MSc in Computing
Advanced Software Development | Data Science

Team Project

Final Report

Magpie

Services at a Glance

Group 3

Saul Burgess	C19349793	Andreas Kraus	D23125112
Kaustubh Trivedi	D23124940	Jessica Fornetti	D23124588
Anais Blenet	D22127697	Yuanshuo Du	D22125495

December 17, 2024

Table of Contents

List of Figures

List of Tables

List of Listings

0.1 Introduction

0.2 User Scenario

0.3 Technical Problem

0.4 Technical Solution

0.4.1 System Overview

0.4.2 System Architecture

0.4.3 Data Sources

0.4.4 Machine Learning

Data collection process

The Mapbox Static Images API was used to retrieve all the satellite images used in this project.

Initially, a training set of 250 satellite images was collected to train the Yolo model.

Our project has two main Python scripts, `parking_detection.py` and `parking_detection_local.py` to localize parking spots. In `parking_detection.py`, the satellite images (Mapbox Satellite) and corresponding road mask images (Mapbox Streets) are retrieved for a specific area contained within a bounding box, defined by its top-left and bottom-right coordinates, directly from the Mapbox API. From the coordinates of the specific bounding box, the center coordinates of all the images necessary to make up that area are calculated, and the images are then retrieved. While in `parking_detection_local.py`, the original parking detection script is adapted to run on a database of locally stored Mapbox Satellite images and the corresponding Mapbox Streets images, covering the entirety of Dublin city.

Extensive testing was done at all the different stages of development of the scripts to identify the optimal thresholds and parameters using test images from Mapbox, in a variety of scenarios including edge cases or cases prone to causing issues.

Yolo model used for car detection

Parking spot detection

Subsequently, to identify the parking spots in each satellite image, the cars detected by the Yolo model are classified into on the road or parked, based on a road mask.

Road mask generation

Many different iterations of the road mask were used, the main iterations and their differences are explained and showed below.

The original mask was a very simple binary mask which identified the road pixels by their color, as the majority of roads were depicted in white and darkened the rest of the pixels in the image.

Through testing and thoroughly analysing the Mapbox Streets images, motorways and national roads were discovered to be depicted in yellow or orange. Therefore, two additional color masks (for orange and yellow) were concatenated to the original binary mask through bitwise operations.

To refine the mask, and reduce misclassification errors, additional annotations contained on the Mapbox Streets images such as street names and white dotted lines which denote a multitude of things (foot paths, planter boxes, pedestrian crosswalks, football fields delimitations, etc.) were removed through morphological operations, leaving only a plain white line for each road. A number of different kernel sizes, different contour thresholds and other parameters were tested to achieve the optimal removal of the street names and additional annotations.

A different approach using Canny Edge Detection was tested, however due to the nature of the Mapbox Streets images, the edges picked up were not significant and the overall performances was much worse than the current mask at the time.

A final improvement to the mask was made to avoid certain misclassification due to the Mapbox Streets road width not correctly reflecting all the lanes of the road and therefore classifying on the road cars as parked. This issue was most prominent for motorways and national roads and was resolved by enlarging those roads using a dilation technique. Multiple different kernel sizes, different number of iterations and kernels of different sizes applied consecutively were tested to find the optimal solution.

Classifiction of cars into on the road or parked

The cars detected by the Yolo model are then sorted into on the road or parked based on the road mask previously generated, which is explained in more detail below.

The model's predictions which are lower than the confidence threshold of 0.4 are discarded as they are more likely to be misclassifications, such as chimneys which were often misclassified as cars.

A bounding box for each of the model's predictions is computed from the center pixel coordinates, the width and the height returned from the model. The overlap between the bounding box and the road pixels is calculated, and if the overlap is higher than the threshold of 0.5, the prediction is discarded, keeping only the parked cars. Different thresholds were tested as well, however overall a harsher threshold works better given that these detections are used later on for the empty parking detection.

For each of the model's predictions for parked cars, the center pixels coordinates are mapped to geographic coordinates (longitude and latitude), and the width, height, rotation and orientation based on the angle are saved.

For testing and debugging purposes, all the bounding boxes of the cars found by the model are drawn onto the satellite image, on the road cars are drawn in blue while parked cars are drawn in red as seen in Figure ...

Empty parking spot detection

Subsequently, empty parking spots are detected between parked cars in each image, based on the parked cars detected by the Yolo model and classified using the road mask. Originally 4 cases were considered, cars parked horizontally in a row or in a column, cars parked vertically in a column and parked vertically side by side. However horizontally parked in a column and vertically parked side by side were removed as they lead to too many misclassifications, namely the vertically parked side by side lead to adding many additional cars in driveways in residential areas. A number of different variations and calculation techniques were tested out, however only the final version will be explained below.

Average parking spot width and length in pixels are calculated for each image, to draw the empty spots more uniformly. Average parking spot width and length in meters are set to 3.05 as found to be the optimal value through testing accounting for the variations in size. Setting this value avoids misclassifications as previously the average width and length in meters were calculated dynamically, however in cases where the averages were lesser than 3, spots tended to overlap.

The parked cars detected are then sorted by latitude for horizontally aligned cars and by longitude for vertically aligned cars, to identify gaps between consecutive cars. For each pair of consecutive spots, the distance in meters in between them is calculated. The gap is adjusted to account for the half cars on both sides, as the coordinates of the parked car represent the center of the car. For the gaps smaller than the maximum gap threshold of 12 meters and larger than the average size of a car, where there is enough space to fit 1 or more car, and where the angle deviation is smaller than a threshold of 35 degrees, to ensure the spots are sufficiently aligned, the number of cars that fit into the gap is calculated. Based on the number of empty spots found, the coordinates

for those empty spots are calculated to be aligned with the 2 consecutive cars and then added to a list of empty spots detected.

Afterwards, duplicate spots defined as spots that overlap too much and where the distance between 2 spots is smaller than a threshold of 1 meter are removed. As well, spots coinciding with cars identified by the Yolo model, where the distance between them is smaller than 1.25 meters are removed from the empty spots found. Subsequently, empty spots that overlap with the road are filter out in a similar manner to the classification done by the road mask.

Following the detection of all the empty spots, the bounding boxes corresponding to each spot are drawn onto the satellite image in green.

Classification of all spots detected

All the parking spots detected, including the parked cars identified by the model and the empty parking spots are then classified into 3 different classes: public (on the street parking), private (residential parking) and parking lot, based on their proximity to the nearest road.

Multiple different clustering approaches were tested out to cluster spots together, such as DBSCAN, HDBSCAN and OPTICS. Overall DBSCAN performed the best and the quickest out of all the clustering algorithms, given that some images contained very few parking spots, meaning clustering was only significant in cases with a minimum number of parking spots. Through testing, the optimal hyperparmeters found for DBSCAN are `eps` : 55 and `min_samples` : 5, which allow the correct identification of the clusters, avoiding misclassifications of residential areas with many cars as parking lots, which was a common misclassification initially, when the hyperparameters and thresholds were not set correctly. `eps` refers to the maximum distance between two points for them to be considered as neighbors and `min_samples` refers to the minimum number of points required to form a cluster. HDBSCAN and OPTICS require a minimum number of samples larger or equal to 2, however given that not all images might contained at least 2 detections, clustering using those algorithms is not possible in all cases, so -1 is assigned to the spots without clusters which is the default behaviour of DBSCAN. HDBSCAN gave similar results to DBSCAN, though not as good, while OPTICS only found smaller clusters which were not as significant.

Parking spots are classified as public if the proximity to the nearest road is less than a pixel threshold of 30, otherwise the parking spots are considered private. This optimal threshold was found through extensive testing on a test set containing various edge cases. Parking spots are classified as parking lots, when clusters of 18 or more spots are identified in an image. Larger clusters are prioritized as parking lots containing less spots are not as relevant to the target user and given that the threshold of 18 spots minimizes the risk of misclassifying residential areas as parking lots.

Afterwards, clusters and classification labels are drawn onto the satellite image. The clusters are color coded and denoted as a circle at the center of the bounding box, while the classification label is written to the right of the bounding box. The private in written in red, public in green and parking lot in white.

Uploading points to the backend server

The longitude, latitude and classification of all the parking spots detected are then saved to a csv file. Potential spot duplicates are dropped, as the same car may be in multiple images given that the images are defined by their center coordinates, which causes adjacent images to overlap and share common areas. (The rightmost part of one image often overlaps with the leftmost part of another image.)

Subsequently, the parking zones and costs associated to each parking spot found are added to the csv file. Dublin defines different parking zones by color, based on their high to moderate demand which changes the price per zone as seen below in Figure ...

The csv file containing the longitude, latitude, classification label, parking zone and parking cost is then uploaded to the PostgreSQL database, in the backend server using the `send_points.py` script.

Evaluation of each submodule of the parking detection model

Each major section of the parking detection model was evaluated individually in order to access the overall performance. The results will be presented and analysed for each section below.

Evaluation of the Yolo model

Evaluation of the classifiction of cars into on the road or parked

The classification of cars into on the road or parked by the road mask is evaluated in the `evaluate_road_mask.py` Python script. The classification is evaluated on the following metrics: Average Intersection over Union (IoU), Balanced Accuracy and Precision, Recall, F1 Score, Accuracy, Specificity per class. These metrics are saved in a csv file for each test image as well as the overall average metrics on the entire test set.

A test set of 50 images was carefully selected to include edge cases and difficult cases and then consistently labelled in the labelling software Label Studio. The labels are drawn without rotation to ensure the maximum overlap, as the labels are exported to `txt` format and include only the pixel coordinates, width and height of the bounding boxes annotated.

The overall results on test set are presented below in Table 1

Metric	Parked	Road
Average IoU	0.66	0.66
Average Balanced Accuracy	0.58	0.58
Average Precision	0.70	0.90
Average Recall	0.75	0.67
Average F1 Score	0.59	0.54
Average Accuracy	0.69	0.64
Average Specificity	0.74	0.79

Table 1: *Performance Metrics for Road Mask Classification*

Analyse metrics + Mention common problems. Reference similar results in object detection

Some images from the test set can be seen below, in Table 2 The predicted parked cars are drawn in red while the true labels are drawn in light pink. The predicted on the road cars are drawn in blue while the true labels are drawn in cyan.

Evaluation of the empty parking detection

The empty parking detection logic is evaluated in the `evaluate_empty_parking_detection.py` Python script. Expliquer script, comment l'orientation est calcule de facon automatique. The classification is evaluated on the following metrics: Average IoU, Average Precision, Average Recall, Average F1 Score, Average Orientation Accuracy, Average Spot Detection Ratio (SDR), Average Spot Detection Error (SDE), Average False Positive Rate (FPR), and Average False Negative Rate (FNR). Explain metrics and how the additional ones are calculated/ what they represent. These metrics are then saved in a csv file for each test image as well as the overall average metrics on the entire test set.

Likewise a test set of 50 images was carefully selected and consistently labelled in Label Studio, without rotation to ensure the maximum overlap. Achieving a high IoU was a significant challenge, specifically in this section, as the positioning of the spots can be difficult to label. To remedy this issue, the IoU threshold was lowered to 0.35, as there was still significant overlap visible when analysing the test images. The problems linked to overlap, is equally due to the differences in orientation as the true labels and the predictions are not calculated in a similar manner. (Expliquer mieux et dire les differences)

The overall results on test set are presented below in Table 3

Analyse metrics + Mention common problems. Reference similar results in object detection

A few images from the test set are displayed below, in Table 4 The predicted empty parking spots are drawn in green while the true labels are in light pink.

Evaluation of the classification of the spots detected into private, public and parking lot

The classification of the spots detected into private, public and parking lot is evaluated in the `evaluate_classification_of_sp` Python script. Dire changement, like all the spots include everything... The classification is evaluated on the following metrics: Average IoU, Balanced Accuracy and Precision, Recall, F1 Score, Accuracy, Specificity per class. These metrics are saved in a csv file for each test image as well as the overall average metrics on the entire test set.

Similarly, a test set of 50 images was carefully selected and consistently labelled in Label Studio, without rotation to ensure the maximum overlap.

The overall results on test set are presented below in Table 5

Analyse metrics + Mention common problems. Reference similar results in object detection

A few images from the test set are shown below, in Table 6 The predicted public spots are drawn in green while the true labels are in dark green. The predicted private spots are highlighted in red while the true labels are in light pink. The predicted parking lot spots are shown in blue while the true labels are in cyan.

Relire tout and add more, ajouter totues les figures. Expliquer parking zones and costs mieux en ajoutant la figure du pdf. Verifier que j'ai explique toutes les fonctions. Refer to each figures, like say as seen in figure x ...

0.4.5 Frontend

Frontend Architecture and Implementation

In the development of this project, we adopted a modern web development stack that incorporated a combination of cutting-edge technologies aimed at optimizing performance, scalability, and maintainability. The integration of these technologies facilitated the creation of a highly efficient and user-friendly web application. This section provides a comprehensive overview of the key technologies used in the frontend of the application, outlining their individual roles and how they collectively contributed to the success of the project.

1. **React**
   React is a widely used and powerful JavaScript/TypeScript library that is primarily designed for building user interfaces. It was chosen for this project due to its ability to facilitate the development of dynamic and responsive web applications. In conjunction with **Next.js**, React allowed for the implementation of a component-driven architecture. This architectural approach enables the reuse of individual UI components throughout the application, which not only streamlined the development process but also ensured greater maintainability in the long term. By breaking down the user interface into smaller, reusable components, we were able to manage and scale the application more effectively. React's virtual DOM mechanism also ensured efficient rendering of changes to the UI, contributing to an enhanced user experience by minimizing the amount of time required to update the interface.
   Moreover, React's active community and vast ecosystem of libraries and tools made it easier to integrate other technologies and features into the application, further improving its overall functionality and performance. As a result, React played a crucial role in enabling a seamless, interactive experience for users, while also enhancing the maintainability and scalability of the codebase.
2. **Next.js**
   Next.js is a robust React framework that extends the capabilities of React by offering features such as server-side rendering (SSR) and static site generation (SSG). These features proved to be essential for optimizing the application's load times, which was critical to the project's success. Server-side rendering allows the server to generate the HTML for the page on each request, as opposed to client-side rendering where the browser must first download and execute JavaScript to generate the page. This process significantly reduces the time required for the page to be displayed to the user, resulting in a faster and more efficient application.

Additionally, Next.js' static site generation capabilities allowed us to pre-render pages at build time, meaning that the application could serve static HTML files for commonly accessed pages, further enhancing performance. The combination of SSR and SSG enabled a smooth and responsive user experience.
Next.js also provided a robust routing system that made it easier to manage the application's navigation. The framework's file-based routing system allowed for simple, declarative routing, which contributed to the overall maintainability of the project. Next.js' out-of-the-box support for features like code splitting and automatic optimization ensured that the application remained highly performant, even as it grew in complexity.

3. **TailwindCSS**
   For styling the application, **TailwindCSS**, a utility-first CSS framework, was chosen for its ability to enable rapid UI development. Unlike traditional CSS frameworks, which rely on pre-defined UI components and styles, TailwindCSS provides low-level utility classes that can be combined to build custom designs. This approach allowed for greater flexibility in styling the application, as developers were able to apply precise and granular control over the UI without the need for writing custom CSS from scratch.
   One of the key advantages of using TailwindCSS was the speed with which we could implement and modify styles. By utilizing utility classes, we were able to quickly experiment with different design configurations and make adjustments on the fly. This approach also minimized the need for additional CSS files, reducing the overall complexity of the styling process.
   In addition to TailwindCSS, we incorporated several tools to enhance the framework's capabilities. **Tailwind-merge** was used to intelligently combine conflicting utility classes, ensuring that styles were applied consistently throughout the application. **TailwindCSS-animate** provided utility classes for adding animations, which helped bring dynamic visual elements to life without needing to write custom animation logic. These tools, when used in tandem, allowed for more advanced styling and complex animations while maintaining the simplicity and ease of use that TailwindCSS is known for.
4. **Mapbox GL and React Map GL**
   The integration of interactive maps and advanced visualization features played a key role in the functionality of the project. For this purpose, we utilized **Mapbox GL** and **React Map GL**. **Mapbox GL** is a powerful mapping library that provides advanced capabilities for rendering interactive maps with smooth transitions and animations. It is built to handle large datasets, enabling the display of dynamic geospatial information with high performance and precision. Mapbox GL was used to power the core mapping functionality of the application, allowing users to interact with maps seamlessly.
   To integrate Mapbox GL with our React application, we used **React Map GL**, a library that provides a simple wrapper around Mapbox GL, making it easier to embed and control interactive maps within React components. React Map GL allowed us to leverage the full potential of Mapbox GL while maintaining a consistent and efficient React-based architecture.
   In addition to basic mapping functionality, we integrated **Deck.gl** for advanced data visualization. Deck.gl is a framework that enhances Mapbox GL by enabling the display of complex data visualizations on top of interactive maps. This was particularly beneficial for our project, as it allowed us to visualize large datasets in an efficient and interactive manner, providing users with a rich and engaging experience.
5. **ShadCN/UI**
   To streamline the development of UI components, we incorporated **ShadCN**, a component library built on top of **Radix Primitives**. ShadCN combines the flexibility of Radix UI with a more opinionated design system, tailored for projects that require custom styling with the power of TailwindCSS. Radix Primitives offers a set of low-level UI components that are highly customizable, accessible, and composable, making them ideal for building complex user interfaces.
   By using ShadCN, we were able to access a suite of pre-built, customizable components that adhered to accessibility standards, ensuring that our application was usable by all users, including those with disabilities. ShadCN's integration with Radix Primitives allowed us to quickly implement accessible and customizable components, reducing the need for additional development and ensuring consistency across the application. The combination of ShadCN's flexible components and TailwindCSS allowed us to build a visually appealing and highly functional user interface with minimal configuration. Furthermore, ShadCN's tight integration with accessibility guidelines ensured that the application met the required standards for usability, providing an inclusive experience for all users.
6. **Radix Primitives**
   Radix Primitives played a crucial role in the design and development of our user interface components. Radix Primitives provides a set of foundational, high-quality UI components that serve as the core building blocks for any application. These components are unstyled and highly customizable, giving developers full control over the look and behaviour of the elements while ensuring accessibility and functionality out of the box.

The core of Radix Primitives includes essential UI elements such as buttons, modals, popovers, sliders, tabs, and other interactive components. These primitives are designed to be flexible and composable, allowing developers to combine them in various ways to create complex UI patterns. The components are built to be accessible, ensuring that the user interface remains usable for all users, including those with disabilities, without requiring additional development effort to meet accessibility standards.
One of the standout features of Radix Primitives is its focus on accessibility. The components are designed with built-in features like keyboard navigation, screen reader support, and proper focus management. This allows the application to be fully functional and accessible across a wide range of devices and assistive technologies, without the need for custom implementation of these features. As accessibility is a fundamental aspect of modern web development, Radix Primitives ensured that the project adhered to the highest standards for inclusive design.
In addition to accessibility, Radix Primitives allows for extensive customization. While the components come with sensible defaults, they are unstyled, which provides the flexibility to apply custom styles that match the overall design language of the application. This level of customization allowed the development team to ensure that the UI components blended seamlessly with the rest of the application's design. By combining Radix Primitives with **TailwindCSS**, we could easily customize the components using utility-first CSS classes, speeding up the styling process and ensuring consistency across the entire application.
The composability of Radix Primitives also played a significant role in the project's development. The library is designed to allow developers to create complex user interface patterns by combining simple components in flexible ways. For instance, we could easily integrate modal dialogs, dropdown menus, and tooltips to form cohesive UI patterns that enhanced the user experience. This approach not only saved development time but also reduced the likelihood of introducing bugs, as we could rely on well-tested, standardized components rather than building these features from scratch.
By leveraging Radix Primitives, we were able to focus on creating a polished and accessible user interface while avoiding the complexity of building low-level components. The ability to use these pre-built, flexible primitives allowed us to deliver a high-quality user experience with minimal overhead. Radix Primitives was an essential tool in ensuring that the application was both user-friendly and accessible, all while maintaining the flexibility to customize and extend the components as needed for the project.

The integration of these technologies into the frontend of the project provided a solid foundation for building a high-performance, scalable, and user-friendly web application. From the server-side rendering capabilities of Next.js to the highly customizable components offered by ShadCN, each technology played a critical role in the overall success of the project. The use of React and Next.js allowed for a modular and efficient development process, while TailwindCSS and its associated tools streamlined the styling process and enabled rapid UI development. The incorporation of Mapbox GL and Deck.gl brought advanced mapping and data visualization capabilities to the application, making it more interactive and engaging for users. Together, these technologies enabled us to deliver an application that met high standards of code quality, user experience, and performance, ensuring its success in achieving the project's objectives.

Client State Management

Introduction

Effective client state management is a crucial aspect of the **Magpie** frontend, as it ensures a seamless and consistent user experience. Given the dynamic nature of the application, with interactive elements such as maps, search filters, and property details, it is essential to maintain synchronization between different UI components while preventing errors and inconsistencies. Managing state efficiently is particularly important when handling large datasets that need to be presented in a user-friendly manner across various parts of the application.

React Context

To achieve this, the application employs state management techniques that allow for centralized control of the application's state. One of the core tools used for this purpose is **React Context**, which provides a simple way to share state across multiple components without having to pass data manually through props. React Context enables the application to manage the user's preferences, search filters, and saved properties in a global state, ensuring that these settings are accessible across various parts of the application. This method of state management is particularly useful for dynamic features, such as filtering search results based on user input or retaining user selections when navigating between different pages or sections of the platform.

Redux

In addition to React Context, the use of more advanced state management libraries such as Redux can be considered for larger-scale applications that require more complex state handling. Redux facilitates a unidirectional data flow and provides a centralized store that allows the state to be updated in a predictable manner. With Redux, the application can ensure that all components are working with the most up-to-date information, whether that involves real-time data from maps or user-selected property preferences. Redux also allows for more complex actions, such as asynchronous data fetching, and can be used in conjunction with middleware like Redux Thunk for handling side effects, further enhancing the application's performance and stability.

Consistent User Experience

Client state management also ensures that the data displayed to the user remains consistent. For instance, when a user interacts with a map, the system must track the map's current state and the filter options that the user has selected. By maintaining this state across different components, users can switch between different views (such as property details and map views) without losing their previous selections. Additionally, state management ensures that updates to the user interface, such as property filtering or adding a property to the favourites list, are reflected across the entire application in real-time, maintaining data integrity and providing users with a smooth browsing experience.

Conclusion

Overall, client state management plays a vital role in improving the usability and responsiveness of the frontend, reducing redundancy and complexity by centralizing control over the application's data. This centralized management allows for easier maintenance and scalability as the application evolves and new features are added.

UI Walkthrough

Landing Page

Introduction

Upon accessing the landing page, users are greeted with a clean and modern layout that introduces them to 'Magpie' - a geographical information service providing a glance at available public services in Dublin. The layout is designed to be visually appealing, with prominent calls to action and easily navigable sections.

Header Section

- **Navigation Bar:** At the top of the page, the navigation bar features the **Magpie logo** on the left side, offering a simple and recognizable brand identity. To the right, navigation links for **About**, **Features**, **Use Cases**, and **Get Started** provide quick access to key sections. Additionally, a **Sign Up** button stands out in black with white text for clear calls to action.
- **Mobile Menu:** For mobile views, a hamburger menu (three horizontal lines) appears, ensuring that the navigation remains compact and accessible on smaller screens.

Hero Section

- **Headline:** The page features a bold, eye-catching headline that reads 'Magpie' in a gradient text style, followed by the tagline 'Services at a Glance.' This is a clear and engaging introduction to the site's purpose.
- **Subheading:** A short description reinforces the value proposition: 'Access Dublin's most comprehensive database of public services; quickly, easily, and reliably.' This emphasizes the platform's utility and target audience.
- **Primary CTA:** A large button titled 'Get Started' invites users to sign up or log in, accompanied by an icon of a map pin to add a visual cue to the call to action.
- **Secondary CTA:** A 'Learn More' button encourages users to explore more details about the platform. It's designed to stand out but with a lighter visual treatment compared to the primary CTA.

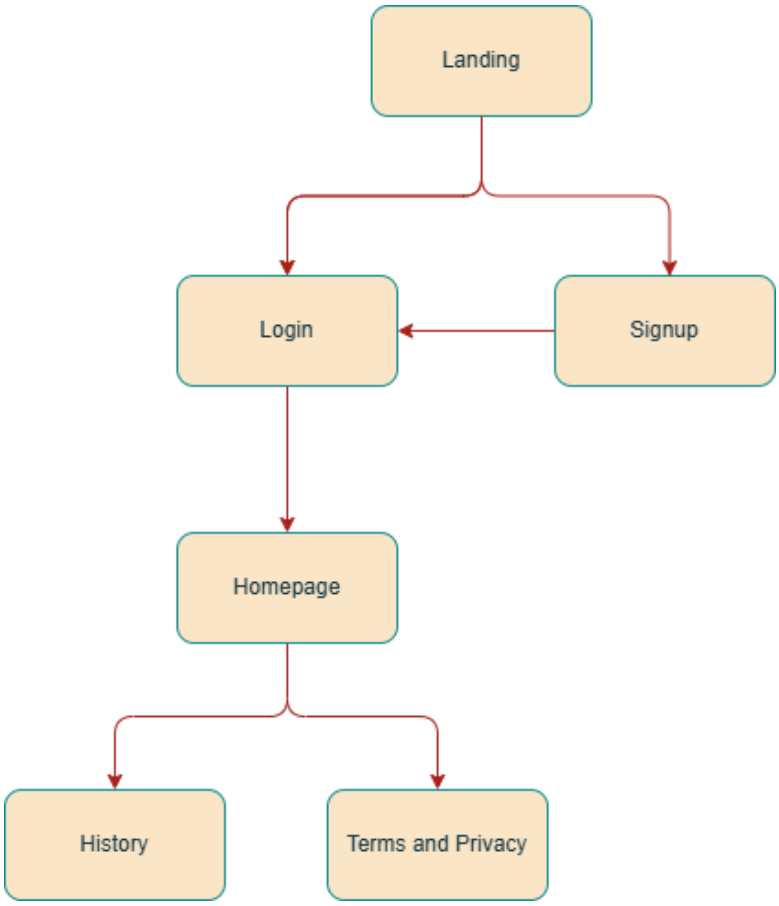

Figure 1: *Site map - magpie*

Figure 2: *Landing page - Header*

About Section

What is Magpie?: This section introduces **Magpie** as a **geographical information service**, explaining its functionality and target users. It emphasizes its usefulness for urban planners, residents, and other stakeholders by providing interactive maps, smart search capabilities, and detailed analysis tools.

How Magpie Uses Machine Learning: This section explains the use of **machine learning** to analyze satellite images and detect parking spaces. It provides insight into the model used (YOLOv8 OBB), detailing how it detects cars and identifies available parking spaces. A breakdown of the process offers technical depth while remaining accessible to a general audience.

Features Section

Unique Features - This section highlights the core capabilities of Magpie:

- **Interactive Map Visualization:** The ability to search, filter, and visualize services through an interactive map.
- **Smart Search & Analysis:** Provides detailed insights with multi-layer data visualizations.
- **Professional Tools:** Features designed to export reports and analyze historical trends.

Each feature is represented by a card that contains an icon, a title, and a brief description. Hovering over these cards gives users more details, encouraging them to explore further.

Use Case section

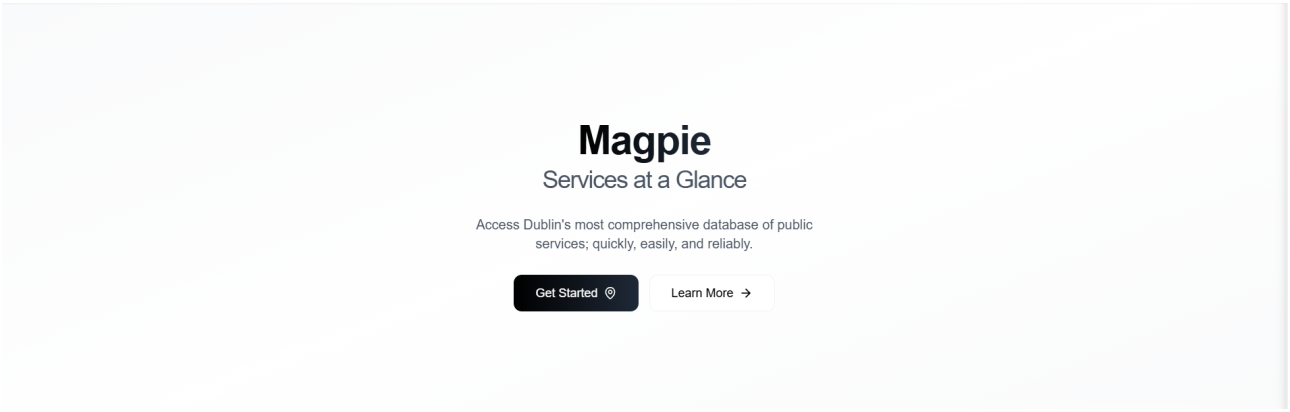

Figure 3: *Landing page - Hero section*

Magpie Works for Everyone: This section presents specific use cases for different user types:

- **Urban Planners & City Authorities:** Features such as optimizing infrastructure, analysing service distribution, and generating reports.
- **General Users & Residents:** Services to find nearby amenities, access transport options, and plan efficient routes.

Call To Action (CTA) Section

- **Ready to Transform How You Plan?:** The page concludes with a strong call to action, urging users to **sign up** and start using Magpie. This section reiterates the platform's value in making data-driven decisions, featuring a large, attention-grabbing button labelled 'Sign Up Now.'
- **Gradient Background:** The footer section features a subtle gradient background, reinforcing the design's sleek and modern aesthetic.

The landing page for **Magpie** is designed to effectively introduce the platform's features, functionality, and benefits through a series of engaging sections. The page focuses on presenting Magpie as a user-friendly tool for urban planners and residents, with clear calls to action and visually appealing design elements. It balances technical information with an approachable layout, ensuring that both potential users and technical stakeholders can easily understand its offerings.

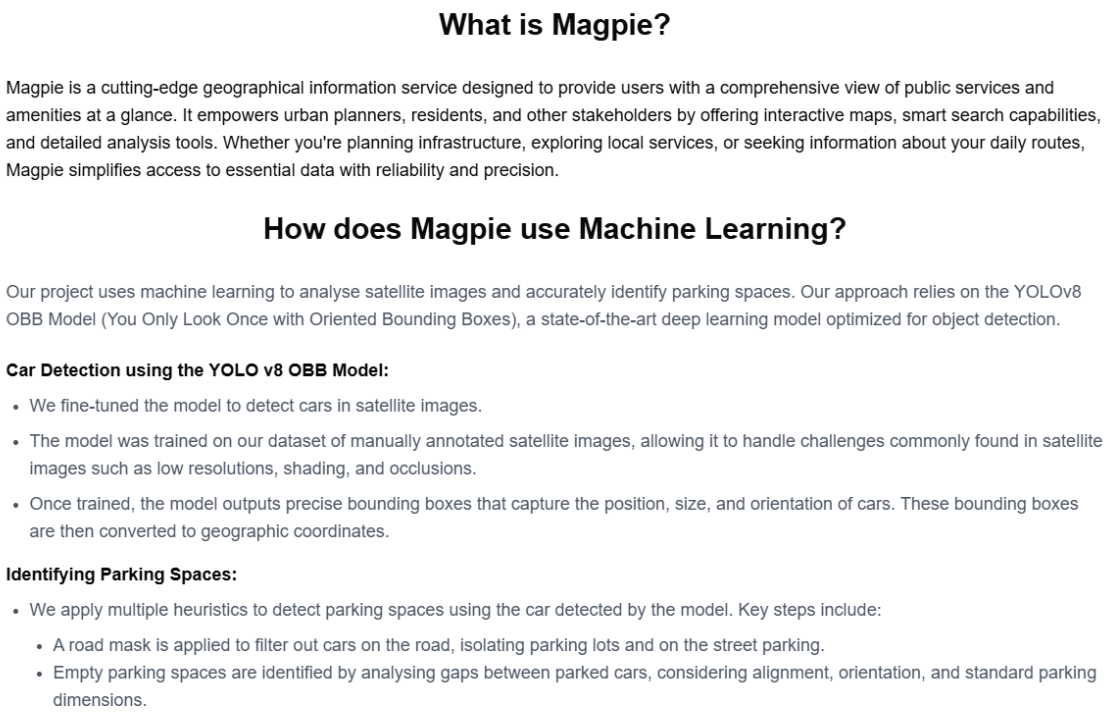

Figure 4: *Landing page - About section*

Signup Page

The **Signup** page serves as the gateway for users to create an account on Magpie. It is designed with a straightforward layout to guide users through the process of entering their personal information securely.

Header Section

The header section plays a crucial role in establishing the identity of the page.

- **Magpie Logo:** Positioned prominently at the top of the page is the **Magpie logo**, providing immediate brand recognition. The logo's size and placement are intentional to make it easy for users to identify the platform and navigate back to the homepage, should they need to.
- **Tagline:** Below the logo, the tagline **'Services at a Glance'** emphasizes the platform's purpose, providing a succinct description of its functionality-offering users quick access to available public services in Dublin.
- **Call to Action:** The headline in the header invites users to **Create your Magpie Account** in large, bold text, making the goal of the page clear and guiding users toward account creation. This is an important call to action that users are encouraged to follow.

Form Inputs

The heart of the Signup page is the **form inputs**, where users input their details to create an account.

- **Required Fields:**
  - **First Name** and **Last Name** fields ensure that the platform has accurate personal information about the user. These fields are essential for building a profile that can be used for various personalization features across the Magpie platform.
  - **Email Address** is also required, which is important for account verification, password recovery, and communication with the platform. Users are reminded that this email will be used for correspondence related to their account.
  - **Username** allows the user to choose a unique identifier for logging into the platform. This is an important step in personalizing the account, as the username will be used alongside the password to access the Magpie dashboard.

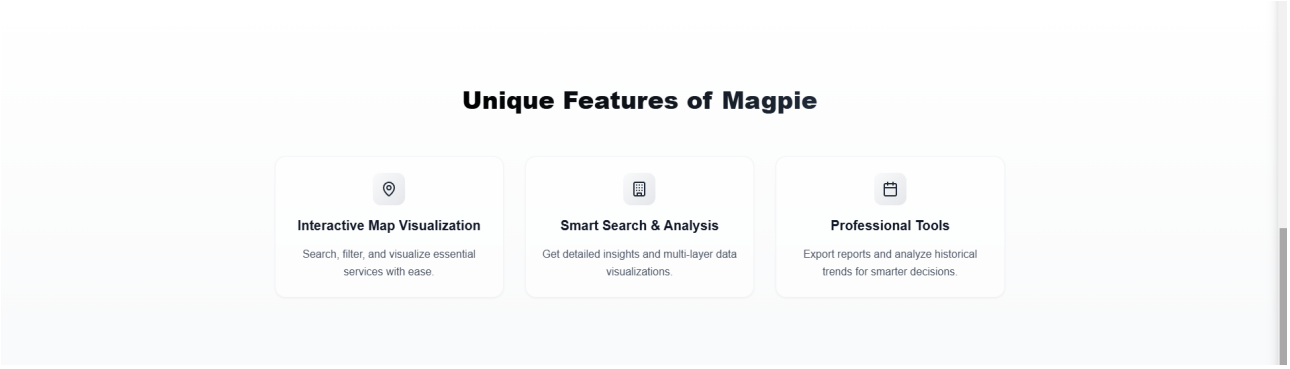

Figure 5: *Landing page - Features section*

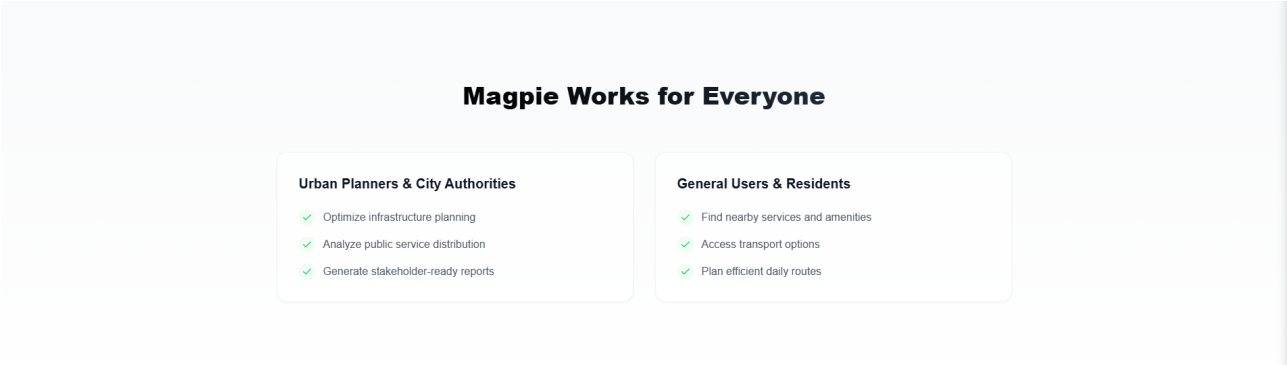

Figure 6: *Landing page - Use Case section*

- **Password Fields:**
  - The **Password** field is critical for securing the account, and users are encouraged to choose a strong, secure password.
  - A **Confirm Password** field is included for verification to reduce errors and ensure that users input their password correctly.
  - **Password Visibility Toggle:** A key feature for convenience is the **toggle** to show or hide the password as users type. This helps users ensure they've entered the correct password without exposing it to others around them.

Call To Action

- **Sign Up Button:** The page features a prominent **Sign Up** button in the center of the form. The button is large and visually distinct to stand out, ensuring that users know where to click to complete their registration once all fields are filled. The button uses an eye-catching color, making it easy to find.
- **Link to Log In:** A secondary, less prominent link underneath the main form invites users who already have an account to **Log in**. This ensures that returning users do not feel lost and can quickly find their way to the login page.

Terms and Privacy

- **Consent to Terms:** At the bottom of the form, there is an important reminder for users that by signing up for the platform, they agree to Magpie's **Terms and Privacy Policy**. This ensures that users are informed about the platform's legal agreements before proceeding.
- **Link to Terms:** A clickable link is provided for users to read the **Terms and Privacy Policy** in detail. The link opens the full document, so users have access to the complete legal information regarding their data, privacy, and the platform's use.

Overall Design and Experience

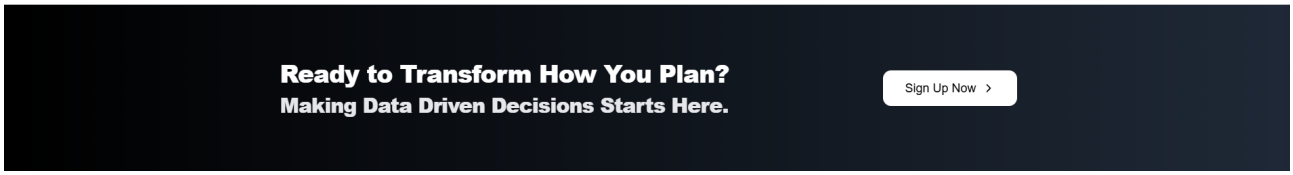

Figure 7: *Landing page - CTA section*

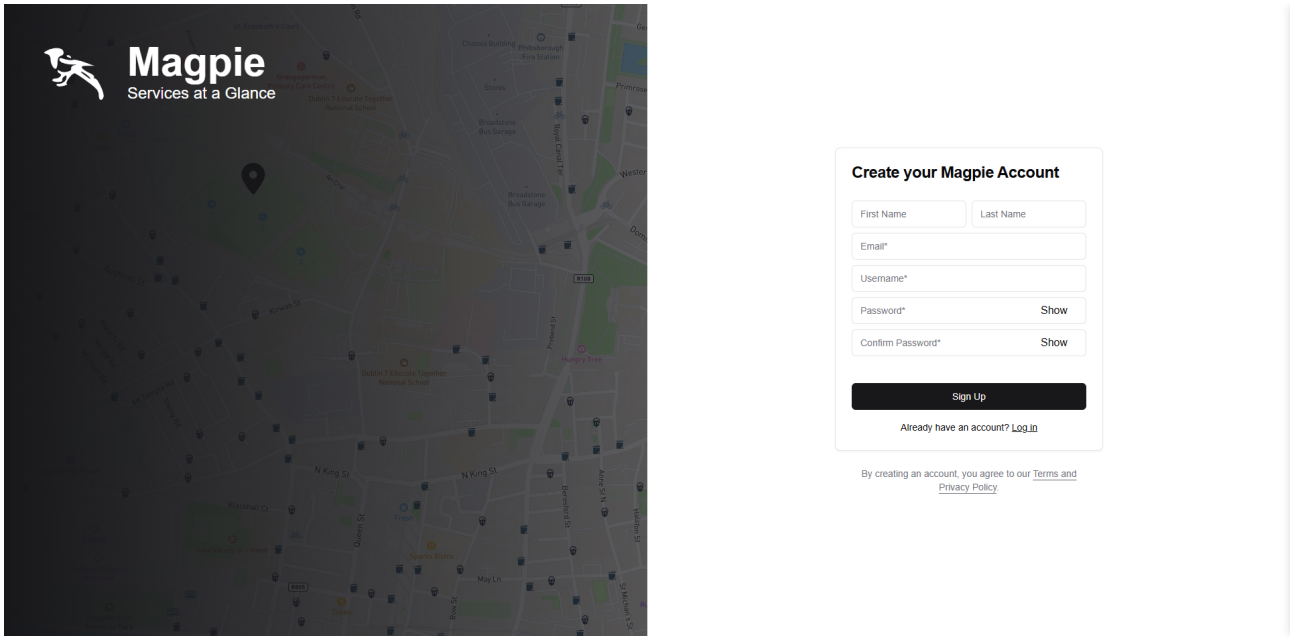

Figure 8: *The Signup Page*

The design of the signup page is intuitive and responsive, providing a smooth experience for users on both desktop and mobile devices. It is minimalistic, focusing on the essential information without distractions, while the color scheme and layout are optimized for clarity and usability. The goal of the page is to simplify the signup process, ensuring that users can easily create their accounts while being informed about the necessary legal terms.

This page ensures that users feel confident about creating an account by making the process as straightforward as possible while providing clear information about data privacy and platform policies. The inclusion of visual cues like the password visibility toggle and clearly marked sections for each field enhances usability, contributing to a positive user experience.

Login Page

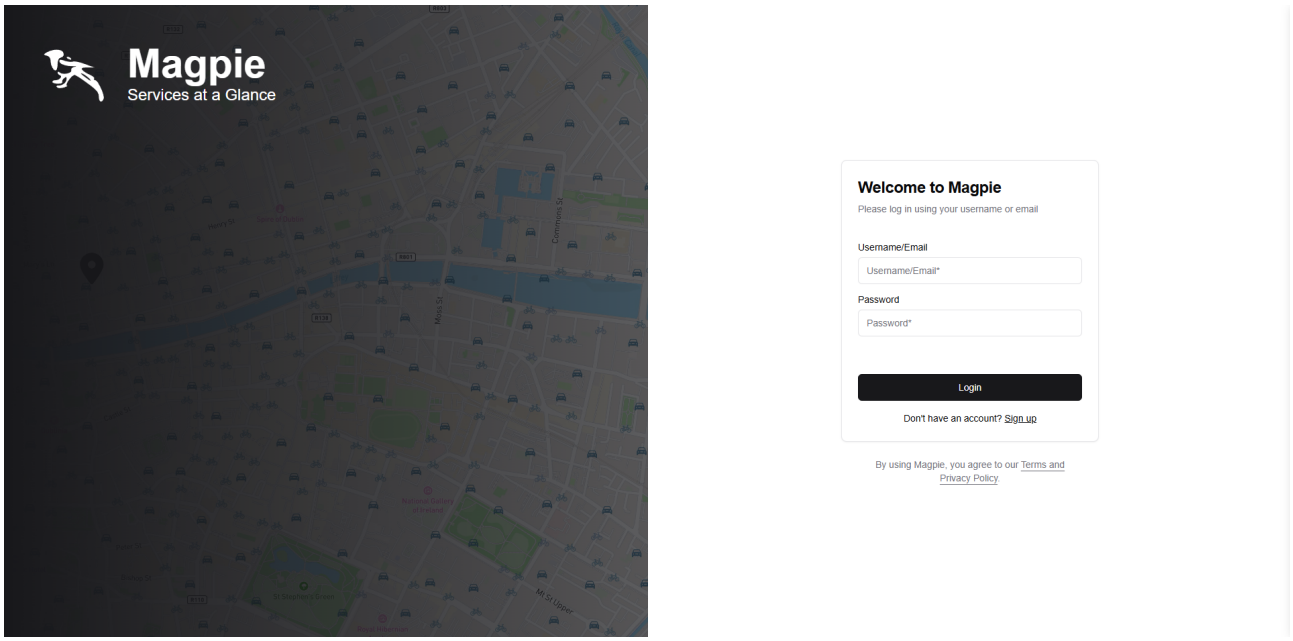

Figure 9: *The Login Page*

The **Login** page is designed to provide a smooth and efficient experience for registered users to access their Magpie accounts. This page ensures that returning users can log in quickly and securely, reinforcing the brand's focus on user experience and data protection.

Header section

The header section plays a crucial role in maintaining consistent branding.0

- **Magpie Logo:** Just like on the signup page, the **Magpie logo** is prominently displayed at the top of the page. The logo serves to create immediate brand recognition and assures users they are on the official platform. It is positioned alongside the tagline **'Services at a Glance'**, which reinforces the site's purpose—providing users with an easy and visual way to view available public services across Dublin.
- **Visual Consistency:** The design maintains visual consistency with the rest of the platform, using the same color schemes and font styles, ensuring that users immediately feel familiar with the interface. This consistency across pages helps enhance the overall user experience and navigational ease.

Form Inputs

The core functionality of the page revolves around the **login form**, which requires users to enter their credentials to access their accounts.

- **Username Input:** The form begins with an input field for the **Username**. This field is critical for user identification, allowing the platform to locate the correct account. The label for the input field is clearly marked, and the field is large enough for easy typing, which is particularly important for mobile users.
- **Password Input:** Below the username, users are required to enter their **Password**. This ensures secure access to the platform and protects user data. The password input field features a 'Show' button, allowing users to toggle between obscured and visible text, which reduces errors during login, especially when using complex passwords. This feature enhances user convenience, making it easier for them to confirm they are entering the correct password.

Homepage

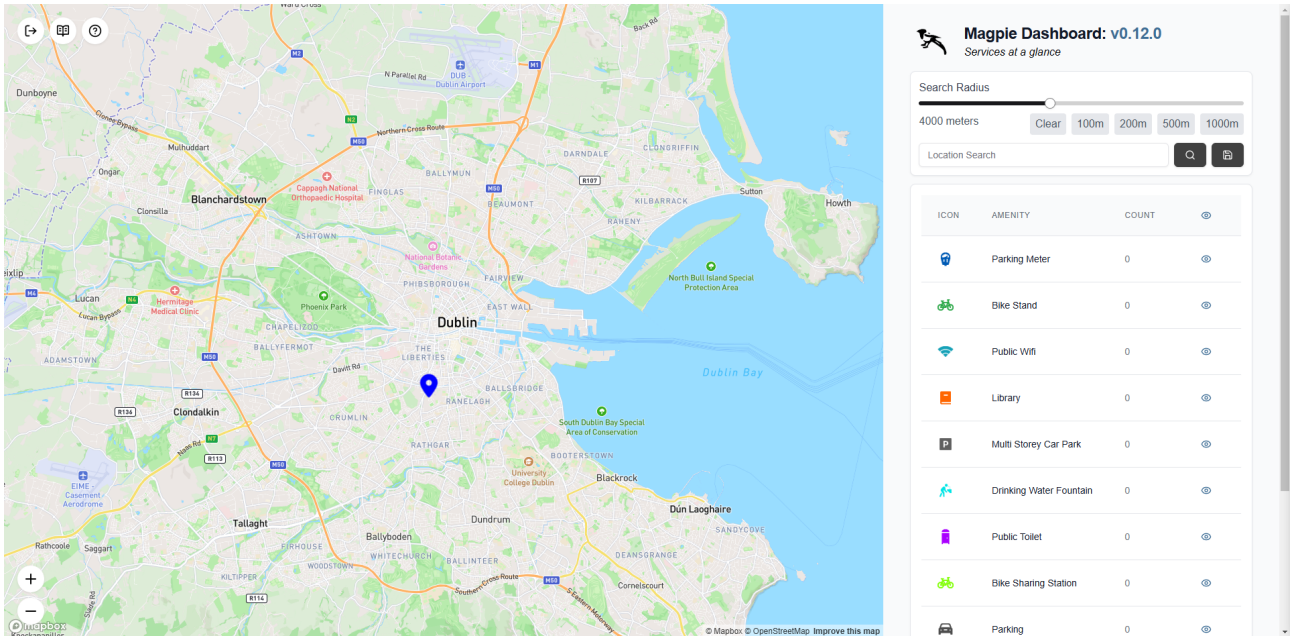

Figure 10: *Homepage*

Introduction

Upon landing on the page, users are greeted with a map interface centred on Dublin and its surroundings, displaying various public services available in the area. The page provides an interactive map to explore amenities and services in the region, such as parking meters, bike stands, public Wi-Fi, libraries, and more.

Top section

- **Logo and Title:** At the top-right corner, you'll see the Magpie logo and the dashboard title.
- **Search Bar and Radius Control:** Beneath the header, there is a search bar to find specific locations, accompanied by a 'Search Radius' slider. This allows users to set the distance range for viewing available services. The slider can adjust the search radius between 1m and 10km.

Map interface

The main focus of the page is the map, which occupies a large portion of the screen

- **Interactive Map:** The map is interactive, powered by Mapbox, where users can zoom in and out using a mouse or touchpad gestures (pinch to zoom). Location markers represent different public services, such as parking meters or bike stands. These markers provide both a visual and interactive experience for users.
- **Service Markers:** Clicking on any marker will display detailed information about the selected service, such as its name, type, and available count.

Sidebar

- **Interactive Map:** The map is interactive, powered by Mapbox, where users can zoom in and out using a mouse or touchpad gestures (pinch to zoom). Location markers represent different public services, such as parking meters or bike stands. These markers provide both a visual and interactive experience for users.
- **Service Markers:** Clicking on any marker will display detailed information about the selected service, such as its name, type, and available count.

Onboarding Tour:

A guided onboarding process walks the user through the various features of the platform, highlighting key elements such as:

1. **Search Radius:** Adjusting the search range.
2. **Marker Data:** Information displayed upon selecting a service marker.
3. **Selecting Amenities:** The ability to toggle amenity visibility.
4. **Map Interaction:** How to click and zoom on the map to select locations.

Footer and Cookie Consent:

Footer: At the bottom of the page, the user is shown a cookie consent banner, offering the option to accept or decline cookies for the site's functionality.

This page is designed to provide users with an easy-to-use and interactive interface to view and search for public services across Dublin. The combination of a detailed map with customizable filters and an intuitive onboarding process ensures that users can quickly understand and **make** use of the platform's features. The inclusion of interactive elements like the service markers and radius control further enhances the user experience.

Prototyping

A prototype is a useful design tool for testing concepts, clarifying requirements, and starting user interaction and feedback. Prototyping methods can be categorized by fidelity—ranging from low-fidelity sketches to high-fidelity digital mockup.

Prototype methods

We use Evolutionary prototyping to continuously update our prototypes.Part of the prototyping process involves dealing with feedback and subsequent revisions. It helps designers test and retest their ideas over and over again. The faster designers are able to test their design concepts and make improvements, the faster they can get to a satisfactory final version. In addition, our team uses Agile development methodologies in prototyping. Agile increases flexibility, collaboration, and rapid feedback cycles to create product prototypes in rapid iterations with continuous ones after collecting feedback and guided ones.

Under the guidance of agile development methodology and Evolutionary strategy, we established the following prototyping process:

Low-fidelity prototypes

- **Sketches**
  We used FigJam for the sketch concept, which allowed us to do a full online brainstorm. we started out with a card format, where the top right side displays the filter and the bottom side displays the rotation of the three icon styles, and the top left and bottom left side have the branding icon and the cookie component, which made the whole page more cluttered. This made the whole page more complicated, so we updated it so that the filter, radius range are on the right as a whole dashboard, and the branding icon is also moved to the dashboard, so that users can better remember our brand.

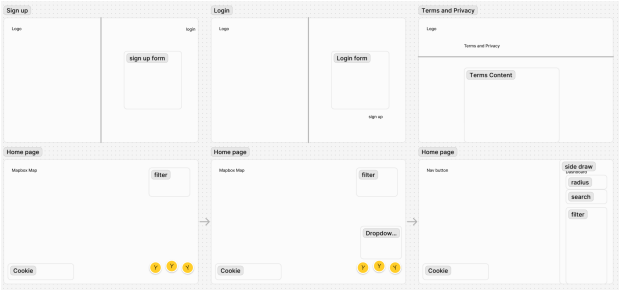

Figure 11: *Evolution of interface design from initial card-based layout to consolidated dashboard approach*

This evolution in our design approach demonstrates the value of iterative prototyping in achieving a balance between functionality and visual clarity. The final layout creates a more focused user experience while maintaining all essential features in an intuitive, accessible format.

- **Wireframe**
  Wireframe as a low-fidelity tool, unlike sketches, wireframes show the structure of an interface design, but often lack detail or colour. We also made wireframes of individual pages to build on, such as the home page and the login/signup page in Figure 12 and Figure 13, which lay out the structure of the prototype.

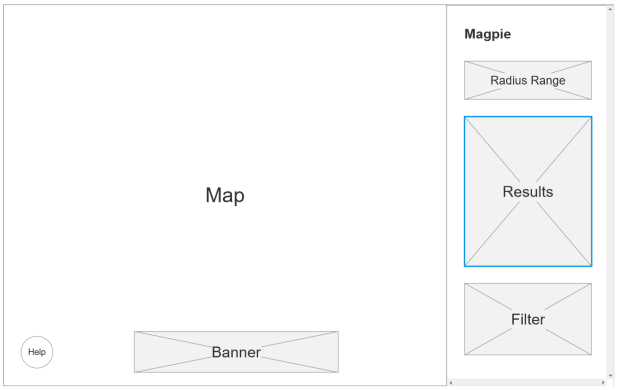

Figure 12: *Wireframe-home*

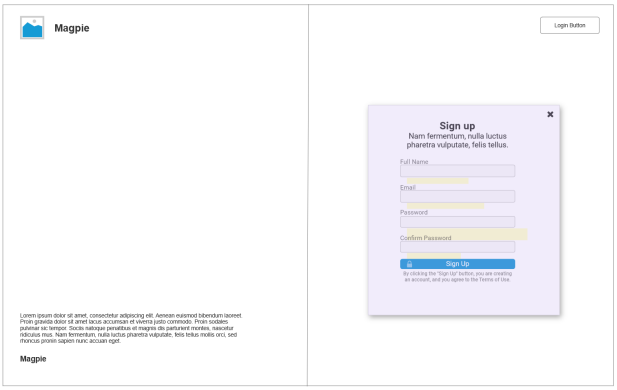

Figure 13: *Wireframe-login/signup*

Medium Fidelity Prototyping

We moved through the prototyping process from low-fidelity sketches and wireframes to medium-fidelity prototypes. This change allowed us to improve the layout, interactions, and structure while keeping the ability to iterate quickly. Medium-fidelity prototypes are often grayscale designs with simple lettering and placeholders, emphasizing functionality and user flow over visual features.

Medium-fidelity prototypes enabled our team to test crucial design decisions such as navigation structures, button placements, and overall usability without being constrained by aesthetic concerns. We created these prototypes in Figma, where we built interactive flows for tasks such as logging in, signing up, and navigating the dashboard. To simulate user interactions, we used more realistic spacing, component alignment, and basic interactive elements like clickable buttons and form fields. The purpose of medium-fidelity prototyping was to collect input on the primary user path, providing a smooth experience from task start to finish. We saved money on later versions by fixing faults at this stage.

High Fidelity Prototyping

High-fidelity prototypes closely resemble the final product in terms of design and functionality. These prototypes include comprehensive visual aspects including the finalized color scheme, typography, iconography, and branding assets, as well as genuine data and interactive transitions. At this point, our team worked on matching the prototype with user expectations and responding to specific feedback from previous versions.

The high-fidelity prototypes were utilized in advanced user testing to assess visual appeal, responsiveness, and accessibility. We focused on how users interacted with the final design features, such as hover effects, animations, and error states. This level of detail meant that the finished product was intuitive, visually consistent, and satisfied usability criteria.

By the time we moved on to high-fidelity prototyping, the emphasis had switched to improving the user experience and preparing the design for deployment. The feedback gathered during this stage directly influenced the UI iteration process, which is documented in the next section and demonstrates how specific issues were handled and changes were made to obtain the final design.

User Interface Iteration

Our iterative UI process focuses on the overall process involved in the User Interface (UI) and User Experience (UX). For example, the visual and interactive components that users use, including buttons, icons, and layouts, while UX covers the overall journey and emotional response of the user during an interaction. The importance of good UI and UX design lies in the improvement of user satisfaction, which then links to better customer retention and enterprise success. It is believed, through studies, that well-thought-out user experiences can increase retention manyfold.**psycray2023** We did extensive reviews of each and every aspect in our iterations- from color scheme and typography to layout and how the navigation structures work-to ensure usability and interactivity. In terms of visual elements, we used a single colour scheme of predominantly black to create a cohesive and less intrusive design. In addition to cultural considerations, the colour black is also inclusive and has a wide audience. The contrast between the accessible text and the background is also stronger, with enough contrast to improve readability for the user.

Iteration 1: Our first iteration used the same design as sketch, but it was more cluttered aesthetically and consistently. The fragmented card-based design made the ribbon look complicated and overwhelming for users. And the first version didn't have a better dashboard. The login/signup page adopts the original shadcn style, which needs to be customized and improved.

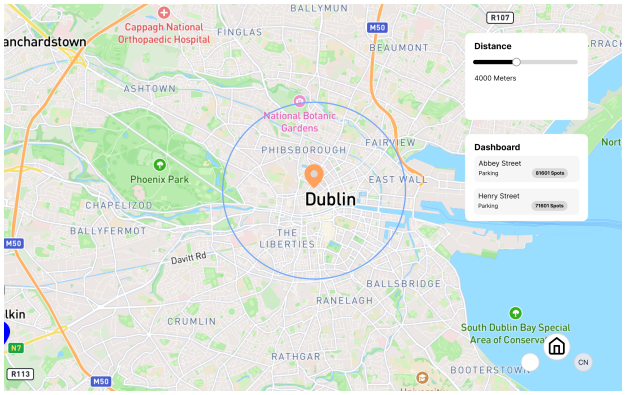

Figure 14: *v1_Home Page*

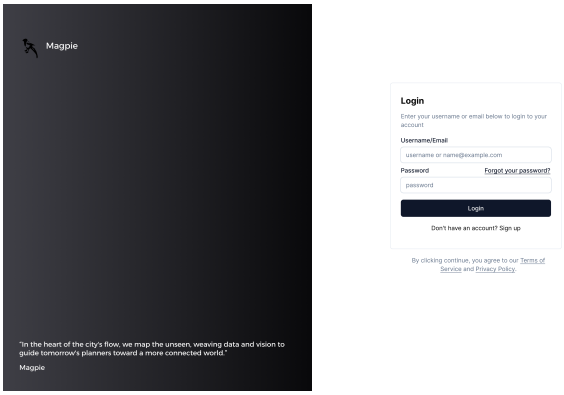

Figure 15: *login/signup*

Iteration 2: In the second version, we introduced the search bar in the initial stage and integrated each card into the dashboard bar on the right. In the later iterations, we also integrated the search bar to the right, making our structure clearer. In terms of interactivity, we ensured that buttons and sliders were easy to use and responded to user operations intuitively. The amenity filter in the second version used checkboxes, but it was not user-friendly.

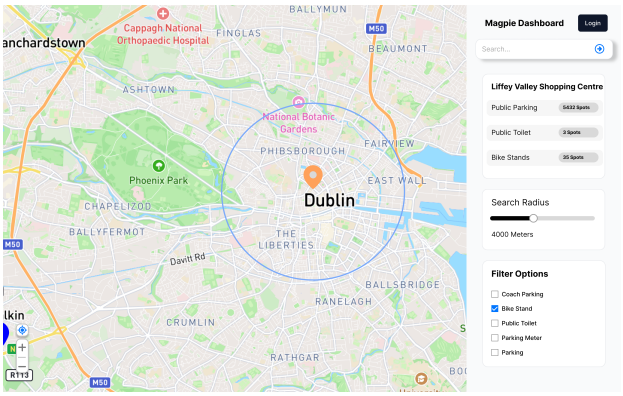

Figure 16: *v2_Home Page*

Iteration 3: For version 3 we have added a onboarding tutorial step to the design, which allows new users to quickly familiarise themselves with the app and creates a clear and intuitive interface that guides the user through

the interactions. In addition we added updates to the history page, term and privacy page.Home pageWe tried to use the drop down menu option to select amenity to display filters, but through user surveys, we found that the interaction between filters and results was not good, and the drop down menu also added complexity. Also some professional users have suggested that the icon is not visible on the map, so the icon will be updated in the fourth iteration.

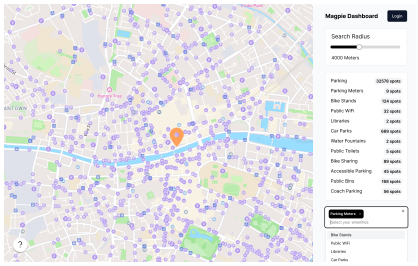

Figure 17: *v3_Home Page*

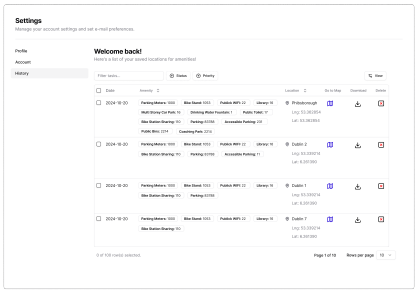

Figure 18: *v3_History*

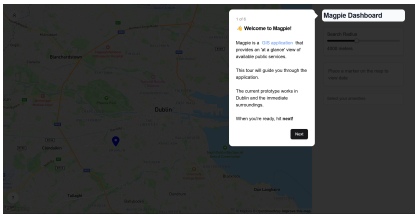

Figure 19: *v3_onboarding*

Iteration 4: For version 4 we have made some major changes to the front end. First of all, the icon has become more understandable and the colours are more differentiated, each amenity is in a different spectrum of colours and can be seen more clearly. The dropdown menu of the third version was cancelled in the fourth version, the burger menu would make the user can not find the corresponding function buttons, so we expanded them so that you can directly click the history entrance and onboarding button. In addition, according to the feedback from professionals, we also added zoom function and designed scale ruler and export image function. Due to time constraints, the latter two were not fully implemented, and the Search function was gathered together on the right side above the dashboard, with more clickable range options such as '100m', '200m', and so on. Filter is a very important component, now you can click on the small eyes to enable the corresponding amenity, which is more intuitive and interactive. In addition, we have created a landing page to showcase our project.

User Feedback and Prototype Evolution

Throughout the prototype design process, user feedback served as a fundamental guiding principle. We implemented a comprehensive framework for collecting and analyzing feedback to ensure continuous optimization based on real user needs and experiences.

Our feedback collection methodology encompassed three main approaches. First, we conducted in-depth, one-on-one user interviews that allowed us to gather detailed qualitative feedback about users' intuitive responses to the interface and identify specific challenges they encountered during their interactions with the platform. These interviews provided valuable insights into the user experience from a personal perspective.

The second approach involved structured usability testing sessions. During these sessions, we created specific task scenarios for users to complete while we carefully observed their behavioral patterns and navigation paths through the interface. This method proved particularly effective in identifying points of confusion and areas where users experienced difficulties or hesitation in their interactions with the platform.

To complement our qualitative research, we implemented a quantitative feedback system through carefully designed questionnaires. These surveys utilized the Likert scale to measure user satisfaction across various aspects of the platform and included open-ended questions to capture additional suggestions and feedback. The quantitative data collected through these questionnaires enabled us to perform statistical analysis of user satisfaction levels and identify trends in user preferences and pain points.

User Feedback and Prototype Evolution

The evolution of our prototype was driven by comprehensive user feedback collected through multiple iterations. Each iteration phase brought valuable insights that shaped the development of our platform, leading to significant improvements in both functionality and user experience.

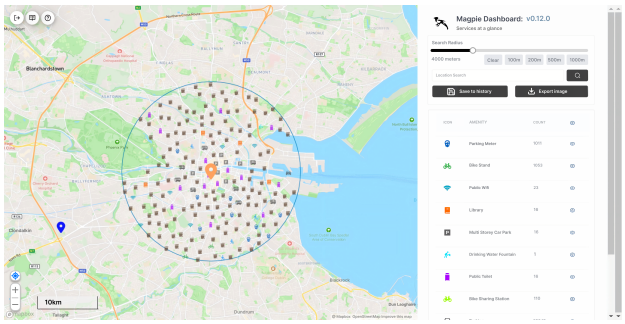

Figure 20: *v4_Home Page*

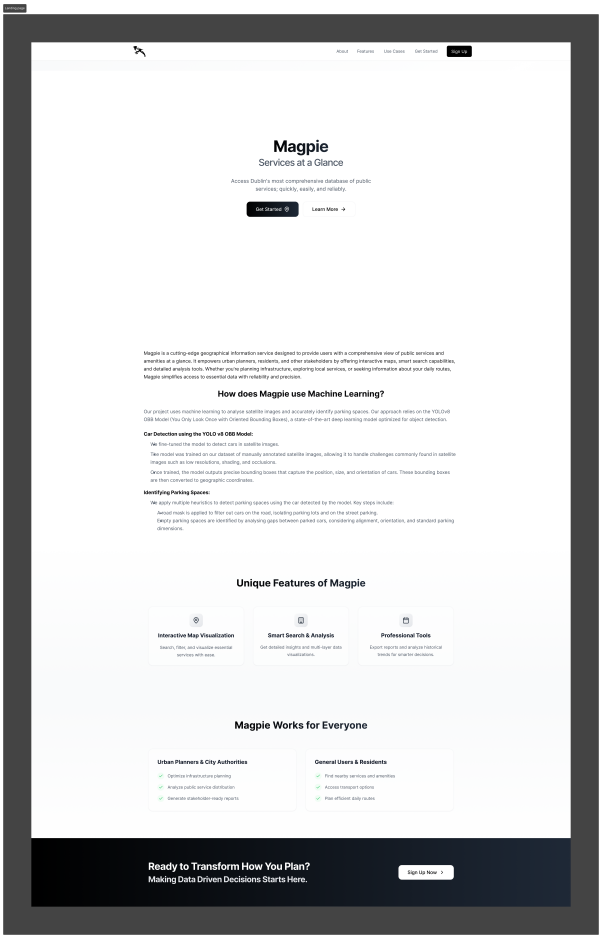

Figure 21: *v4_Landing Page*

First Iteration Our initial prototype revealed several fundamental challenges through user testing. The interface layout, while functional, presented users with a fragmented experience due to its card-based design. Users reported difficulty in managing multiple interface elements scattered across the screen, particularly noting the disconnected placement of filters and radius controls. The login and signup pages, which utilized default shadcn styling, lacked customization and brand consistency. Additionally, the map interface, while central to the application's functionality, needed refinement in terms of icon visibility and interaction feedback.

In response to these early findings, we consolidated the interface elements into a unified dashboard on the right side of the screen. This significant architectural change improved the user experience by creating a more cohesive and intuitive layout. The authentication flow was also carefully considered, with the team ultimately deciding to maintain separate login and signup steps to ensure security while improving the visual design and branding consistency of these crucial entry points.

Second Iteration The second iteration focused on enhancing the core functionality and user interaction patterns. A major addition was the introduction of the search bar, initially placed at the top of the screen. However, user testing revealed that this placement, while conventional, didn't align optimally with the workflow of our specific application. The filter system, implemented as checkboxes, proved to be less intuitive than anticipated, with users expressing a desire for more immediate visual feedback during their interactions.

We responded by integrating the search functionality into the right-side dashboard, creating a more consolidated control center for all user interactions. The amenity filters were redesigned to provide clearer visual feedback, though this iteration still highlighted the need for further refinement in the filter interface design. These changes marked significant progress in creating a more streamlined user experience, but also illuminated areas requiring additional attention in subsequent iterations.

Third Iteration The third iteration brought substantial improvements to user onboarding and interface clarity. We introduced a comprehensive tutorial system to guide new users through the platform's features. This addition was complemented by updates to the history page and the implementation of clear terms and privacy policies. However, user testing revealed that our attempt to streamline the amenity selection through a dropdown menu actually increased interaction complexity. Professional users particularly noted difficulties with map icon visibility, highlighting the need for better visual distinction between different amenity types.

The feedback from this phase proved especially valuable in identifying the delicate balance between functionality consolidation and accessibility. While the dropdown menu was intended to simplify the interface, it actually created an additional layer of complexity that hindered user efficiency. This insight directly influenced our approach to the fourth iteration's design decisions.

Fourth Iteration Our fourth and most refined iteration introduced comprehensive changes based on accumulated user feedback. The icon system was completely revamped, implementing a differentiated color spectrum for various amenities to enhance visibility and quick recognition. We abandoned the dropdown menu approach in favor of directly accessible function buttons, significantly improving the discoverability of key features such as history access and the onboarding tutorial.

Professional user feedback led to the addition of advanced features such as zoom controls and scale rulers, though some of these features remained in development due to time constraints. The search functionality was enhanced with predefined range options (100m, 200m, etc.) and relocated to the dashboard's upper section for better accessibility. Perhaps most significantly, the filter system was redesigned with intuitive toggle controls, allowing users to enable or disable amenities through simple, visible interactions.

Ongoing Development Through these iterations, we identified several areas for future enhancement. These include improving cross-browser compatibility, particularly addressing layout issues in Edge browsers where dashboard elements overlap with the search interface. The accuracy of amenity locations requires verification through systematic testing, and the search functionality needs expansion to handle various address format conventions. Additionally, professional users have requested advanced features such as map extraction capabilities with scale indicators and amenity density visualization tools.

This iterative development process has demonstrated the value of consistent user feedback in creating a more robust and user-friendly platform. Each iteration brought us closer to our goal of providing an intuitive, efficient tool for both casual users and professional urban planners, while also highlighting the importance of continuous refinement and adaptation to user needs.

0.4.6 Backend

Backend Technologies

Go At the start of the project, a portion of the team deliberated about which technologies to choose for the backend. Options using JavaScript such as NodeJS or Deno with accompanying frameworks like ExpressJS or NestJS were considered, as was Rust and its Rocket framework. They would have been excellent choices as they are well established in the industry and tried-and-tested.

The majority of the team has development experience with JavaScript, so going with that would have made a lot of sense. However, it was planned from the very beginning to deploy this application on a server hosted by one of the team members. Therefore, the usage of computing and memory resources was very important to them, as they did not want to strain their Kubernetes cluster more than necessary. Since NodeJS runs on the JavaScript runtime V8, which also powers Google Chrome, our experience indicated that it would be quite resource intensive to run.

Due to this, the team shifted towards using compiled rather than interpreted languages, as these are generally more resource efficient. Since small executables and low memory usage was desired, languages and frameworks that run on virtual machines such as Java with Spring or C# with .Net were not deemed to be viable options. As a result, only Rust and Go were seriously considered at this point.

Rust offers a small package size, strong memory safety, an excellent ecosystem and build-tools. The team member

that would focus on backend development had recent experience in writing Rust code. However, Rust development can be very tricky and time consuming. Additionally, in our experience Rust has above average compilation times. In a project with a fixed deadline and the expectation of rapid development, choosing Rust would have been detrimental. The team recognised that choosing Rust would come with many drawbacks while offering only few benefits.

This left choosing Go as the logical conclusion. A large drawback that the team identified with Go was the missing experience in the team. Two team members had used Go before, but they last used it a few years back. However, since Go syntax and the languages concepts hadn't changed much since then, it was deemed possible to quickly get up to speed, much more quickly than Rust would have allowed. The final decision fell on using Go, as it offered small binaries, great memory efficiency, a solid ecosystem of libraries and build-tools and a feature-rich, built-in library for creating REST-APIs. Go also removes a lot of the pitfalls that Rust suffers from: it has a simple and approachable syntax and a very low-friction, high-speed development experience. This was also the deciding factor, since it was made clear to the team that getting an MVP up and running as quickly as possible was imperative for the project.

PostgreSQL The plan for the MVP of Magpie set the simple goal of delivering useful data to users. To achieve this, the data needs to be stored in a way in which it can be efficiently accessed during operation. Since the goal of the project was to provide a tool to professionals, data integrity at every level was an important consideration. Additional requirements for the data storage solution were good support for geospatial data, quick retrieval of a large number of points and ease-of-use.

There is a plethora of database solutions available today. The team considered the most common types: document databases like MongoDB and multi-model databases like PostgreSQL (sometimes called RDBMS).

While document databases like MongoDB have their place in the current development landscape, it soon became clear that SQL-based, multi-model RDBMSs would be best suited for the job. They offer tried-and-tested performance and reliability and they are well equipped to guarantee data integrity. But the deciding reason was the pre-existing experience the lead developer for the backend had with these types of databases.

After some deliberation, PostgreSQL was chosen as the database solution for this project. It was combined with the PostGIS extension to add support for geospatial data and queries. The decision was not cut and dried as most established multi-model databases (like Oracle DB or MySQL) offer the functionality that the requirements were asking for. However, the Kubernetes cluster this project was going to be deployed on already had a fully configured PostgreSQL server running on it. Making use of pre-existing infrastructure is the obvious choice in an agile project that had the prime goal of hitting the ground running.

sqlc In modern software development, there are two common ways for an application to interact with a database via SQL. There is the old school way of writing raw SQL queries, put them into prepared statements and execute them against the database. This gives the developer very granular control over the way their queries are structured and how they run them. But this method requires a significant amount of work in designing and implementing a translation layer between the database and the application. Data that the application wants to send to the database needs to be prepared into a format the database queries expect. Data that the application wants to retrieve from the database needs to be parsed back into the data model that's used by the application.

To make interfacing with databases easier, so called ORMs (Object Relational Models) were developed. These are libraries that the application developer can include. Database queries are not done in SQL, but in the same programming language the application is written in. The ORM provides database interface functions that take in the data in the format the application uses. To actually perform any queries or make any changes to the database, the ORM runs its own SQL queries against the database via a compatible driver internally. ORMs abstract away the direct communication with the database and provide a simpler wrapper in the programming language that is used anyway. This can make it easier and quicker to develop an application that makes use of a database, but it takes away some of the agency that the developer has.

The team wasn't really happy with either of these solutions, so an alternative called sqlc was chosen. This tool combines the freedom that using SQL gives developers with the productivity increase that ORMs offer. sqlc flips the typical ORM workflow on its head. This means, the developer writes standard SQL queries and schemas which are then passed to the CLI of sqlc. In accordance with a configuration file provided by the developer, sqlc then automatically generates type-safe bindings for the queries that were defined in the SQL files (**sqlc_introduction**).

This approach gives the developer more control about the queries that are executed. At the same time, it eliminates the need to write boilerplate wrapper code for accessing the database, just like an ORM would eliminate the need to write SQL queries. The tool treats SQL, which is a structured and typed language, as a source of truth. It also enables developers to reuse their queries across systems as sqlc offers code generation for multiple programming languages like Go, Python or Typescript and databases like PostgreSQL, MySQL or SQLite (**sqlc_documentation_language_support**).

The tool respects best practices to prevent SQL injection attacks. It generates code that only utilises constant strings and parametrised queries (**sqlc_injection**). Using a tool like this, the team was quickly able to connect the backend server to the database. Eliminating the need to update the Go code manually each time the database schema or queries changed was vital for the backend keeping pace with the other developers.

```yaml
version: "2"
sql:
- schema: "sql/migrations"
  queries: "sql/queries_private.sql"
  engine: "postgresql"
  gen:
    go:
      package: "db"
      sql_package: "pgx/v5"
      out: "internal/db/private"
      build_tags: "private"
      emit_json_tags: true
      overrides:
        - db_type: "geometry"
          go_type:
            import: "github.com/twpayne/go-geom"
            pointer: true
            type: "Point"
- schema: "sql/migrations"
  queries: "sql/queries_public.sql"
  engine: "postgresql"
  gen:
    go:
      package: "db"
      sql_package: "pgx/v5"
      out: "internal/db/public"
      build_tags: "public"
      emit_json_tags: true
      overrides:
        - db_type: "geometry"
          go_type:
            import: "github.com/twpayne/go-geom"
            pointer: true
            type: "Point"
```

Listing 1: *An example of a sqlc configuration file with two targets with separate query inputs*

golang-migrate The complete database development workflow for the backend server is visualised in Figure 22.

bcrypt

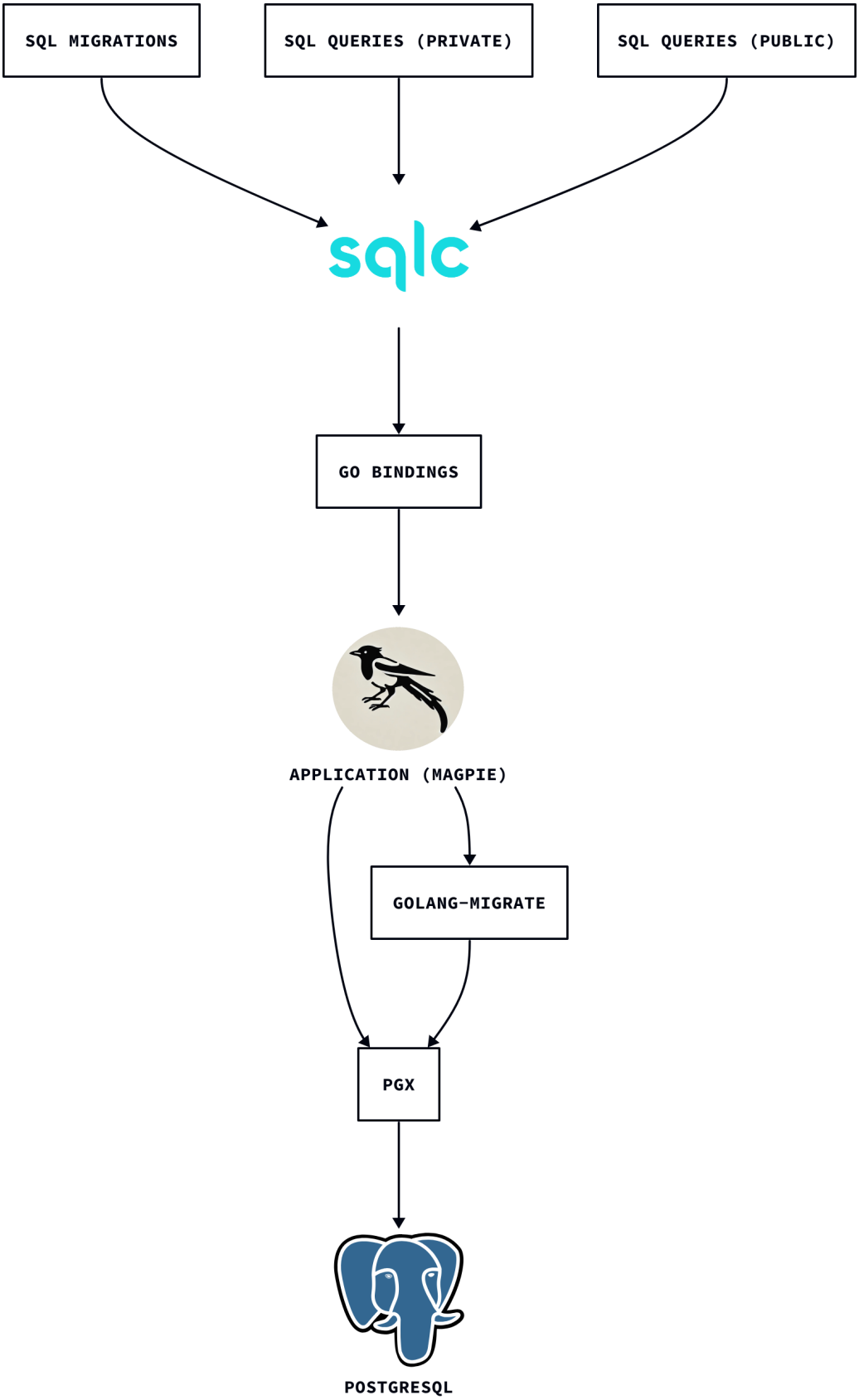

Figure 22: *Database Workflow Diagram*

```
-- name: GetPointsInRadius :many
SELECT Id, LongLat::geometry, Type from points
WHERE ST_DWithin(
  LongLat::geography,
  ST_SetSRID(ST_MakePoint(@longitude::float, @latitude::float), 4326)::geography,
  @radius::float
) AND (
  @types::point_type[] IS NULL OR Type = ANY(@types::point_type[])
);
```

Listing 2: *An example of a SQL query with annotations used by sqlc*

Backend Development

As discussed in 0.4.2 System Architecture, the backend is comprised of two standalone REST-API servers. Right after the decision to split the backend was taken, a first implementation using two separate codebases was created. This approach was functional, meaning it was able to produce two distinct backends with different feature sets.

But it soon became obvious that this approach was not sustainable. While simply splitting the backend into two codebases was a quick and easy solution, it would almost certainly lead to significantly increased development times in the future. The two backends have a fair amount of identical functionality and differ just in the route handlers and the database queries. Leaving the codebases separate would result in many changes being made twice – once in the private backend and once in the public backend.

After careful consideration and discussions with the team, the decision was made to revert the change that split the backend. The desired result of two distinct backends would have to be achieved in a different way.

To accomplish this, a feature of the Go programming language called *build-tags* was utilised. Usage examples of this feature showcase it by creating multiple binaries that have different feature sets, for instance multiple different payment tiers for a single software. This was a great solution for this problem. It allowed for certain files to be excluded during compilation based on if they were needed for private or public functionality. While the development of such a program split by build-tags is more challenging than developing a single binary, it is much more streamlined than keeping functionality identical between two separate projects.

Middlewares

Authentication

Authorisation

Data Validation

Error Handling

Testing

0.4.7 Deployment

```
const getPointsInRadius = `-- name: GetPointsInRadius :many
SELECT Id, LongLat::geometry, Type from points
WHERE ST_DWithin(
  LongLat::geography,
  ST_SetSRID(ST_MakePoint($1::float, $2::float), 4326)::geography,
  $3::float
) AND (
  $4::point_type[] IS NULL OR Type = ANY($4::point_type[])
)
`

type GetPointsInRadiusParams struct {
  Longitude float64     `json:"longitude"`
  Latitude  float64     `json:"latitude"`
  Radius    float64     `json:"radius"`
  Types     []PointType `json:"types"`
}

type GetPointsInRadiusRow struct {
  ID      int64          `json:"id"`
  Longlat *go_geom.Point `json:"longlat"`
  Type    PointType      `json:"type"`
}

func (q *Queries) GetPointsInRadius(
    ctx context.Context, arg GetPointsInRadiusParams
  ) ([]GetPointsInRadiusRow, error) {
  rows, err := q.db.Query(ctx, getPointsInRadius,
    arg.Longitude,
    arg.Latitude,
    arg.Radius,
    arg.Types,
  )
  if err != nil {
    return nil, err
  }
  defer rows.Close()
  var items []GetPointsInRadiusRow
  for rows.Next() {
    var i GetPointsInRadiusRow
    if err := rows.Scan(&i.ID, &i.Longlat, &i.Type); err != nil {
      return nil, err
    }
    items = append(items, i)
  }
  if err := rows.Err(); err != nil {
    return nil, err
  }
  return items, nil
}
```

Listing 3: *An example of a Go binding generated by sqlc from the SQL query in Listing 2*

0.5 Software Management

0.5.1 Why is Software Management important?

In Software Development, it is said that the speed of work is often dictated by the quality of the tools used to perform that work. Knowing this, *Magpie* makes use of a number of tools to ensure that the development process is as efficient as possible.

This chapter will discuss the tools we used to create *Magpie* and how they were used to manage the project. We will also discuss the importance of the tools used and how they helped aid development and maintain a solid pace.

0.5.2 What is DevOps?

> "DevOps is about fast, flexible development and provisioning business processes. It efficiently integrates development, delivery, and operations, thus facilitating a lean, fluid connection of these traditionally separated silos" **DevOps**

DevOps is a set of principles and practices that combines software development **(Dev)** and IT operations **(Ops)**. The primary means of reducing friction between development and operations is through automation, and the use of tools to multiply progress, and reduce effort. This can be broadly sorted into two categories:

- **Continuous Integration (CI)**: This is the practice of automatically building and testing code every time a developer commits changes to version control. This ensures that the code is always in a working state. **DevOps**.
- **Continuous Deployment (CD)**: This is the practice of automatically deploying code to production servers every time a change is made. This ensures that the application is always up–to–date and that new features can be released quickly. **DevOps**.

The next sections will detail the tools used to implement **Continuous Integration** and **Continuous Deployment.**

0.5.3 Continuous Integration

In *Magpie*, the CI workflow is focused around **GitHub** and it's suite of tools. We decided on using this approach, because it enables a powerful flow:

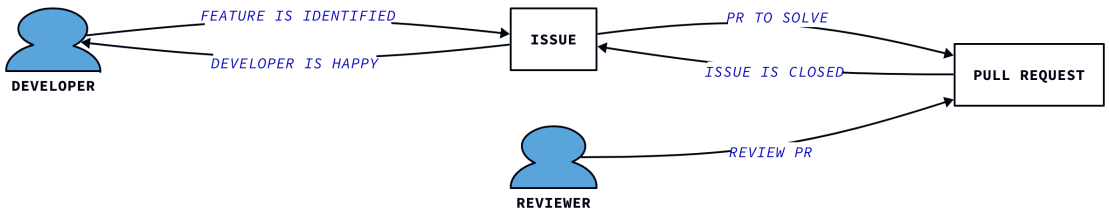

Figure 23: *The flow of work in Magpie*

Issues

GitHub Issues are used to track tasks, enhancements, and bugs to be worked on. In our experience, they are a effective way to keep track of what needs to be done, and what has been done. **GitHub Issues** can be assigned to team members, and can be linked to **Pull Requests**.

Issues represent the primary unit of work in projects using **GitHub**'s flow. In the below example, a checklist is presented- with each item indicating a task that needs to be completed. An issue's checkboxes, can also be assigned to **Pull Requests**.

On the right, the issue displays the **Assignees**, **Labels**, and **Projects** associated with the issue. This allows for easy filtering and sorting of issues, and helps to keep the project organized.

Under the description, the issue keeps track of edits, comments, and other interactions. This allows for communication to stay in context with the issue being discussed, and requirements- or additional information- to be added as needed.

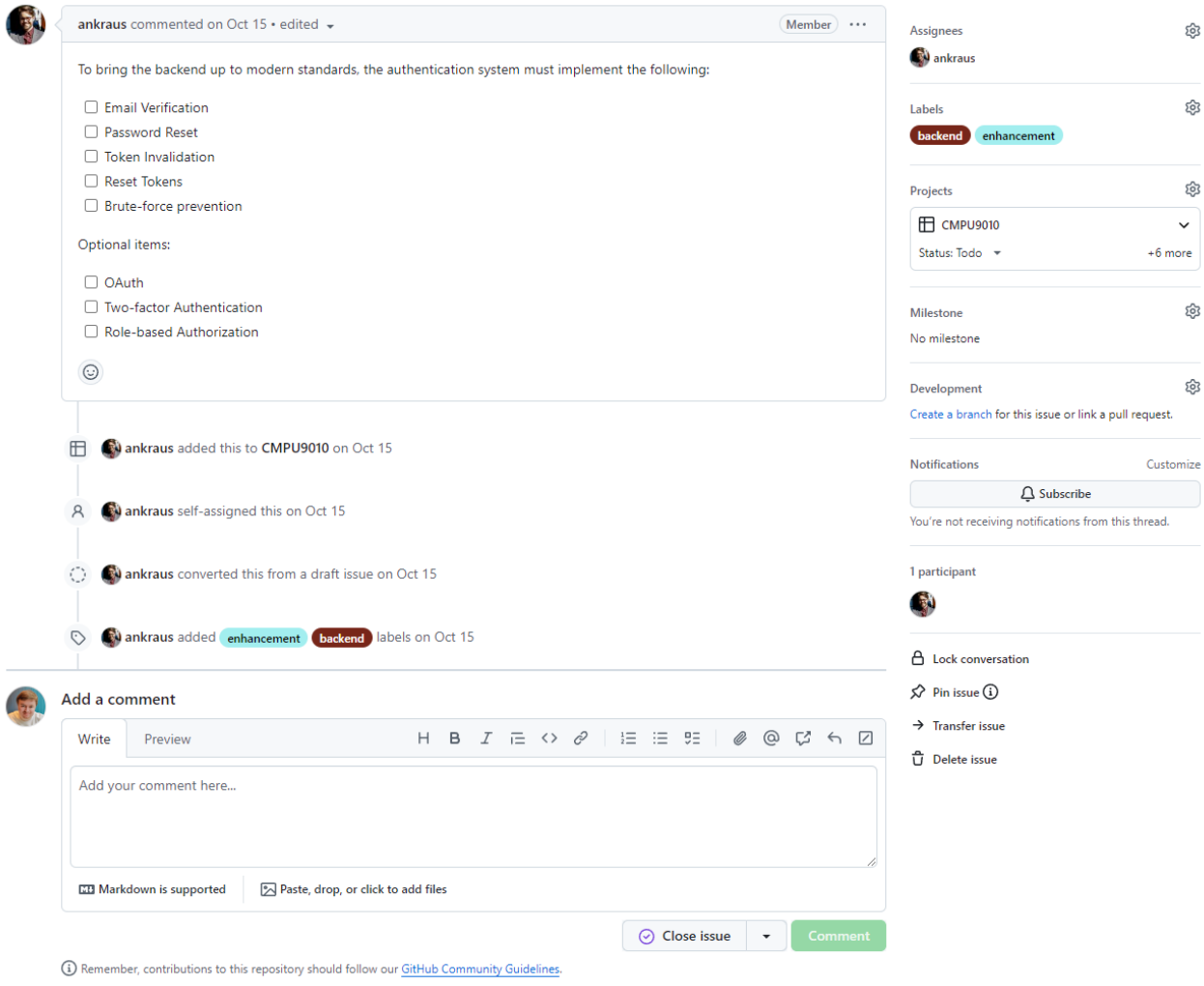

Figure 24: *An example of a GitHub Issue*

In general, issues are a great way to keep track of what needs to be done, however, in our experience, they can be tedious to write manually. To help with this, templates can be used.

To use a template, create a directory called **.github** in the root of the project, then create another directory called **ISSUE_TEMPLATE**, note that the capitalization is important. Inside this directory, any markdown file created in the prescribed format can be used as a template for issues, you can have as many templates as you like.

In *Magpie*, we used a template for **Bug Reports**, **Documentation Requests**, and **Feature Requests**. This reduced the friction of creating issues by minimising the amount of effort related to drafting an issue (using the template) and creating clear delineations between different categories of issues.

```
---
name: Bug Report
about: Create a report to help us improve
title: ''
labels: bug
assignees: ''
---

**Describe the bug**
A clear and concise description of what the bug is.

**To Reproduce**
Steps to reproduce the behavior:

1. Go to '...'
2. Click on '....'
3. Scroll down to '....'
4. See error

**Expected behaviour**
A clear and concise description of what you expected to happen.

**Screenshots**
If applicable, add screenshots to help explain your problem.

**Additional context**
Add any other context about the problem here.
```

Listing 4: *An example of an issue template used in Magpie*

Pull Requests

Pull Requests are used to merge code from one branch to another. In our experience, they are a great way to review code, and ensure that it is up to standard. They can be linked to **Issues**, and like issues, can be assigned to team members.

In the below example, the interface is largely similar to that of an issue, however there are a few key differences. Note that under projects, you can set a task that will be present in the Kanban board that will be discussed in the next section.

If this PR solves a milestone, or contributes to an issue containing a milestone, it will be displayed here. *Magpie* did not use milestones with the exception of the interim report, however we feel it can be valuable in projects with more clearly defined goals, such as major releases.

Note, that the pull requests we used also included automated CI jobs. These will be discussed in a later section.

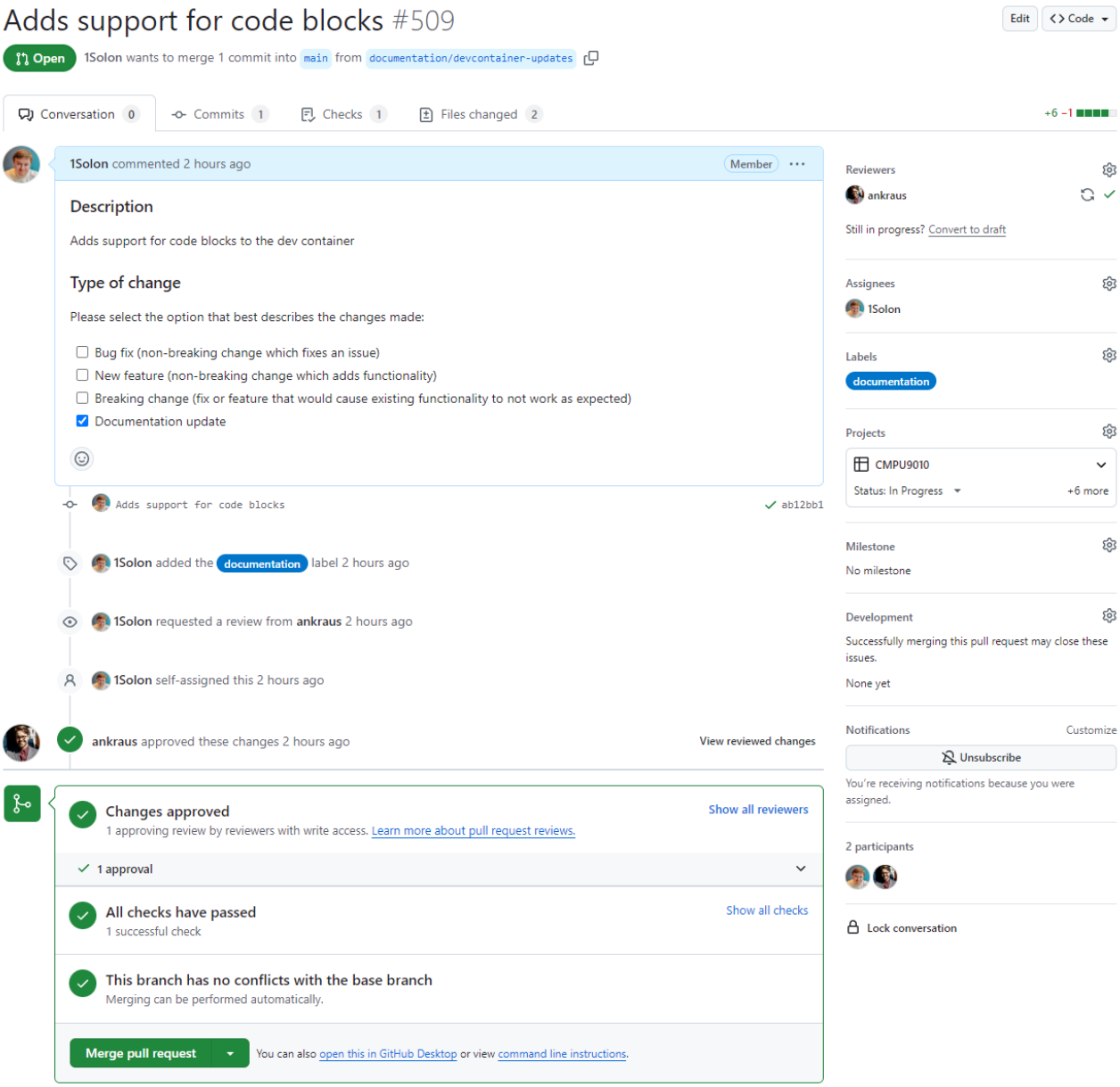

Figure 25: *An example of a GitHub Pull Request*

Like **Issues**, *Magpie* leverages **Pull Request Templates** to reduce friction. Templates reduce friction by minimising the amount of writing necessary when drafting a PR.

During the project, we at times ignored issue templates because because the benefits of writing fully-detailed issues, did not always return greater utility. However, we never ignored pull request templates.

PRs need a high level description of the changes made, and having a consistent format for this description was important to ease the burden on code reviewers, particularly working in a team environment.

```
### Description

Replace this with a summary of the change. Please also include relevant
motivation and context.

Fixes # (issue)

### Type of change

Please select the option that best describes the changes made:

- [ ] Bug fix (non-breaking change which fixes an issue)
- [ ] New feature (non-breaking change which adds functionality)
- [ ] Breaking change (fix or feature that would break existing functionality)
- [ ] Documentation update

### Changes

Replace this with a list of changes made in the pull request.
```

Listing 5: *An example of a pull request template used in Magpie*

Projects

GitHub Projects are used to track the progress of work in a project. In our experience, they are a great way to keep track of what needs to be done, and what has been done.

GitHub Projects can be used to create a **Kanban Board**. In our experience, this is a great way to visualise the progress of work in a project. In the below example, the board is divided into columns, each representing a different stage of work.

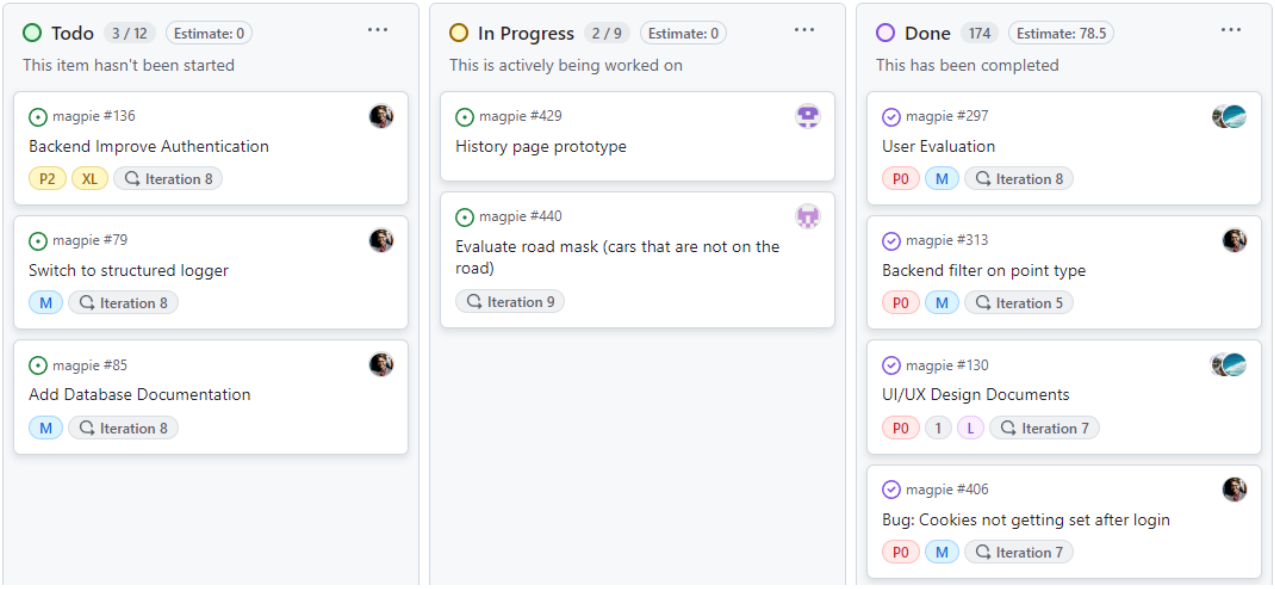

Figure 26: *An example of a GitHub Kanban Board*

In *Magpie*, we predominantly used the **Kanban Board** to provide a high-level overview of tasks within the project. In general, this allowed all team members to see what was being worked on, and what needed to be done.

As a card is just a representation of an issue, clicking on it will take you to the issue page. This closes the loop on the flow identified in the beginning of this section.

GitHub Projects includes a **Roadmap** tab. Unlike the **Kanban Board** tab, which gives an overview of task status, the **Roadmap** gives a temporal view of the timing associated with those tasks.

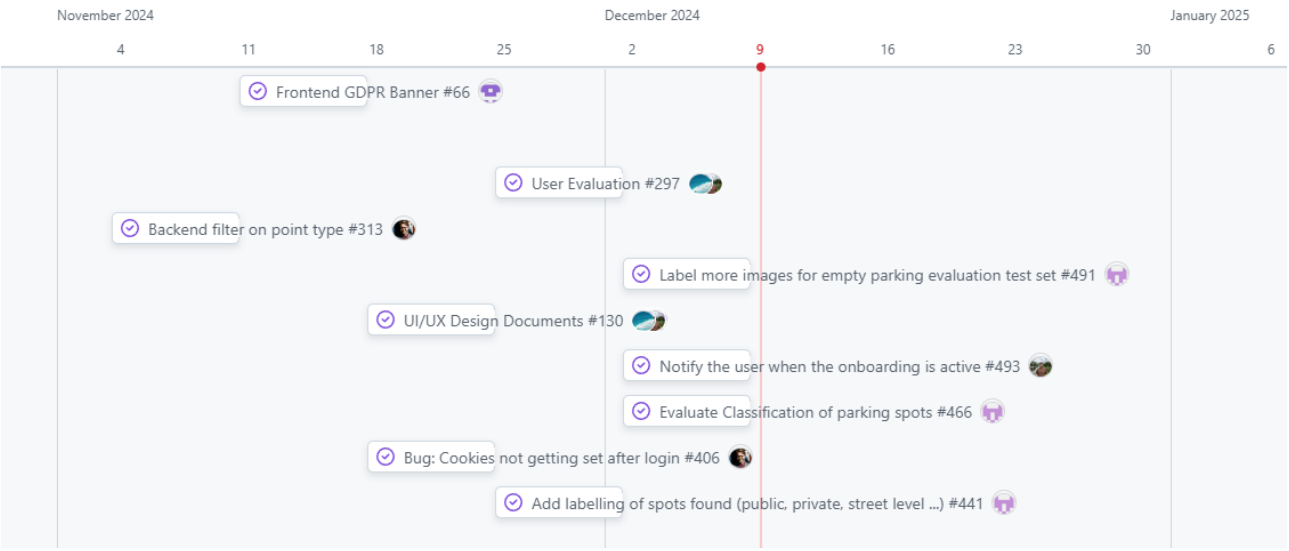

Figure 27: *An example of a GitHub Roadmap*

0.5.4 Actions

GitHub Actions are used to automate tasks in a project. In **Magpie**, we used **GitHub Actions** to automate tests, housekeeping and building tasks. Housekeeping and tests fall comfortably within CI, however building is more closely aligned to CD- as such, it will be discussed in a later section.

Houskeeping

Magpie uses a branch naming scheme:

- **main**: The main branch of the project. This is the branch that is deployed to production.
- **backend/**: A branch containing backend features.
- **frontend/**: A branch containing frontend features.
- **documentation/**: A branch containing documentation.
- **python/**: A branch containing Machine Learning features.
- **update/**: A branch containing updates to the project.
- **misc/**: A branch containing anything not in the above categories.

This was initially done to trigger specific actions based on the name of the branch. However, it became a neat means of seeing what branch was dealing with what task, in our experience, in projects like this with upwards of 8+ branches at a time, being able to, at-a-glance, determine what the branch was dealing with was very useful.

We decided against using branch rules to trigger actions because, by default, they do not function as expected. For example:

```yaml
on:
    push:
        branches:
        - 'feature/*'
```

Listing 6: *An example of a GitHub Actions workflow that will not work*

From the above, you may expect that the workflow will trigger on any branches starting with **feature/**. However, this will only trigger on any branches being *merged into* **feature/**. This is not the desired behaviour, and as such, we decided against using this particular method.

To solve this problem, we instead triggered actions based on what *directory* was being changed. This was done by using the **paths** key in the workflow.

```yaml
on:
    push:
        paths:
        - 'Backend/**'
```

Listing 7: *An example of a GitHub Actions workflow that will work*

In *Magpie*, to maintain our branch naming scheme, it was important to have automation in place to ensure that branches were named correctly. This was done using a **GitHub Action** that would check the branch name, and if it did not match the scheme, would fail the action.

Note that, in a **Pull Request**, certain actions will come up as checks. Checks will prevent a PR from being merged if they fail.

```yaml
name: Branch Checks

on:
pull_request:

jobs:
validate-name:
    runs-on: ubuntu-latest

    steps:
    - name: Check out repository
        uses: actions/checkout@v4

    - name: Get branch name
        id: get_branch_name
        run: echo "branch=${GITHUB_HEAD_REF}" >> $GITHUB_OUTPUT

    - name: Validate branch name
        run: |
        BRANCH_NAME="${{ steps.get_branch_name.outputs.branch }}"
        if [[ ! "$BRANCH_NAME" =~ ^(
            backend/ frontend/|
            documentation/ python/|
            distribution/ misc/|
            update/
            ).* ]]; then
            echo "Branch name '$BRANCH_NAME' is not valid."
            echo "Rename the branch to match one of the following patterns:"
            echo "'backend/', 'frontend/',"
            echo "'documentation/', 'distribution/',"
            echo "'update/', 'python/',"
            echo "or 'misc/'."
            exit 1
        fi
    shell: bash
```

Listing 8: *A github action that checks the branch name (this was edited for brevity)*

The above action runs on every pull request, and checks the branch name. If the branch name does not match the pattern, the action will fail, and the PR will not be able to be merged.

Checks

GitHub Actions can be used to run checks on code. In *Magpie*, we used checks to run automated testing in the fronted and backend. The frontend and backend checks are discussed in those sections.

0.6 Evaluation

0.6.1 Components of usability

User experience design and testing

Usability is a *'quality attribute that assesses how easy interfaces are to use'* (**usabilitycomponentsnielsen**). The best way to assess the usability of an interface is to conduct usability testing.
Also referred to as *user testing* is a popular UX research methodology used to uncover problems in the user interface design of an application and learn about the target user's preferences (**usertestingdefinition**).

There are 3 core elements to user testing: **the facilitator**, **the participant** and **the tasks**. The facilitator administers the tasks to the participant, observes their behaviour and listens to their feedback.

Magpie has remedied the first challenge of fragmented information on amenities. We will now address the second challenge: making the access to this information easy, quick & accessible.

Our approach is as follows:

1. Round up users from the market research + seek out others
2. Conduct online usability sessions to discuss Magpie, explore the features, gather feedback,
3. Synthesize notes from sessions and summarize points to improve
4. Iteratively implement/improve features

We interviewed 11 users in total. Some have decided to remain anonymous, therefore have been labelled as *Anonymous n°*, while others will be referred to by their name.
They have been divided into the following categories: 6 general users, 3 targeted users, 1 UI/UX expert and 1 accessibility expert.
General users are defined as those who use Magpie casually for personal interests.
Targeted users are defined as those who use Magpie as a tool for their work.

Both controlled & uncontrolled approaches were used for the sessions.
The controlled sessions were based around a strict list of tasks the user would complete and used metrics such as time taken, difficulty and task success rate.
The uncontrolled sessions let the users freely roam the application while we observed their behaviour interacting with each element and initiate discussion to obtain feedback on features to improve.

A table with a list of general tasks is used to quantitatively evaluate each feature the user interacted with. Metrics measured are task difficulty and task success rate. The list of general tasks increased as the test sessions went on because new features were being added iteratively.
The difficulty of the task is related to its status and how much time a user spent on it. The status of a task can either be 'complete', 'pass', or 'fail' where:

- *complete* is attributed when the user completes the task on their own
- *pass* is attributed when the user was able to complete the task but with our help
- *fail* is attributed when the user was not able to complete the task even with our help

Lastly, a short satisfaction survey is administered at the end of each session quantify user experience and provide a baseline for improvements. User behaviour is also observed during the session to complement these quantitative metrics.

The evaluation of Magpie has been divided into the following sections:

1. General users test sessions
2. Targeted users test sessions

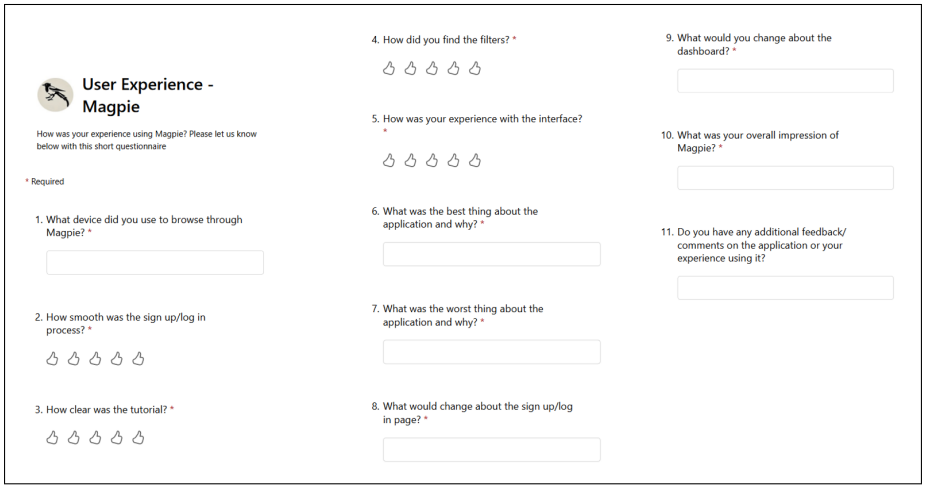

Figure 28: *User Evaluation - Satisfaction survey questions*

3. UX/UI expert review
4. Accessibility expert review

0.6.2 General user evaluation

User 1 - Brendan

Brendan is a professor with technological background. They are considered a casual user.
An uncontrolled test session was conducted where they tested each feature of Magpie to find faults, system failures and bugs.

Main takeaways from Brendan's session: Magpie has potential for use by certain types of users but needs a lot more functionality. Here is a breakdown of the feedback:

- **Log in/Sign up:** email verification is important, especially for a service advertised towards professionals. Also, Brendan did not understand the need for username to be created, the username should automatically be the email address. As a group, we decided to keep the username field as a preference and for future work as the user will be greeted by the username chosen when the log in„ or it will display in the profile menu.

- **Tutorial:** the content of step 2 needs to be reworked and use more descriptive language. Also, the positioning of certain elements is off in Firefox browser. For step 4, the word 'dozen' should be changed to 'several' or similar especially if we are planning on adding more amenity data. Lastly, step 5 should point to the map, so that is a bug we need to investigate bug.

- **Dashboard:** we should implement a button to clear the marker and all the points from the map, right now the user is forced to reload the page to do that. In addition, it could be beneficial to certain users to leave the count of toggled off amenities. Also, compress the list of amenities to avoid scrolling.
  We found Firefox to cause a window size issue that we have been unable to address.

- **Map:** plus and minus buttons should be added to zoom in and out of the map, especially for users who are not familiar with mouse technology. Also, clicking on amenity icons should provide more information. Right now, it feels a bit empty. Lastly, some of the selected icons are not visually striking, perhaps find a way to emphasize them.

- **Technical:** when clicking on map, there is a slight offset between where the marker appears and where the user clicked. There is cause for investigation there. Also, maybe think of implementing something to avoid mis-clicking and loosing the original marker position.

- **Miscellaneous:** Brendan asked why we were requesting location and what we were doing with this data. Accepting Magpie to use the user's location only puts down their own location marker on the map indicating where they are located. The location information is not processed by the Magpie server, it is processed on the user's device.
  Additionally, we need a landing page to present Magpie and put forward the machine learning aspect of the project.

Overall: the interface is nice, but you might want to focus on producing a much more data-driven approach for the interface if you want to attract those kinds of users.

Score from survey:

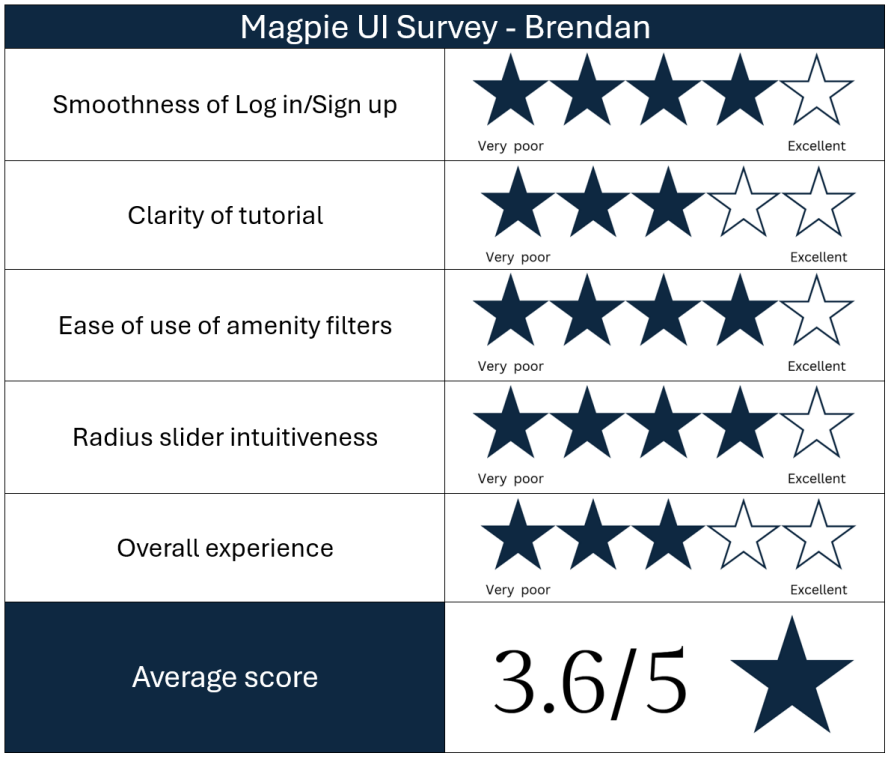

Figure 29: *User Evaluation - UI Score Brendan*

User 2 - Anonymous 1

This user has a background in technology at the doctorate level.
This session was mostly uncontrolled, they browsed the application, tested the features and discussed their thoughts with us.

Main takeaways from the session: this user really enjoyed the presentation of information on the application, they found the amenities easy to understand and recognizable in the radius of the map.
They tried to interact with the locked elements of the tutorial, and really enjoyed the confetti at the end of it. Also, before placing a marker on the map because when they accepted location tracking, their own marker appeared.
To make more improvements, they suggested the following:

- **More features:** They feel like a search bar would help in the quest for information in specific location visually unknown to the user. They also though the points would appear automatically on the map but instead she had to place her own marker. Perhaps there is an issue with onboarding not being retained.
- **Dashboard & Map:** There is an icon, the water fountain one that is neon blue, therefore blends into the white background of the dashboard and in the map; probably needs to be changed. Also, more data for example on transportation would be a big plus.
- **Misc:** If you want to appeal to more general users, adding more features like location sharing, integrating social interactions and make it mobile responsive would be the way to go. Also, a higher level of information.

Overall: it is a very good application, a very interesting idea to gather all this information in one place.

Score from survey:

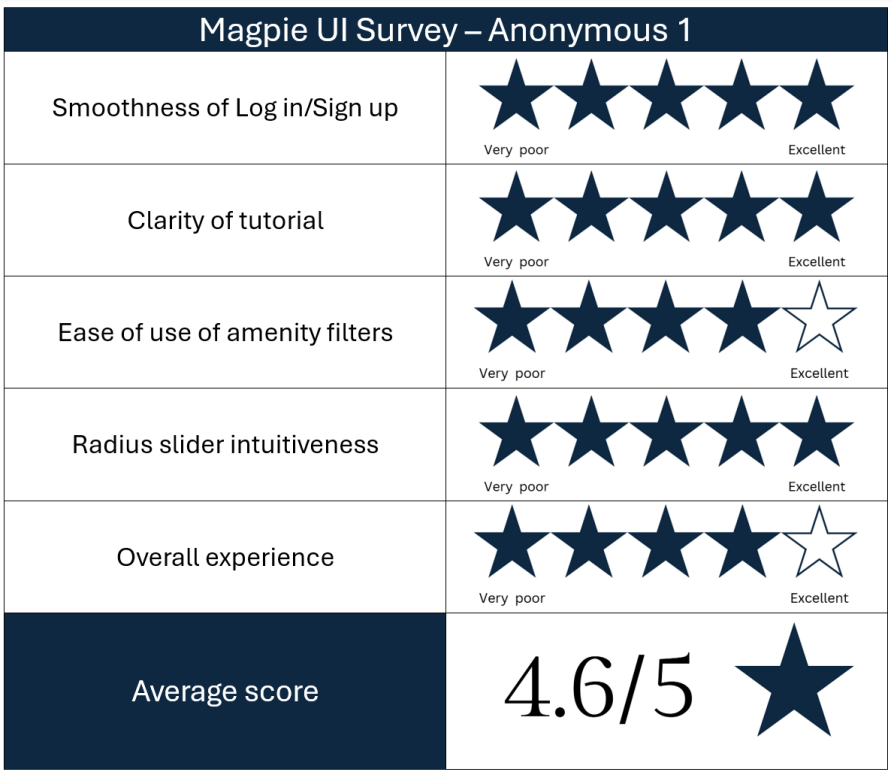

Figure 30: *User Evaluation - UI Score Anonymous 1*

User 3 - Paul

Our next session was with Paul, a student in technological undergraduate degree. They are classified as a general user. They left their contact email in the market research survey.
We initially wanted this to be a controlled test session by giving him specific tasks, but found that challenging as he intuitively went on to explore the application on his own.
Main takeaways from Paul's session: the map and the amenity data displayed is 'excellent', they would find it useful for local areas of the city. One aspect they advise we improve on is to make the choice of amenities more intuitive. This is further supported by they're behaviour trying to click on the icon and amenity title on the dashboard to toggle it on and off, as well as the difficulties they encountered as shown in the general task table. Another point to improve on is to make the profile and tutorial icons more visible, demonstrated by the time it took to find them.

Score from survey:

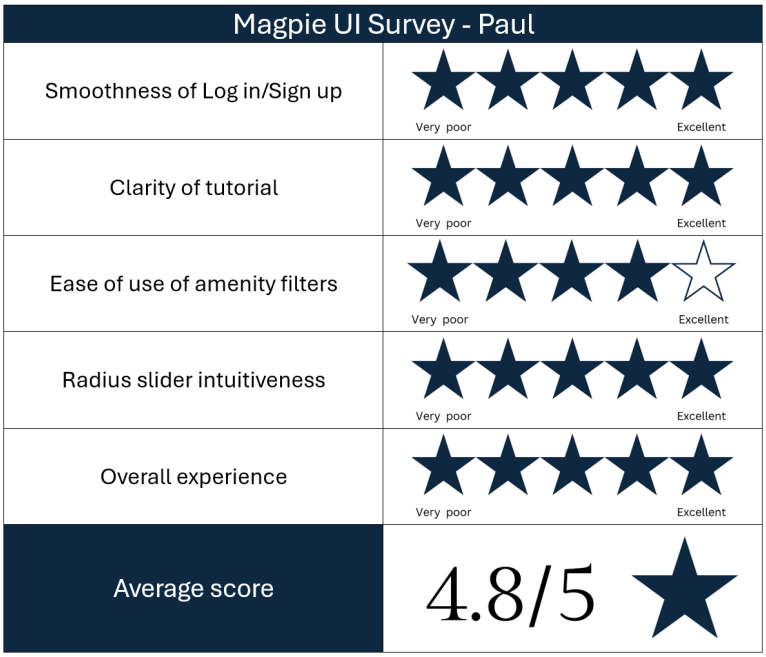

Figure 31: *User Evaluation - UI Score Paul*

User 4 - Livia

Our next session was with Livia, another student in a technological undergraduate degree. They are also identified as a casual user who also left their contact in the market research survey.
This session also started out as a controlled test with a defined set of tasks, but just like Paul, Livia went on to explore the application skipping the tasks.

Main takeaways from Livia's session: very interesting project, useful and great; overall a very clear website. Biggest point of discontent for Livia was the tutorial, they let us know they has dyslexia and the tutorial could've been worded more effectively to cater to them and others with learning/visual impediments. In addition, they tried to interact with the locked elements during the tutorial, suggesting intuition to put in practice what they are reading to validate the information absorbed.
They also suggested adding more amenities such as public transports stops, scooter stands and student hubs. They liked how it was easier to understand the information visually compared to Google maps or Apple maps.

Score from survey:

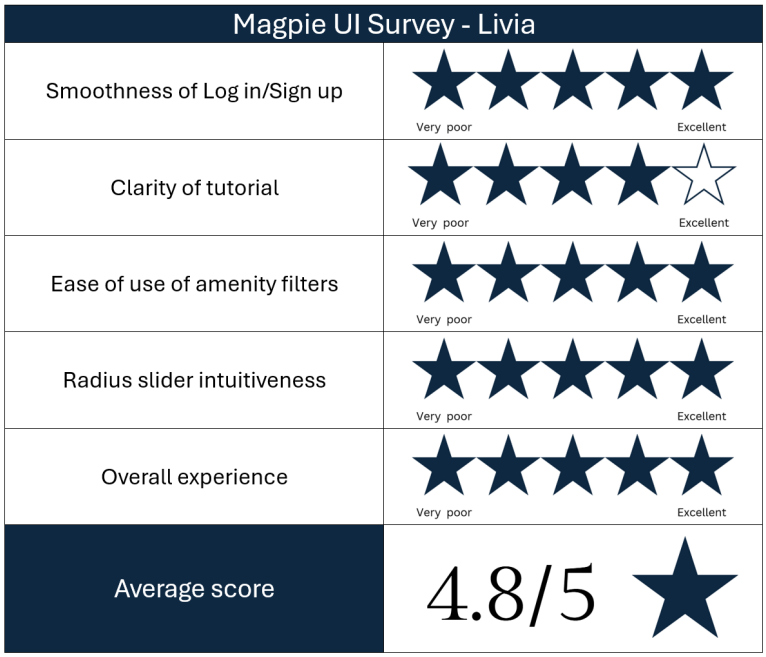

Figure 32: *User Evaluation - UI Score Livia*

User 5 - Ben

Our next test session was with Ben, another student in a technological post-graduate degree. They are also identified as a casual user who also left their contact in the market research survey.

Starting this session, we took a more uncontrolled approach and let the users free roam the application without giving them specific tasks to complete. We guided them in the beginning and initiated certain discussions but overall let the users take the reign and think aloud during their exploration process.

Main takeaways from Ben's session: the application does exactly what we described it to do- a GIS application to give a at a glance of amenities in Dublin. The overall impression is that it's a very helpful application, easy to use and effective.
Points to improve are the loading times for the amenity points, perhaps directly being logged in after sign up to avoid repetitive steps, make the profile and tutorial icons more visible as they blend into the map, remove mac keyboard icons from the profile bubble, and if possible add more information on each amenity perhaps with tooltips, or add more amenity data like public transportation.

Score from survey:

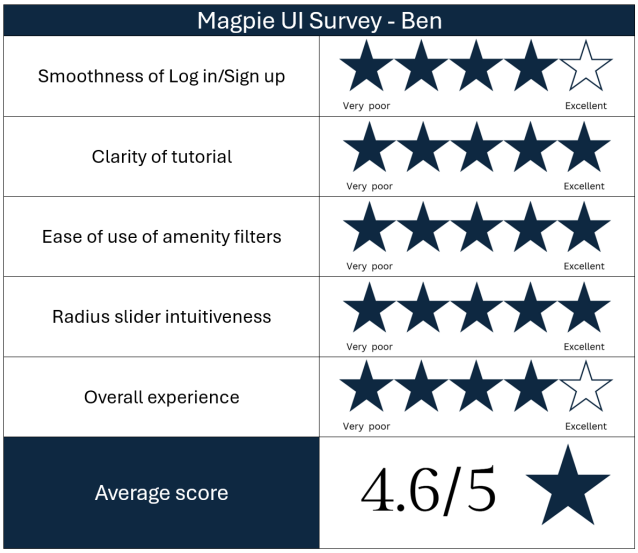

Figure 33: *User Evaluation - UI Score Ben*

User 6 - Jakub

Jakub is a professional with a construction and technological background. They were recruited towards the end of product development to test Magpie. They are considered a casual user.

This was an uncontrolled test session where Jakub discovered the application on their own, discussing each feature, testing each feature, and were then given a small scenario 'Put yourself in the shoes of an urban planner…' to obtain a different kind of feedback from previous sessions.

Main takeaways from Jakub's session: this tool simplifies the search for amenities however there are certain items that need to be considered to improve the application:

- **Log in/Sign up:** Email verification should be included, so that the user can confirm they have successfully signed up and also for security purposes.
- **Map:** some of the amenity data doesn't have accurate locations, for example public toilets seem to be off by longitude, and multi-storey parking data seems incomplete. Also, water fountains are very hard to find on the map, we should consider changing its colour. Same with the profile and tutorial icon, they are hard to spot on the map. They would also like to double click on the map to clear it, more intuitive for them. And last thing, amenities with small count are hard to find in the radius, maybe make them more visible somehow.
- **History feature:** doesn't see the use for a casual user, and again same for this tool, doesn't see the use for them as a casual user but could be useful for a target user.
- **Extra features:** Search functionality would be very useful for those that are lookin for a spot but don't know where it is located visually. Also, an export feature would be useful for the scenario of urban planning, if I'm to put a report together, a visual from this tool would be helpful for illustration.

A notable behaviour indicator from them was that they tried to interact with the elements during onboarding, as have previous users. Due to technical limitations, we have been unable to make that happen. Future work.

Overall, the tooltips for the icons is very interesting, a suggestion would be to add the type of parking, zoning information and tariff to the car parking amenity.

Score from survey:

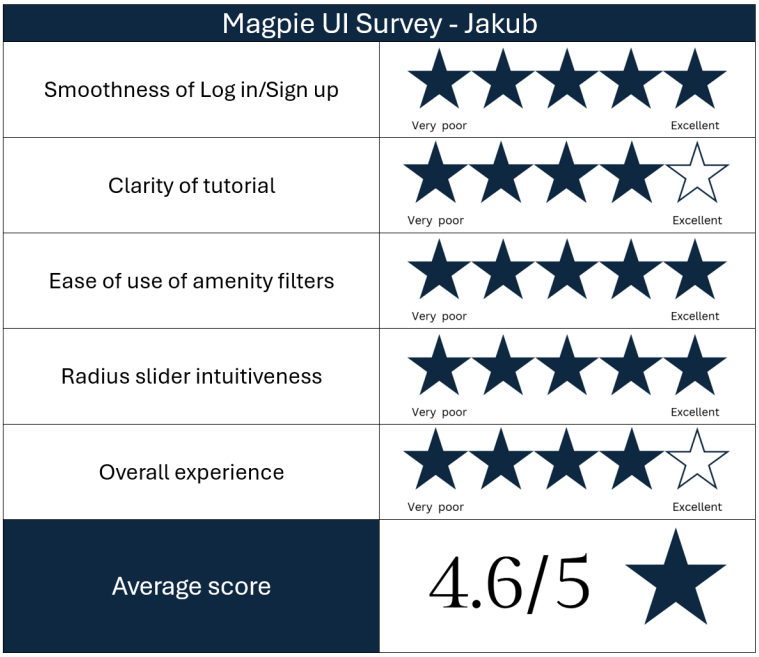

Figure 34: *User Evaluation - UI Score Jakub*

General user overview

General users provided very valuable feedback throughout the usability testing phase of the project. New features have been implemented, reviewed and removed thanks to these sessions.
For example, zoom buttons were added, tooltips were implemented and icons were changed. Most notable points raised during these sessions have been the want for more high level information on the amenities, more amenity data, the visual feedback from the system when it is loading the points and the onboarding.

Magpie UI Survey – General user Summary	
Smoothness of Log in/Sign up	4.67
Clarity of tutorial	4.33
Ease of use of amenity filters	4.50
Radius slider intuitiveness	4.83
Overall experience	4.17
Average score	4.5/5

Figure 35: *User Evaluation - UI General Score Average*

To conclude, general users gave Magpie a score of 4.5 out of 5.
There was no notable increase in score as the user evaluation progressed between the users, except in the beginning from the session with Brendan where we scored 3.6 out of 5 to the session with Anonymous 1 where we scored 4.6 out 5. Both users experienced the same version of Magpie, so the difference in score could come down to one personal preference.

0.6.3 Targeted user evaluation

Targeted users are users who would use Magpie as a tool for their work. We interviewed 3 individuals who fit this criteria.

Usability test sessions took the following format:

1. Getting to know
2. Introduction of Magpie
3. Exploring Magpie + Discussion
4. Satisfaction survey + End of session

Getting to know the professional user is the first step. It is relevant to the session because it helps inform the type of targeted user they are, their knowledge of technology, their experience using amenity data, the tools they have used and their perspective regarding amenities.

User 7 - Bryan Boyle

Bryan Boyle is a lecturer at the University of Cork in Occupational sciences and Therapy.
With a doctorate in computer science and occupational therapy, they have published several papers related to inclusivity in public spaces (**bryanboyleplaygroundinclusion2023**), and the role of technology in the lives of individuals with disabilities (**bryanboylechildrenautism2022**).

They left their contact in the market research survey. They are categorized as a *targeted user* because they use amenity data for their work, and Magpie could potentially be a tool they use. Firstly, we got to know Bryan Boyle, his professional experience and the amenity data he uses for his work.
He is mostly into research and therefore looks for datasets related to PHD topics of child development and inclusivity.

What do you use amenity data for in your professional life?	▪ Find out about school geography ▪ Explore amenities and facilities around schools ▪ Research into inclusivity and child development
What amenity data are you usually accessing?	▪ Location of schools ▪ Facilities around schools ▪ Transportation
How are you currently accessing this data?	▪ Data.gov mostly
What issues have you run into accessing and/or using this data?	▪ Accessing data that is not publicly available

Figure 36: *User Evaluation - Bryan Boyle information*

Next, we introduced Magpie and started the uncontrolled session.
The approach was to guide Bryan Boyle through the homepage whilst introducing Magpie, and then let them roam around on their own while discussing his their thoughts. They were able to complete most of the general tasks, except for zooming in and out. This is because they were not familiar with mouse technology, the onboarding was not clear enough and zoom buttons were not present on the map.

Here were the main takeaways from Bryan's session:

- **Data:** they mainly looked at amenities related to accessibility such as accessible parking, public toilets, water fountains, etc…They would like to see more data such as public transportation, facilities like schools, hospitals and stations. What really interests them is the type and quality of an amenity surrounding the specific location they are searching for, necessitating higher-level information to add to the tooltips.

- **General:** they really enjoyed using the search bar, they found it intuitive and quicker to use than placing a marker on the map. They are very interested in the project and look forward to seeing the final product.

- **Behaviour:** Bryan seems to have less than average proficiency with technology. Throughout the session, we noticed that first they used Edge as a browser, they got lost looking for the tab among the many windows they had open, their browser window was not full screen and additionally struggled zooming into the map.

The survey answers below complement his vocal feedback. **Overall**, Bryan found Magpie to be nice, simple and very straightforward, demonstrated by his 5 out of 5 score for the user interface. He would use it as a tool for his PHD research, mainly on desktop which further solidifies our user persona and why we prioritized desktop compatibility before mobile.

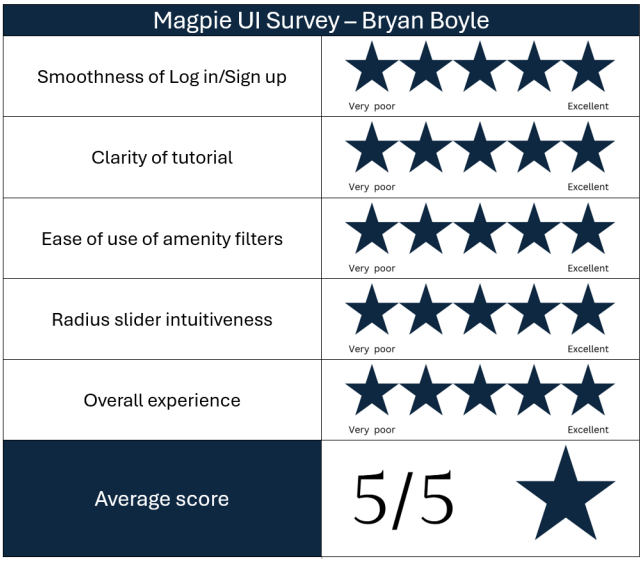

Figure 37: *User Evaluation - UI Score Bryan Boyle*

User 8 - Anonymous 2

Professional user Anonymous 2 (Sarah) has a background in urban planning, transport planning and education. We seeked out Anonymous user 2 through social media channels and they agreed to take part in a usability testing session.
They mostly use amenity data to understand an existing area that has plans for upgrade or demolitions, as well as studying transportation routes to plan new stops and service new areas.

What do you use amenity data for in your professional life?	▪ Explore amenities/facilities along planned transport routes ▪ Understand components of existing area for planning
What amenity data are you usually accessing?	▪ Bus routes ▪ Schools ▪ Zoning information ▪ Public spaces ▪ Public transportation
How are you currently accessing this data?	▪ Myplan.ie ▪ Local development plans online ▪ Google maps ▪ County council datasets
What issues have you run into accessing and/or using this data?	▪ Little information on the true level (quality, quantity, capacity) of amenities

Figure 38: *User Evaluation - Anonymous 2 information*

Next, Anonymous 2 loaded the Magpie application and started exploring, whilst discussing each and every feature. They were able to complete all general tasks while skipping a few.
One question they asked was really interesting: *how do we define amenities?*
Different professions within urban/transport/city planning define amenities in different ways, and they found our way of defining it interesting and slightly different from the way they would've defined it: *'Amenities are any sort of public facilities that you can use.'*

Here were the main takeaways from Anonymous 2's session:

- **Data:** the tooltips on each icon is very interesting. Perhaps cross-check the information there and add/remove details. For example, the bike sharing amenity tooltip has name, number and address but the number seems to be an id and not the number of bikes at the bike sharing station.

- **Additional features:** for their work, Anonymous 2 likes to have a satellite view of the area they are inspecting. They found that when they are zoomed in at the street level, they find it hard to understand the map. Perhaps adding a satellite layer for enhanced visualisation.
  Also, an export feature would be amazing, with a scale and table of the amenities found on the location selected on the map.

- **Miscellaneous:** the car parking amenity has lots of potential because it relates to the use of public space which is very contested and a big issue in urban planning. Magpie displays that information in an easy manner and there is so much potential to expand the parking detection and help fuel research on how much cars take up public space in cities and inefficiencies using public space for parking especially with limited land.

Overall, this is a fascinating and really useful project in its own right. The radius slider is great, the amenity count is very useful, Anonymous 2 can see Magpie being used for planning and research not so much casual user. Points to improve would be more amenity data and revise the information on the amenity tooltips. They scored Magpie's UI a 4.6 out of 5, the tutorial and the filters losing a star for lack of clarity and intuitiveness.

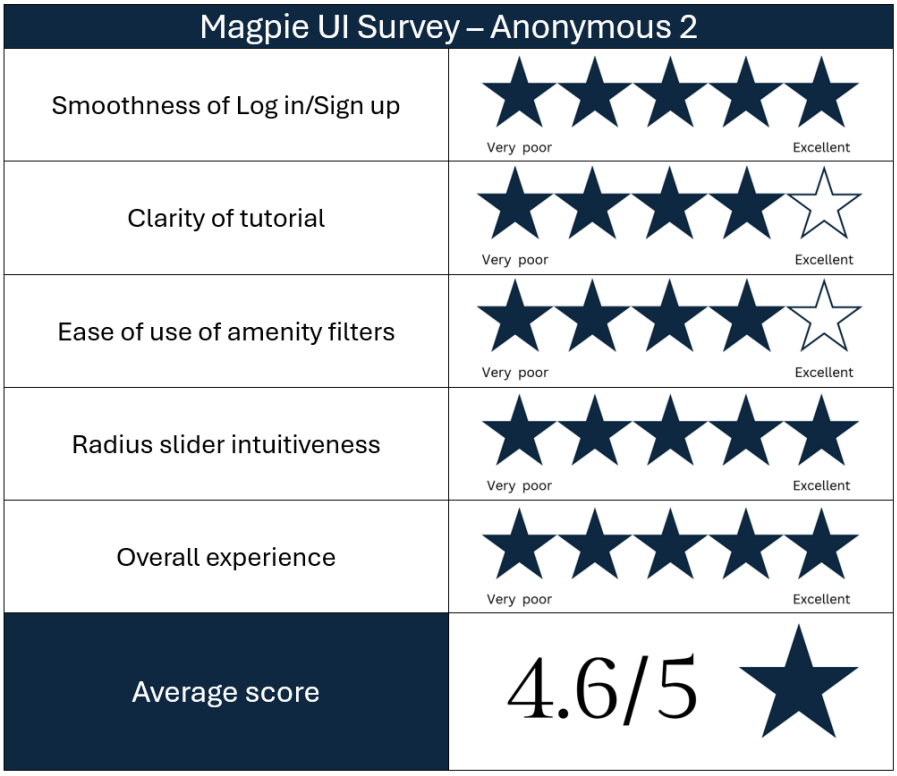

Figure 39: *User Evaluation - UI Score Anonymous 2*

User 9 - Anonymous 3

Professional user Anonymous 3 (Odran) has a background in town planning, economics, local development and education. We seeked out Anonymous user 3 through social media channels and they agreed to take part in a usability testing session.
They mostly use amenity data for socio-economic research and consulting for community project developments.

What do you use amenity data for in your professional life?	▪ Consulting for community engagement ▪ Retail planning ▪ Rural planning ▪ Socio-economic research
What amenity data are you usually accessing?	▪ Socio-economic data ▪ Planning application information ▪ Development maps
How are you currently accessing this data?	▪ Google maps ▪ County council datasets ▪ Publicly available datasets ▪ On the ground
What issues have you run into accessing and/or using this data?	▪ Data doesn't exist ▪ Data not publicly available

Figure 40: *User Evaluation - Anonymous 3 information*

Before the session started, Anonymous 3 told us they qualified their technological proficiency as 'very low' and to be patient with him. This was confirmed when he struggled to share his screen at the start of the session, however he was very comfortable navigating the map, zooming in and out and using the features of the dashboard. Here were the main takeaways:

- **Amenities:** some minor content changes related to the labelling of amenities, for example 'library' should be 'public library', 'public wifi' should be 'free public wifi' etc…Also, it would be beneficial to differentiate the types of bike sharing system (Dublin Bikes, Bleeper, Moby Bikes).
  More data should be added like schools, public transportation, zoning information, vacant buildings and population data to calculate amenity per capita for example.

- **Features:** giving the user the chance to draw their own radius could be very interesting. Also, adding a '300 meters' preset for the radius slider would cater to planners more because of the 15 minute walkable rule.
  This rule defines that 300m is a walkable distance for abled individuals and is used a lot in retail/town planning.

- **Miscellaneous:** Anonymous 3 suggested we take a look at the website *Dublin Smart City* and *POBAL* for more datasets relevant to our project. Improve Magpie with the suggestions above could help with research into evaluating the walkability of Dublin city.

Overall, Anonymous 3 gave some really interesting feedback, different from the other two targeted users. They believe Magpie has commercial potential but needs a lot more datasets and a lot more details to turn this into the go-to tool for planners in any sector. They expressed how they would've enjoyed having this tool earlier on for their consulting work.

They scored Magpie's UI a 4.6 out of 5, the log in and sign up losing a star for lack of email verification and smoothness.

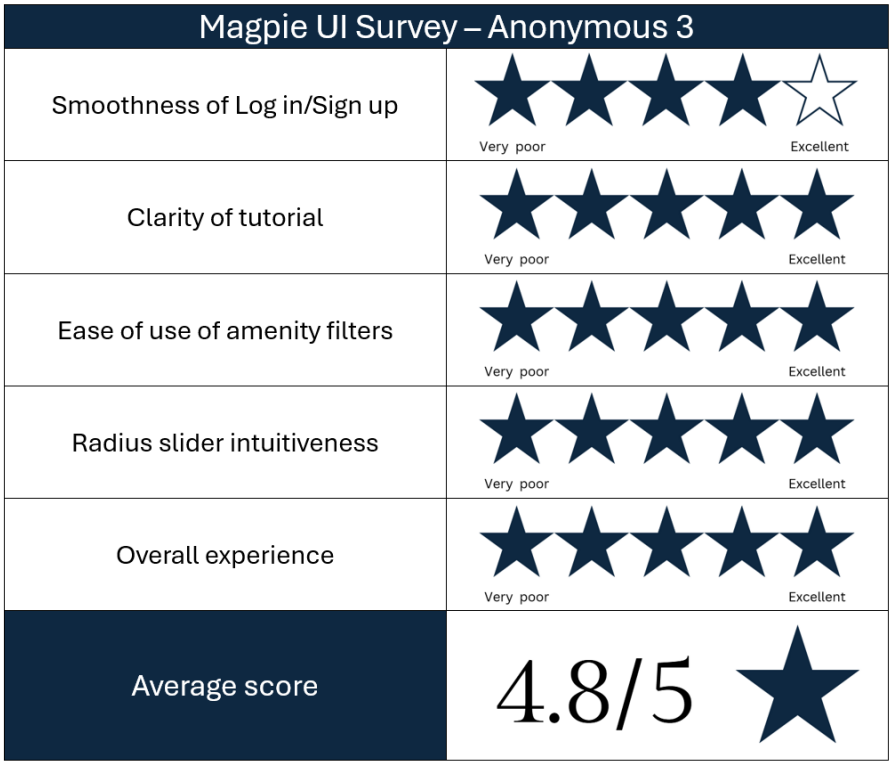

Figure 41: *User Evaluation - UI Score Anonymous 3*

Targeted users overview

Interviewing these three users was a really amazing experience. It's easy to put all planners into one basket however, they all have different needs, experiences and wants and these sessions really opened our eyes on that.

The main changes these users would like to see are **more data**, more **precise information** on the amenity tooltips, an **export feature** and improvements to the onboarding.

The scores they gave on Magpie's UI averaged to 4.8 out of 5, which is incredible. This score validates Magpie's project vision and end goal: to create a easy to use tool to give a glance view of amenities in an area.

Magpie UI Survey – Targeted user Summary	
Smoothness of Log in/Sign up	4.67
Clarity of tutorial	4.67
Ease of use of amenity filters	4.67
Radius slider intuitiveness	5.00
Overall experience	5.00
Average score	4.8/5

Figure 42: *User Evaluation - UI Target Score Average*

0.6.4 UI/UX Expert Review

The goal of this review is to evaluate the user interface of Magpie at different stages of development with regards to key UI/UX general guidelines.
UI design guidelines differ from UX ones because of their fundamental differences: one is user based and the other isn't. **UI design** aims to create a visually appealing, intuitive and userfriendly interface that allows for easy navigation and interaction with the application.
On the other hand, **UX design** aims to create a meaningful user experience through designing an entire user journey. Together, they work to shape the path to a successful and enjoyable digital product (**uiuxguidelines2023**).

10 major guidelines were devised in the 1990's by Jakob Nielsen, the 'father' of web design. These guidelines have shaped several UI and UX design creations in the last decade (**uiuxguidelinesnielsen2016**). Below are the ones outlined as most relevant to Magpie's core:

- **User control & freedom:** the users should be able to retrace their steps such as undoing and redoing previous actions, exiting processes midway etc…
- **Recognition over recall:** ensure the system is not heavy on user's cognitive load and prioritizes easily recognizable design items over one's that rely on memory
- **Aesthetic & minimalist design:** keep the visual information to a minimum, display only necessary components and clearly visible unambiguous means of navigating through the application
- **Help & documentation:** provide documentation that is easily accessible, relevant and worded in a way that will guide users to the desired outcome

We requested a review of our system from UI/UX professional Andrea Curley.
A professor at a top technological university, specialised in web development, web design and user experience.
Two session were conducted: one on November 13 after the publication of our first minimum viable product, and the other on December 9 at the end of the development timeline. Both sessions were conducted online through a videoconference meeting on Teams where we presented Magpie, explored the features, discussed them, and administered a questionnaire at the end.

The questionnaire is divided into 3 sections pertaining to *visual design*, *information architecture*, and *compliance*.
Different types of questions were included, such as 'Closed' 'Scale' and 'Open' to allow for both the quantitative and qualitative measure of Magpie UI.
Open-ended questions tend to require more cognitive efforts, thus answers can vary in quality and quantity depending on the individual. Close-ended questions on the other hand offer standardized responses which can be more easily quantified and require less cognitive effort (**mixsurveyquestions2020**);

Session 1

The first session provided very valuable insights on Magpie's workflow, user interface and technical components, as well as helped us uncover critical bugs. These were the main takeaways:

Landing Page: Upon loading Magpie, Andrea Curley was directly taken to the mapview, which was not supposed to happen. After the review, we investigated the cause of this event and uncovered a bug in the authentication which we have been working on. Following this event, she suggested creating a landing page or some sort of introduction to ease the user into discovering Magpie.

Onboarding: Due to the bug explained above, the onboarding did not automatically start as it should have upon login. Nevertheless, Andrea Curley said that the user may want to intuitively press on the elements being highlighted during the tutorial, as she tried to do. This adds to the feedback received during casual testing for the implementation of this feature. Unfortunately, due to a technical limitations we are not able to solve it, only provide make certain changes to dissuade the user of doing so.
In addition, she suggested there should be an option to exit the tutorial at any time for users who don't want to sit through it. Lastly, the tutorial should be more visually striking and engaging in order to leave a lasting impression on the user.

Dashboard & Map: Currently, the hierarchy of items on the dashboard does not make sense to the average, and is not intuitive to use. All the amenities are displayed when only one is selected (as shown in the figure below) and their count displays zero, which the user might interpret as there are zero other amenities in the area in addition to the one I selected.
The icons on the map are not visible enough, and zooming in & out on the map may not be intuitive to the range of users and devices. Adding zoom buttons could help bridge that gap.
At the moment, Andrea Curley noted that there is a disconnect between the map and the dashboard whereas they should be looked as one. She suggested adding amenity icons to the dashboard to help bridge that gap.

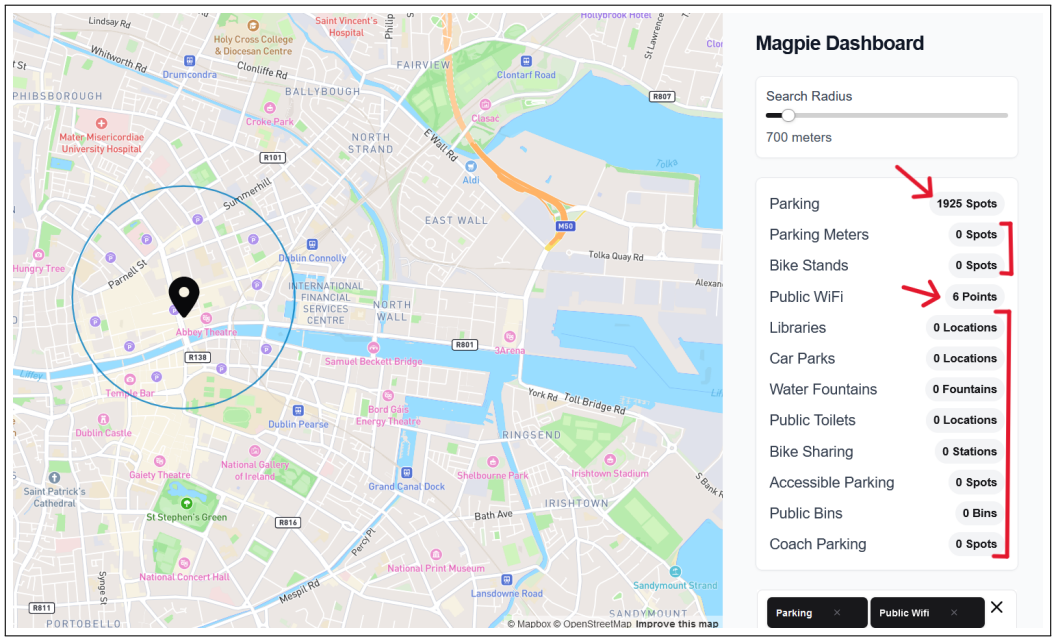

Figure 43: *V.0.1 Magpie Dashboard & Map when 2 amenities are selected*

Filters: If there are no amenities found in the radius of search, a message should pop up to tell the user so. Currently, it is not very clear if there are amenities present in the chosen area especially due to the small size & faded color of certain icons.

Log in/sign up: When trying to log in with credentials that don't exist, the system should return a proper error such as 'username doesn't exist'. Log out and account sign up went smoothly. Andrea Curley questioned the benefits of logging in for Magpie, to which we stated: Magpie was conceived with the idea of providing a service to working professionals; therefore logging in will allow the implementation of further features such as safeguarding their previous searches, storing exported reports, connecting with other members of your organization and much more.

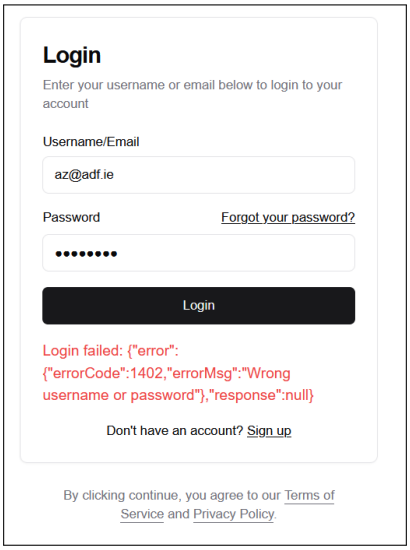

Figure 44: *Error message when using inexistant username and password*

Survey response: Below are the answers to the expert review survey. The response helps complement the oral feedback received during the session and provide some quantitative data as a baseline for the next evaluation. A score has been attributed to each section of the survey based on the responses from Andrea.

The first section covers usability of the main items of Magpie's user interface which scored 3.5 out of 5. The items which brought the score down are the onboarding and the clarity of the map visualization, further supported by the vocal feedback Andrea provided on the un-intuitive flow and disconnect between the map and the dashboard.

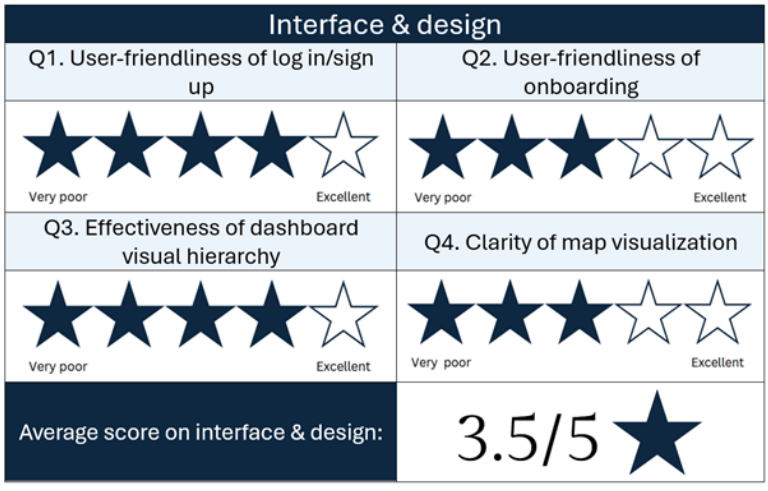

Figure 45: *UX review 1 - UI score*

The second section covers the information architecture and data quality of the amenities. It looks at the features displaying the information and the score reflects the problems aforementioned with the dashboard as well as the incomplete profile menu. Furthermore, Q8 asks Andrea to assume the value of Magpie as a tool for different use cases where our target users (Urban planners and Event planners) were rated as 'Valuable'. This rating is useful but it remains an assumption.

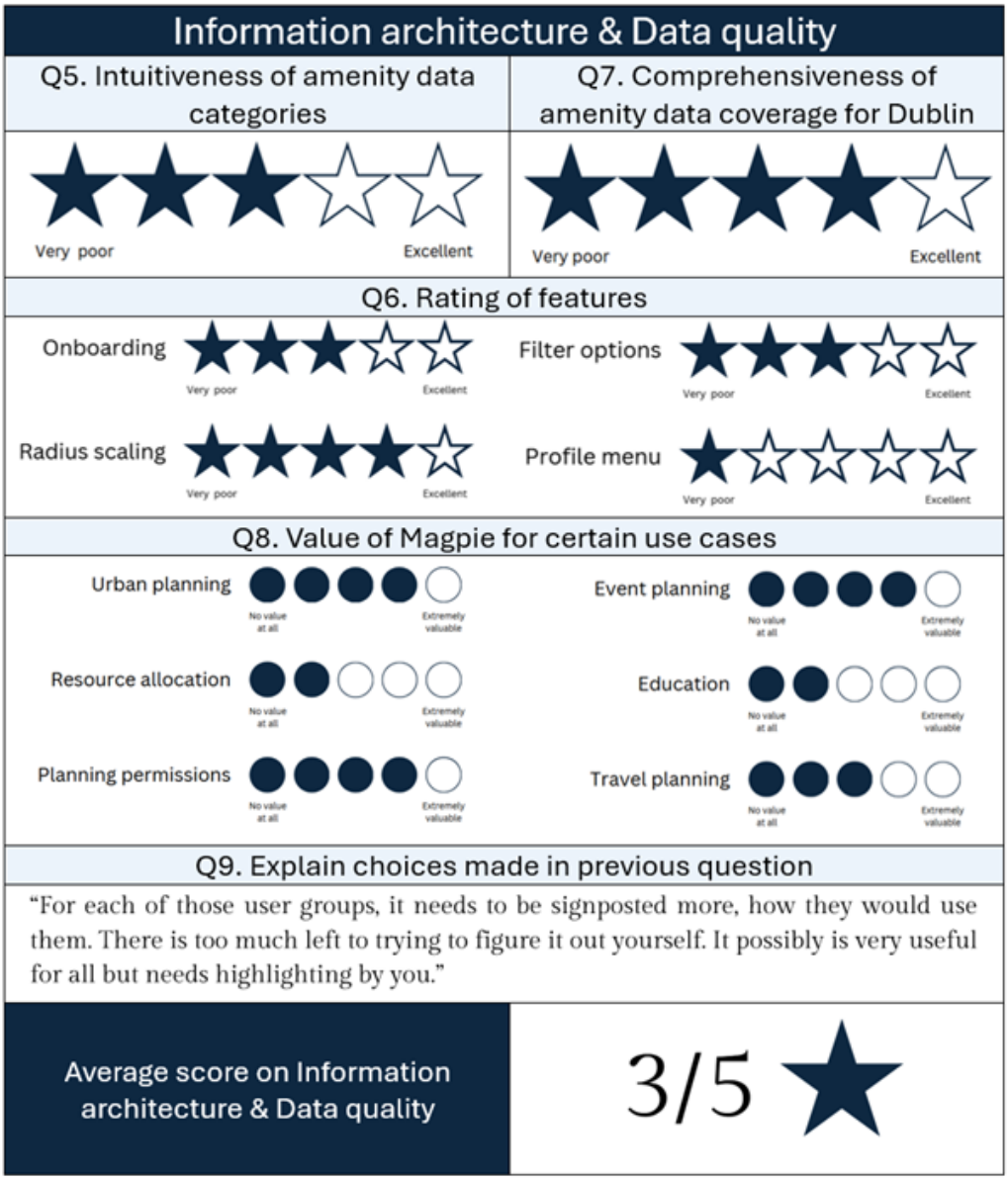

Figure 46: *UX review 1 - Information Architecture score*

Lastly, the third section covers the technical aspects of Magpie. The low score of 2.88 out of 5 reflects a major authentication bug encountered during the testing session, as well as severe lagging of the points on the map due to technical difficulties.

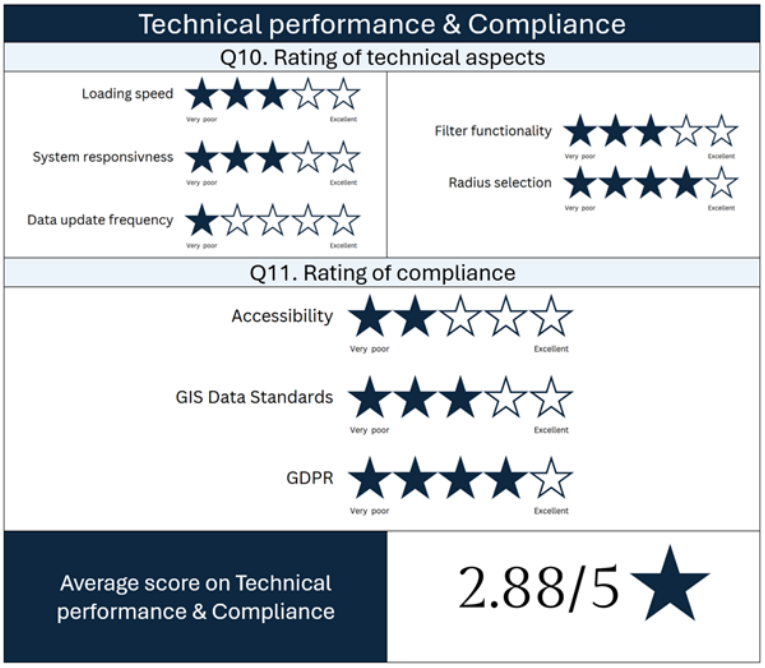

Figure 47: *UX review 1 - Technical Performance score*

Overall, Magpie scored 3.13 out of 5 for this first UX review session. This is the baseline, and the objective for the next session is to score above 3.5 out of 5 overall, and significantly improve the scores for the onboarding, the dashboard flow and the system responsiveness.

UX review – Session 1 scores	
Part 1	3.50
Part 2	3.00
Part 3	2.88
Total	3.13/5

Figure 48: *UX review 1 - Overall score*

Summary: To conclude the first session, Andrea Curley found our interface sleek and minimalistic. However, she suggested that if we want to remain with this style, we need to ensure there is as little room as possible for ambiguity and confusion. The user needs to find it easy to move from one feature to another and understand the triggers. Currently, Magpie looks so sleek that the user may not be able to see what they want.

Session 2

The second session informed us of the points we were able to improve on from the first session as well as features to be looked at in the future. These were the main takeaways:

Landing page: The new landing page provides a proper introduction of Magpie, however some adjustments can be made to the navigation bar, some visual feedback to let the user know the button they have clicked works; such as a bold overlay on the navigation item or different colour highlight. In addition, if you really want to push Magpie as a GIS application for professional urban planner users, it needs to be put at the forefront.
Lastly, a log in button should also be present on the main page alongside the ”sign up” button.

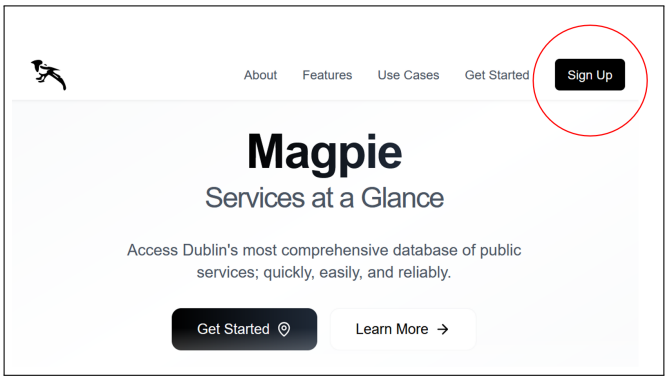

Figure 49: *Magpie V.0.12 - Landing page sign up button*

Dashboard: The rearranged dashboard has immensely improved the flow of selecting/deselecting amenities, adjusting the radius and clearing the map entirely of the points. The disconnect between the map and the dashboard has been addressed with custom icons and colours and labels.
Some suggestions to improve user experience would be to make the amenity row clickable so as to make it easier for the user to select and deselect it, as opposed to having to click the eye directly.

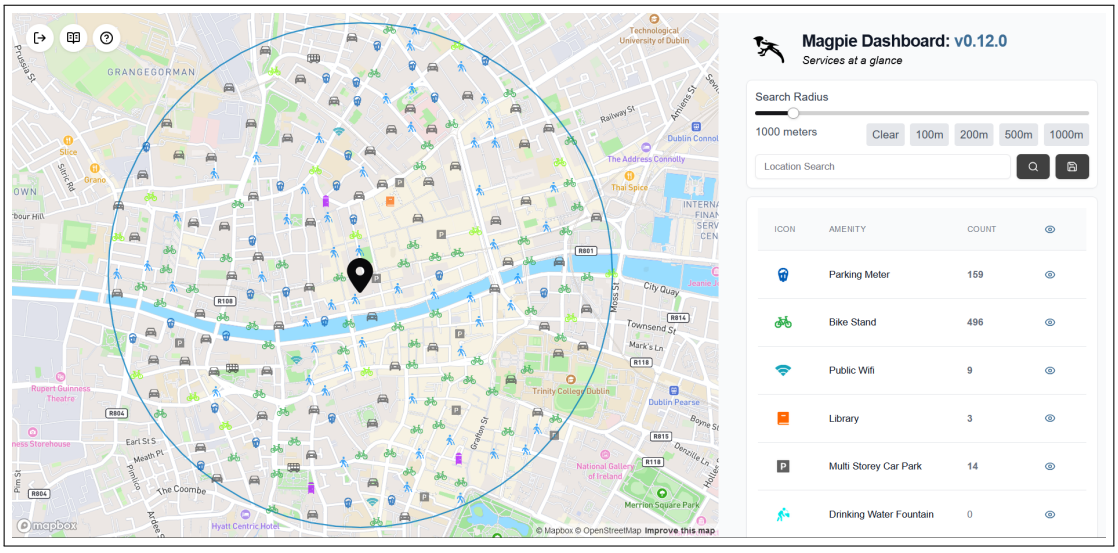

Figure 50: *Magpie V.0.12 - New Dashboard*

Map: The map looks very clean and the points are loading much better. However, one small issue to mention is the amount of icons being loaded within the search radius. At the moment, the icons group together under one when the view is very zoomed out, and expand and disperse the more you zoom in. However, certain icons like the 'Public Bins' are more in numbers visually than any other.
This is due to the way we have rendered the points on the map, which initially did not support it so we had to find a way around it through the layers. The items in the top most layer appear the most and 'Public Bin' is at the top right now. This is also something that will be detailed in future work.

Figure 51: *Magpie V.0.12 - Amenity Icons grouping relative to zoom level*

Search & Save functionality: Good addition however it gives inconsistent results. This is due the the API we are using which does a keyword search through our database to retrieve results instead of a semantic search. Therefore street names with special characters or with many words may not always return accurate or any results unfortunately, or why if the search is saved it may not carry the search term.
Last suggestion on this point is to be able to press enter to start the search, instead of being confined to the search button. The search feature will be further addressed in the future work section of the report on how to switch it to a semantic.

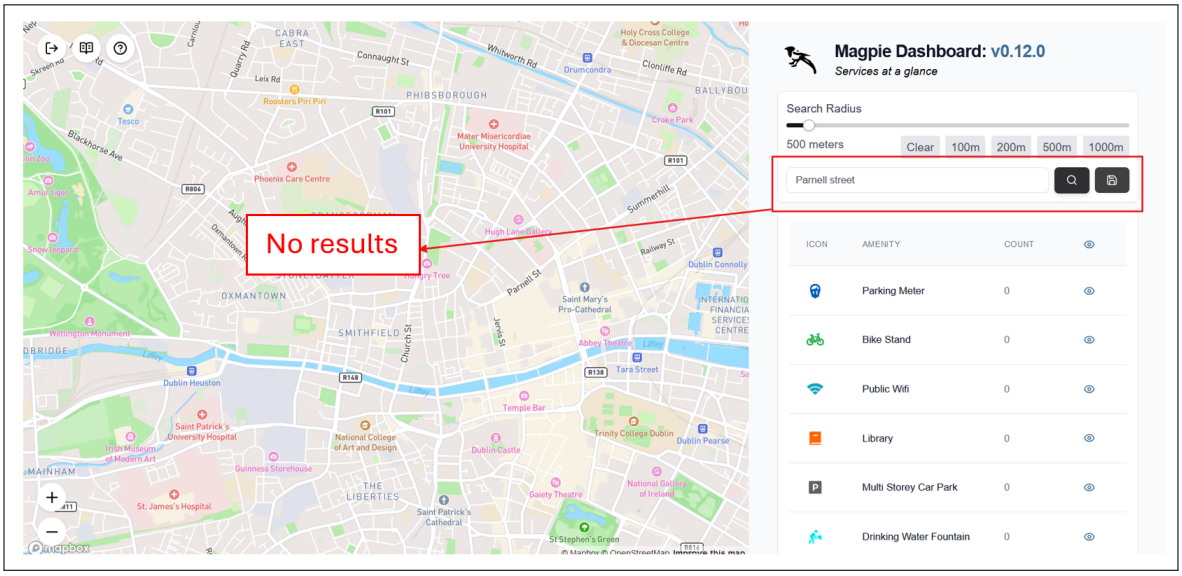

Figure 52: *V.0.12 Magpie Search Failing*

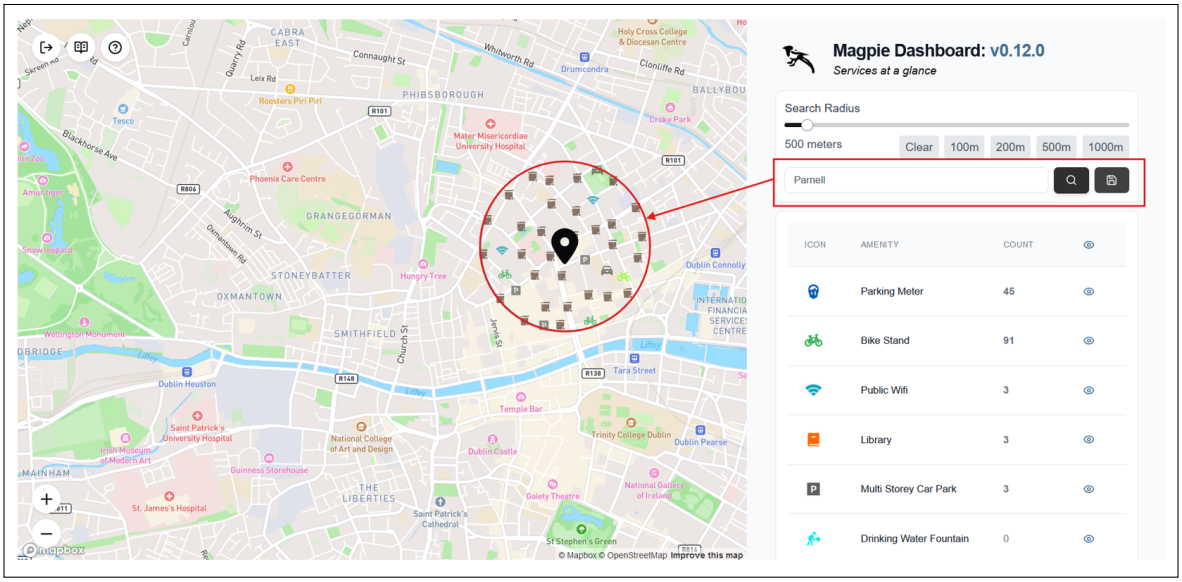

Figure 53: *V.0.12 Magpie Search Working*

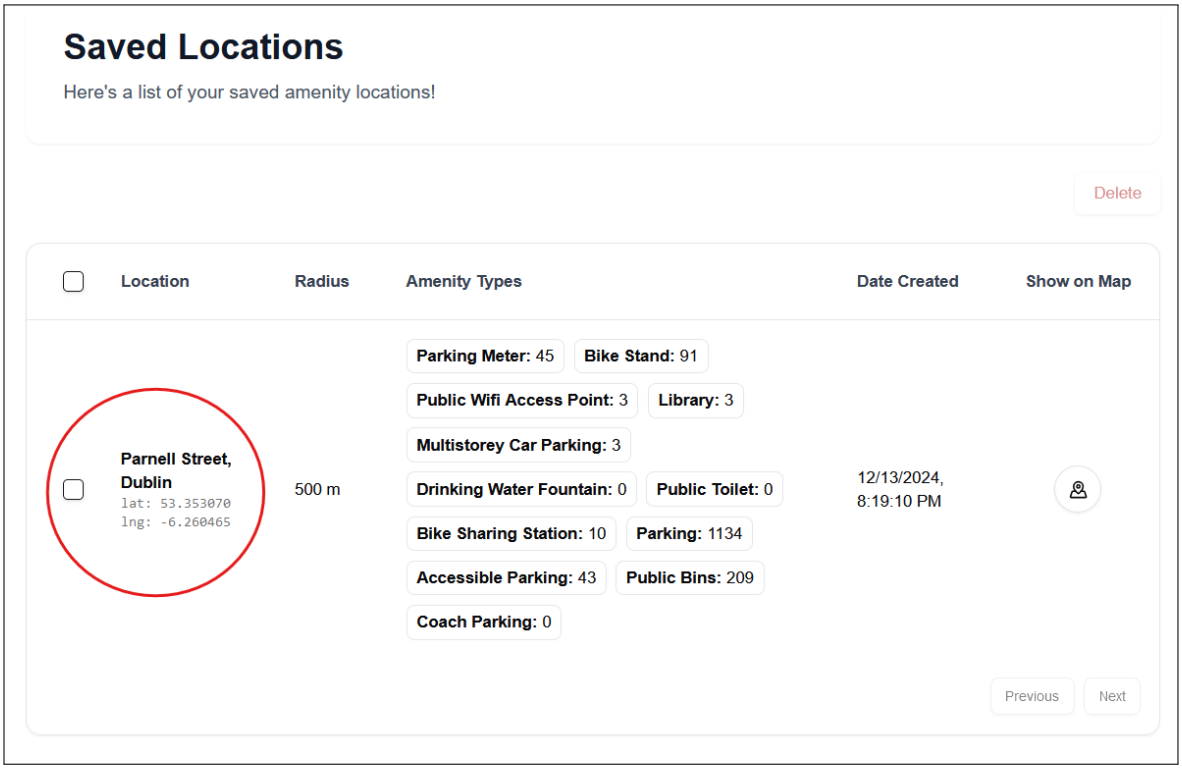

Figure 54: *V.0.12 Magpie Saved Location*

Survey responses: Andrea Curley scored the UI of Magpie a 4 out of 5, which is a net increase of 0.5 points from the first session. The items which gained half a star are the onboarding and the map.
Changes made which contributing to this increase are the fixing of the overlapping elements during onboarding and the updated icons with tooltips on the map.

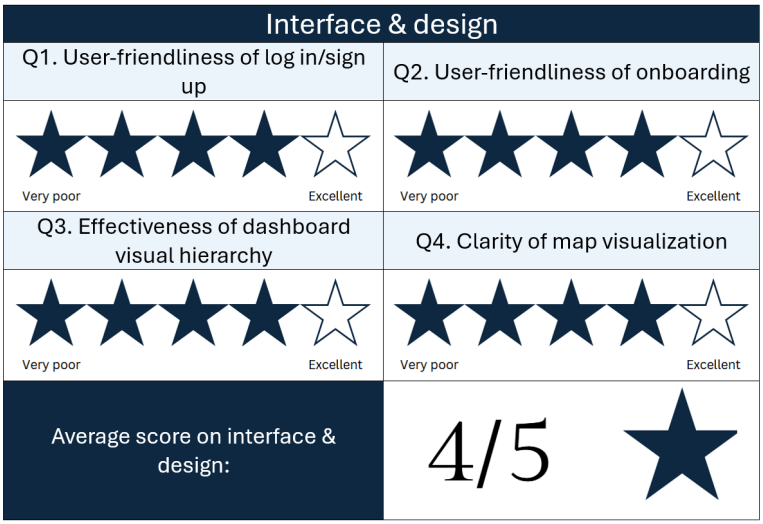

Figure 55: *UX review 2 - UI score*

The second section scored a 3.63 out of 5, a net increase of 0.63 points from the first session. Points which improved are the intuitiveness of amenity categories complemented by adding the icons beside the amenity name, the profile menu with brand new visible icons, and improved onboarding mentioned previously.
In addition, Andrea Curley increased the value rating for resource allocation and planning permissions, due to the discussion we had during the second session regarding the sessions I had with urban planners.

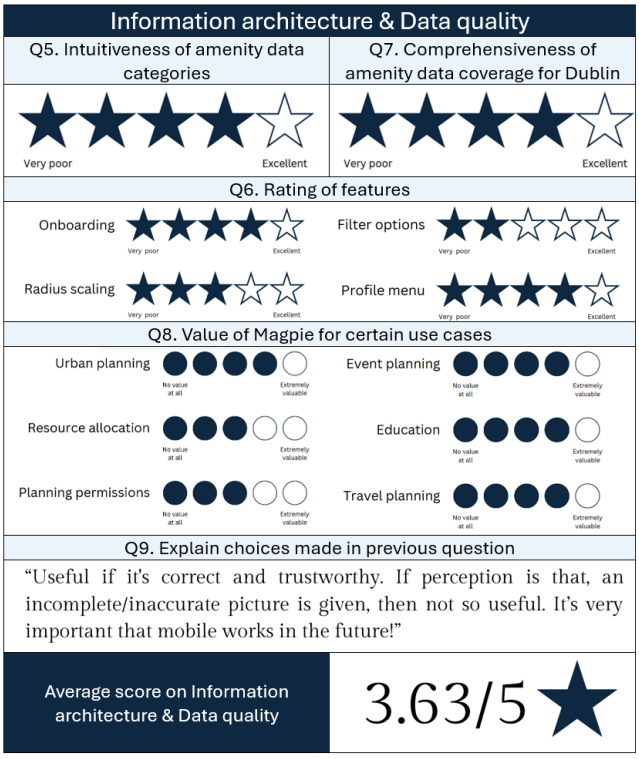

Figure 56: *UX review 2 - Information Architecture score*

Lastly, the third section scored 3.1 out of 5, a good increase from the first session. This is reflected by the solve of the authentication bug and the feedback applied from the first session to improve the UI and accessibility.

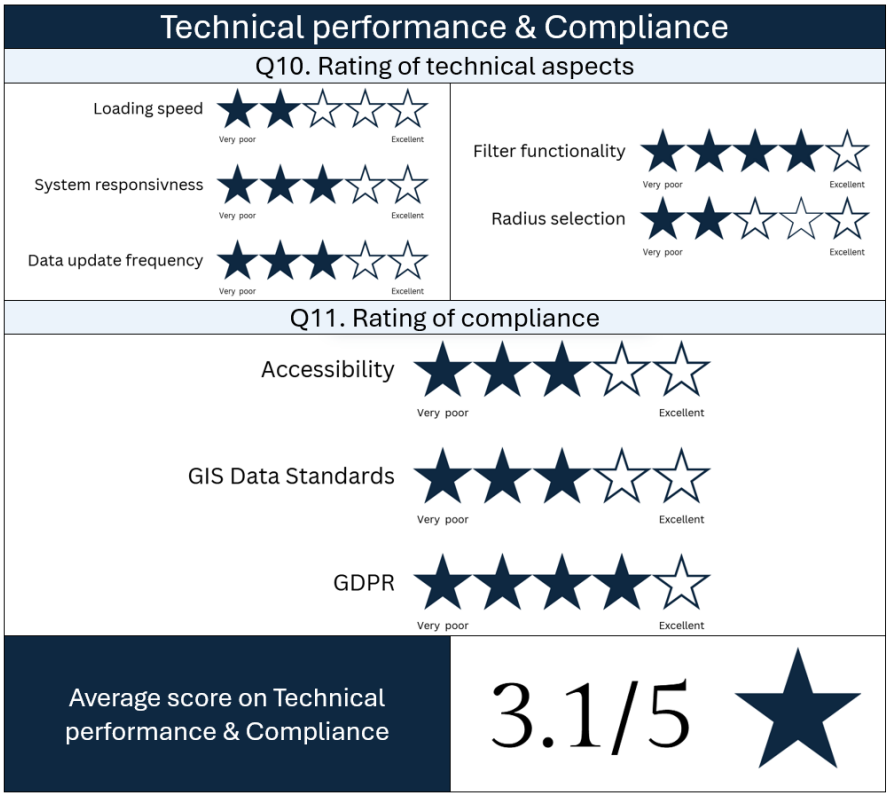

Figure 57: *UX review 2 - Technical Performance score*

Overall: The feedback from the previous review has been onboarded very well, the application looks clean and is easy to use. The goal of surpassing 3.5 out of 5 has been achieved with a total score of 3.58 out of 5. However, more visual feedback is needed for features like the search functionality and the selection/deselection of amenities, as well as the accuracy of the points displayed.

UX review – Session 2 scores	
Part 1	4.00
Part 2	3.63
Part 3	3.10
Total	3.58/5

Figure 58: *UX review 2 - Overall score*

0.6.5 Accessibility Expert Review

In the context of this project, accessibility refers to considerations for users with visual, auditory, cognitive and/or physical impairments; inclusiveness refers to considerations for users for whom English isn't their first language, older users and users with gender or affective issues; and universal design refers to compatibility of Magpie with different devices, browsers and networks.

The accessibility general guidelines we'll be evaluating Magpie on are from the EU Web Accessibility Directive which directly affect public bodies in Ireland although map systems are exempt. This directive came into effect at the end of 2016 and aims to *make public sector websites and mobile applications more accessible, and to harmonise varying standards within the European Union (…)* (**webaccessibilitydirective2016**).

Scope of directive	Public sector bodies website and mobile applications across EU member states
Compliance requirements	Web Accessibility Content Guidelines (WCAG) 2.1 Level AA • Perceivable: content with text alternatives, subtitles, captions • Adaptable: meaningful design across different devices and screen sizes • Distinguishable: contrasting elements and colours, sizable text, content on hover • Operable: interface with support for keyboard navigation, assistive devices • Navigable: consistent & predictable user interface, easy to understand language
Exemptions	• Limited functionality websites (legacy) • Websites not for essential public services • Websites with content that cannot reasonably be made accessible • Disproportionate burden

Figure 59: *WCAG 2.1 Scope, Guidelines & Exemptions*

There is an inherent limitation of accessibility for mapbased systems due to their visual nature.
Visual impairment is categorized as having a type of eye disorder and/or a degree of vision loss; an estimate of 2.2 billion people globally have vision impairment (**whoworldreportvision2019**) and 297,000 (5.6%) of the population in Ireland(**visionirelandcensus**).

The paper by **accessibilitywebmapsrecommendations2017** which relied on data from two projects aimed at developing webmap applications for visually impaired individuals has presented several recommendations, the following which we will guide Magpie's accessibility evaluation and subsequent iterative development:

- **UI components:** Creating a UI that is simple, understandable and follows a clear predictable layout. Implement only necessary control elements, group them by similar focus and place them in areas that are familiar (similar in other programs). The UI should be flat while avoiding dropdowns, scrolling and nested/overlapping elements. Control elements overlapping the map makes it difficult for visually impaired users to read the map and use the controls, so this practice should be avoided.

- **UI design:** Size and colour of UI components should be chosen to provide high contrast between different elements; if symbols are used they should be easily recognizable and complex backgrounds should be avoided.

- **UI language:** Familiar and easy recognizable terms should be used on the interface so as to not scare off users with unknown technical terms and make the application accessible to all.

- **UI interaction:** A user should be able to interact with the interface using both a mouse and keyboard on desktop; keyboard accessibility is key for improving accessibility for visually impaired users.

We requested a review from Accessibility expert Damian Gordon.
He served on the board of the National Disability Authority, who advise the Irish Government on all matters related to disability, and he has worked with a wide range of disability organisations, include the Centre Remedial Clinic, the National Council for the Blind, Arthritis Ireland, and the AgingWell Network. He has contributed to the development of hardware, software, legislation and training related to disability awareness and accessibility.

The goal of this review is to evaluate the accessibility, inclusivity & universal design of Magpie with regards to the guidelines and recommendations detailed above.

The review was conducted on November 21st during the usability testing phase of Magpie. The session was conducted online through videoconference meeting on Teams where we presented Magpie, gave Damian Gordon a scenario to follow to explore the application, discuss the scenario and the accessibility guidelines and then a survey.

The scenario given can be seen in the table below. A scenario was given so as to simulate the use of Magpie by one of our target users through the lens of an accessibility expert.

The questionnaire covers UI design, interaction and language and aims to complement the vocal feedback of the session. It is made up of 3 sections with a mixture of closed questions rating different parts of the system, and open questions to provide more qualitative feedback.

We are also observing Damian Gordon's behaviour and if he was able to complete general tasks related to the main features of Magpie. The results can be seen in the table below, it recorded if they were able to complete the task, how difficult it was for them based on behavioural cues and any errors bugs encountered during the task. We were not able to record the time taken for each task due to the informal administration of them, the user was not aware we were 'grading' them in a sense.

The difficulty of the task is related to if they were able to complete it and how much they struggled during it. The status of a task can either be 'complete', 'pass' or 'fail' where 'complete' is attributed when the user does the task on their own, 'pass' is attributed when the user was able to complete the task but with our help, and 'fail' when the user were not able to do the task even with our help.

The review provided some useful insights with regards to missing features to complete the scenario and areas of Magpie that comply with accessibility standards. Below were the main takeaways:

Dashboard: Damian Gordon had some trouble locating the recreation centre during the scenario, and advise that have a search bar would remove the difficulty and give options for those who may be visually impaired to find a location on the map, and also for those who may not know where the area they're looking for is located.

Onboarding: Damian Gordon found the steps in the onboarding very wordy and too long, which may cause some users to skip through or not retain the information present there. In addition, some of the wording could be improved related to navigating the map as shown in the figure below. This issue was further illustrated by the difficulty Damian Gordon encountered trying to zoom in and out on the map using his mousepad, having not understood how to do it from the onboarding step 5 that explain how to do it with a 'mouse' and a 'touchpad'. Addressing the choice of words, the content and flow of the onboarding is important to ensure it is understood by all kinds of users.

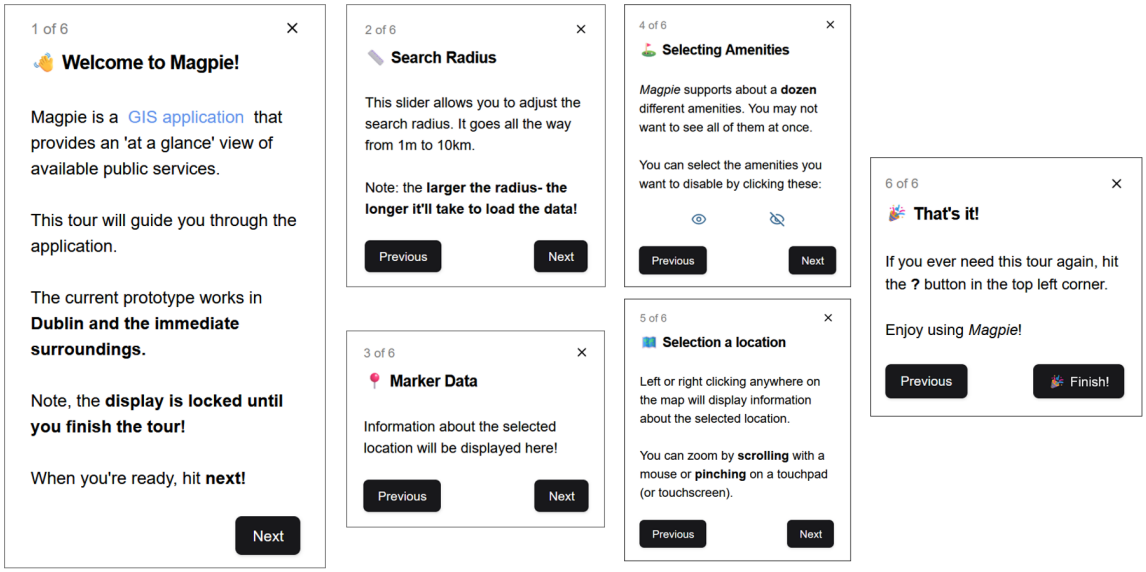

Figure 60: *V.0.1 Magpie Onboarding Steps*

Map: Going back to the zooming feature Damian Gordon struggled with, adding zoom in and zoom out buttons directly on the map would offer an alternative to those not able to use the mouse for the action, further improving Magpie's accessibility.
In addition, the profile and onboarding icons blended into the map which made it difficult for Damian Gordon to go back to the onboarding or log out as shown in the figure below.

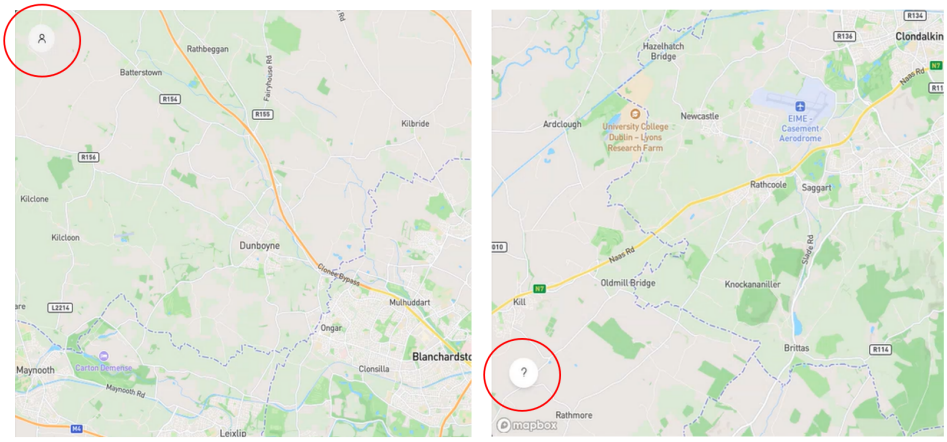

Figure 61: *V.0.1 Magpie Profile & Onboarding icons*

Lastly, his survey responses confirmed that Magpie adheres to the WCAG guidelines detailed above as well as fulfils the major goal of ease of use. The survey covered 3 major sections: UI components, UI Design & Language and UI interaction.

UI Components evaluate the position, quantity and accessibility of control elements such as the navigation buttons, the map controls, the dashboard toggles, etc…Damian Gordon rated this section a 4.75 out of 5, the placement of control elements loosing one point because some of them were nested and therefore hard to notice or navigate to, like the profile and tutorial icons on the map mentioned above.

UI Components	
Q1. Clarity of interface layout	Q2. Placement of control elements
★★★★★ Very poor Excellent	★★★★☆ Very poor Excellent
Q3. Ease of use of control elements	Q4. UI design for visual impairment
★★★★★ Very poor Excellent	★★★★★ Very poor Excellent
Q5. Additional comments	
"The system aligns well with the WCAG guidelines. The layout and navigation are clear and consistent. The system is compatible with the most popular screenreaders."	
Average score on UI components:	4.75/5 ★

Figure 62: *Accessibility review UI Components Score*

UI Design & Language covers the style of UI components as well as the textual language used on the interface. Damian Gordon rated this aspect of Magpie at 5 out of 5, noting that there is a clear and effective use of language and icons which both are easily recognizable and clear.

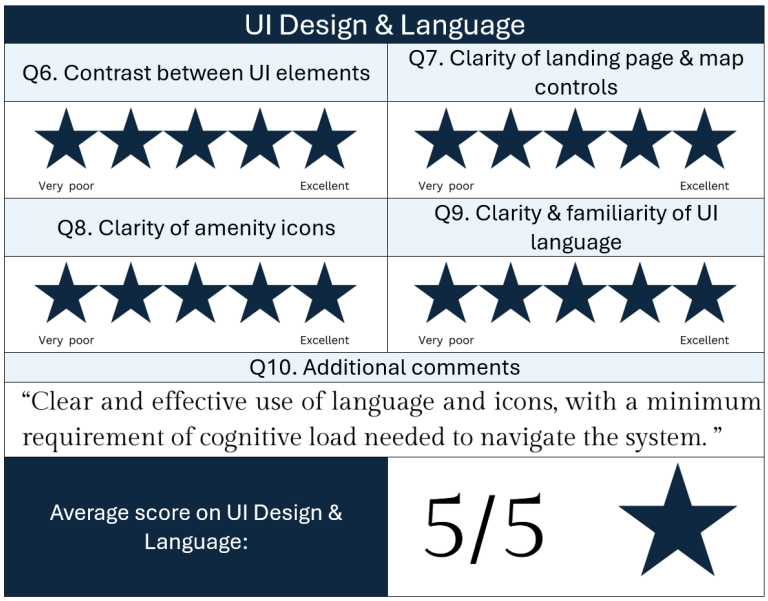

Figure 63: *Accessibility review UI Design & Language Score*

The last section, **UI Interaction** covers the range of interaction possible with the Magpie UI using different hardware and software tools. Damian Gordon also rated this section a 5 out of 5, citing good compatibility with industry standard screen readers and ease of navigation with both keyboard and mouse.

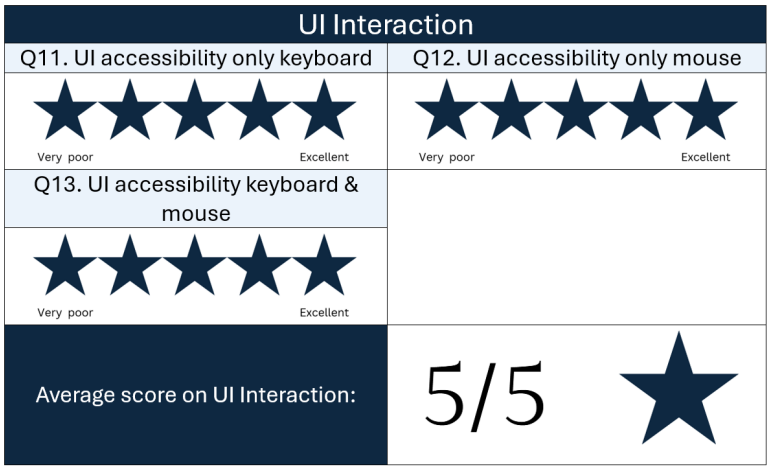

Figure 64: *Accessibility review UI Interaction Score*

Overall, Magpie scores a 4.92 out of 5 score in accessibility, with perfect scores in both UI design, language and interaction. Points to improve relate to increasing accessibility by reducing the number of nested elements, and making adjustments to the tutorial as per vocal recommendation during the testing session.

Accessibility review scores	
Part 1	4.75
Part 2	5.00
Part 3	5.00
Total	4.92/5 ★

Figure 65: *Accessibility review Overall Score*

0.7 Future Work

0.7.1 Machine Learning

0.7.2 Frontend

0.7.3 Backend

0.7.4 Deployment

0.8 Conclusion

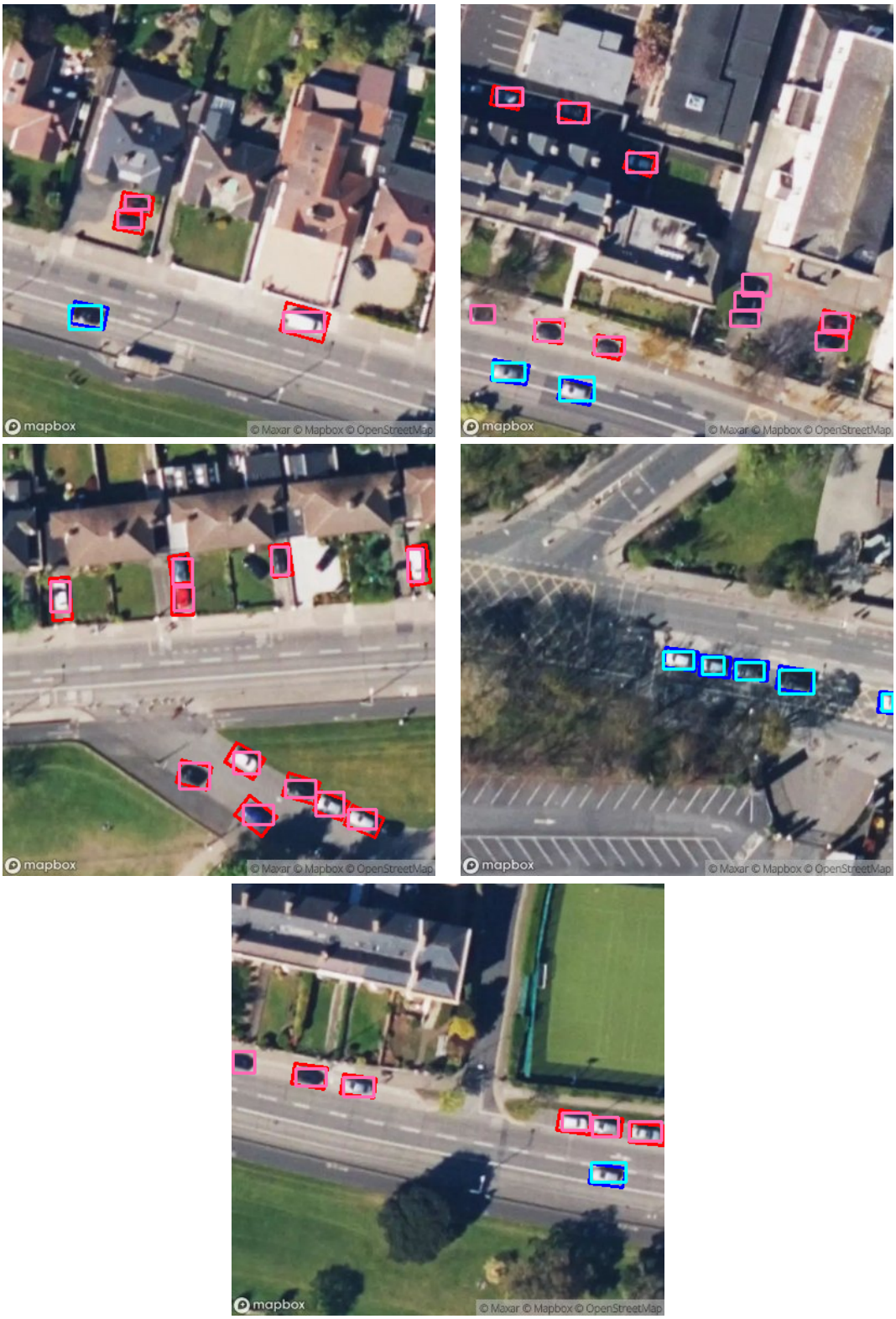

Table 2: *Images from the Road Mask Classification Test Set*

Metric	Value
Average IoU	0.59
Average Precision	0.77
Average Recall	0.70
Average F1 Score	0.69
Average Orientation Accuracy	0.64
Average Spot Detection Ratio (SDR)	1.05
Average Spot Detection Error (SDE)	1.44
Average False Positive Rate (FPR)	0.23
Average False Negative Rate (FNR)	0.22

Table 3: *Performance Metrics for Empty Parking Detection*

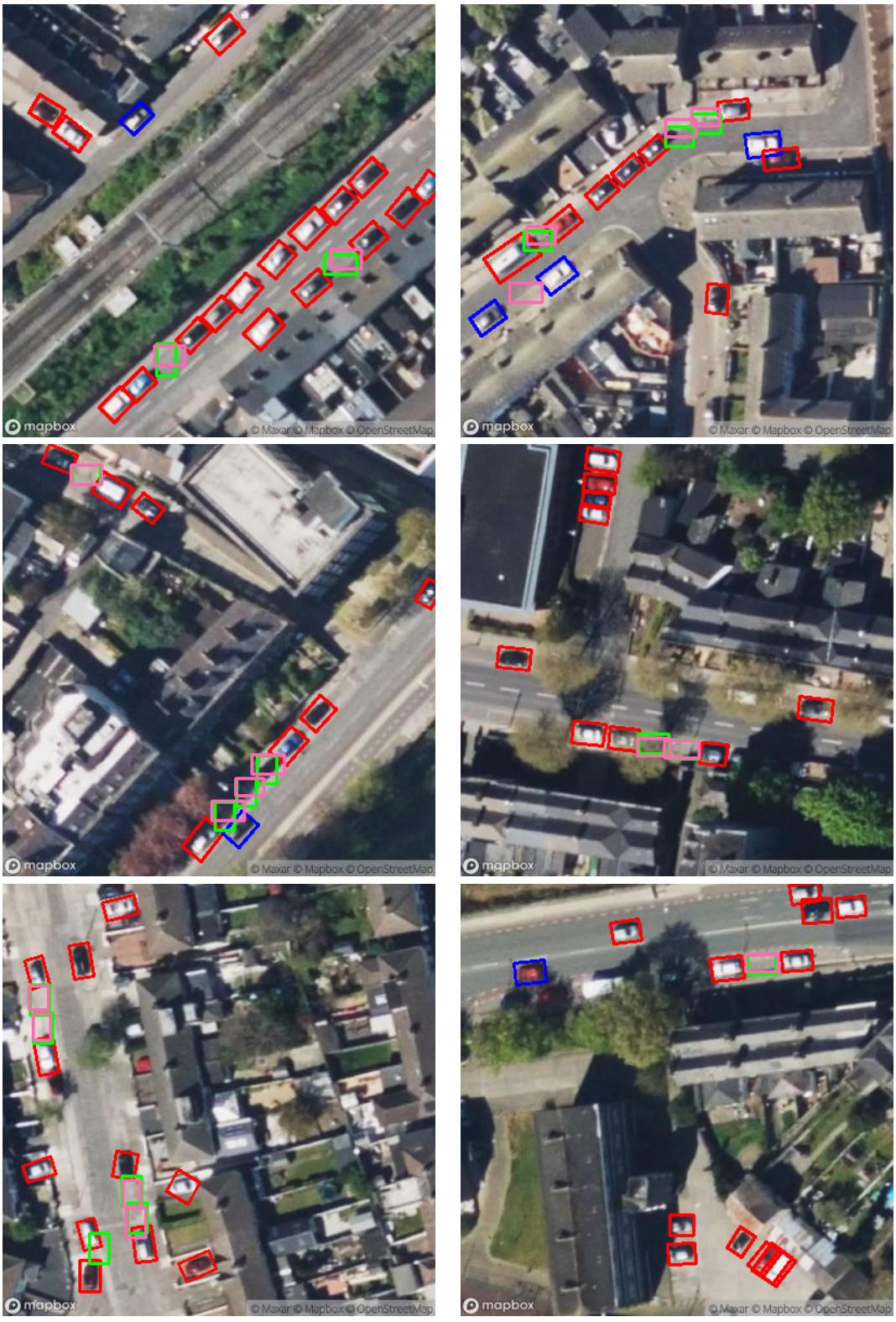

Table 4: *Images from the Empty Parking Detection Test Set*

Metric	Public	Private	Parking Lot
Average IoU	0.60	0.60	0.60
Average Balanced Accuracy	0.57	0.57	0.57
Average Precision	0.73	0.65	0.59
Average Recall	0.72	0.75	0.74
Average F1 Score	0.68	0.62	0.61
Average Accuracy	0.67	0.54	0.63
Average Specificity	0.59	0.54	0.41

Table 5: *Performance Metrics for Parking Spot Classification*

Table 6: *Images from the Parking Spot Classification Test Set*

Step	Stage	Tasks
1	Requirement Collection and Analysis	• Identify core functional requirements through user interviews and research • Analyze the strengths and weaknesses of competitors and existing solutions • Prioritize prototype design and create an iteration plan
2	Prototype Design and Development	• Create prototypes using tools like FigJam and Figma • Follow a consistent design system and guidelines • Focus on intuitive and user-friendly interaction design
3	User Testing and Feedback	• Conduct user usability testing • Gather qualitative and quantitative user feedback • Analyze test data and develop improvement plans
4	Iterative Optimization	• Adjust design solutions based on feedback • Continuously improve the user experience • Ensure the design meets project objectives

Table 7: *Process Flow Table: Design and Development Stages*

Table 8: *Usability testing Tasks - Paul*

Task	Status	Time taken	Difficulty	Errors
Load Magpie application	Complete	20s	1	N/A
Sign up	Complete	42s	1	N/A
Complete tutorial	Complete	60s	1	N/A
Place cursor on map and adjust radius to 250m	Fail	Skipped	Skipped	Skipped
Zoom in to road name level	Complete	5s	1	N/A
Place cursor on another area	Complete	5s	1	N/A
Zoom out to see full radius	Fail	Skipped	Skipped	Skipped
Filter to only view ”Parking meter” data	Pass	120s	3	Required help
Filter to toggle off all amenities	Pass	37s	3	Required help
Go through tutorial and exit at Step 3	Pass	30s	3	Couldn’t find icon
Log out	Complete	20s	2	N/A

Table 9: *Usability testing Tasks - Livia*

Task	Status	Time taken	Difficulty	Errors
Load Magpie application	Complete	5s	1	N/A
Sign up	Complete	16s	1	N/A
Complete tutorial	Complete	44s	1	N/A
Place cursor on map and adjust radius to 250m	Complete	6s	1	N/A
Zoom in to road name level	Complete	8s	1	N/A
Place cursor on another area	Fail	Skipped	Skipped	Skipped
Zoom out to see full radius	Fail	Skipped	Skipped	Skipped
Filter to only view ”Parking meter” data	Fail	Skipped	Skipped	Skipped
Filter to toggle off all amenities	Complete	18s	1	N/A
Go through tutorial and exit at Step 3	Complete	24s	2	N/A
Log out	Complete	20s	2	N/A

Table 10: *Usability testing Tasks - Ben*

Task	Status	Difficulty	Errors
Load Magpie application	Complete	1	N/A
Sign up	Complete	1	N/A
Log in	Complete	1	N/A
Complete tutorial	Complete	1	N/A
Place cursor on map	Complete	1	N/A
Zoom in and out	Complete	1	N/A
Hold map and navigate	Complete	1	N/A
Adjust radius big/small	Complete	1	N/A
Clear marker & radius	Skipped	Skipped	Skipped
Deselect all amenities	Complete	1	N/A
Select one or more amenities	Complete	1	N/A
Find tutorial and exit midway	Skipped	Skipped	Skipped
Log out	Complete	1	N/A

Table 11: *Usability testing Tasks - Jakub*

Task	Status	Difficulty	Errors
Load Magpie application	Complete	1	N/A
Sign up	Complete	1	N/A
Log in	Complete	1	N/A
Complete tutorial	Complete	1	N/A
Place cursor on map	Complete	1	N/A
Zoom in and out	Complete	1	N/A
Hold map and navigate	Complete	1	N/A
Adjust radius big/small	Complete	1	N/A
Clear marker & radius	Pass	3	Did not see button
Deselect all amenities	Complete	1	N/A
Select one or more amenities	Complete	1	N/A
Find tutorial and exit midway	Skipped	Skipped	Skipped
Log out	Complete	1	N/A

Table 12: *Usability testing Tasks - Bryan*

Task	Status	Difficulty	Errors
Load Magpie application	Complete	2	N/A
Sign up	Complete	1	N/A
Log in	Complete	1	N/A
Complete tutorial	Complete	1	N/A
Place cursor on map	Pass	3	Required help
Zoom in and out	Fail	5	Required help
Hold map and navigate	Complete	1	N/A
Adjust radius big/small	Complete	1	N/A
Clear marker & radius	Complete	2	N/A
Deselect all amenities	Complete	2	N/A
Select one or more amenities	Complete	1	N/A
Find tutorial and exit midway	Complete	2	N/A
Log out	Complete	1	N/A

Table 13: *Usability testing Tasks - Anonymous 2*

Task	Status	Difficulty	Errors
Load Magpie application	Complete	2	N/A
Sign up	Pass	2	Tried to sign up on log in page
Log in	Complete	1	N/A
Complete tutorial	Complete	2	N/A
Place cursor on map	Complete	1	N/A
Zoom in and out	Complete	1	N/A
Hold map and navigate	Complete	1	N/A
Adjust radius big/small	Complete	1	N/A
Clear marker & radius	Skipped	N/A	N/A
Deselect all amenities	Skipped	N/A	N/A
Select one or more amenities	Complete	1	N/A
Find tutorial and exit midway	Skipped	N/A	N/A
Log out	Skipped	N/A	N/A

Table 14: *Expert Review Questionnaire*

	Question	Question type
Q1	How user friendly is the log-in/sign up page?	Closed
Q2	How user-friendly is the on-boarding process	Closed
Q3	How effective is the visual hierarchy of the information on the dashboard?	Closed
Q4	Rate the clarity of the map visualization	Closed
Q5	How intuitive is the organization of amenity data categories?	Closed
Q6	Rate the following features from Worst (1) to Best (5)	Scale
Q7	How comprehensive is the amenity data coverage for Dublin city?	Closed
Q8	How valuable do you think this tool would be for the following use cases - 1: Not valuable at all, 5: Extremely valuable	Scale
Q9	Any additional comments on why this tool would useful/impractical for the above use cases?	Open
Q10	Evaluate the following technical aspects from Worst (1) to Best (5)	Scale
Q11	Rate the application's compliance with the items below from Worst (1) to Best (5)	Scale
Q12	Any additional comments regarding our application?	Open

Table 15: *Scenario for the Accessibility review*

Scenario:
You are an architect contracted by Dublin City Council to expand the Dominick Street Recreation Centre, located on Dominick Street Lower, Dublin 1. As part of your assignment, you need to plan the expansion in a way that integrates effectively with the surrounding community and existing amenities.
Scenario tasks:
1. **Locate the Community Centre:** Use the GIS application to locate the Dominick Street Recreation Centre on the map. 2. **Identify nearby amenities:** Select the public amenities you think are relevant within a 500-meter radius of the recreation centre. These can include but are not limited to: bicycle stands, parking spaces, public wi-fi spots, public toilets. 3. **Analyse amenity density:** Based on your findings, determine which types of amenities are abundant and which are lacking around the centre. 4. **Assess accesibility:** Check how accessible the recreation centre is by identifying nearby transportation options. Note any gaps in accessibility that might need addressing. 5. **Plan for additional amenities:** Suggest which new amenities should be added as part of the recreation centre's expansion to better serve the community. For example: if car parking is insufficient, recommend additional parking spaces.

Table 16: *General tasks score for the Accessibility review*

General task:	**Status**	**Difficulty**	**Errors**
Load Magpie application	Complete	1	
Sign up new account	Complete	1	
Log in	Complete	1	
Go through onboarding	Pass	2	Technical issues with onboarding overlay + user tried to click on locked elements
Place cursor on map	Complete	1	
Zoom in and out	Fail	5	Did not know how to use the mouse + onboarding confusing
Hold map and navigate	Complete	2	
Adjust radius big/small	Complete	1	
Clear marker and radius from map	N/A	N/A	Feature not used
Deselect all amenities	Complete	1	
Choose certain amenities	Complete	1	
Find onboarding and exit midway	Pass	4	Could not find onboarding button
Logout	Pass	3	Could not find profile button

0.9 Appendix A: ABC

0.10 Appendix B: XYZ