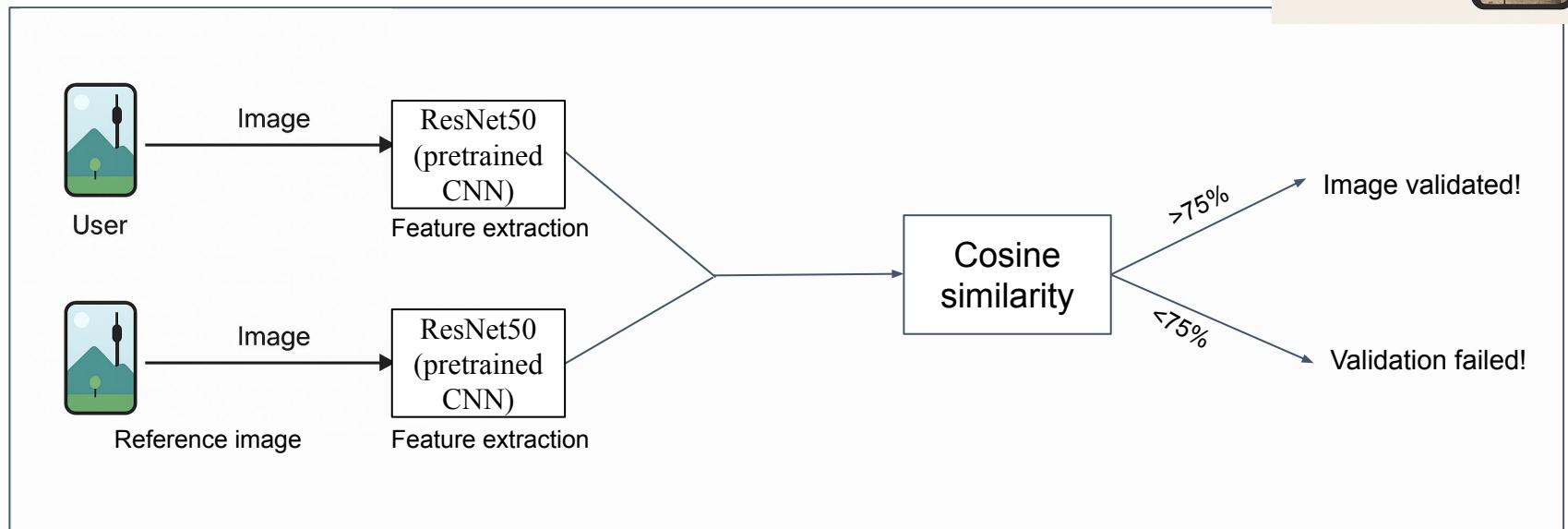
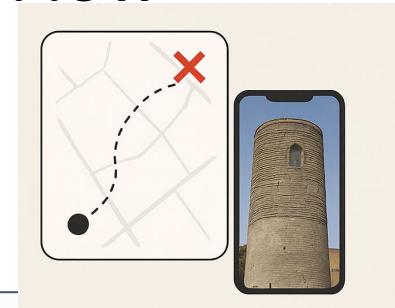


# Geocaching Architecture & Image Validation Flow

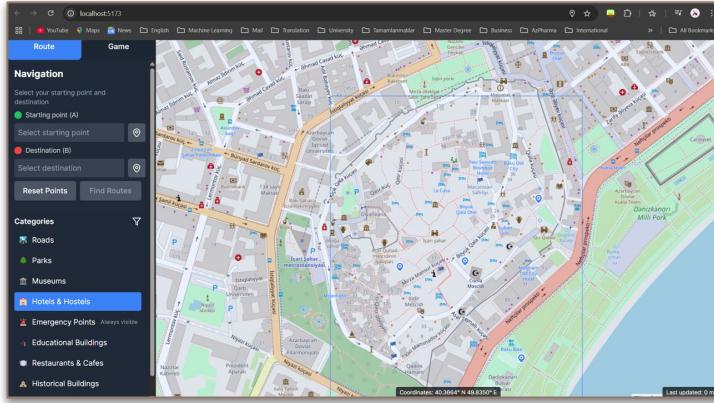
**Geocaching** is like a GPS-based *treasure hunt*.

You find hidden spots using **clues and maps**.

In our system, users take a photo to prove they found it.

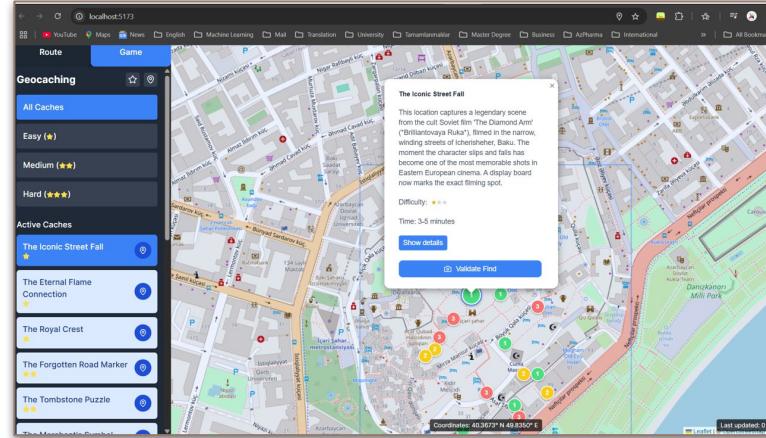


# User Interface Overview: *Where It All Begins*

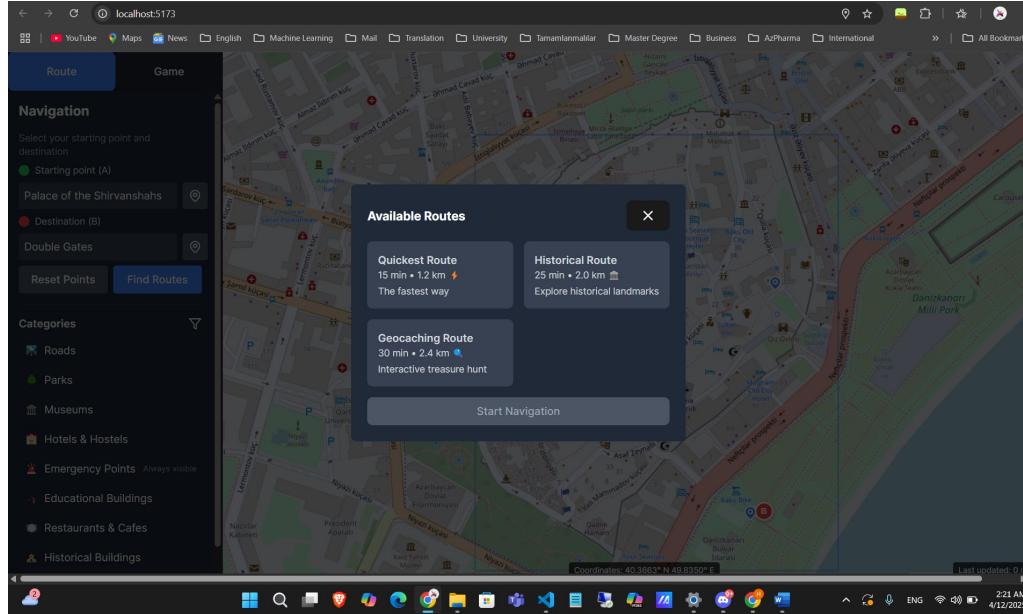


- Choose **start and end points**
- Filter by **location type** (e.g. hotels, parks)
- View optimized **routes**
- Built with **React + Leaflet**, powered by Neo4j and PostgreSQL

- Select **caches by difficulty** (★ to ★★★★)
- View **cultural info + time to complete**
- “**Validate Find**” with image upload
- Makes urban exploration **fun and interactive**



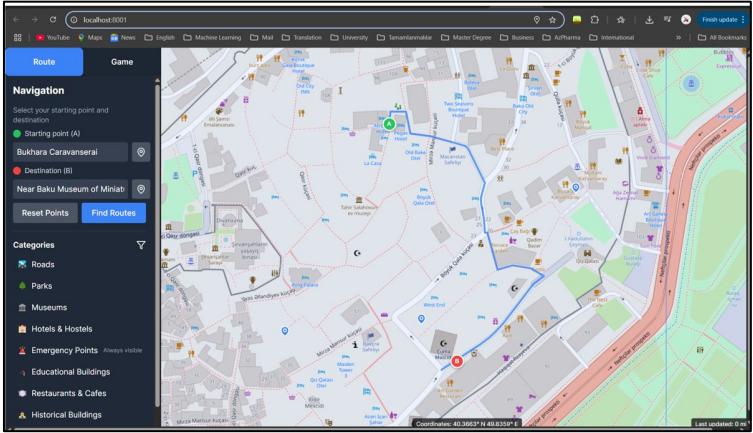
# Route Selection & Pathfinding Results



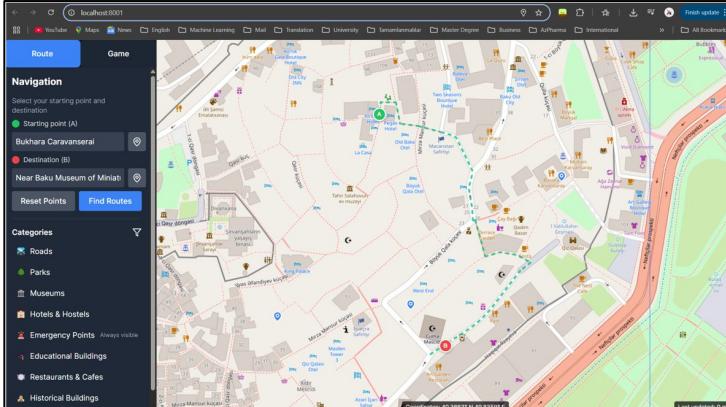
User selects from three modes:

- **Quickest** 🚶 — Fastest route
- **Historical** 🏛 — Prioritizes cultural landmarks
- **Geocaching** 📌 — Includes hidden cache locations

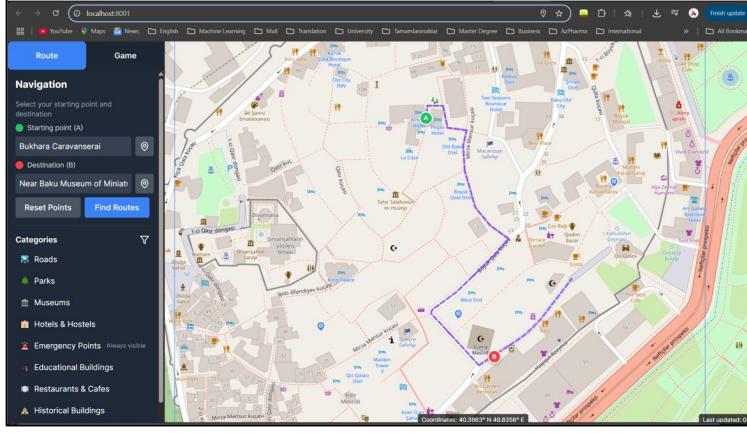
# Pathfinding Results



Quickest



Geocaching



Historical

# Image Validation in Action

This screenshot shows a geocaching application interface. On the left, a sidebar lists challenges: 'All Caches', 'Easy (•)', 'Medium (••)', 'Hard (•••)', and 'Active Caches'. The main area displays a map of a city street with various geocache locations marked. A callout box for 'The Iconic Street Fall' challenge provides details: it's an easy challenge (difficulty 1/4), located at coordinates 40.3670°N 69.4349°E, and requires taking a photo of a specific scene from a fallen board. A blue button at the bottom right says 'Validate Find'.

## Step 1: Read the Challenge

User receives geocache details including story, difficulty, task, and validation rules.

This screenshot shows the user interface after the photo has been taken. A callout box says 'Validate Your Find' and instructs the user to take a photo of the cache location. Below it, a modal window titled 'Choose File' shows a thumbnail of the uploaded image (WhatsApp Image 2...). A blue 'Validate' button is visible at the bottom right of the modal.

## Step 2: Upload a Photo

User uploads an image replicating the scene described in the challenge.

This screenshot shows the validation results. A callout box says 'Success!' and indicates the image was successfully validated with 84.1% accuracy. A green progress bar shows the accuracy level. A blue 'Accept' button is at the bottom right. The background map and sidebar are identical to the previous steps.

## Step 3: Get Validated

✓ Image validated successfully with **84.1% accuracy**.

*Note: All reference photos were collected and validated by us on-site.*

# Conclusion

## *From Confusion to Connection in Icherisheher*

- A complete navigation system made for complex historic cities
- Integrated route planning, cultural geocaching, and image validation
- Built using **React**, **Leaflet**, **Neo4j**, **PostgreSQL**, and **ResNet50**



### ♦ What We Made

- Makes heritage exploration **interactive**, not passive
- Helps both tourists and locals engage meaningfully with the city
- Encourages cultural learning through **gamified discovery**



### ♦ Why It Matters

- Dual-database setup: **PostgreSQL + Neo4j**
- Real-world **geocache validation using computer vision**
- Modular and scalable — adaptable to any cultural city



### ♦ Why It's Better and Different

# Future Work & Next Steps



## Expansion

- Offline mode for low-connectivity areas
- Augmented Reality overlays
- Global deployment to other heritage cities

## User-Centered

- Larger, diverse user testing
- Adaptive difficulty in geocaching

## Image Validation

- Exploring different deep learning architectures
- Testing alternative feature extractors

# Thank you for navigating Icherisheher with us!

