Software Requirement Specification

Sistem Pelaporan Prestasi Mahasiswa

Versi : 1.0
Tanggal : 12 November 2025

Pendahuluan

1.1 Tujuan Dokumen

Dokumen ini menjelaskan requirement untuk pengembangan aplikasi back-end sistem pelaporan prestasi mahasiswa yang memungkinkan mahasiswa melaporkan prestasi, dosen wali memverifikasi, dan admin mengelola sistem secara keseluruhan.

1.2 Ruang Lingkup

Sistem ini mencakup:

- Manajemen pengguna dengan Role-Based Access Control (RBAC)
- Pelaporan prestasi mahasiswa dengan field dinamis
- Verifikasi prestasi oleh dosen wali
- Dashboard dan statistik prestasi

1.3 Definisi dan Akronim

- **RBAC**: Role-Based Access Control
- **API**: Application Programming Interface
- **JWT**: JSON Web Token
- **REST**: Representational State Transfer

2. Deskripsi Umum Sistem

2.1 Perspektif Produk

Sistem ini adalah aplikasi back-end berbasis REST API yang menyediakan layanan untuk mengelola pelaporan prestasi mahasiswa dengan dukungan multi-role dan field prestasi yang fleksibel.

2.2 Fungsi Produk

- Autentikasi dan otorisasi berbasis role
- CRUD prestasi mahasiswa
- Workflow approval prestasi
- Manajemen pengguna dan role
- Pelaporan dan analitik prestasi

2.3 Karakteristik Pengguna

Role	Deskripsi	Hak Akses
Admin	Pengelola sistem	Full access ke semua fitur
Mahasiswa	Pelapor prestasi	Create, read, update prestasi sendiri
Dosen Wali	Verifikator prestasi	Read, verify prestasi mahasiswa bimbingannya

3. Arsitektur Database

3.1 PostgreSQL (RBAC & Data Relasional)

3.1.1 Tabel users

```
users {
  id: UUID PRIMARY KEY
  username: VARCHAR(50) UNIQUE NOT NULL
  email: VARCHAR(100) UNIQUE NOT NULL
  password_hash: VARCHAR(255) NOT NULL
  full_name: VARCHAR(100) NOT NULL
  role_id: UUID FOREIGN KEY -> roles.id
  is_active: BOOLEAN DEFAULT true
  created_at: TIMESTAMP DEFAULT NOW()
  updated_at: TIMESTAMP DEFAULT NOW()
}
```

3.1.2 Tabel roles

```
roles {
  id: UUID PRIMARY KEY
  name: VARCHAR(50) UNIQUE NOT NULL
  description: TEXT
  created_at: TIMESTAMP DEFAULT NOW()
}
```

Data awal:

- Admin
- Mahasiswa
- Dosen Wali

3.1.3 Tabel permissions

```
permissions {
  id: UUID PRIMARY KEY
  name: VARCHAR(100) UNIQUE NOT NULL
  resource: VARCHAR(50) NOT NULL
  action: VARCHAR(50) NOT NULL
  description: TEXT
}
```

Contoh permissions:

- achievement:create
- achievement:read
- achievement:update
- achievement:delete

- achievement:verify
- user:manage

3.1.4 Tabel role_permissions

```
role_permissions {
  role_id: UUID FOREIGN KEY -> roles.id
  permission_id: UUID FOREIGN KEY -> permissions.id
  PRIMARY KEY (role_id, permission_id)
}
```

3.1.5 Tabel students

```
students {
  id: UUID PRIMARY KEY
  user_id: UUID FOREIGN KEY -> users.id
  student_id: VARCHAR(20) UNIQUE NOT NULL
  program_study: VARCHAR(100)
  academic_year: VARCHAR(10)
  advisor_id: UUID FOREIGN KEY -> lecturers.id
  created_at: TIMESTAMP DEFAULT NOW()
}
```

3.1.6 Tabel lecturers

```
lecturers {
  id: UUID PRIMARY KEY
  user_id: UUID FOREIGN KEY -> users.id
  lecturer_id: VARCHAR(20) UNIQUE NOT NULL
  department: VARCHAR(100)
  created_at: TIMESTAMP DEFAULT NOW()
}
```

3.1.7 Tabel achievement_references

```
achievement_references {
  id: UUID PRIMARY KEY
  student_id: UUID FOREIGN KEY -> students.id
  mongo_achievement_id: VARCHAR(24) NOT NULL
  status: ENUM('draft', 'submitted', 'verified', 'rejected')
  submitted_at: TIMESTAMP
  verified_at: TIMESTAMP
  verified_by: UUID FOREIGN KEY -> users.id
  rejection_note: TEXT
  created_at: TIMESTAMP DEFAULT NOW()
  updated_at: TIMESTAMP DEFAULT NOW()
}
```

3.2 MongoDB (Data Prestasi Dinamis)

3.2.1 Collection achievements

```
{
  _id: ObjectId,
  studentId: UUID (reference to PostgreSQL),
  achievementType: String, // 'academic', 'competition', 'organization',
'publication', 'certification', 'other'
  title: String,
  description: String,

  // Field dinamis berdasarkan tipe prestasi
  details: {
    // Untuk competition
    competitionName?: String,
    competitionLevel?: String, // 'international', 'national', 'regional',
'local'
    rank?: Number,
    medalType?: String,

    // Untuk publication
    publicationType?: String, // 'journal', 'conference', 'book'
    publicationTitle?: String,
    authors?: [String],
    publisher?: String,
    issn?: String,

    // Untuk organization
    organizationName?: String,
    position?: String,
    period?: {
      start: Date,
      end: Date
    },

    // Untuk certification
    certificationName?: String,
    issuedBy?: String,
    certificationNumber?: String,
    validUntil?: Date,

    // Field umum yang bisa ada
    eventDate?: Date,
    location?: String,
    organizer?: String,
    score?: Number,
```

```
    customFields?: Object // untuk field tambahan yang tidak terdefinisi
  },

  attachments: [{
    fileName: String,
    fileUrl: String,
    fileType: String,
    uploadedAt: Date
  }],

  tags: [String],
  points: Number, // poin prestasi untuk keperluan scoring

  createdAt: Date,
  updatedAt: Date
}
```

4. Functional Requirements

4.1 Autentikasi & Otorisasi

FR-001: Login

Deskripsi: Pengguna dapat login menggunakan username/email dan password
Actor: Semua role
Flow:

1. User mengirim kredensial
2. Sistem memvalidasi kredensial
3. Sistem mengecek status aktif user
4. Sistem generate JWT token dengan role dan permissions
5. Return token dan user profile

FR-002: RBAC Middleware

Deskripsi: Setiap endpoint dilindungi dengan permission check
Actor: Sistem
Flow:

1. Ekstrak JWT dari header
2. Validasi token
3. Load user permissions dari cache/database
4. Check apakah user memiliki permission yang diperlukan
5. Allow/deny request

4.2 Manajemen Prestasi (Mahasiswa)

FR-003: Submit Prestasi

Deskripsi: Mahasiswa dapat menambahkan laporan prestasi
Actor: Mahasiswa
Precondition: User terautentikasi sebagai mahasiswa
Flow:

1. Mahasiswa mengisi data prestasi
2. Mahasiswa upload dokumen pendukung
3. Sistem simpan ke MongoDB (achievement) dan PostgreSQL (reference)
4. Status awal: 'draft'
5. Return achievement data

FR-004: Submit untuk Verifikasi

Deskripsi: Mahasiswa submit prestasi draft untuk diverifikasi
Actor: Mahasiswa
Precondition: Prestasi berstatus 'draft'
Flow:

1. Mahasiswa submit prestasi

2. Update status menjadi 'submitted'
3. Create notification untuk dosen wali
4. Return updated status

FR-005: Hapus Prestasi

Deskripsi: Mahasiswa dapat menghapus prestasi draft
Actor: Mahasiswa
Precondition: Status 'draft'
Flow:

1. Soft delete data di MongoDB
2. Update reference di PostgreSQL
3. Return success message

4.3 Verifikasi Prestasi (Dosen Wali)

FR-006: View Prestasi Mahasiswa Bimbingan

Deskripsi: Dosen wali melihat daftar prestasi mahasiswa bimbingannya
Actor: Dosen Wali
Flow:

1. Get list student IDs dari tabel students where advisor_id
2. Get achievements references dengan filter student_ids
3. Fetch detail dari MongoDB
4. Return list dengan pagination

FR-007: Verify Prestasi

Deskripsi: Dosen wali memverifikasi prestasi mahasiswa
Actor: Dosen Wali
Precondition: Status 'submitted'
Flow:

1. Dosen review prestasi detail
2. Dosen approve prestasi
3. Update status menjadi 'verified'
4. Set verified_by dan verified_at
5. Return updated status

FR-008: Reject Prestasi

Deskripsi: Dosen wali menolak prestasi dengan catatan
Actor: Dosen Wali
Precondition: Status 'submitted'
Flow:

1. Dosen input rejection note

2. Update status menjadi 'rejected'
3. Save rejection_note
4. Create notification untuk mahasiswa
5. Return updated status

4.4 Manajemen Sistem (Admin)

FR-009: Manage Users

Deskripsi: Admin dapat CRUD users dan assign roles
Actor: Admin
Flow:

1. Create/update/delete user
2. Assign role
3. Set student/lecturer profile
4. Set advisor untuk mahasiswa

FR-010: View All Achievements

Deskripsi: Admin dapat melihat semua prestasi
Actor: Admin
Flow:

1. Get all achievement references
2. Fetch details dari MongoDB
3. Apply filters dan sorting
4. Return dengan pagination

4.5 Reporting & Analytics

FR-011: Achievement Statistics

Deskripsi: Generate statistik prestasi
Actor: Mahasiswa (own), Dosen Wali (advisee), Admin (all)
Output:

- Total prestasi per tipe
- Total prestasi per periode
- Top mahasiswa berprestasi
- Distribusi tingkat kompetisi

5. API Endpoints

5.1 Authentication

POST /api/v1/auth/login
POST /api/v1/auth/refresh
POST /api/v1/auth/logout
GET /api/v1/auth/profile

5.2 Users (Admin)

GET /api/v1/users
GET /api/v1/users/:id
POST /api/v1/users
PUT /api/v1/users/:id
DELETE /api/v1/users/:id
PUT /api/v1/users/:id/role

5.4 Achievements

GET /api/v1/achievements // List (filtered by role)
GET /api/v1/achievements/:id // Detail
POST /api/v1/achievements // Create (Mahasiswa)
PUT /api/v1/achievements/:id // Update (Mahasiswa)
DELETE /api/v1/achievements/:id // Delete (Mahasiswa)
POST /api/v1/achievements/:id/submit // Submit for verification
POST /api/v1/achievements/:id/verify // Verify (Dosen Wali)
POST /api/v1/achievements/:id/reject // Reject (Dosen Wali)
GET /api/v1/achievements/:id/history // Status history
POST /api/v1/achievements/:id/attachments // Upload files

5.5 Students & Lecturers

GET /api/v1/students
GET /api/v1/students/:id
GET /api/v1/students/:id/achievements
PUT /api/v1/students/:id/advisor
GET /api/v1/lecturers
GET /api/v1/lecturers/:id/advisees

5.8 Reports & Analytics

GET /api/v1/reports/statistics
GET /api/v1/reports/student/:id

6. Testing Strategy

6.1 Unit Testing

- Test individual functions dan methods
- Mock external dependencies

7 Additional

7.1 Swagger untuk API documentation
7.2 Github Repository

Appendix

Sample Request/Response

1. Login Request

```
POST /api/v1/auth/login
{
  "username": "mahasiswa123",
  "password": "SecurePass123!"
}
```

2. Login Response

```
{
  "status": "success",
  "data": {
    "token": "eyJhbGciOiJIUzI1NiIs...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIs...",
    "user": {
      "id": "uuid-here",
      "username": "mahasiswa123",
      "fullName": "John Doe",
      "role": "Mahasiswa",
      "permissions": ["achievement:create", "achievement:read",
"achievement:update"]
    }
  }
}
```

3. Create Achievement Request

```
POST /api/v1/achievements
Authorization: Bearer {token}

{
  "achievementType": "competition",
  "title": "Juara 1 Hackathon Nasional 2025",
  "description": "Memenangkan kompetisi hackathon tingkat nasional",
  "details": {
    "competitionName": "Indonesia Tech Innovation Challenge",
    "competitionLevel": "national",
    "rank": 1,
    "medalType": "gold",
    "eventDate": "2025-10-15",
    "location": "Jakarta",
    "organizer": "Kementerian Pendidikan"
  },
  "tags": ["teknologi", "hackathon", "programming"]
}
```

4. Error Codes

Code	**Message**	**Description**
400	Bad Request	Invalid input data
401	Unauthorized	Missing or invalid token
403	Forbidden	Insufficient permissions
404	Not Found	Resource not found
409	Conflict	Duplicate entry
422	Unprocessable Entity	Validation error
500	Internal Server Error	Server error