Software Requirements Specification (SRS)

1. Introduction

1.1 Purpose

Dokumen ini mendefinisikan persyaratan perangkat lunak untuk website Fakultas Ilmu Komputer. Tujuan dari website ini adalah untuk memberikan platform yang interaktif bagi mahasiswa dan staf fakultas untuk mengakses informasi penting, termasuk berita, agenda, proyek, jurnal, dan ruang diskusi.

1.2 Scope

Website ini akan memiliki dua peran pengguna: Admin dan Pengguna. Admin memiliki kemampuan untuk mengelola konten di website, sementara Pengguna dapat melihat dan berinteraksi dengan konten. Website ini mencakup fitur untuk berita fakultas, informasi acara, showcase proyek, jurnal atau paper, dan ruang diskusi.

1.3 Definitions, Acronyms, and Abbreviations

- **Admin**: Pengguna dengan hak akses untuk mengelola dan memodifikasi konten di website.
- **Pengguna**: Pengguna biasa yang dapat melihat dan berinteraksi dengan konten.
- **ERD**: Entity-Relationship Diagram.
- **API**: Application Programming Interface.

2. Functional Requirements

2.1 Home Page

- **Description**: Halaman utama yang menampilkan ringkasan berita terbaru, acara mendatang, dan fitur penting lainnya.
- **Features**:
    - Tampilkan berita terbaru.
    - Tampilkan acara mendatang.
    - Tautan ke halaman detail berita, agenda, proyek, jurnal, dan ruang diskusi.

2.2 Fasilkom News

- **Description**: Halaman yang menampilkan daftar berita fakultas.
- **Features**:
    - **Tambah Berita** (Admin): Admin dapat menambahkan berita baru.
    - **Edit Berita** (Admin): Admin dapat memperbarui berita yang ada.
    - **Hapus Berita** (Admin): Admin dapat menghapus berita yang tidak relevan.
    - **Tampilkan Berita** (Pengguna): Pengguna dapat melihat berita dengan ringkasan.

    - **Tampilan Detail Berita** (Pengguna): Pengguna dapat mengklik berita untuk melihat detail lengkap, termasuk judul, tanggal, gambar, dan konten lengkap.

2.3 Event Information (Agenda)

- **Description**: Halaman yang menampilkan daftar acara fakultas.
- **Features**:
    - **Tambah Acara** (Admin): Admin dapat menambahkan acara baru.
    - **Edit Acara** (Admin): Admin dapat memperbarui acara yang ada.
    - **Hapus Acara** (Admin): Admin dapat menghapus acara yang tidak relevan.
    - **Tampilkan Acara** (Pengguna): Pengguna dapat melihat daftar acara mendatang.
    - **Tampilan Detail Acara** (Pengguna): Pengguna dapat mengklik acara untuk melihat detail lengkap, termasuk nama acara, deskripsi, tanggal, waktu, lokasi, dan informasi tambahan.

2.4 Project Showcase

- **Description**: Halaman yang menampilkan proyek-proyek yang dikerjakan oleh mahasiswa.
- **Features**:
    - **Tambah Proyek** (Admin): Admin dapat menambahkan proyek baru.
    - **Edit Proyek** (Admin): Admin dapat memperbarui informasi proyek yang ada.
    - **Hapus Proyek** (Admin): Admin dapat menghapus proyek yang tidak relevan.
    - **Tampilkan Proyek** (Pengguna): Pengguna dapat melihat daftar proyek.
    - **Tampilan Detail Proyek** (Pengguna): Pengguna dapat mengklik proyek untuk melihat detail lengkap, termasuk judul proyek, deskripsi, gambar, dan nama mahasiswa.

2.5 Journal/Paper

- **Description**: Halaman yang menampilkan jurnal atau paper akademis dari fakultas.
- **Features**:
    - **Unggah Jurnal/Paper** (Admin): Admin dapat mengunggah jurnal atau paper baru.
    - **Edit Jurnal/Paper** (Admin): Admin dapat memperbarui informasi jurnal atau paper yang ada.
    - **Hapus Jurnal/Paper** (Admin): Admin dapat menghapus jurnal atau paper yang tidak relevan.
    - **Tampilkan Jurnal/Paper** (Pengguna): Pengguna dapat melihat daftar jurnal atau paper.
    - **Tampilan Detail Jurnal/Paper** (Pengguna): Pengguna dapat mengklik jurnal atau paper untuk melihat detail lengkap, termasuk judul, penulis, abstrak, dan link unduhan.

2.6 Discussion Room

- **Description**: Halaman yang menyediakan ruang untuk diskusi antar pengguna.
- **Features**:

    - **Buat Diskusi** (Pengguna): Pengguna dapat memulai diskusi baru dengan teks dan gambar.
    - **Balas Pesan** (Pengguna): Pengguna dapat membalas pesan dalam diskusi dengan teks dan gambar.
    - **Moderasi Diskusi** (Admin): Admin dapat mengelola dan memoderasi diskusi, termasuk menghapus pesan yang tidak sesuai.
    - **Tampilan Diskusi** (Pengguna): Pengguna dapat melihat diskusi dan balasan, serta gambar yang diunggah dalam diskusi.

3. Non-Functional Requirements

3.1 Performance

- Website harus cepat dalam memuat halaman dan merespons interaksi pengguna.
- Harus mampu menangani sejumlah pengguna secara bersamaan tanpa penurunan performa yang signifikan.

3.2 Security

- **Authentication**: Implementasi autentikasi pengguna dengan username dan password.
- **Authorization**: Kontrol akses berdasarkan peran (admin atau pengguna) untuk melindungi fitur dan data.
- **Data Protection**: Lindungi data pengguna dan informasi sensitif dengan enkripsi dan praktik keamanan lainnya.

3.3 Usability

- Antarmuka pengguna harus intuitif dan mudah digunakan, dengan navigasi yang jelas.
- Halaman harus responsif dan kompatibel dengan berbagai perangkat dan browser.

3.4 Reliability

- Sistem harus memiliki mekanisme untuk pemulihan dari kesalahan dan cadangan data.
- Harus mampu menangani kesalahan dengan cara yang tidak mempengaruhi pengalaman pengguna secara signifikan.

4. System Architecture

4.1 Overview

- **Frontend**: Menggunakan React, Bootstrap, HTML, CSS, dan JavaScript untuk membangun antarmuka pengguna yang interaktif.
- **Backend**: Menggunakan Node.js dengan Express untuk menangani logika server dan API.
- **Database**: Menggunakan MySQL untuk menyimpan data aplikasi.

4.2 Components

- **Frontend Components**: Halaman utama, daftar berita, halaman detail berita, halaman acara, halaman detail acara, halaman proyek, halaman detail proyek, halaman jurnal/paper, halaman detail jurnal/paper, dan ruang diskusi.
- **Backend Components**: API endpoints untuk mengelola berita, acara, proyek, jurnal/paper, dan diskusi.
- **Database Schema**: Tabel untuk pengguna, berita, acara, proyek, jurnal, dan diskusi, dengan relasi yang mendukung fitur-fitur aplikasi.

API Specification

1. Authentication

POST /api/login

- **Description**: Authenticates a user and returns a token.

Request Body:
json
Copy code

```
{
  "username": "user",
  "password": "pass"
}
```

Response:
json
Copy code

```
{
  "token": "jwt-token",
  "user": {
    "id": "user-id",
    "username": "user",
    "role": "admin" // or "user"
  }
}
```

POST /api/signup

- **Description**: Registers a new user.

Request Body:
json
Copy code

```
{
  "username": "newuser",
  "email": "user@example.com",
  "password": "pass"
}
```

Response:
json
Copy code

```
{
  "message": "Registration successful. Please check your email to
verify your account."
}
```

GET /api/verify-email

- **Description**: Verifies the user's email address.
- **Request Parameters**: `token`

Response:
json
Copy code

```
{
  "message": "Email verified successfully."
}
```

2. News

GET /api/news

- **Description**: Retrieves a list of news articles.

Response:
json
Copy code

```
[
  {
    "newsId": "1",
    "title": "Title",
    "summary": "Brief summary of the news",
    "date": "2024-08-20",
    "image": "image-url"
  },
  ...
]
```

GET /api/news/{newsId}

- **Description**: Retrieves detailed information about a specific news article.

Response:
json
Copy code

```
{
  "newsId": "1",
  "title": "Full Title",
  "content": "Full content of the news",
  "date": "2024-08-20",
  "image": "full-image-url"
}
```

POST /api/news

- **Description**: Creates a new news article (Admin only).

Request Body:
json
Copy code

```
{
  "title": "News Title",
  "content": "Full content of the news",
  "date": "2024-08-20",
  "image": "image-url" // Optional
}
```

Response:
json
Copy code

```
{
  "newsId": "new-news-id",
  "message": "News created successfully."
}
```

PUT /api/news/{newsId}

- **Description**: Updates an existing news article (Admin only).

Request Body:
json
Copy code

```
{
  "title": "Updated Title",
  "content": "Updated content",
  "date": "2024-08-20",
  "image": "new-image-url" // Optional
}
```

Response:
json
Copy code

```
{
  "message": "News updated successfully."
}
```

DELETE /api/news/{newsId}

- **Description**: Deletes a news article (Admin only).

Response:
json
Copy code

```
{
  "message": "News deleted successfully."
}
```

3. Agenda (Events)

GET /api/events

- **Description**: Retrieves a list of events.

Response:
json
Copy code

```
[
  {
    "eventId": "1",
    "title": "Event Title",
```

```
    "summary": "Brief summary of the event",
    "date": "2024-08-20",
    "time": "10:00 AM",
    "location": "Location",
    "image": "image-url"
  },
  ...
]
```

GET /api/events/{eventId}

- **Description**: Retrieves detailed information about a specific event.

Response:
json
Copy code

```
{
  "eventId": "1",
  "title": "Full Event Title",
  "description": "Full description of the event",
  "date": "2024-08-20",
  "time": "10:00 AM",
  "location": "Full location details",
  "image": "full-image-url"
}
```

POST /api/events

- **Description**: Creates a new event (Admin only).

Request Body:
json
Copy code

```
{
  "title": "Event Title",
  "description": "Full description of the event",
  "date": "2024-08-20",
  "time": "10:00 AM",
  "location": "Location",
  "image": "image-url" // Optional
}
```

Response:
json
Copy code

```
{
  "eventId": "new-event-id",
  "message": "Event created successfully."
}
```

PUT /api/events/{eventId}

- **Description**: Updates an existing event (Admin only).

Request Body:
json
Copy code

```
{
  "title": "Updated Title",
  "description": "Updated description",
  "date": "2024-08-20",
  "time": "10:00 AM",
  "location": "Updated location",
  "image": "new-image-url" // Optional
}
```

Response:
json
Copy code

```
{
  "message": "Event updated successfully."
}
```

DELETE /api/events/{eventId}

- **Description**: Deletes an event (Admin only).

Response:
json
Copy code

```
{
  "message": "Event deleted successfully."
}
```

4. Project Showcase

GET /api/projects

- **Description**: Retrieves a list of projects.

Response:
json
Copy code

```
[
  {
    "projectId": "1",
    "title": "Project Title",
    "summary": "Brief summary of the project",
    "images": ["image-url1", "image-url2"],
    "studentNames": ["Student Name"],
    "supervisorId": "supervisor-id"
  },
  ...
]
```

GET /api/projects/{projectId}

- **Description**: Retrieves detailed information about a specific project.

Response:
json
Copy code

```
{
  "projectId": "1",
  "title": "Full Project Title",
  "description": "Full description of the project",
  "images": ["full-image-url1", "full-image-url2"],
  "studentNames": ["Full Student Names"],
  "supervisorId": "full-supervisor-id"
}
```

POST /api/projects

- **Description**: Creates a new project (Admin only).

Request Body:
json
Copy code

```
{
  "title": "Project Title",
  "description": "Full description of the project",
  "images": ["image-url1", "image-url2"], // Optional
  "studentNames": ["Student Name"],
  "supervisorId": "supervisor-id"
}
```

Response:
json
Copy code

```
{
  "projectId": "new-project-id",
  "message": "Project created successfully."
}
```

PUT /api/projects/{projectId}

- **Description**: Updates an existing project (Admin only).

Request Body:
json
Copy code

```
{
  "title": "Updated Title",
  "description": "Updated description",
  "images": ["new-image-url1", "new-image-url2"], // Optional
  "studentNames": ["Updated Student Names"],
  "supervisorId": "updated-supervisor-id"
}
```

Response:
json
Copy code

```
{
  "message": "Project updated successfully."
}
```

DELETE /api/projects/{projectId}

- **Description**: Deletes a project (Admin only).

Response:
json
Copy code

```
{
  "message": "Project deleted successfully."
}
```

5. Journal/Paper

GET /api/journals

- **Description**: Retrieves a list of journals or papers.

Response:
json
Copy code

```
[
  {
    "journalId": "1",
    "title": "Title of Journal/Paper",
    "authors": "Author Names",
    "abstract": "Abstract of the journal/paper",
    "downloadLink": "download-url",
    "uploadDate": "2024-08-20"
  },
  ...
]
```

GET /api/journals/{journalId}

- **Description**: Retrieves detailed information about a specific journal or paper.

Response:
json
Copy code

```
{
  "journalId": "1",
  "title": "Title of Journal/Paper",
```

```
  "authors": "Author Names",
  "abstract": "Full abstract",
  "downloadLink": "download-url",
  "uploadDate": "2024-08-20"
}
```

POST /api/journals

- **Description**: Uploads a new journal or paper.

Request Body:
json
Copy code

```
{
  "title": "Title of Journal/Paper",
  "authors": "Author Names",
  "abstract": "Abstract of the journal/paper",
  "downloadLink": "upload-url"
}
```

Response:
json
Copy code

```
{
  "message": "Journal/Paper uploaded successfully.",
  "journalId": "new-journal-id"
}
```

PUT /api/journals/{journalId}

- **Description**: Updates an existing journal or paper.

Request Body:
json
Copy code

```
{
  "title": "Updated Title",
  "authors": "Updated Author Names",
  "abstract": "Updated abstract",
  "downloadLink": "updated-url"
```

```
}
```

Response:
json
Copy code
```
{
  "message": "Journal/Paper updated successfully."
}
```

DELETE /api/journals/{journalId}

- **Description**: Deletes a specific journal or paper.

Response:
json
Copy code
```
{
  "message": "Journal/Paper deleted successfully."
}
```

6. Discussion Room

GET /api/discussions

- **Description**: Retrieves a list of discussions.

Response:
json
Copy code
```
[
  {
    "discussionId": "1",
    "title": "Discussion Title",
    "summary": "Brief summary of the discussion",
    "createdDate": "2024-08-20"
  },
  ...
]
```

GET /api/discussions/{discussionId}

- **Description**: Retrieves detailed information about a specific discussion.

Response:
json
Copy code

```
{
  "discussionId": "1",
  "title": "Full Discussion Title",
  "content": "Full content of the discussion",
  "createdDate": "2024-08-20",
  "comments": [
    {
      "commentId": "1",
      "userId": "user-id",
      "content": "Comment content",
      "createdDate": "2024-08-20",
      "image": "comment-image-url" // Optional
    },
    ...
  ]
}
```

POST /api/discussions

- **Description**: Creates a new discussion.

Request Body:
json
Copy code

```
{
  "title": "Discussion Title",
  "content": "Content of the discussion",
  "image": "base64-encoded-image-or-file-url" // Optional
}
```

Response:
json
Copy code

```
{
  "message": "Discussion created successfully.",
```

```
  "discussionId": "new-discussion-id"
}
```

POST /api/discussions/{discussionId}/comments

- **Description**: Adds a comment to a specific discussion.

Request Body:
json
Copy code
```
{
  "content": "Content of the comment",
  "image": "base64-encoded-image-or-file-url" // Optional
}
```

-

Response:
json
Copy code
```
{
  "message": "Comment added successfully.",
  "commentId": "new-comment-id"
}
```

PUT /api/discussions/{discussionId}

- **Description**: Updates an existing discussion.

Request Body:
json
Copy code
```
{
  "title": "Updated Title",
  "content": "Updated content",
  "image": "updated-base64-encoded-image-or-file-url" // Optional
}
```
Response:
json
Copy code
```
{
  "message": "Discussion updated successfully."
}
```

DELETE /api/discussions/{discussionId}

- **Description**: Deletes a specific discussion.

Response:
json
Copy code

```
{
  "message": "Discussion deleted successfully."
}
```

DELETE /api/discussions/{discussionId}/comments/{commentId}

- **Description**: Deletes a specific comment from a discussion.

Response:
json
Copy code

```
{
  "message": "Comment deleted successfully."
}
```