TransLingua Project Documentation

1 Project Overview

TransLingua is a multilingual translation web application designed to enable seamless communication across different languages. The system provides secure authentication, translation functionality, history management, and role-based authorization.

The application follows a **Layered Architecture (MVC pattern)** and is built using **Spring Boot** for backend development and **HTML/CSS/JavaScript** for frontend.

2 System Architecture

TransLingua follows a Layered Architecture consisting of:

Presentation Layer

- User Interface
- Login & Registration Forms
- Translation Input & Output
- History Display

Controller Layer

- Handles HTTP Requests
- Maps REST endpoints
- Validates user inputs

Service Layer

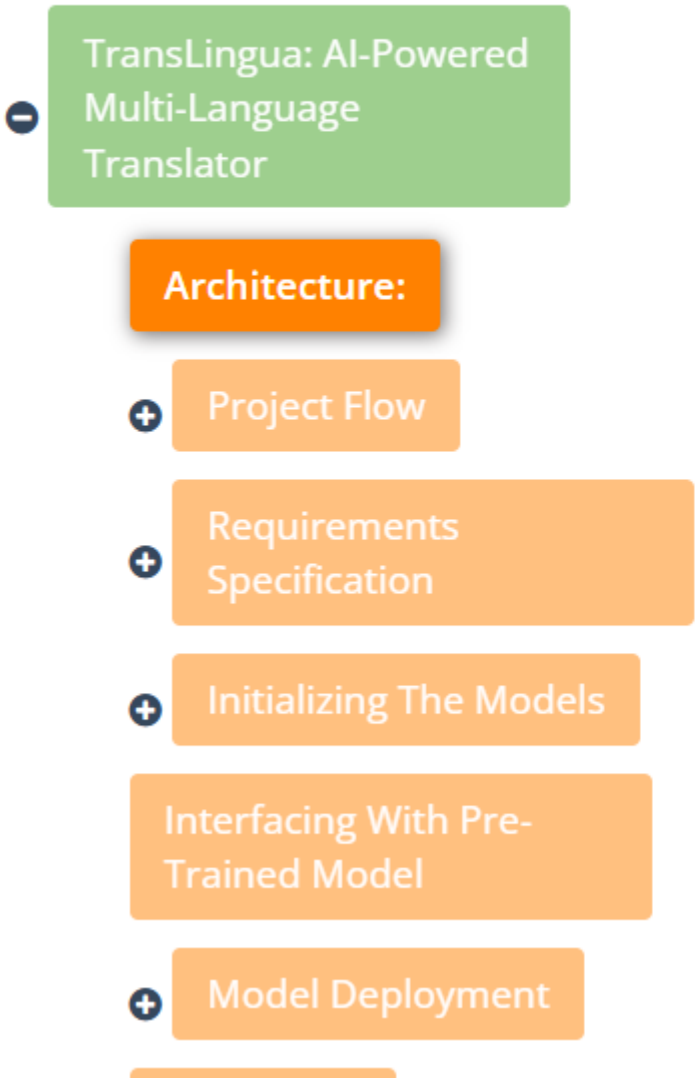
- Contains business logic
- Handles translation processing
- Manages authentication logic

Repository Layer

- Communicates with database
- Uses JPA/Hibernate

Database Layer

- Stores users
- Stores translation history
- Manages roles



- Text input translation
- Multiple language selection
- Instant translation response

□ Translation History

- Stores previous translations
- Allows retrieval
- Linked to authenticated user

□□ Admin Controls

- View users
- Delete users
- Monitor system usage

5□ Database Design

User Entity

- id (Primary Key)
- username
- email
- password (Encrypted)
- role

Translation History Entity

- id (Primary Key)
- sourceText
- targetText
- sourceLang
- targetLang
- userId (Foreign Key)

Relationship: One User → Many Translations

6 API Documentation

Authentication APIs

- POST /register
- POST /login

Translation APIs

- POST /translate
- GET /history
- DELETE /history/{id}

Admin APIs

- GET /users
- DELETE /user/{id}

All APIs follow REST standards.

7 Security Implementation

- JWT-based stateless authentication
- BCrypt password encryption
- Role-based endpoint restriction
- Token expiration handling

Security ensures safe data handling and protected access.

8□ Process Flow

Login Flow

User → Login Form → Controller → Authentication Manager → JWT Token → Response

Translation Flow

User → Input Text → Controller → Service Layer → Translation API → Save to DB → Response

9□ Non-Functional Requirements

Performance

- Optimized database queries
- Fast REST responses

Scalability

- Stateless JWT authentication
- Cloud-deployable architecture

Maintainability

- Layered architecture
- Clear separation of concerns

Security

- Secure password storage
- Role-based control

□ Deployment

- Backend deployed on cloud server (optional)
- Database hosted locally or cloud
- Version control managed through GitHub
- Tested using Postman

1□1□ Future Enhancements

- Voice-based translation

- AI-powered contextual translation
- Real-time chat translation
- Mobile application version
- Analytics dashboard