Software Requirements Specification

for

VIDHAN : A Gamified Approach to Learning the Indian Constitution with AI Assistance.

Version 1.0.0 approved

Prepared by Yashkumar Patil, Tejas Patil,

Dhanashree Lomte, Aditi Khedkar

Sinhgad College of Engineering

Department of Computer Engineering

19 / 10 / 2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version

1. Introduction

1.1 Purpose

The purpose of this Software Requirements Specification (SRS) document is to define the functional and non-functional requirements for the web-based educational platform **VIDHAN: A Gamified Approach to Learning the Indian Constitution with AI Assistance**.

VIDHAN aims to simplify the complex language of the Indian Constitution by integrating artificial intelligence and gamified learning techniques. The system allows users—especially students to interact with an chatbot that explains constitutional articles and related topics in an easy-to-understand manner. It also includes interactive games, quizzes, and progress tracking to make learning engaging and rewarding.

This SRS outlines the system's features, interfaces, constraints, and overall behavior to ensure a clear understanding among developers, stakeholders, and testers. It covers all modules of the project, including the AI chatbot, gamified learning, admin, and sub-admin dashboards.

1.2 Document Conventions

The following conventions are used in this document:

- **Bold text** denotes section headings or key terms.
- *Italic text* highlights module names or features.
- Functional requirements are labeled as **REQ-** followed by a number (e.g., REQ-1).
- Non-functional requirements are labeled as **REQ-NF-** followed by a number.
- Priorities are classified as **High**, **Medium**, or **Low** based on their importance.
- The document follows the **IEEE Std 830-1998** format for Software Requirements Specifications.

1.3 Intended Audience and Reading Suggestions

This document is intended for:

- Developers: To understand system architecture, module interactions, and implementation logic.

- Project Managers: To monitor progress, allocate resources, and ensure requirements coverage.
- Testers: To design test cases based on defined functional and non-functional requirements.
- Educators and End Users: To gain insight into system functionality and expected outcomes.

Reading Suggestions:

- Readers new to the system should start with Section 2 (Overall Description) for a general overview.
- Developers should focus on Section 3 (External Interface Requirements) and Section 4 (System Features).
- Testers and managers should review Sections 5 and 6 for non-functional and other operational requirements.

1.4 Product Scope

VIDHAN is a **React + Java-based interactive learning platform** designed to promote constitutional literacy using technology and gamification.

It helps users explore, understand, and memorize key articles of the Indian Constitution through:

- AI-Powered Chatbot Assistance: Answers queries like *"What is Article 307?"*, *"Explain Article 304"*, and suggests related provisions.
- Gamified Learning Environment**:** Includes quiz battles, drag-and-drop article matching, and puzzles like *Unlock the Preamble*.
- Personalized Learning Dashboard: Tracks user performance, rewards, and progress.
- Admin & Sub-Admin Dashboards**:** Manage content, daily updates, and quiz/game moderation.

The project's goal is to make learning civic education more engaging by blending knowledge, play, and AI-powered assistance.

It aligns with national educational goals by promoting awareness of fundamental rights, duties, and constitutional principles among learners.

1.5 References

The following documents and resources were used in preparing this SRS:

1. IEEE Std 830-1998 – *IEEE Recommended Practice for Software Requirements Specifications*.
2. *The Constitution of India*, Government of India – https://legislative.gov.in/constitution-of-india.
3. React.js Documentation – https://react.dev/.
4. Java Spring Boot Documentation – https://spring.io/projects/spring-boot.
5. MySQL Documentation – https://dev.mysql.com/doc/.
6. AWS Cloud Hosting Documentation – https://aws.amazon.com/.
7. Phaser.js Game Framework – https://phaser.io/.
8. WCAG 2.1 Accessibility Guidelines – https://www.w3.org/TR/WCAG21/.

2. Overall Description

2.1 Product Perspective

VIDHAN is a upgraded, standalone web-based educational platform developed to simplify and promote constitutional literacy among students and citizens.

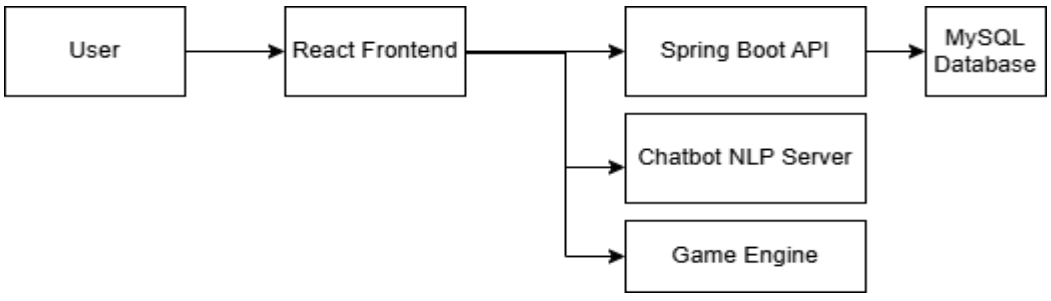
It combines conversational learning Chatbot with interactive gamification to make the study of the Indian Constitution engaging and accessible.

The system is based on a three-tier architecture:

Layer	**Technology**	**Function**
Frontend (Presentation Layer)	React.js, Material UI, Tailwind CSS, Phaser.js	User interaction, game UI, chatbot interface
Backend (Application Layer)	Java Spring Boot (REST API)	Business logic, user management, AI integration
Database Layer	MySQL	Stores user data, constitutional content, game scores
Hosting Environment	AWS	Cloud deployment and storage
AI/NLP Integration	Python-based microservice or API	Processes user queries and generates AI-driven responses

The product is designed to run independently on browsers without requiring additional installations.
It communicates via RESTful APIs, and the backend can integrate with external NLP APIs or models for enhanced chatbot responses.

Context Diagram Overview:



2.2 Product Functions

The main functions of the VIDHAN system are summarized as follows:

1. **AI Chatbot Assistance**: Provides intelligent explanations of constitutional articles and related information through text or voice-based input.
2. **Gamified Learning Module:** Engages users through quizzes, puzzles, and mini-games designed to test and reinforce knowledge.
3. **Learning Dashboard:** Displays user progress, completed modules, badges, and recommended topics.
4. **Admin Dashboard**: Allows authorized administrators to manage constitutional content, articles, schedules, and amendments.
5. **Sub-Admin Dashboard:** Enables content moderators to manage daily quiz updates, challenges, and user feedback.
6. **User Profile Management:** Allows users to register, log in, view progress, and update personal information.
7. **Leaderboard & Rewards**: Tracks user rankings, badges, and achievements to encourage participation.

Each of these functions interacts with the backend APIs and database to ensure data persistence, real-time updates, and secure access.

2.3 User Classes and Characteristics

User Class	Description	Technical Expertise	Access Level
Student/User	Learns about the Constitution through chatbot and games.	Basic computer knowledge	Limited (view & play)
Teacher	Uses the system for teaching, monitoring, and assigning activities.	Intermediate	Moderate (monitor & create content)
Admin	Manages articles, users, and AI dataset.	Advanced	Full control
Sub-Admin	Manages daily quizzes, updates, and moderation tasks.	Intermediate	Partial administrative rights

The system is designed to be easy to use for students while also offering powerful management tools for administrators**.**

2.4 Operating Environment

VIDHAN will function as a cloud-hosted web application accessible via standard web browsers and mobile devices.

Client-Side Environment:

- Operating Systems: Windows, macOS, Android, iOS
- Browsers: Chrome, Edge, Firefox, Safari
- Internet Connectivity: Minimum 2 Mbps
- Display: Responsive design supporting 720p and above

Server-Side Environment:

- Hosting: AWS
- Database: MySQL
- Backend: Java Spring Boot
- Frontend: React.js
- Optional: Python microservice for AI NLP processing
- Storage: AWS

2.5 Design and Implementation Constraints

- The system must adhere to React and Java Spring Boot frameworks for frontend and backend development.
- Database shall be implemented using **MySQL**, and hosted on AWS
- Network communication must follow HTTPS and RESTful API protocols.
- Integration of NLP chatbot requires either a local model or external API (e.g., OpenAI API, Hugging Face).
- Games must be built using Phaser.js or React Game Kit to maintain browser compatibility.
- Access control must be enforced using JWT-based authentication.
- The design should comply with WCAG 2.1 accessibility standards.
- Latency for chatbot response should not exceed 5 seconds.

2.6 User Documentation

The following user documentation will be provided with the product:

1. **User Manual**: Step-by-step guide for navigating and using VIDHAN's modules.
2. **Quick Start Guide**: For first-time users to set up accounts and access chatbot/games.
3. **Online Help Section**: Built-in FAQ and troubleshooting support.
4. **Video Tutorials**: Demonstrations of gameplay, AI chat usage, and learning tools.
5. **Admin Guide**: Instructions for managing content, users, and game updates.

Documentation will be available in **HTML** and accessible from within the website.

2.7 Assumptions and Dependencies

Assumptions:

- Users will have a stable internet connection.
- The AI chatbot API or local NLP model will be functional and updated regularly.
- AWS hosting services will provide reliable uptime and performance.
- Users will access the system primarily through modern browsers.

Dependencies:

- The chatbot module depends on NLP libraries or third-party APIs for accurate language interpretation.
- Gamified modules rely on Phaser.js/React Game Kit libraries.
- Database and hosting depend on AWS and MySQL.
- The performance of the AI assistant depends on the response time of the NLP engine and server bandwidth.
- The project relies on continuous updates to the Constitutional content dataset managed by Admin and Sub-Admin.

3. External Interface Requirements

3.1 User Interfaces

The VIDHAN web application shall provide a responsive, intuitive, and visually interactive user interface designed for students, teachers, and administrators.

It shall follow modern web design standards, ensuring usability and accessibility on both desktop and mobile devices.

User Interface Features:

- **Homepage:** Overview of modules : *Learn, Play, Chatbot, Dashboard, About Us*.
- **AI Chatbot Window:**
    - Floating assistant on every page.
    - Accepts text and voice input.
    - Displays chat history and related articles.
- **Gamified Learning Screen:**
    - Displays interactive quiz or puzzle interface.
    - Shows timer, question progress, and points.
- **Leaderboard Section:**
    - Displays top scorers, user badges, and achievements.
- **User Dashboard:**
    - Shows learning statistics, completed modules, and recommended topics.
- **Admin/Sub-Admin Dashboard:**
    - Provides content management tools for articles, quizzes, and game updates.

Design Guidelines:

- Built with React.js for dynamic rendering.
- Uses Material UI and Tailwind CSS for styling.
- Supports dark and light themes.
- Accessible via standard browsers (Chrome, Edge, Firefox, Safari).
- Follows WCAG 2.1 accessibility standards (keyboard navigation, screen reader support).

3.2 Hardware Interfaces

The VIDHAN platform is designed for standard client-server interaction over the internet. There are no specialized hardware dependencies beyond typical computing devices.

Client-Side Requirements:

- Device: Desktop, laptop, tablet, or smartphone.
- Minimum RAM: 4 GB.
- CPU: Dual-core 2.0 GHz or higher.
- Input: Keyboard, mouse, or touchscreen for gameplay and chatbot interaction.

Server-Side Requirements:

- Hosting: AWS
- Storage: AWS
- Database: MySQL
- Minimum Server Specs: 8 GB RAM, 4-core CPU, 100 GB SSD storage.

3.3 Software Interfaces

The software will integrate multiple frameworks and components for its functionality.

Frontend Interfaces:

- **React.js** for building the user interface.
- **Axios** for making REST API requests to the backend.
- **Phaser.js / React Game Kit** for game modules.

- **React Router** for navigation control.

Backend Interfaces:

- **Java Spring Boot REST API** to handle user authentication, chatbot processing, and content management.
- **MySQL Database** for storing user profiles, scores, articles, and chatbot responses.
- **Python NLP** for advanced question understanding and contextual responses.

Data Flow:

User request → React frontend → Spring Boot API → MySQL Database → Response returned as JSON → Displayed in UI.

3.4 Communications Interfaces

Communication Protocols:

- The system shall use HTTPS for all client-server communications to ensure data confidentiality and integrity.
- All API communications between the frontend and backend shall follow RESTful principles using JSON data format.

Network Interfaces:

- Client devices communicate with the website server via standard internet connection.
- Internal communication between backend services (e.g., Java ↔ Python NLP microservice) shall use secure local API endpoints.

Security & Synchronization:

- All data transfers encrypted with SSL/TLS.
- Authentication managed through JWT tokens.
- User sessions synced with the database for persistent login and progress tracking.
- Chatbot service uses WebSocket or long polling for real-time conversational response.

Communication Standards:

- **HTTP/HTTPS** for web requests.
- **Web-Socket** for real-time chatbot communication.
- **CORS policy** configured to allow frontend-backend interaction from authorized domains only.

4. System Features

4.1 AI Chatbot Assistant

4.1.1 Description and Priority:

This is a **High Priority** feature.

The AI Chatbot serves as an intelligent learning companion that assists users in understanding the Constitution.

It answers user queries such as *"What is Article 307?"*, *"Explain Article 304"*, or *"Which Articles are related to Article 302?"*.

It provides concise explanations, related sections, and contextual information using NLP (Natural Language Processing) techniques.

4.1.2 Stimulus/Response Sequence:

- **User Input:** The user enters or speaks a question related to the Indian Constitution.
- **System Response:** The chatbot processes the question using the NLP model and retrieves relevant content from the database.
- **Output:** The AI assistant displays the explanation, highlights related articles, and may suggest quiz links for practice.

User inputs a question → Chatbot processes via NLP → Displays structured explanation or related articles.

4.1.3 Functional Requirements:

- **REQ-1:** The system shall accept user queries in text input.
- **REQ-2:** The system shall interpret questions using NLP and return relevant article information.
- **REQ-3:** The system shall display related articles and definitions for each query.
- **REQ-4:** The system shall store frequently asked questions for quicker future responses.
- **REQ-5:** The system shall allow voice input for accessibility.

4.2 Gamified Learning Module

4.2.1 Description and Priority:

This is a **High Priority** feature.

The gamified learning module enhances engagement and comprehension through interactive games that reinforce constitutional knowledge.

The games are built using Phaser.js**,** React Game Kit**,** or React-Three-Fiber for rich interactivity.

4.2.2 Examples of Games:

- **Quiz Battle:** Timed multiple-choice quizzes.
- **Match the Article:** Drag and drop articles to their titles.
- **Unlock the Preamble:** Puzzle game using article clues.
- **Leaderboard:** Displays user ranks and rewards.

4.2.3 Stimulus/Response Sequences:

1. **User Action:** The user selects a game type (e.g., quiz battle, article puzzle).
2. **System Response:** The system loads the selected game and displays instructions.
3. **User Plays:** The user answers questions or completes challenges.
4. **System Response:** The system evaluates performance, awards points, and updates the leaderboard.

User Selects a Game → Loads Game and Logic → User Plays Games → Progress is Recorded → System Evaluates Performance → Awards Points Updates Leaderboard

4.2.4 Functional Requirements:

- **REQ-6:** The system shall offer various games like Quiz Battles, Match the Article, and Unlock the Preamble.
- **REQ-7:** The system shall update the user's score and progress after each game.
- **REQ-8:** The system shall maintain a leaderboard based on user performance.

- **REQ-9:** The system shall display rewards, badges, or achievements upon successful completion.
- **REQ-10:** The system shall store game results in the database for future analytics.

4.3 Learning Dashboard

4.3.1 Description and Priority:

This is a **Medium Priority** feature.

The Learning Dashboard acts as the user's personal console to monitor learning progress, completed quizzes, achievements, and recommended next steps.

It also provides visual insights such as graphs and statistics of learning performance.

4.3.2 Stimulus/Response Sequences

1. **User Login:** The user logs into their account.
2. **System Response:** The dashboard displays current progress, unlocked levels, and suggested modules.
3. **User Interaction:** The user clicks a topic or game for continuation.
4. **System Response:** The system loads the respective module or game page.

4.3.3 Functional Requirements:

- **REQ-11:** The system shall show user progress with visual indicators (progress bar or graph).
- **REQ-12:** The system shall display completed, pending, and upcoming modules.
- **REQ-13:** The system shall provide personalized recommendations based on user activity.
- **REQ-14:** The system shall sync progress data from the AI chatbot and games.
- **REQ-15:** The system shall provide download or share options for performance reports.

4.4 Admin Dashboard (Constitutional Data Management)

4.4.1 Description and Priority

This is a **High Priority** feature.

The Admin Dashboard allows the administrator to manage constitutional content such as articles, schedules, amendments, and learning resources.

Admins can add, edit, or remove constitutional data and manage access rights for sub-admins.

4.4.2 Stimulus/Response Sequences

1. **Admin Login:** The administrator authenticates using secure credentials.
2. **System Response:** The dashboard displays options for managing content, users, and analytics.
3. **Admin Action:** The admin updates constitutional articles or adds new sections.
4. **System Response:** The system saves updates and refreshes data for public view.

4.4.3 Functional Requirements

- **REQ-16:** The system shall allow the admin to add, edit, or delete constitutional data.
- **REQ-17:** The system shall maintain version control for all article updates.
- **REQ-18:** The system shall restrict data update permissions to verified admin accounts.
- **REQ-19:** The system shall log every change with date, time, and user ID.
- **REQ-20:** The system shall synchronize updated content across all modules.

4.5 Sub-Admin Dashboard (Daily Task and Game Management)

4.5.1 Description and Priority

This is a **Medium Priority** feature.

The Sub-Admin Dashboard is designed for sub-administrators who handle daily operational activities, such as adding quiz questions, updating games, and moderating user-generated feedback.

4.5.2 Stimulus/Response Sequences

1. **Sub-Admin Login:** The sub-admin logs into the dashboard with verified credentials.
2. **System Response:** Displays modules for managing quizzes, games, and user feedback.
3. **Sub-Admin Action:** Updates quiz content or uploads new daily challenges.
4. **System Response:** The updates are reflected on the public learning and gaming modules.

4.5.3 Functional Requirements

- **REQ-21:** The system shall allow sub-admins to manage quizzes and daily challenges.
- **REQ-22:** The system shall permit sub-admins to update or replace existing games.
- **REQ-23:** The system shall allow sub-admins to view and resolve user-reported issues.
- **REQ-24:** The system shall restrict sub-admin privileges to non-critical operations only.
- **REQ-25:** The system shall notify the main admin of all changes made by sub-admins.

5. Other Non-functional Requirements

5.1 Performance Requirements

- **REQ-NF1:** The system shall handle a minimum of 500 concurrent users without degradation in response time.
- **REQ-NF2:** The average page load time shall not exceed 3 seconds under normal network conditions.
- **REQ-NF3:** The AI chatbot shall provide responses within 2 seconds for common queries and within 5 seconds for complex queries requiring database or API lookup.
- **REQ-NF4:** The game modules shall operate smoothly at a minimum of 30 FPS (frames per second) to ensure seamless user experience.
- **REQ-NF5:** The system shall maintain a server uptime of 99.5%**.**

5.2 Safety Requirements

- **REQ-NF6:** The system shall automatically back up all critical data (user progress, scores, constitutional content) to AWS.
- **REQ-NF7:** The system shall provide an auto-save feature during gameplay and AI interactions to prevent data loss due to unexpected shutdowns.
- **REQ-NF8:** The system shall restrict unauthorized modification or deletion of constitutional data to prevent corruption of learning content.
- **REQ-NF9:** The system shall include error-handling mechanisms to safely recover from unexpected crashes or interruptions without losing user data.
- **REQ-NF10:** All application logs shall be monitored for anomalies or failures to prevent cascading errors.

5.3 Security Requirements

- **REQ-NF11:** All communication between frontend and backend shall be encrypted using HTTPS with SSL/TLS protocols.
- **REQ-NF12:** The system shall implement JWT (JSON Web Token) authentication for user sessions to ensure secure access.

- **REQ-NF13:** All user passwords shall be stored using hashing algorithms (e.g., bcrypt or SHA-256).
- **REQ-NF14:** Role-based access control shall be enforced:
    - Admin: Full system privileges.
    - Sub-Admin: Limited content and quiz management privileges.
    - User: Learning and gameplay access only.
- **REQ-NF15:** The system shall automatically log out inactive users after 15 minutes of inactivity for security purposes.
- **REQ-NF16:** The system shall prevent SQL injection**,** cross-site scripting (XSS), and cross-site request forgery (CSRF) attacks.

5.4 Software Quality Attributes

Attribute	**Description**
Usability	The interface shall be intuitive, visually appealing, and accessible on both desktop and mobile browsers.
Reliability	The system shall provide consistent and accurate responses from the AI chatbot and ensure quiz data integrity.
Scalability	The architecture shall support horizontal scaling to handle increased user load using AWS Auto Scaling.
Maintainability	Code shall follow modular design using React components and RESTful APIs for easy updates.
Portability	The system shall operate across major browsers (Chrome, Edge, Firefox) and platforms (Windows, Android, iOS).
Availability	Minimum 99.5% uptime.

5.5 Business Rules

- **REQ-NF17:** Only registered users can access personalized progress tracking, leaderboards, and AI chat history.
- **REQ-NF18:** Admins shall verify all new constitutional updates before publication.
- **REQ-NF19:** Sub-admins shall manage daily quiz/game updates but cannot alter core constitutional data.

- **REQ-NF20:** Users must agree to the Terms of Use and Privacy Policy before registration.
- **REQ-NF21:** The platform shall not display political bias or misinformation in any AI-generated explanation or quiz content.
- **REQ-NF22:** Any modification to the AI chatbot dataset or constitutional content shall require approval from the primary admin.

6. Other Requirements

- The system shall comply with Indian educational content standards.
- Database shall store all quiz logs for analytics.
- Use AWS SES for user notifications and verification emails.

Appendix A: Glossary

Term	**Description**
NLP	Natural Language Processing
UI	User Interface
AWS	Amazon Web Services
AI	Artificial Intelligence
JWT	JSON Web Token

Appendix B: Analysis Models

Includes:

- **Use Case Diagram**
- **DFD Level 0 & 1**
- **ER Diagram**
- **Class Diagram**

Appendix C: To Be Determined List

TBD No.	**Description**
TBD-1	Details about Integration of Chatbot module
TBD-2	Final Selection of Game Library(Phaser.js or React Game Kit)