

French-Romanian dictionary

Name: Adriana Mihnea

Group: 30236

Table of Contents

[Deliverable 1 3](#_Toc64843130)

[Project Specification 3](#_Toc64843131)

[Functional Requirements 3](#_Toc64843132)

[Use Case Model 3](#_Toc64843133)

[Use Cases Identification 3](#_Toc64843134)

[UML Use Case Diagrams 3](#_Toc64843135)

[Supplementary Specification 3](#_Toc64843136)

[Non-functional Requirements 3](#_Toc64843137)

[Design Constraints 3](#_Toc64843138)

[Glossary 3](#_Toc64843139)

[Deliverable 2 3](#_Toc64843140)

[Domain Model 3](#_Toc64843141)

[Architectural Design 4](#_Toc64843142)

[Conceptual Architecture 4](#_Toc64843143)

[Package Design 4](#_Toc64843144)

[Component and Deployment Diagram 4](#_Toc64843145)

[Deliverable 3 4](#_Toc64843146)

[Design Model 4](#_Toc64843147)

[Dynamic Behavior 4](#_Toc64843148)

[Class Diagram 4](#_Toc64843149)

[Data Model 4](#_Toc64843150)

[System Testing 4](#_Toc64843151)

[Future Improvements 4](#_Toc64843152)

[Conclusion 4](#_Toc64843153)

[Bibliography 4](#_Toc64843154)

# Deliverable 1

## Project Specification

The French-romanian dictionary is a web application developed to assist users in understanding and utilizing the French language effectively. It serves as a comprehensive tool for language learning and reference, providing features such as dictionary lookup.

## Functional Requirements

* Add words in the dictionary: users should be able to create new words and add their translation.
* Search functionality: users should be able to enter French words or phrases into the search bar.
* Update words: users should be able to look for a specific word and make changes regarding its translation
* Delete functionality: users should be able to delete words from the dictionary

## Use Case Model:

### Use Cases Identification

### Use-Case 1: Dictionary Lookup

### Level: Primary

### Primary Actor: User

Main Success Scenario:

* User launches the French Dictionary App.
* User enters a French word or phrase into the search bar.
* The app displays search results matching the entered query.
* User selects a word from the search results.

Extensions:

* If the entered word or phrase does not match any entries in the dictionary:
  + The app displays a message indicating that no results were found.
  + User may revise the search query or try again later.

Use-Case 2: Admin CRUD Operations

Level: Secondary

Primary Actor: Admin

Main Success Scenario:

* Admin logs into the French Dictionary App with administrative credentials.
* The app authenticates the admin's credentials and grants access to administrative features.
* Admin navigates to the "Manage Words" section.
* Admin selects the desired CRUD operation:
  + Create: Admin adds a new word to the dictionary, providing details such as word, definition, translation, pronunciation, part of speech, synonyms, antonyms, and example sentences.
  + Read: Admin views existing words in the dictionary, along with their details.
  + Update: Admin modifies the details of an existing word, such as its definition, translation, or other attributes.
  + Delete: Admin removes a word from the dictionary.
* The app updates the dictionary database based on the admin's action and displays a success message.

Extensions:

* If the admin encounters an error during any CRUD operation:
  + The app displays an error message indicating the issue encountered.
  + Admin may retry the operation or contact support for assistance.

### UML Use Case Diagrams

A diagram of a dictionary

Description automatically generated

## Supplementary Specification

### Non-functional Requirements

1. Performance:
   * The French Dictionary App should have low latency in response to user interactions, especially during dictionary lookup and administrative CRUD operations.
   * Reason: A fast and responsive application enhances user experience, ensuring that users can quickly find information and perform tasks without experiencing delays or lags.
2. Security:
   * User authentication and authorization mechanisms should be implemented to ensure that only authorized users, such as admins, can access administrative features.
   * All sensitive data, including user credentials and administrative actions, should be encrypted to prevent unauthorized access or tampering.
   * Reason: Security measures protect user data and prevent unauthorized access, maintaining the integrity and confidentiality of user information.

### Design Constraints

1. Programming Language and Framework:

* The app must be developed using Java Spring Booth and React mandated by the project requirements or organizational standards.
* Reason: Adherence to a predefined programming language and framework ensures consistency in development, maintenance, and support, aligning with project goals and technical guidelines.

1. Database Management System:

* The app must utilize a designated database management system (DBMS) for storing and managing word data and user information, in this case, MySql.
* Reason: Standardizing on a specific DBMS facilitates data consistency, security, and scalability, streamlining development efforts and ensuring compatibility with existing infrastructure.

1. Third-Party Libraries and APIs:

* The app may be constrained to use specific third-party libraries, APIs, or services for certain functionalities, such as authentication.
* Reason: Integration with predefined libraries and APIs can expedite development, leverage existing functionality, and ensure consistency in user experience and feature implementation.

## Glossary

[Present the noteworthy terms and their definition, format and validation rules if appropriate.]

# Deliverable 2

## Domain Model

|  |  |
| --- | --- |
| **WordInFrench Class:**  Attributes:   * id: Long (Primary Key) * wordInFrench: String | **WordInRomanian Class:**  Attributes:   * id: Long (Primary Key) * wordInRomanian: String |
| **Relationships**:  One-to-Many with **WordInRomanian** (Mapped by **translations** attribute) | **Relationships**:  Many-to-One with **WordInFrench** (Mapped by **wordInFrench** attribute) |

|  |  |
| --- | --- |
| **User Class:**  **Attributes**:   * **id**: Integer (Primary Key) * **username**: String (Unique) * **password**: String * **roles**: List of **Role** * **accountNonExpired**: Boolean (Default: true) * **accountNonLocked**: Boolean (Default: true) * **credentialsNonExpired**: Boolean (Default: true) * **enabled**: Boolean (Default: true) * **firstName**: String * **lastName**: String * **emailAddress**: String (Unique) * **birthdate**: LocalDate | **Role Class:**  **Attributes**:   * **id**: Integer (Primary Key) * **role**: String (Unique)   **Relationships**:   * Many-to-Many with **User** |

A screenshot of a computer

Description automatically generated

## 

## Architectural Design

### Conceptual Architecture

The application has been made using a layered architecture:

1. **Presentation Layer**:
   * This layer contains the controllers responsible for handling incoming HTTP requests, interpreting them, and invoking the appropriate service methods. Controllers are responsible for marshalling data between the HTTP request/response and the service layer.
2. **Service Layer**:
   * The service layer encapsulates the business logic of the application. It contains service classes that implement use cases and orchestrate interactions between different parts of the system.
3. **Repository Layer**:
   * The repository layer is responsible for interacting with the database or any other data storage mechanism. It contains repository interfaces and classes that provide CRUD (Create, Read, Update, Delete) operations for accessing and manipulating data entities.
4. **Model Layer**:
   * The model layer represents the domain-specific entities and data structures used by the application. It includes entity classes and other domain objects that model the core concepts of the system.

### Package Design

*A screenshot of a computer

Description automatically generated*

### Component and Deployment Diagram

A diagram of a web server

Description automatically generated

# Deliverable 3

## Design Model

### Dynamic Behavior

[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]

### Class Diagram

[Create the UML class diagram; apply GoF patterns and motivate your choice]

## Data Model

[Create the data model for the system.]

# System Testing

[Describe the testing methides and some test cases.]

# Future Improvements

[Present some features that apply to the application scope.]

# Conclusion

# Bibliography