TUT Emalahleni Campus Event Management System Documentation

Project Development Team

August 2025

Contents

1 Project Overview **2**
1.1 Problem Statement 2
1.2 Solution Overview 2
1.3 Target Users 2

2 Technical Stack and Tools **2**
2.1 Frontend 2
2.2 Backend 2
2.3 Database 2
2.4 Additional Tools 2

3 Project Setup Instructions **3**
3.1 Prerequisites 3
3.2 Backend Setup 3
3.3 Frontend Setup 4
3.4 Docker Setup 4
3.5 Database Schema 4

4 Core Functionalities **4**
4.1 Event Organizer Features 4
4.2 Student Features 5
4.3 System Features 5

5 System Requirements **5**
5.1 Technical Specifications 5
5.2 Compliance and Constraints 5

6 Success Metrics **5**

7 Expected Outcomes **6**

8 Maintenance and Support **6**

1 Project Overview

1.1 Problem Statement

Event organizers at Tshwane University of Technology (Emalahleni Campus) currently rely on fragmented communication methods such as emails, WhatsApp groups, and Google Forms. This leads to inefficiencies, poor attendance tracking, and missed opportunities for student engagement with campus events.

1.2 Solution Overview

The TUT Emalahleni Campus Event Management System is a centralized web-based platform designed to streamline event promotion, student notifications, and registration processes. It integrates with the WhatsApp Business API for automated messaging and provides a user-friendly interface for both organizers and students.

1.3 Target Users

- **Primary:** Event organizers (staff, student organizations, clubs)
- **Secondary:** Students (event attendees)
- **Tertiary:** Campus administrators (for oversight and reporting)

2 Technical Stack and Tools

2.1 Frontend

- **React:** JavaScript library for building a responsive and interactive user interface.
- **Tailwind CSS:** Utility-first CSS framework for styling the frontend.
- **Axios:** HTTP client for API requests to the backend.

2.2 Backend

- **Python (FastAPI):** High-performance web framework for building RESTful APIs.
- **PyWhatsApp:** Library for integrating with the WhatsApp Business API.
- **PySMS:** Library for SMS backup notifications.

2.3 Database

- **PostgreSQL:** Relational database for storing user, event, and registration data.
- **SQLAlchemy:** ORM for database interactions in Python.

2.4 Additional Tools

- **Docker:** Containerization for consistent development and deployment.
- **Git:** Version control for collaborative development.

- **Nginx:** Web server for reverse proxy and load balancing.
- **JWT (JSON Web Tokens):** For secure user authentication.
- **qrcode:** Python library for generating QR codes for event check-ins.

3 Project Setup Instructions

3.1 Prerequisites

- **Node.js** (v18 or higher)
- **Python** (v3.10 or higher)
- **PostgreSQL** (v15 or higher)
- **Docker** and **Docker Compose** (for containerized setup)
- **Git** (for version control)
- WhatsApp Business API credentials
- SMS service provider credentials (e.g., Twilio)

3.2 Backend Setup

1. Clone the repository:

   ```
   git clone https://github.com/tut-emalahleni/event-management-system.git
   cd event-management-system/backend
   ```

2. Create a virtual environment and install dependencies:

   ```
   python -m venv venv
   source venv/bin/activate  % On Windows: venv\Scripts\activate
   pip install -r requirements.txt
   ```

3. Configure environment variables in `.env`:

   ```
   DATABASE_URL=postgresql://user:password@localhost:5432/tut_events
   WHATSAPP_API_KEY=your_whatsapp_api_key
   SMS_API_KEY=your_sms_api_key
   JWT_SECRET=your_jwt_secret
   ```

4. Initialize the PostgreSQL database:

   ```
   psql -U postgres -c "CREATE DATABASE tut_events;"
   alembic upgrade head
   ```

5. Start the FastAPI server:

   ```
   uvicorn main:app --host 0.0.0.0 --port 8000
   ```

3.3 Frontend Setup

1. Navigate to the frontend directory:

```
cd ../frontend
```

2. Install dependencies:

```
npm install
```

3. Configure environment variables in `.env`:

```
REACT_APP_API_URL=http://localhost:8000
```

4. Start the React development server:

```
npm start
```

3.4 Docker Setup

1. Ensure Docker and Docker Compose are installed.
2. Build and run the application:

```
docker-compose up --build
```

3. Access the application at `http://localhost:3000`.

3.5 Database Schema

The PostgreSQL database includes the following main tables:

- **users:** Stores user information (student number, staff ID, role).
- **events:** Stores event details (title, date, venue, capacity, etc.).
- **registrations:** Tracks student registrations and waitlists.
- **notifications:** Logs WhatsApp and SMS notifications.

4 Core Functionalities

4.1 Event Organizer Features

- Create and edit event listings with details (date, time, venue, capacity, description, images).
- Set registration deadlines and capacity limits.
- View attendee lists and registration analytics (e.g., attendance rates).
- Send bulk WhatsApp notifications using the WhatsApp Business API.
- Generate QR codes for event check-ins using the `qrcode` library.

4.2 Student Features

- Browse events by category, date, or organizer.
- Receive automated WhatsApp notifications for relevant events.
- Register for events with confirmation emails and QR codes.
- View personal event calendar and registration history.
- Cancel registrations before the event deadline.

4.3 System Features

- **WhatsApp API Integration:** Sends automated notifications and reminders.
- **Authentication:** JWT-based login tied to university credentials.
- **Event Categorization:** Supports academic, social, sports, and career events.
- **Capacity Management:** Handles registration limits and waitlists.
- **Mobile-Responsive Design:** Built with Tailwind CSS for accessibility.

5 System Requirements

5.1 Technical Specifications

- Web-based platform accessible via modern browsers (Chrome, Firefox, Safari).
- Integration with WhatsApp Business API for notifications.
- PostgreSQL database for secure data storage.
- JWT-based secure login system integrated with university credentials.
- SMS backup option using a service like Twilio.

5.2 Compliance and Constraints

- Adheres to university IT policies and data protection regulations (e.g., POPIA).
- Budget-friendly deployment using open-source tools and affordable APIs.
- User-friendly interface for non-technical organizers.
- Optional integration with existing university systems via APIs.

6 Success Metrics

- Increase in event attendance rates by 20% within six months.
- Reduction in no-show percentages by 15%.
- 80% of students report improved event discovery.
- Registration process completed in under 2 minutes per user.
- Comprehensive data collection for engagement analysis.

7 Expected Outcomes

- Centralized platform reducing administrative overhead by 30%.
- Improved student engagement with a 25% increase in event participation.
- Reliable communication channel via WhatsApp and SMS.
- Data-driven insights for optimizing campus event planning.

8 Maintenance and Support

- Regular database backups using PostgreSQL tools.
- Monthly updates to React and Python dependencies.
- Monitoring API usage (WhatsApp, SMS) to ensure cost efficiency.
- Support ticketing system for organizers and students.