



A Study and development on the impact of a collaborative restaurant application

by

Ryan Michael Hopkins

Supervisor: Divya Chidambaram

Project submitted as part of the requirements for the
award of

BSc (Hons) Computing (Software Engineering)

May 2026

DECLARATION

I, Ryan Michael Hopkins declare that I am the sole author of this Project; that all references cited have been consulted; that I have conducted all work of which this is a record, and that the finished work lies within the prescribed word limits.

This work has not previously been accepted as part of any other degree submission.

Date : 26/05/2026

FORM OF CONSENT

I Ryan Michael Hopkins hereby consent that my Project, submitted in candidature for the B.Sc.(Hons) Computing (Software Engineering) degree, if successful, may be made available for inter-library loan or photocopying (subject to the law of copyright), and that the title and abstract may be made available to outside organisations.

Date : 26/05/2026

Table of Contents

List of Abbreviations	5
Glossary of Key Terms.....	5
1. Introduction.....	7
1.2 Overview	7
1.2 Motivation	7
1.3 Research Claim.....	7
1.4 Aim.....	7
1.5 objectives.....	8
2. Literature review	8
2.1 Existing restaurant discovery apps.....	8
2.1.1 Yelp	8
2.1.2 TripAdvisor ns	9
2.1.3 Google Maps ns	9
2.1.4 Comparative summary ns	10
2.2 Location based Services on mobile applications	10
2.3 User experience in Food applications.....	11
2.4 Review and rating system.....	12
2.5 Gap in the market	13
3. Methodology	13
3.1 Chosen methodology – Waterfall	13
3.2 Research Methodology.....	14
4. Project plan.....	14
4.1 Project Timeline.....	14
4.2 Milestones	15
4.3 Resources	17
4.4 Risk Assessment	18
4.5 Plan evolution ns	19
5. Design and implementation	21
5.1 requirements	21
5.2 Architecture	22
5.3 UI/UX design	23
5.4 Google place api integration	24
5.5 Key features included	25

5.6 Challenges encountered and resolved	26
5.7 Application flow	27
6. Testing and feedback	30
6.1 Testing Strategy	30
6.2 Test cases	30
6.3 Usability testing	32
6.4 Participant demographics	32
6.5 Feedback Analysis.....	33
7. Conclusion.....	33
7.1 Critical Analysis of the final product	34
7.2 Evaluation against existing products	34
7.3 Strengths of the project	35
7.4 Limitations and weaknesses	35
7.5 Methodology evaluation	36
7.6 Ethical Considerations	36
7.7 Future work and recommendations	37
7.8 Summary and conclusion	37
8. Bibliography	38

List of Abbreviations

Technical:

- API – Application Programming Interface
- GPS – Global Positioning System
- iOS – iPhone Operating System
- IDE – Integrated Development Environment
- UI – User Interface
- UX – User Experience
- SDK – Software Development Kit
- SwiftUI – Swift User Interface
- GPS – Global Positioning System

Methodology and research:

- SUS – System Usability Scale
- GDPR – General Data Protection Regulation
- ToS – Terms of Service
- MVP – Minimum Viable Product
- LMNS – locative mobile social networking

Glossary of Key Terms

- Apple MapKit – Apple’s native iOS framework for displaying interactive maps within an application. MapKit was used as a placeholder in the development of the rebuild, this was later replaced by Google Maps SDK.
- Churn – The rate at which users stop using an application after initial adoption, reducing churn is a key measure of user retention, this is directly addressed in the research portion of the report.
- Collaborative feature – A software function that enables multiple users to interact with shared content simultaneously. In this project, the collaborative features commonly discussed include the group voting system and shared lobby sessions.
- CoreLocation – Apple’s iOS framework for accessing the device location. CoreLocation was used originally to retrieve the user’s current latitude and longitude coordinated, which were passed to the Google Places API to return nearby restaurant results. This is later replaced by a postcode search in the location selector screen.
- Firebase (Firestore) – Googles cloud hosted real time database platform, Firebase Firestore was used in this project to store lobby details, including group voting data and session state.
- GDPR (General Data Protection Regulation) – UK and EU legislation governing the collection, storage and use of personal data. The application was designed to comply with GDPR from the beginning, collecting only the users Nickname, and storing the lobby session details without any additional personal data

- Google Place API – Google API that returns data about locations including restaurants, shops and landmarks. The API was used to perform nearby restaurant searches and retrieve details such as name, address, opening hours, photos, reviews and ratings.
- place_id – The unique identifier assigned to each location by the Google Places API. In this project, this identifier is stored in firebase to identify a restaurant with other details such as ratings, name and location always ensuring compliance with the Places ToS.
- GPS (Global Positioning System) – A navigation system used to determine the geographic location of a device. In this project, GPS coordinates were originally used to search for nearby restaurants, this is superseded by the location selector screen.
- iOS – Apple’s operating system for iPhone and iPad devices. The application was developed natively for iOS using Swift and SwiftUI, targeting iOS 15 and above
- Retention – In this context retention refers to the proportion of the users who continue to use an application after their first adoption. This project aims to investigate whether collaborative social features increase user retention in restaurant discovery.
- SUS (System Usability Scale) – A standardised questionnaire, that produces a usability score out of 100. A score above 68 is considered above average.
- Swift – Apple’s programming language for developing applications on iOS and other platforms. Swift replaced Expo and React Native as the development language.
- SwiftUI – Apple’s UI framework for building native iOS interfaces. SwiftUI was used to construct all screens in the application.
- Terms of service – The legal agreement between a service and its user with instructions on how that service may be used. The Google Places API terms of service prohibit long-term storage of data.
- Xcode – Apple’s integrated development environment (IDE) used for Swift and SwiftUI development. Xcode includes the iOS simulator and debugging tools, this software can only be run on a MacBook.
- Waterfall – A sequential methodology for software development which each phase must be complete before the next begins. The waterfall model was adopted for this project due to its suitability.

1. Introduction

1.2 Overview

the project presents the design, development and evaluation of a collaborative discovery application for iOS, built natively in Swift and SwiftUI. The application, titled Food Match, enables groups to jointly browse and vote on restaurants, through a shared lobby session powered by Firebase Firestore and the Google Places API.

it should be noted that the application was initially developed using Expo and React Native before a fatal flaw for the project required a pivot to Swift and SwiftUI. This required a full rebuild, discussed further in section 4.5.

This project investigates whether collaborative social features in a restaurant discovery application produces a measurable increase in user retention intent compared to existing platforms that offer only individual use functionality. The findings are evaluated through a combination of structured usability testing with participants, the system usability scale and qualitative feedback gathered during testing sessions with participants.

1.2 Motivation

Existing restaurant discovery platforms are designed with a single user in mind. Despite dining being a social activity, with research suggesting that the majority of restaurants are shared experiences [1]. This suggests a potentially missed opportunity with increasing user retention through additional collaboration between friends.

The motivation for this project was to address the gap directly, and build an application that treats group dining as the primary use case, for restaurant discovery.

1.3 Research Claim

The hypothesis of this project is that integrating collaborative social features into a restaurant discovery application leads to increased user retention compared to applications that offer individual use functionality only.

1.4 Aim

The aim of this project is to design build and evaluate a native iOS restaurant discovery application that ensures group decision making is the center of the user experience. The project seeks to demonstrate that the integration of social features, specifically a real time

group voting system and shared lobby sessions can produce a more engaging experience than existing applications.

1.5 Objectives

The following objects were defined at the beginning of the project and were used to measure its success throughout the development lifecycle:

1. to design and develop a native iOS restaurant discover application using Swift and SwiftUI.
2. To integrate the Google Places API to enable location away restaurant search with support for cuisine and dietary filters.
3. To implement a real time group voting system using Firebase Firestore as the collaborative mechanism supporting the research claim
4. To conduct structured usability testing with a minimum of 10 participants.
5. To evaluate the impact of collaborative features on user retention through participation feedback analysis and system usability scale scoring
6. To achieve an above average system usability score exceeding 68 points.

These objectives are revisited in chapter 7, where its evaluated against the project outcome to determine the degree to which the project met its stated goals

2. Literature review

2.1 Existing restaurant discovery apps

Restaurant discovery apps have existed for quite some time, here are a few examples:

2.1.1 Yelp

Yelp is one of the most widely used restaurant discovery platforms originally launched in 2004 and operating across multiple continents from the Americas to Austrailia at its core, Yelp functions through crowd sourced information, allowing users to submit star rating, written reviews, photographs, and checkins for local businesses. Based on these metrics Yelp ranks the order in which they show in the search results

Features that Yelp offer include filtering options based on cuisine type, price range, distand and whether a restaurant is currently open, a feature that research has found increase user

trust in a platform as positive interactions between businesses and customers signal responsiveness from the business. [2] its mobile application includes a map view and reservation through OpenTable a commercial booking platform.

However, Yelp comes with notable weaknesses of increasing relevance to this project. The platform is well documented as vulnerable and manipulated reviews that are difficult to distinguish from genuine ones. As seen with the report, fake review detection [3] they estimate that 20% of all reviews on yelp originate from paid fake review writers, a figure that significantly undermines the reliability of the platform. A further limitation is that Yelps user base is predominantly in the United States. Which has a knock-on affect for the effectiveness in the United Kingdom where local review coverage in smaller towns is sparse.

Yelp offers no mechanics for collaborative group decision making. Its collections feature allows users to save and share lists of restaurants however these are read only for recipients.

2.1.2 TripAdvisor

TripAdvisor is one of the largest travel and dining review platforms globally, hosting over one billion reviews across hotels, attractions and restaurants as of 2022 [4] unlike Yelp and Google Maps, TripAdvisor is a travel first company meaning restaurant discovery is not its primary focus. This is reflected in the platforms design, which is heavily integrated with hotel and attraction booking making it particularly well suited to tourists. Its data coverage is especially strong for restaurants in tourist hotspots and major cities.

However, TripAdvisor has fake reviews on the platform, however the rate of which the fake reviews is a lot less at around 4.8% in comparison to Yelps 20%, [5]

Of the three platforms reviewed TripAdvisor has the least social functionality. There are no shared lists, no group voting, no social notifications of any kind. The platform is designed entirely for an individual decision making it the least suitable of the three for the group dining.

2.1.3 Google Maps

Google Maps is the most widely used restaurant discovery tool globally, as of 2023, Google Maps has over two billion monthly active users, with approximately five billion location searches processed each day [6]. Users can search for specific restaurants by name, cuisine

type or as part of a broader local restaurant query. Each restaurant has a dedicated review section allowing customers to leave a star rating and written reviews which are averaged out into an overall score. Google Maps benefits from deep integration with Google Search, meaning restaurant listings appear directly in search results alongside Maps data.

In terms of social features, Google Maps offers more than TripAdvisor but remains limited, users can save restaurants to personalised lists, and these lists can be shared with other users. However, similarly to Yelp's collections, shared lists on Google Maps are view only for recipients which limits the collaborative nature of the feature. Google previously offered a following feature which allowed users to follow each other and view recommended restaurants. However this feature was removed, further reducing the platform's social functionality.

2.1.4 Comparative summary

Across all three platforms, a consistent pattern emerges: restaurant discovery remains an individual use activity, despite being inherently social in practice. Yelp, TripAdvisor, and Google Maps each have been designed around a single user making a decision. None of the three platforms offer a meaningful mechanism for groups of users to collaborate and find restaurants together, despite the fact that dining out is one of the most social activities in everyday life.

This disconnect between the social nature of dining and the solitary nature of existing tools is well supported by Clauzel et al [1] stating that nearly three quarters of restaurant visits are shared with others, however restaurant selection is commonly decided by one person. Users are expected to individually research restaurants and then coordinate externally, before returning to the discovery app to make a booking. This fragmentation represents a usability failure and a missed retention opportunity.

From a feature perspective, the gaps are consistent across all three platforms, Yelp's collection feature allows users to save and share lists of restaurants however these are read only for recipients. There is no mechanism for friends to add to, edit or vote on a shared list. TripAdvisor offers no shared collaboration functions whatsoever, and a lack of Cuisine tags, it is obvious that TripAdvisor is not a restaurant finder but an activity finder for tourists.

2.2 Location based Services on mobile applications

Location based services have become a foundational component in mobile applications, enabling software to deliver relevant content based on the user's location. In the context of food and restaurant discovery, location data is the primary input that makes a generic search personalised, returning restaurants within walking distance rather than across the county or city.

The widespread GPS enabled smartphones fundamentally changed how users discover services near them. As Daniel Sutko States [7], locative mobile social networking can increase users' spatial awareness and meet more people in public, as LMSNs assist in traditional social norms.

In practice, restaurant discovery application use location data in a couple of distinct ways. The most common is a proximity search, in which the user's coordinates are passed on. This returns restaurants which are within the defined radius of the user. Platforms such as Google Maps and Yelp have built their core restaurant discovery experiences around this model displaying results as pins on a map with a scrollable list ordered by proximity to the users. A secondary use of location in food application is radius filtering, allowing users to define a certain area which will be relevant for them. Some users may be willing to walk 500 meters and others can walk up to 2 kilometres. This was identified in this project's usability testing, and was implemented.

Location data also adds ethical and privacy considerations. On iOS, Apple requires applications to request user permission before accessing location data via the CoreLocation framework and restricts background location access to cases where it is genuinely necessary. This project complies with these requirements, by requesting location permission on first launch and providing a manual postcode entry alternative for users who decline GPS access. a design decision supported by feedback received in usability testing.

A notable limitation of GPS based location discovery is the assumption that users are currently present at the intended city or region. In practice, a user may wish to discover restaurants near a destination they are travelling to. This limitation was identified by participants during usability testing and directly motivated the additional manual location screen, which allows users to enter a postcode as an alternative search.

2.3 User experience in Food applications

User experience (UX) refers to users' interactions with a system, encompassing usability, accessibility, emotional response, and the degree to which the product meets users' needs. As Donald Norman states Poorly designed objects can be difficult and frustrating to use [6]. In food discovery applications, UX design is particularly consequential because the decision to use a platform is completely voluntary. Users will abandon an application that frustrates them, in favour of a familiar alternative such as Google Maps.

Jakob Nielsen [8] established ten heuristics for user interface design that remain as the foundation in UX. Of these, visibility of system status, Match between the system and real

world, and user control and freedom apply most to a restaurant discovery application. Visibility of the system status refers to the application communicating clearly what it is doing. For example a loading indication while restaurants are being fetched from the Google Place API. Match between system and real world means using language familiar to the user, for example labelling a button rather than using an icon. User control and freedom means providing escape routes for the user, allowing a user to leave a lobby or change their location without being trapped in the application.

Accessibility is another part of UX design with increasing regulatory attention. The web content accessibility guidelines or (WCAG) 2.1 at level AA is the international standard for accessible digital interfaces covering requirements for colour contrast, text resizing and screen reader compatibility. As Ben Billow [9] describes a curb cut effect where accessible features increase usability for everyone. The curb cut effect was previously coined by Michael Pachovas [10]

2.4 Review and rating system

Review and rating systems are the main way by which discovery platforms communicate quality to their users. A star rating average from hundreds or thousands of user reviews provides an efficient signal that reduces potential anxiety when considering a new restaurant. Susan Mudambi [11] found that review extremity has an influence on consumer purchasing decision, with highly negative reviews having a strong deterrent, this finding has an impact on restaurant discovery as review extremity will reduce the likelihood of a user going to a restaurant with a one star rating.

However, the integrity of crowd sourced systems has always been questionable as current Wang, Zhang and Qian [3] estimate that approximately 20% of reviews on Yelp are estimated to be fake and originate from paid fake review writers, representing a substantial proportion of content on which users make decisions.

This statistic means one in five reviews on a major platform cannot be trusted, the star rating system which strengthens user trust is fundamentally compromised. Directly weakening the case for relying on existing platforms to make decisions. One way to fix this issue is having verified reviews, ensuring the customer has gone to the restaurant by giving a tag. This potentially can stop paid writers from having an impact on small restaurants as their review holds less weight.

2.5 Gap in the market

The previous sections of the literature review have established three dominant restaurant discovery platforms, those being: Yelp, TripAdvisor and Google Maps. Each serve their primary use case but all assume that restaurant discovery is an individual activity, however dining is inherently a social activity, in which studies have shown that the majority of restaurants are shared experiences [1]. The gap this project addresses is the lack of real time collaborative decision making as an integrated feature.

The consequences of this gap are observable in within everyday behaviour. Groups of friends who want to choose a restaurant together resort to external messaging platforms, sharing links through WhatsApp, or simply having one person decide for the group. This process is inefficient and represents a missed opportunity for restaurant discovery platforms to embed themselves deeper into their users social life, this in turn will increase the frequency and lower the user churn rate even more.

The relationship between social connectivity and user retention has been demonstrated in large scale platform data as analysed with usage data from 500,000 snapchat users found that new users with the larger social connections demonstrated lower churn rates than socially isolated users [12]. This finding has implications with the collaborative restaurant experience, where socially isolated users will have a higher churn rate due to the core functionality not suiting their lifestyle. However, there are other ways to lower the churn rate by increasing other features such as gamification, for all users but aiming it directly towards those who socially isolate themselves to increase retention.

In summary, the literature review reveals a clear gap, restaurant discovery is a social activity that existing platforms treat as an individual one. The lack of collaborative features such as; real time group voting, shared filters and social engagement mechanics leaves a gap in the market. This project directly aims to address the gap through the design and implementation of a collaborative restaurant discovery application and investigate whether the results produce measurable evidence of increased user retention.

3. Methodology

3.1 Chosen methodology – Waterfall

The Waterfall Methodology described by Winston Royce [13] is a sequential software development methodology, in which each phase, must be completed and verified before the

next phase begins. This linear structure produces a clear documentable development cycle. Ensuring that the projects scope and objectives are fully defined before any code is written.

Waterfall was selected as the development methodology for two primary reasons. First, the research claim and feature set were defined with sufficient precision at the outset of the project to support a formal requirements phase. The collaborative restaurant discovery concept, the Google Places API integration, and the Firebase lobby system were all identified in the scope before development started.

Second, this project was undertaken by a single developer with no team coordination requirements. one of the advantages of Agile methodologies is the facilitation of distributed team collaboration through sprint meetings, for a solo developer there is no need for sprint meetings as this would add unnecessary oversight without any benefit to coordination.

3.2 Research Methodology

Alongside the software development methodology, this project adopted a mix of methods, to approach the evaluation of the research claim. Quantitative data was gathered through the System Usability Scale (SUS) questionnaire, which produces a numerical usability score on a scale from 100. Qualitative data will be gathered through a discussion with participants, getting their thoughts as they use the application. This combination gives a mixture of both statistical resources, and direct user feedback.

4. Project plan

4.1 Project Timeline

The project was conducted over an eight month period, starting in October 2025 and concluding with the final submission on 26th In line with the waterfall methodology adopted for this project, development was organised into a series of sequential phases, each with defined start and end points. The Waterfall model, originally formalised by Royce [13], states that each phase of a software project must be completed and verified before the subsequent phase begins, providing a structured, documentable development process well suited to projects with clearly defined requirements and fixed deadlines. The full timeline was planned and tracked using a Gantt Chart. This was produced in Microsoft Excel, which provided a week by week breakdown of all tasks throughout the project.

The first phase, requirements ran throughout October 2025. During this period, the project proposal was drafted and approved by the project supervisor. Sommerville [14] notes that a well-defined requirements phase is critical to the success of waterfall projects.

The design phase took place in November and December 2025, during which the system architecture was established, the database was designed, and UI/UX wireframes were produced.

The literature review phase ran concurrently from November 2025 through to May 2026, representing the longest activity of the project. Given the depth needed to support the research claim that collaborative social features in a restaurant discovery application lead to increased user retention. The literature review required continuous reading, writing and refinement throughout the project. This approach ensures that the literature remains relevant even if the project's goal changes slightly, as this document continuously updates and evolves throughout the project's implementation.

The implementation phase began in December 2025 with the establishment of the development environment, initially using Expo and React Native as the intended frontend framework. The backend API, built using Node.js, Express, and Firebase, was developed in January 2026, followed by the setup and implementation of the Google Places API and the initial Map view. However, in February 2026, a critical event occurred. Throughout the development process using Expo, a fatal flaw derailed the implementation. When multiple users used the Restaurant finder, the app would not load on separate devices, despite repeated attempts to resolve the issue. This type of constraint was not originally recognised as a potential risk, and unfortunately, this resulted in a full code rebuild, the aim of which was to remove Expo and replace it with a native Swift build. As this issue was found early enough, it gave time for research into Swift. The Swift rebuild was completed by April 2026, while the backend had been completed months earlier. The frontend needed to be rebuilt from the ground up. The longest frontend screen was the swipe view. This was due to the Expo build using a specific library for the swipe function, whereas the Swift build needed the entire function built. This pivot was managed within the project's existing timeline, with significant trade-offs in documentation and reporting.

The testing phase started in late April 2026, focusing on ensuring a stable application ready for publication and covering the research claim. During this stage, the System Usability Scale (SUS) was administered following testers' feedback; the results were analysed and written up in early May. The final report phase ran from late April through mid-May, overlapping with the end of testing. All documentation being completed before the end date of the 26th of May 2026

4.2 Milestones

Milestones represent key checkpoints within a project timeline at which a significant portion is completed or a critical decision is made. In the context of effective project management milestones are a zero-time event but are a powerful way of supporting the project [15], for this project

milestones, were defined at the end of each waterfall phase, as well as at notable unplanned events that required a long response.

Milestone 1 Requirements phase completion (October 2025)

The first milestone marked the completion of the requirements phase at the end of October, by this point the project proposal had been approved. All requirements had been documented and initial source gathering for the literature review had begun. The sign off at the end of October was necessary for progressing into the design phase, as this keeps the methodology consistent with the sequential structure of the waterfall methodology as discussed by Royce [13].

Milestone 2 Design phase completion (December 2025)

The second milestone marked the completion of the design phase in mid-December. Systems architecture had been established with the database schema, other tasks completed include a wireframe, and the system architecture documentation. After which this discussed with the project supervisor in early January, where a few potential flaws were pointed out, and redesigned.

Milestone 3 Expo failure identified (February 2026)

The third milestone was unplanned and represented a critical risk event rather than a phase completion. In February 2026, a build failure within Expo if the pivot to swift. This failure exposed a potential issue with completing the project before the final submission, as the entire frontend code had to be rewritten.

Milestone 4 Swift MVP complete (April 2026)

Following the decision to rebuild the frontend in swift, the fourth milestone marked the completion of a minimum viable product in April 2026. All core features have been implemented, these include; location aware restaurant search, Google sdk map view, group voting and google user reviews. This milestone confirmed the implementation phase had been completed despite disruption and that the project could proceed to testing.

Milestone 5 Literature review (May 2026)

The fifth milestone marked the completion of the literature review in early May 2026. Unlike other phases, the literature review was conducted as a continuous activity spanning the full duration of the project, from November 2025 onwards, this was due to the iterative nature of research required throughout the development and testing.

Milestone 6 testing phase complete (May 2026)

The sixth milestone marked the conclusion of the testing phase on the 10th of May 2026. By this point unit and integration testing using manual tests, and usability testing sessions with 20 participants had all been completed. The system usability scale questionnaire had been analysed, and the findings had been written up in the testing and feedback chapter of the report. The completion of testing confirmed that the application had been evaluated against all criteria.

Milestone 7 final submission (22 May 2026)

The seventh and final milestone was the submission of the completed project report on the 22nd of May 2026 representing the culmination of the full eight month project. All chapters had been drafted, proofread and formatted in accordance with all university guidelines.

4.3 Resources

The following hardware, software tools, and external services were used throughout the development of this project.

Hardware

- MacBook - required as the only platform able to run Xcode, Swift development cannot be performed on windows or Linux based operating systems.
- iPhone - required as the application is developed for this platform. Emulators were used for testing and feedback stages.
- Internet Connection – required for google places API calls, Firebase database updates and access to the group activities.

Development Tools

- Xcode – Apple’s integrated development environment (IDE) used for all swift development, debugging and device deployment
- Swift /SwiftUI – The programming language and UI framework used for the iOS frontend application
- Firebase – Google’s cloud-hosted databased used to store lobby session data and group voting information.
- Google Places API – used to search for nearby restaurants retrieving details such as: Location Name, Address, Photos, and user reviews.

- CoreLocation – Apple’s framework used to retrieve the users GPS location, This was previously used in development.

4.4 Risk Assessment

The risk assessment was treated as a living piece of documentation throughout the project. An initial risk assessment was made during the planning phase, this was modified in February 2026 following the swift pivot, however the risk assessment focussed on the backend over the frontend

Google places API rate limit exceeded during development

Likelihood: Medium ~ Impact: High

Mitigation: minimising the usage of the google places API by only calling certain features such as the photos just once this reduces unnecessary calls and ensures that features worked another mitigation factor is creating a billing cap in the google cloud console.

Feature creep causing delays to MVP completion

Likelihood: High ~ Impact: High

Mitigation: defining a fixed minimum viable product scope at the start. Any additional features could be added as future work or developed at the end once the MVP is finalised.

Low usability testing participation

Likelihood: Medium ~ Impact: Medium

Mitigation: recruiting participants early through private life and university. Creating a target number of participants with a variety of backgrounds.

Time pressure from other academic modules

Likelihood: High ~ Impact: High

Mitigation: ensure that the Gantt chart is completed early into the development and ensure that all deadlines are met and any delays are documented.

Updated risk assessment after swift pivot

After the critical issue exposed by Expo in February 2026, an updated version of the risk assessment was carried out. The following risks were added following the update to the backend.

Expo failed to build on multiple devices

Likelihood: Occurred ~ Impact: Critical

Mitigation: Decision to rebuild the frontend entirely in Swift / SwiftUI. All Expo dependencies removed. Additional weeks added to absorb the delay of rebuilding the application. Deadline Maintained for the 26 May 2026.

Swift rebuild scope exceeding available time

Likelihood: High ~ Impact: High

Mitigation: reduced feature scope to base MVP, increased collaborative features such as shared lists deferred to future work. Additional buffer weeks provided a safety net.

Firebase data structure incompatible with real time voting requirements

Likelihood: Medium ~ Impact: High

Mitigation: Restructured Firebase collection to store lobby state as a single document updated each vote, ensuring all users see consistent data.

Reduced implementation time impacting test coverage

Likelihood: High ~ Impact: Medium

Mitigation: Prioritised end to end testing of the core user flow. Testing reduced however still include 20 users that provide feedback from different backgrounds and demographics.

4.5 Plan evolution ns

The project plan had several significant changes throughout the development lifecycle. This section documents the key modifications from the original plan and the reasoning behind each decision.

Literature review extended to run continuously

Originally planned as a 3 week phase completed before implementation began, the literature review as extended to run continuously throughout the project from November 2025 to May 2026. This decision was made to ensure the theoretical grounding of the research claim remained up to date with design decisions, especially with the swift pivot the research particularly around the collaboration design.

Expo replaced by Swift/ SwiftUI

The most significant change to the project plan. Expo was originally chosen for its cross-platform capability, the project failed to build on multiple devices which made this development platform unviable for the project, after 5 days of trying to figure out the future of the project, the decision was made to pivot straight into Swift. The decision to move straight into Swift had three main points, the first one being, Xcode tools made debugging easier and more straightforward than other cross platform IDEs. The second point was avoiding the same compatibility problem that caused the Expo Failure. The final reason was due to documentation, as many cross platform IDEs are very under documented which would lead to more difficulty in debugging or just general development of the application.

Apple Mapkit replaced by Google Maps SDK

A direct consequence of dealing with APIs is complying with their terms of service, while developing the Swift version of the application the Apple Mapkit was used to debug and ensure that the data pulled from Google Places was readable cross products, however during research of the Google Places ToS, data on a non-Google map is a violation of their terms of service. From this Apple Mapkit was removed and replaced by Google Maps, to ensure that the application was in line with their ToS. [16]

Collaborative shared lists deferred to future work

The original MVP included a shared restaurant list that users could build and share with their friends. Following the time lost in the pivot, this feature was moved into future work, this decision was to protect the core MVP that directly leads to the research claim.

Location selector screen added following usability testing

While not originally in scope, the location selector screen was added very early into testing as the initial 5 participants asked about including this. Adding this screen added increased depth to the application and made the restaurant selection process even more collaborative.

5. Design and implementation

5.1 requirements

This section outlines the functional, non-functional and hardware requirements that governed the development of the application. These requirements were defined during the design phase, this helped with being consistent with the waterfall methodology.

Functional Requirements

- Location aware restaurant search using either the device location using the lobby selector screen
- Lobby selector screen allowing users to choose a manual postcode entry
- Google Places API integrated to retrieve nearby restaurant data based on the user's location
- Google Maps SDK used to display restaurants in a map view with interactive pins showing the restaurant selected by everyone.
- restaurant cards displaying the name, cuisine, rating and distance (in map view)
- Group voting system, users vote by liking or skipping the restaurant, and a tiebreaker vote if multiple restaurants are on the same amount of likes
- Lobby filters include Cuisine, Distance and dietary filters
- Adjustable search radius allowing users to broaden or narrow the scope of results.
- Firebase synchronisation for all lobby states, users receive the same data.

Non-functional requirements

- Results must load within one and a half seconds on a 5g mobile connection
- Application must run on iOS 15 and above
- GDPR Compliance no personal data stored beyond the user nickname
- Google Place API ToS compliant, only place_ID, restaurant name, and address stored in firebase. All other data fetched on demand.

Hardware requirements

- iPhone, required to run the restaurant discovery application
- MacBook, required to run Xcode, the only supported development environment for Swift applications.

- Internet connection, required for Google Places API calls and Firebase data synchronisation

5.2 Architecture

The application follows a client server architecture, with the swift iOS client communicating with three external services, these are: the Google Places API, Firebase and the Google Maps SDK

Client – Swift / Swift UI

The frontend application built entirely in swift and SwiftUI. Handles all the user interaction including the lobby screen, filters. Swipe cards and location selector and interacts with the Google Maps SDK. Communicates directly with the Google Places API and Firebase

Google Places API – Restaurant data

The Swift app calls the Google Places search directly, passing the user's location selected with their active filters. Results are assigned a swipe card and only liked restaurants are held on the firebase database.

Firebase (Firestore) – Lobby details and real time database

Firebase Firestore holds the session data, this includes what users are in the lobby, what phase the lobby is in, filters selected and which restaurants have received votes. This service acts as the sever. Enabling the collaborative group voting without requiring a custom backend server with CRUD functions.

Google Maps SDK – Map display

The Maps SDK was used for the mapping framework, due to the native compatibility with the Google Places API. It previously replaced the Apple MapKit after finalising the MVP completion, for two main reasons the native compatibility with the Places API and the Compliance with Places ToS.

5.3 UI/UX design

Initial Wireframes were produced using a Paint prior to development in Expo, this provided a very basic reference for the layout of each screen and the user flow. The interface was refined throughout development within Expo and later Swift/ SwiftUI. See figure 1 for the early Paint wireframe

Design Decisions

- *Lobby Screen with filter pills displayed* – Cuisine and Dietary filters are at the surface rather than being hidden in another menu, reducing the number of menus before the search. The aim of this is to reduce friction in the user experience, while studies show that users who are tested on two different systems will choose the one with more friction [17], however Ronald Rasch states that systems with more friction has more dishonesty [18].
- *Swipe card interface* – restaurant information is presented as their own individual cards, the user swipes through, reducing the strain of looking through a long list, each card shows the name of restaurant rating and address.
- *Map and list hybrid* – the locations that users liked during the swipe phase are displayed on the map and listed in a footer bar, allowing users to get the directs and share a list of all restaurants that have been liked.
- *Location selector screen* – added post-testing the first 5 participants gave feedback where they would prefer to select a location. This screen allows the users to choose the location instead of being based on the GPS location of the user. At the same time GPS location has been taken out of the application.
- *Like button replaced heart icon* – changed post testing after 10 participants did not think the heart icon represented likes. This greatly increased the amount of likes per lobby. Previously the users thought that the one like would only be

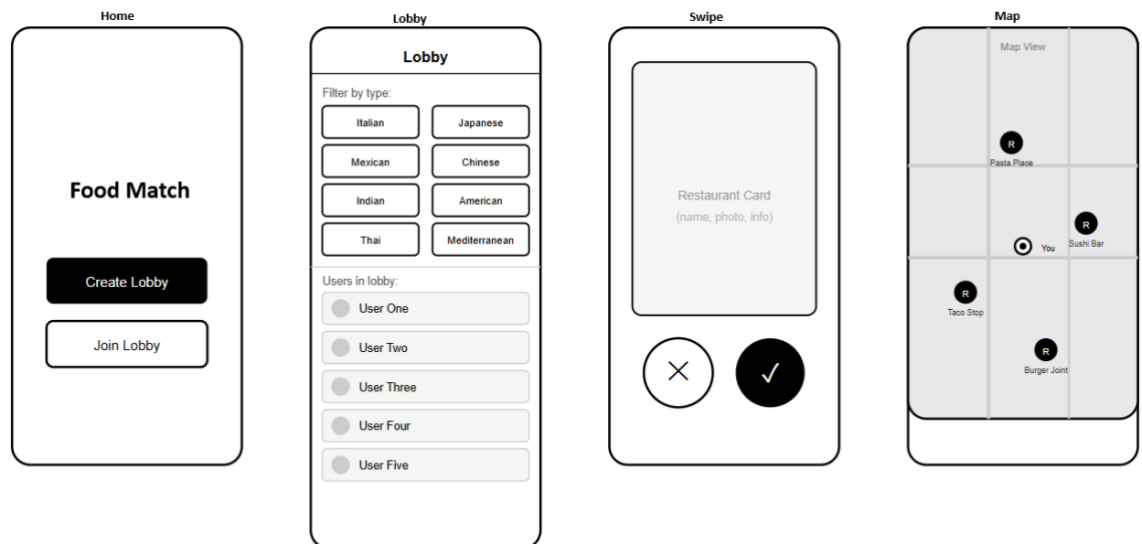


Figure 1, simple wireframe created

5.4 Google place api integration

The Google Places API is the sole source of restaurant data within the application. The integration is requested on demand, fetching data when needed rather than storing it in bulk.

Integration flow

Step 1 – The user opens the lobby screen and the app retrieves the location through the location selector screen.

Step 2 – the app sends a nearby search request directly to the Google Places API, passing the coordinated, search radius, and any active filters.

Step 3 – Google returns a list of matching restaurants. The app parses the response and renders result as a swipe card.

Step 4 – when a user votes yes on a restaurant, the place_id, name, and address are written to the firebase real time database, so all lobby members can see liked restaurants.

Step 5 – if a user taps a restaurant for directs, the app sends the user to the restaurants Google page from there, they can get directions, place a booking or save it for later.

The searchPlaces function, shown in figure 2, handles the full process of executing a nearby search based on the user's location. While the function begins by determining the location source, it prioritises a manually entered location from the locator screen over the device GPS, using Swift's nil coalescing operator (??). A nil coalescing operator is an additional safety net that checks whether a variable is defined; if not, it falls back to the backup value. This directly reflects the design iteration

added following usability testing, where the first 5 testers wanted to search somewhere other than their current position.

```
func searchPlaces(filters: [String]? = nil) async {
    // Prefer manual location (from picker) over GPS
    let location = manualLocation ?? currentLocation

    guard let location else {
        print("No location available")
        locationManager.requestLocation()
        return
    }

    let keywords = (filters ?? selectedCuisines.map { $0.rawValue }).joined(separator: "|")
    let lat = location.coordinate.latitude
    let lng = location.coordinate.longitude
    let radius = Int(manualLocation != nil ? searchRadius : 1500)

    let urlString = "https://maps.googleapis.com/maps/api/place/nearbysearch/json"
    + "?location=\(lat),\(lng)"
    + "&radius=\(radius)&type=restaurant"
    + (keywords.isEmpty ? "" : "&keyword=\(keywords)")
    + "&key=\(apiKey)"

    print("Searching: \(urlString)")
    guard let url = URL(string: urlString) else { return }

    DispatchQueue.main.async { self.loading = true }

    do {
        let (data, _) = try await URLSession.shared.data(from: url)
        let decoded = try JSONDecoder().decode(PlacesResponse.self, from: data)
        let mapped = decoded.results.map { result in
            Place(
                placeId: result.placeId,
                name: result.name,
                rating: result.rating,
                vicinity: result.vicinity,
                geometry: result.geometry.map {
                    Place.Geometry(location: Place.Coordinate(lat: $0.location.lat, lng: $0.location.lng))
                },
                photos: result.photos.map {

```

Figure 2, this figure shows the code behind the Google Places search

5.5 Key features included

- Location aware restaurant search – using a manually entered postcode via the location selector screen, or the device GPS as a fallback
- Google Places API integration – nearby search and place details requests with filters.
- Swipe based restaurant card browsing – cards restaurant name, cuisine, rating and address
- Interactive Google Maps SDK – Used to show the user locations of each restaurant they liked
- Group voting system – thumbs up and down per restaurant all scores synchronised in the firebase server
- Firebase real time database – all users in the same lobby see live updates as votes are case

5.6 Challenges encountered and resolved

Challenge: Expo failed to build on multiple devices at once

Resolution: After 5 days of trying to fix the application, the decision was made to rebuild the entire frontend in Swift and SwiftUI. This was the most significant challenge of the project, taking a significant portion of time to readjust.

Challenge: Replacing the Apple MapKit with Google Maps SDK

Resolution: After switching to swift, the Apple MapKit was the natural choice to test the Map experience, however, to resolve a potential Google ToS violation, the decision was made to change to the Google Maps SDK

Challenge: Location discoverability

Resolution: during early testing participants discussed that a location selector, would greatly improve the flow of the application, users would be able to plan for the future rather based on their current location. A location selector page was setup to allow for users to select where to search see figure 3 for the location selector page

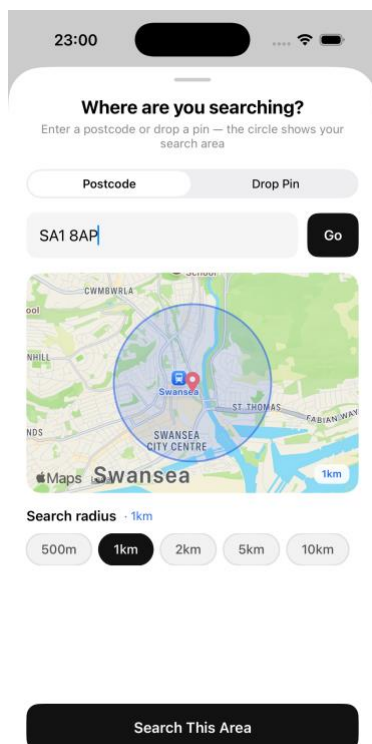


Figure 3, the location selector page

5.7 Application flow

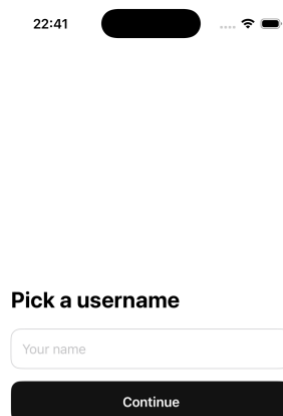
The application flow begins when the user first launches the application and is presented with an onboarding screen, prompting them to enter a username. This screen is intentionally minimal, requiring a display name before the user proceeds. See figure 4 for a visual representation of the onboarding screen

Following the onboarding screen the user moves to the main index screen, where they are presented with two options, either create a new lobby or join an existing lobby. Creating a lobby generates a unique lobby code which can be shared with other users. Multiple users can join the lobby by entering the code. See figure 5 to see the index page, with the join box open visible. Before the lobby search can be initiated the user is required to set a manual location using the location selector screen. This addition was added after direct feedback in the usability testing, with the first 5 participants all suggesting this was added. See figure 3 (previously used) for the location selection

Once a location has been set, the user modifies their search by selected filters in the lobby, see figure 6 for the lobby view with different filters.

After filters are confirmed, the application moves to the swipe view, where a maximum of 20 restaurants results is fetched from the Google Places API. This limit was implemented deliberately to reduce unnecessary API calls. See figure 7 for the swipe view

Once the majority of the lobby members tap ready, the application transitions to the map view, rendering the shortlisted restaurants as pins using the Google Maps SDK, see figure 9 for the map view.



The screenshot shows a mobile application interface for selecting a username. At the top, the status bar displays the time 22:41, a black pill-shaped notification area, and icons for cellular signal, Wi-Fi, and battery. Below the status bar, the text "Pick a username" is displayed in bold. Underneath, there is a text input field with the placeholder text "Your name". At the bottom of the screen, there is a dark blue button with the text "Continue" in white.

Figure 4 onboarding username selection



Food Match

Create Lobby

Enter lobby code

Join

Figure 5 Main index page with join lobby text box



Food Match

LOBBY CODE

W B J 4 I

Players (1)

R

RYAN (you)

Waiting...

What are you in the mood for?

Pick as many as you like, we'll find the overlap.

Italian

Japanese

Mexican

Chinese

Indian

American

Thai

Mediterranean

Korean

Vegan

Seafood

BBQ



Set Search Location

Start Now

Set a search location before starting

Mark Ready (0/1)

Leave Lobby

Figure 6, Lobby view with filters

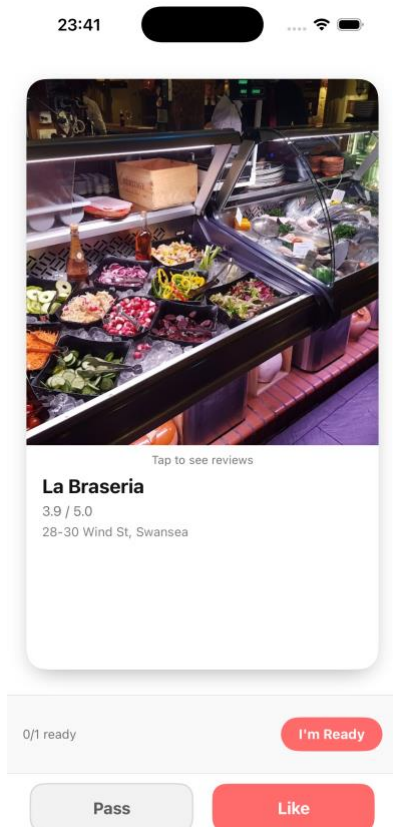


Figure 7 the swipe view

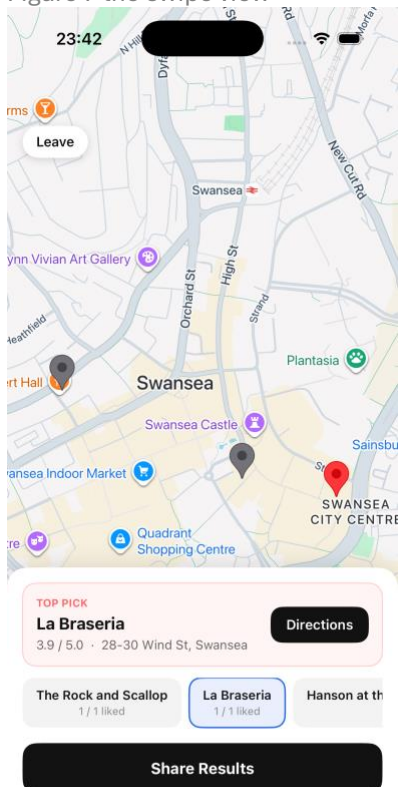


Figure 8 Google Maps SDK

6. Testing and feedback

6.1 Testing Strategy

Testing was conducted manually using the iOS debugger on Xcode, the testing strategy walked through each functional requirement of the application, ensuring that the MVP was functioning correctly. Tests were defined in advance with a clear input, expected outcome, and actual outcome recorded for each test. Manual testing was chosen as the primary approach due to the time constraints, with usability testing conducted alongside to evaluate the application from a new user perspective.

A total of 15 manual test cases were evaluated across the core features of the application, results were recorded as pass or fail, with any failures documented.

Testing was completed on an emulator rather than on a physical device, emulator does not perfectly replicate all real world conditions particularly around GPS accuracy and network latency.

6.2 Test cases

Test Nu	Test case	Feature	Expected outcome	Actual outcome	Result
1	Open app and grant location permissions	CoreLocation /GPS	App retrieves current coordinates	Coordinates retrieved, was not used for usability testing (after location selector added)	Pass
2	Deny location permissions	CoreLocation Location selector	User is able to add a postcode manually to search	Location selector screen displayed correctly	Pass
3	Enter valid UK postcode in location selector	Location selector	App accepts postcode and uses it as the search	Postcode accepted and is used for the search	Pass
4	Enter invalid Postcode in location selector	Location Selector	App rejects postcode and displays error message	Error message displayed "Postcode not	Pass

				found. Try again”	
5	Search for restaurants with no filters applied	Google Places API	Returns 20 nearby restaurants with default categories	Restaurants returned and displayed on swipe cards	Pass
6	Apply cuisine filter and search	Filters	Results restricted to selected cuisine only	Filtered results returned	Pass
7	Apply Vegan Filter	Filters	Results restricted to select restaurants with vegan options	Restaurants with vegan options displayed	Pass
8	Adjust search radius and search	Search radius	Results updated to reflect new radius	All results remain inside of the search radius	Pass
9	Swipe through restaurant card	Swipe interface	Cards display restaurant name, Cuisine, Rating, and distance	All card data displayed correctly	Pass
10	Tap a restaurant card to see reviews	Google Place API	Results are displayed from google places	Top 5 reviews are displayed to the user	Pass
11	Switch from swipe view to map view	Google Maps SDK	Map renders with restaurant pins at correct locations	Pins are displayed correctly on map	Pass
12	Tap a pin on the map view	Google Maps SDK	Restaurant information pops up	Restaurant card pop up	Pass
13	Create a lobby and invite a second user	Firebase lobby	Lobby is created in firebase and second user joins successfully	Lobby created and joined correctly	Pass
14	Cast a vote on a restaurant in the swipe	Swipe likes	User successfully likes restaurant added to Firebase	Restaurant is added to Firebase correctly	Pass

15	First time app use sends user to make a username	UsernameView	User sent to a name entry page	User is sent to a page where they set a name	Pass
----	--	--------------	--------------------------------	--	------

6.3 Usability testing

Usability testing was conducted with twenty participants, while the users were testing the applications they were asked to talk about their thoughts and reactions as they completed a set of tasks, this methodology created usability issues that quantitative testing couldn't detect, as it gave reasoning behind user thoughts rather than just the outcome.

Protocol

Each participant was briefed verbally on the purpose of the study, and how their feedback would be used, and that participation was entirely voluntary. Verbal consent was obtained by all participants.

Participants were asked to complete tasks without assistance, these tasks include: opening the lobby screen and applying a filter, using the location selector to enter a postcode, swiping through restaurant cards, and finally joining a group lobby and casting votes

Following the tasks, all participants completed the System Usability scale questionnaire, this is a questionnaire created by John Brooke, in 1995 [19] stating the need of a low-cost usability scale that can be used for global assessments of systems usability. The applications overall score achieved was 76 out of 100, which sits above the acceptable threshold of 68

6.4 Participant demographics

Twenty people participated in the testing of this application, these participants were close friends and family all with varied levels of experience of technical background. As these participants are from a close circle there's a chance of proximity bias. The demographic breakdown of participants is as follows:

- Total participants: 20
- Age range: 19-56
- Gender: mixed
- Technical background: Mixed, Participants ranged from non-technical everyday smartphone users, to those with programming experience.
- Recruitment method: personal network

- Consent: verbal consent obtained from all participants prior to each session of feedback

6.5 Feedback Analysis

The following themes emerged from analysis of the testing sessions, and discussions with the participants. Results are represented out of 20 participants

Positive findings

18/20 – found the swipe interface intuitive with the like buttons

16/20 – identified the map view as the most valuable feature

14/20 – found the app visually clean and easy to navigate on first use

12/20 – found the filter system intuitive and used it without guidance

10/20 – described the app as fast and responsive on device

Negative findings

11/20 – wanted a quicker lobby invite system to get friend into the lobby faster

8/20 – initially ignored the filters only engaging with them after prompting during the testing session

8/20 – wanted shared persistent lists for a more collaborative experience, this was identified as future work

7/20 – wanted to see more restaurant information while swiping such as opening times and menu options

Collaborative feature findings

18/20 – Stated they would return to the app over an application built for individual use

15/20 – indicated they would use the app again specifically to plan a group meal with friends

15/20 – found the app more useful than existing platforms for group dining decisions.

14/20 – said the group voting was easier than their usual method of deciding with friends typically messaging back and forth on a separate platform.

12/20 said they would choose this application over other restaurant finders for group dining

7. Conclusion

7.1 Critical Analysis of the final product

The application successfully delivers a location aware collaborative restaurant discovery experience on iOS. The core research claim, that integrating collaborative social features leads to increased user retention, is supported by the usability testing findings, with 15 out of 20 participants indicating they would return to the application specifically for group dining scenarios and 14 out of 20 stating the voting feature was preferred to their existing method of decision making

However, the claim is only partially proven within the scope of this project. As true retention is measured over weeks or months of use, whereas participant tested this application two times. The evidence gathered is more intent rather than demonstrated retention.

with regards to the project objectives defined in section 1.5 all six were met within the available timeframe:

1. a native iOS restaurant discover application using Swift and SwiftUI, designed and developed - Delivered
2. The Google Places API was integrated with location aware search and filter support - delivered
3. a real time group voting system implemented using Firebase Firestore - Delivered
4. structured usability testing was conducted with 20 participants - delivered.
5. The impact of collaborative features on retention intent was evaluated through participant feedback
6. An above average SUS score was achieved of 76 exceeding the 68 at the beginning.

7.2 Evaluation against existing products

Compared to the three platforms reviewed in the literature review, the application addressed the collaborative gap that all three leave unfilled

vs Yelp

Yelp offers a social layer through friend activity feeds, and shareable collections but these are passive and read only for recipients. Whereas the group voting system enables active and synchronised collaboration between all users, simultaneously rather than viewing each others lists after independently building that list

Vs TripAdvisor

TripAdvisor has no collaborative functionality whatsoever, it is entirely designed for individuals decision making. This application addresses the group dining use case that TripAdvisor ignores entirely.

Vs Google Maps

Google Maps offers shareable lists, but these are view only for recipients and generate no re engagement triggers. This application's real time firebase voting, provides a meaningful interactive collaborative experience, though Google Maps significantly outperforms this application on coverage and data variety

7.3 Strengths of the project

- SUS score of 76 - above the accepted average threshold of 68, indicating the application is well received by users on first interaction.
- Collaborative voting featured validated – 14 of 20 participants explicitly preferred the in app voting system over their existing group decision methods.
- Expo to swift pivot resolved without endangering the project, demonstrating effective risk management and adaptability.
- Google ToS compliance maintained throughout – storage of only key information such as place_id, name and addresses stored in firebase, with all other data fetched on demand
- Realtime firebase synchronisation – the lobby data instantly updates for all members, simultaneously, providing the live collaborative experience which aims to prove the research claim
- Location selector iteration – a feature just outside of the original scope, was designed, built and validated within the available timeframe in direct response to feedback from usability participants.

7.4 Limitations and weaknesses

Retention measured as intent and not behaviour – due to the length of the project meaningful retention cannot be measured as actual return visit rate. A longer study over several weeks would be required to substantiate the research claim.

Small number participants – twenty participants were recruited via personal network which creates a potential for proximity bias. These participants with a proximity bias could provide more favourable feedback than a random sample. A larger and more diverse sample size could give a more reliable source of data.

Crowd sourced dietary tags – the dietary filter results depend entirely on data returned by the Google Places API, which does not consistently tag attributes across all restaurant listings. Users may encounter inaccurate dietary information for some establishments. This is one of the main limitations identified in the literature review regarding Yelp's inconsistency.

Automated testing not implemented – due to the time constraints from the swift pivot in February 2026 automated unit and integration testing was not implemented. All testing was conducted manually using the iOS simulator within Xcode. A full automated testing environment using XCTest could improve stability on the program. Providing reproduceable and scalable tests.

7.5 Methodology evaluation

The waterfall methodology provided a structured framework that suited the independent development. The formal requirements phase ensured the research claim and feature set were defined before any form of development took place, reducing the risk of scope creep.

The methodologies rigidity proved mainly beneficial with the expo build error occurring during development, sufficient time remained to pivot into the Swift development, if the error happened during testing, it would have been incredibly difficult to be able to dedicate enough time to the rebuild.

It is also worth noting that Royce [13] pointed out that a purely sequential waterfall without feedback loops carries risks in software development, the usability testing iterations introduced a degree of feedback that partially mitigated this. Although this came late in the project lifecycle

7.6 Ethical Considerations

The following ethical considerations were addressed throughout the project.

Participant consent – verbal consent was obtained from all 20 participants prior to each usability testing session. Participants had been informed of the study’s purpose, how their feedback would be used, and that participation was voluntary and could be withdrawn.

Data minimisation – no personal information was collected from participants, feedback was recorded as frequency counts rather than assigned to individuals.

Google Places API ToS – the application was designed to comply with their terms of service, only storing the necessary data for collaboration to be possible.

GDPR compliance – no personal user data is stored beyond a nickname the user chose when they initially open the application. Location data is only accessed with explicit permission and is not retained once the session ends.

7.7 Future work and recommendations

The following extensions and improvements are recommended for future development of the application.

1. Persistent collaboration lists, allow users to build a list to send to their friends outside of the live lobby session, enabling more collaboration between users
2. Retention study – conduct a study over four to six weeks with a larger more diverse participant group to measure actual return rates rather than intent producing more robust evidence for the research claim.
3. Android version – extend the application to Android using android studio, to build the application. This will increase the potential user base significantly
4. Expand map coverage – the current implementation is optimised for UK postcodes, expanding the location selector to support international address would broaden the potential userbase.
5. Use XCTest to automate tests, from which this should improve stability of the application.

7.8 Summary and conclusion

This project set out to investigate whether collaborative social features inside a restaurant discovery application leads to increased user retention, compared to individual use alternatives. The evidence gathered throughout the project provides meaningful support for this claim.

All six project objectives were met within the available timeframe despite the significant unplanned rebuild of the front end from Expo and React Native to Swift and SwiftUI in February 2026, while research evidence gathered showed support for the research claim, it is important to acknowledge however that the evidence that supports retention intent rather than

demonstrated retention behaviour. A longer study over several weeks would be required to fully substantiate the research claim.

8. Bibliography

- [1] N. G. C. R. Amélie Clauzel, "Dining alone or together? The effect of group size on the service customer experience," *Journal of Retailing and Consumer Service*, vol. 47, pp. 222-228, 2019.
- [2] S. Beverley A., P. Helen E. and B. Ralf, "Online travel reviews as persuasive communication: The effects of content type, source, and certification logos on consumer behavior," *Tourism Management*, vol. 39, no. 1, pp. 1-9, 2013.
- [3] W. Zehui, Z. Yuzhu and Q. Tianpei, "Fake Review Detection on Yelp," 2017.
- [4] Tripadvisor, "Travelers Push Tripadvisor Past 1 Billion Reviews & Opinions! | Tripadvisor," Tripadvisor, 1 February 2022. [Online]. Available: <https://ir.tripadvisor.com/news-releases/news-release-details/travelers-push-tripadvisor-past-1-billion-reviews-opinions>. [Accessed 10 May 2026].
- [5] Tripadvisor, "Tripadvisor's Review Transparency Report highlights fight against fake reviews and strong growth in user-generated content," TripAdvisor, 12 April 2023. [Online]. Available: <https://tripadvisor.mediaroom.com/review-transparency-report-2023-ANZ>. [Accessed 20 May 2026].
- [6] H. Maria, "Essential Google Maps Statistics & Trends to Watch in 2025," Loopex Digital, 3 April 2025. [Online]. Available: <https://www.loopexdigital.com/blog/google-maps-statistics>. [Accessed 20 May 2026].
- [7] S. Daniel M. and d. S. e. S. Adriana, "Location-aware mobile media and urban sociability," *New Media & Society*, vol. 13, no. 5, pp. 807-823, 2011.
- [8] J. Nielsen, "10 Heuristics for User Interface Design," Nielsen Norman Group, 24 April 1994. [Online]. Available: <https://www.nngroup.com/articles/ten-usability-heuristics/>. [Accessed 20 May 2026].
- [9] B. Billow, "Why Accessibility Benefits Everyone," Isadora Digital Agency, 30 January 2025. [Online]. Available: <https://isadoradigitalagency.com/insights/accessible-design>. [Accessed 20 May 2025].
- [10] A. Blackwell, "The Curb-Cut Effect (SSIR)," Stanford Social Innovation Review, 30 December 2017. [Online]. Available: https://ssir.org/articles/entry/the_curb_cut_effect. [Accessed 21 May 2026].
- [11] M. Susan M. and S. David, "What Makes a Helpful Online Review? A Study of Customer Reviews on Amazon.Com1," *MIS Quarterly*, vol. 34, no. 1, pp. 185-200, 2010.
- [12] Y. Carl, S. Xiaolin, J. Luo and H. Jiawei, "I Know You'll Be Back," Association for Computing Machinery, New York, NY, USA, 2018.
- [13] R. Winston, "MANAGING THE DEVELOPMENT OF LARGE SOFTWARE SYSTEMS," The Institute of Electrical and Electronics Engineers, 1970.
- [14] I. Sommerville, *Software Engineering*, London: Pearson, 2016.
- [15] S. Hakeem Omolade, "Evaluation of Motivating and Requiring Factors for Milestones in IT Projects," *Procedia Manufacturing*, vol. 51, pp. 1469-1477, 2020.

- [16] Google, "Policies and attributions for Places API," Google, 1 January 2026. [Online]. Available: <https://developers.google.com/maps/documentation/places/web-service/policies#attribution-requirements>. [Accessed 18 May 2026].
- [17] M. Thomas, H. Sarah and S. Ulrik, "Design Friction," *Proceedings of the 31st European Conference on Cognitive Ergonomics*, pp. 41 - 44, 2019.
- [18] R. Ronald and Y. Jeffrey, "c Friction in the Design of Information Systems: The Design Dilemma," *Journal of Information Systems*, vol. 11, no. 1, pp. 19-30, 1997.
- [19] B. John, "SUS -A quick and dirty usability scale Usability and context," Redhatch Consulting Ltd, 1995.
- [20] S. Beverley A. and B. Victoria, "The impact of online reviews on hotel booking intentions and perception of trust," *Tourism Management*, vol. 32, no. 6, pp. 1310 - 1323, 2011.
- [21] R. M. Baecker, *Readings in Human-Computer Interaction*, Elsevier, 2014.