03/10/2025

5DV247-Planning Document

Project #4 – Kallbad Trip

Leader: DOGLIOLI RUPPERT Germain
Members: BITOUN Nathan, MATER Hugo

Table of Contents

1 – Single-sentence Requirements

Functional Requirements

1. **Interactive map display**: The platform will show all Swedish bathing sites on an interactive map.

2. **Real-time API integration**: Data about water temperature, quality ratings, and location info will be pulled directly from the HaV (Havs- och vattenmyndigheten) REST API every time a user loads a site page.

3. **User registration and login**: People need to be able to create an account with email/password so they can leave reviews and track their visits.

4. **Detailed site information pages**: Each bathing site should have its own page showing:

   - Current water temperature (from API)
   - Water quality status (from API)
   - Location coordinates and map view
   - User reviews and average rating

5. **Review and rating system**: Logged-in users can write a text review and give a rating (probably 1-5 stars) for any bathing site they've visited.

7. **Personal visit history**: Users who are logged in can see a list of all the sites they've reviewed or marked as visited.

8. **Admin moderation tools**: We need at least one admin account that can delete inappropriate reviews and ban users if necessary.

8. **Persistent data storage**: All user accounts, reviews, ratings, and visit history will be stored in a PostgreSQL database.

9. **Filter and search functionality**: Users should be able to filter sites by:

   - Water temperature range (e.g., "show only sites above 18°C")
   - Water quality (e.g., "excellent" vs "acceptable")
   - Geographic location (maybe by region or county)

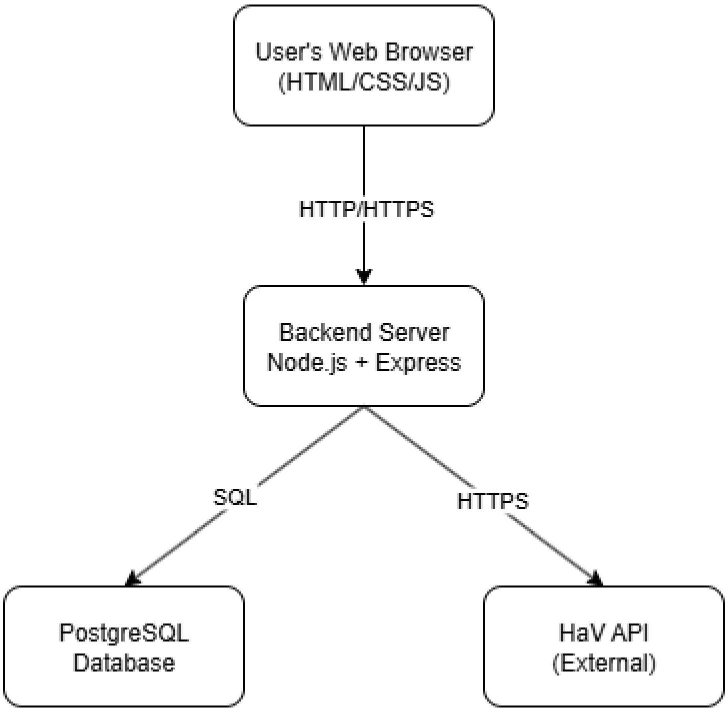
10. **Browser accessibility**: The whole platform needs to work in modern web browsers (Chrome, Firefox, Safari, Edge). No mobile app for now, just responsive web.

Non-Functional Requirements

- **Separation of concerns**: We're keeping frontend (vanilla JS/HTML/CSS), backend (Node.js/Express), and database (PostgreSQL) clearly separated.
- **Data freshness**: Since the HaV API provides real-time data, we want to fetch it on-demand rather than storing stale information.
- **Responsive design**: The interface should work decently on both desktop and mobile browsers. We're not going super fancy with the design, but it needs to be usable on a phone.
- **Basic security practices**: We'll hash passwords, use HTTPS in production, and validate/sanitize all user inputs to prevent SQL injection and XSS attacks.

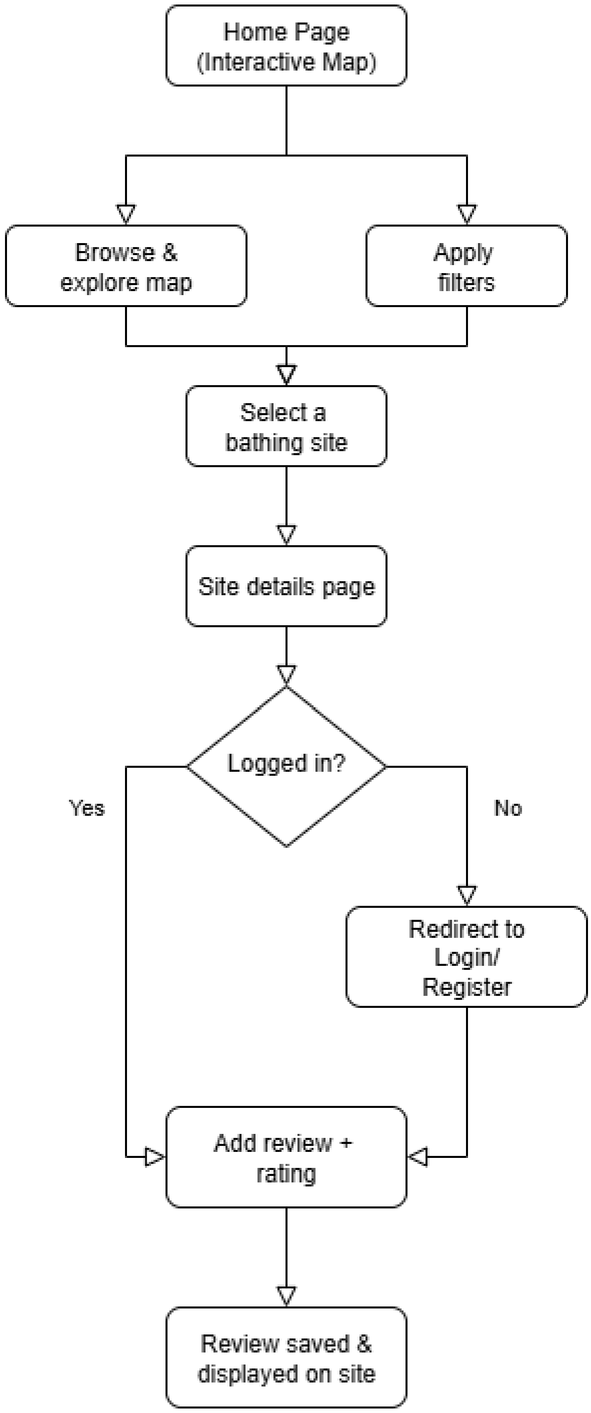
2 – Design Diagrams

2.1 – System Architecture

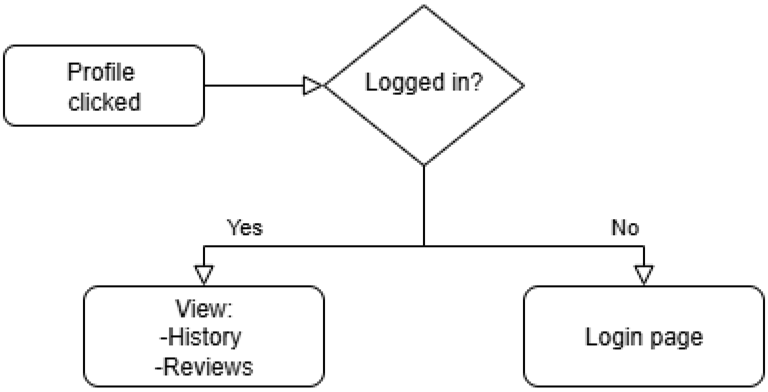


2.2 – User Flow

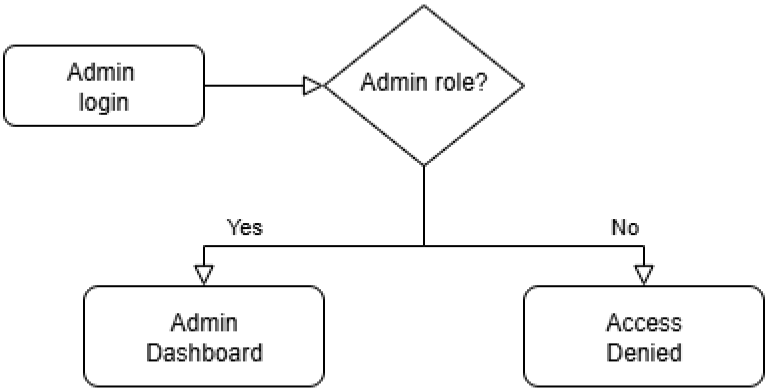
Main User Journey



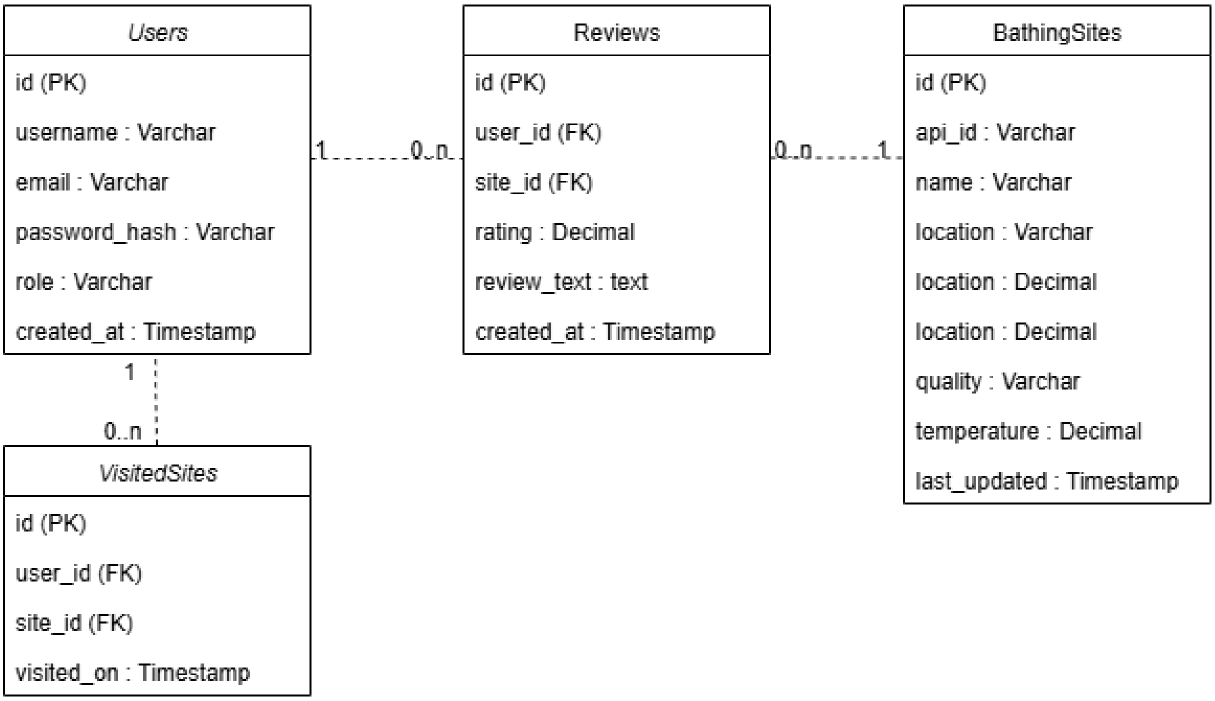
User Account



Admin Panel



2.3 – Data Model (ER Diagram)



3 – Schedule & Gantt Chart with Milestones

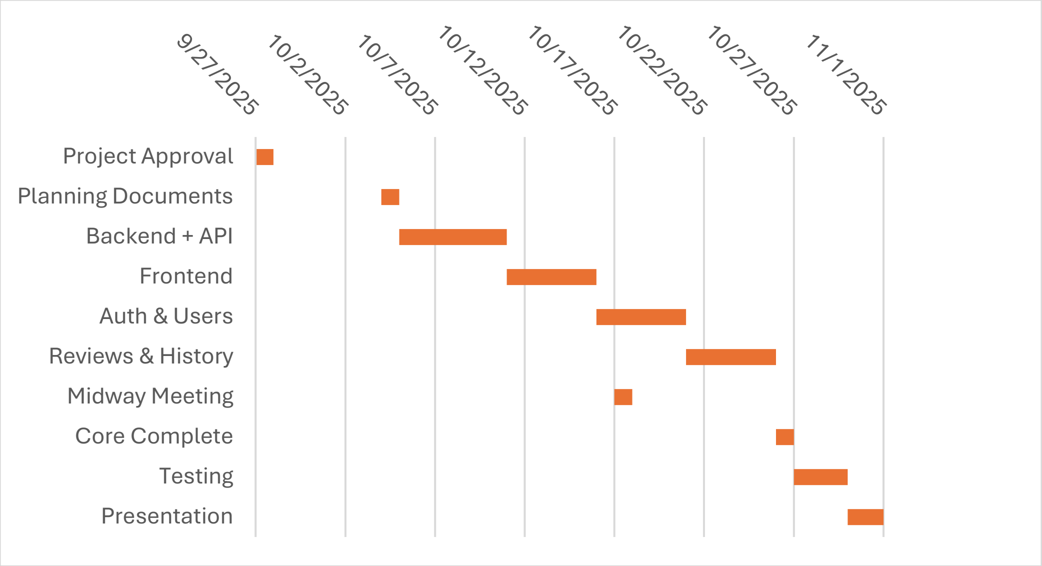
3.1 – Milestones

Date	Milestone	Status
Sept 27	Project proposal approved by instructor	Done
Oct 4	Planning documents submitted	This document
Oct 17	Midway progress meeting	Upcoming
Oct 25	All core features implemented and tested	Target
Oct 30	Final presentation (video + report)	Deadline

3.2 – Task Breakdown

Week	Dates	Tasks	Who
Week 1	Oct 5-11	Set up GitHub repo Initialize Node.js backend Setup PostgreSQL database Test HaV API connection	Nathan + Hugo
Week 2	Oct 12-18	Build basic frontend (map + site list) Implement API data fetching on frontend **Oct 17: Midway meeting**	Germain + Hugo
Week 3	Oct 19-25	User authentication (register/login) Review submission form Admin moderation panel	All team
Week 4	Oct 26-30	Final testing & bug fixes Record demo video Write 5-page report **Oct 30: Submit!**	All team

3.3 – Gantt Chart


9/27/2025
10/2/2025
10/7/2025
10/12/2025
10/17/2025
10/22/2025
10/27/2025
11/1/2025
Project Approval
Planning Documents
Backend + API
Frontend
Auth & Users
Reviews & History
Midway Meeting
Core Complete
Testing
Presentation