Software Requirements Specification

for

KolabIT

Prepared by Sanchit Raut

Batch: B2

Sanchit Raut	2023800095	sanchit.raut23@spit.ac.in
Vedant Kannurkar	2024801005	vedant.kannurkar@spit.ac.in

Instructor:	Prof. Varsha Hole
Course:	Software Engineering

Date:

Revisions

Version	Primary Author(s)	Description of Version	Date Completed
Draft Type and Number	Full Name	Information about the revision. This table does not need to be filled in whenever a document is touched, only when the version is being upgraded.	00/00/00

1 Introduction

1.1 Document Purpose

This document specifies the software requirements for KolabIT, Release 1.0 (Version 1.0). It details the system's purpose, features, interfaces, functionalities, constraints, and responses to user interactions. This SRS covers the entire KolabIT system, which is a standalone web application without subsystems. The purpose of this document is to provide a comprehensive guide for stakeholders, including developers, instructors, and evaluators, to ensure alignment on project goals and to serve as a proposal for academic evaluation at Sardar Patel Institute of Technology.

1.2 Product Scope

KolabIT is a web platform designed to connect college students based on their academic and technical skills, fostering collaboration for group projects, hackathons, and peer-to-peer learning. The system enables students to create profiles showcasing skills, certifications, and portfolios; search for peers with specific skills; engage in a community forum for Q&A; rate and review peers; collaborate on projects with task management and messaging features; and view analytics on profile visibility and engagement. The benefits include maximizing student productivity by creating an internal talent marketplace, reducing time to find collaborators, and enhancing campus networking. It is accessible via web browsers and assumes no external integrations except email notifications.

1.3 Intended Audience and Document Overview

This document is intended for developers who will implement the system, project managers or evaluators (such as instructors and professors) who will assess the project, and potential users (students and admins) for understanding the system's capabilities. The SRS is organized as follows: Introduction provides an overview; Overall Description gives a high-level perspective; Specific Requirements detail functional and non-functional aspects; Other Non-functional Requirements cover performance, safety, security, and quality attributes; and appendices include supporting information like a data dictionary. Readers should start with the Introduction and Overall Description for context, then proceed to Specific Requirements for developers, or Other Non-functional Requirements for evaluators focused on quality.

1.4 Definitions, Acronyms and Abbreviations

TERM	DESCRIPTION
Forum	A Q&A section for posting and answering questions.
JWT	JSON Web Token, used for authentication.
Skill Profile	A user's digital profile containing skills, certifications, and project details.
SMTP	Simple Mail Transfer Protocol, for email notifications.
SRS	Software Requirements Specification.

1.5 Document Conventions

This document follows IEEE 830-1998 formatting requirements. Text is in Arial font size 11, single-spaced with 1" margins. Section and subsection titles are bolded and numbered. Italics are used for emphasis or placeholders. Naming conventions include camelCase for variable names in code references and descriptive titles for use cases.

1.6 References and Acknowledgments

- IEEE 830-1998
- https://www.geeksforgeeks.org/software-engineering/software-requirement-specification-srs-format/
- https://en.wikipedia.org/wiki/Software_requirements_specification

2 Overall Description

2.1 Product Perspective

KolabIT is a new, self-contained web application aimed at addressing the need for skill-based collaboration among college students, replacing informal methods like social media groups or word-of-mouth. It creates a centralized platform within the campus ecosystem, interfacing with email services for notifications but not with external systems. The system interacts with users via web browsers and a backend server.

<DIAGRAM BAKI HAI>

2.2 Product Functionality

- Register and login users.
- Create and update user profiles with skills, certifications, and portfolios.
- Search for peers based on skills.
- Post and answer questions in a community forum.
- Rate and review peers.
- Create and join collaborative projects.
- Send messages in real-time.
- View analytics on profile engagement.
- Manage users and moderate content.

<DIAGRAM BAKI>

2.3 Users and Characteristics

- Student User: Primary users, college students familiar with web forms, search engines, browsing, and email. They use most functions frequently and are the most important to satisfy core collaboration features.

- Admin: Secondary users, such as faculty or IT staff, proficient in web navigation and basic database management. They use moderation tools less frequently but critically for system integrity.

2.4 Operating Environment

This web application can be deployed on Linux or Windows machines.

- ❖ Minimum RAM: 512MB
- ❖ 20GB Storage Space
- ❖ Intel Dual Core Processor
- ❖ Internet Connectivity with Ports configured

It can be accessed through machines with web browsers supporting HTML, JavaScript, and modern standards (e.g., Chrome version 29 or above, Firefox 3.5 or above, Edge). The portal supports mobile or desktop access with internet and browser compatibility.

2.5 Design and Implementation Constraints

Design Constraints:

- ❖ Security: User data and profiles must be secured against unauthorized access using HTTPS and JWT.
- ❖ Fault Tolerance: Data should not be corrupted in case of crashes; use MongoDB backups.
- ❖ Database: Limited to MongoDB for flexible schemas.
- ❖ Search: Elasticsearch for queries.
- ❖ Frontend: React.js and Bootstrap for responsiveness
- ❖ Backend: Node Express.

2.6 User Documentation

User documentation includes in-app help (tooltips, FAQs), a downloadable PDF manual covering registration to collaboration, and admin guides for moderation. Tutorials will be available online for reporting issues, with a simple "How It Works" page in HTML.

2.7 Assumptions and Dependencies

- ❖ Users have basic computer knowledge and email access.
- ❖ Students are from college environments.
- ❖ Stable internet connection required.
- ❖ Dependency on SMTP for reliable email delivery.
- ❖ The system handles up to 5,000 users; scaling needed beyond.

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

This software needs the following user interfaces:

i) Registration Window:

- ❖ User: Student or Admin
- ❖ Properties:
    - ➢ Used for entry of details like name, email, password, college ID.
    - ➢ Text fields for input; validates uniqueness and sends confirmation email.
    - ➢ Redirects to login upon success.

ii) Login Window:

- ❖ User: Student, Admin
- ❖ Properties:
    - ➢ Fields for email and password; buttons for login or sign up.
    - ➢ Authenticates with JWT; redirects to appropriate dashboard.
    - ➢ Error messages for invalid credentials.

iii) Student Dashboard:

- ❖ User: Registered Student
- ❖ Properties:
    - ➢ Opens after student login.
    - ➢ Options to create/update profile (skills, certifications, portfolios).
    - ➢ Search bar for skills; forum for Q&A; project creation/join buttons.
    - ➢ Messaging interface; analytics charts for engagement.
    - ➢ View upcoming projects or forums and register.

iv) Admin Dashboard:

- ❖ User: Admin
- ❖ Properties:
    - ➢ Opens after admin login.
    - ➢ Options to manage users (view, suspend, delete).
    - ➢ Moderate forum/content; notify users via email.
    - ➢ Update system details.

v) Profile/Page Views:

- ❖ User: Student
- ❖ Properties:
    - ➢ Displays skills, ratings; rate button (1-5 stars, comments).
    - ➢ Search results show highlighted matches.

3.1.2 Hardware Interfaces

The program communicates with the filesystem and database via Node.js/Express backend. Users interact via keyboard/display through browser GUI. System requires at least 256MB RAM, internet, and supports printing reports if needed.

3.1.3 Software Interfaces

Runs on Linux/Windows; uses MongoDB for databases (user, profile, project records), Elasticsearch for search. Data shared via REST APIs.

3.1.4 Communications Interfaces

Includes email (SMTP), web browser protocols (HTTPS), real-time chat (WebSockets via Socket.io). Standards: HTTP/HTTPS; security via encryption.

3.2 Functional Requirements

- **3.2.1 Register User**
  - 3.2.1.1 Collect Personal Information
    - Input:
      - Name, email, password, college ID.
    - Output:
      - View updated details; proceed to profile setup.
      - Store temporarily in database.
    - Description:
      - Validate uniqueness; no full match with existing entries.

- **3.2.2 Login User**
  - Input:
    - Email, password.
  - Output:
    - Redirect to dashboard if authenticated.
  - Description:
    - Use JWT; handle errors.

- **3.2.3 Create/Update Profile**
  - 3.2.3.1 Collect Skills and Details
    - Input:
      - Skills, certifications, portfolio links.
    - Output:
      - Updated profile view; save to MongoDB.
    - Description:
      - Validate inputs; permanent storage.

- **3.2.4 Search Skills**
  - Input:

- Keywords (e.g., "Python").
- Output:
  - List of matching profiles with highlights.
- Description:
  - Query Elasticsearch; show "No matches" if none.

❖ **3.2.5 Post Forum Question**

- Input:
  - Title, body, category.
- Output:
  - Posted question; notify followers.
- Description:
  - Save to database; validate content.

❖ **3.2.6 Answer Forum Question**

- Input:
  - Answer text.
- Output:
  - Posted answer; notify poster.
- Description:
  - Associate with question ID.

❖ **3.2.7 Rate User**

- Input:
  - 1-5 stars, optional comment.
- Output:
  - Updated peer profile.
- Description:
  - Only after collaboration; build trust scores.

❖ **3.2.8 Create Project**

- Input:
  - Title, description, required skills.
- Output:
  - Created project; notify matching users.
- Description:
  - Save and auto-match.

❖ **3.2.9 Join Project**

- Input:
  - Project selection.
- Output:
  - Added to project; notify creator.
- Description:
  - Browse and join eligible ones.

- **3.2.10 Send Message**
  - Input:
    - Message text.
  - Output:
    - Delivered in real-time; notify recipient.
  - Description:
    - Via Socket.io; timestamped.

- **3.2.11 View Analytics**
  - Input:
    - Dashboard selection.
  - Output:
    - Metrics (views, invites, ratings) in charts.
  - Description:
    - Retrieve from database.

- **3.2.12 Manage Users (Admin)**
  - Input:
    - User list; actions (view, suspend, delete).
  - Output:
    - Updated database.
  - Description:
    - Role-based; moderate content.

3.3 Behaviour Requirements

3.3.1 Use Case View

<A use case defines a goal-oriented set of interactions between external actors and the system under consideration. Since sometimes we will not be able to specify completely the behaviour of the system by just State Diagrams, we use use-cases to complete what we have already started in section 3.3.1.

TO DO: Provide a use case diagram which will encapsulate the entire system and all possible actors. Do not include detailed use case descriptions (these will be needed when you will be working on the Test Plan), but make sure to include a short description of what every use-case is, who are the actors in your diagram. For more information please refer to your UML guide and the MiniThermostat SRS example file.>

4 Other Non-functional Requirements

4.1 Performance Requirements

- Quick: Under 2s responses for 500 users.
- Searches in <1s.
- Emails in 5s.
- Chats instant.
- Scales to 5k users smoothly. Important: No lags in core features like search or chat.

4.2 Safety and Security Requirements

- Backups to prevent data loss.
- Block unauthorized actions.
- Protect privacy—no sharing without okay.
- Security: Hash passwords, JWT auth, HTTPS everywhere, admin-only moderation. Vital: Guards against hacks, ensures trust.

4.3 Software Quality Attributes

- **Reliability**
  - Error-proof with validations, 99.9% uptime, auto-recover. Meaningful: Users can count on it during crunch times like hackathons.
- **Usability**
  - Learn in <10 mins; intuitive flows. Key: Simple for busy students.
- **Scalability**
  - Grow via database tweaks; handles campus growth.
- **Maintainability**
  - Modular setup for easy updates. Important: Future-proof for new features.
- **Portability**
  - Works across browsers/OS; no lock-ins.

5 Other Requirements

<This section is **Optional**. *Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.*>

Appendix A – Data Dictionary

<Data dictionary is used to track all the different variables, states and functional requirements that you described in your document. Make sure to include the complete list of all constants, state variables (and their possible states), inputs and outputs in a table. In the table, include the description of these items as well as all related operations and requirements.>