□ TransLingua – System Design Phase

1□ Introduction to System Design

The System Design phase translates project requirements into a structured technical blueprint. It defines how components interact, how data flows through the system, and how scalability and security are maintained.

For TransLingua, the design follows a **Layered Architecture (MVC Pattern)** to ensure maintainability, modularity, and scalability.

□ 2□ High-Level Architecture Design

□ Architectural Pattern: Layered Architecture (MVC)

Architecture Layers:

Presentation Layer (Frontend)

- Built using HTML, CSS, JavaScript
- Handles user input and displays translated output
- Communicates with backend via REST APIs

Controller Layer

- Receives HTTP requests
- Maps endpoints (e.g., `/login`, `/translate`)
- Validates request data

Service Layer

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

Repository Layer

- Handles database operations
- Uses JPA/Hibernate
- Communicates with MySQL

Database Layer

- Stores user data

- Stores translation history
- Maintains role-based access records

□ 3□ Security Architecture

TransLingua implements **JWT-based Authentication** using:

- Spring Security
- JWT Token Provider
- Role-Based Authorization (Admin/User)

Authentication Flow:

1. User enters login credentials
2. Credentials validated in backend
3. JWT token generated
4. Token sent to frontend
5. Token attached in Authorization header for future requests

This ensures:

- Stateless authentication
- Secure API access
- Role-based access control

□ 4□ Database Design (ER Model)

Core Entities:

□ User Table

Field	Type	Description
id	Long	Primary Key
username	String	Unique
email	String	User email
password	String	Encrypted
role	String	USER / ADMIN

□ Translation History Table

Field	Type	Description

id	Long	Primary Key
sourceText	Text	Original text
targetText	Text	Translated text
sourceLang	String	Input language
targetLang	String	Output language
userId	Long	Foreign Key

Relationship:

One User → Many Translations

User ID acts as Foreign Key in Translation table.

□ 5□ Process Flow Design

Translation Flow:

1. User
2. Frontend Form
3. REST API Call
4. Controller
5. Service Layer
6. Translation API Integration
7. Database Save
8. Response Sent to User

□ 6□ API Design

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 RESTful conventions.

⚙ 7□ Non-Functional Design Considerations

Performance

- Optimized REST responses
- Efficient DB indexing

Scalability

- Stateless JWT authentication
- Can be deployed on cloud servers

Maintainability

- Clear separation of concerns
- Modular code structure

Security

- Password hashing (BCrypt)
- JWT expiration handling
- Role-based endpoint restriction

□ 8□ Design Decisions Justification

Decision	Reason
Spring Boot	Rapid backend development
MySQL	Reliable relational storage
JWT	Secure stateless authentication
Layered Architecture	Clean code separation
REST APIs	Standard communication protocol