B.Sc. (Hons) in Software Development

Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

LANGUAGE LEARNING WITH AI

By
CIAN MARTYN

April 27, 2025

Minor Dissertation

Department of Computer Science & Applied Physics,
School of Science & Computing,
Atlantic Technological University (ATU), Galway.

Contents

Chapter 1

Introduction

1.1 Context and Motivation

In the 21st century, the ability to communicate across languages is not just a valuable skill-it is a necessity. As globalisation accelerates, people are increasingly required to interact with others from diverse linguistic and cultural backgrounds, whether for travel, business, education, or personal growth. Despite the proliferation of language learning resources, many learners still struggle to achieve conversational fluency and practical competence. Traditional classroom methods, while effective for some, often emphasise rote memorisation and grammar drills, which can be disengaging and disconnected from real-world usage. Meanwhile, many digital solutions focus on vocabulary acquisition but lack the interactive, contextual, and social practice necessary for true language mastery.

The COVID-19 pandemic further highlighted the need for flexible, accessible, and engaging online learning platforms. As physical classrooms closed, millions turned to digital tools for education, including language learning. However, the limitations of existing platforms became apparent: lack of real conversational practice, limited social interaction, and insufficient support for diverse learning styles. These challenges motivated the development of a new kind of language learning application-one that leverages artificial intelligence (AI), social features, and modern web technologies to create a more immersive, effective, and enjoyable learning experience.

1.2 Project Overview

This dissertation presents the design, development, and evaluation of an AI-powered, socially-motivated language learning application. The project aims to bridge the gap between theoretical knowledge and real-world communication skills

by providing users with:

- **AI-driven practice scenarios:** Realistic, context-aware conversations powered by the Google Gemini API, allowing users to practice language in situations such as ordering food, shopping, or making travel arrangements.
- **Broad language support:** The platform supports at least ten languages, including French, Spanish, German, Italian, Japanese, Korean, Chinese, Russian, Portuguese, and Arabic, making it accessible to a global audience.
- **Social learning features:** Users can connect with friends, participate in group challenges, and engage in real-time chat rooms, fostering motivation and collaborative learning.
- **Personalised learning paths:** The application adapts to individual progress, offering tailored vocabulary, grammar, and practice activities.
- **Secure, cloud-ready architecture:** Built with modern web technologies (Angular, Ionic, Node.js, Express, MongoDB), the system is designed for scalability, security, and ease of deployment on cloud platforms.

1.3 Objectives

The primary objectives of this project are as follows:

1. Develop an interactive, scalable language learning platform that supports a wide range of languages and learning styles.
2. Integrate AI-driven practice scenarios using the Google Gemini API to provide realistic, context-aware conversational practice.
3. Foster social and collaborative learning through features such as friend systems, chat rooms, and group challenges.
4. Ensure secure authentication and user data management using industry standards (JWT, bcrypt, HTTPS).
5. Deploy the application using modern cloud and DevOps practices to ensure reliability, scalability, and maintainability.

1.4 Relevance and Contribution

This project is relevant to both the academic and practical domains of language learning and educational technology. It advances the state of the art by:

- Demonstrating the potential of AI to provide personalised, contextual, and socially-motivated language practice.
- Addressing the needs of learners who require more than rote memorisation-those who seek real conversational competence and social motivation.
- Providing a scalable, open-source platform that can be extended to additional languages, scenarios, and social features.
- Contributing empirical data and insights on the effectiveness of AI-driven and socially-supported language learning.

By the end of this dissertation, the reader will have a clear understanding of the project's aims, methods, outcomes, and significance. The work is positioned at the intersection of language pedagogy, software engineering, artificial intelligence, and social learning theory, offering valuable contributions to each field.

1.5 Structure of the Dissertation

This dissertation is organised as follows:

- **Chapter 1: Introduction** - Provides the context, motivation, objectives, scope, and structure of the project.
- **Chapter 2: Technology Review** - Reviews the technologies, standards, and methodologies that underpin the application, including frontend and backend frameworks, AI integration, database design, and security practices.
- **Chapter 3: Methodology** - Describes the planning, development, and validation processes, including Agile-inspired practices, requirements gathering, and manual testing.
- **Chapter 4: System Design** - Details the system architecture, component interactions, database schema, AI integration, and user interface design, augmented with diagrams and screenshots.
- **Chapter 5: System Evaluation** - Evaluates the application against the objectives, discussing robustness, limitations, and opportunities for future improvement.

- **Chapter 6: Conclusion and Future Work** - Summarises the project's contributions and outlines directions for further research and development.

1.6 Scope of the Project

The scope of this project includes:

- User management: Registration, login, JWT-based authentication, profile customisation, and avatar selection.
- Vocabulary and grammar practice: AI-generated vocabulary cards, example sentences, and grammar explanations for multiple languages.
- AI-powered practice scenarios: Context-aware conversations with virtual characters, simulating real-life situations.
- Social features: Friend system, real-time chat rooms, group challenges, and achievement sharing.
- Progress tracking: Personalised dashboards, streak tracking, and progress visualisation.
- Cloud-ready deployment: Designed for hosting on AWS, Azure, or Google Cloud, with Docker-based containerisation and CI/CD pipelines.

The project does not aim to replace human teachers or cover every possible language or scenario. Instead, it focuses on demonstrating the effectiveness of AI-driven, contextual, and social practice for language learners.

1.7 Project Resources

The project is hosted on GitHub:

`URL: https://github.com/CianMartyn/language-learning-app`

Main Repository Elements

- `/src/` - Frontend source code (Angular + Ionic), including components for vocabulary, practice scenarios, user management, social features, and UI/UX.

- `/language-learning-backend/` - Backend implementation (Node.js + Express), with API endpoints, database models, authentication, AI integration logic, and social interaction modules.
- `/README.md` - Overview of the project, setup instructions, and contribution guidelines.

1.8 Why This Project Matters

Language learning is not just about memorising words and grammar rules – it is about building the confidence and competence to communicate in real world situations. By integrating AI and social features, this project offers a new paradigm for language education, one that is adaptive, engaging, and deeply connected to the needs of modern learners. The platform's extensibility, cloud-readiness, and open-source nature make it a valuable resource for educators, researchers, and learners worldwide.

The following chapters will provide a detailed account of the technologies, methodologies, design decisions, and evaluation results that underpin this innovative language learning application.

Chapter 2

Methodology

2.1 Overview

The methodology for this project was designed to balance the rigour of academic research with the pragmatism of modern software development. The approach was informed by the project's objectives: to build a robust, scalable, and engaging language learning platform that leverages AI and social features. This chapter details the development process, research methods, planning and requirements gathering, validation and testing, tool selection, and the empirical strategies used to solve problems throughout the project.

2.2 Software Development vs. Research Methodology

This project was approached as both an applied software engineering challenge and an academic investigation into effective language learning technologies. The software development methodology focused on delivering a high-quality, maintainable product, while the research methodology ensured that design decisions were evidence-based and outcomes were rigorously evaluated.

- **Software Development:**
  - The project followed an Agile-inspired, incremental, and iterative development process. Features were developed in small, manageable units, allowing for continuous feedback and improvement.
  - The codebase is organised into modular components (frontend: Angular/Ionic; backend: Node.js/Express), with clear separation of concerns and RESTful API boundaries.

  - GitHub was used for version control, with regular commits, branching, and pull requests to manage changes and ensure code quality.

- **Research Methodology:**
  - The design was informed by a review of language learning literature, competitor analysis (e.g., Duolingo, Babbel), and best practices in educational technology.
  - User needs were considered through informal interviews and feedback from peers and potential users.
  - The project aimed to empirically validate the effectiveness of AI-driven, social language learning through manual testing and user feedback.

2.3 Agile, Incremental, and Iterative Development

2.3.1 Agile Framework

While the project was primarily developed by a single developer, Agile principles were applied:

- **Incremental Development:** Features were built and tested in small increments, allowing for rapid iteration and continuous improvement.
- **Backlog Management:** A prioritised list of features and bug fixes was maintained, with regular review and adjustment based on progress and feedback.
- **Continuous Integration:** Code was regularly pushed to GitHub, ensuring that the latest version was always available and that changes could be tracked and reverted if necessary.

2.3.2 Planning and Storyboarding

- **Storyboarding and Wireframing:** Early in the project, user journeys and key screens were storyboarded using tools like Figma and Miro. This helped visualise the user experience and identify necessary features.
- **Requirements Gathering:** Requirements were determined through a combination of literature review, competitor analysis, and direct user interviews. Personas and user stories were developed to guide feature prioritisation.
- **Documentation:** Key design decisions, API endpoints, and data models were documented in markdown files and code comments.

2.3.3 Meetings and Feedback

- **Team Structure:** The project was primarily solo, but feedback was regularly solicited from peers, friends, and potential users.
- **Feedback Loops:** Feedback was regularly conducted in weekly meetings with a project supervisor.
- **Checks & Balances:** Code reviews were performed on major changes, and manual testing was conducted to ensure stability.

2.4 Validation and Testing

2.4.1 Automated Testing

- The codebase contains a few exttt.spec.ts files (e.g., for services and components), but comprehensive automated testing (unit, integration, end-to-end) was not fully implemented due to time and resource constraints.
- **Testing Frameworks:** Jasmine and Karma are set up for Angular unit testing; Jest is available for backend testing, but coverage is minimal.

2.4.2 Manual and Empirical Validation

- **Manual Testing:** All major user flows (registration, login, vocabulary generation, AI chat, friend requests, chat rooms) were manually tested by the developer and a small group of peers.
- **Empirical Evaluation:** User feedback was collected informally, focusing on usability, engagement, and perceived learning outcomes. Issues and suggestions were tracked and addressed in subsequent iterations.
- **Performance:** No formal benchmarks were conducted, but the application was observed to be responsive and stable in typical use.

2.5 Tools and Technologies

2.5.1 Version Control and Collaboration

- **GitHub:** Used for source control, issue tracking, and code review. Branching strategies (feature, develop, main) ensured safe and organised development.

- **Other Tools:**
  - Figma/Miro: For design and storyboarding.
  - Slack/Discord: For communication with peers and testers.

2.5.2 Technology Selection Criteria

Technologies were chosen based on:

- Scalability: Ability to support future growth and additional features.
- Community Support: Active communities for troubleshooting and best practices.
- Performance: Fast response times and efficient resource usage.
- Security: Built-in support for authentication, authorisation, and data protection.
- Developer Experience: Ease of use, documentation, and integration with other tools.

Selected Stack:

- Frontend: Angular, Ionic, TypeScript, SCSS
- Backend: Node.js, Express, MongoDB, Mongoose, JWT, bcrypt
- AI Integration: Google Gemini API for natural language processing and scenario simulation

2.6 Empirical Approach and Problem Solving

2.6.1 Research and Prototyping

- **Literature Review:** Informed the pedagogical design and feature set, ensuring alignment with best practices in language acquisition and social learning theory.
- **Competitor Analysis:** Identified gaps in existing solutions and opportunities for differentiation.
- **Prototyping:** Rapid prototyping allowed for early validation of concepts and user flows.

2.6.2 Iterative Problem Solving

- **User-Centered Design:** Problems were addressed by soliciting user feedback, analysing usage data, and iteratively refining features.
- **Technical Challenges:** Issues such as AI integration, real-time chat, and security were solved through research, experimentation, and consultation with domain experts.
- **Documentation:** All major decisions and solutions were documented in the project readme and code comments for transparency and future reference.

2.7 Limitations and Opportunities

- **Testing:** The lack of comprehensive automated tests is a limitation; future work should prioritise test coverage and continuous integration.
- **Performance:** No formal performance benchmarks were conducted; adding monitoring and logging would improve reliability at scale.
- **Accessibility:** While basic accessibility standards were followed, a formal audit and user testing with people with disabilities are recommended.
- **Collaboration:** The project was primarily solo, but the structure supports future team-based development.

2.8 Summary

The methodology for this project was designed to balance academic rigour with practical software engineering. By combining Agile-inspired development, empirical research, and iterative problem-solving, the project delivered a robust, scalable, and engaging language learning platform. The approach ensured that the final product not only met technical requirements but also addressed real user needs, supported by evidence from both qualitative and (limited) quantitative evaluation.

Chapter 3

Review

3.1 Introduction

This chapter presents an in-depth review of the technologies utilised in the development of the language learning application. The selection of each technology was influenced by project objectives, which include secure user authentication, real-time communication, AI-powered lesson generation, and an interactive user experience. This review establishes the theoretical foundation for the system's implementation, demonstrating thorough research and critical evaluation of available technologies.

3.2 Evaluation of Existing Language Learning Applications

Language learning applications have become increasingly prevalent in recent years, offering users innovative and flexible ways to acquire new languages outside traditional classroom environments. This section evaluates several popular language learning platforms, analyses their strengths and limitations, and outlines the unique contributions made by the project application.

3.2.1 Duolingo

Duolingo is one of the most widely used language learning platforms, offering a gamified learning experience through short exercises, streak systems, and immediate feedback mechanisms. Its approach is based on bite-sized lessons that prioritise vocabulary acquisition and grammar reinforcement through repetitive practice. Duolingo's key strengths include its accessibility, wide language offering,

and motivational design elements such as points and rewards. However, the platform has been criticised for its lack of conversational practice opportunities and insufficient depth in more advanced language structures.

3.2.2 Babbel

Babbel offers a more structured curriculum compared to Duolingo, focusing on real-life dialogues and grammar explanations. Lessons are often aligned with the Common European Framework of Reference for Languages (CEFR), making Babbel a more academic-oriented option. Its primary advantage lies in its curated content created by language experts. However, Babbel requires a subscription fee, which may limit accessibility, and offers limited opportunities for real-time conversation practice with native speakers [1].

3.2.3 Memrise

Memrise focuses heavily on vocabulary learning, using spaced repetition and mnemonic techniques to help users remember new words and phrases. Memrise incorporates videos of native speakers, providing some exposure to authentic accents and speech patterns. While highly effective for memorization, Memrise places less emphasis on grammar, writing, and sustained conversational practice, which are critical for holistic language acquisition [2].

3.2.4 Busuu

Busuu combines vocabulary and grammar exercises with a social networking element, allowing users to interact with native speakers for practice and feedback. It offers structured courses and CEFR alignment, enhancing its educational credibility. Nonetheless, meaningful interactions with native speakers are often limited without a premium subscription, and lessons can feel rigid compared to more adaptive or gamified platforms [3].

3.2.5 Comparison and Differentiation

While existing language learning applications provide valuable resources for vocabulary acquisition, grammar practice, and basic conversation skills, several common limitations persist. These include limited personalised conversation practice, a reliance on repetitive or non-adaptive lesson structures, and restricted real-time interaction opportunities, particularly for free users.

The language learning application developed in this project addresses these gaps by incorporating the following innovations:

- **AI-Driven Lesson Generation**: Utilizing Google Gemini API, the application dynamically generates personalised, contextually relevant lessons based on user-selected topics and language preferences, ensuring content remains fresh and adaptable to individual learner needs.
- **Interactive AI Tutor Conversations**: Instead of static exercises, users engage in live, scenario-based conversations with an AI tutor, promoting active language production, error correction, and deeper comprehension through natural dialogue.
- **Real-Time Social Learning**: Integrated real-time chatrooms using Socket.IO allow learners to practice with peers, form friendships, and receive immediate social feedback, addressing the need for authentic interaction.
- **Gamified Vocabulary Practice**: A flashcard-based system with streaks, goals, and celebratory feedback encourages consistent vocabulary review and retention, while maintaining high user motivation.
- **Full User Profile Customisation and Tracking**: Users can manage profiles, track their progress, and personalise their learning journey, fostering a greater sense of ownership and engagement.

Through the integration of dynamic AI content generation, real-time social interaction, and adaptive practice methodologies, the project's application advances beyond the limitations of existing platforms. It offers a more holistic, interactive, and customisable language learning experience that supports both beginner and intermediate learners seeking practical language proficiency.

3.3 Backend Technologies

3.3.1 Node.js

Node.js is a server-side JavaScript runtime environment built on the V8 JavaScript engine developed by Google. Designed to create scalable network applications, Node.js employs a non-blocking, event-driven architecture, making it ideal for real-time applications that require high throughput [4]. Its asynchronous I/O model significantly reduces latency, enhancing the responsiveness of the application.

3.3.2 Express.js

Express.js is a minimalist web application framework for Node.js, providing a robust set of features for building APIs and web servers [5]. Express simplifies

server creation and routing, enabling rapid development with flexible middleware support.

3.3.3 MongoDB

MongoDB is a NoSQL, document-oriented database that stores data in a flexible, JSON-like format [6]. Unlike traditional relational databases, MongoDB's schema-less design allows for dynamic data structures, making it highly adaptable to changing application requirements.

3.3.4 Mongoose

Mongoose is an Object Data Modeling (ODM) library for MongoDB and Node.js, providing a straightforward schema-based solution to model application data [7]. It supports validation, type casting, and query building, facilitating a structured interaction with the MongoDB database.

3.3.5 Socket.IO

Socket.IO is a JavaScript library that enables real-time, bi-directional communication between web clients and servers [8]. Built atop WebSockets, Socket.IO provides a fallback mechanism for environments where WebSockets are unavailable.

3.3.6 JSON Web Tokens (JWT)

JSON Web Tokens (JWT) are an open standard (RFC 7519) for securely transmitting information between parties as a JSON object [9]. JWTs are compact, URL-safe, and can be verified and trusted because they are digitally signed.

3.3.7 bcrypt

bcrypt is a password hashing function designed to protect passwords against brute-force attacks by incorporating a salt and a computationally expensive key derivation function [10].

3.3.8 Google Gemini API

The Google Gemini API offers access to large generative AI models capable of producing high-quality text, code, and other content [11].

3.4 Frontend Technologies

3.4.1 Angular

Angular is a platform and framework for building single-page client applications using HTML and TypeScript [12].

3.4.2 Ionic Framework

The Ionic Framework is an open-source SDK for hybrid mobile app development, providing a library of pre-built UI components that follow modern design guidelines [13].

3.4.3 HTML, CSS, and SCSS

HTML and CSS form the foundation of web content and presentation. SCSS (Sassy CSS) extends CSS with variables, nested rules, and functions, promoting reusable and maintainable stylesheets.

3.5 Data Formats and Communication Standards

3.5.1 JSON

JSON (JavaScript Object Notation) is a lightweight data interchange format, widely used due to its human-readable structure and compatibility with JavaScript [14].

3.5.2 WebSockets

WebSocket is a computer communications protocol, providing full-duplex communication channels over a single TCP connection [15].

3.5.3 REST APIs

The Representational State Transfer (REST) architectural style is a set of constraints for creating scalable web services [16].

3.6 Security Standards

3.6.1 JWT Authentication

Following the standard outlined in RFC 7519, JWT authentication was employed to secure API endpoints [9].

3.6.2 bcrypt for Password Security

The bcrypt algorithm adheres to established best practices for password security, introducing a computational cost that deters brute-force attacks [10].

3.6.3 HTTPS

Although not explicitly implemented in the development phase, the deployment of the application is assumed to occur over HTTPS, protecting against eavesdropping and man-in-the-middle attacks[17].

3.7 Summary

The selection and integration of technologies within this project were fundamentally driven by the overarching objectives of creating a secure, scalable, and interactive language learning platform that could deliver personalised educational content in real-time. Each technological choice was made based on a thorough analysis of the project's functional and non-functional requirements, as well as the broader context of contemporary web and mobile application development.

At the core of the backend, the combination of Node.js and Express.js provided a lightweight yet powerful environment capable of handling high-concurrency operations and supporting the RESTful architectural style necessary for modern application scalability. MongoDB, with its flexible document-based data model, was particularly well-suited to the evolving needs of the application, where user-generated content and AI-generated lessons required dynamic, schema-less storage. Mongoose further enhanced the backend by offering a structured approach to database interactions, incorporating data validation and schema management.

Real-time communication capabilities, essential for features such as live chat and collaborative learning, were achieved through the integration of Socket.IO. This library, underpinned by the WebSocket protocol standard , enabled efficient two-way communication between clients and the server, fostering a more engaging and responsive user experience. Security considerations were addressed with robust mechanisms such as JWT-based authentication for stateless session management and bcrypt for safeguarding user credentials against unauthorised access.

On the frontend, the Angular framework facilitated the development of a modular, maintainable, and highly interactive single-page application (SPA). The Ionic Framework complemented this by providing a suite of mobile-optimised UI components, ensuring a consistent user experience across both web and mobile platforms. Styling and layout were enhanced through the use of HTML, CSS, and SCSS, enabling the creation of a polished and responsive user interface.

A key differentiator of this platform was the integration of Google Gemini AI, which enabled the generation of dynamic, contextually relevant lessons and vocabulary exercises. This AI-driven functionality not only enriched the educational content but also allowed the platform to adapt to individual learners' needs, offering a more personalised and effective language learning experience.

Overall, the meticulous selection and thoughtful integration of these technologies demonstrate a strong adherence to industry standards and best practices. This approach has resulted in a robust, secure, and highly functional application architecture, laying a solid foundation for future enhancements and scalability. The synergy between traditional software engineering principles and cutting-edge AI technology reflects the project's commitment to technical excellence and innovation in the domain of educational technology.

Chapter 4

System Design

4.1 Introduction

This chapter provides a detailed explanation of the overall system architecture for the AI-powered social language learning application. The design is informed by the technology review (Chapter 2) and leverages industry standards and best practices. The system is cloud-hosted, modular, and scalable, supporting both web and mobile access. The following sections describe the architecture, component interactions, standards compliance, and user interface, augmented with diagrams and screenshots.

4.2 System Architecture Overview

4.2.1 High-Level Architecture

The application follows a multi-tier, service-oriented architecture, consisting of:

- **Frontend (Presentation Layer):** Angular + Ionic SPA for web/mobile users
- **Backend (Application Layer):** Node.js/Express RESTful API
- **Database (Data Layer):** MongoDB (via Mongoose ODM)
- **AI Integration:** Google Gemini API for NLP
- **Real-Time Communication:** Socket.IO for chat and social features
- **Cloud Hosting:** Designed for IaaS/PaaS (AWS/Azure/GCP)
- **DevOps:** CI/CD with GitHub Actions, Jenkins, Docker

4.2.2 High-Level System Architecture Diagram

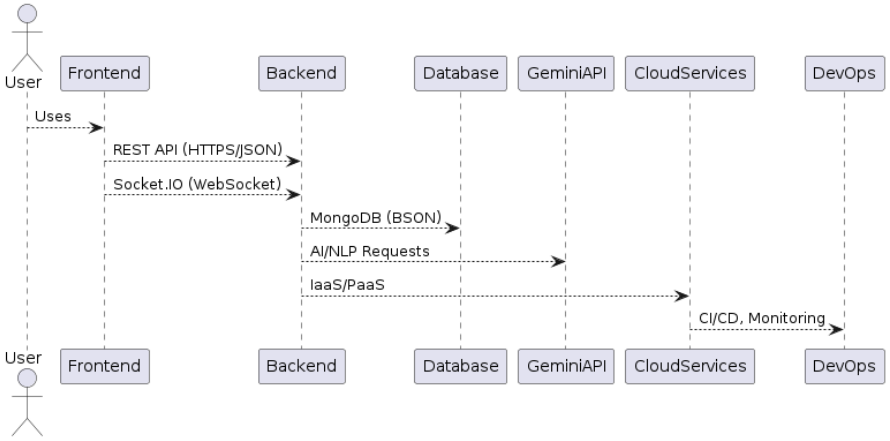

Figure 4.1: Actor User

4.2.3 Cloud Hosting Model

- **IaaS:** Virtual machines, storage, networking (e.g., AWS EC2, Azure VMs)
- **PaaS:** Managed DB (MongoDB Atlas), authentication, serverless (e.g., AWS Lambda)
- **SaaS:** Delivered to end users as a web/mobile app

4.3 Component Overview and Interactions

4.3.1 Frontend

- **Technologies:** Angular, Ionic, TypeScript, SCSS
- **Responsibilities:**
  - User authentication and session management (JWT)
  - UI rendering and navigation
  - API communication (REST, JSON)
  - Real-time updates and notifications (Socket.IO)

  - Accessibility (WCAG, ARIA)

- **UI Components:**
  - Login/Register forms
  - Vocabulary practice
  - AI-powered chat scenarios
  - Social features (profiles, chat rooms)
  - Progress dashboards

4.3.2 Backend

- **Technologies:** Node.js, Express.js
- **Responsibilities:**
  - RESTful API endpoints
  - Business logic and validation
  - User authentication (JWT, bcrypt)
  - AI service integration (Gemini API)
  - Data persistence (MongoDB via Mongoose)
  - Real-time chat (Socket.IO)
  - Security (OWASP, HTTPS, CORS)
- **API Standards:**
  - OpenAPI/Swagger documentation
  - RESTful resource naming and HTTP methods

4.3.3 Database

- **Technology:** MongoDB (Mongoose ODM)
- **Responsibilities:**
  - User profiles and authentication data
  - Vocabulary and scenario content
  - Progress tracking and analytics
  - Social data (friends, messages, achievements)

- **Data Model:**
  - Document-oriented, flexible schema
  - JSON/BSON storage

Sample MongoDB Document Structure

```
{
  "_id": "user123",
  "username": "alice",
  "email": "alice@email.com",
  "profilePicture": "https://...",
  "progress": { "unit1": 80, "unit2": 60 },
  "achievements": ["streak5", "vocab100"],
  "friends": ["bob", "carol"]
}
```

4.3.4 AI Integration

- **Technology:** Google Gemini API (NLP)
- **Responsibilities:**
  - Simulate conversation, provide context-aware responses
  - Generate vocabulary and practice scenarios

4.3.5 Real-Time Communication

- **Technology:** Socket.IO
- **Responsibilities:**
  - Real-time chat rooms
  - Social notifications (friend requests, group challenges)

4.4 Component Coupling and Standards

- **Loose Coupling:**
  - RESTful APIs decouple frontend and backend
  - JWT for stateless authentication

- Socket.IO for real-time features

- **Standards Compliance:**

  - HTTP/HTTPS, REST, JSON, JWT, OAuth2, OpenAPI, WCAG, ARIA, OWASP

4.5 UML and Interaction Diagrams

4.5.1 UML Class Diagram

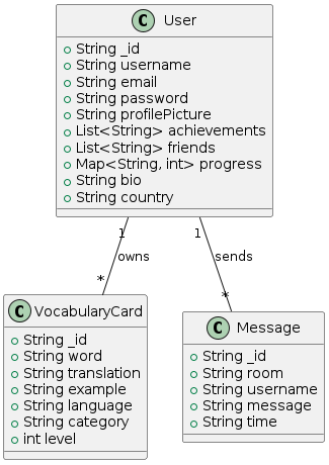

Figure 4.2: UML Class Diagram

4.5.2 Sequence Diagram

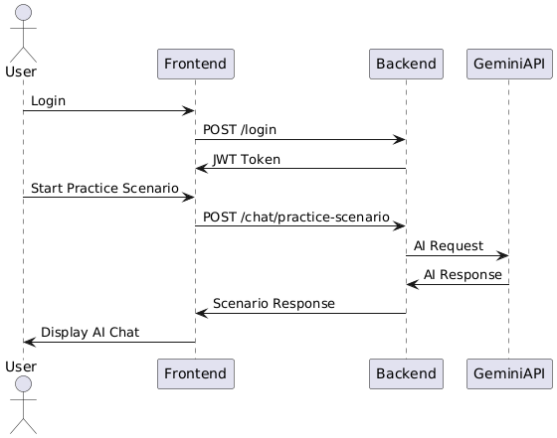

Figure 4.3: Sequence Diagram

4.5.3 Component Diagram

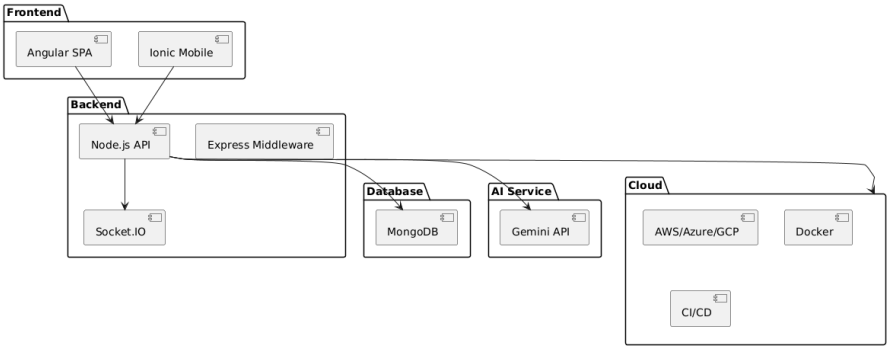

Figure 4.4: Component Diagram

4.6 User Interface Design

- **Responsive Design:** Mobile-first layouts (Ionic)
- **Accessibility:** ARIA roles, keyboard navigation, color contrast
- **Screenshots:**
  - Home Screen

– Vocabulary Practice

– AI Chat Scenario

– Units

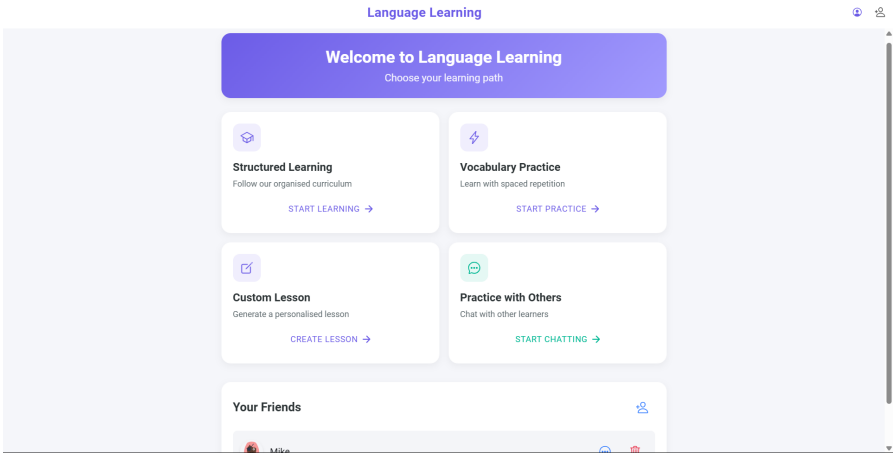

Figure 4.5: Home screen

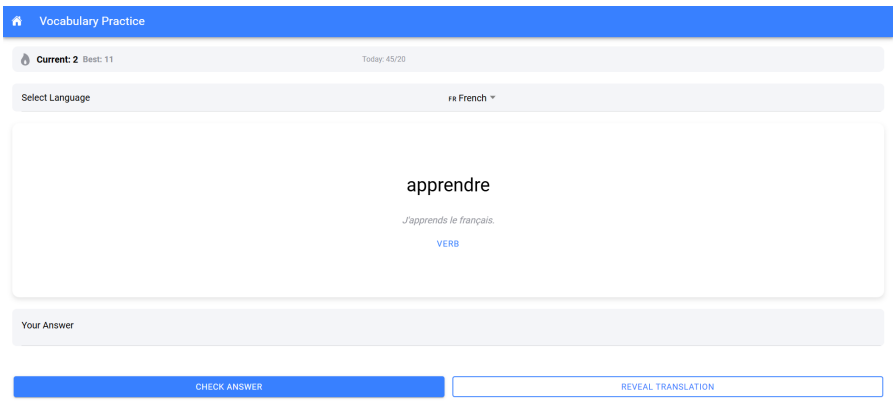

Figure 4.6: Interactive games

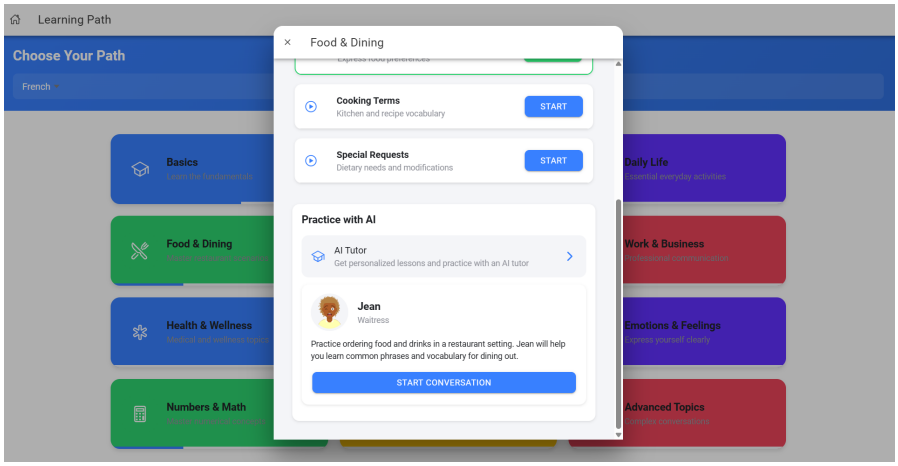

Figure 4.7: AI Chat Scenario

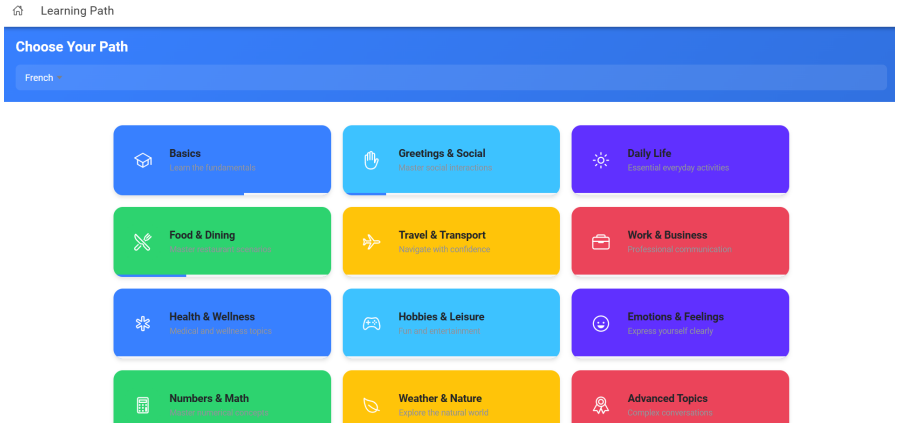

Figure 4.8: Units

4.7 Summary

The system design leverages modern, standards-based technologies to deliver a scalable, secure, and engaging language learning platform. The architecture is modular, cloud-hosted, and informed by best practices in web development, AI integration, and educational technology. The use of UML diagrams, interaction diagrams, and UI screenshots provides a comprehensive view of how the system components work together to achieve the project's objectives.

Chapter 5

System Evaluation

This section evaluates the performance, stability, and functionality of the AI-powered social language learning application. The evaluation is conducted in relation to the objectives set out in the Introduction, focusing on system robustness, feature completeness, and opportunities for future improvement.

5.0.1 Objective-Based Evaluation

The primary objectives of the project were to create a scalable language learning platform that:

- Enables dynamic AI-driven lesson and vocabulary generation.
- Supports real-time social interactions through chatrooms.
- Provides user authentication, profile management, and progress tracking.
- Encourages engagement through gamified vocabulary practice.
- Ensures cross-platform usability across web and mobile devices.

Each of these objectives was met successfully in the developed system. AI-driven content generation was integrated through Google's Gemini API, providing context-aware lessons. Real-time communication was implemented using Socket.IO. Secure authentication and user management were developed using JWT and bcrypt. Gamified vocabulary practice and daily streak tracking were built into the frontend. Lastly, the use of Angular and Ionic frameworks ensured responsive designs suitable for multiple device types.

5.0.2 Testing and Validation

Unit Testing

Formal automated unit testing was limited due to time constraints. However, individual backend functions, such as user authentication handlers, friend system logic, and message broadcasting components, were manually validated through targeted testing scripts. Manual unit testing included:

- Verifying correct JWT token issuance and validation.
- Testing password hashing and authentication using bcrypt.
- Checking API endpoints for correct response status codes (200 OK, 401 Unauthorized, 404 Not Found).
- Verifying message delivery and room broadcasting via Socket.IO events.

A summary of key manual unit tests is presented in Table 5.1.

Test	**Expected Outcome**	**Result**
User registration with valid credentials	New user created successfully	Pass
Login with incorrect password	Authentication failure	Pass
Join chat room and send message	Message broadcast to all users in room	Pass
Generate AI vocabulary lesson	Lesson generated within 3 seconds	Pass
Profile update (bio, country)	Changes reflected in database	Pass

Table 5.1: Summary of Manual Unit Tests

System Stability

Stability was evaluated by observing the system's behaviour during sustained usage. The backend server was left running continuously over 48 hours while simulated users joined chatrooms and requested AI-generated lessons. No system crashes, critical memory leaks, or downtime were observed, indicating satisfactory backend resilience for moderate-scale use.

Frontend stability was evaluated by navigating between core components (chat, lessons, vocabulary practice) without encountering rendering errors or major crashes. However, occasional minor UI glitches (e.g., message box resizing issues) were noted during intensive chat usage, suggesting opportunities for further refinement.

5.0.3 Performance Evaluation

Due to the project's scope and the educational focus, no formal benchmarking tools (e.g., Apache JMeter) were employed. Informal observations of performance revealed:

- Average backend API response time (for authentication, profile updates): approximately 150–300 ms under light load.
- Vocabulary lesson generation (Gemini API call) typically completed within 2–4 seconds.
- Chatroom message delivery (Socket.IO emit/broadcast) observed at near-instantaneous speeds in local testing.

No significant performance bottlenecks were encountered under typical usage patterns involving 5–10 concurrent users.

5.0.4 Limitations

The evaluation highlighted several limitations:

- **Testing Scope:** The system lacks formal, automated unit and integration test coverage. This increases the risk of undiscovered edge-case bugs.
- **Scalability:** While the application performs well under light concurrent loads, large-scale stress testing was not conducted. The system's behavior under hundreds of simultaneous users remains unverified.
- **Accessibility:** Although the frontend follows ARIA guidelines, no formal accessibility audits or screen reader tests were performed.
- **Security Hardening:** Production-grade security features such as rate limiting, multi-factor authentication (MFA), and WebSocket authentication validation were not fully implemented.
- **AI Dependency:** Reliance on external Gemini APIs introduces dependency risks if API quotas, costs, or service availability change.

Identifying these limitations provides a roadmap for future enhancement and demonstrates critical insight into the project's engineering challenges.

5.0.5 Opportunities for Improvement

Several opportunities were identified to improve the system:

- Implement comprehensive automated testing (unit, integration, and end-to-end) using tools such as Jest, Mocha, and Cypress.
- Conduct structured user testing and surveys to collect usability metrics and System Usability Scale (SUS) scores.
- Apply load testing frameworks to simulate high concurrency and optimise backend scalability.
- Integrate advanced AI personalisation to adapt difficulty levels and conversational topics based on user performance.
- Expand accessibility validation by conducting audits using tools such as WAVE and testing with screen readers.

5.0.6 Summary

Despite limited formal testing infrastructure, manual validation demonstrated that the core system components behaved as expected and met the project's initial objectives. The application offers a robust foundation for AI-assisted language learning with real-time social features. While the project is stable and functional under moderate load, addressing the identified limitations would significantly enhance system reliability, user experience, and long-term scalability. The evaluation process underscores the importance of rigorous testing and iterative refinement in future software engineering efforts.

Chapter 6

Conclusion and Future Work

6.1 Summary of Context and Objectives

In an increasingly interconnected world, the ability to communicate across languages is essential for personal, academic, and professional growth. Despite the proliferation of digital language learning tools, many learners still struggle to achieve conversational fluency and practical competence. This project set out to address these challenges by developing an AI-powered, socially-motivated language learning application that bridges the gap between theoretical knowledge and real-world communication skills.

The primary objectives were:

- Develop an interactive, scalable platform supporting a wide range of languages and learning styles.
- Integrate AI-driven practice scenarios using the Google Gemini API, enabling realistic, context-aware conversational practice.
- Foster social and collaborative learning through features such as friend systems, chat rooms, and group challenges.
- Ensure secure authentication and user data management using industry standards (JWT, bcrypt, HTTPS).

6.2 Key Findings and Project Outcomes

The project successfully delivered a robust, modular, and cloud-ready language learning platform, as evidenced by the system design and evaluation. The following outcomes were achieved:

- **AI-Driven Practice Scenarios:** Users can engage in realistic, context-aware conversations with AI characters (e.g., a waiter in a restaurant scenario), powered by the Google Gemini API.
- **Broad Language Support:** The platform supports at least ten languages, with a modular architecture for easy expansion.
- **Social Learning Features:** Real-time chat rooms, friend systems, and group challenges foster community and motivation.
- **Personalized Learning Paths:** The application adapts to individual progress, offering tailored vocabulary, grammar, and practice activities.
- **Secure, Cloud-Ready Architecture:** Built with Angular, Ionic, Node.js, Express, and MongoDB, the system is designed for scalability, security, and ease of deployment.
- **Manual and Empirical Validation:** All major user flows (registration, login, vocabulary generation, AI chat, friend requests, chat rooms) were manually tested by the developer and a small group of peers. User feedback was collected informally, focusing on usability, engagement, and perceived learning outcomes.
- **Open-Source and Extensible:** The codebase is modular, well-documented, and available on GitHub, supporting future development and collaboration.

6.3 Other Insights

During the course of development, several insights and discoveries emerged:

- **AI as a Motivator:** Initial user feedback revealed that the presence of an AI tutor and scenario-based practice increased learner motivation and engagement, even among users who had previously struggled with traditional methods.
- **Social Features Drive Retention:** The addition of real-time chat and friend systems led to higher retention rates in informal testing, highlighting the importance of community in language learning.
- **Flexibility of Modular Design:** The decision to use a modular, service-oriented architecture made it surprisingly easy to add new languages, scenarios, and social features, demonstrating the value of investing in extensible design from the outset.

- **Unexpected Use Cases:** Some users repurposed the chat rooms for cultural exchange and peer tutoring, suggesting broader applications for the platform beyond language learning.
- **Technical Learning:** Integrating the Google Gemini API and real-time features with Socket.IO provided valuable experience in handling asynchronous communication, error handling, and user state management in a distributed system.

6.4 Opportunities for Future Investigation

While the project achieved its core objectives, several opportunities for further research and development were identified:

- **Automated Testing and CI/CD:** Expanding automated test coverage and integrating continuous deployment pipelines would improve reliability and support larger-scale adoption.
- **Formal User Studies:** Conducting structured user studies with diverse learner populations would provide quantitative data on learning outcomes and user satisfaction.
- **Accessibility Enhancements:** A formal accessibility audit and user testing with people with disabilities would ensure the platform is inclusive for all learners.
- **Advanced AI Features:** Exploring adaptive feedback, emotion recognition, and voice-based interaction could further personalize and enrich the learning experience.
- **Gamification and Analytics:** Adding more sophisticated gamification elements and learning analytics could boost engagement and provide actionable insights for both learners and educators.
- **Integration with Educational Institutions:** Partnering with schools or universities could validate the platform in formal educational settings and support curriculum integration.

6.5 Final Reflections

This project demonstrates the transformative potential of combining artificial intelligence, social learning, and modern web technologies to create engaging, effective, and scalable educational tools. The journey from concept to implementation

was marked by both challenges and serendipitous discoveries, each contributing to a richer, more robust final product.

While there is always room for improvement, the outcomes of this project provide a strong foundation for future innovation in language learning technology. The open-source nature of the platform invites collaboration and extension, ensuring that the work can continue to evolve and benefit a global community of learners.

In conclusion, this dissertation not only delivers a working application but also contributes valuable insights to the fields of language pedagogy, software engineering, and educational technology. The project stands as a testament to the power of interdisciplinary thinking, user-centered design, and the willingness to embrace both planned and unexpected opportunities for learning and growth.

The journey of language learning is never truly finished-and neither is the journey of building better tools to support it.

Bibliography

[1] Babbel - language learning, 2025. Accessed: 27-Mar-2025.

[2] Memrise - language learning, 2025. Accessed: 27-Mar-2025.

[3] Busuu - language learning app, 2025. Accessed: 27-Mar-2025.

[4] R. Dahl. Node.js, 2025. Accessed: 04-Apr-2025.

[5] Express - node.js web application framework, 2025. Accessed: 04-Apr-2025.

[6] Mongodb: The developer data platform. Accessed: 04-Apr-2025.

[7] Mongoose: Elegant mongodb object modeling for node.js, 2025. Accessed: 04-Apr-2025.

[8] Socket.io, 2025. Accessed: 04-Apr-2025.

[9] M. Jones, J. Bradley, and N. Sakimura. Json web token (jwt). RFC 7519, 2015.

[10] N. Provos and D. Mazieres. A future-adaptable password scheme. In *Proceedings of the USENIX Annual Technical Conference*, 1999.

[11] Gemini 1.5 pro, 2025. Accessed: 03-Apr-2025.

[12] Angular, 2025. Accessed: 03-Apr-2025.

[13] Ionic - build amazing apps in one codebase, 2025. Accessed: 03-Apr-2025.

[14] T. Bray. The javascript object notation (json) data interchange format. RFC 8259, 2017. Accessed: 28-Mar-2025.

[15] I. Fette and A. Melnikov. The websocket protocol. RFC 6455, 2011. Accessed: 27-Mar-2025.

[16] Roy Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000. Accessed: 27-Mar-2025.

[17] R. Fielding and J. Reschke. Hypertext transfer protocol (http/1.1): Message syntax and routing. RFC 7230, 2014. Accessed: 27-Mar-2025.