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 11, 2024

Table of Contents

List of Figures

List of Tables

List of Listings

1 An example of a sqlc configuration file with two targets with separate query inputs 9
2 An example of a SQL query with annotations used by sqlc . 11
3 An example of a Go binding generated by sqlc from the SQL query in Listing **??** 53
4 An example of an issue template used in *Magpie* 54
5 An example of a pull request template used in *Magpie* . . . 54
6 An example of a GitHub Actions workflow that will not work 54
7 An example of a GitHub Actions workflow that will work . 55
8 A github action that checks the branch name (this was edited for brevity) 55

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 **??**

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 **??** 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 **??**

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

A few images from the test set are displayed below, in Table **??** 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_spo` 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 **??**

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

A few images from the test set are shown below, in Table **??** 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

0.5 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.

0.5.1 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.

0.5.2 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.

Wireframe

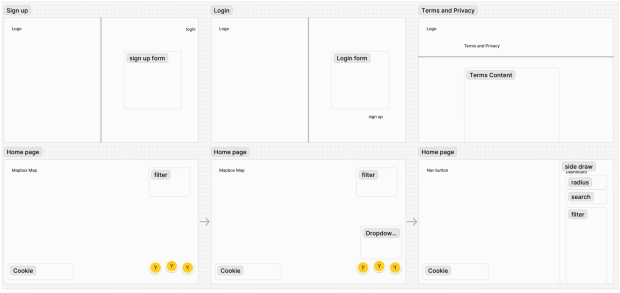

Figure 1: *Evolution of interface design from initial card-based layout to consolidated dashboard approach*

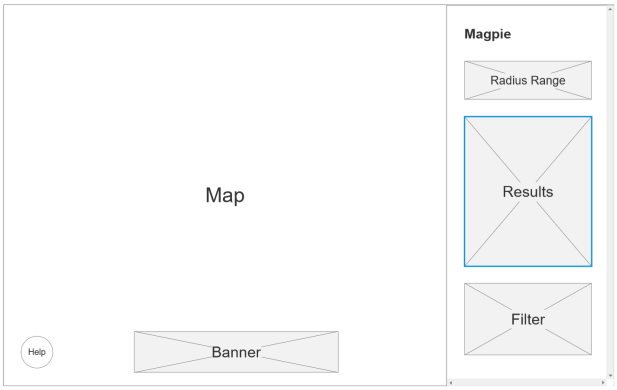

Figure 2: *Wireframe-home*

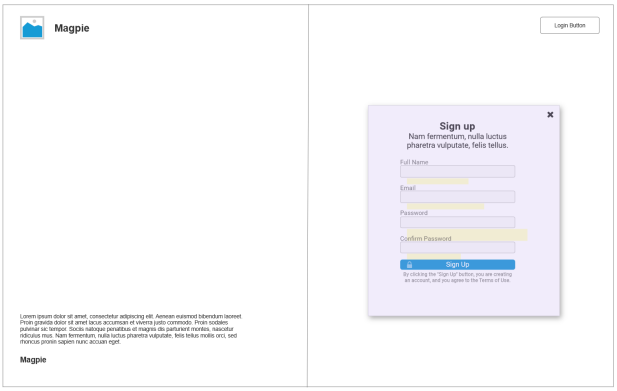

Figure 3: *Wireframe-login/signup*

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 ?? and Figure ??, which lay out the structure of the prototype.

0.5.3 Medium and High Fidelity Prototyping

Medium-fidelity prototypes offer more detail, serving as a transitional phase between initial ideas and advanced user testing.

As we move from medium to high fidelity, the prototypes become more refined and detailed, incorporating more realistic interactions and visual elements. This transition allows us to test more specific aspects of the user experience and gather more precise feedback.

High-fidelity prototypes closely resemble the final product in both form and function.

0.5.4 User Interface Iteration

Iteration 1: The first version of the prototype design is very similar to the sketch and wireframe designs, but it did not take into account the needs and experiences of the users, so it needs to be iterated. The card-based design makes the entire home screen very cluttered and scattered, and the login/signup page adopts the original shadcn style, which needs to be customized and improved.

Iteration 2: The second version of the design adopts the current version's sidebar layout, making the right side the user operation area, providing a dashboard with search radius and filter options, and adding the early search functionality.

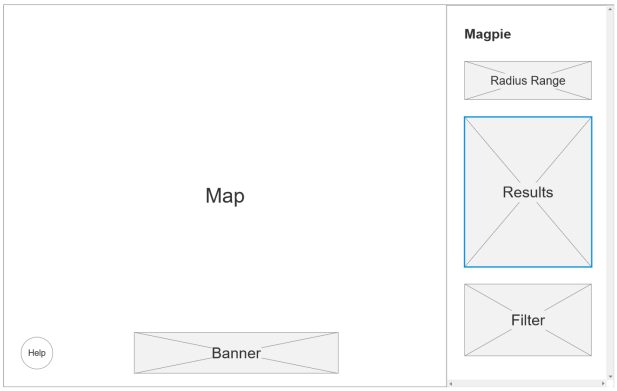

Figure 4: *Wireframe-home*

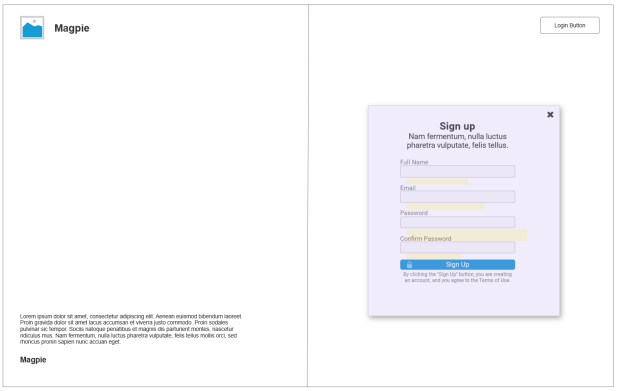

Figure 5: *Wireframe-login/signup*

Iteration 3: The third iteration of the design introduces a more detailed dashboard on the right side. It now includes a comprehensive list of amenities with the number of available spots for each, such as parking, bike stands, and public WiFi. This provides users with more specific information at a glance. Additionally, a new dropdown menu has been added at the bottom of the dashboard, allowing users to select specific amenities they are interested in. This enhances the user experience by offering more customization and easier access to desired information.

Iteration 4: The fourth iteration introduces several significant enhancements to improve user functionality and data visualization. The interface now includes quick radius selection buttons (100m, 200m, 500m, 1000m) for more precise area control, and new features like "Save to history" and "Export image" have been added to help users preserve their searches. The dashboard has been refined with eye icons for each amenity type, allowing users to toggle visibility of specific amenities on the map. Additionally, a comprehensive landing page has been implemented, explaining Magpie's core features including machine learning capabilities, interactive visualization tools, and target user groups.

0.5.5 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 ??.

bcrypt

Backend Development

As discussed in **?? ??**, 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

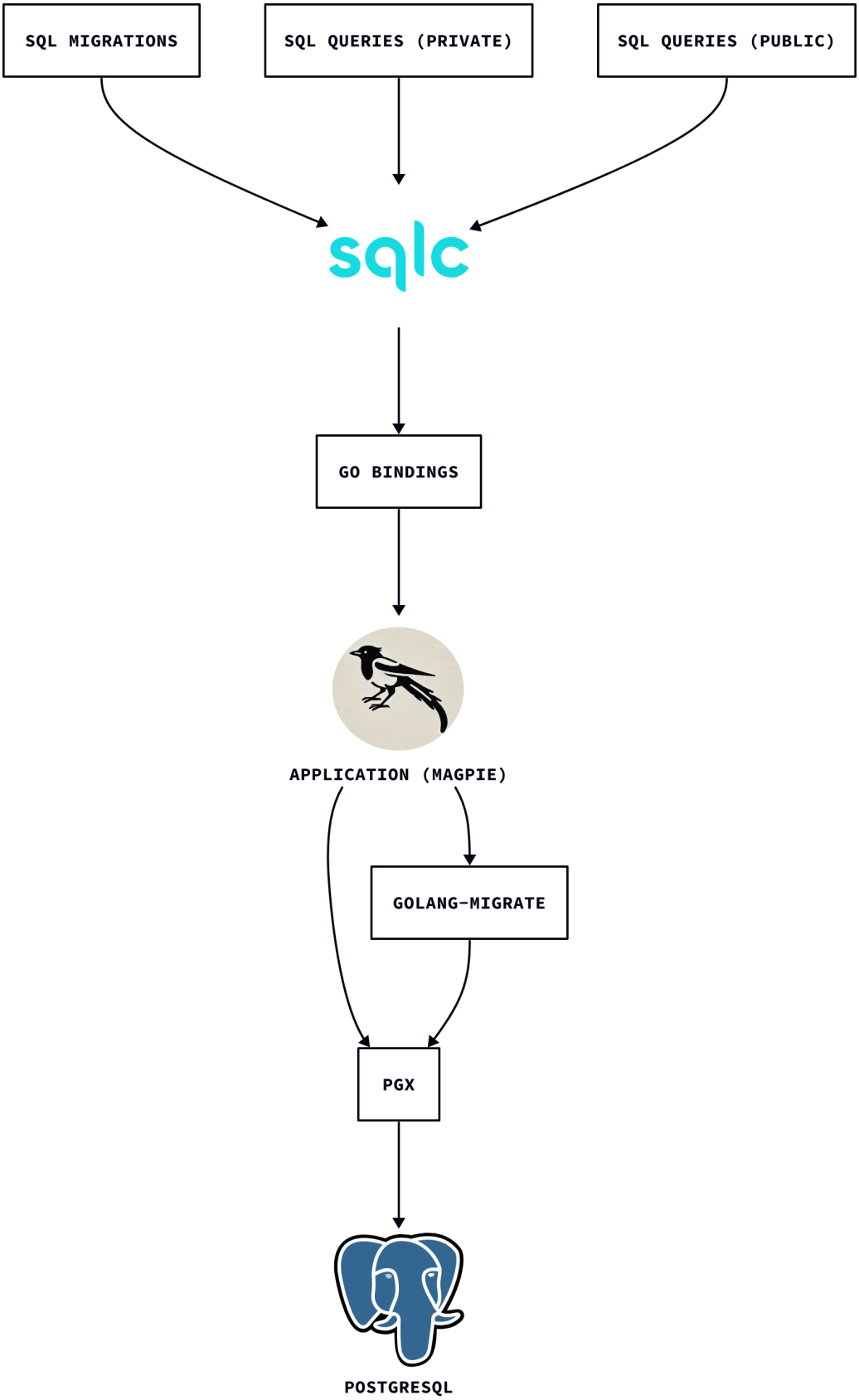

Figure 6: *Database Workflow Diagram*

```
-- 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*

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

0.5.6 Deployment

0.6 Software Management

0.6.1 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.

0.6.2 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.**

0.6.3 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:

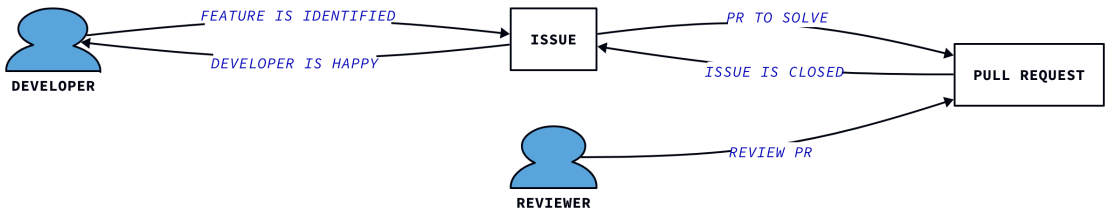

Figure 7: *The flow of work in Magpie*

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.

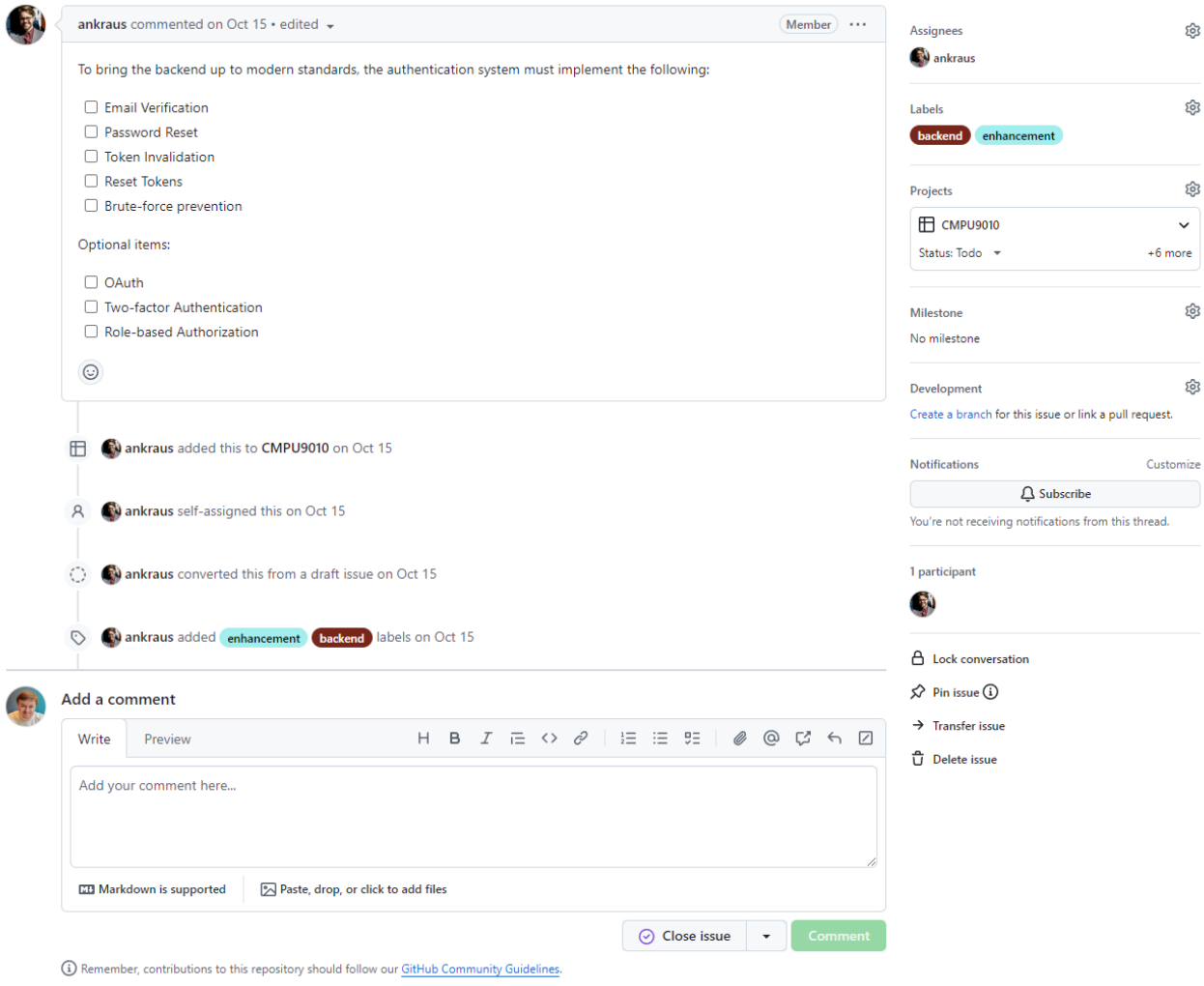

Figure 8: *An example of a GitHub Issue*

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.

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.

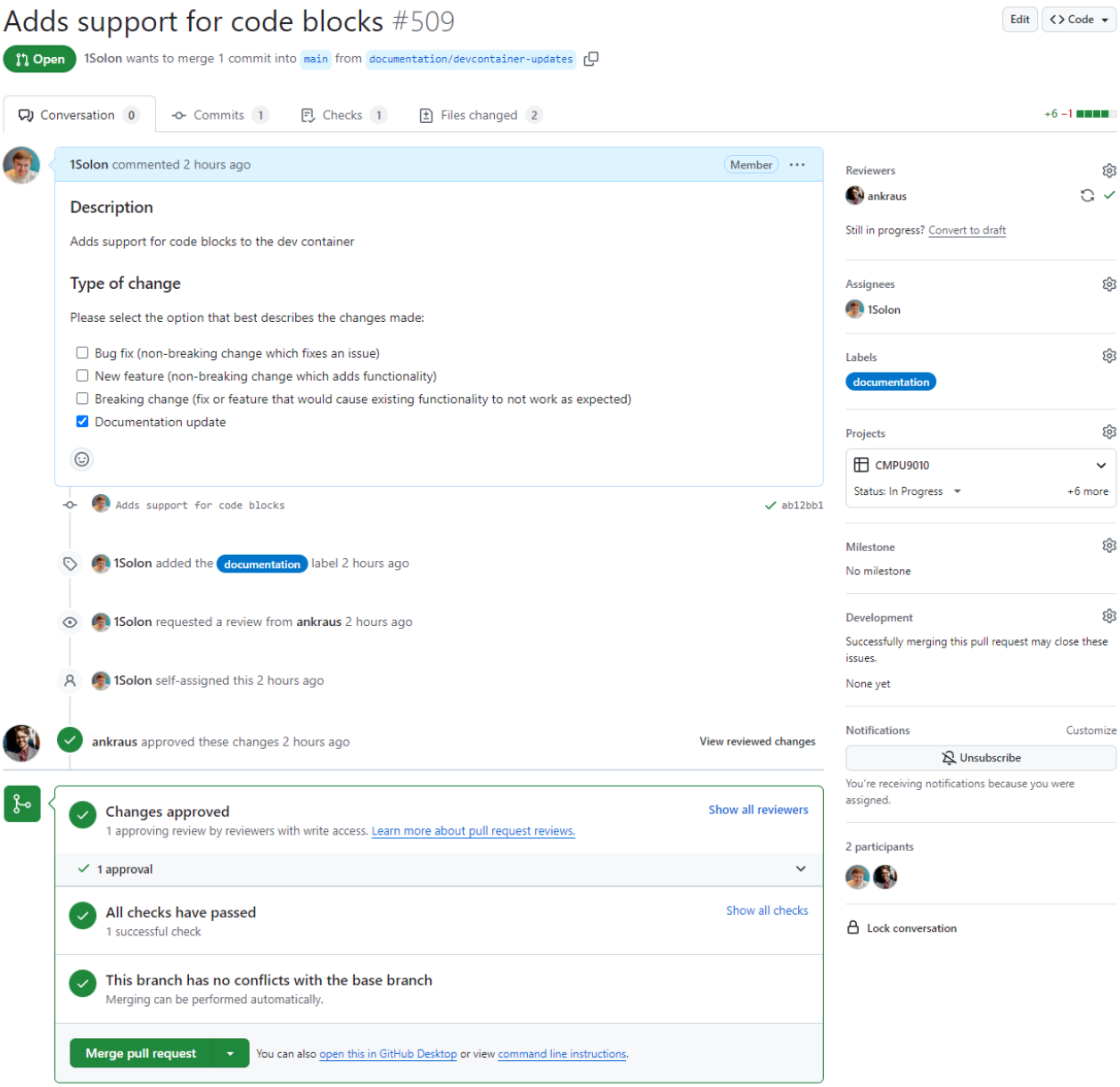

Figure 9: *An example of a GitHub Pull Request*

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.

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.

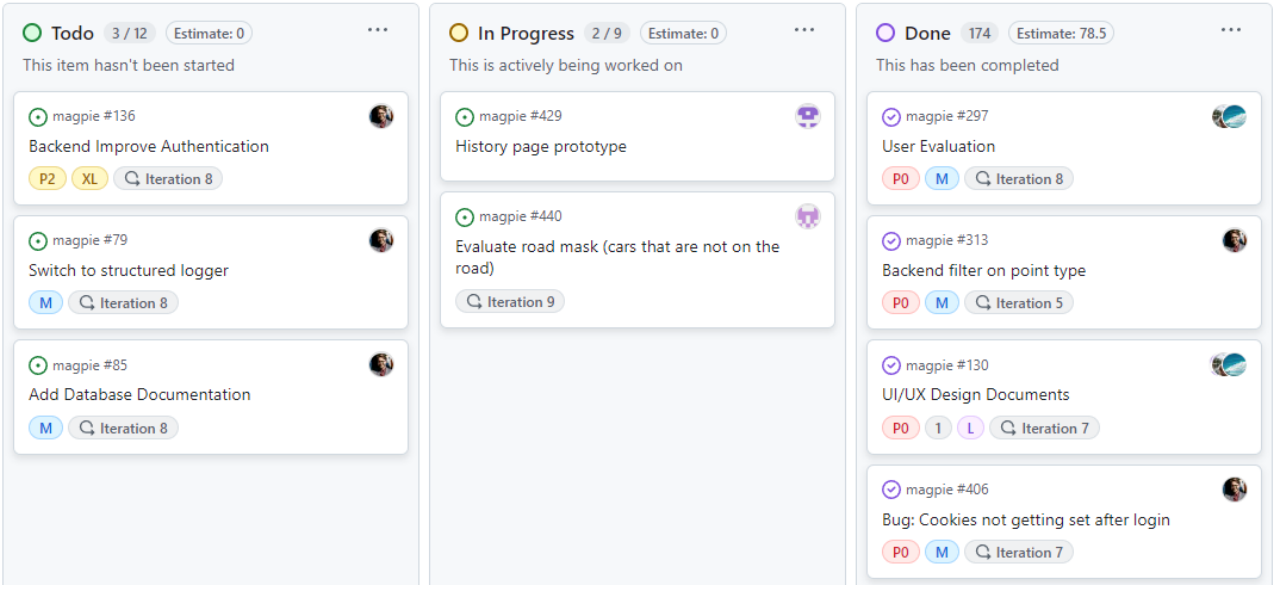

Figure 10: *An example of a GitHub Kanban Board*

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.

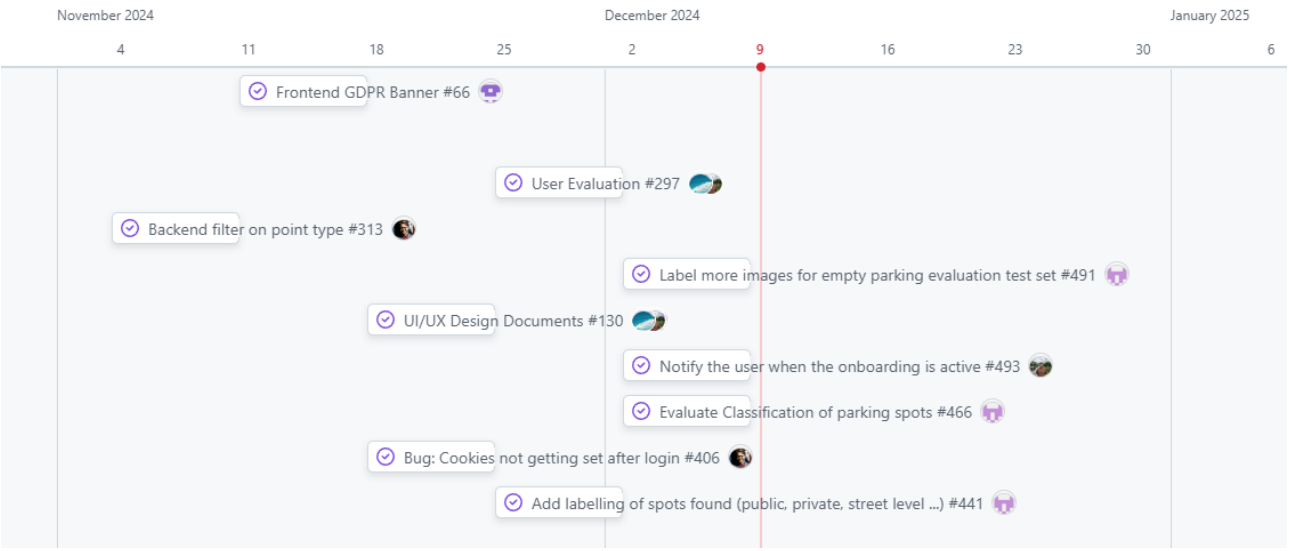

Figure 11: *An example of a GitHub Roadmap*

0.6.4 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:

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.

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.

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.

0.7 Evaluation

0.7.1 User Evaluation

Components of usability

Based on Nielsen 2012 (https://medium.com/@iizzathisharah/the-five-usability-components-by-jakob-nielsen-detailed-insights-and-examples-90695af5ffb6)

- learnability
- efficiency
- memorabiliy
- errors
- satisfaction

Usability testing

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.

The goal of the user evaluation is to gain feedback from real users, learn if Magpie works as expected and assess overall user interface. 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. 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.

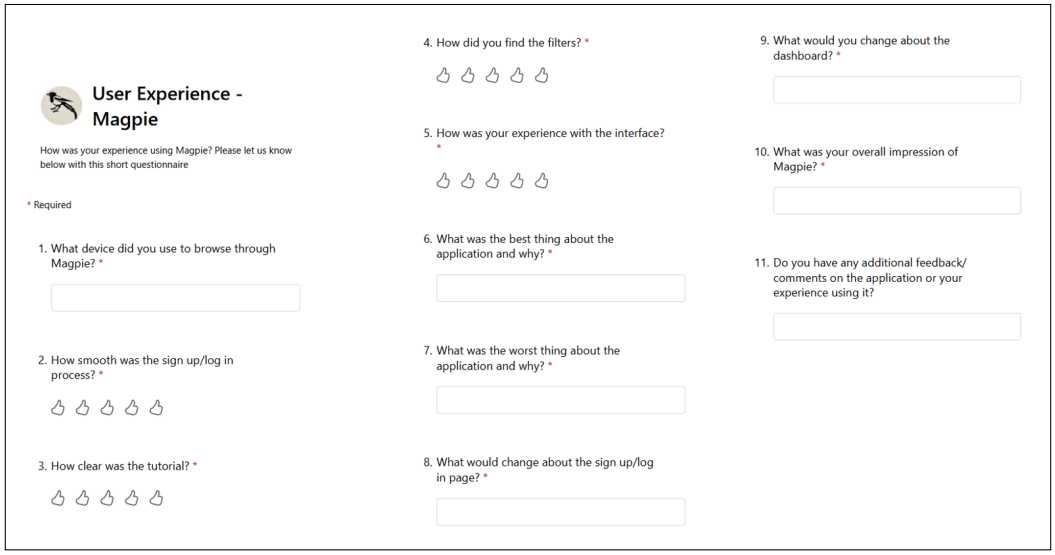

Figure 12: *User Evaluation - Satisfaction survey questions*

User 1 - Brendan

Brendan is a professor with technological background. They are considered a casual user.

This was an uncontrolled test session where Brendan 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:** why is a username required? Username should be the email. And email verification is a must!
- **Tutorial:** content of step 2 needs to be reworked, explain radius of what. Also, the positioning of certain elements is off in Firefox browser. For step 4, content wording change "dozen" for something else especially if you're planning on adding more amenity data. The step 5 should point to the map, investigate the bug.
- **Dashboard:** you should implement a button to clear the market and all the points from the map, right now the user has to reload. Perhaps think of leaving the count of toggled off amenities, it would depend on the use case. Maybe implement a quick reset button for selected amenities. Compress list of amenities to avoid scrolling (window size issue in Firefox browser)
- **Map:** you need to add plus and minus buttons to zoom in and out of the map. Can I have more information from the amenity icons by clicking on them? Perhaps a marked up image that detects this space (google street view), information on the spot (private, free, carpark). Also, make the selected icons more visually striking, a little hover maybe
- **Technical:** when clicking on map, there is a slight offset between where the marker appears and where the user clicked; implement something to avoid mis-clicking and loosing original marker position; Icons move when you toggle on and off certain amenities
- **Misc:** why are we requesting location? For the marker data (maybe we should let the user know this is where they are on the map?); need a landing page to present magpie; is the version prototype logo useful? Maybe for dev; show more information and the ml images we worked on (personal preference); reiterates we should put the ml forward (maybe on the about us)

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. Brendan was very tempted to click on the amenity icons of the map to find out more information.

Score from survey:

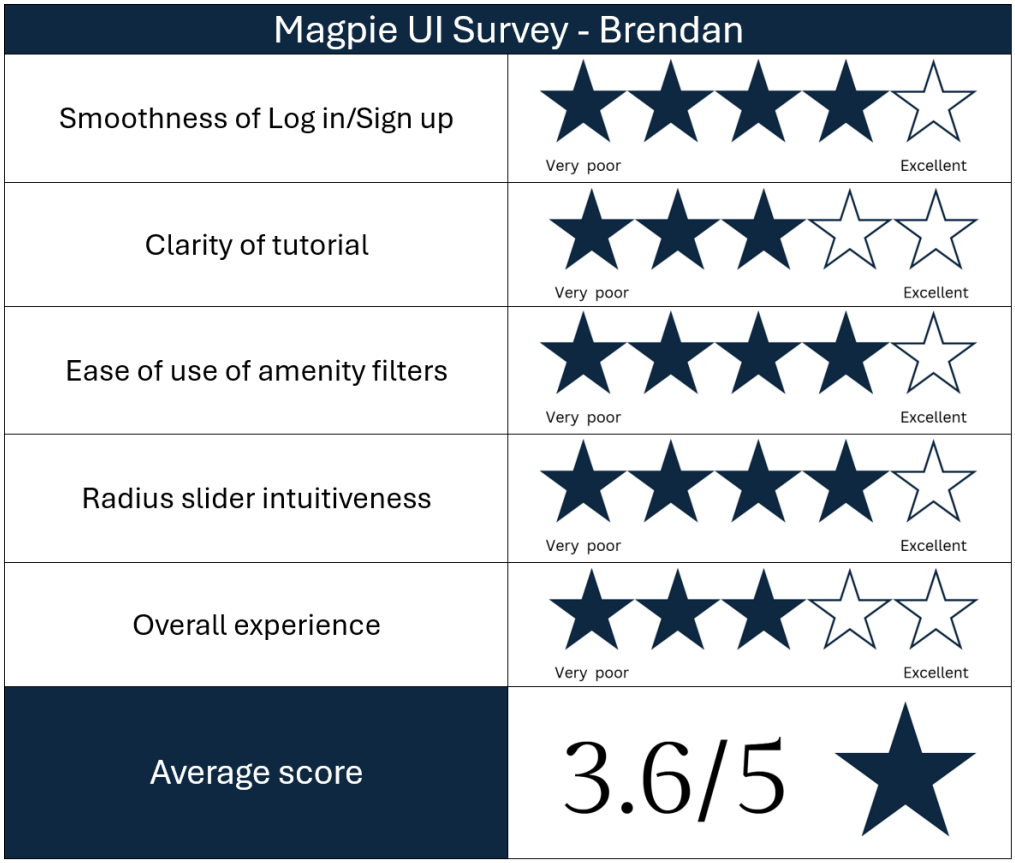

Figure 13: *User Evaluation - UI Score Brendan*

User 2 - Anonymous (Maira)

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:

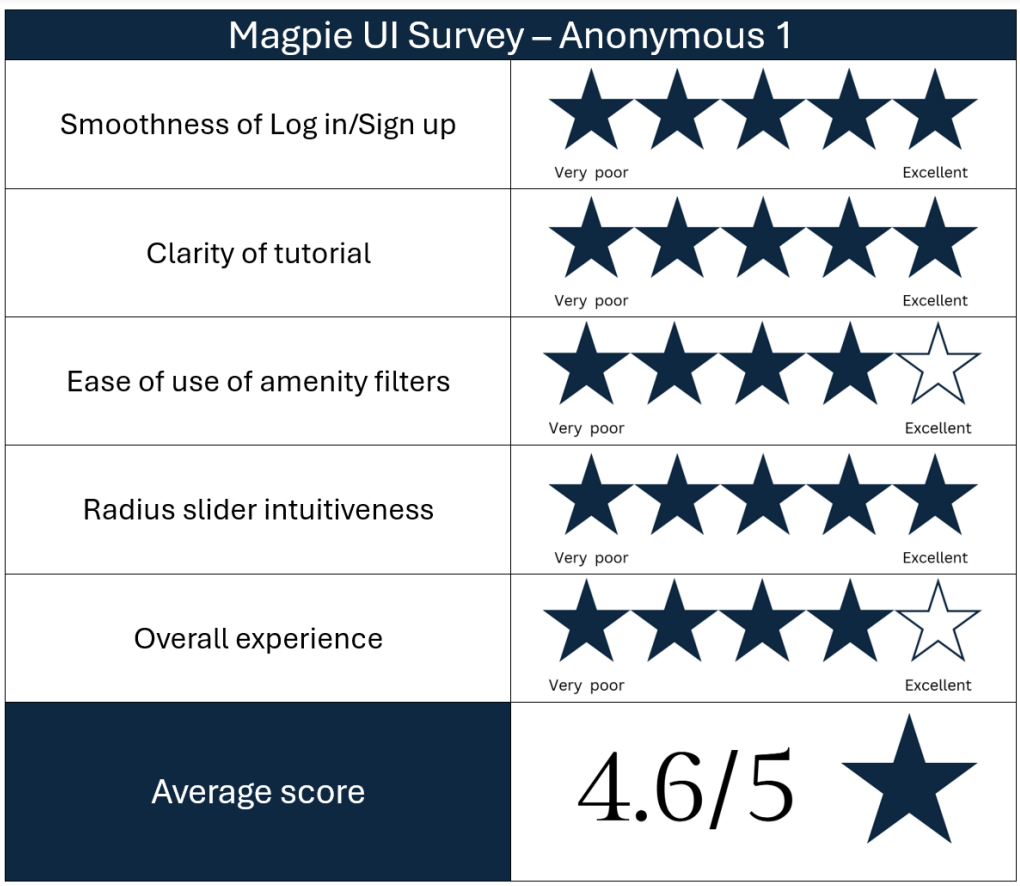

Figure 14: *User Evaluation - UI Score Anonymous 1*

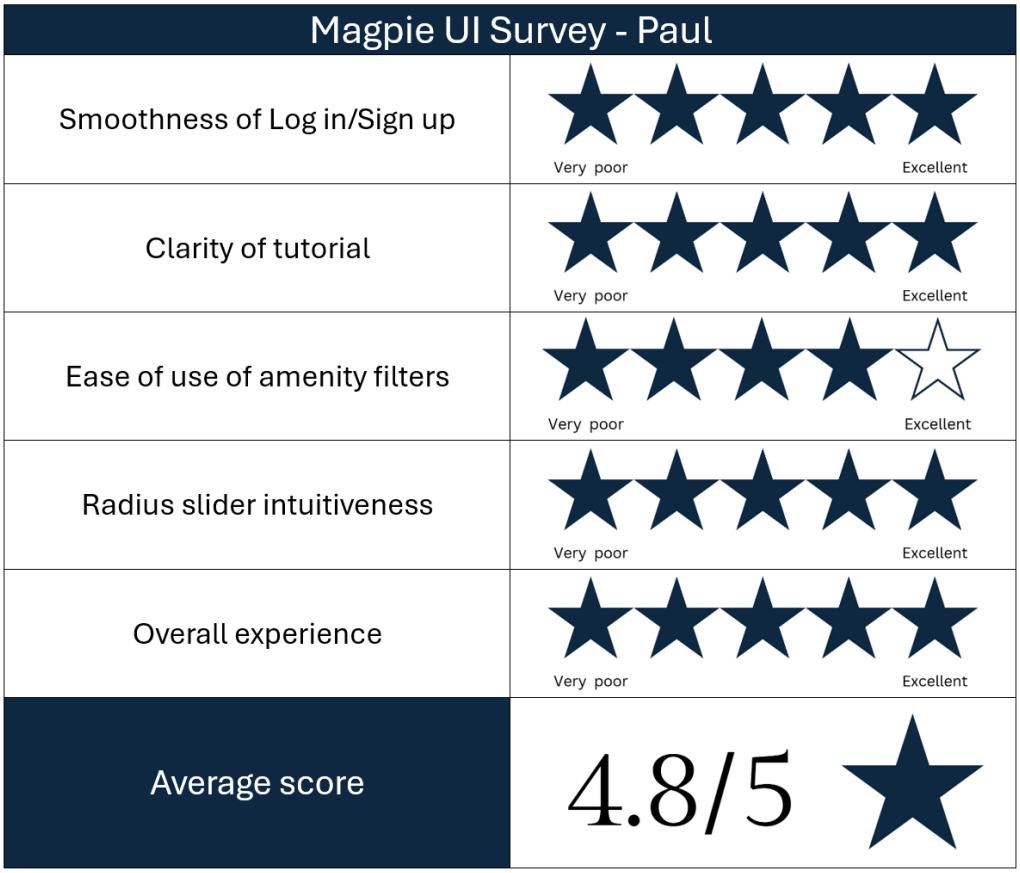

Figure 15: *User Evaluation - UI Score Paul*

User 3 - Paul

Our second controlled session was with Paul, a student in technological undergraduate degree. They are classified as a general user, not one we have identified as our target. 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 display 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 his behaviour trying to click on the icon and subsequent amenity title on the dashboard to toggle it on and off, as well as the difficulties he 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 him to find them and complete the general tasks.

Score from survey:

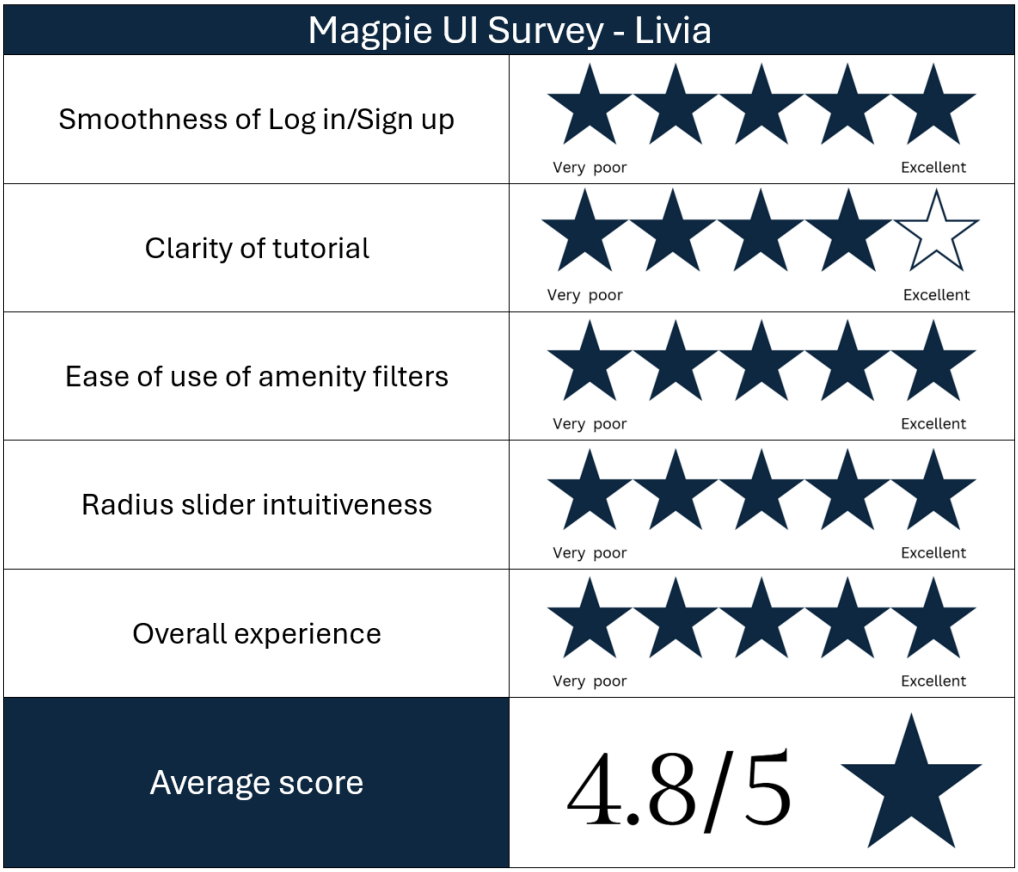

Figure 16: *User Evaluation - UI Score Livia*

User 4 - Livia

Our third controlled 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:

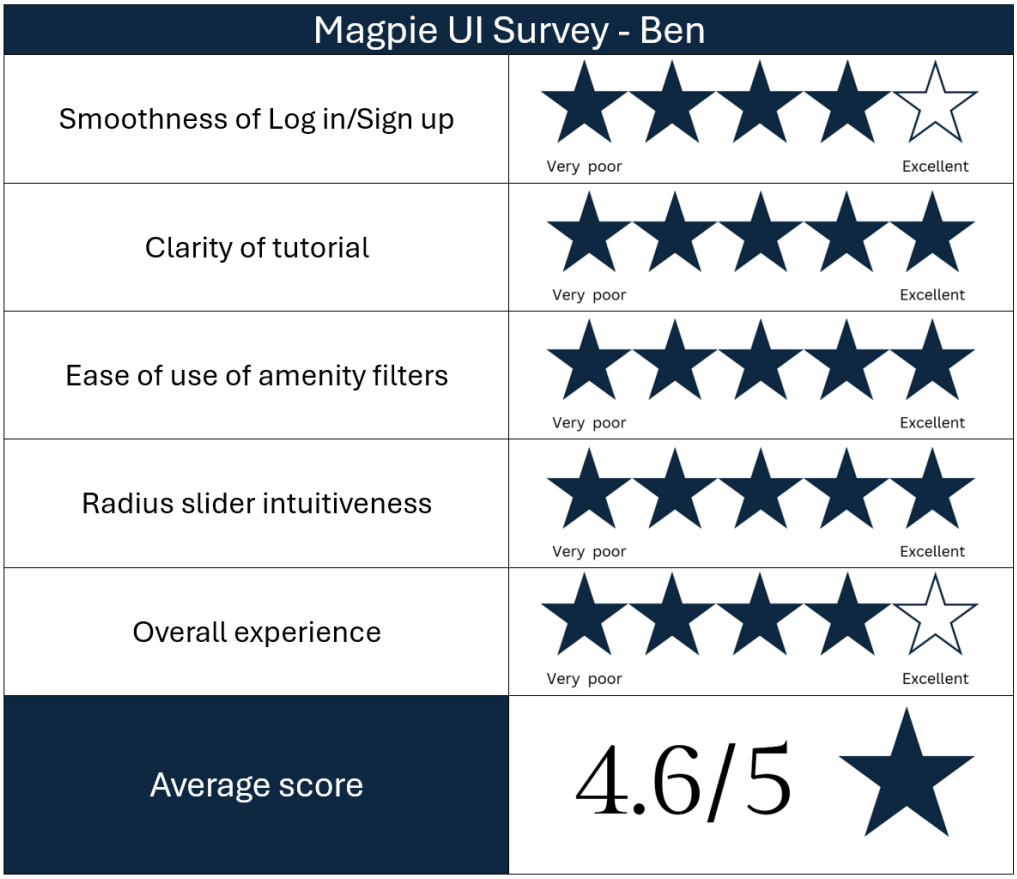

Figure 17: *User Evaluation - UI Score Ben*

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:

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 to car parking and the zoning information as well as tariff. **Score from survey:**

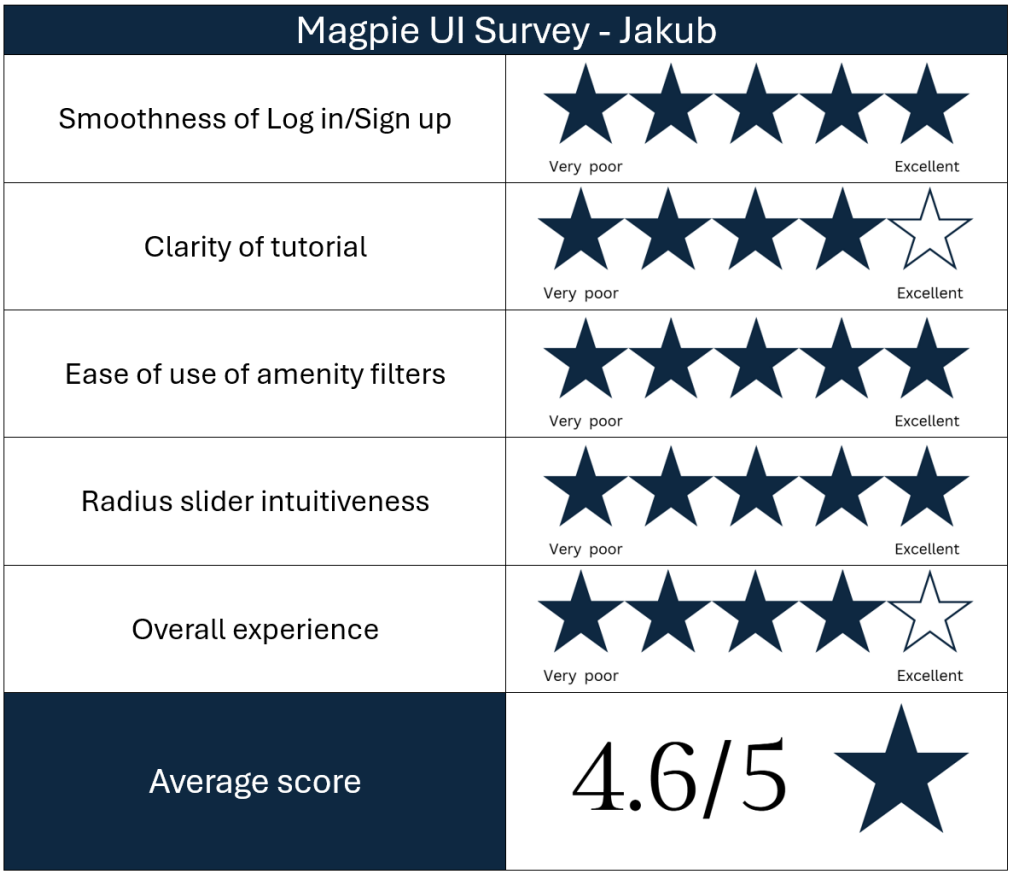

Figure 18: *User Evaluation - UI Score Jakub*

User 7 - Bryan Boyle

Professional user Bryan

User 8 - Anonymous

Professional user Sarah

User 9 - Anonymous

Professional user Odran

0.7.2 UI/UX Expert Review

We requested a review of our system from UI/UX professional Andrea Curley. INTRODUCE ANDREA BACKGROUND .
The goal of this review is to evaluate the user interface of Magpie at different stages of development and rate its user-friendliness in regards to key UI/UX general guidelines.
UX general guidelines: outline them here

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 and took the following form:

1. Presentation of Magpie
2. Free-roaming of the application by Andrea Curley
3. Discussion
4. Questionnaire & end of review

The questionnaire includes questions on visual design, information architecture, data quality & integration, technical performance, compliance and overall assessment of Magpie as shown in the table below. Different types of questions were included, such as "Closed" "Scale" and "Open" to allow for both the quantitative and qualitative measure of Magpie. SUPPORTING EVIDENCE ON MIXED SURVEY QUESTIONS

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, Professor 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, Professor 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, Professor Curley 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 shoud 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 figure 3.3) 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.
Currently, Professor 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.

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 the icons.

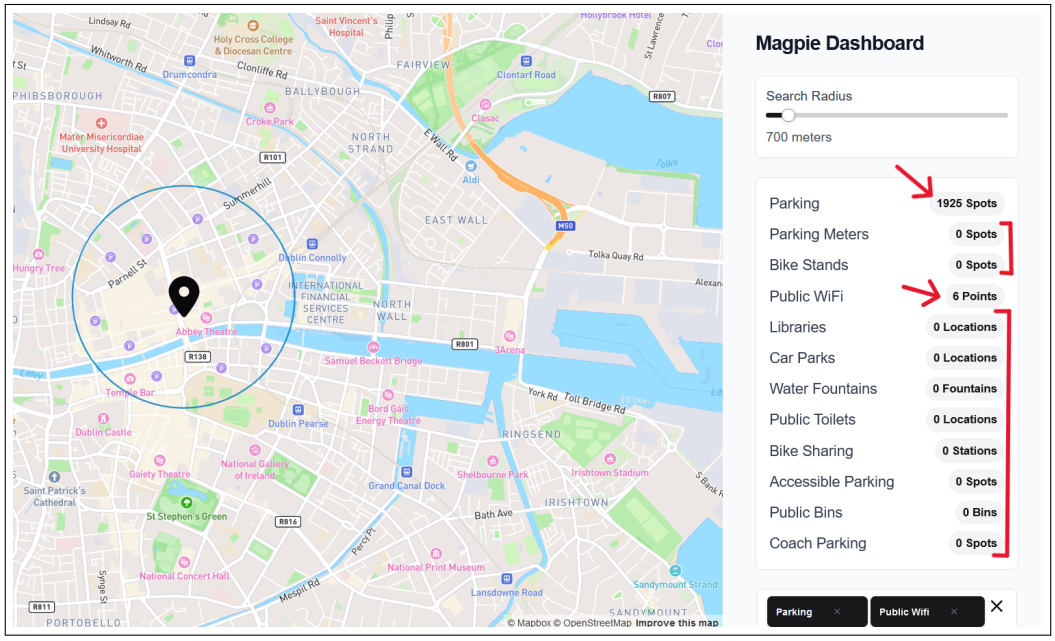

Figure 19: *V.0.1 Magpie Dashboard & Map when 2 amenities are selected*

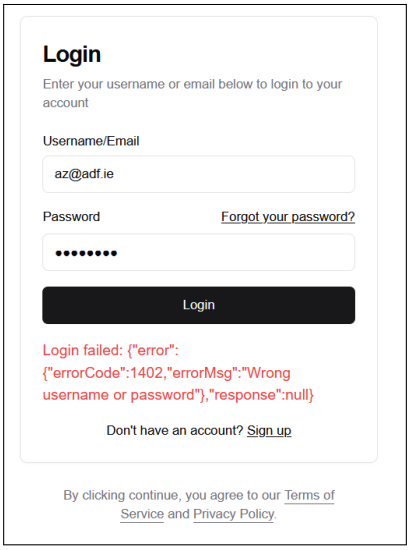

Figure 20: *Error message when using inexistant username and password*

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. Professor Curley questioned the benefits of logging 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.

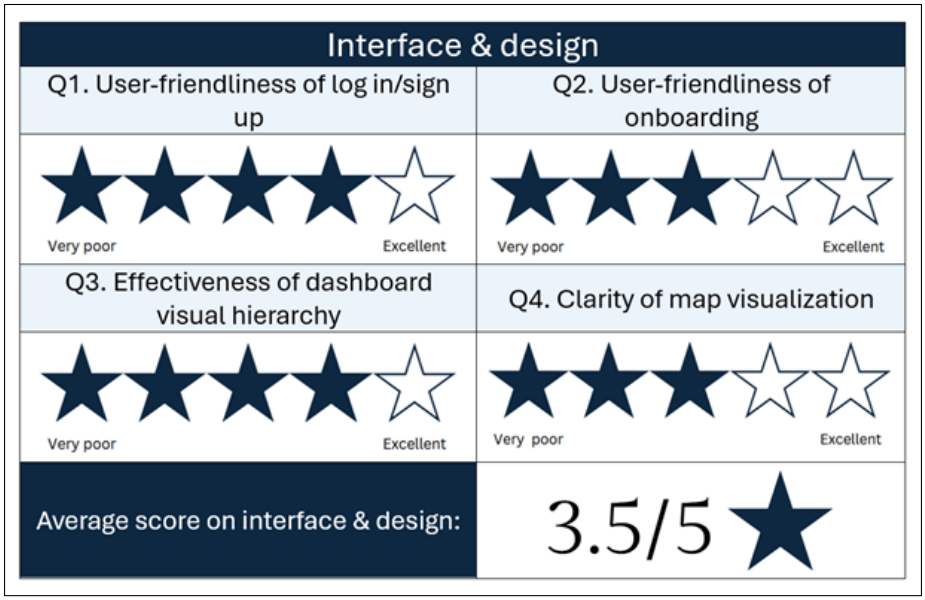

Figure 21: *UX Expert survey response on Magpie UI*

Survey response: Below are the answers to the expert review survey. The response to the survey help complement the oral feedback received during the expert review and provide some quantitative data as a baseline for the next evaluation. The two open-ended questions, to which she asked us to elaborate further, will help us review future open-ended questions and ensure they give enough information for the user to answer.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.

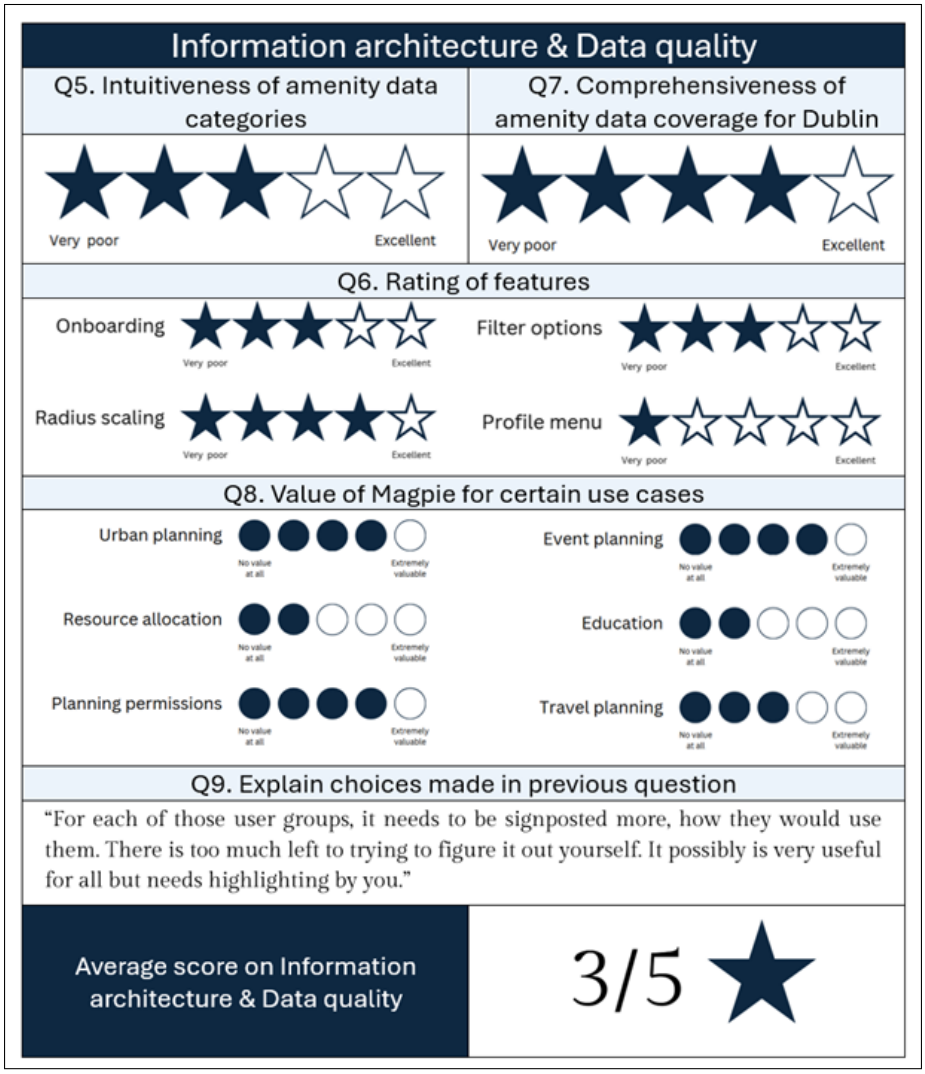

Figure 22: *UX Expert survey response on Magpie Information Architecture*

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.

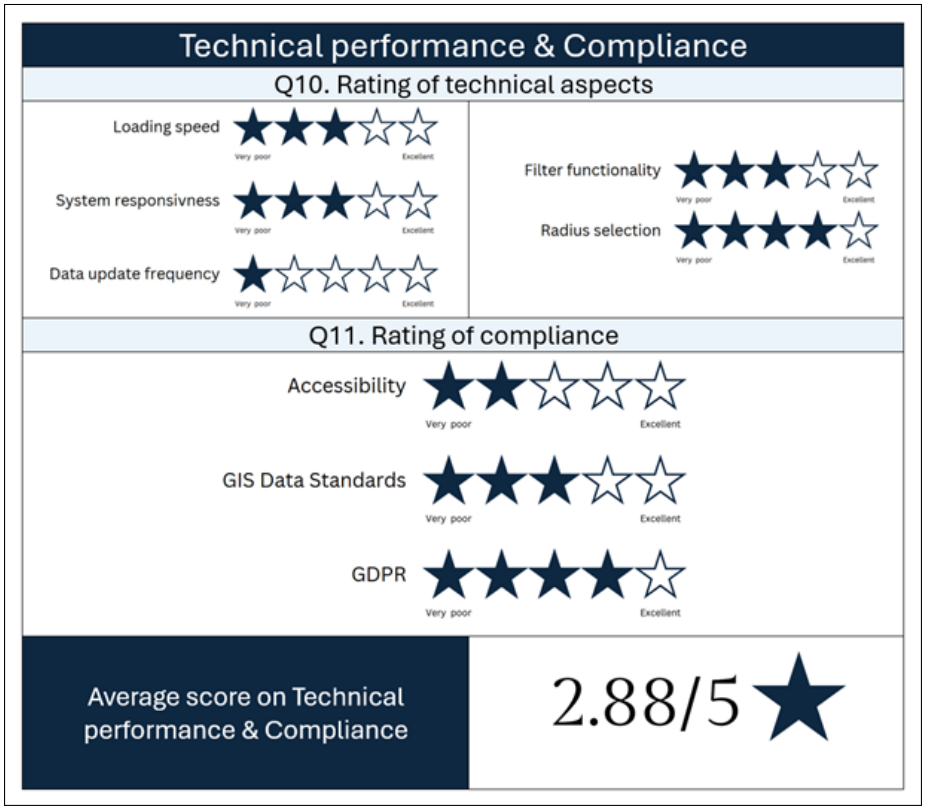

Figure 23: *UX Expert survey response on Magpie Technical Performance*

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

Figure 24: *UX Expert score of Magpie*

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

Summary: To conclude the first session, Professor 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.

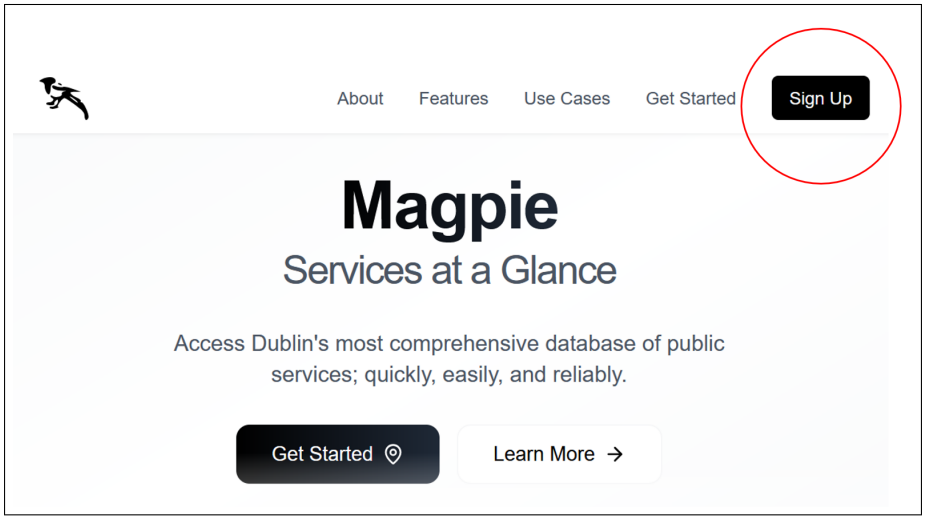

Figure 25: *Magpie Landing page - Sign up button*

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.

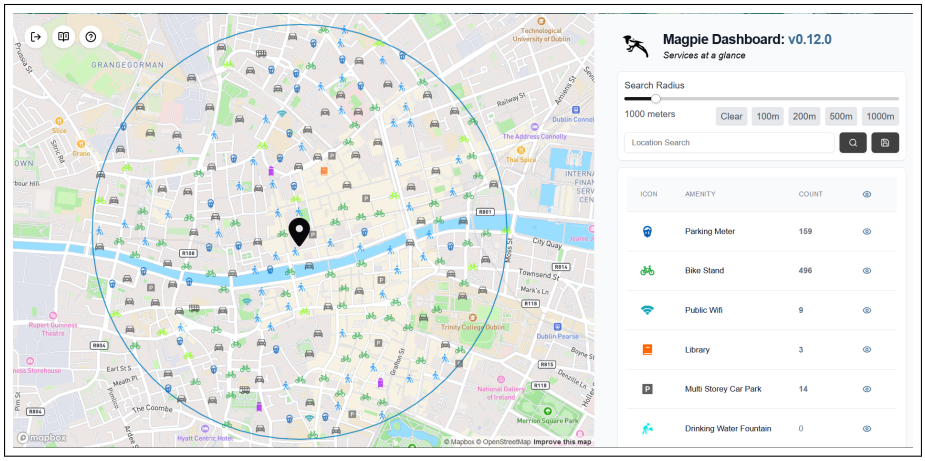

Figure 26: *Magpie New Dashboard*

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 27: *Magpie Map - Amenity Icons grouping relative to zoom level*

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.

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.
The search feature will be further addressed in the future work section of the report on how to switch it to a semantic.
Last suggestion on this point is to be able to press enter to start the search, instead of being confined to the search button.

Survey responses:

Overall: The feedback from the previous review has been onboarded very well, the application looks clean and is easy to use. 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.

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 28: *WCAG 2.1 Scope, Guidelines & Exemptions*

0.7.3 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**).

There is an inherent limitation of accessibility for map-based systems, especially for visually impaired users. 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 web-map applications for visually impaired individuals has presented several recommendations, the following which we will guide Magpie's accessibility evaluation:

- **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.

CREATE NEW SURVEY TO RATE ACCESSIBILITY MAGPIE BASED ON ABOVE

We requested a review of our system 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 Aging Well 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 and universal design of the Magpie and evaluate it with regards to the accessibility 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 presentend 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 lense of an accesibility expert.
The questionnaire is the same as the one given to the users who participated in the usability testing. In hindsight, a tailored questionnaire should've been created for Damian Gordon, to specifically evaluate key accessibility items in Magpie.

Throughout the session, we were 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.

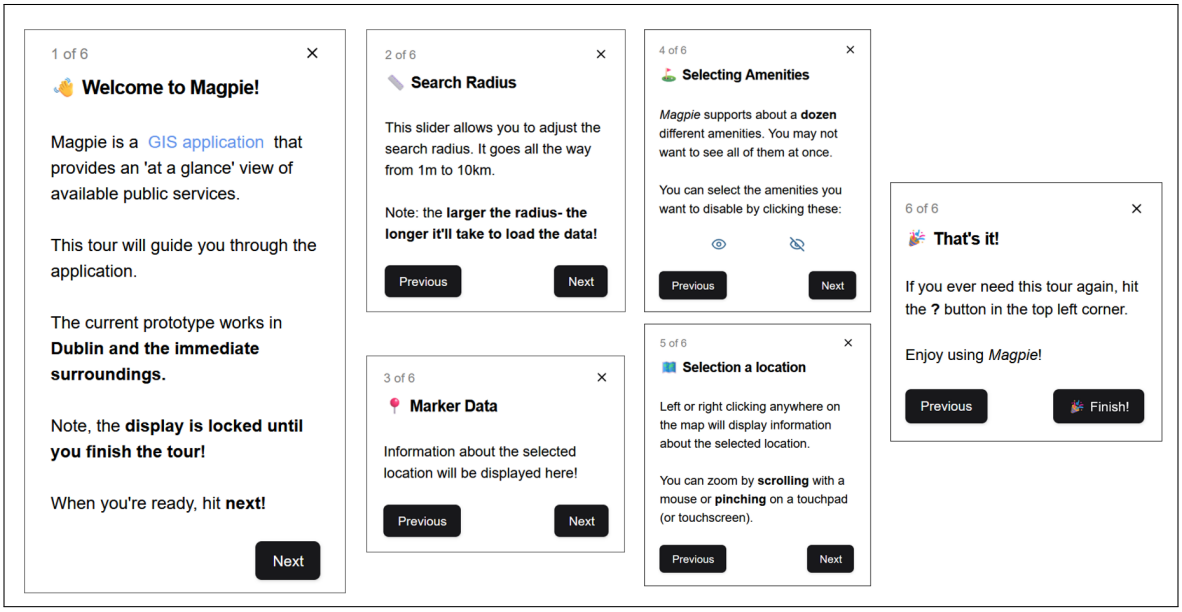

Figure 29: *V.0.1 Magpie Onboarding Steps*

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.

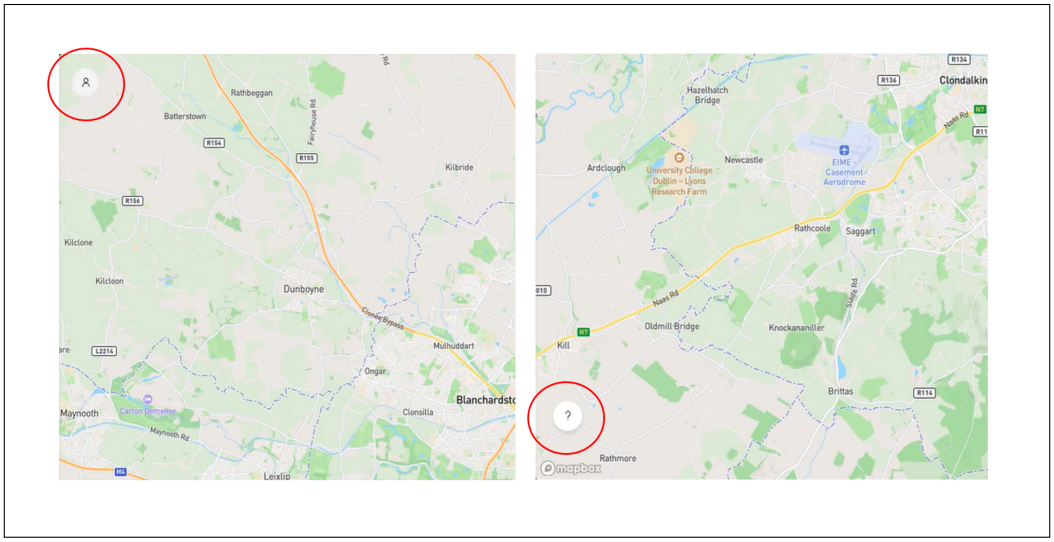

Figure 30: *V.0.1 Magpie Profile & Onboarding icons*

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.

Lastly, his survey responses BLA BLA BLA TEXT TEXT BLA...

0.8 Future Work

0.8.1 Machine Learning

0.8.2 Frontend

0.8.3 Backend

0.8.4 Deployment

0.9 Conclusion

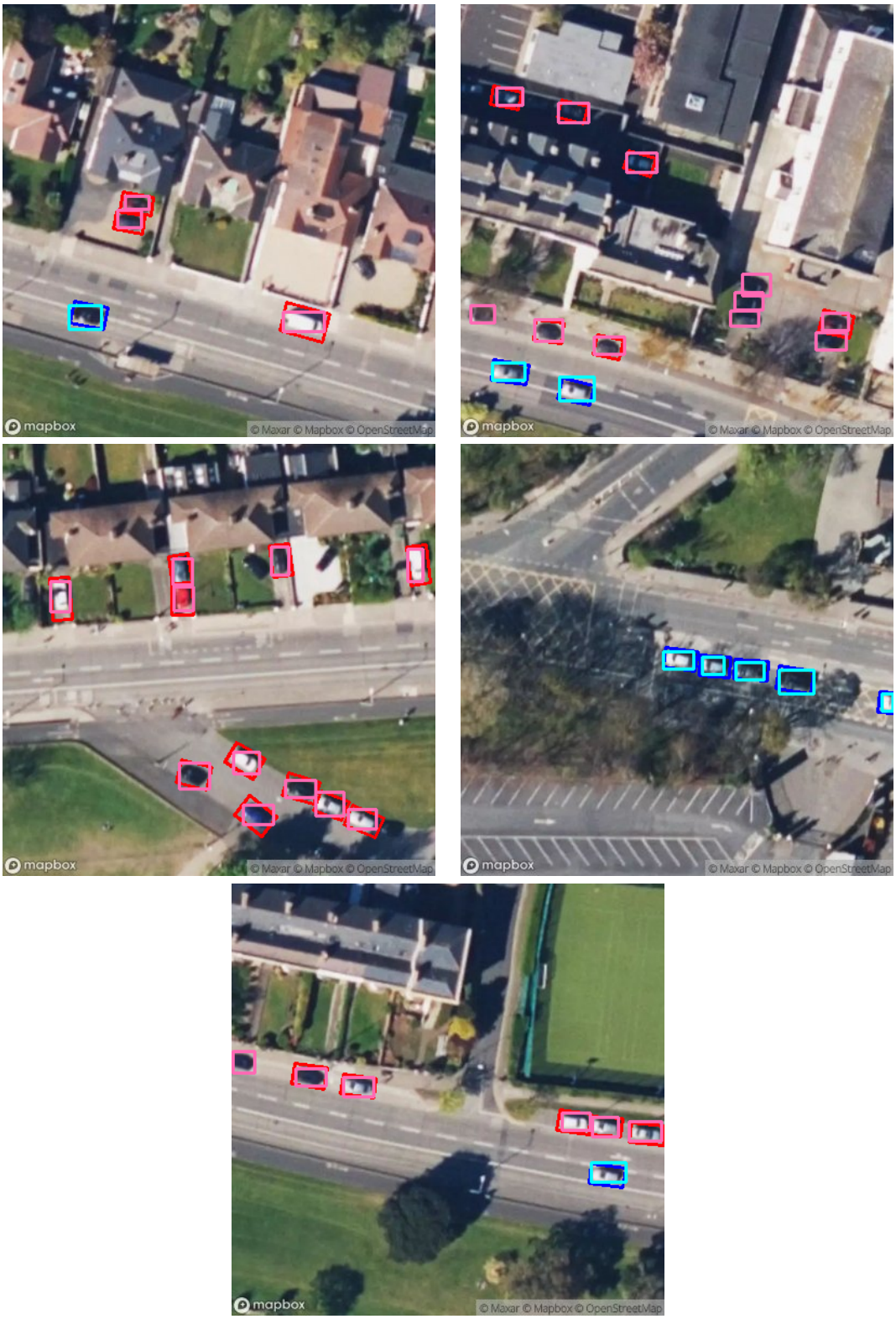

Table 2: *Images from the Road Mask Classification Test Set*

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*

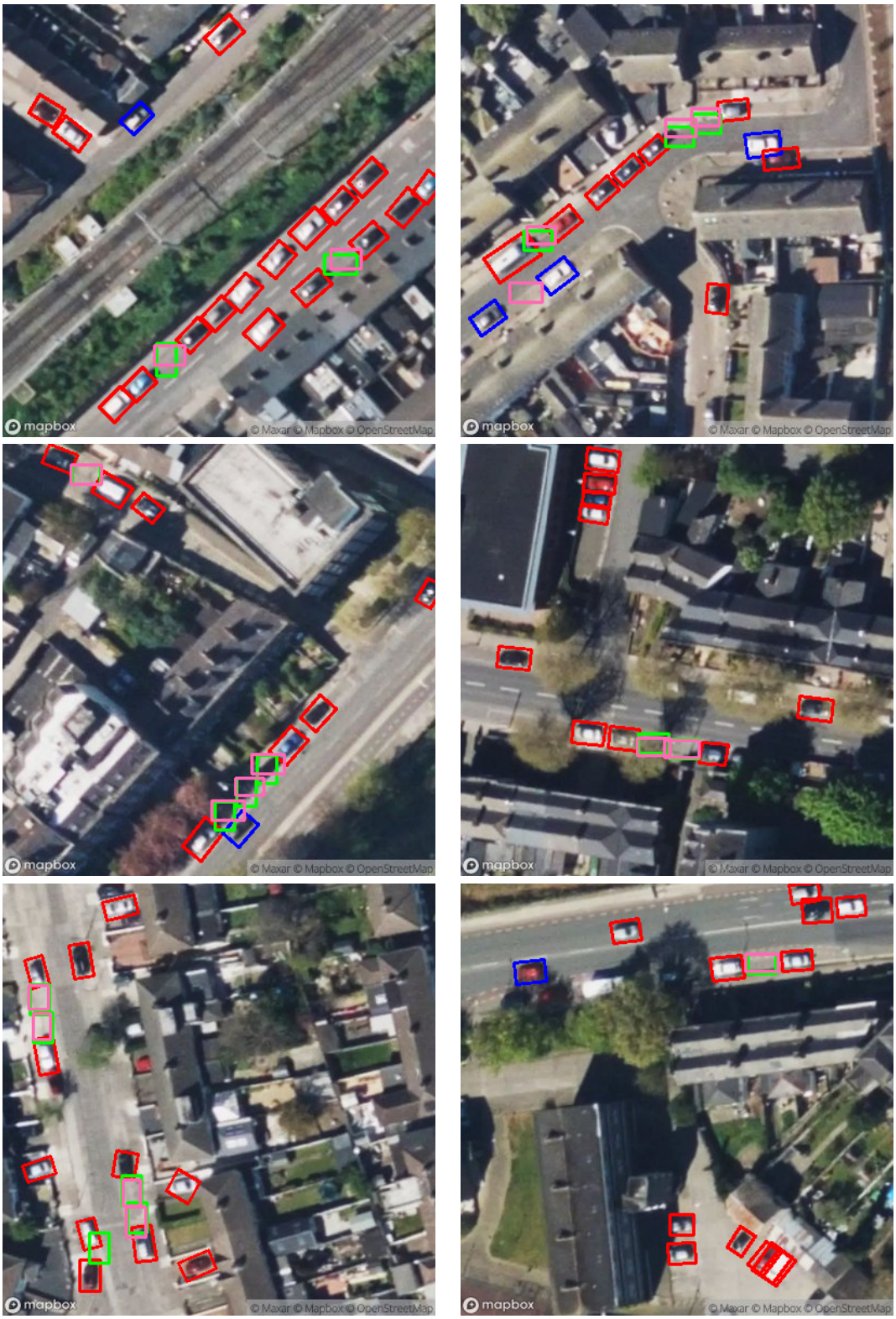

Table 4: *Images from the Empty Parking Detection Test Set*

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*

```
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* **??**

```
---
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*

```
### 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*

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

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

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

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

```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)*

Table 7: *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 8: *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 9: *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 10: *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 11: *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 12: *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 13: *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 explanation 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

0.10 Appendix A: ABC

0.11 Appendix B: XYZ