# Development Report

## Tech Consultancy Web Application with Chatbot Integration

Divya Lamichhane
Yoobee College of Creative Innovation

July 10, 2025

# Contents

# List of Figures

# 1 Introduction

WhanauTech a web application is focused on the Tech consultation aiming to support the local and Maori Communities. The application enables users to book appointments with tech advisors, asking some queries via a chatbot, and receive guidance on technology, cybersecurity, cloud solutions and many more.

## 1.1 Technologies Used

- React (for frontend)
- Flask (for backend)
- SQlite
- TogetherAI Api for chatbot integration
- GitHub
- Terminal

# 2 Team Formation

Although the project was originally intended to be completed in a team setting, I encountered challenges in forming a group due to language barriers. After discussing the situation with my supervisor, Dr. Mohammad Norouzifard, it was agreed that I would continue the development as a solo project.

Throughout the project, I maintained regular communication with Dr. Norouzifard, who provided valuable guidance on both technical implementation and culturally appropriate design. His feedback was instrumental in shaping a solution that not only meets functional requirements but also aligns with the needs of local and Māori communities.

In keeping with the principles of **Te Tiriti o Waitangi**:

- **Partnership:** I collaborated closely with my supervisor and remained mindful of potential Māori end users throughout the planning and development process.
- **Participation:** I incorporated features that support Māori participation, such as bilingual chatbot functionality and culturally inclusive UX considerations.
- **Protection:** I ensured that the digital application respected Māori language, values, and information sovereignty through careful design and feature implementation.

Despite working independently, I approached the project with a strong commitment to equity, inclusiveness, and cultural responsiveness, aligning closely with the principles of Te Tiriti o Waitangi.

# 3 Project Kickoff

The project officially commenced following a consultation meeting with my supervisor, Dr. Mohammed Nourifard. In the absence of a team and client-facing stakeholder, this

meeting served as the project's kickoff session, where we clarified the project's scope, deliverables, and alignment with both technical and cultural goals.

Key goals established during the kickoff included:

- Developing a web-based platform for tech consultancy services
- Building a user-friendly booking system for connecting with tech advisors
- Integrating a bilingual chatbot to assist users in both English and te reo Māori

The principles of **Te Tiriti o Waitangi** were discussed and embedded into the project plan:

- **Partnership:** Establishing an ongoing advisory relationship with my supervisor to review cultural aspects of the application.
- **Participation:** Ensuring the application's features support access and usability for Māori users.
- **Protection:** Safeguarding te reo Māori and cultural relevance, particularly within the chatbot design and overall user interface.

Through this meeting, a foundational understanding of the project vision was formed, balancing the technical requirements with cultural sensitivity and inclusiveness. This served as a guiding reference throughout the development process.

# 4 Requirement Gathering

The requirement gathering phase began with a brainstorming session where I generated ideas for the features and functionalities of the web application. This reflective process helped identify key needs such as user appointment booking, chatbot support, and bilingual translation capabilities.

The requirement gathering phase began with a brainstorming session where I generated ideas for the features and functionalities of the web application (see Figure 1).

Following brainstorming, I worked closely with my supervisor to refine and clarify these ideas into actionable requirements.

Key functional requirements identified include:

- An online booking system allowing users to schedule appointments with technology advisors.
- A chatbot integrated to provide immediate responses and support.
- Bilingual support, enabling users to interact with the chatbot and interface in both English and te reo Māori through translation features.
- Responsive design to ensure usability across different devices, including mobile phones and tablets.

The requirements were documented clearly to guide development and ensure that the application is user-friendly, reliable, and meets the client's expectations.

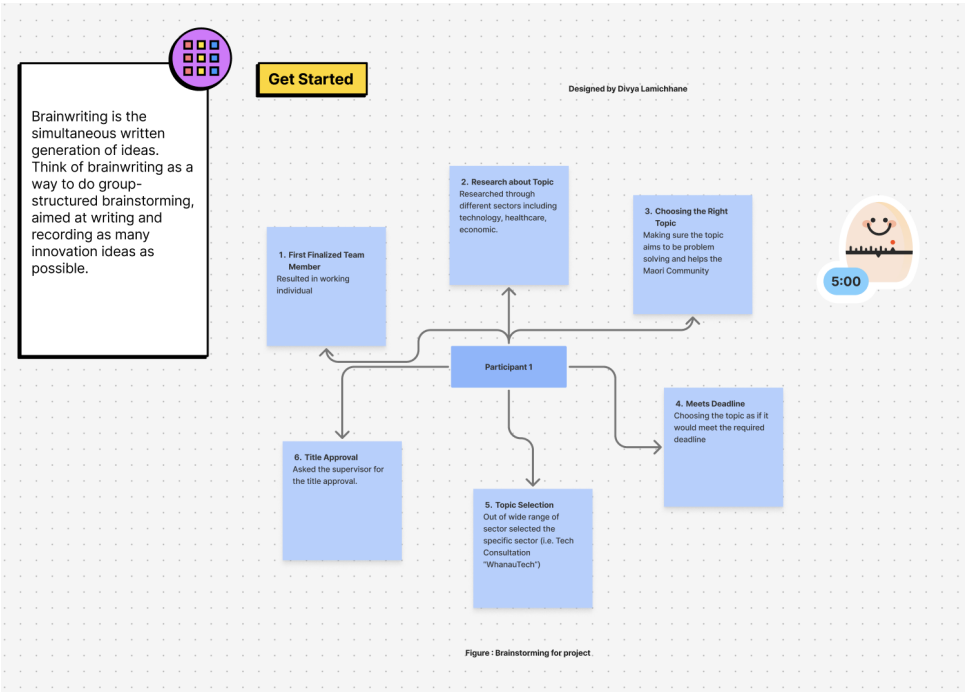


Figure 1: Brainstorming mind map for feature ideas

The translation feature was specifically prioritized to enhance accessibility and support users who prefer to engage in te reo Māori, recognizing the importance of language inclusivity as a value-added feature.

This phase laid a solid foundation for the subsequent design, development, and testing of the application.

# 5 Requirement Analysis and Prioritization

After gathering the project requirements, I analyzed each feature based on its business value, technical feasibility, and potential impact on users. The goal was to prioritize features that deliver the most benefit to the community while ensuring smooth project delivery within the available timeframe.

To organize and prioritize the requirements, I applied the MoSCoW method, categorizing features into:

- **Must Have:** Core functionalities essential for the app's operation, including the booking system, chatbot integration, and bilingual support.
- **Should Have:** Important features that enhance user experience, such as email confirmations and an admin dashboard.
- **Could Have:** Additional features like user feedback forms and calendar views, which could improve usability but were not critical for the initial release.
- **Won't Have (for now):** Features like advanced AI features or extra integrations deferred for future development due to scope or time constraints.

This prioritization ensured that the application delivered immediate value to users, particularly by focusing on accessible consultation booking and bilingual chatbot assistance, which support community inclusivity.

The clear definition of acceptance criteria for each prioritized feature guided the development process and testing, ensuring alignment with both user needs and project goals.

# 6 Design

Following requirement analysis, the design phase focused on creating wireframes and planning the system architecture to ensure a user-friendly and scalable application.

A wireframe was developed to visualize the booking interface and chatbot integration, prioritizing simplicity and accessibility across devices (see Figure 2).

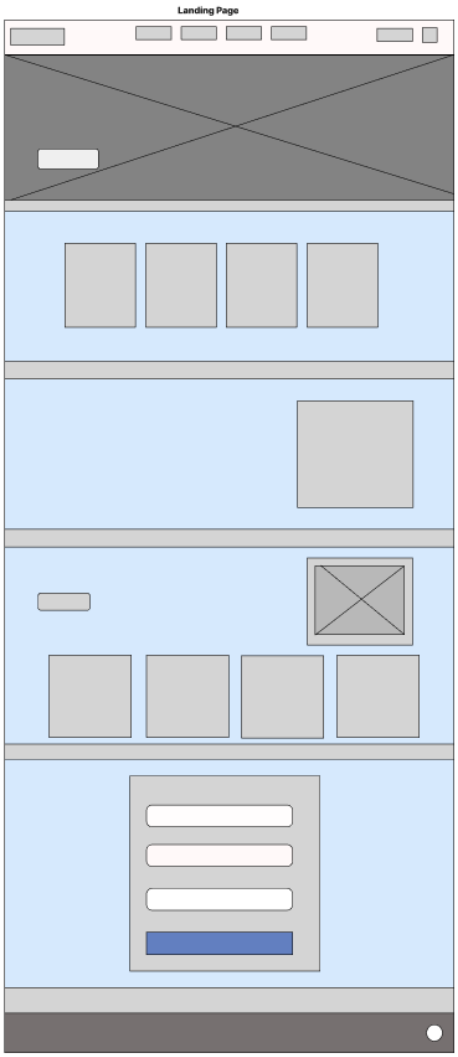


Figure 2: Wireframe of the landing page

The system architecture was designed with a React frontend communicating via RESTful APIs to a Flask backend. The chatbot was integrated using TogetherAI's API to support

bilingual interactions.

Key design considerations included:

- Responsive design for mobile and desktop users.
- Modular React components for ease of maintenance.
- Secure API endpoints with input validation.
- Support for bilingual chatbot responses and UI translations.

# 7 Implementation

The implementation phase involved developing the frontend, backend, and chatbot features based on the design specifications.

**Frontend:** Developed in React, the UI includes components for advisor listings, booking forms, chatbot interface, and service listings. React hooks and state management were used to handle user interactions and data flow.

**Backend:** The Flask API manages booking data. It connects to a database to store booking details and user information securely.

**Chatbot Integration:** The chatbot leverages TogetherAI's API, configured to process both English and te reo Māori inputs. It provides immediate responses and guides users toward booking appointments for advanced inquiries.

Throughout implementation, version control with Git and continuous testing ensured code quality and stability. Challenges included managing asynchronous chatbot responses and implementing bilingual support, which were resolved through iterative testing and refinement.

The final product meets the project requirements, offering a responsive, accessible, and culturally aware tech consultancy platform.

# 8 Agile Development Approach

## 8.1 Sprint Planning and Timeline

To efficiently manage the project, I created a Gantt chart that outlines sprint schedules and major milestones. The development was organized across the following phases, as visualized in the Gantt chart (see Figure 3):

- **Planning:** Initial brainstorming and finalizing the project topic were followed by requirement gathering. This laid the foundation for the system's purpose, target users, and functionality.
- **Designing:** Wireframes were created to visually map out the core structure of the application. A basic dashboard layout was developed as a starting point.
- **Frontend Development:** This phase focused on refining the wireframes by adding pages such as booking, login, and chatbot interface. I also began implementing

frontend components for the chatbot feature and planned how to integrate OpenAI's API for multilingual interaction.

- **Backend Development:** The backend involved integrating APIs, initially experimenting with TogetherAI for chatbot logic. Data fetching, advisor listings, and booking form submissions were implemented using Flask RESTful endpoints.
- **Testing and Deployment:** The application was tested using `pytest` to ensure backend logic and API responses functioned correctly. Deployment was carried out using Render, making the application accessible online.

This sprint-based planning approach allowed for iterative development, continuous feedback, and regular progress tracking. Each sprint milestone helped ensure that the application evolved from basic structure to a functional, bilingual web application.

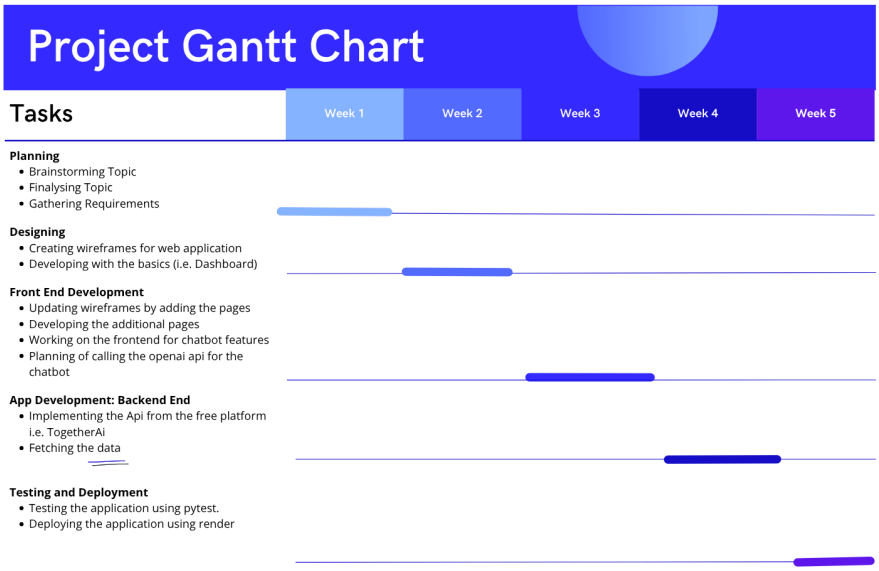


Figure 3: Gantt Chart Showing Project Timeline and Sprint Breakdown

## 8.2 Work Breakdown Structure (WBS)

To organize tasks and technical deliverables, I developed a Work Breakdown Structure:

Figure 4: Work Breakdown Structure

Describe sprint planning, tools used (GitHub, Trello, etc.), and how agile was applied in a culturally informed way.

## 8.3 Product Backlog

To support agile development, I created a product backlog containing prioritized user stories and technical tasks. Each item was assigned a priority level and mapped to a sprint. This backlog helped guide sprint planning, progress tracking, and iterative delivery of features.

| User Story / Task | Priority | Status | Sprint |
|---|---|---|---|
| As a user, I want to book an appointment with a tech advisor. | High | Done | Sprint 2 |
| As a user, I want to interact with a bilingual chatbot for support. | High | In Progress | Sprint 3 |
| Design landing page and create wireframes. | High | Done | Sprint 1 |
| Implement chatbot API using TogetherAI/OpenAI. | High | In Progress | Sprint 3 |
| Test and deploy application on Render. | High | Planned | Sprint 4 |
| Add email confirmations after booking. | Medium | Planned | Sprint 4 |
| Enable user feedback submission form. | Low | Not Started | Future |

Table 1: Product Backlog for WhānauTech Application

## 8.4 Sprint Retrospective

At the end of each sprint, I reflected on the outcomes, challenges, and areas for improvement. This process helped me continuously adapt my workflow and prioritise tasks based on technical complexity and time constraints.

- **What went well:**
  - Maintaining a clear sprint plan using the Gantt chart and backlog kept the project on track.
  - Core functionalities like the booking system and chatbot interface were implemented successfully within the planned sprints.
  - Frequent testing helped catch and resolve issues early.
- **What could be improved:**
  - Time estimation for some tasks (e.g., chatbot integration) was too optimistic.
  - Some tasks had to be pushed to later sprints due to unexpected complexity.
  - Working alone meant that I had to balance all roles, which slowed some areas of progress.
- **What I will do differently:**
  - Break down complex tasks into smaller, more manageable subtasks during backlog creation.
  - Allocate more buffer time for research-intensive features like API integrations.

  - Consider forming a collaborative team or seeking peer feedback earlier in future projects.

This reflection loop allowed me to stay focused, adjust quickly, and finish the project with improved confidence in agile practices and solo development.

# 9 Cost Estimation

Although this was an individual academic project, I conducted a basic cost estimation exercise to simulate real-world development planning. The cost estimation includes time invested, potential labour cost, and the value of tools and platforms used during development.

## 9.1 Estimated Hours and Labour Cost

The following breakdown represents the approximate time spent on each development phase:

- **Planning and Research:** 10 hours
- **Requirement Gathering and Brainstorming:** 8 hours
- **Wireframing and Design:** 6 hours
- **Frontend Development (React):** 20 hours
- **Backend Development (Flask APIs):** 20 hours
- **Chatbot Integration:** 10 hours
- **Testing and Debugging:** 10 hours
- **Deployment and Documentation:** 6 hours

**Total estimated hours:** 90 hours

Assuming an industry-average developer rate of NZ$40 per hour:

$$EstimatedLabourCost = 90hrs \times 40 = \textbf{NZ\$3,600}$$

## 9.2 Tools and Resources Used

Most of the tools and platforms used were open-source or free for educational use:

- **Frontend:** React (free)
- **Backend:** Flask + Python (free)
- **Deployment:** Render (free tier)
- **Version Control:** Git and GitHub (free)
- **Design Tools:** draw.io / Figma (free)
- **Chatbot API:** OpenAI / TogetherAI (trial/free API credits)

### 9.3 Summary

If developed commercially, the project would require an estimated budget of NZ$3,600+ depending on labour and any additional premium APIs or services. However, through strategic use of free and open-source tools, the actual development was achieved at no financial cost beyond time investment..

## 10 Acceptance Criteria and Testing

To ensure the application was functional, reliable, and aligned with user expectations, a combination of manual testing and automated testing was used throughout the development process. Each feature was developed with clear acceptance criteria to guide implementation and validation.

### 10.1 Testing Approach

- **Unit Testing:** I used `pytest` to write and run unit tests on the backend Flask routes, validating API responses, data formats, and booking logic. This ensured the server returned appropriate status codes and handled edge cases correctly (e.g., invalid booking times, missing input fields).
- **Manual Testing:** Frontend components were tested manually using test scenarios and checklists. User flows — such as booking an appointment, interacting with the chatbot, or navigating the dashboard — were validated across browsers (Chrome, Firefox) and screen sizes to ensure responsive behavior.
- **Integration Testing:** I manually tested the communication between the frontend and backend. This included checking that form submissions correctly triggered Flask APIs, that responses were handled by React components, and that data (like bookings) were stored and retrieved as expected.
- **Chatbot Testing:** The chatbot was tested using both English and te reo Māori inputs. It was validated for basic intent recognition, appropriate responses, and correct fallback suggestions when advanced questions were asked (e.g., prompting users to book a tech advisor).

### 10.2 Acceptance Criteria Definition

Each user-facing feature was associated with specific acceptance criteria. For example:

- **Booking System:**
  - Users can select a date, time, and advisor from the frontend.
  - Booking is stored in the backend and visible in the admin panel.
  - Duplicate or conflicting times are not allowed.
- **Chatbot:**
  - Responds instantly to user queries in English and te reo Māori.
  - Provides relevant tech support responses or booking guidance.

  - Uses OpenAI/TogetherAI integration without frontend delay.
- **Mobile Responsiveness:**
  - All pages render cleanly on mobile devices.
  - Inputs and buttons are accessible and clickable at small screen widths.

### 10.3 Ensuring Community Fit

Although this was an individual project, basic community needs were considered in feature design:

- Bilingual chatbot responses provided inclusivity for te reo Māori speakers.
- Simple UI with clear navigation ensured accessibility for users with limited digital skills.
- Feature scope focused on practical tech support for common user needs (booking, advice).

Overall, continuous validation and testing ensured that the final application met both technical goals and basic user accessibility expectations.

## 11 Reflection and Lessons Learned

### 11.1 Technical Learning and Skill Development

Throughout this project, I significantly improved my technical capabilities in full-stack web development. I deepened my understanding of:

- React for building responsive, component-based frontends.
- Flask for creating secure and scalable backend APIs.
- RESTful communication between frontend and backend services.
- Integration of third-party APIs, including OpenAI/TogetherAI for chatbot functionality.
- Unit testing using `pytest`, and deploying full-stack apps using Render.

I also learned to manage asynchronous data flows, state management in React, and form validation — all crucial for building user-friendly web applications.

### 11.2 Challenges and Problem Solving

One major challenge was working on this project solo due to language barriers and the difficulty of forming a diverse team. While it meant managing every role (designer, developer, tester, project manager), it also helped me develop strong self-discipline, time management, and adaptability.

Other technical challenges included:

- Integrating a bilingual chatbot using third-party APIs while ensuring performance and accuracy.
- Managing API calls and error handling gracefully on the frontend.
- Ensuring responsive design across all devices.

To overcome these, I consulted documentation, experimented through trial-and-error, and sought support from my supervisor where needed.

## 11.3 Teamwork and Self-Management

Although this was an individual effort, I treated the project with the discipline of a team-based agile environment. I used sprint planning, version control with Git, and consistent documentation to simulate professional development practices.

This experience taught me the value of clear planning, realistic time management, and iterative development. It also highlighted the importance of maintaining motivation and accountability without a team.

## 11.4 Cultural Considerations and Te Tiriti o Waitangi

As this application is designed to support users from diverse communities, I aimed to embed inclusivity by integrating bilingual (English and te reo Māori) support in the chatbot and interface. While I did not work directly with Māori team members, I took steps to reflect partnership principles by:

- Recognizing language as a cultural taonga (treasure) and enabling Māori language use in the application.
- Prioritizing clear, accessible design to ensure equitable access to digital services.
- Seeking supervisor feedback to ensure the solution was appropriate and respectful in its cultural presentation.

I acknowledge that there is room for deeper integration of Te Tiriti principles in future iterations — such as through direct consultation with Māori advisors or community users — but this project provided a strong foundational step in inclusive, community-aware development.

## 11.5 Ongoing Learning and Future Improvements

This project has inspired me to continue learning about:

- Advanced backend features such as authentication, role-based access, and database optimization.
- AI-driven user interfaces with more personalized responses.
- Better integration of Te Tiriti o Waitangi principles through co-design with community members.

In future work, I would also aim to form a more diverse team and implement continuous user feedback to improve both usability and cultural fit.

# 12 Conclusion

This project resulted in the successful development of a fully functional web-based tech consultancy application that allows users to book appointments, receive guidance via a bilingual chatbot, and access support services efficiently. Built using React for the frontend and Flask for the backend, the system integrates modern web technologies with a culturally inclusive design approach.

Despite working independently, I adopted agile development practices including sprint planning, version control, and iterative testing. The challenges of solo development, such as time management and full-stack responsibility, were met with a proactive mindset and support from my supervisor.

The incorporation of te reo Māori in the chatbot demonstrates a first step toward honoring Te Tiriti o Waitangi through inclusive digital services. While the project remains a prototype, it lays the foundation for future enhancements such as deeper Māori engagement, more advanced chatbot capabilities, and real-world deployment for community use.

This experience has strengthened my technical skills, problem-solving abilities, and awareness of the cultural responsibilities of software developers in Aotearoa New Zealand.

# References

Add links to your GitHub repo, documentation, or any other sources used.

# Appendix A: Wireframes

# Appendix B: UML Diagrams


Figure 5: Wireframe of the Landing Page

booking_wireframe.png

Figure 6: Wireframe of the Booking Page


Figure 7: Wireframe of Chatbot Interface


Figure 8: Use Case Diagram


Figure 9: Class Diagram


Figure 10: Activity Diagram - Booking Flow