Software Design Specification for;

Skill Language Exchange Platform

**Prepared by:**

**Ch.Arsha Akshara – Se22uari037**

**K.Pushpak Vishnu – Se22uari136**

**N.Mokshitha –Se22uari111**

**S.Apurupa – Se22uari156**

**Document Information**

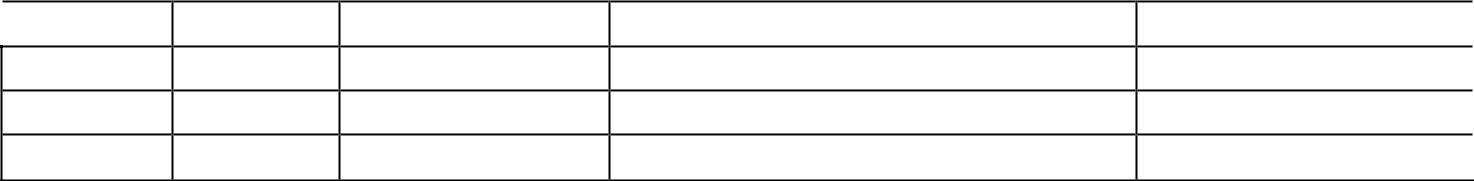
**Title: Project Manager:**

**Prepared By:**

**Document Version No: Document Version Date:**

**Preparation Date:**

**Version History**



**Ver. No. Ver. Date Revised By**

**Description**

**Filename**

**Table of Contents**

1. [INTRODUCTION 4](#_TOC_250025)
   1. [Purpose 4](#_TOC_250024)
   2. [Scope 4](#_TOC_250023)
   3. [Definitions, Acronyms, and Abbreviations 4](#_TOC_250022)
   4. [References 4](#_TOC_250021)
2. [USE CASE VIEW 4](#_TOC_250020)
   1. [USE CASE 4](#_TOC_250019)
3. [DESIGN OVERVIEW 4](#_TOC_250018)
   1. [Design Goals and Constraints 5](#_TOC_250017)
   2. [Design Assumptions 5](#_TOC_250016)
   3. [Significant Design Packages 5](#_TOC_250015)
   4. [Dependent External Interfaces 5](#_TOC_250014)
   5. Implemented Application External Interfaces 5
4. [LOGICAL VIEW 5](#_TOC_250013)
   1. [Design Model 6](#_TOC_250012)
   2. [Use Case Realization 6](#_TOC_250011)
5. [DATA VIEW 6](#_TOC_250010)
   1. [Domain Model 6](#_TOC_250009)
   2. [D ata Model (persistent data view) 6](#_TOC_250008)
      1. [Data Dictionary 6](#_TOC_250007)
6. [EXCEPTION HANDLING 6](#_TOC_250006)
7. [CONFIGURABLE PARAMETERS 6](#_TOC_250005)
8. [QUALITY OF SERVICE 7](#_TOC_250004)
   1. [Availability 7](#_TOC_250003)
   2. [Security and Authorization 7](#_TOC_250002)
   3. [Load and Performance Implications 7](#_TOC_250001)
   4. [Monitoring and Control 7](#_TOC_250000)

# Introduction

### Purpose

The purpose of this Software Design Specification is to outline the technical and architectural design of the Skill Exchange Platform. The system allows users to connect and exchange knowledge or skills in a mutually beneficial manner. This document aims to guide the development team in implementing the core functionalities such as Language Exchange, Peer-to-Peer Tutoring, and Skill Bartering. The main audience includes developers, testers, and academic supervision

### Scope

This project focuses on building a web-based platform for university students to exchange skills. The three core features include:

Language Exchange: Connecting users interested in learning and teaching different languages.

Peer-to-Peer Tutoring: Facilitating academic assistance among students.

Skill Exchange: Implementing a bartering system where users are matched only when both want the other’s skill, ensuring fair and meaningful exchange.

The scope includes frontend and backend development, user authentication, and matching logic based on mutual interest.

### Definitions, Acronyms, and Abbreviations

### Skill Exchange – A system that allows users to trade skills only when both parties benefit.

### P2P – Peer-to-Peer.

### TenderMax-like System – A mutual-matching mechanism based on reciprocal requirements.

### UX – User Experience.

### References

### University Guidelines for Project Documentation.

### IEEE Software Design Standards.

### TenderMax (as a conceptual reference for the matching logic).

# 

# Use Case View

This section provides an overview of the significant use cases for the **Skill Language Exchange Platform**. The platform aims to connect individuals who wish to learn new languages with native or fluent speakers of those languages through a structured skill exchange model. Users can register as learners, tutors, or both, and engage in language exchange sessions based on mutual availability and preferences.

The core functionalities of the system include:

* User registration and authentication
* Profile management with language proficiency indicators
* Search and match mechanism for language exchange partners
* Scheduling and conducting exchange sessions (via chat, video, or audio)
* Review and rating system post-session
* Admin panel for monitoring and moderation

### Use Case

#### 

#### ****UC01 – User Registration and Login****

* **Description:** Allows new users to sign up as learners or tutors (or both) and existing users to log in securely.
* **Actors:** Learner, Tutor
* **Steps:**
  1. User accesses the platform.
  2. Provides necessary details (email, password, languages).
  3. Receives confirmation and gains access.

#### ****UC02 – Profile Management****

* **Description:** Enables users to manage their profile, including languages known, learning goals, proficiency levels, and availability.
* **Actors:** Learner, Tutor
* **Steps:**
  1. User navigates to profile.
  2. Edits personal and language information.
  3. Saves changes.

#### ****UC03 – Matchmaking****

* **Description:** Suggests potential language partners based on language preferences and availability.
* **Actors:** Learner, Tutor
* **Steps:**
  1. User searches or browses partners.
  2. System filters based on criteria.
  3. User selects a match and sends request.

#### ****UC04 – Session Scheduling and Communication****

* **Description:** Allows users to schedule language exchange sessions and conduct them via chat, voice, or video.
* **Actors:** Learner, Tutor
* **Steps:**
  1. Users agree on a time.
  2. Platform facilitates the session using integrated tools.
  3. Session ends and is logged.

#### ****UC05 – Feedback and Ratings****

* **Description:** After each session, users can rate and review their partners to help others in the community.
* **Actors:** Learner, Tutor
* **Steps:**
  1. User selects session history.
  2. Provides a rating and optional comments.
  3. Submission is stored for others to see.

# Design Overview

This section provides an overview of the entire software design for the Skill Exchange Platform. The platform is designed to connect learners and mentors where users can register, search for skills, book sessions, reschedule, cancel, and provide feedback for completed sessions. The system focuses on providing an interactive and user-friendly environment while managing users, skills, sessions, and feedback efficiently.

**3.1 Design Goals and Constraints:**

Design Goals: Provide an easy-to-use platform for learners to search, book, and attend sessions from

mentors.

Ensure secure registration, login, and profile management.

Allow flexibility for rescheduling, cancelling, or marking sessions as missed.

Provide smooth communication between learners and mentors through notifications.

Ensure modular design for easy maintenance and future upgrades.

Constraints:

Internet connection is mandatory for accessing the platform.

Mentor availability depends on their schedule.

The platform should be responsive and accessible across multiple devices (desktop and mobile).

Session duration and timings are based on mentor availability.

User registration requires valid email and phone number

### 3.2 Design Assumptions:

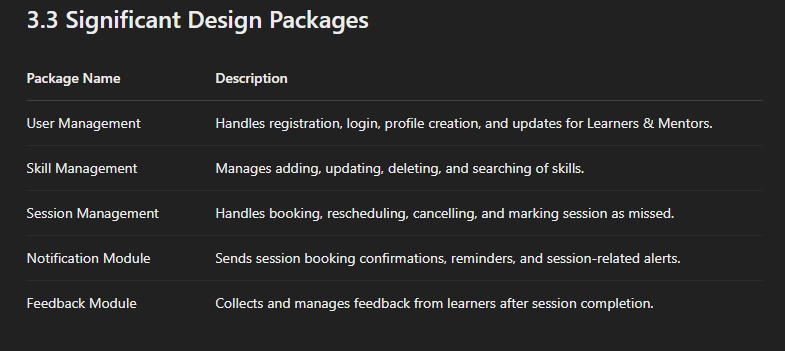
### Users (Learners & Mentors) have basic knowledge of using online platforms.

### Mentors will update their session availability regularly.

### Users will provide genuine feedback after sessions.

### Notifications will reach users without delay through Email/SMS services.

### Significant Design Packages



### 3.4 Dependent External Interfaces

### 

### 3.5 Implemented Application External Interfaces (and SOA web services)

### 

# Logical View

This section explains the internal structure of the Skill Exchange Platform. It describes the application architecture in terms of layers, modules, classes, and their interactions to perform key use cases. Each layer and module has a specific responsibility in the system and works collaboratively to ensure smooth platform functionality.

# 4.1 Design Model

# The design model for Skill Exchange Platform follows a layered architecture consisting of:

# Layers in the Architecture:

# 1. Presentation Layer (User Interface)

# 2. Business Logic Layer (Controllers & Services)

# 3. Data Access Layer (DAO Classes)

# 4. Database Layer

# 

### 

### 

### 

### 

### 

### 4.2 Use Case Realization

### This section explains how each of the key use cases described in Section 2 is implemented within the design of the Skill Language Exchange Platform. It highlights the interaction between modules, classes, and system components through sequence of operations to fulfill the desired functionality.

### UC01 – User Registration and Login

### Flow of Execution:

### User opens the application and selects Register/Login.

### UserController captures user details.

### UserService validates the input.

### UserDAO interacts with the database to store/retrieve user data.

### System confirms successful registration/login.

### Modules Involved:

### UserController

### UserService

### UserDAO

### Database

### UC02 – Profile Management

### Flow of Execution:

### User navigates to Profile Section.

### UserController fetches existing data.

### UserService allows update to language proficiency, availability, and learning goals.

### UserDAO updates the user information in the database.

### Changes are reflected in the user’s profile.

### Modules Involved:

### UserController

### UserService

### UserDAO

### Database

### UC03 – Matchmaking

### Flow of Execution:

### User searches for available partners.

### MatchmakingController receives search request.

### MatchmakingService filters users based on language preference & availability.

### UserDAO fetches potential partners from database.

### Matches are displayed to the user.

### Modules Involved:

### MatchmakingController

### MatchmakingService

### UserDAO

### Database

### UC04 – Session Scheduling and Communication

### Flow of Execution:

### User sends request for a session.

### SessionController handles scheduling request.

### SessionService verifies availability of both users.

### SessionDAO saves session details into database.

### Platform provides chat/video/audio tools for communication during the session.

### Session status updated post completion.

### Modules Involved:

### SessionController

### SessionService

### SessionDAO

### Database

### Communication Tools Integration

### UC05 – Feedback and Ratings

### Flow of Execution:

### User views completed sessions.

### FeedbackController provides feedback form.

### FeedbackService captures rating and comments.

### FeedbackDAO stores feedback into database.

### Feedback is visible for future reference.

### Modules Involved:

### FeedbackController

### FeedbackService

### FeedbackDAO

### Database

# Data View

# This section outlines the persistent data structure and the way information is stored and related in the system. It includes the domain model and data model for the Skill Exchange platform, providing a clear view of how users, skills, sessions, and matches are interconnected.

# 5.1 Domain Model

### 

### The domain model represents the core entities and their relationships within the system. It includes:

### User: Represents an individual registered on the platform.

### Skill: Represents a skill a user can offer or wants to learn.

### Exchange Type: Can be Language Exchange, Peer-to-Peer Tutoring, or Skill Bartering.

### Match: Represents a successful mutual agreement between two users.

### Session: Represents a scheduled meeting for an exchange.

### Review: Feedback given after an exchange session.

### Entity Relationships:

### A User can offer or request multiple Skills.

### A Skill can belong to many users (many-to-many via an association table).

### A Match is created only when both users want each other's offered skill (especially in the Skill Bartering system).

### A Session is linked to a Match.

### Review is linked to both Session and User.

### Data Model

### (Persistent Data View)

### The data model includes the following tables with key fields:

### Users (user\_id, name, email, password\_hash, bio, etc.)

### Skills (skill\_id, skill\_name, category)

### UserSkills (user\_id, skill\_id, type [offered/requested])

### Matches (match\_id, user1\_id, user2\_id, skill1\_id, skill2\_id, status)

### Sessions (session\_id, match\_id, date\_time, duration, location/URL)

### Reviews (review\_id, session\_id, reviewer\_id, rating, comments)

**5.2.1 Data Dictionary :**

Entity Attribute Description

User user\_id Unique identifier for each user

name Full name of the user

email User’s email address (used for login)

password\_hash Encrypted password for secure authentication

bio Short biography or profile description

Skill skill\_id Unique identifier for each skill

skill\_name Name of the skill (e.g., Python, French)

category Category of the skill (Language, Academic, Creative, etc.)

UserSkills user\_id Foreign key linking to the user

skill\_id Foreign key linking to the skill

type Type of skill relation – 'offered' or 'requested'

Match match\_id Unique identifier for a match

user1\_id ID of the first user in the match

user2\_id ID of the second user in the match

skill1\_id Skill offered by user1

skill2\_id Skill offered by user2

status Status of the match (pending, confirmed, completed)

Session session\_id Unique identifier for an exchange session

match\_id Foreign key linking to the match

date\_time Scheduled date and time of the session

duration Length of the session in minutes/hours

Review review\_id Unique ID for each review

session\_id Foreign key linking to the session being reviewed

reviewer\_id ID of the user giving the review

rating Numerical rating (e.g., 1 to 5 stars)

comments Text feedback or comments

# Exception Handling

This section describes exceptions defined in the Skill Exchange Platform, their triggers, handling mechanisms, logging practices, and follow-up actions.

**6.1 Defined Exceptions and Circumstances**

Core Functional Exceptions

AuthenticationFailedException:

Trigger: Invalid credentials during login.

UnauthorizedAccessException:

Trigger: Unauthorized attempt to edit another user’s profile or access restricted features.

UserNotFoundException:

Trigger: Invalid user ID in requests (e.g., accessing a deleted profile).

Skill Bartering Exceptions

MatchingFailedException:

Trigger: Algorithm failure to find skill matches after maximum retries.

MutualInterestNotFoundException (Custom Exception):

Trigger: No mutual interest found during skill/language matching (handled as a user-facing message, not an error).

Peer-to-Peer Tutoring Exceptions

TutorNotFoundException:

Trigger: Requested tutor profile not found (e.g., tutor deactivates account).

InvalidTutoringRequestException:

Trigger: Malformed request (e.g., invalid time slot, missing subject).

Data & Infrastructure Exceptions

DatabaseOperationFailedException:

Trigger: CRUD operation failures (e.g., connection timeout, deadlock).

DataValidationException:

Trigger: Invalid input (e.g., empty fields, invalid email format).

ExternalServiceUnavailableException:

Trigger: Failure to communicate with third-party services (e.g., payment gateway, SMS API).

TooManyRequestsException :

Trigger: Rate limits exceeded (e.g., spam matching requests).

**6.2 Exception Handling Strategies**

Recovery Mechanisms

ExternalServiceUnavailableException:

Retry with exponential backoff (3 attempts).

Enable circuit breaker to temporarily disable the service after repeated failures.

DatabaseOperationFailedException:

Fallback to cached data (e.g., skill recommendations) in read-only mode.

ChatConnectionException :

Auto-reconnect WebSocket (5 attempts) or switch to email/SMS notifications.

User Communication

MutualInterestNotFoundException:

Display: “No matches found. Adjust your preferences or try later!”

DataValidationException:

Highlight invalid form fields (e.g., red borders for empty inputs).

TutorNotFoundException:

Suggest alternative tutors or prompt users to join a waitlist.

Transaction Management

Rollback database transactions for multi-step operations (e.g., tutoring bookings).

Use idempotency keys to prevent duplicate requests during retries.

**6.3 Exception Logging**

Log Structure

Fields: Timestamp, user ID (masked), exception type, error message, stack trace, HTTP method/URL (for APIs).

Data Masking:

Redact sensitive data (e.g., passwords, chat messages).

Comply with FERPA/GDPR for log retention (30 days for debug logs, 1 year for audit logs).

Tools

Centralized Logging: ELK Stack (Elasticsearch, Logstash, Kibana) or Sentry for error tracking.

Monitoring: Prometheus/Grafana dashboards to track exception rates and system health.

**6.4 Follow-Up Actions**

Alerting

Critical Alerts (e.g., DatabaseOperationFailedException):

Immediate Slack/email notifications to admins.

Non-Critical Alerts (e.g., DataValidationException):

Daily summary reports.

Resolution Workflows

Automated Retries: Background workers (Celery) reprocess failed matches/tutoring bookings.

Incident Reports: Document root causes (e.g., “AWS RDS outage caused DatabaseOperationFailedException on 2023-10-05”).

SLA: Resolve critical exceptions within 4 hours.

User Support

Status Page: Public dashboard for real-time updates on outages (e.g., chat service downtime).

Follow-Up Emails: Notify users when resolved (e.g., “Tutors are now available for Calculus!”).

**6.5 Testing & Validation :**

Unit Tests: Validate exception triggers (e.g., InvalidTutoringRequestException for missing time slots).

Chaos Testing: Simulate external service failures (e.g., disable SMS API to test ExternalServiceUnavailableException).

**6.6 API-Specific Handling:**

REST Error Responses

401 Unauthorized: AuthenticationFailedException.

409 Conflict: SessionBookingConflictException.

429 Too Many Requests: TooManyRequestsException.

# 

# 7 Configurable Parameters

# This section outlines the configurable parameters used in the ****Skill Language Exchange Platform****. These parameters allow for customization of system behavior to meet different user needs or deployment environments. Some parameters are dynamic and can be changed without restarting the application, while others require a restart.

# 

# 

# 

# 8 Quality of Service

# This section describes aspects of the design related to application availability, security, performance, and monitoring and control in production for the Skill Exchange Platform.

**8.1 Availability**

For the Skill Exchange Platform, availability considerations include:

Basic Uptime: The platform should aim for reasonable uptime during university hours and peak usage times.

Maintenance Windows: Planned maintenance for database updates or software deployments should be scheduled during off-peak hours to minimize disruption to users.

Error Handling: The application should be designed to gracefully handle errors and prevent cascading failures, ensuring that one component's issue doesn't bring down the entire platform.

Recovery Strategy (Basic): In case of a system failure, a basic plan for restoring the application to its last stable state should be in place (e.g., regular backups, simple restart procedures).

**8.2 Security and Authorization**

For the Skill Exchange Platform, security and authorization considerations include:

User Authentication: Secure methods for user registration and login (e.g., password hashing). Integration with the university's existing authentication system (if feasible and required) could be considered.

Role-Based Access Control (RBAC): Define basic user roles (e.g., student user) and ensure that users can only access features relevant to their role.

Data Privacy: Design the database to protect user data (profile information, exchanged skills, interaction history). Consider data encryption for sensitive information.

Authorization for Features:

Skill Exchange (Bartering): Only logged-in users should be able to view profiles, express interest in skill exchange, and engage in matching.

Language Exchange: Similar to skill exchange, only authenticated users should access this feature.

Peer-to-Peer Tutoring: Access to listing tutoring offers, requesting tutoring, and communication should be restricted to logged-in users.

Management of User Access: For this university-level project, user management might be limited to self-registration and profile management. Administrative interfaces for managing users might not be a primary focus but could be mentioned as a potential future enhancement.

Custom Security Logic:

Mutual matching requirement (like Tinder) ensures no user interaction occurs without bilateral consent.

Privacy settings allow users to hide specific skills or tutoring availability.

**8.3 Load and Performance Implications**

For the Skill Exchange Platform, load and performance implications include:

Expected Concurrent Users: Estimate the potential number of concurrent users based on the university's student population and anticipated adoption rates.

Skill Exchange Interactions: Consider the potential volume of users browsing profiles and initiating skill exchange requests. The matching algorithm's efficiency will be important.

Language Exchange Interactions: Estimate the number of users participating in language exchange and the frequency of their interactions.

Peer-to-Peer Tutoring Requests: Consider the number of students offering and requesting tutoring services.

Database Performance: Design database queries to efficiently handle user searches, skill matching, and communication data retrieval. Consider indexing strategies for frequently accessed data.

Response Times: Aim for reasonable response times for key user actions (e.g., profile loading, search results, initiating exchange requests) to ensure a positive user experience.

Scalability Considerations (Basic): While large-scale scalability might not be a primary concern, the design should consider basic principles that would allow for future growth if needed (e.g., modular design).

Design Implications:

Caching (Redis) for frequent queries (e.g., skill recommendations, match lists).

Asynchronous processing (e.g., Celery) for matchmaking algorithms to avoid UI delays.

Database indexing on user IDs and skill tags to optimize search.

**8.4 Monitoring and control**

For the Skill Exchange Platform, monitoring and control considerations include:

Basic Logging: Implement logging to record important application events, errors, and user actions. This will be helpful for debugging and understanding application behavior.

Application Health Checks: Implement basic health checks to verify if the core components of the application (e.g., web server, database connection) are functioning correctly.

Performance Monitoring (Basic): While detailed performance monitoring might be beyond the scope, consider tracking basic metrics like request response times and error rates through logs.

Error Reporting: Implement a mechanism to report errors to developers or administrators for investigation and resolution.

Controllable Processes-

Background Workers: Matchmaking daemons for skill/language exchanges. Reminder bots for scheduled tutoring sessions (email/notification triggers).

Housekeeping Tasks: Weekly archiving of inactive chat logs. Monthly cleanup of unverified accounts.

Monitoring Metrics-

Performance: API response times, server CPU/memory usage.

User Activity: Daily matches made, active tutoring sessions, language exchange chat volume.

Alerts: Server downtime, failed login attempts, or database connection errors (via tools like Prometheus/Grafana).