SkillSwap: A Peer-to-Peer Skill Exchange Platform

Submitted by

Pranathasree - CB.PS.I5DAS22135

Sayana S Nair - CB.PS.I5DAS22144

Krishna Midula K - CB.PS.I5DAS22127

Sri Sruthi M N - CB.PS.I5DAS22157

Problem Definition

In many academic environments, students possess valuable skills but lack an effective platform to share and exchange them with peers. Existing learning platforms are often monetised, externally driven, or focus on one-way content delivery rather than peer-to-peer interaction. This results in underutilization of internal student expertise, limited collaborative learning, and unequal access to mentorship.

SkillSwap addresses this problem by providing a peer-to-peer skill exchange platform where students can both teach and learn skills within a university ecosystem. Instead of monetary transactions, the platform uses a token-based reward system, ensuring fairness and encouraging active participation. The system also integrates ML-based mentor recommendations to simplify mentor discovery and improve learning outcomes.

Agile Methodology and Roles

Member Name	Roll Number	Agile Role
Sri Sruthi M N	CB.PS.I5DAS22157	Scrum Master
Sayana S Nair	CB.PS.I5DAS22144	Product Owner
Pranathasree	CB.PS.I5DAS22135	Development Team Member
Krishna Midula K	CB.PS.I5DAS22127	Development Team Member

The project follows the **Agile Scrum methodology**, chosen for its adaptability, iterative development approach, and emphasis on continuous feedback.

Agile Roles

- **Scrum Master (Sri Sruthi M N):**
  - Facilitates Agile ceremonies
  - Ensures adherence to Scrum practices
  - Coordinates inter-module dependencies
  - Maintains the common project documentation
  - Also contributes as a development team member
- **Product Owner (Sayana S Nair:**
  - Defines and prioritises requirements
  - Maintains the product backlog
  - Ensures features align with user needs
  - Validates completed functionality
  - Also contributes to development
- **Development Team Members (Pranathasree and Krishna Midula K):**
  - Implement assigned modules
  - Participate in sprint planning, reviews, and retrospectives
  - Collaborate with Scrum Master and Product Owner

Modularisation and Module Ownership

The system is modularised to ensure separation of concerns, scalability, and parallel development.

S. No.	Module	Description	Owner
1	User Management	Registration, login, authentication	Pranathasree
2	Skill Management	Skill CRUD, tagging, categorization	Pranathasree
3	Search and Discovery	Search, filters, trending skills	Sayana S Nair
4	Session Management	Session workflow & scheduling	Sayana S Nair
5	Matching and Recommendation	ML-based mentor recommendation	Sri Sruthi M N
6	Token and Rewards	Token Earning and Spending	Sri Sruthi M N
7	Ratings and Review	Feedback and Quality Control	Sri Sruthi M N
8	Notification	Alerts and Reminders	Krishna Midula K
9	Admin	Moderation and System Control	Krishna Midula K
10	Analytics and Reporting	Usage and Performance Analytics	Krishna Midula K

Module Description, Key Features and Tasks

1. User Management Module

Purpose

The User Management Module is responsible for handling user identity, authentication, and access control. It ensures that only authorised users can access the system and that users can maintain their profiles securely.

Key Features

- User registration and login
- JWT-based authentication
- Profile creation and editing
- Role management (Learner / Mentor / Admin)
- Password reset functionality

Tasks

- Validate user credentials during login
- Generate and verify JWT tokens

- Store and update user profile information
- Manage user roles and permissions
- Handle password encryption and reset requests

2. Skill Management Module

Purpose

The Skill Management Module manages the skills that users offer or want to learn. It forms the core data layer used by search and recommendation modules.

Key Features

- Add, edit, and remove skills
- Skill categorisation (Beginner / Intermediate / Advanced)
- Skill descriptions
- Skill tagging for better discovery

Tasks

- Store skill details in the database
- Link skills to users (offered/required)
- Validate skill data and prevent duplicates
- Update skill metadata on edits
- Provide skill data to search and ML modules

3. Search & Discovery Module

Purpose

The Search & Discovery Module allows users to manually explore available skills and mentors using keyword search and filters.

Key Features

- Keyword-based skill search
- Advanced filters (category, level, popularity)
- Trending skills
- Recently active mentors

Tasks

- Process search queries from users

- Apply filters and sorting to search results
- Fetch skill and mentor data efficiently
- Rank results based on relevance and popularity
- Display search results in a user-friendly format

4. Session Management Module

Purpose

The Session Management Module manages the complete learning workflow between learners and mentors.

Key Features

- Session request creation
- Accept or decline session requests
- Scheduling date and time
- Session status tracking (Pending, Confirmed, Completed)

Tasks

- Create and store session requests
- Update session status based on mentor actions
- Validate mentor availability
- Trigger notifications for session updates
- Track session lifecycle from request to completion

5. Matching & Recommendation Module (ML-Based)

Purpose

The Matching & Recommendation Module automatically suggests suitable mentors to learners using machine learning techniques.

Key Features

- Skill vectorisation (TF-IDF / Sentence-BERT)
- Cosine similarity-based matching
- Compatibility scoring
- Top-5 mentor recommendations

Tasks

- Preprocess skill text data
- Generate numerical representations of skills
- Compute similarity scores between users
- Rank mentors based on compatibility
- Provide recommendation results via API

6. Token & Rewards Module

Purpose

The Token & Rewards Module enables a fair, non-monetary exchange system where users earn tokens by teaching and spend tokens to learn.

Key Features

- Token earning for completed sessions
- Token spending for booking sessions
- Token wallet
- Token transaction history

Tasks

- Update token balances atomically
- Record token transactions
- Validate sufficient balance before spending
- Prevent duplicate or invalid transactions
- Provide token data for analytics

7. Ratings & Review Module

Purpose

The Ratings & Review Module maintains quality and trust by allowing learners to evaluate mentors after sessions.

Key Features

- Mentor rating (star-based)
- Written feedback
- Average rating display
- Spam prevention

Tasks

- Store ratings and reviews
- Validate review eligibility (completed sessions only)
- Calculate and update average mentor ratings
- Prevent multiple reviews for the same session
- Fetch reviews for display

8. Notification Module

Purpose

The Notification Module keeps users informed about important system events and updates.

Key Features

- Session request notifications
- Acceptance and rejection alerts
- Session reminders
- Email and in-app notifications

Tasks

- Create notification records
- Trigger notifications on system events
- Deliver notifications via supported channels
- Track read/unread notification status
- Avoid duplicate notifications

9. Admin Module

Purpose

The Admin Module provides system-level control and moderation capabilities.

Key Features

- Admin authentication
- View users and sessions
- Handle abuse and misuse reports
- Block or unblock users

Tasks

- Authenticate admin users
- Retrieve system-wide user and session data
- Process abuse reports
- Enforce administrative actions
- Coordinate with the notification and analytics modules

10. Analytics & Reporting Module

Purpose

The Analytics & Reporting Module provides insights into platform usage and system health.

Key Features

- User activity metrics
- Popular skills and categories
- Session success rates
- Token economy analytics
- Growth charts

Tasks

- Aggregate data from multiple modules
- Execute optimised analytical queries
- Generate datasets for visualisation
- Render charts using Chart.js
- Support decision-making for administrators

Requirements Engineering (Project-Level)

Inception

MEETING DETAILS

Date: November 24, 2025
Duration: 2 hours

Attendees:

- Development Team Member: Pranathsree (User Management & Skills)
- Product Owner: Sayana S Nair (Product Owner - Search & Sessions)
- Scrum Master: Sri Sruthi M N (Scrum Master - ML & Tokens)
- Development Team Member: Krishna Midula K (Notifications & Admin)

Initial Problem Statement

"Students possess valuable skills but have no structured platform to share knowledge with peers. Additionally, existing learning platforms are expensive and do not facilitate much collaborative learning among students."

Problem Breakdown

During the inception meeting, the team identified the following specific problems:

1. **Lack of Structured Peer Learning**
   - Students informally help each other, but without organisation
   - No way to discover who can teach what skills
   - Knowledge exchange happens randomly, not systematically
2. **Cost Barrier**
   - Commercial learning platforms (Udemy, Coursera, etc.) are expensive
   - Private tutoring costs ₹500-2000 per hour
   - Students cannot afford quality skill development
3. **Limited Collaboration**
   - Existing platforms are one-way (instructor → student)
   - No peer-to-peer interaction model
   - Students who are experts in one area but beginners in another have no platform
4. **No Recognition System**
   - Students who teach don't get recognised or rewarded
   - No rewards for knowledge sharing
   - Imbalanced exchange (some always teach, some always learn)

Stakeholder Identification

Stakeholder	Description	Interest in System
Learners	Students seeking to learn new skills	Learn new skills
Mentors	Students offering skills	Teach and earn rewards
University	Institution hosting the system	Safe and productive learning
Admin	System moderators	Oversight and misuse prevention
Development Team	Builders of the platform	Feasible and maintainable system

Stakeholder	Description	Interest in System
Faculty Advisor	Project guide/ supervisor	Academic oversight, guidance

Stakeholder Needs

Learners:

- Easy way to find mentors for desired skills
- Affordable/free learning option
- Sessions that fit into a busy or unpredictable school schedule
- Provision to look into the ratings/reviews of the mentor profiles

Mentors:

- Recognition for teaching efforts
- Ability to showcase expertise
- Easier session management

University:

- Enhance student skill development
- Encourage a collaborative culture
- Low-cost implementation
- Data on student interests and skills

System Scope Definition

In-Scope

- Peer-to-peer skill exchange
- Token-based reward system
- ML-based mentor recommendation
- Session scheduling & management
- Ratings, reviews, analytics

Out-of-Scope

- Monetary payments
- Cross-university access (future scope)
- Video conferencing (uses external tools)
- Certification issuance (future scope)

Elicitation

Requirements were gathered using:

- Collaborative requirements gathering by discussions among team members
- Analysis of similar platforms
- Simulated user personas using ChatGPT

- Quality Function Deployment (QFD)

Collaborative Requirements Gathering

Date: November 26, 2025
Participants: All 4 team members
Duration: 3 hours

Each team member independently identified:

- **Objects** (system entities)
- **Services** (system functions)
- **Constraints** (limitations and rules)
- **Performance Criteria** (measurable targets)

Consolidated Results

Objects Identified (17): User, Profile, Skill, Session, Token, Wallet, Transaction, Review, Rating, Search Query, Filter, Booking, Recommendation, Notification, Report, Admin, Analytics

Services Identified (20): Register User, Login, Reset Password, Add Skill (Teach), Add Skill (Learn), Search Skills, Filter Results, Request Session, Accept/Decline Session, Complete Session, Generate Recommendations, Calculate Similarity, Earn Tokens, Spend Tokens, Submit Review, View Reviews, Send Notification, Submit Report, Block User, View Analytics

Key Constraints (9):

- University email required
- Password must be a minimum of 8 characters with a special character
- Learners need ≥10 tokens to book sessions
- Sessions cannot be booked <2 hours in advance
- Maximum 5 pending requests per user
- One review per session
- Token transactions must be atomic
- Initial token allocation: 20 tokens
- Session cost: 10 tokens (fixed)

Performance Targets:

- Registration/Login: < 2 seconds
- Search results: < 2 seconds
- ML recommendations: < 3 seconds
- Notification delivery: < 2 seconds
- System uptime: 99%+

Simulated User Interviews (ChatGPT)

Date: November 27-28, 2025
Method: Structured ChatGPT interviews simulating 3 stakeholder types
Total Stakeholders: 3 (Learner, Mentor, Administrator)

Elicitation Technique

Simulated user interviews were conducted using ChatGPT to represent three primary stakeholder perspectives: learners, mentors, and administrators. Structured prompts were designed to elicit needs, concerns, and expectations without suggesting technical solutions.

User Needs (Derived from Simulated Interviews)

Learner Needs

From learner interview analysis, the following needs were identified:

1. Learners need clarity about whether a mentor is trustworthy and competent
2. Learners need a learning environment that is respectful and encouraging
3. Learners need flexibility in scheduling sessions and choosing how they learn
4. Learners need the effort exchanged between learners and mentors to feel fair
5. Learners need to feel safe and comfortable while asking questions and learning
6. Learners need mentors to be reliable and committed once a session is agreed upon
7. Learners need both system-guided suggestions and the freedom to choose mentors themselves

Mentor Needs

From mentor interview analysis, the following needs were identified:

1. Mentors need recognition for their teaching proportional to the effort invested
2. Mentors need evaluation methods that are fair and unbiased
3. Mentors need protection from learners misusing their time or effort
4. Mentors need a system that supports long-term participation without exhaustion
5. Mentors need visibility that reflects the quality of their contributions
6. Mentors need the platform to prioritise quality of teaching over quantity
7. Mentors need clear, consistent, and transparent feedback mechanisms

Administrator Needs

From the administrator interview analysis, the following needs were identified:

1. Administrators need an overall view of platform activities and usage
2. Administrators need to detect misuse or fraudulent behaviour early
3. Administrators need to ensure fairness and equal participation across users
4. Administrators need confidence that academic rules and ethics are followed
5. Administrators need evidence that the platform provides value to the institution
6. Administrators need clear mechanisms to intervene when violations occur
7. Administrators need to maintain oversight without interfering in daily user activity

Requirements Gathered by Analysis of Similar Platforms

Other Platform	Strength Adopted	Weakness Avoided	Requirement Extracted
Udemy	Strong rating system	Expensive, no peer interaction	5-star ratings + reviews for mentor sessions
Reddit	Free peer help	Unstructured, no accountability	Structured sessions, verified users
Tutoring Centers	Personalized guidance and one-on-one attention	Limited hours, in-person only	Extended hours, online + in-person
YouTube	Free, accessible	No personalization based on skills	ML-based recommendations

Collaborative Requirement Mapping

Elicitation Source	Identified Need	Resulting System Requirement	Target Module
Simulated Learner Interview	Learners need clarity about mentor competence	The system shall display mentor ratings and reviews for every mentor profile	Ratings & Review
Simulated Learner Interview	Learners need respectful and encouraging interactions	The system shall allow reporting of inappropriate behavior and enforce conduct rules	Admin, Notification
Simulated Learner Interview	Learners need flexibility in scheduling	The system shall allow users to select preferred dates and times for sessions	Session Management
Simulated Learner Interview	Learners need fair effort exchange	The system shall use a token-based mechanism for teaching and learning	Token & Rewards
Simulated Learner Interview	Learners need reliable mentor commitment	The system shall track session states	Session Management
Simulated Learner Interview	Learners need guidance plus freedom of choice For choosing mentors	The system shall support both manual search and automated recommendations	Search & Discovery, ML Recommendation
Simulated Mentor Interview	Mentors need recognition proportional to effort	The system shall issue tokens based on completed teaching sessions	Token & Rewards
Simulated Mentor Interview	Mentors need fair and unbiased evaluation	The system shall restrict one review per completed session	Ratings & Review
Simulated Mentor Interview	Mentors need protection from misuse of time	The system shall limit pending session requests per learner	Session Management
Simulated Mentor Interview	Mentors need long-term participation without exhaustion	The system shall limit sessions and requests per mentor	Session Management
Simulated Mentor Interview	Mentors need visibility based on quality	The system shall rank mentors using skill relevance and ratings	Recommendation
Simulated Mentor Interview	Mentors need transparent feedback	The system shall provide access to reviews and rating history	Ratings & Review
Simulated Administrator Interview	Administrators need overall view of platform activities and usage	The system shall provide dashboards showing usage and activity metrics	Analytics

Elicitation Source	Identified Need	Resulting System Requirement	Target Module
Simulated Administrator Interview	Administrators need early misuse detection	The system shall log token transactions and abnormal activity patterns	Analytics, Admin
Simulated Administrator Interview	Administrators need fairness monitoring	The system shall track token distribution and participation balance	Analytics
Simulated Administrator Interview	Administrators need academic integrity assurance	The system shall allow reporting and moderation of misuse	Admin
Simulated Administrator Interview	Administrators need evidence that the platform provides value to the institution	The system shall generate reports on engagement and learning trends	Analytics
Simulated Administrator Interview	Administrators need intervention mechanisms	The system shall allow blocking or restricting users	Admin
Similar Platform Analysis (Udemy)	Need for trust through feedback and ratings	The system shall implement a 5-star rating and review system	Ratings & Review
Similar Platform Analysis (Reddit)	Need for accountability	The system shall require verified users and structured sessions	User Management, Session Management
Similar Platform Analysis (Tutoring Centers)	Need for personalized guidance	The system shall support one-on-one scheduled mentoring sessions	Session Management
Similar Platform Analysis (YouTube)	Need for skill-based personalization	The system shall generate skill-based mentor recommendations	Recommendation

Quality Function Deployment (QFD)

Date: December 5, 2025
Duration: 4 hours
Participants: All 4 team members
Purpose: Translate customer needs into technical requirements and prioritise system components

Quality Function Deployment (QFD) was employed to maximise customer satisfaction by translating user needs into technical specifications. The team divided the requirements into three types as defined by QFD methodology:

QFD Requirement Classification

Normal Requirements (Explicitly Stated)

Requirements directly expressed during elicitation activities:

Normal Requirement	Source	Stakeholder Type
Display mentor ratings and reviews	Learner interview	Learner
Search for mentors by skill name	Learner interview	Learner
Schedule sessions with flexible time slots	Learner interview	Learner
Earn tokens after teaching sessions	Mentor interview	Mentor
View token balance and transaction history	Mentor interview	Mentor
Track session states (Pending/ Confirmed/Completed)	Mentor interview	Mentor
Report inappropriate behavior	Admin interview	Administrator
View analytics on platform usage	Admin interview	Administrator
Block or restrict users	Admin interview	Administrator
Generate skill popularity reports	Admin interview	Administrator

Expected Requirements (Implicit/Fundamental)

Requirements so fundamental they were not explicitly stated but are essential for customer satisfaction:

Expected Requirement	Purpose	Derived From
System response time < 2-3 seconds	Users expect instant feedback	Team Discussion
99%+ system uptime	Platform must be reliably available	Team Discussion
Secure password storage	Security is non-negotiable	Team Discussion
Atomic token transactions	Financial integrity	Team Discussion (constraints)
One review per session constraint	Prevents spam and maintains trust	Team Discussion (constraints)

Exciting Requirements (Beyond Expectations)

Features that exceed customer expectations and create satisfaction:

Exciting Requirement	Impact	Derived From
ML-powered personalized recommendations	Effortless mentor discovery	Team Discussion + addressing gaps of other existing platforms
Hybrid search + recommendation system	Enables both guided and manual discovery	Learner Interview
Token-based economy (no money)	Removes financial barrier entirely	Team Discussion

Exciting Requirement	Impact	Derived From
Quality-based mentor ranking	Rewards teaching efforts	Mentor Interview
Session commitment tracking	Ensures accountability	Learner Interview
Portfolio upload for mentors	Visual credibility proof	Team Discussion

Customer Voice Table

The Customer Voice Table summarises the needs and expectations expressed by different stakeholders during elicitation.

Customer Need (Voice)	Source
"I need to know if a mentor is trustworthy"	Learner Interview + Team Discussion (Review object)
"Learning should be affordable"	Learner Interview + Team Discussion (Token object)
"I need flexible scheduling"	Learner Interview + Team Discussion (Session object)
"I want my teaching effort recognized fairly"	Mentor Interview + Team Discussion (Token object + ML Ranking)
"Platform must be safe from harassment"	Admin interview + Team Discussion (Report, Admin objects)
"I want recommendations, not just search"	Team Discussion (Recommendation Object)
"System should feel supportive, not stressful"	Learner Interview
"I need to find mentors easily"	Learner Interview + Team Discussion (Search, Filter, Recommendation objects)
"Teaching quality should matter more than quantity"	Mentor Interview + Team Discussion (Rating Service)
"I need an overall view of platform activities and usage"	Admin interview + Team Discussion (Analytics object)
"Mentors should be reliable and committed"	Learner Interview + Team Discussion (Session states)
"Evaluation should be fair and unbiased"	Mentor Interview + Team Discussion (Review constraints)
"I need to avoid burnout from teaching"	Mentor Interview

Technical Requirements Identification

From customer needs, the following technical requirements were identified and mapped to project modules:

Customer Need	Technical Response	Target Module
Mentor competence and trustwothiness clarity	Ratings and reviews per mentor	Ratings & Review

Customer Need	Technical Response	Target Module
Respectful learning environment	Reporting system and admin intervention	Admin, Notification
Flexible scheduling	Session scheduling with constraints	Session Management
Fair effort exchange	Token-based system	Token & Rewards
Reliable mentor commitment	Session state tracking	Session Management
Guided + manual mentor choice	Hybrid search + ML recommendations	Search & Discovery, ML Recommendations
Fair evaluation	One review per session rule	Ratings & Review
Protection from misuse of time	Request and session limits	Session Management
Quality-based mentor rankings	Ranking using skills and ratings	ML Recommendation
Transparent feedback	Review history access	Ratings & Review
Overview of platform activities	Usage and activity dashboards	Analytics
Early misuse detection	Transaction logs and anomaly tracking	Admin, Analytics
Academic integrity	Reporting tools	Admin
Institutional value evidence	Engagement and trend reports	Analytics

Elaboration

Key actors (Learner, Mentor, Admin) and system entities were identified. Major workflows such as skill exchange, session management, and token usage were elaborated.

Conflict Resolution

During requirements elicitation and elaboration, conflicts arose due to **conflicting stakeholder expectations**, **resource constraints**, and **fairness expectations** among learners, mentors, and administrators.
Conflict resolution was carried out collaboratively through team discussions to ensure a balanced system design.

Identified Requirement Conflicts Based on Stakeholders' Needs after Elicitation

Conflict 1: Learner Flexibility vs Mentor Availability

Issue/Risk	Learner Perspective	Mentor Perspective
Issue	Learners want flexible scheduling of sessions	Mentors want control over which time the sessions are scheduled
Risk	Missed sessions, frustration	Burnout, overcommitment

Resolution Strategy

- Introduced that mentors could accept or decline session requests

- Added the session states such as ‘Pending’ and ‘Confirmed’
- The system allows maximum pending requests per user

Resolved Requirement

The system shall allow learners to request sessions while granting mentors full control to accept or decline requests.

Conflict 2: Fair Effort Exchange vs Free Learning

Issue/Risk	Learner Perspective	Mentor Perspective
Issue	Desire for low-cost/free learning	Effort must be recognized
Risk	Discouraged exploration (due to cost)	Burnout/Exploitation if free

Resolution Strategy

- Introduced a token-based system
- Fixed session cost and initial token allocation
- No real money involved

Resolved Requirement

The system shall use a token-based mechanism to ensure fairness in teaching and learning efforts, while ensuring no real money usage.

Conflict 3: Collaborative Learning vs. Academic Dishonesty

Issue/Risk	Learner/Mentor Perspective	Admin Perspective
Issue	Informal peer-to-peer learning platform where collaboration is easier	Admins worry that without oversight, this informal platform could be misused for cheating
Risk	Collaboration on solving assignments	If the platform is used for plagiarism, the institution faces legal and reputational liability.

Resolution Strategy

- Introduced a provision for reporting sessions that are misused
- Given the admins a high-level view of how tokens are being used, while ensuring no access to private conversations
- Logged session and token activities

Resolved Requirement

The system shall provide reporting and administrative controls to ensure ethical and academic integrity.

Conflict 4: Reputation vs. Privacy

Issue/Risk	Learner/Admin Perspective	Admin Perspective
Issue	Need to judge mentor quality	Mentors don't want to feel "exposed" or have their every move tracked
Risk	Collaboration on solving assignments	If the platform is used for plagiarism, the institution faces legal and reputational liability.

Resolution Strategy

- Reviews are restricted to one per completed session, which ensures that there is no spamming
- Aggregated ratings and overall feedback of the mentors are shown to the learners, rather than displaying for each and every session
- No public session transcripts

Resolved Requirement

The system shall display aggregated mentor ratings and reviews while preserving user privacy.

Conflict 5: Algorithmic Guidance vs. User Autonomy

Issue/Risk	The System Feature	Learner Preferance
Issue	To recommend the best mentors to learners for their goal based on compatibility scores using ML algorithms	Want guidance with choosing the best mentor for their goal but also may have some some specific preferences in mind like location, teaching style, etc.
Risk	Complete automation could sometimes hide the good mentors who do not fit the ‘ideal’ data profile	Without help, users get overwhelmed by too many options and could leave the platform.

Resolution Strategy

- The system combines manual search with filters along with ML-based recommendations
- Users retain the final choice

Resolved Requirement

The system shall support both automated mentor recommendations and manual search and selection.

Functional Requirements (FR)

Functional requirements describe what the system shall do.
All requirements below are derived from elicitation, collaborative analysis, QFD, and requirement mapping.

FR-UM: User Management Module

FR-UM-01
The system shall allow users to register using a valid university email address.

FR-UM-02
The system shall authenticate users using secure login credentials.

FR-UM-03
The system shall allow users to reset their passwords using a secure verification mechanism.

FR-UM-04
The system shall create and maintain a user profile containing role information (Learner/Mentor).

FR-UM-05
The system shall restrict access to features based on the user's assigned role.

FR-SK: Skill Management Module

FR-SK-01
The system shall allow mentors to add, edit, and remove skills they offer for teaching.

FR-SK-02
The system shall allow learners to specify skills they want to learn.

FR-SK-03
The system shall store skill details, including name, description and category.

FR-SK-04
The system shall prevent duplicate skill entries for the same user.

FR-SD: Search & Discovery Module

FR-SD-01
The system shall allow users to search for skills and mentors using a keyword-based search.

FR-SD-02
The system shall allow users to filter search results by skill category and rating.

FR-SD-03
The system shall display search results ranked based on relevance and popularity.

FR-SE: Session Management Module

FR-SE-01
The system shall allow learners to request a learning session with a selected mentor.

FR-SE-02
The system shall allow learners to select a preferred date and time for a session request.

FR-SE-03
The system shall validate mentor availability before creating a session request.

FR-SE-04
The system shall assign a status to each session request (Pending, Confirmed, Rejected, Completed).

FR-SE-05
The system shall allow mentors to accept or decline session requests.

FR-SE-06
The system shall prevent session booking less than two hours before the scheduled time.

FR-ML: Matching & Recommendation Module

FR-ML-01
The system shall generate personalised mentor recommendations based on learner skill preferences.

FR-ML-02
The system shall compute similarity between learner skills and mentor skills using vector-based techniques.

FR-ML-03
The system shall rank mentors based on compatibility score, skill relevance, and ratings.

FR-ML-04
The system shall return a maximum of five recommended mentors per learner request.

FR-ML-05
The system shall automatically update recommendations when learner or mentor skills change.

FR-TR: Token & Rewards Module

FR-TR-01
The system shall allocate an initial token balance to every newly registered user.

FR-TR-02
The system shall deduct tokens from learners when a session request is confirmed.

FR-TR-03
The system shall award tokens to mentors upon successful completion of a session.

FR-TR-04
The system shall maintain a transaction history for all token earn and spend operations.

FR-TR-05
The system shall ensure all token transactions are atomic.

FR-RR: Ratings & Review Module

FR-RR-01
The system shall allow learners to submit a rating and review after completing a session.

FR-RR-02
The system shall restrict learners to one review per completed session.

FR-RR-03
The system shall calculate and display an average rating for each mentor.

FR-RR-04
The system shall allow mentors to view feedback and rating history.

FR-NO: Notification Module

FR-NO-01
The system shall notify mentors when a learner requests a session.

FR-NO-02
The system shall notify learners when a mentor accepts or declines a session request.

FR-NO-03
The system shall support in-app notifications and optional email notifications.

FR-AD: Admin Module

FR-AD-01
The system shall allow administrators to view registered users and session activity.

FR-AD-02
The system shall allow administrators to block or restrict users for policy violations.

FR-AD-03
The system shall allow administrators to review and resolve reported issues.

FR-AN: Analytics & Reporting Module

FR-AN-01
The system shall provide administrators with dashboards showing platform usage statistics.

FR-AN-02
The system shall generate reports on session activity, skill popularity, and token distribution.

FR-AN-03
The system shall support exporting analytics reports for administrative review.

Non-Functional Requirements (NFR)

Non-functional requirements specify **how well the system shall perform** its functions. These requirements address performance, security, reliability, usability, scalability, and maintainability.

All NFRs below are derived from:

- Collaborative requirements gathering
- Consolidated constraints and performance targets
- QFD expected requirements
- Administrative and user expectations

NFR-PERF: Performance Requirements

NFR-PERF-01
The system shall complete user registration and login operations within 2 seconds for at least 95% of requests.

NFR-PERF-02
The system shall return skill search and discovery results within 2 seconds under normal load conditions.

NFR-PERF-03
The system shall generate and return mentor recommendations within 3 seconds per request.

NFR-PERF-04
The system shall deliver in-app notifications within 2 seconds of the triggering event.

NFR-REL: Reliability & Availability Requirements

NFR-REL-01
The system shall maintain an operational availability of at least 99% uptime, excluding scheduled maintenance.

NFR-REL-02
The system shall preserve session, token, and transaction data in the event of partial system failures.

NFR-REL-03
The system shall recover gracefully from service or database failures without data loss.

NFR-SEC: Security Requirements

NFR-SEC-01
The system shall require registration using a valid university email address.

NFR-SEC-02
The system shall store all user passwords in hashed and salted form.

NFR-SEC-03
The system shall authenticate users using JWT-based secure sessions.

NFR-SEC-04
The system shall restrict access to administrative functions to authorized users only.

NFR-SEC-05
The system shall ensure that token transactions are protected against unauthorized modification.

NFR-DATA: Data Integrity & Consistency Requirements

NFR-DATA-01
All token transactions shall be atomic, ensuring either full completion or rollback.

NFR-DATA-02
The system shall maintain consistency between session status, token balance, and transaction records.

NFR-DATA-03
The system shall prevent duplicate reviews for the same completed session.

NFR-USAB: Usability Requirements

NFR-USAB-01
The system shall provide a clear and intuitive user interface for learners, mentors, and administrators.

NFR-USAB-02
The system shall provide informative error messages when user actions violate constraints or rules.

NFR-USAB-03
The system shall minimise the number of steps required to perform common actions such as searching skills, requesting sessions, and submitting reviews.

NFR-SCAL: Scalability Requirements

NFR-SCAL-01
The system shall support concurrent access by users within a single university without performance degradation.

NFR-SCAL-02
The system architecture shall allow future expansion to multiple institutions without major redesign.

NFR-MAINT: Maintainability & Modularity Requirements

NFR-MAINT-01
The system shall follow a modular architecture to allow independent updates to system components.

NFR-MAINT-02
The system shall expose REST-based APIs to support maintainability and extensibility.

NFR-MAINT-03
The system shall maintain logs for debugging, monitoring, and audit purposes.

NFR-COMP: Compliance & Ethical Requirements

NFR-COMP-01
The system shall enforce rules that prevent academic integrity violations, such as misuse for exam or assignment solving.

NFR-COMP-02
The system shall allow administrators to intervene in cases of reported misconduct.

NFR-COMP-03
The system shall ensure fair participation and prevent dominance or misuse by individual users.

Constraints & Business Rules

Constraints and business rules define limitations, policies, and conditions under which the system must operate.
These rules are non-negotiable, derived from collaborative requirements gathering, team discussions, and institutional policies.

They ensure **fairness, security, consistency, and system integrity**.

System Constraints

System constraints restrict how the system can be used or implemented.

Authentication & Identity Constraints

CON-AUTH-01
Users shall be allowed to register only using a valid university email address.

CON-AUTH-02
User passwords shall have a minimum length of 8 characters and include at least one special character.

Token Economy Constraints

CON-TOK-01
A learner shall possess at least 10 tokens before submitting a session request.

CON-TOK-02
Each completed learning session shall have a fixed cost of 10 tokens.

CON-TOK-03
Every newly registered user shall receive an initial allocation of 20 tokens.

CON-TOK-04
Token transactions shall be atomic, ensuring either full completion or rollback.

Session Scheduling Constraints

CON-SES-01
Sessions shall not be booked less than 2 hours before the scheduled start time.

CON-SES-02
A user shall not have more than 5 pending session requests at any given time.

CON-SES-03
Mentors shall not be allowed to accept overlapping session requests.

Review & Feedback Constraints

CON-REV-01
A learner shall be allowed to submit only one review per completed session.

CON-REV-02
Reviews shall be permitted only after a session is marked as completed.

Business Rules

Business rules define policies that govern platform behaviour and ensure alignment with fairness and academic values.

Participation & Fairness Rules

BR-FAIR-01
Teaching effort and learning effort shall be balanced using a token-based exchange mechanism.

BR-FAIR-02
Mentors shall be rewarded only for successfully completed sessions.

BR-FAIR-03
The platform shall prioritise quality of teaching over quantity.

Conduct & Safety Rules

BR-COND-01
Users shall adhere to respectful and appropriate conduct during all platform interactions.

BR-COND-02
The system shall allow users to report inappropriate behaviour.

BR-COND-03
Administrators shall have the authority to investigate and act on reported violations.

Academic Integrity Rules

BR-ACAD-01
The platform shall not be used for exam cheating or direct assignment completion.

BR-ACAD-02
Sessions identified as violating academic integrity shall be reviewed, and disciplinary action shall be taken.

Administrative Control Rules

BR-ADMIN-01
Administrators shall have the authority to block or restrict users violating platform policies.

BR-ADMIN-02
Administrative intervention shall be proportionate and logged for accountability.

Identification of Key Actors and System Entities

Key Actors

Based on elicitation and the technical requirements, the following key actors were identified:

Actor	Description
Learner	A student who uses the platform to learn skills
Mentor	A student who offers skills and conducts learning sessions
Administrator	University-authorized user responsible for oversight

These actors represent all external stakeholders interacting with the system.

System Entities (Objects)

From collaborative requirements gathering, the following core system entities were identified:

- User

- Profile
- Skill
- Session
- Token
- Wallet
- Transaction
- Review
- Rating
- Search Query
- Filter
- Booking
- Recommendation
- Notification
- Report
- Admin
- Analytics

These entities form the conceptual data model of the system and directly helped in the design of the Class Diagram.

System Services

The following system services were identified to support the requirements:

Register User, Login, Reset Password, Add Skill (Teach), Add Skill (Learn), Search Skills, Filter Results, Request Session, Accept/Decline Session, Complete Session, Generate Recommendations, Earn Tokens, Spend Tokens, Submit Review, Send Notification, Submit Report, Block User, View Analytics.

These services were mapped to functional requirements and later refined into UML interactions.

UML Modelling — System Level

After identifying actors, system entities, and services, UML diagrams were developed to model:

- System behaviour (Use Case, Activity, Sequence)
- System structure (Class Diagram)
- System lifecycle control (State Transition Diagram)

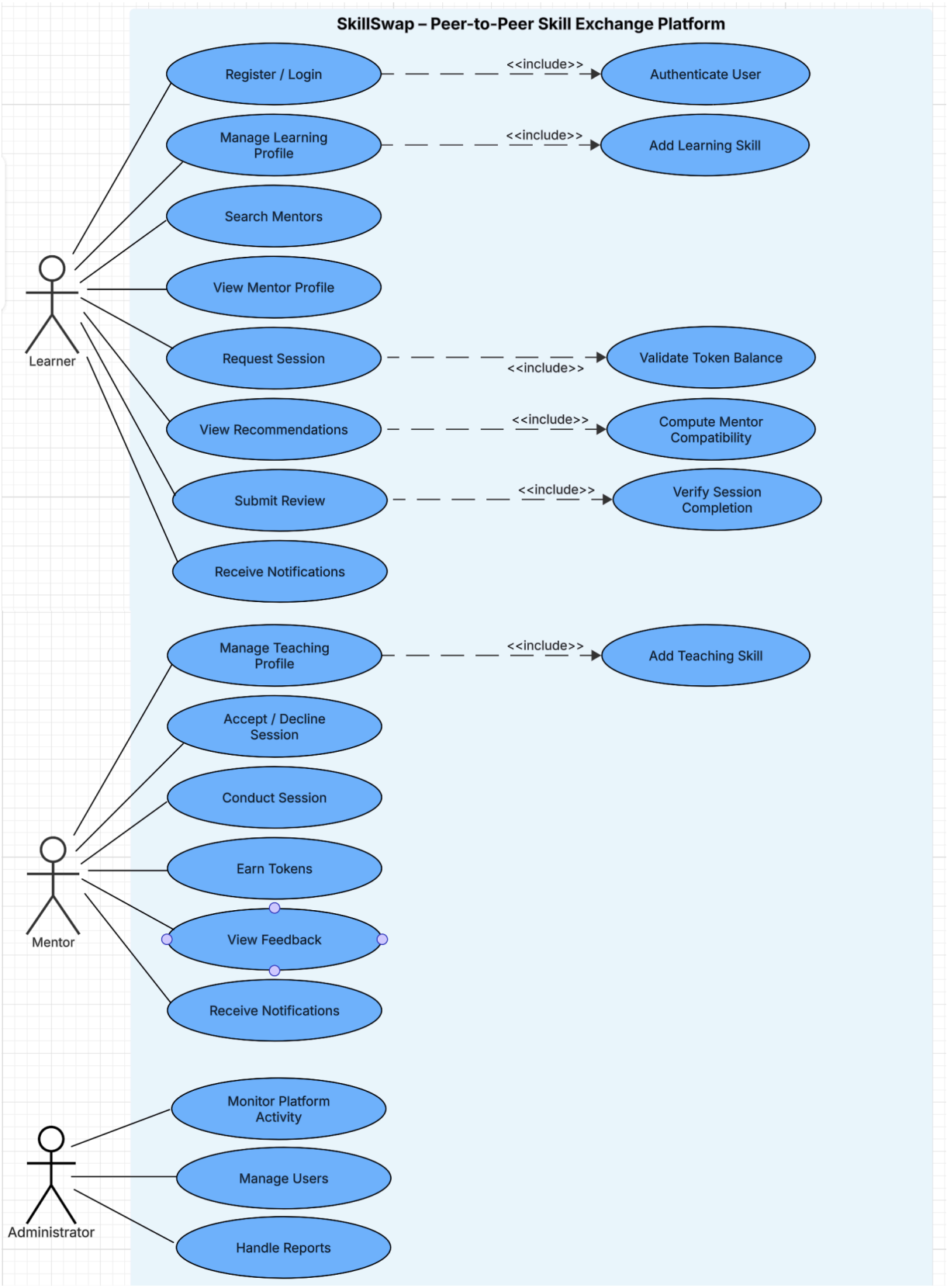
Use Case Diagram

Purpose

The **Use Case Diagram** represents the **functional scope of the SkillSwap system** and shows how different stakeholders interact with the system.
It captures what the system does.

This diagram is derived directly from:

- Functional Requirements (FRs)
- Constraints & Business Rules
- Collaborative requirement mapping



Activity Diagram (Session Request Workflow)

Purpose

The Activity Diagram models the end-to-end workflow for requesting a learning session, highlighting decision points, validations, and constraints.
It captures *business logic* and *control flow* for the most critical interaction in SkillSwap.

Derived from:

- FR-SE (Session Management)
- Token constraints (CON-TOK)
- Scheduling constraints (CON-SES)

Workflow Description

1. Learner initiates a session request.
2. System verifies learner authentication.
3. Learner selects a mentor.
4. System checks learner token balance (≥ 10 tokens).
5. Learner selects preferred date and time.
6. System validates:
    - Mentor availability
    - Booking is ≥ 2 hours in advance
    - Pending request limit not exceeded
7. If validation passes:
    - Session is created with a Pending status
    - Notification is sent to the mentor
8. If validation fails:
    - An appropriate error message is shown
    - Workflow terminates

Diagram - to be added

Sequence Diagram (Session Request)

Purpose

The Sequence Diagram illustrates time-ordered interactions between system components when a learner requests a session.
It emphasises validation order, service coordination, and state transitions.

Derived from:

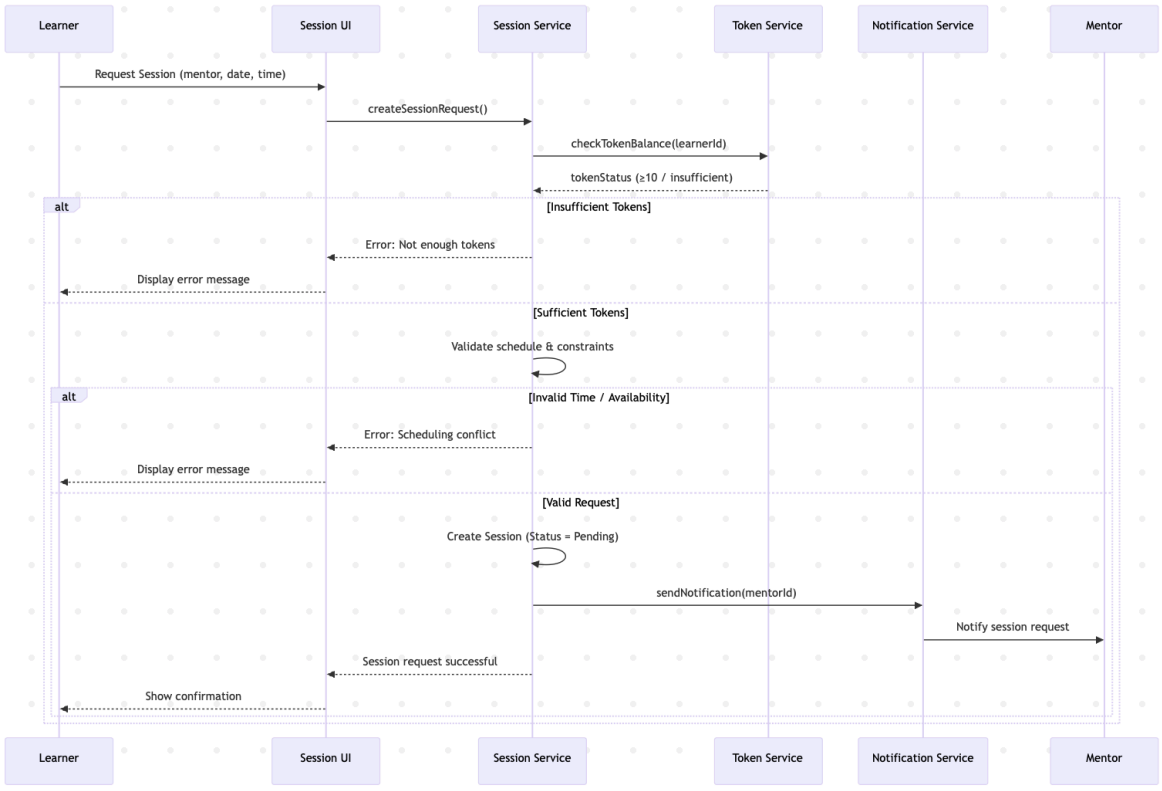
- Activity Diagram (Session Request)
- Functional Requirements (Session Management, Token Module)
- Constraints (token limit, time window, pending request limit)

Participating Objects

- **Learner** – Initiates the session request
- **Session UI** – Frontend interface
- **Session Service** – Core session logic
- **Token Service** – Validates token balance
- **Mentor** – Recipient of the request
- **Notification Service** – Sends alerts

Interaction Flow

1. Learner submits a session request via the UI.
2. Session Service validates learner authentication.
3. Token Service checks if learner has ≥ 10 tokens.
4. Session Service validates scheduling constraints.
5. Session is created with a Pending status.



6. Notification Service alerts the mentor.
7. Confirmation is returned to the learner.

Class Diagram

Purpose

The Class Diagram identifies:

- Core entities (classes)
- Their attributes
- Key operations
- Relationships (association, aggregation, dependency)

This diagram consolidates:

- Objects identified during collaborative requirements gathering
- Services identified during elicitation
- Constraints enforced across modules

Key Design Decisions

- One User can act as *Learner*, *Mentor*, or *Admin* (role-based design)
- Tokens and Sessions are transaction-controlled
- Reviews are strictly one per session
- Recommendations depend on Skills, Ratings, and Activity
- Analytics and Admin modules observe but do not modify core entities directly

Core Classes & Responsibilities

User

- Base entity for all platform users

Profile

- Stores extended user information

Skill

- Represents teachable/learnable skills

Session

- Represents a scheduled mentoring interaction

TokenWallet & TokenTransaction

- Handles token economy with atomic transactions

Review & Rating

- Maintains trust and credibility

Recommendation

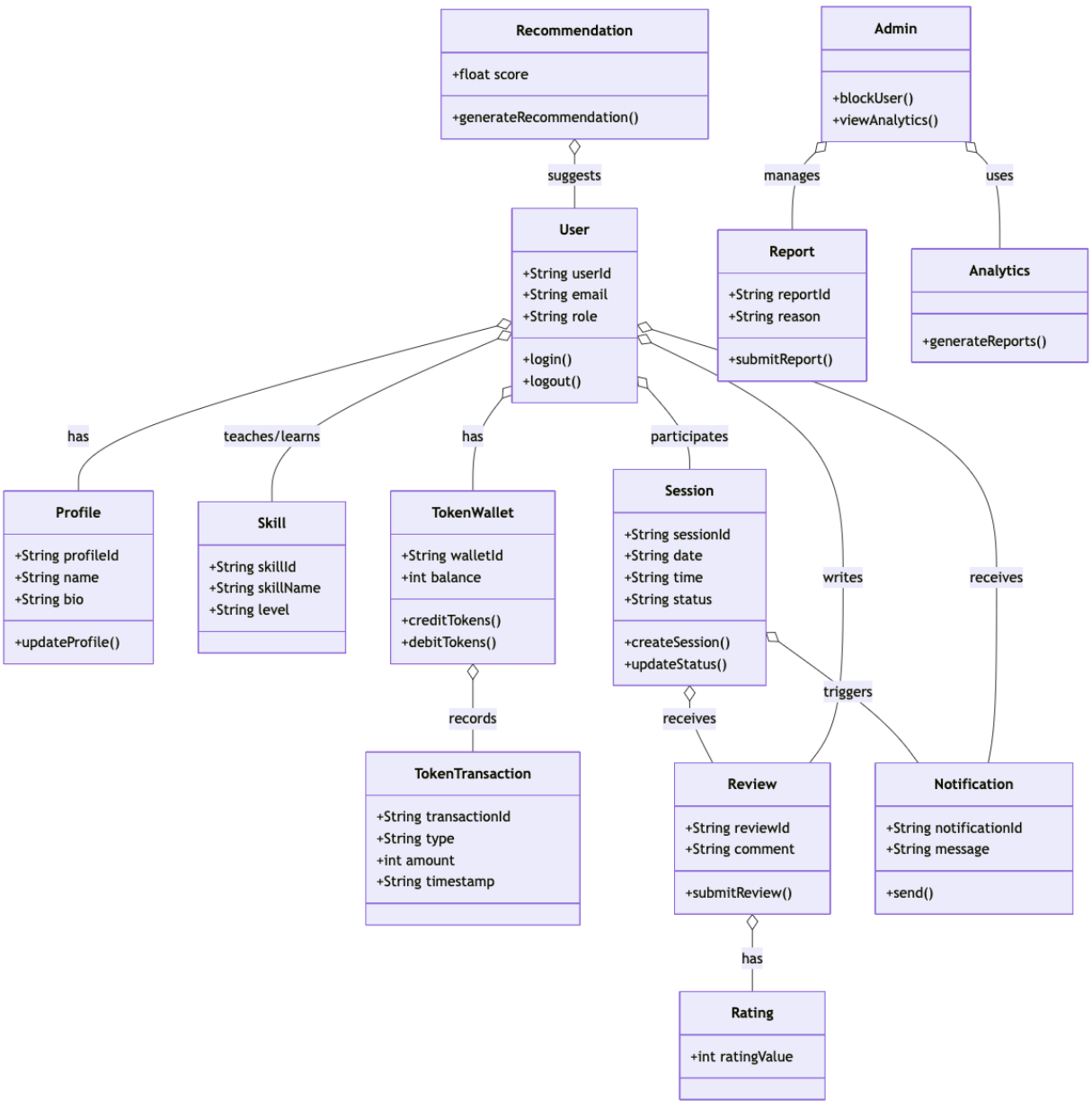
- Represents ML-based mentor matching

Notification

- Handles system alerts

Report & Admin

- Governance and moderation



State Transition Diagram (Session Lifecycle)

Purpose

The State Transition Diagram models the lifecycle of a mentoring session from creation to completion.
It shows:

- All possible states of a session
- Valid transitions between states
- Events and conditions triggering each transition

This diagram is especially important because it enforces:

- Mentor commitment
- Learner fairness
- Token integrity
- One-review-per-session constraint

States Identified (From Requirements & Constraints)

State	Meaning
Requested	Learner has sent a session request
Pending	Awaiting mentor action
Confirmed	Mentor has accepted the request
Rejected	Mentor declined the request
Cancelled	Session cancelled by learner/mentor
Completed	Session successfully conducted
Reviewed	Learner has submitted review

Key Constraints Enforced via States

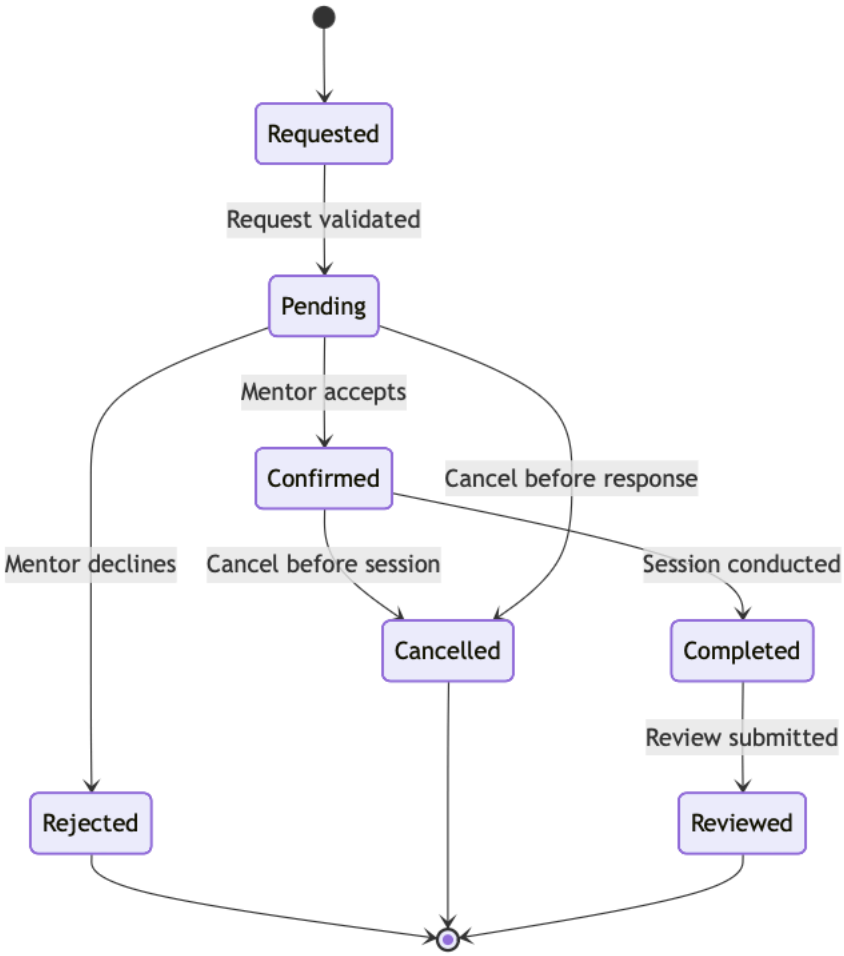
- Session **cannot jump** directly from Pending → Completed
- Only **Confirmed sessions** can be Completed
- Only **Completed sessions** can be Reviewed
- Only **one review per session**
- Token transfer happens **only on Completed**

State Transition Flow

1. Learner submits a session request → **Requested**

2. System validates tokens & availability → **Pending**
3. Mentor:
    - Accepts → **Confirmed**
    - Declines → **Rejected**
4. Either party may cancel (before start time) → **Cancelled**
5. Confirmed session occurs → **Completed**
6. Learner submits feedback → **Reviewed** (final state)

The State Transition Diagram ensures strict control over session flow, preventing invalid transitions such as completing or reviewing a session that was never confirmed. This directly supports fairness, accountability, and token integrity requirements identified during elicitation.



Negotiation

During requirements negotiation, the team evaluated competing stakeholder requirements against project constraints such as academic scope, implementation feasibility, fairness, and system complexity.
Negotiation ensured that the final requirement set was balanced, achievable, and aligned with objectives.

Negotiation Factors Considered

The following factors guided negotiation decisions:

- Academic project time constraints
- Fairness among learners and mentors
- System complexity and maintainability
- Institutional policies and ethics
- User experience and adoption

Negotiated Requirements and Decisions

Negotiation 1: Advanced ML vs Simpler Recommendation Model

Aspect	Consideration
Requested	Highly advanced AI recommendation
Constraint	Academic scope, limited training data
Decision	Use TF-IDF / SBERT with cosine similarity
Justification	Balances personalization with feasibility

Negotiated Outcome

A lightweight, explainable recommendation model was selected to ensure accuracy without excessive complexity.

Negotiation 2: Free Sessions vs Fair Compensation

Aspect	Learner	Mentor
Preference	Free learning	Recognition for effort
Decision	Token-based economy	
Justification	Ensures fairness without monetary exchange	

Negotiation 3: Unlimited Requests vs Platform Stability

Aspect	Consideration
Requested	Unlimited session requests
Risk	Spam, mentor overload
Decision	Maximum 5 pending requests per user

Justification	Prevents misuse and burnout

Negotiation 4: Full Transparency vs User Privacy

Aspect	Consideration
Requested	Detailed activity visibility
Risk	Privacy violation
Decision	Aggregated ratings and analytics only
Justification	Protects privacy while maintaining trust

Negotiation 5: Feature Richness vs Academic Feasibility

Aspect	Consideration
Requested	Mobile app, live video
Constraint	Time and scope
Decision	Web-based system with future extensibility
Justification	Focus on core learning workflow

Deferred Requirements

Some requirements were intentionally deferred to future phases:

Deferred Feature	Reason
Live video sessions	High complexity
Mobile application	Out of current scope
AI-based fraud detection	Requires real usage data

Outcome of Negotiation

- Core stakeholder needs retained
- System complexity kept manageable
- Fairness enforced through constraints
- Clear distinction between core and future features

Negotiation ensured a realistic, high-value requirement set without compromising system integrity or academic feasibility.

Specification

Overview

This section presents the **Software Requirements Specification (SRS)** for **SkillSwap – A Peer-to-Peer Skill Exchange Platform**. The purpose of this specification is to formally define the functional and non-functional requirements of the system in a clear, precise, and verifiable manner.

The requirements documented in this section are the result of a structured Requirements Engineering process, which included:

- Collaborative requirements gathering among team members
- Simulated user interviews using ChatGPT personas (Learner, Mentor, Administrator)
- Analysis of similar platforms
- Quality Function Deployment (QFD)
- Requirements elaboration, negotiation, and conflict resolution

This specification represents the final, negotiated set of requirements, agreed upon by all stakeholders and constrained by academic scope, system feasibility, and institutional policies.

The SRS serves as:

- A reference for system design and implementation
- A basis for verification and validation
- A contract-like agreement describing what the system shall do and how well it shall perform

The scope of this specification includes:

- User management and authentication
- Skill management for teaching and learning
- Search, discovery, and recommendation of mentors
- Session scheduling and management
- Token-based reward and fairness mechanisms
- Ratings, reviews, and feedback
- Notifications, administrative controls, and analytics

The system is initially intended for deployment within a single university environment, with architectural provisions to support future expansion to multiple institutions.

Subsequent sections of this document detail:

- External interface requirements
- Functional requirements
- Non-functional requirements
- System constraints and business rules
- Assumptions and dependencies

External Interface Requirements

This section describes how **SkillSwap** interacts with users, external software components, and communication mechanisms. These interfaces define how information is exchanged between the system and external entities.

User Interface Requirements

The system shall provide a web-based user interface accessible through standard web browsers.

General UI Requirements

- The user interface shall be intuitive and role-specific for Learners, Mentors, and Administrators.
- The interface shall provide clear navigation between major system functions.
- The interface shall display informative messages for successful operations, errors, and constraint violations.
- The interface shall be responsive and usable on desktop and tablet devices.

Learner Interface

- The system shall allow learners to:
    - Register and log in
    - Create and update their profile
    - Add skills they want to learn
    - Search and filter mentors
    - View mentor profiles, ratings, and reviews
    - Request and track learning sessions
    - View token balance and transaction history
    - Submit ratings and reviews after session completion
    - Receive in-app and optional email notifications

Mentor Interface

- The system shall allow mentors to:
    - Register and log in
    - Create and update their profile
    - Add, edit, and remove skills offered for teaching
    - View and manage session requests
    - Accept or decline session requests

- View earnings, token balance, and transaction history
- View ratings, reviews, and feedback history
- Receive notifications related to session requests and updates

Administrator Interface

- The system shall allow administrators to:
  - Log in with elevated privileges
  - View users, sessions, and reports
  - Monitor platform activity through dashboards
  - Review reported issues and take corrective actions
  - Block or restrict users violating policies
  - Access analytics and downloadable reports

Software Interface Requirements

The system shall interact with the following software components:

- **Backend Framework**
  - The system shall use FastAPI to expose RESTful APIs for all system functionalities.
- **Database**
  - The system shall use PostgreSQL for persistent data storage, including users, sessions, tokens, reviews, and logs.
- **Machine Learning Components**
  - The system shall integrate ML models developed using scikit-learn and/or Sentence-BERT for mentor recommendation.
- **Notification Services**
  - The system shall interface with an email service (SMTP or equivalent) for optional email notifications.
- **Analytics & Visualization**
  - The system shall support data visualisation libraries (e.g., Chart.js) for administrative dashboards.

All software interfaces shall use JSON-based request and response formats.

API Interface Requirements

- The system shall expose REST-based APIs for all major modules, including:
  - Authentication and user management

    - Skill management
    - Search and discovery
    - Session management
    - Recommendation services
    - Token and rewards
    - Ratings and reviews
    - Notifications
    - Administration and analytics
- APIs shall follow standard HTTP methods (GET, POST, PUT/PATCH, DELETE).
- APIs shall return appropriate HTTP status codes for success and failure cases.
- APIs shall enforce authentication and authorisation using JWT tokens.

Hardware Interface Requirements

- No specialised hardware interfaces are required.
- The system shall run on standard server infrastructure capable of hosting:
    - Web applications
    - Database services
    - Machine learning models
- Users shall access the system using standard computing devices such as laptops or desktops with internet connectivity.

Communication Interface Requirements

- The system shall communicate over HTTPS to ensure secure data transmission.
- Client–server communication shall follow RESTful principles.
- Notification delivery shall support:
    - In-app notifications
    - Optional email notifications
- All communication involving sensitive data (authentication, tokens, administrative actions) shall be encrypted in transit.

Error Handling and Messaging

- The system shall provide meaningful error messages for:
    - Authentication failures

    - Authorization violations
    - Constraint violations (e.g., insufficient tokens, invalid scheduling)
    - System or network errors
- Error messages shall be user-friendly and shall not expose sensitive system details.

Functional Requirements

This section defines the functional requirements of the SkillSwap system. Functional requirements specify what the system shall do to support peer-to-peer skill exchange.
All requirements listed below are derived from:

- Requirements elicitation
- Collaborative analysis
- Quality Function Deployment (QFD)
- Requirements elaboration and conflict resolution

Each requirement is uniquely identified for traceability.

User Management Module (FR-UM)

FR-UM-01
The system shall allow users to register using a valid university email address.

FR-UM-02
The system shall authenticate users using secure login credentials.

FR-UM-03
The system shall allow users to reset their passwords using a secure verification mechanism.

FR-UM-04
The system shall create and maintain a user profile containing role information (Learner or Mentor).

FR-UM-05
The system shall restrict access to features based on the user's assigned role.

Skill Management Module (FR-SK)

FR-SK-01
The system shall allow mentors to add, edit, and remove skills they offer for teaching.

FR-SK-02
The system shall allow learners to specify skills they want to learn.

FR-SK-03
The system shall store skill details, including skill name, description, and category.

FR-SK-04
The system shall prevent duplicate skill entries for the same user.

Search and Discovery Module (FR-SD)

FR-SD-01
The system shall allow users to search for skills and mentors using keyword-based search.

FR-SD-02
The system shall allow users to filter search results by skill category and mentor rating.

FR-SD-03
The system shall display search results ranked based on relevance and popularity.

Session Management Module (FR-SE)

FR-SE-01
The system shall allow learners to request a learning session with a selected mentor.

FR-SE-02
The system shall allow learners to select a preferred date and time when requesting a session.

FR-SE-03
The system shall validate mentor availability before creating a session request.

FR-SE-04
The system shall assign a status to each session request, including Pending, Confirmed, Rejected, and Completed.

FR-SE-05
The system shall allow mentors to accept or decline session requests.

FR-SE-06
The system shall prevent session booking less than two hours before the scheduled start time.

Matching and Recommendation Module (FR-ML)

FR-ML-01
The system shall generate personalised mentor recommendations based on learner skill preferences.

FR-ML-02
The system shall compute similarity between learner skills and mentor skills using vector-based techniques.

FR-ML-03
The system shall rank mentors based on compatibility score, skill relevance, and ratings.

FR-ML-04
The system shall return a maximum of five recommended mentors per learner request.

FR-ML-05
The system shall automatically update recommendations when learner or mentor skills change.

Token and Rewards Module (FR-TR)

FR-TR-01
The system shall allocate an initial token balance to every newly registered user.

FR-TR-02
The system shall deduct tokens from learners when a session request is confirmed.

FR-TR-03
The system shall award tokens to mentors upon successful completion of a session.

FR-TR-04
The system shall maintain a transaction history for all token earning and spending operations.

FR-TR-05
The system shall ensure that all token transactions are atomic.

Ratings and Review Module (FR-RR)

FR-RR-01
The system shall allow learners to submit a rating and review after completing a session.

FR-RR-02
The system shall restrict learners to one review per completed session.

FR-RR-03
The system shall calculate and display an average rating for each mentor.

FR-RR-04
The system shall allow mentors to view feedback and rating history.

Notification Module (FR-NO)

FR-NO-01
The system shall notify mentors when a learner requests a session.

FR-NO-02
The system shall notify learners when a mentor accepts or declines a session request.

FR-NO-03
The system shall support both in-app notifications and optional email notifications.

Administration Module (FR-AD)

FR-AD-01
The system shall allow administrators to view registered users and session activity.

FR-AD-02
The system shall allow administrators to block or restrict users for policy violations.

FR-AD-03
The system shall allow administrators to review and resolve reported issues.

Analytics and Reporting Module (FR-AN)

FR-AN-01
The system shall provide administrators with dashboards showing platform usage statistics.

FR-AN-02
The system shall generate reports on session activity, skill popularity, and token distribution.

FR-AN-03
The system shall support exporting analytics reports for administrative review.

Non-Functional Requirements

This section defines the non-functional requirements (NFRs) of the SkillSwap system.
Non-functional requirements specify how well the system shall perform its functions and establish quality attributes such as performance, reliability, security, usability, scalability, maintainability, and compliance.

All requirements in this section are derived from:

- Collaborative requirements gathering
- Consolidated performance targets
- QFD expected requirements
- Stakeholder expectations (Learners, Mentors, Administrators)

Each requirement is uniquely identified for traceability.

Performance Requirements (NFR-PERF)

NFR-PERF-01
The system shall complete user registration and login operations within 2 seconds for at least 95% of requests under normal load conditions.

NFR-PERF-02
The system shall return skill search and discovery results within 2 seconds under normal load conditions.

NFR-PERF-03
The system shall generate and return mentor recommendations within 3 seconds per request.

NFR-PERF-04
The system shall deliver in-app notifications within 2 seconds of the triggering event.

Reliability and Availability Requirements (NFR-REL)

NFR-REL-01
The system shall maintain an operational availability of at least 99% uptime, excluding scheduled maintenance.

NFR-REL-02
The system shall preserve session, token, and transaction data in the event of partial system failures.

NFR-REL-03
The system shall recover gracefully from service or database failures without data loss.

Security Requirements (NFR-SEC)

NFR-SEC-01
The system shall require registration using a valid university email address.

NFR-SEC-02
The system shall store all user passwords in hashed and salted form.

NFR-SEC-03
The system shall authenticate users using JWT-based secure sessions.

NFR-SEC-04
The system shall restrict access to administrative functions to authorized users only.

NFR-SEC-05
The system shall ensure that token transactions are protected against unauthorized modification.

Data Integrity and Consistency Requirements (NFR-DATA)

NFR-DATA-01
All token transactions shall be atomic, ensuring either full completion or rollback.

NFR-DATA-02
The system shall maintain consistency between **s**ession status, token balance, and transaction records.

NFR-DATA-03
The system shall prevent duplicate reviews for the same completed session.

Usability Requirements (NFR-USAB)

NFR-USAB-01
The system shall provide a clear and intuitive user interface for learners, mentors, and administrators.

NFR-USAB-02
The system shall provide informative error messages when user actions violate constraints or business rules.

NFR-USAB-03
The system shall minimise the number of steps required to perform common actions such as:

- Searching skills
- Requesting sessions
- Submitting reviews

Scalability Requirements (NFR-SCAL)

NFR-SCAL-01
The system shall support concurrent access by users within a single university without performance degradation.

NFR-SCAL-02
The system architecture shall allow future expansion to multiple institutions without major redesign.

Maintainability and Modularity Requirements (NFR-MAINT)

NFR-MAINT-01
The system shall follow a modular architecture to allow independent updates to system components.

NFR-MAINT-02
The system shall expose REST-based APIs to support maintainability and extensibility.

NFR-MAINT-03
The system shall maintain logs for debugging, monitoring, and audit purposes.

Compliance and Ethical Requirements (NFR-COMP)

NFR-COMP-01
The system shall enforce rules that prevent academic integrity violations, such as misuse for exam or assignment solving.

NFR-COMP-02
The system shall allow administrators to intervene in cases of reported misconduct.

NFR-COMP-03
The system shall ensure fair participation and prevent dominance or misuse by individual users.

Constraints

This section defines the constraints under which the SkillSwap system must operate. Constraints represent non-negotiable limitations, rules, or conditions imposed by institutional policies, system design decisions, fairness considerations, and technical feasibility.

All constraints listed below were derived from:

- Collaborative requirements gathering
- Consolidated team discussions
- Institutional and academic policies
- Token-based fairness model decisions

Each constraint is uniquely identified for traceability.

Authentication and Identity Constraints (CON-AUTH)

CON-AUTH-01
Users shall be allowed to register **o**nly using a valid university email address.

CON-AUTH-02
User passwords shall have a minimum length of 8 characters and include at least one special character.

Token Economy Constraints (CON-TOK)

CON-TOK-01
A learner shall possess at least 10 tokens before submitting a session request.

CON-TOK-02
Each completed learning session shall have a fixed cost of 10 tokens.

CON-TOK-03
Every newly registered user shall receive an initial allocation of 20 tokens.

CON-TOK-04
All token transactions shall be atomic, ensuring either full completion or rollback.

Session Scheduling Constraints (CON-SES)

CON-SES-01
Sessions shall not be booked less than 2 hours before the scheduled start time.

CON-SES-02
A user shall not have more than 5 pending session requests at any given time.

CON-SES-03
Mentors shall not be allowed to accept overlapping session requests.

Review and Feedback Constraints (CON-REV)

CON-REV-01
A learner shall be allowed to submit only one review per completed session.

CON-REV-02
Reviews shall be permitted only after a session is marked as completed.

Business Rules

This section defines the business rules governing the operation of the SkillSwap platform. Business rules represent organisational policies, ethical guidelines, and fairness principles that regulate system behaviour beyond technical constraints.

These rules ensure that the platform:

- Promotes fair participation
- Encourages high-quality peer learning
- Maintains academic integrity
- Enables accountable administrative control

All business rules were derived from:

- Stakeholder interviews (Learners, Mentors, Administrators)
- Conflict resolution during requirements elaboration
- Institutional and academic policies

Each business rule is uniquely identified for traceability.

Participation and Fairness Rules (BR-FAIR)

BR-FAIR-01
Teaching effort and learning effort shall be balanced using a token-based exchange mechanism.

BR-FAIR-02
Mentors shall be rewarded only for successfully completed sessions.

BR-FAIR-03
The platform shall prioritise quality of teaching over quantity when evaluating mentor contributions.

Conduct and Safety Rules (BR-COND)

BR-COND-01
Users shall adhere to respectful and appropriate conduct during all platform interactions.

BR-COND-02
The system shall allow users to report inappropriate behaviour or misuse.

BR-COND-03
Administrators shall have the authority to investigate reported violations and take corrective actions.

Academic Integrity Rules (BR-ACAD)

BR-ACAD-01
The platform shall not be used for exam cheating or direct assignment completion.

BR-ACAD-02
Sessions identified as violating academic integrity shall be reviewed, and appropriate disciplinary actions shall be taken.

Administrative Control Rules (BR-ADMIN)

BR-ADMIN-01
Administrators shall have the authority to block or restrict users violating platform policies.

BR-ADMIN-02
Administrative interventions shall be proportionate, justified, and logged for accountability and audit purposes.

Validation

Requirements validation was carried out to ensure that the requirements specification is complete, consistent, correct, feasible, and unambiguous, and that it conforms to the standards defined for the project, process, and product.

In this project, the requirements specification consists of:

- Functional Requirements
- Non-Functional Requirements
- Constraints and Business Rules
- UML models
- Quality Function Deployment (QFD)

Validation Approach (Technical Review)

Add this (short, factual):

- Validation was performed through technical reviews
- Reviews involved all four team members, representing different stakeholder viewpoints.
- The objective was to detect:
  - ambiguity
  - inconsistency
  - missing requirements
  - conflicts
  - unrealistic expectations

Executed Requirements Validation Checklist

Validation Question (from textbook)	How it was validated in SkillSwap
Are requirements stated clearly and unambiguously?	All requirements are written as "shall" statements with measurable conditions (tokens, time limits, response times).
Is the source of each requirement identified?	Each requirement is traceable to simulated interviews, collaborative workshops, or platform analysis.
Are requirements bounded quantitatively?	Performance limits, token values, session limits, and uptime targets are numerically defined.
Are related requirements cross-referenced?	Requirements are mapped to modules, UML diagrams, and QFD tables.
Do requirements violate system constraints?	All functional requirements conform to defined token, session, and review constraints.
Is each requirement testable or observable?	Each requirement corresponds to observable behavior in UML diagrams.
Are requirements traceable to system models?	Use case, activity, sequence, class, and state diagrams cover all major requirements.
Are requirements aligned with system objectives?	All requirements support fairness, trust, peer learning, and institutional oversight.
Are implicit requirements identified?	Security, reliability, and atomicity are captured as NFRs and constraints.

Model-Based Validation (UML Walkthroughs)

UML Model	Validated Requirements
Use Case Diagram	FR-UM, FR-SK, FR-SE, FR-TR, FR-RR, FR-AD
Activity Diagram (Session Request)	FR-SE-01 to FR-SE-06, CON-SES-01

Sequence Diagram (Session Request)	FR-SE, FR-NO, FR-TR
Class Diagram	FR-UM, FR-SK, FR-TR, FR-RR
State Transition Diagram	FR-SE-04, FR-SE-05, FR-TR

- Walkthroughs ensured no missing steps
- Invalid state transitions were prevented
- Constraints were enforced at each stage

This replaces test execution at the design stage.

Conflict-Based Validation

- Conflicting requirements identified during elicitation were reviewed and resolved collaboratively.
- Each resolved conflict resulted in a refined requirement consistent with constraints and stakeholder needs.
- This ensured internal consistency and feasibility of the specification.

Validation Outcome

- No unresolved inconsistencies remain
- All requirements are:
    - clear
    - complete
    - consistent
    - feasible
    - traceable
- The validated specification is approved for design and implementation

Requirements Management

Requirements management is the process of identifying, controlling, tracking, and documenting changes to requirements throughout the project lifecycle. As stated in the textbook, requirements for computer-based systems are expected to evolve, and effective requirements management ensures that changes do not compromise system consistency, feasibility, or stakeholder objectives.

Given that SkillSwap is a moderate-sized academic project, requirements management was implemented in a structured but lightweight manner, avoiding unnecessary overhead while still ensuring control and traceability.

Requirements Identification and Organisation

Each requirement in the SkillSwap system was assigned a unique identifier to enable tracking and change control. Requirements were categorized into the following groups:

- Functional Requirements (FR)
- Non-Functional Requirements (NFR)
- Constraints (CON)
- Business Rules (BR)

Additionally, requirements were grouped by system module, such as:

- User Management
- Skill Management
- Session Management
- Token & Rewards
- Matching & Recommendation
- Ratings & Review
- Notification
- Admin
- Analytics & Reporting

This structured identification scheme ensured that every requirement could be uniquely referenced during validation, traceability, and future modification.

Change Identification and Change Requests

Although no external clients were involved, requirement changes arose during:

- Conflict resolution discussions
- QFD analysis
- UML walkthroughs
- Peer reviews during validation

When a change was identified, it was treated as an informal change request, evaluated collaboratively by the team.

Examples of changes handled during the project include:

- Introducing a token-based mechanism to resolve fairness concerns
- Limiting pending session requests to prevent mentor burnout
- Adding session states (Pending, Confirmed, Completed) to ensure accountability
- Refining performance requirements into quantifiable limits

This table illustrates some requirement changes recorded during the project. These changes were identified during elicitation, validation, and conflict resolution activities and were evaluated collaboratively before approval.

Table: Requirement Change Log

Change ID	Date	Change Description	Reason for Change	Impacted Requirements / Modules	Impact Analysis	Decision
CR-01	Nov 28, 2025	Introduced a token-based mechanism for session booking and rewards	Learner–mentor fairness conflict identified during elicitation	FR-TR-01 to FR-TR-05, Token & Rewards Module	Added token wallet, transaction handling, and constraints; minor increase in system complexity	Approved
CR-02	Nov 30, 2025	Added session states (Pending, Confirmed, Rejected, Completed)	To ensure mentor control and accountability in scheduling	FR-SE-04, Session Management Module	Required updates to activity, sequence, and state diagrams	Approved
CR-03	Dec 01, 2025	Limited maximum pending session requests to five per user	Prevent mentor burnout and request flooding	FR-SE-05, CON-SES-02	Improved system fairness; no negative performance impact	Approved
CR-04	Dec 02, 2025	Restricted reviews to one per completed session	Prevent spam and biased mentor ratings	FR-RR-02, Ratings & Review Module	Simplified review logic; improved trust	Approved
CR-05	Dec 03, 2025	Added combined manual search and ML-based recommendations	Balance algorithmic guidance with user autonomy	FR-SD-01, FR-ML-01	Required coordination between Search and Recommendation modules	Approved
CR-06	Dec 05, 2025	Quantified performance requirements (≤2–3 seconds)	Remove ambiguity in non-functional requirements	NFR-PERF-01 to NFR-PERF-04	Enabled objective validation; no design impact	Approved

Change Evaluation and Impact Analysis

Each proposed change was evaluated based on:

- Impact on existing functional requirements
- Compatibility with constraints and business rules
- Effect on system complexity and feasibility
- Alignment with stakeholder needs and system objectives

Before accepting a change, the team assessed:

- Which modules would be affected
- Whether new constraints were required
- Whether UML models needed updates
- Whether traceability links remained valid

Only changes that improved clarity, fairness, or feasibility were approved.

Change Control and Approval Process

Change control was handled through **collaborative decision-making** rather than formal approval boards, as appropriate for an academic project.

The change control process followed these steps:

1. Identification of the requirement change
2. Discussion of the rationale and impact
3. Agreement among all team members
4. Update of affected requirements, models, and documentation
5. Re-validation of modified requirements

This ensured that no requirement was changed in isolation and that system consistency was preserved.

Requirements Traceability Management

Traceability was maintained throughout the project to ensure that:

- Every requirement could be traced back to its source
- Every requirement was represented in design artefacts
- Changes could be tracked across documents and models

Traceability links included:

- Elicitation source → Requirement
- Requirement → System module
- Requirement → UML diagrams
- Requirement → QFD classification (Normal, Expected, Exciting)

This traceability allowed the team to quickly assess the impact of changes and ensured alignment between stakeholder needs and system design.

Version Control and Documentation Updates

All requirement documents were version-controlled using a shared repository. Updates to requirements were accompanied by:

- Clear identification of modified sections
- Consistent requirement identifiers
- Corresponding updates to UML diagrams and tables

This ensured that the most recent and validated version of the specification was always available and avoided inconsistencies between documents.

Scope Control and Prevention of Requirements Creep

To prevent uncontrolled growth of requirements (requirements creep), the team followed these principles:

- Only core features aligned with the project scope were accepted
- Advanced or optional features were documented as future enhancements
- Exciting requirements were carefully evaluated for feasibility before inclusion

This ensured that the project remained achievable within the academic timeline while still delivering a meaningful and coherent system.

Requirements Management Outcome

As a result of the requirements management activities:

- All requirement changes were controlled and documented
- No unresolved inconsistencies remained in the specification
- Traceability between stakeholder needs and system design was preserved
- The final requirements set represents a stable and agreed-upon baseline for design and implementation

The managed requirements specification provides a reliable foundation for subsequent development and evaluation phases.

Process Model (Project-Level)

The project adopts the **Agile Scrum process model**.

Justification

- Supports incremental delivery
- Handles evolving requirements
- Encourages collaboration and continuous feedback
- Suitable for small, modular academic projects

Development is carried out in short iterations, with regular reviews and retrospectives.

Process Flow (Project-Level)

The overall process flow follows:

1. Sprint Planning
2. Requirement selection from backlog
3. Module implementation
4. Review and validation
5. Incremental refinement

This ensures continuous improvement and early issue detection.

Design Approach and Architecture

Design Overview

The design phase translates the validated and approved requirements of the SkillSwap system into a structured solution that defines *how* the system will be built. While the requirements specification describes *what* the system must do, the design phase focuses on *how those requirements are realised* through architecture, components, and interactions.

The design is derived directly from:

- Validated functional and non-functional requirements
- Identified system entities and services
- Constraints and business rules
- Conflict resolution decisions
- Quality Function Deployment (QFD) outcomes

The primary goals of the design phase are:

- To ensure clear separation of responsibilities among system components
- To achieve modularity and maintainability, allowing independent development and future enhancement
- To support scalability, enabling expansion beyond a single institution
- To ensure security, performance, and reliability as specified in non-functional requirements

The design provides a blueprint for implementation and acts as a reference for developers, testers, and evaluators. All design decisions are traceable back to validated requirements, ensuring consistency and correctness throughout the development lifecycle.

Design Type

The SkillSwap system follows an **Object-Oriented Design (OOD)** approach.

Justification for Object-Oriented Design

Object-oriented design was chosen because the system naturally consists of well-defined real-world entities such as users, skills, sessions, tokens, and reviews. Each of these entities encapsulates both data and behavior, making OOD an appropriate and effective design choice.

The key reasons for adopting object-oriented design include:

- **Natural mapping to problem domain**
  Entities identified during requirements elicitation (User, Skill, Session, Token, Review, etc.) map directly to classes and objects.
- **Encapsulation and data protection**
  Sensitive data such as authentication credentials and token balances are encapsulated within classes, reducing unintended access and misuse.

- **Reusability and extensibility**
  Common behaviors can be reused through inheritance and composition, supporting future enhancements such as new roles or additional modules.

- **Maintainability**
  Changes to one module (e.g., recommendation logic or token rules) can be implemented with minimal impact on other components.

- **Support for UML modeling**
  Object-oriented design integrates seamlessly with UML diagrams such as class diagrams, sequence diagrams, and state transition diagrams, which are required artifacts for this project.

Design Style

The object-oriented design is implemented using a **modular layered design style**, where:

- Each module represents a distinct functional responsibility
- Interactions occur through well-defined interfaces
- Business logic is separated from presentation and data access layers

This design style supports:

- Independent module development
- Easier testing and debugging
- Better alignment with REST-based service architecture

Design Scope

The object-oriented design applies across all major modules of the system, including:

- User Management
- Skill Management
- Search & Discovery
- Session Management
- Matching & Recommendation
- Token & Rewards
- Ratings & Review
- Notification
- Administration
- Analytics & Reporting

Each module will be represented through appropriate UML diagrams in subsequent sections of the design documentation.

Design Principles

The design of the SkillSwap system follows established software design principles to ensure that the system is robust, maintainable, scalable, and aligned with the validated requirements. These principles guide architectural decisions, class structuring, module interactions, and future extensibility.

Each principle applied is directly traceable to issues identified during requirements elicitation, conflict resolution, and QFD analysis.

Modularity

The system is designed as a collection of independent yet cohesive modules, each responsible for a specific functional area.

Application in SkillSwap:

- User Management
- Skill Management
- Search & Discovery
- Session Management
- Matching & Recommendation
- Token & Rewards
- Ratings & Review
- Notification
- Admin
- Analytics & Reporting

Each module:

- Has clearly defined responsibilities
- Exposes well-defined interfaces
- Can be developed, tested, and maintained independently

Benefit:
Improves maintainability, supports parallel development, and simplifies future enhancements.

Separation of Concerns

Different system concerns are separated across layers and modules to avoid tight coupling.

Application in SkillSwap:

- Presentation logic (UI interactions)
- Business logic (session rules, token rules, recommendation logic)

- Data management (users, sessions, transactions)

For example:

- Token validation logic is handled only by the Token module
- Recommendation logic is isolated within the ML module
- Administrative controls are restricted to the Admin module

Benefit:
Reduces complexity, improves readability, and makes the system easier to debug and modify.

Encapsulation

Each class encapsulates its internal data and exposes behavior only through controlled interfaces.

Application in SkillSwap:

- User credentials and authentication data are encapsulated within the User entity
- Token balances and transaction history are accessible only through token service operations
- Session state transitions are controlled by session management logic

Benefit:
Protects data integrity, enhances security, and prevents unauthorized manipulation of system state.

Low Coupling and High Cohesion

Modules are designed to minimize dependencies between them while ensuring that related responsibilities remain grouped together.

Application in SkillSwap:

- The Session module interacts with the Token module only through defined service calls
- The Recommendation module consumes skill data but does not manage skill creation
- The Analytics module reads aggregated data without affecting transactional logic

Benefit:
Improves system stability and allows changes in one module without cascading failures.

Abstraction

The design abstracts complex operations behind simplified interfaces.

Application in SkillSwap:

- Recommendation generation abstracts ML complexity from the rest of the system
- Token transactions abstract database-level operations from business logic

- Notification delivery abstracts multiple channels (in-app, email)

Benefit:
Simplifies understanding and usage of system components and supports future replacement or enhancement of internal implementations.

Reusability

Design elements are created to be reusable across multiple modules and scenarios.

Application in SkillSwap:

- User entity reused across learner, mentor, and admin roles
- Notification mechanism reused for session updates, reviews, and administrative actions
- Reporting components reused across different analytics views

Benefit:
Reduces redundancy, improves consistency, and lowers development effort.

Scalability-Oriented Design

The system is designed with scalability in mind, even though initial deployment targets a single university.

Application in SkillSwap:

- Stateless REST APIs
- Modular services that can be scaled independently
- Analytics and recommendation components designed to handle increasing data volume

Benefit:
Allows seamless expansion to multiple institutions without major architectural redesign.

Security by Design

Security considerations are embedded into the design rather than added as an afterthought.

Application in SkillSwap:

- Role-based access control for learners, mentors, and administrators
- Encapsulated authentication and authorization mechanisms
- Controlled access to sensitive administrative and analytics data

Benefit:
Reduces risk of misuse, protects user data, and ensures compliance with institutional policies.

Traceability to Requirements

Every design decision adheres to validated requirements derived from:

- Collaborative requirements gathering
- Simulated user interviews
- QFD analysis
- Conflict resolution documentation

Each module and design element can be traced back to specific functional or non-functional requirements.

Benefit:
Ensures design correctness and simplifies validation, testing, and future maintenance.

Architecture Style

The SkillSwap system adopts a Layered, Modular, Service-Oriented Architectural Style implemented using REST-based APIs. This architectural choice was driven by the system's functional complexity, scalability requirements, maintainability goals, and the need to support independent evolution of system modules.

The selected architectural style aligns with the outcomes of:

- Requirements elicitation
- Quality Function Deployment (QFD)
- Non-functional requirements analysis
- Maintainability and scalability constraints

Selected Architectural Style

Primary Style:
Layered Architecture with Modular Services (REST-based)

Supporting Style:
Service-Oriented Architecture (SOA) principles

Architectural Layers

The system is structured into the following logical layers:

1. Presentation Layer (User Interface Layer)

Responsibility:

- Handles user interaction and input
- Displays system responses and feedback

Actors Served:

- Learners
- Mentors

- Administrators

Examples:

- Registration and login screens
- Skill selection and search interfaces
- Session request and scheduling screens
- Admin dashboards and analytics views

Justification:
Separates user interaction logic from business rules, improving usability and ease of change.

2. Application / Service Layer (Business Logic Layer)

Responsibility:

- Implements core system functionality
- Enforces business rules and constraints
- Coordinates interactions between modules

Key Services:

- User Management Service
- Skill Management Service
- Session Management Service
- Matching & Recommendation Service
- Token & Rewards Service
- Ratings & Review Service
- Notification Service
- Admin Service
- Analytics Service

Justification:
Encapsulates system behavior and ensures consistent enforcement of functional and non-functional requirements.

3. Data Access Layer

Responsibility:

- Manages persistence and retrieval of system data
- Enforces data integrity and transactional consistency

Managed Entities:

- Users and Profiles
- Skills
- Sessions
- Tokens and Transactions
- Reviews and Ratings
- Notifications
- Reports and Analytics Data

Justification:
Isolates database interactions from business logic, supporting maintainability and data integrity.

4. Infrastructure Layer

Responsibility:

- Provides technical services required by the system
- Supports scalability, security, and reliability

Components:

- Authentication and authorization mechanisms
- Logging and monitoring services
- API gateway and routing
- Database and storage services

Justification:
Ensures reliable system operation and supports non-functional requirements such as security, availability, and performance.

Module-Oriented Decomposition

Each functional area of SkillSwap is implemented as an independent **logical service module**.

Module	Responsibility
User Management	Registration, authentication, role handling
Skill Management	Managing skills offered and requested
Search & Discovery	Keyword search and filtering
Session Management	Scheduling, session lifecycle control
Matching & Recommendation	ML-based mentor recommendations
Token & Rewards	Token allocation, spending, and earning

Ratings & Review	Feedback and mentor evaluation
Notification	In-app and email alerts
Admin	Moderation, blocking, policy enforcement
Analytics & Reporting	Usage metrics and institutional insights

This decomposition supports low coupling and high cohesion.

Communication Style

The modules communicate using RESTful APIs over HTTP.

Characteristics:

- Stateless interactions
- JSON-based request and response formats
- Clear endpoint definitions

Examples:

- POST /sessions
- GET /recommendations
- POST /tokens/earn
- GET /analytics/usage

Justification:
REST enables scalability, interoperability, and ease of integration with future systems.

Architectural Justification

The chosen architectural style was selected because it:

- Supports modular development and independent testing
- Aligns with maintainability and extensibility NFRs
- Enables future expansion to multiple institutions
- Simplifies enforcement of security and business rules
- Integrates well with ML-based components

Alignment with Requirements

Requirement Category	Architectural Support
Performance	Stateless APIs, independent services
Scalability	Horizontal scaling of services

Security	Layered access control
Maintainability	Modular structure
Reliability	Isolation of failures
Compliance	Centralized admin and analytics services

The layered, modular, service-oriented architecture provides a robust foundation for the SkillSwap system. It ensures strong alignment with stakeholder needs, supports future growth, and enables clean implementation of complex features such as ML-based recommendations and token-based fairness mechanisms.

Architecture Diagrams

System Architecture Diagrams

The architecture of the SkillSwap system is represented using three complementary views to provide clarity at different abstraction levels. These diagrams collectively illustrate the system's environment, internal structure, and detailed component interactions.

Architectural Context Diagram

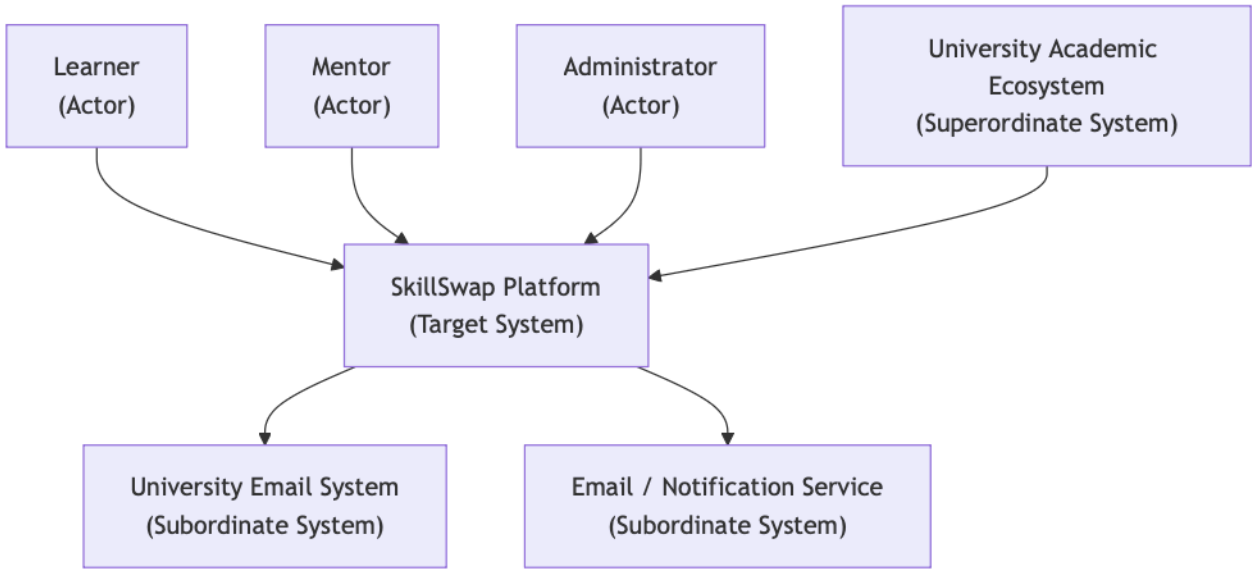
Purpose

The Architectural Context Diagram shows SkillSwap as a single system and illustrates how it interacts with external actors and external systems.
This diagram defines system boundaries, which is essential before internal design.

External Actors & Systems

- Learner
- Mentor
- Administrator
- University Authentication System (Email verification)
- Notification Services (Email/SMS – optional)

Explanation



- Users interact with SkillSwap through web or mobile interfaces.
- SkillSwap verifies identity using university email systems.
- Notifications may be delivered via external email services.
- Administrators oversee system activity without interfering in user interactions.

Basic Architecture Diagram (High-Level Architecture)

Purpose

This diagram presents the **high-level internal structure** of SkillSwap using a **layered architectural view**.

Layers Represented

1. Presentation Layer
2. Application / Service Layer
3. Data Layer
4. Infrastructure Services

Explanation

- The Presentation Layer handles all user interactions.
- The Service Layer implements business logic and rules.
- The Data Layer manages persistence and integrity.
- Infrastructure services provide cross-cutting concerns like security and logging.

Diagram - to be included

Detailed Architecture Diagram

Purpose

The Detailed Architecture Diagram expands the service layer into individual modules, showing how responsibilities are distributed and how modules interact.

This diagram demonstrates:

- Modularity
- Separation of concerns
- Clear responsibility assignment

Key Modules Included

- User Management
- Skill Management
- Search & Discovery
- Session Management
- Matching & Recommendation (ML)
- Token & Rewards
- Ratings & Review
- Notification
- Admin
- Analytics & Reporting

Architectural Design Justification

Design Goal	How Architecture Supports It
Modularity	Independent service modules
Scalability	Services can scale independently
Maintainability	Clear separation of responsibilities
Security	Centralized authentication & role control
Reliability	Isolated failures do not crash entire system
Extensibility	New modules can be added easily

Mapping Architecture to Requirements

Requirement Type	Architectural Support
Functional	Service-specific modules
Performance	Stateless REST APIs
Security	Auth & Admin services
Maintainability	Layered modular design
Compliance	Admin + Analytics oversight

The architectural design of SkillSwap follows a layered, modular, service-oriented approach that directly reflects validated requirements, supports scalability, and enables future institutional expansion.

Diagram - to be included

Design Patterns

Design patterns provide reusable, proven solutions to recurring software design problems.
In SkillSwap, design patterns were selected to improve modularity, scalability, maintainability, and flexibility, while aligning with the layered and service-oriented architecture.

The patterns chosen are derived from system requirements, architectural decisions, and module responsibilities.

Model–View–Controller (MVC) Pattern

Pattern Category

Architectural / Structural Pattern

Problem Addressed

The system requires:

- Clear separation between user interface, business logic, and data
- Independent evolution of UI and backend logic
- Maintainable and testable code structure

Pattern Description

The MVC pattern separates the application into:

- **Model**: Core data and business logic
- **View**: User interface components
- **Controller**: Handles user input and coordinates between Model and View

Application in SkillSwap

- **Model**: User, Skill, Session, Token, Review entities
- **View**: Web UI (Learner, Mentor, Admin dashboards)
- **Controller**: FastAPI route handlers and service controllers

Justification

- Prevents tight coupling between UI and logic
- Supports parallel frontend and backend development
- Enhances maintainability and readability

Context

The system includes distinct user interfaces for learners, mentors, and administrators, interacting with shared business logic and data models.

System of Forces

- Need to support multiple user roles with different views
- Requirement to minimise UI-driven changes to business logic
- Maintainability and extensibility requirements
- Usability requirement for clean and intuitive interfaces

Consequences

- Clear separation of concerns improves maintainability
- Enables parallel development of UI and backend logic
- Reduces coupling between presentation and data layers
- Requires disciplined adherence to MVC boundaries

Repository Pattern

Pattern Category

Structural Pattern

Problem Addressed

Direct database access across multiple modules would:

- Increase coupling
- Make future database changes difficult
- Complicate testing

Pattern Description

The Repository Pattern abstracts data access logic into a dedicated layer, providing a clean interface between business logic and persistence.

Application in SkillSwap

- Separate repositories for:
    - UserRepository
    - SkillRepository
    - SessionRepository
    - TokenRepository
- Business services interact with repositories instead of raw SQL

Justification

- Centralizes data access logic
- Makes database changes transparent to services
- Enables easier unit testing and mocking

Context

Multiple modules interact with persistent data such as users, skills, sessions, tokens, and reviews stored in a relational database.

System of Forces

- Need to isolate business logic from database implementation
- Requirement for maintainable and testable codebase
- Possibility of future database migration or schema evolution
- Data integrity constraints (atomic token transactions, consistent session states)

Consequences

- Centralises and standardises data access operations
- Improves testability by enabling mocking of data repositories
- Reduces ripple effects when database schemas or technologies change
- Adds a structural layer that must be carefully maintained

Observer Pattern

Pattern Category

Behavioral Pattern

Problem Addressed

Multiple components must react automatically to certain events, such as:

- Session acceptance
- Token updates
- Status changes

Pattern Description

The Observer Pattern allows objects (observers) to subscribe to events generated by a subject and be notified when changes occur.

Application in SkillSwap

- Session status changes trigger:
    - Notification service

    - Token service updates
- Observers listen for domain events like:
    - SessionConfirmed
    - SessionCompleted

Justification

- Decouples event producers from consumers
- Enables asynchronous, scalable notifications
- Avoids hard-coded dependencies

Context

Session lifecycle events (request, acceptance, completion) must trigger updates across multiple modules such as Notifications, Tokens, and Analytics.

System of Forces

- Loose coupling required between session management and dependent services
- Need for automatic and immediate updates when session state changes
- Requirement to avoid manual coordination across modules
- Reliability requirement: notifications must be delivered within 2 seconds

Consequences

- Enables event-driven updates without tight inter-module dependencies
- Improves extensibility by allowing new observers to be added easily
- Slight increase in event-handling complexity
- Enhances responsiveness and user experience through real-time updates

Strategy Pattern

Pattern Category

Behavioral Pattern

Problem Addressed

The system must support:

- Multiple recommendation techniques
- Future experimentation with ML models
- Algorithm switching without impacting other modules

Pattern Description

The Strategy Pattern encapsulates interchangeable algorithms behind a common interface.

Application in SkillSwap

- Recommendation strategies:
  - TF-IDF based similarity
  - SBERT embedding-based similarity
- Strategy selected dynamically based on configuration

Justification

- Enables ML experimentation without refactoring
- Promotes extensibility
- Keeps recommendation logic modular

Context

The Matching & Recommendation module must support mentor recommendations based on evolving algorithms (e.g., TF-IDF, SBERT) and changing weighting criteria (skill relevance, ratings, activity).

System of Forces

- Need to experiment with and replace recommendation algorithms over time
- Requirement to avoid modifying dependent modules when algorithms change
- Performance constraint: recommendations must be generated within 3 seconds
- Maintainability requirement: ML logic should evolve independently
- Scalability requirement for future enhancement of recommendation techniques

Consequences

- Enables algorithm flexibility without impacting other system components
- Promotes separation of concerns between recommendation logic and application flow
- Introduces an additional abstraction layer that slightly increases design complexity
- Simplifies future integration of improved or hybrid ML models

Factory Pattern

Pattern Category

Creational Pattern

Problem Addressed

Object creation logic for domain entities and services should not be scattered across the codebase.

Pattern Description

The Factory Pattern centralizes object creation logic and returns appropriate instances based on context.

Application in SkillSwap

- Creation of:
    - User objects (Learner, Mentor)
    - Notification handlers
    - Recommendation strategy instances

Justification

- Simplifies object creation
- Improves consistency
- Reduces duplication

Context

The system must instantiate different types of domain objects such as users, sessions, and notifications based on role or state.

System of Forces

- Object creation logic should not be scattered across modules
- Requirement for consistency in object initialisation
- Extensibility requirement for future object types

Consequences

- Centralises object creation logic
- Improves consistency and reduces duplication
- Slightly increases indirection during instantiation
- Facilitates future extensions without modifying client code

Singleton Pattern (Controlled Usage)

Pattern Category

Creational Pattern

Problem Addressed

Certain services must have **a single shared instance**:

- Configuration
- Logging
- Database connection pool

Pattern Description

The Singleton Pattern ensures a class has only one instance and provides a global access point.

Application in SkillSwap

- Logging service
- Configuration manager
- Database connection pool

Justification

- Prevents resource duplication
- Ensures consistent configuration
- Used cautiously to avoid tight coupling

Context

Shared system resources such as configuration settings and logging services must be accessed consistently across all modules.

System of Forces

- Need for a single authoritative instance
- Requirement to avoid configuration inconsistencies
- Logging must be accessible across modules

Consequences

- Ensures controlled access to shared resources
- Simplifies configuration management
- Must be used cautiously to avoid global state misuse
- Potential testing challenges if not carefully managed

Pattern–Module Mapping Summary

Design Pattern	Modules Applied
MVC	All UI & backend layers

Repository	User, Skill, Session, Token
Observer	Notification, Token, Session
Strategy	Matching & Recommendation
Factory	User creation, ML strategy selection
Singleton	Logging, DB connection

Design Pattern Justification Summary

The selected design patterns:

- Align with object-oriented design principles
- Support modular and scalable architecture
- Facilitate future enhancements and maintenance
- Reduce coupling and improve cohesion

These patterns collectively ensure that SkillSwap remains robust, extensible, and adaptable as it scales from a single university to a broader ecosystem.

Conclusion

The SkillSwap project demonstrates the application of end-to-end Software Engineering principles, including Agile development, modular design, systematic requirements engineering, UML modelling, and performance evaluation. The system is designed with a controlled initial scope (single university) and a scalable architecture that supports future expansion.