Dissertation Plan

Title: A Web-Based Geospatial Platform for Efficient and Usable Site Analysis in Design Workflows

Introduction

Site analysis is crucial in early architectural, landscape, and urban design, where designers rely on diverse geospatial data, ranging from OpenStreetMap and environmental datasets to planning portals and topographic models, to understand potential sites. However, this data is often manually collected, cleaned, and imported into design software like AutoCAD, causing issues such as distortion, misalignment, inefficiency, and reduced quality and reliability of inputs. These challenges increase time and costs while risking poor decisions at critical early stages. This dissertation addresses these problems by developing a web-based geospatial platform tailored to designers' needs in the built environment, aiming to reduce errors, save time and cost, and enable more accurate, data-driven design decisions. Emphasising clarity, usability, and practical insight, the platform will provide an intuitive tool for site understanding through dynamic visualisations, contextual analysis tools, and intelligent data layers that support informed decision-making throughout the design process.

Aims

The platform will integrate advanced visualisation methods that abstract and reframe raw data into meaningful site insights. These will include pedestrian flow graphs (derived from GPS data and path networks), local resource clustering (such as categorised businesses or material suppliers), and topographic profiles generated by user-drawn paths. Additional layers may display terrain contours, shadow overlays, section diagrams, and data-driven summaries. A collapsible side panel will provide access to charts, network diagrams, elevation sections, and other complementary views, ensuring rich spatial understanding without overwhelming the main map. The platform will also support interactive features: users can draw lines to generate elevation profiles, click on buildings to view nearby amenities or facilities, and select zones to auto-generate usage graphs or movement patterns. Filters will allow users to highlight features of interest, such as materials suppliers, high-access areas, or flood-prone zones. These features are tailored to real-world architectural workflows and aim to reduce the time and complexity involved in conducting early-stage site analysis. This project aims to contribute to the growing field of digital design tools by creating an open, modular platform that supports designers in conducting high-quality site analysis with minimal technical overhead. It aims to demonstrate how spatial data workflows can be improved through automation and thoughtful interface design, while providing empirical evidence on the usability and performance benefits of the proposed system compared to traditional methods.

Objectives

1. Automate data acquisition from diverse open geospatial sources, including OSM, national environment and planning datasets, and elevation models.
2. Normalise and validate geospatial data to ensure semantic and geometric consistency across sources.
3. Visualise spatial datasets interactively in a browser-based platform using modern web mapping libraries.

4. Enable export to industry-standard formats such as GeoJSON and DXF, with minimal loss of data loyalty.
5. Evaluate usability and performance through user testing with design professionals, employing both qualitative and quantitative methods.

Methods

The methodology consists of five key stages:

Literature and Technology Review:

A critical review of current site analysis practices, data interoperability standards, and relevant research in HCI (Human-Computer Interaction), cartography, and digital design tools.

Data Acquisition and Processing Pipeline:

Development of a reproducible pipeline for downloading, cleaning, and formatting spatial data from open sources such as Overpass API, government data portals, and environmental databases, using Python.

Iterative Platform Development:

The core prototype will be developed using modern web technologies including MapLibre GL JS. Key features will include interactive map rendering, data layer toggles, attribute querying, spatial annotation, drawing and measurement tools, and a responsive, intuitive web interface.

Export and Integration Features:

The platform will include modules to export spatial data in formats directly usable by design software (GeoJSON and DXF), as well as documented workflows for integration with QGIS and AutoCAD.

User Testing and Evaluation:

Design professionals will participate in scenario-based testing and usability studies (e.g. System Usability Scale, think-aloud interviews) to assess improvements in task efficiency, error rates, and user satisfaction.

Performance targets include fast map load times (e.g. under 2 seconds), a System Usability Scale (SUS) (e.g. a score of 70 or above) and clear qualitative evidence of improved site understanding.

Deliverables

Deliverables include a working prototype of the web-based platform, reproducible data pipelines, sample export scripts or plugins, well-documented source code, and a technical report detailing system architecture, design decisions, testing protocols, and results. A critical reflection on the platform's limitations and future directions will also be included.

Potential risks have been identified and mitigation strategies planned. Large datasets may affect map performance (medium risk), which will be addressed through optimisation techniques such as vector tiling and lazy loading. Export compatibility issues (medium risk) will be managed through comprehensive testing and fallback formats. Limited availability of testers (low to medium risk) will be mitigated through early recruitment and flexible scheduling.

Desirables and potential future scope

A number of advanced or speculative features are desirable for future development or if time and resources permit. Artificial Intelligence could enhance insights and automate analysis through clustering algorithms (e.g., K-means) to group local resources, pathfinding algorithms to identify optimal walking routes considering sunlight, slope, or accessibility, and summarisation tools to generate auto-tagged site overviews from metadata. Rule-based logic might automatically flag high-access concern zones near schools or steep terrain. Potential future extensions include basic 3D analysis, terrain extrusions, solar simulations, and advanced visual analytics like connectivity graphs or movement pattern diagrams. A modular plugin system could enable third-party contributions of features or data layers. While the initial focus is on desktop use, mobile responsiveness and offline functionality are also possible future enhancements. These features will be scoped based on technical feasibility, performance, and user feedback, with some likely extending beyond the current project timeframe but representing valuable directions for further research and development.

Progress report

Weeks 1–2: Literature Review
In the initial phase a literature review was conducted to explore the role of web-based GIS tools in spatial planning and data visualisation. Papers which examine a web-based GIS with a variety of uses, including supporting small businesses across the world, were selected and evaluated. This review was valuable in grounding the project within broader concerns like economic inclusion, sustainability, and digital transformation, and it demonstrated how technical tools can have a socially meaningful impact.

Weeks 3–4: Proposal Drafting and Data Research
During this period, the initial project proposal was drafted outlining aims, objectives, and anticipated risks. OpenStreetMap (OSM) was studied in depth, learning about its structure (nodes, ways, and relations), tagging systems, and its limitations which highlighted the absence of vertical (z-axis) data. Elevation data sources were explored to compensate for this, including OpenTopoMap, Copernicus DEM, SRTM, and Mapbox Terrain-RGB.
In parallel, front-end development tutorials (HTML, CSS, JavaScript) ensured up to date technical know-how. Existing tools were explored and compared including Cadmapper, Digimap, GridReferenceFinder, FreeMapTools, 15mincity.ai, and wireframes for the platform's interface were drafted. These exercises helped shape a user-focused design direction, emphasising interactivity, simplicity, and task-based map views for early-stage planning workflows.

Week 5: Base Map Development
Using Leaflet JS, an interactive map was developed centring on Canterbury, incorporating layers for listed buildings and tree data (via GeoJSON). Error handling was implemented after issues to improve user feedback when data fails to load, and basic UI controls, including a checkbox-based layer toggle menu and hover effects for interactivity, were implemented. Listed buildings included popups with name, grade, and Historic England links. However, polygon rendering issues emerged as many features appeared as small triangles or misaligned shapes, prompting a reconsideration of how features should be visually represented (points vs polygons). Tree data was underdeveloped in the popups and spatial

clustering was considered. The feedback at this stage was to investigate other platforms, especially for rendering performance, scalability, and styling flexibility.

Week 6: Rendering Platform Evaluation
Four rendering platforms were researched and tested: Leaflet, MapLibre GL JS, Cesium JS, and Deck.gl, using a set of evaluation criteria including:

- Performance and load time
- Rendering style (2D, 3D support)
- Interactivity (e.g. popups, hover states)
- Customisation options
- Code complexity
- Mobile responsiveness
- Community support and documentation

Leaflet was found to be simple and well-documented, but its performance degraded with large datasets and it lacked native 3D or vector tile support. Cesium JS and Deck.gl offered high-performance 3D and advanced visual analytics respectively, but were overly complex and resource-heavy for my application. MapLibre GL JS stood out as the most balanced option, offering hardware-accelerated rendering, native vector tile support, smooth zoom, scalable styles, and optional 3D building extrusion, all with strong documentation and community usage.

Feature	**MapLibre GL**	**Leaflet**
Rendering tech	WebGL, hardware accelerated, smooth	Canvas/SVG, less smooth
3D & extrusions	Built in support for 3D buildings & terrain	Mostly 2D, limited 3D plugins
Performance with lots of data	Handles thousands of features smoothly	Can slow down with many features
Vector tiles support	Support for vector tiles and styles	Needs plugins for vector tiles
Customisation	Style based with JSON styles	Plugins, very flexible
Coding Complexity	Steeper, requires understanding of vector tiles and styles	Easier, lots of tutorials and plugins
Community	Growing, but smaller than Leaflet	Huge, very mature

The rendering of the tree dataset was tested in both Leaflet and MapLibre GL. While Leaflet was easier to set up, MapLibre handled the same data more efficiently, with smoother performance and greater visual fidelity. Using MapTiler, vector tile styling was also explored including custom base maps (e.g. dark mode, satellite) and dynamic theming, all of which showed clear advantages in terms of visual clarity and user experience.

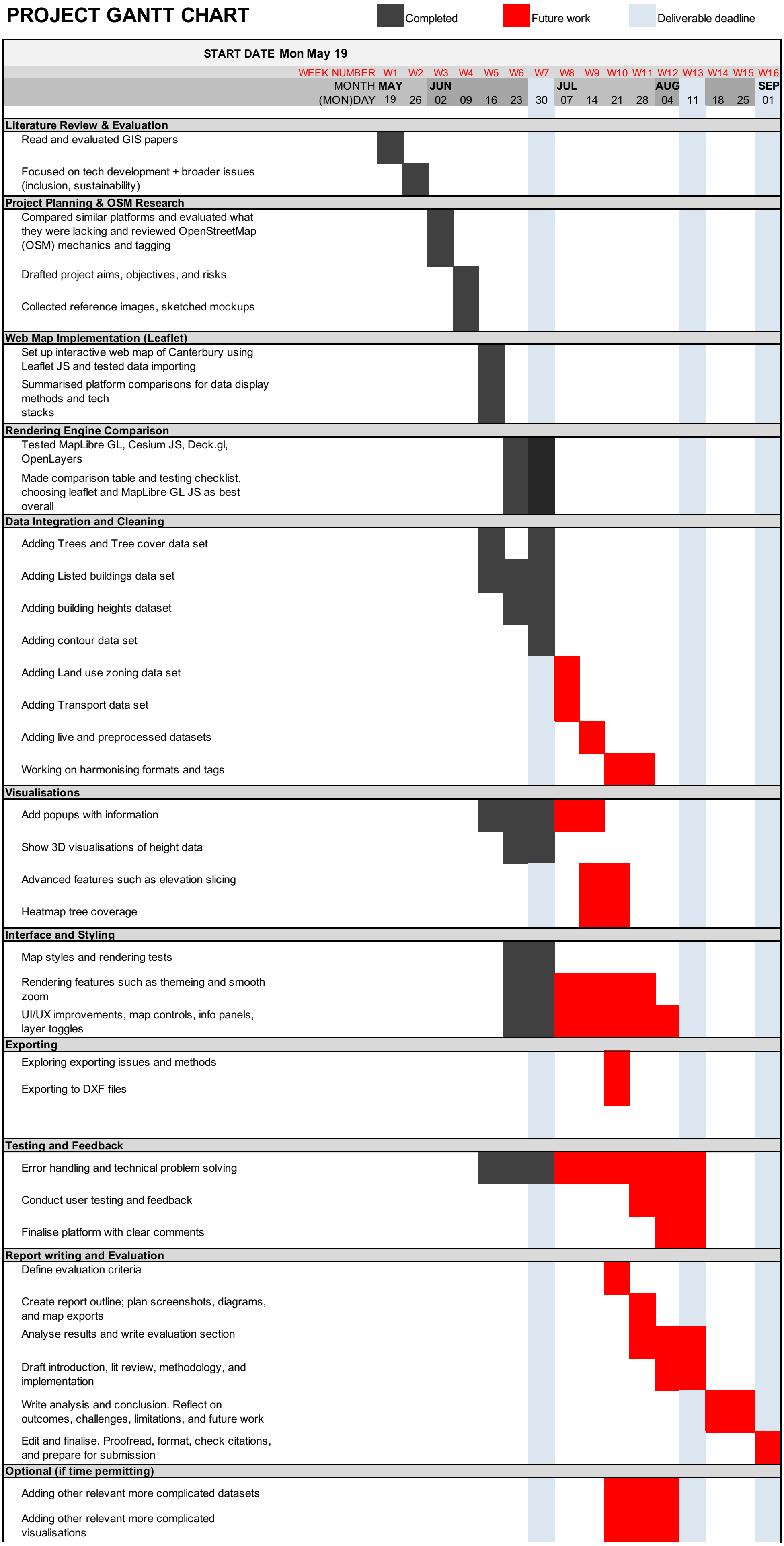
Decision-Making: Rendering Platform
The decision to move from Leaflet to MapLibre GL JS was grounded in practical testing and academic evidence. MapLibre GL is increasingly used in open-source urban planning and mobility platforms, and it offers future-proof compatibility with evolving data formats (e.g., vector tiles, WebGL styling). Despite its steeper learning curve, the performance gains, styling flexibility, and 3D capabilities were compelling advantages.

Errors and Challenges Identified

- Initial polygon rendering issues in Leaflet reduced the interpretability of listed buildings, highlighting the need for either reprocessing data or switching visualisation modes (e.g from polygon to point).
- Code refinement and adjustment needed when transitioning from Leaflet to MapLibre GL, which introduces potential integration delays.
- Some data layers (e.g elevation) require API calls or preprocessing, increasing complexity for live interactivity (e.g elevation slicing).

Timescale and Planning:

PROJECT GANTT CHART



Sprint	Date (2025)	Goal	Subtasks	Deliverable	Notes
Sprint 1	June 2–15	Conduct literature review and frame VIA/LCA context	Review literature on GIS + LCA/VIA; Evaluate key paper; Note visual analysis needs	Literature summary and project context outlined	Sets theoretical foundation for project scope
Sprint 2	June 16–29	Draft proposal, research data sources, and mock UI	Draft proposal; Study OSM mechanics; Research elevation data; Create UI mockups	Project proposal complete; design mockups prepared	Ensure realistic goals and dataset access
Sprint 3	June 30–July 6	Build base map in Leaflet and load initial datasets	Set up Leaflet map; Load listed buildings + trees; Add error handling; Basic UI	Interactive base map functional; datasets visualised	Foundation for all later visual tools
Sprint 4	July 7–13	Compare renderers and move to MapLibre GL JS	Test MapLibre, Cesium, Deck.gl; Compare renderers; Choose MapLibre GL; Begin migration	Renderer chosen and integrated with base map	Supports smoother rendering and scalability
Sprint 5	July 14–20	Integrate VIA/LCA datasets (e.g. land use, zoning, tree cover)	Add tree data, zoning, contours; Clean and normalise formats; Ensure visual clarity	Platform displays landscape data relevant to VIA/LCA	Main content layer for landscape analysis
Sprint 6	July 21–27	Implement elevation profile tool and scenario toggles	Load DEM; Create draw-to-profile tool; Add tree scenario toggle in UI	Elevation profile and scenario toggle working in UI	Key for visual impact assessment scenarios
Sprint 7	July 28–Aug 3	Build viewpoint/photo markers and optional viewshed tool	Add 'Drop Viewpoint' tool; Link popup photos; Research basic viewshed methods	Perception and viewpoint tools available in map	Adds human-perception element to analysis
Sprint 8	Aug 4–10	Conduct usability testing and collect feedback	Create test plan; Recruit participants or simulate tasks; Log and analyse feedback	Usability findings collected and synthesised	Inform future improvements and assess usability
Sprint 9	Aug 11–14	Refine tool and write evaluation section	Implement feedback-based improvements; Write evaluation analysis; Final edits	Platform refined; evaluation written into corpus	Finalise structure and critical reflection
Sprint 10 (Final)	Aug 15	Submit corpus and final platform	Final proofread; Prepare figures/screenshots; Submit before deadline	Submission of report and working web map	Deadline: Aug 15 — no new features added