# 1) Project Overview

**Project Name:** Numerology Analysis Software (NAS)

**One-liner:** Privacy-first, production-grade numerology insights app that generates core numerology numbers, reports, and personalized recommendations with an auditable calculation engine.

**Disclaimer:** Numerology is an interpretive, non-scientific practice. The app must clearly state that results are for entertainment/self-reflection, not medical, legal, or financial advice.

---

# 2) Goals & Success Criteria

**Primary Goals** - Compute standard numerology profiles (Life Path, Expression, Soul Urge, Personality, Maturity, Karmic Lessons, etc.) accurately and consistently. - Provide clean, multilingual UX with shareable, exportable reports (PDF/HTML). - Ensure security (PII minimization), auditability (reproducible calculations), and scalability (10k–100k MAU).

**Success Metrics** - >98% unit test coverage on calculation engine. - P95 request latency < 300ms for calculations. - Error rate < 0.1% over 30 days. - Report generation time < 2s (P95).

---

# 3) Scope

**In-Scope** - User flows: quick analysis (no login), saved profiles (with login), multi-profile comparison. - Calculation engine: Pythagorean & Chaldean systems; Western date rules (with optional locale calendars later). - Features: detailed report with interpretations, compatibility analysis (opt-in), name variant suggestions, lucky days/colors (config-driven), export to PDF, share via link. - Admin console: manage interpretation texts, rule versions, feature flags.

**Out-of-Scope (v1)** - Astrology, palmistry, or tarot integrations. - Payments/subscriptions (plan for v1.1+). - Native mobile apps (use responsive PWA instead).

---

# 4) User Personas & Use Cases

**Personas** - *Seeker (Guest):* wants a quick profile without creating an account. - *Enthusiast (Registered):* saves multiple profiles, compares, exports reports. - *Consultant (Pro):* prepares client reports, customizes interpretations. - *Admin/Editor:* curates interpretation content, toggles features.

**Core Use Cases** 1. Enter name & DOB → view core numbers & interpretations. 2. Save profile to account; revisit and export as PDF. 3. Compare two profiles for compatibility. 4. Admin updates interpretation text or rule version, publishes to staging → production.

---

# 5) Functional Requirements (SRS)

**FR-1 Input & Validation** - FR-1.1: Accept name (Unicode), optional nickname, and DOB (YYYY-MM-DD). Validate date (Gregorian) and age ≥ 13 for account creation. - FR-1.2: Locale-aware name handling (remove diacritics or map via configurable transliteration).

**FR-2 Calculation Engine** - FR-2.1: Support calculation systems: Pythagorean and Chaldean (toggle at runtime). - FR-2.2: Compute: Life Path (sum of DOB digits reduced to 1–9/11/22/33), Expression (full name letters → numbers → reduced), Soul Urge (vowels only), Personality (consonants only), Maturity (Life Path + Expression → reduced), Birthday number (day of month), Karmic Lessons (missing digits 1–9 in name), Balance number (initials), Pinnacles & Challenges (by life cycles), Personal Year/Month/Day. - FR-2.3: Master numbers (11, 22, 33) preserved where applicable; support rule configs (e.g., when to reduce). - FR-2.4: Versioned rules and mappings; every result tagged with `ruleset_id`.

**FR-3 Interpretations & Content** - FR-3.1: For each computed number, fetch interpretation text per locale. - FR-3.2: Compatibility report (e.g., Expression vs. Soul Urge compatibility matrix; configurable). - FR-3.3: Lucky attributes (days, colors, gemstones) from content table or CMS.

**FR-4 Reporting** - FR-4.1: Generate shareable HTML report and downloadable PDF. - FR-4.2: Include inputs, computed numbers, brief methodology, and disclaimer.

**FR-5 Accounts & Profiles** - FR-5.1: OAuth (Google/Email+Password) with email verification. - FR-5.2: Users can create, read, update, delete profiles they own. - FR-5.3: Soft delete & data export (GDPR-like).

**FR-6 Admin & Content Ops** - FR-6.1: Role-based access (Admin, Editor, Support). - FR-6.2: Edit interpretation texts, publish via staging → production with approvals. - FR-6.3: Feature flags to roll out new rules or texts gradually.

**FR-7 Internationalization** - FR-7.1: UI strings i18n; at least English + Hindi. - FR-7.2: Name transliteration strategies per locale (config-driven).

**FR-8 Observability** - FR-8.1: Structured logs with correlation IDs. - FR-8.2: Metrics: request latency, error rates, calc anomalies, PDF failures.

---

# 6) Non-Functional Requirements (SRS)

- **Security & Privacy:** PII minimization (store only necessary; allow guest mode), salted+hashed passwords (Argon2/BCrypt), JWT with rotation, HTTPS everywhere, rate limit (e.g., 100 req/min/IP), content security policy.
- **Reliability:** 99.9% uptime target; blue-green deploys; daily backups retained 14 days.
- **Performance:** P95 < 300ms for calc API; P95 < 2s for PDF; concurrency baseline 300 RPS.
- **Scalability:** Stateless services; horizontal scale; CDN for static assets and PDFs.
- **Compliance:** GDPR-like data export/delete; clear disclaimer in UI and reports.
- **Accessibility:** WCAG 2.1 AA; keyboard navigation; ARIA labels.
- **Maintainability:** 90%+ unit coverage on core engine; typed APIs; linting + formatting.

---

# 7) Domain Model & Data Design (ERD summary)

**Key Entities** - `users(id, email, pass_hash, name, locale, created_at, role)` - `profiles(id, user_id, full_name, dob, system, locale, created_at, deleted_at)` - `calculations(id, profile_id, ruleset_id, payload_json, results_json, created_at)` - `interpretations(id, key, locale, body_md, version, published_at)` - `compatibility_rules(id, version, matrix_json)` - `rulesets(id, name, mapping_json, reduction_rules_json, active)` - `audit_logs(id, actor_id, action, entity, entity_id, before_json, after_json, ts)` - `feature_flags(key, value_json, env)`

**Indexes** - `profiles(user_id)`, `calculations(profile_id, created_at DESC)`, `interpretations(key, locale)`

**Storage** - PostgreSQL (relational), S3/GCS for PDFs.

---

# 8) Calculation Details (spec excerpts)

**Pythagorean Letter Mapping (A1–Z26 → 1–9)**

```
1: A J S
2: B K T
3: C L U
4: D M V
5: E N W
6: F O X
7: G P Y
8: H Q Z
9: I R
```

Reduce sums to 1–9 while preserving master numbers 11/22/33 where rules apply.

**Core Numbers** - *Life Path:* Sum of YYYY+MM+DD digits, reduce; keep 11/22/33. - *Expression (Destiny):* All letters in full birth name → mapped → sum → reduce. - *Soul Urge (Heart's Desire):* Vowels only (A,E,I,O,U; Y per rule flag), sum → reduce. - *Personality:* Consonants only, sum → reduce. - *Maturity:* Expression + Life Path (pre-reduction) → reduce. - *Birthday:* Day of month → reduce. - *Balance:* Sum initials' values → reduce. - *Karmic Lessons:* Digits (1–9) absent from name mapping results. - *Pinnacles & Challenges:* Based on life cycles (configurable formulas); document exact steps in `rulesets` JSON with examples and unit tests.

**Chaldean System** (alternative mapping) - Maintain `mapping_json` per ruleset; engine selects by `profiles.system`.

**Determinism & Audit** - For any input `(name, dob, ruleset_id)`, the engine returns a deterministic `results_json` plus a `steps[]` trail with intermediate sums.

---

# 9) System Architecture (SDS)

**Recommended Stack (plays well with your Java skills)** - **Frontend:** React + Vite (TypeScript), Tailwind, i18next, React Query. - **Backend:** Java 21 + Spring Boot 3 (Web, Security, Validation), Spring Data JPA, MapStruct. - **Calc Engine:** Separate Spring module (`nas-calc-core`) published as internal library; pure Java for performance & testability. - **DB:** PostgreSQL 15; Flyway for migrations. - **Cache:** Redis (rate limiting, session blacklist, hot interpretations). - **Search/Logs:** OpenSearch/ELK; Grafana + Prometheus for metrics. - **PDF:** wkhtmltopdf via Headless Chrome/Playwright service. - **Auth:** JWT (access+refresh) with rotation; OAuth2 (Google) via Spring Security. - **Infra:** Docker, Kubernetes (or Render/Fly.io to start), CloudFront/Cloudflare CDN.

**High-Level Components** - *Gateway/API:* Auth, rate limit, routing. - *Profile Service:* CRUD profiles; triggers calculations. - *Calc Service (library inside Profile Service for v1):* Deterministic numerology engine. - *Content Service:* Interpretations, rulesets, compatibility matrices. - *Report Service:* HTML template → PDF; stores to object storage. - *Admin UI:* Manage content, feature flags, releases.

**Key APIs (sample)**

```
POST   /api/v1/calc/profile        { fullName, dob, system }
GET    /api/v1/calc/{calcId}       → results_json + steps[]
POST   /api/v1/profiles            → create user profile
GET    /api/v1/profiles/{id}
GET    /api/v1/profiles/{id}/report → presigned URL
GET    /api/v1/content/interp?key=expression_3&locale=en
POST   /api/v1/admin/interp        (Admin)
```

**Sequence (Quick Analysis)** 1. Client validates inputs → POST /calc/profile. 2. API invokes Calc Engine with `ruleset_id` and inputs. 3. Engine returns `results_json` + `steps[]`. 4. API looks up interpretation texts per key/locale. 5. Compose response; optional background job creates PDF.

**Security** - JWT rotation, short-lived access tokens (15m), refresh (7d), IP+device binding (optional). - Input sanitization, output encoding, CSP, rate limiting, WAF. - Data minimization: no storage in guest mode; hashed email, avoid storing raw DOB unless user saves profile.

**Observability** - Correlation ID per request; structured logs. - Metrics: calc latency, PDF generation time, cache hit rate, 4xx/5xx. - Alerts on error spikes, slow queries, memory.

---

# 10) UI/UX (Brief)

- Stepper form: Name → DOB → System (Pythagorean/Chaldean) → Results.
- Results page: card per number with explanation + expand for steps.
- Sticky disclaimer; Export buttons (PDF/Share).
- Accessibility: keyboard first; high contrast mode.

---

# 11) Test Strategy

**Unit Tests** - 100+ golden test cases with fixed inputs and expected outputs for each ruleset. - Edge cases: leap day DOB, diacritics (e.g., “José”), names with hyphens/apostrophes, single-name cultures.

**Integration Tests** - API contracts (Spring MockMvc), DB migrations, PDF service.

**E2E Tests** - Playwright flows: quick analysis, save profile, export PDF.

**Non-Functional** - Load test (k6): 300 RPS sustained, spike to 1k RPS. - Security test: OWASP ZAP, dependency scanning (Snyk), secrets scan (git-secrets).

---

# 12) DevOps & Deployment

**Environments:** `dev` → `staging` → `prod` with separate DBs.

**CI/CD (GitHub Actions)** - Build → Test → Static analysis → Docker build → Push → Deploy (staging) → Manual approval → Prod.

**Versioning** - Semantic versioning for services; `ruleset_id` for numerology logic; content versions for interpretations.

**Backups & DR** - Nightly DB snapshots; object storage lifecycle rules; restore runbook.

---

# 13) Product Backlog (MVP → v1.2)

**MVP (Sprint 1–2)** - Core UI (Name/DOB form, results page) - Pythagorean engine (Life Path, Expression, Soul Urge, Personality, Maturity, Birthday, Balance, Karmic Lessons) - English interpretations (seed content) - HTML report export (no PDF yet)

**v1.0 (Sprint 3–4)** - PDF export - Accounts & saved profiles - Admin content editor & feature flags - Observability baseline (metrics, logs)

**v1.1 (Sprint 5)** - Chaldean ruleset - Compatibility report - Hindi localization

**v1.2 (Sprint 6)** - Pinnacles/Challenges - Personal year/month/day calculator - Name variant suggestions (transliteration helper)

---

# 14) Acceptance Criteria (samples)

- *Life Path Calculation:* Given `DOB=1995-12-31`, Pythagorean ruleset `R1`, API returns `lifePath.value=4`, with `steps=[1+9+9+5+1+2+3+1=31 → 3+1=4]`.

- *PDF Export:* For any calculation, `GET /report` returns a valid PDF (A4), size < 2MB, generated < 2s P95.
- *Guest Mode:* If user is not authenticated, no PII is persisted; transient calculation stored in memory < 15 minutes.

---

# 15) Risks & Mitigations

- **Ambiguity in rules:** Use versioned rulesets and golden tests; expose steps in response.
- **PII risk:** Guest mode + minimal storage + encryption at rest + deletion tooling.
- **Content quality:** Admin workflows, review approvals, A/B flagging.
- **Scaling PDF generation:** Queue + autoscaled workers; cache previously generated reports.

---

# 16) SDS – Detailed Module Design

**Module:** `nas-calc-core` **(Java library)** - `NameNormalizer` → normalize Unicode, strip accents (configurable), tokenize. - `LetterMapper` → mapping strategy (Pythagorean/Chaldean) from `ruleset_id`. - `Reducer` → digit reduction with master-number guard. - `Calculators` → `LifePathCalculator`, `ExpressionCalculator`, `SoulUrgeCalculator`, etc. - `AuditTrail` → captures intermediate sums/decisions. - 100% pure functions; no I/O.

**Module:** `nas-api` - Spring REST controllers, DTOs with Bean Validation, exception mappers. - Service layer orchestrates calc + content + report. - Persistence via JPA repositories.

**Module:** `nas-content` - CRUD for interpretations, rulesets; simple WYSIWYG editor; versioning.

**Module:** `nas-report` - Server-side HTML templates (Thymeleaf/Handlebars) → Headless Chrome → PDF.

**Frontend** - React TS with routes: `/` (form), `/result/:id`, `/profiles`, `/admin`. - React Query for API; i18next for translations; Tailwind components.

---

# 17) Data Protection & Ethics

- Prominent disclaimer; avoid deterministic predictions about health/finance.
- Age gate (13+); parental consent banner if required by locale.
- Allow full data deletion and report anonymization.

---

# 18) Artifacts to Deliver

- SRS (this doc sections 3–8, 10–11)
- SDS (sections 9, 12, 16)
- API Spec (OpenAPI/Swagger YAML)

- Test Plan (section 11 expanded)
- Architecture Diagram (C4 level 1–3)
- ER Diagram & Sequence Diagrams
- CI/CD pipeline YAML & IaC (Terraform) for cloud

---

# 19) Next Steps (for your team)

1. Initialize monorepo (`apps/frontend`, `services/api`, `libs/calc-core`).
2. Define `ruleset_id=R1` (Pythagorean) with JSON mapping & sample goldens.
3. Implement `LifePath`, `Expression`, `SoulUrge`, `Personality` with tests.
4. Build MVP UI and `/calc/profile` endpoint.
5. Add PDF export.
6. Prepare SRS/SDS PDFs from this blueprint (can be auto-generated).

---

# 20) Sample OpenAPI (excerpt)

```
openapi: 3.0.3
info:
  title: Numerology Analysis API
  version: 1.0.0
paths:
  /api/v1/calc/profile:
    post:
      summary: Calculate numerology profile
      requestBody:
        required: true
        content:
          application/json:
            schema:
              type: object
              properties:
                fullName: { type: string }
                dob: { type: string, format: date }
                system: { type: string, enum: [PYTHAGOREAN, CHALDEAN] }
      responses:
        '200':
          description: OK
          content:
            application/json:
              schema:
                type: object
                properties:
                  calcId: { type: string }
                  rulesetId: { type: string }
```

```
                results: { type: object }
                steps: { type: array, items: { type: string } }
```