

Computer Science - LinguaLink Documentation

Christian Mendez, Haruki Horiuchi, Ajani Cole, Serkens Delice, Ashley Agbonkhese

1. Executive Summary (O4)

LinguaLink is a lightweight language-learning tool designed to help international students quickly understand unfamiliar words and phrases while studying. Many students rely heavily on translation tools, which interrupts workflow and reduces learning efficiency. LinguaLink addresses this by providing fast, context-aware translations, quizzes, and mini-lessons in a single platform.

The system is currently implemented as a Java-based console application using Spring Boot and Maven. Its modular design separates translation, quizzes, and lesson components, allowing for flexibility and scalability. Key features include automatic language detection, glossary organization, and offline caching of recent translations.

LinguaLink benefits students by reducing context switching, improving comprehension, and reinforcing vocabulary through interactive learning. Early results show improved usability and efficiency compared to traditional translation workflows. Future iterations aim to expand into a GUI and web-based platform.

2. Problem & Requirements (O1)

2.1 Context & Stakeholders

- **Primary users:** International students learning a new language
- **Secondary users:** Educators and self-learners
- **Stakeholders:** Developers, academic institutions

2.2 Problem Statement

Students frequently interrupt their workflow to use external translation tools, making studying inefficient and fragmented.

2.3 Functional Requirements

1. Provide fast, accurate translations with example sentences

2. Automatically detect user input language
3. Maintain a glossary for saved words and phrases
4. Offer quizzes and mini-lessons for reinforcement
5. Support offline usage via cached translations

2.4 Non-Functional Requirements

1. High performance (low latency translation responses)
2. Maintainable and modular object-oriented design
3. Scalability for adding new languages
4. Usability through simple interface design

2.5 Constraints

- Java-based implementation
- Console interface (current version)
- Limited external API usage (offline support focus)

2.6 Success Criteria

- Reduced time spent switching between tools
- Accurate translations with contextual examples
- Positive usability feedback from users

2.7 Risks

- Translation accuracy limitations
- Scaling complexity for multiple languages
- Limited UX due to console interface

2.8 Prioritized Backlog (Top Items)

1. Language detection module
2. Translation engine integration
3. Glossary storage system
4. Quiz generation system
5. Mini-lesson module
6. Offline caching mechanism
7. File-based vocabulary input
8. Modular architecture refactor
9. Error handling improvements

10. Performance optimization
 11. UI/UX improvements (future GUI)
 12. Multi-language expansion
 13. API integration for advanced translations
-

3. System Overview & Architecture (O2)

3.1 System Overview

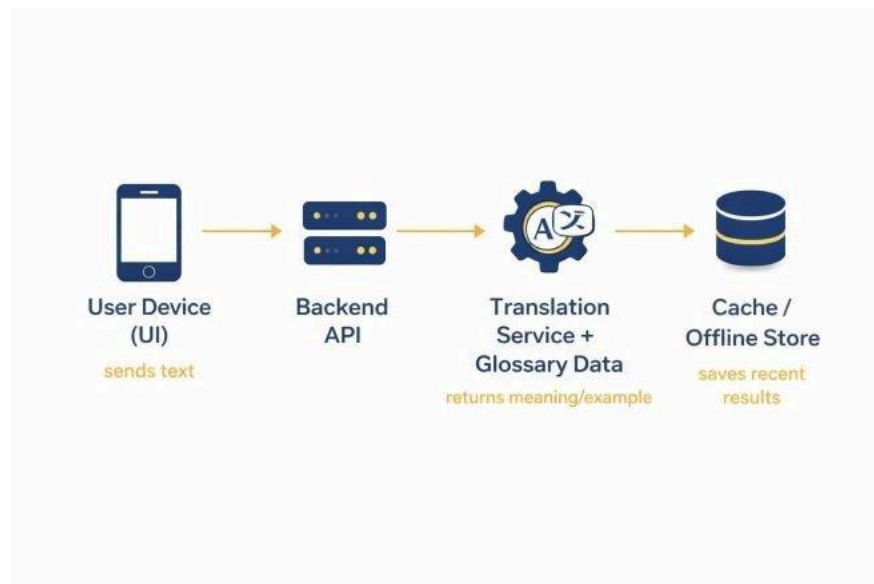
LinguaLink is built using a layered Java architecture that separates concerns into distinct components:

- **UI Layer:** Console-based user interaction
- **Logic Layer:** Translation, quizzes, and lessons
- **Data Layer:** External files for vocabulary and questions

3.2 Architecture Description

The system follows a modular, layered design:

- Input/output handled by console interface
- Translation, quiz, and lesson modules operate independently
- Data is loaded dynamically from external sources



3.3 Technologies Used

- Java
- Spring Boot
- Maven
- VS Code
- Git

3.4 API & Data Overview

- MyMemory API
 - Data stored in structured external files (e.g., JSON or text)
 - Cached translations stored locally for offline access
-

5. Implementation Notes (O2)

5.1 Repository Structure

- `/src` – Core application code
- `/translation` – Translation logic
- `/quiz` – Quiz functionality
- `/lessons` – Lesson modules
- `/data` – Vocabulary and question files

5.2 Coding Conventions

- Object-oriented design principles
- Clear separation of concerns
- Modular and reusable components

5.3 Design Patterns & Decisions

- Layered architecture for scalability
- File-based data management for flexibility
- Modular services for easy feature expansion

5.4 Tradeoffs

- Console UI chosen for simplicity over user experience
 - Offline-first design limits advanced translation features
 - File-based storage vs. database scalability
-

6. Testing & Evaluation (O3)

6.1 Testing Strategy

- **Unit Testing:** Individual modules (translation, quiz, lessons)
- **Integration Testing:** Interaction between modules
- **End-to-End Testing:** Full workflow simulation

6.2 Metrics & KPIs

- Translation response time
- Accuracy of translations
- Quiz performance metrics
- System reliability

6.3 Performance Testing

- Measured latency for translation requests
- Efficiency of cached data retrieval

6.4 Defect Summary

- Minor bugs in input handling
 - Occasional inconsistencies in translation output
-

7. Security, Privacy, Accessibility & Compliance (O5)

7.1 Security

- Input validation to prevent malformed data
- Secure handling of cached data

7.2 Privacy

- No personal user data stored
- Local-only data storage for offline functionality

7.3 Accessibility

- Simple text-based interface ensures broad accessibility
- Future GUI will include accessibility enhancements

7.4 Compliance & Ethics

- Ethical use of translation tools
 - Supports inclusive learning for international students
-

8. Deployment & Operations

8.1 Deployment

- Packaged using Maven
- Runs as a local Java application

8.2 Operations

- Minimal setup required
 - Easily configurable through external data files
 - VSCode, Java, JDK, Spring Boot, Maven,
-

9. Limitations & Future Work (O4)

9.1 Limitations

- Console-based UI limits usability
- Limited translation sophistication without APIs
- No real-time cloud synchronization

9.2 Future Work

1. Develop GUI or web-based interface

2. Integrate advanced translation APIs
3. Add speech recognition and pronunciation features
4. Expand language support
5. Improve personalization and learning analytics

Links

Poster: [LinguaLink Capstone II Poster \(1\) 1.pptx](#)

Intro Video: <https://youtu.be/OJiDt4mT-j0?si=kfNYVDvXERFrbo>

Demo Video: <https://youtu.be/kmHSsRAzRvE>