CampusGuessr: Gamification for Campus Navigation

Maxine Yang
COMPSCI 732
University of Auckland
Auckland, New Zealand
myan565@aucklanduni.ac.nz

***Abstract*—This report outlines the planning and development of '*CampusGuessr*', a multi-player application designed to gamify the process of learning the University of Auckland campus. We discuss the motivation, design, implementation, and challenges encountered throughout development. The final product exceeded our initial goals, resulting in a polished, fully functional game that engages users and demonstrates potential for improving spatial learning among university students.**

I. Introduction

Navigating through a university campus can be an overwhelming and challenging task for new students at the University of Auckland. It often takes time for students to locate their lecture and lab rooms, and they may initially struggle to comprehend the inconsistent layouts of certain buildings and the methodology by which rooms are numbered. Although the university does provide some guidance in the form of the *Campus Maps* application and building layout plans, these resources can be confusing and promote a more passive style of learning locations, in which students only learn locations as a result of necessity as opposed to active learning. This passive learning can result in poor retention and reduced spatial confidence.

To address this issue, the WD-40 student software team has developed *CampusGuessr*[1] as a solution to simplify and enhance this learning experience. *CampusGuessr* is an educational, web-based game that challenges the user by providing a visual cue in the form of a room image, which the user then has 6 attempts to guess the corresponding room code, receiving points based on speed and accuracy. The game incorporates elements of social gamification, including multiplayer support, score tracking, and a public leaderboard to encourage friendly competition. These elements are designed to promote active learning and repeated interaction with university locations, allowing students to build familiarity and retain knowledge of the campus layout out of interest rather than necessity. Gamification strategies have been demonstrated to be effective in helping students maintain motivation for learning [2] and retain information in the long-term, especially when social elements such as multiplayer are involved [1]. By focusing on play, *CampusGuessr* reduces the need for students to rely on campus resources in the future and helps them internalise spatial information more effectively.

[1]https://campusguessr.fly.dev/

Our goal was to create a full-stack web application that promotes better spatial learning of university locations, in a way that encourages exploration, active learning, and even competition. To achieve this, we aimed to include:

- An accessible, responsive, and intuitive user interface, to ensure that new students can quickly understand and use the application.
- Core gameplay functionality, including text input, room displays, and answer matching.
- Progress-tracking and scoring to promote engagement and retention.
- Multiplayer aspects to promote social competition, including user accounts and a global leaderboard, should time permit.

Ultimately, the goal of *CampusGuessr* is to assist with spatial navigation, and help students better adjust into university life while promoting their confidence and autonomy.

II. Related Work

A. Current Alternatives

The University of Auckland currently provides the *Kāhu / Campus Maps*[2] web application to assist students with campus navigation. In this application, users can search for room codes or building names using a search bar, and access quick links to commonly used areas of campus, such as toilets and student services. However, while functional, the application does not promote active engagement or long-term learning. Users rely on it passively, consulting only when necessary, resulting in minimal spatial retention. In contrast, our approach emphasizes gamification and active learning to help users internalise spatial knowledge more effectively.

B. Gamification

To inform our approach to gamification, we looked at three key examples of popular web-based games with educational value.

1) **GeoGuessr**[3]: *GeoGuessr* is a web-based multiplayer game in which users compete to find various geographical locations around the world. Users are provided with a

[2]https://maps.auckland.ac.nz/auckland/venues
[3]https://www.geoguessr.com/

Google Street View[4] location to explore, and must guess the area they are in before the timer is up via a pin on a world map. We drew inspiration from the application's educational value in teaching geographical knowledge, and its use of multiplayer gamification features such as a leaderboard and score tracking system. Similarly, our application uses visual cues, albeit in the form of campus locations, and incorporates scoring and competitive elements such as leaderboards to drive engagement and learning.

2) **TimeGuessr**[5]: *TimeGuessr* is a variation of the popular *GeoGuessr* web application that differs by challenging users to provide both the approximate timeframe and geographical location of an image. From this related work we primarily borrowed inspirations for input and user interface design, such as image display and handling user flow.
3) **Wordle**[6]: *Wordle* is a web-based game with a guessing mechanism involving six guesses for a five-letter word. Players input different letters and are provided with hints as to whether the letter is correct, in the wrong placement, or not in the word at all. We opted to extend this guessing mechanism to university room codes as opposed to letters in a word, and took inspiration from the usage of limited attempts and feedback hints.

By drawing from existing gamified systems and addressing the limitations of alternative campus navigation tools, we aimed to design an experience that is both educational and enjoyable.

III. Design

This section outlines the design process for the *CampusGuessr* web application. Our design process focused on three key areas: the user interface and experience (UI/UX), the architecture of the web application, and the selection of our desired tech stack. Each of these areas involved numerous discussions and refinement to ensure agreement in the team.

A. UI/UX Design

The design of the *CampusGuessr* user interface was collaboratively developed using the Figma[7] design tool. Our Figma board was organised into multiple subsections, which included component mockups and visual references (figure 1). This structure helped ensure clarity and alignment across our team throughout the design process. Although the entire team contributed, the overall process was led by the corresponding domain owner of the user interface, as discussed in Section 6 (Methodology).

Inspired by *Wordle*, we designed a 6-character fixed input system for our game. Initially, we designed a flexible input system to accommodate various room code formats to support all realistic university locations. However, to maintain a more consistent and intuitive player experience, and due to the commonality of 6-figure room codes at the University of Auckland, we shifted to a fixed format for consistency and simplicity. Our input boxes mimic Wordle's style by using colour-coded feedback, such as green for correct, yellow for partially correct, and grey for incorrect. We believed that this design approach was a simple, intuitive method to take single-character inputs and provide understandable hints. Furthermore, by using a familiar input mechanism, our game would be easier to learn and play, look more visually appealing, and reduce cognitive load. However, this simplification did come at the cost of limiting support for some multi-character inputs.

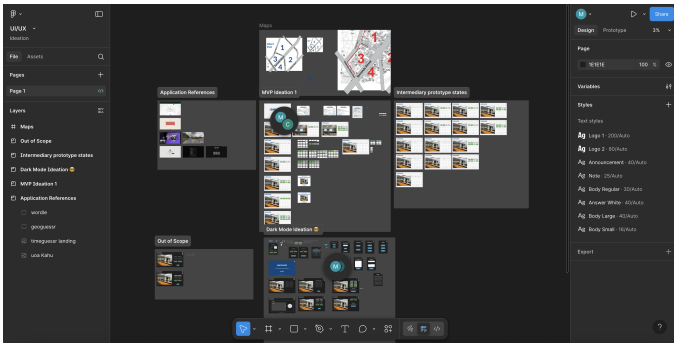

Fig. 1. Figma Design board for UI/UX ideation, consisting of numerous subsections with their own components and frames.

Our minimalist layout came as inspiration from the page design of *TimeGuessr*, which features a prominent logo, clean layouts, and clear visual cues. This was chosen over the *GeoGuessr* designs to minimise any potential cognitive load by only providing necessary information to new users.

To match the identity of the University of Auckland, we selected key colours for our design palette based on the official University of Auckland *Campus Maps* website. This helped us create a more cohesive connection to the campus environment. All of these application references for our design can be found in figure 2. Our final version of the landing and game page designs can be found in figures 3 and 4.

B. Architecture

The architecture of *CampusGuessr* is based on a modular client-server model with three key components: the frontend interface, backend actions and services, and database. Our user interface was developed using React and TypeScript. It consists of multiple interactive pages, including the landing page, game page, user authentication, account page, leaderboard, and help modal. React components are used to manage the visual layout and game state. Server-side actions manage communication between the frontend and backend, handling tasks such as retrieving user data or recording game progress. Service modules interact directly with MongoDB, performing CRUD operations on users, rooms, and score records. Our architecture diagram can be found in figure 5.

C. Tech Stack

Our chosen technology stack extends upon the traditional MERN (MongoDB, Express, React, Node.js) stack. Each tech-

[4]https://www.google.com/streetview/
[5]https://timeguessr.com/
[6]https://www.nytimes.com/games/wordle/index.html
[7]urlhttps://www.figma.com/

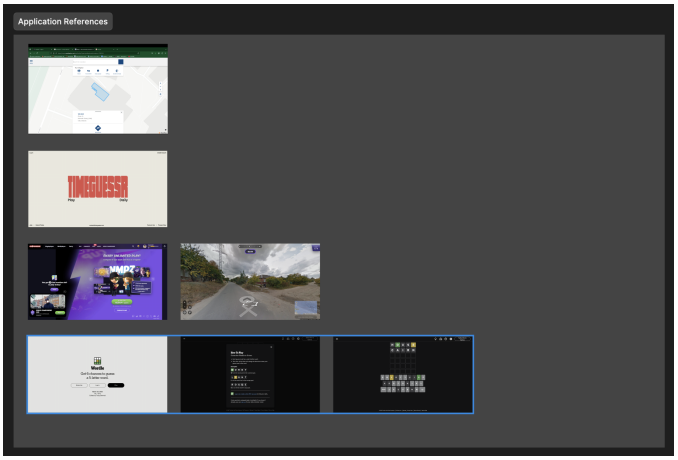

Fig. 2. Inspirations taken from the University of Auckland, TimeGuessr, GeoGuessr, and Wordle for use in our own design, displayed on our Figma board.

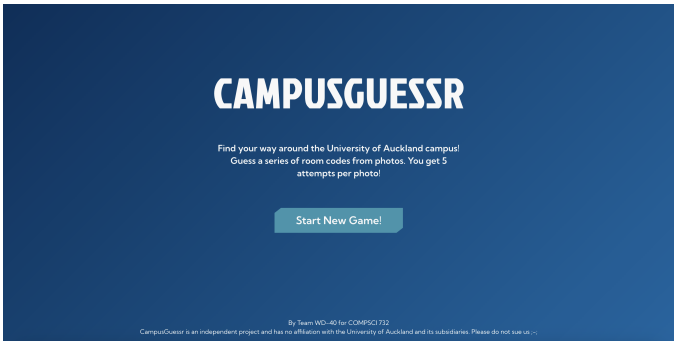

Fig. 3. Final design for CampusGuessr's landing page.

nology was chosen to enhance our developmental efficiency, productivity, scalability and cost-effectiveness.

For our frontend, we used the following:

- **Next.js**: We used Next.js as the foundation for both our frontend and backend development due to its built-in support for rendering and routing, which greatly streamlined our development process. Additionally, Next.js provides image optimisation, which was especially beneficial given our image-heavy gameplay interface.

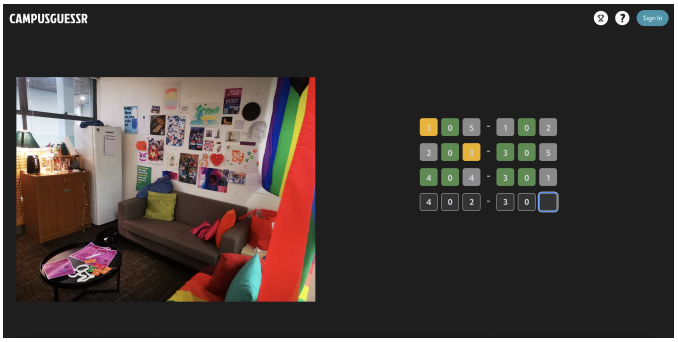

Fig. 4. Final design for CampusGuessr's game page.

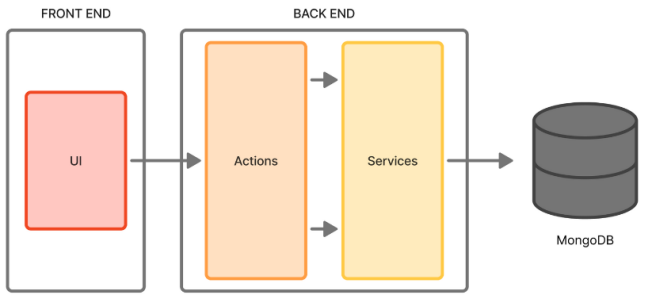

Fig. 5. The CampusGuessr architecture diagram, demonstrating interactions between the frontend, backend, and database.

- **React**: We used React as the core UI library to help us build our modular and reusable UI components. This allowed us to better structure our interface.
- **TypeScript**: TypeScript provided static type checking improved our productivity through better code readability and maintainability.
- **TailwindCSS v4**: This utility-first CSS framework allowed us to rapidly build and apply consistent styling across all our components.

For our backend, database, deployment and testing, we used:

- **MongoDB**: A NoSQL database, we chose MongoDB to help us store user and room information. The document-oriented storage and flexible schema made this database a good fit for our data models, especially as they tended to evolve as the project continued.
- **Cloudflare R2**: Cloudflare allowed us to store objects such as gameplay images. Storing images within the primary MongoDB database would be costly, and as such we selected Cloudflare R2 as a cost-effective solution that allowed us to use 10GB of storage per month. For integration, we stored links to images in Cloudflare R2.
- **Fly.io**: We deployed our application on Fly.io, which allows any potential user to access our application on the web. Fly was specifically selected due cost-effectiveness and prior experience with using the technology.
- **Vitest**: For unit testing, we used Vitest, which enabled us to write tests to validate key logic functions and ensure correctness in data-handling.

IV. Implementation

CampusGuessr was implemented as a full-stack web application with a clear separation between the frontend interface, backend services, and data storage. Our development process was driven by our aforementioned design prototypes and informed by testing. During planning, we also established an initial Minimum Viable Product (MVP) which outlined the foundational goals and core components of the application as shown in figure 6. While the 'must-haves' section of our MVP served as our baseline, we significantly extended it during development by incorporating additional features, as outlined below.

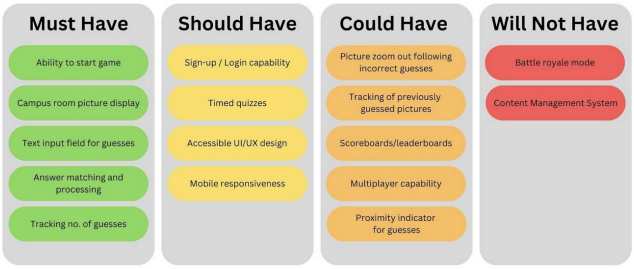

Fig. 6. Our initial scope for the project, in which our minimum viable product encompasses the 'must-have' column.

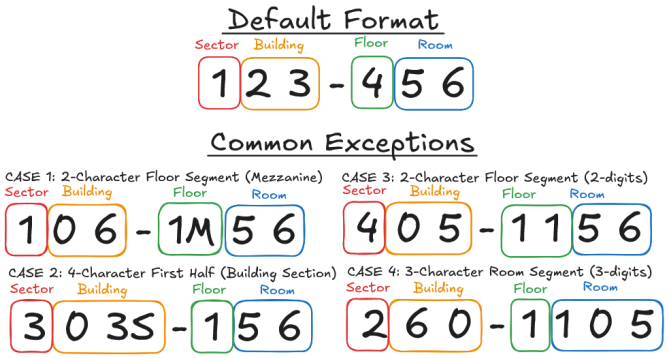

Fig. 7. Default and alternative formats for room codes at the University of Auckland.

A. Frontend Features

- **Landing Page**: A responsive welcome page introducing the game, including a logo and start button.
- **Game Page**: The main gameplay interface, which displays a location image and various input fields for users to guess the sector, building, floor and room of a given area.
- **Input Fields**: Our input system provides feedback based on the correctness of the user's input. Initially, the MVP supported only green (correct) and grey (incorrect) indicators, but we extended this functionality to provide a yellow hint input display for guesses one step away from the correct answer. We also explored support for alternative room code formats as shown in figure 7, including multi-digit fields such as for building '303S'. Unfortunately, due to integration issues, functionality for these cases was excluded from the final deployment.
- **Profile Page**: The profile page displays information about the user, including their win rate, profile picture, and number of games played.
- **Leaderboard**: The leaderboard displays the top 15 users ranked by score in descending order.
- **Result Modals**: After each round, a result modal displays the user's performance, including their score, the correct room code, and a button to start the next round.

B. Backend Features and Data Handling

- **Authentication**: Implemented via Google OAuth, leveraging existing university email accounts to ensure a secure login.
- **Game Storing**: Game progress is saved and resumed through API endpoints that track current attempts, so that if the user's session is interrupted, they can come back to their previous game.
- **Scoring**: Scores are computed and stored after each round, with routes for retrieving score data such as a leaderboard score tally.
- **Image, Room, and User Data Storage**: Room images are uploaded to Cloudflare R2 and associated metadata is stored in MongoDB. User data, such as per-user game statistics like win/loss records, are also stored in MongoDB.

C. Personal Contributions

Prior to this project, my experience was primarily in frontend web development. Working on backend services and database integration presented a new and rewarding challenge. My primary responsibilities focused on UI/UX development, with additional contributions to backend functionality. Specifically, my contributions included:

- **Scoring System**: I implemented the game scoring logic within the `calculateScore.ts` utility. The scoring system provides scores based on accuracy and speed by awarding a 180 point maximum per round. For each incorrectly guessed input character, 5 points are deducted. This ensures that quick and accurate guesses gain the highest reward. I also built the corresponding action and service to store user scores in the database, and integrated these scores into the result modals for display at the end of each round.
- **Post-Game Navigation**: I implemented a conditional feature on the game page that allows player to proceed to a new round directly after finishing a game should they accidentally close the results modal.
- **Landing Page**: I assisted with the design and implementation of the Landing Page, and made it responsive to ensure compatibility across different screen sizes and devices.
- **Profile Page**: I connected the profile page to the backend, and in doing so, modified the database schema to store the number of games won and the number of games lost by a given user for progress display purposes. I also helped implement a sign-out button at the bottom of the profile page.
- **Leaderboard**: I initially designed the leaderboard interface and contributed to its implementation by creating a service and action to retrieve the 15 top-scoring users from the database.

This project gave me the opportunity to step beyond my comfort zone in frontend development and explore full-stack development, which was a valuable learning experience.

V. Testing

Our testing strategy primarily used Vitest, which is a Vite-native unit testing framework. Given the nature of our requirements, which tended to rapidly evolve as the project continued, our efforts focused primarily on the critical components of our application. For example, in our testing directory, we conducted unit tests on the "uploadToMongo" function to ensure that we could reliably store room data in the database, and on "getRandomRoom" to check that the website would properly retrieve and render new rooms. We also performed UI tests, including checking button-redirection and rendering. We attempted to write tests for every service and component, but were unable to complete this due to time constraints.

VI. Methodology

A. Project Management

We adopted a domain ownership approach to project management, in which each member of the team was assigned responsibility for a specific domain of full-stack development. The idea of this approach is to avoid unnecessary time being spent on decision making and planning by having a single person make the final decisions and calls on a given domain if needed. Each individual can still work on other domains, but should defer to that given domain owner for leadership and guidance when necessary. For this project, we had the following domains:

- **Project Management**
- **Data Management**
- **User Interface**
- **Dev/Ops**
- **Quality Assurance**
- **Business Logic**
- **Integration**

I served as the User Interface domain lead, in which I was responsible for the UI/UX design process and frontend development. In this role, I led the implementation of core interface components and provided guidance on visual and interactive design decisions. I also resolved frontend-related questions and conflicts raised by the team where necessary.

B. Workflow

Our workflow for this project followed an Agile methodology structured around two-week sprints. We conducted regular weekly meetings to maintain consistent communication, review sprint progress, and allocate or re-allocate tasks, allowing us to adjust for any changes in our development plan as they occurred. For example, by having these mid-sprint meetings, if an individual realised that they did not have enough time or ability to complete all of their tasks for the sprint, we could quickly and effectively re-allocate tasks or provide the necessary support before the expected due date for that task.

Our codebase was managed through use of the Github Flow strategy, in which we created a new branch from the main branch for each new feature we wanted to implement. Completion of a feature would result in a pull request being opened, which would then be reviewed by a minimum of two reviewers for approval before merging back into the main branch. Pull requests were to be structured to include clear summaries of the feature and instructions for effective review to streamline the review process. These pull requests were typically linked to a specific Github Issue, which would be closed following pull request approval, enabling a clear allocation of tasks between team members. This process facilitated efficient team collaboration, ensured that all features were thoroughly reviewed, and maintained a stable main branch.

VII. Tooling

In order to maintain momentum throughout development and stay organised, we used the following tools:

- **Discord** was our primary platform for communication throughout the project. We set up a dedicated server with multiple channels to track bugs, coordinate reviews, and make quick decisions. This structure enabled asynchronous collaboration and ensured all of our members were kept in sync despite differing schedules. An example of our channels can be found in figure 8.
- **Github** served as our version control system. We made extensive use of **Issues** to track bugs and features, and **Actions** to manage deployment workflows.
- **Google Docs** served as a platform to record our meeting minutes, keep track of sprint planning notes, and outline shared design and development decisions.

To maintain high-quality and consistent code, we also integrated several tools into our development environment:

- **ESLint**, which helped enforce consistent code standards and catch errors early.
- **Prettier**, which helped ensure consistent formatting for stylistic consistency.

Personally, I also made use of AI-assisted tools during my development process:

- **ChatGPT** was primarily used to debug unexpected behaviour and brainstorm implementation strategies. However, given its tendency to hallucinate, I mostly avoided relying on it for full code generation and would carefully review any suggestions.
- **Github CoPilot** assisted in generating boilerplate code and minor refactoring, streamlining development tasks.

Overall, the above tools greatly enhanced our productivity and organisation, especially given our time constraints.

VIII. Discussion

In general, we have successfully achieved most of our goals for this project. In fact, we managed to implement more features than we initially believed we would accomplish in our MVP. Our initial goals focused on building a basic guessing game with a limited set of locations and single-player functionality. We extended beyond this baseline to include features such as multiplayer with Google OAuth authentication, a profile page, game saving, a robust scoring system, a leaderboard, and a hint system. Our final product delivers

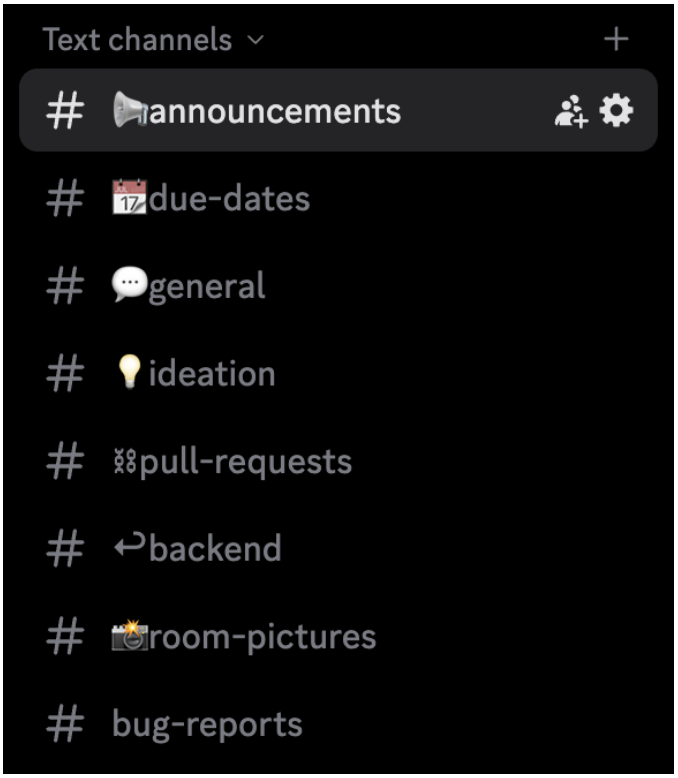

Fig. 8. The servers on our Discord channel, used for organising our communication for development. Note that there are separate channels for different issues, including one to keep track of pull requests, and another for reporting bugs.

a cohesive and engaging learning experience that effectively accomplishes its purpose as a gamified web application for campus locations.

That said, there were still some key areas where our project could have been improved:

- **Testing**: One of the main shortcomings of our project was the lack of comprehensive testing as a result of poor time management. While we tested some of the most important features, we did not implement enough thorough testing. This became apparent during our final merges, where bugs occurred late due to insufficient test coverage. In the future, we could benefit from earlier and more systematic testing.
- **Documentation**: Our GitHub repository documentation could have been better organised and more robust. In particular, the README lacked detailed setup instructions for configuring environment variables and running the development server, and we did not provide access to the deployed server.
- **Task Duplication and Confusion**: In the initial stages of our project, we encountered some inefficiencies due to overlapping work or unclear expectations on implementing certain features. Some team members occasionally began working on the same or similar features due a lack of an established tracking system, leading to some confusion. We mitigated this in later sprints by pushing for consistent use of GitHub Issues and tracking tasks on our sprint planning documentation in Google Docs. However, we could have benefited from being much more clear about our expectations for each feature.
- **Time Management**: Our biggest challenge as a team was our time management, as every member of the team had a large number of commitments outside of this project. As a result, we struggled to maintain a consistent development pace, and we tended to underestimate how long it would take to implement or debug some of our features, which led to a large workload being pushed towards the end of the timeline. This resulted in some rushed debugging and implementation. Although our communication and teamwork were strong, having clearer task breakdowns, more realistic scheduling, and earlier testing would have greatly improved this process.

IX. Conclusion

Overall, *CampusGuessr* was a rewarding project that allowed us to explore full-stack web development while experiencing real-world design and technical challenges. Despite being initially unsure of how much we could achieve within the project timeframe, we ultimately surpassed our goals and developed a fun, functional and visually engaging game experience.

Personally, this was my first significant experience working with backend systems and databases, which proved to be both challenging and highly valuable. Tasks such as integrating the scoring system, updating database schemas, and building full data flows between the frontend and backend taught me valuable technical skills and pushed me well outside my comfort zone. After this experience, I feel much more confident in my technical skills and capabilities as a developer.

As a team, I believe we all demonstrated a strong level of passion, collaboration, and creative problem-solving for this project. We've learned the importance of clear communication, early testing, maintaining documentation, and realistic planning, which will stick with us into future technical work.

There are quite a few avenues for future work to extend upon and improve *CampusGuessr* as it stands. First, we could improve input formatting to support a wider range of room code formats, such as multi-character components. This would make the game much more flexible and reflective of real campus locations. Additionally, we could better enhance the mobile experience. The game is technically mobile compatible and responsive to smaller screens, but the input mechanism is not as streamlined as it is on desktop format and could be improved. Furthermore, we could extend our current features to include more gamification, such as having timed gameplay or streaks. Finally, it would be interesting to conduct a more research-based investigation into this project, to test whether the application actively helps new university students learn locations and navigate around campus. This could be accomplished by having one set of students only use campus maps, while the second group supplements their usual university

resources with *CampusGuessr*, and comparing their ability to identify and find locations on campus.

Overall, this project was a valuable learning experience that allowed us to apply theory to a real-world application in a collaborative setting. If refined and tested further, *CampusGuessr* could be used as part of university orientation events or be integrated into student support platforms, offering a fun and engaging way to help students familiarise themselves with the University of Auckland campus environments.

REFERENCES

[1] M. Krause, M. Mogalle, H. Pohl, and J. J. Williams, "A Playful Game Changer," Proceedings of the Second (2015) ACM Conference on Learning @ Scale - L@S '15, 2015, doi: https://doi.org/10.1145/2724660.2724665.

[2] O. Noran, "On gamification in action learning," Proceedings of the Australasian Computer Science Week Multiconference on - ACSW '16, 2016, doi: https://doi.org/10.1145/2843043.2843344.