Software Requirements Specification (SRS) for AAJNVL Official Website

1. Introduction

1.1 Purpose

This document outlines the software requirements for the official website of the Alumni Association of Jawahar Navodaya Vidyalaya Lepakshi (AAJNVL). The website aims to connect alumni, facilitate registrations, showcase activities, provide support networks, and enable administrative management. It will serve as a central platform for alumni engagement, event management (e.g., NPL 2025), and contributions to the alma mater and society.

The primary goals are:

- Foster alumni connections and support.
- Ensure secure data handling and user verification.
- Provide an intuitive, mobile-responsive interface.
- Support scalability for future enhancements like mobile apps or AI integrations.

1.2 Scope

The website will include public-facing pages (e.g., Home, About, Gallery) and authenticated sections (e.g., Registration Portal, Nava Bandham Seva). It covers alumni and faculty registrations, event management, directories, idea submissions, and admin tools. Out-of-scope items include:

- Development of a separate mobile app (listed as a future enhancement).
- Integration with external systems like WhatsApp groups or Google Calendar (optional future features).
- Payment gateways for sponsorships or donations (unless specified).

1.3 Definitions and Acronyms

- **AAJNVL**: Alumni Association of Jawahar Navodaya Vidyalaya Lepakshi.
- **JNV**: Jawahar Navodaya Vidyalaya.
- **NPL**: Navodaya Premier League (cricket event in 2025).
- **OTP**: One-Time Password for verification.
- **CAPTCHA**: Completely Automated Public Turing test to tell Computers and Humans Apart.
- **Admin Dashboard**: Secure backend interface for governing body to manage content and users.
- **Alumni ID**: Unique identifier generated post-registration (e.g., AAJNVL-YYYY-XXX).

1.4 References

- Original Requirement Document: Provided by AAJNVL.
- Example Website: https://www.nittrichyalumni.org/ (for design inspiration, especially activities section).

1.5 Overview

This SRS is structured into overall description, specific requirements, and supporting information.

2. Overall Description

2.1 Product Perspective

The website builds on the legacy of JNV Lepakshi, emphasizing values like "Enter to Learn, Leave to Serve." It replaces any informal communication channels (e.g., social media groups) with a centralized, secure platform. Inspired by sites like nittrichyalumni.org, it focuses on community building, event promotion, and support services.

2.2 Product Functions

- **Public Access**: View home, about, activities, gallery, sponsors, and contact pages.
- **User Registration**: Secure forms for alumni and faculty with OTP verification and admin approval.
- **Authenticated Features**: Access directories, submit ideas, request help, view job postings.
- **Event Management**: Dedicated section for NPL 2025 with registrations, schedules, and live updates.
- **Support Network (Nava Bandham Seva)**: Directories for health, career, and emergency assistance.
- **Admin Functions**: Manage users, content, events, and data exports.
- **Content Management**: Dynamic sections for activities, achievements, and galleries.

2.3 User Classes and Characteristics

- **Public Visitors**: General users browsing information; no login required.
- **Alumni Users**: Registered and approved alumni; access to directories and support features.
- **Faculty Users**: Registered current/retired faculty; limited access to directories.
- **Governing Body/Admins**: Full access to dashboard for approvals, updates, and exports.
- **Sponsors**: View-only for their listings; submissions via form.

Assumptions: Users have basic internet access and familiarity with web forms. Mobile responsiveness is critical for rural users.

2.4 Operating Environment

- **Client-Side**: Modern web browsers (Chrome, Firefox, Safari, Edge) on desktop/mobile.
- **Server-Side**: Cloud-hosted (e.g., Firebase backend).
- **Tech Stack**: Firebase (auth, database, storage), Next.js (frontend framework), Tailwind CSS (styling).

2.5 Design and Implementation Constraints

- Theme: Clean, modern layout using school colors (blue, white, saffron).
- Responsiveness: Mobile-first design.
- Security: HTTPS, data encryption, role-based access.
- Performance: Load times < 3 seconds; scalable for 1000+ users.

2.6 Assumptions and Dependencies

- Assumptions: Governing body provides content (e.g., logos, photos); users consent to data usage.
- Dependencies: Third-party services for OTP (e.g., via Firebase Auth), social media embeds, and potential email APIs.

3. Specific Requirements

3.1 External Interface Requirements

- **User Interfaces**: Responsive web pages with navigation menu, forms, carousels, and grids.
- **Hardware Interfaces**: None (web-based).
- **Software Interfaces**: Integration with social media (links), YouTube/Facebook for embeds, and potential Google Maps (future).
- **Communication Interfaces**: HTTPS for all traffic; email/SMS for OTP and notifications.

3.2 Functional Requirements

3.2.1 Home Page

- Display header with logo, titles, social icons, and navigation menu.
- Sections: Intro, quick links to activities/events.

3.2.2 About JNV Lepakshi

- Intro paragraph, campus photo, timeline of milestones.

3.2.3 About AAJNVL

- Motto, highlights (formation year, batches, registered alumni count – auto-updated).
- Batch details with search/filter.

3.2.4 Governing Body

- Card-based display with photos, details, and masked contacts.
- Admin editable.

3.2.5 Activities

- Carousel/ticker for categories (e.g., Career Guidance, NPL 2025).
- Each links to detail pages with photos and contacts.

3.2.6 Registration Portal

- **Alumni Form**: Fields as specified; OTP on email/mobile; duplicate checks; photo upload.
- **Faculty Form**: Similar fields; OTP verification.
- Post-submission: Admin approval; generate Alumni ID; login access.

3.2.7 NPL 2025

- Committee display.
- Player/team registration forms.
- Schedule, points table (admin-updatable).
- Live streaming embeds, gallery.

3.2.8 Nava Bandham Seva

- **Health Assistance**: Searchable directory; request help button (masked).
- **Career Assistance**: Directory with filters; job posting form; mentorship tags.
- **Emergency Support**: Blood donor search; broadcast requests.

- Privacy: Masked contacts, consent-based visibility.

3.2.9 Navodayan Spark

- Idea submission form; admin review; public display.

3.2.10 Navodaya Patrons

- Sponsor listings with logos/messages; rotating banner; admin approval.

3.2.11 Admin Dashboard

- User management (approve/reject/edit).
- Content updates (news, events, achievements).
- Data exports (CSV/Excel/PDF).
- Notifications/newsletters.

3.2.12 Footer

- Contact info, social links, copyright.

3.3 Non-Functional Requirements

- **Performance**: Handle 500 concurrent users; fast queries via indexed database.
- **Security**: Encrypt sensitive data (Aadhaar, contacts); CAPTCHA/OTP; role-based access; regular backups.
- **Usability**: Intuitive navigation; accessibility (ARIA labels, alt text for images).
- **Reliability**: 99% uptime; error handling for forms.
- **Scalability**: Support growing user base; modular design.
- **Maintainability**: Clean code; documentation.

3.4 Supporting Information

- Testing: Unit, integration, user acceptance.
- Deployment: Continuous integration via Git/Vercel.
- Maintenance: Annual updates for governing body changes.

Tech Stack Feasibility

Yes, it is possible to build this website with Firebase, Next.js, and Tailwind CSS. Here's why:

- **Firebase**: Handles authentication (OTP via Firebase Auth), real-time database (Firestore for users, activities, directories), storage (for photos/uploads), and Cloud Functions for server-side logic (e.g., approvals, notifications, data exports). It also supports HTTPS and encryption out-of-the-box.
- **Next.js**: Ideal for SSR/SSG for SEO-friendly pages, dynamic routing (e.g., for activities details), and forms. It integrates seamlessly with Firebase.
- **Tailwind CSS**: Perfect for rapid, responsive styling with custom themes (e.g., school colors).

No major additions needed, but consider:

- **Optional Extras**:
  - Twilio or similar for SMS OTP if Firebase's email OTP isn't sufficient (Firebase supports phone auth natively).
  - React libraries like React Hook Form for forms, Swiper for carousels, or TanStack Query for data fetching.
  - Vercel for deployment (free tier sufficient for start).
- Limitations: If heavy compute is needed (e.g., AI chat in future), Firebase Functions have limits; might need to add Google Cloud Run later.

Suggested Architecture

High-Level Architecture

- **Frontend Layer**: Next.js app with pages/routes for each section (e.g., /home, /registration). Use Tailwind for styling. Components: Reusable cards, forms, carousels.
- **Backend Layer**: Firebase as BaaS (Backend-as-a-Service).
  - **Auth**: Firebase Authentication for logins, OTP.

    - **Database**: Firestore collections (e.g., users, activities, npl_teams) with security rules for role-based access.
    - **Storage**: Firebase Storage for images/photos.
    - **Server-Side Logic**: Cloud Functions for triggers (e.g., on registration: send approval email; on approval: generate ID).
- **Data Flow**:
    1. User interacts with Next.js UI.
    2. API calls to Firebase (via SDK) for CRUD operations.
    3. Admin dashboard as a protected route in Next.js, fetching data via Firebase Admin SDK.
- **Deployment**:
    - Host Next.js on Vercel (automatic CI/CD).
    - Firebase for backend; sync via GitHub actions.
- **Security Architecture**: Use Firebase Security Rules to enforce access (e.g., only admins can approve users). Client-side validation + server-side checks.
- **Scalability**: Firestore scales automatically; Next.js supports incremental static regeneration for static pages.
- **Development Flow**:
    - Set up Next.js project with Tailwind.
    - Integrate Firebase SDK.
    - Build pages iteratively: Start with static (Home/About), then dynamic (Registrations), then admin.
    - Test with emulators (Firebase Local Emulator Suite).

This architecture is cost-effective (Firebase free tier for low traffic) and quick to prototype.