# Presentation Document for Event Management System Project

## Introduction

- **Project Name**: Event Management System
- **Purpose**: To streamline the process of creating, editing, and managing events along with RSVP functionalities.
- **Technologies Used**: Django, Tailwind CSS, Docker, GitHub Actions

## Project Overview

1. **Event Creation and Management**:
    - Users can create, edit, and delete events.
    - Event details include title, description, date, and category.
    - Tailwind CSS is used for a modern, responsive design.
2. **RSVP System**:
    - Users can RSVP to events with statuses: accepted, declined, tentative.
    - RSVP status is stored and displayed on the event detail page.

## Key Features

1. **User Authentication**:
    - Registration and login functionality.
    - Secure access to event management features.
2. **Event Management**:
    - Simple and intuitive forms for event creation and editing.
    - Events can be viewed in detail with options to edit or delete.
3. **RSVP Management**:
    - Users can respond to events with their RSVP status.
    - Current RSVP status is displayed on the event detail page.
4. **Modern UI Design**:
    - Tailwind CSS used for a clean, modern, and responsive interface.
    - Consistent color schemes and responsive layout for a better user experience.

## Video Explanation Script

1. **Introduction (0:00 - 1:00)**
    - Briefly introduce yourself and the project.
    - Explain the purpose and goals of the Event Management System.
2. **User Authentication (1:00 - 2:00)**
    - Demonstrate user registration and login.
    - Show how authenticated users can access event management features.
3. **Event Management (2:00 - 3:00)**
    - Walk through the process of creating a new event.
    - Edit an existing event and delete an event.
4. **RSVP System (3:00 - 4:00)**

    - Show how users can RSVP to events.
    - Display current RSVP status on the event detail page.
5. **Modern UI Design (4:00 - 5:00)**
    - Highlight the use of Tailwind CSS for a modern look.
    - Showcase responsive design elements.

## GitHub Actions for Docker Deployment and Hosting

### GitHub Actions Workflow for Docker and Hosting on GCP (Cheapest Option)

1. **Set Up Your GCP Project and Container Registry**:
    - Enable the Container Registry API in your GCP project.
    - Create a service account with permissions to push images to the Container Registry.
    - Download the service account key JSON file.
2. **GitHub Secrets**:
    - Add the following secrets to your GitHub repository:
        - `GCP_PROJECT_ID`: Your GCP project ID.
        - `GCP_SA_KEY`: The JSON key file for the service account.
3. **GitHub Actions Workflow File** (`.github/workflows/docker-gcp.yml`):

```
name: Docker Image CI

on:
  push:
    branches:
      - main
    tags:
      - 'v*.*.*'

jobs:
  build-and-deploy:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v4

    - name: Set up Docker Buildx
      uses: docker/setup-buildx-action@v3

    - name: Authenticate to Google Cloud
      uses: google-github-actions/auth@v1
      with:
        credentials_json: ${{ secrets.GCP_SA_KEY }}

    - name: Set up Google Cloud SDK
      uses: google-github-actions/setup-gcloud@v1
      with:
        project_id: ${{ secrets.GCP_PROJECT_ID }}
        install_components: 'gcloud'

    - name: Configure Docker to use gcloud as a credential helper
```

```
      run: |
        gcloud auth configure-docker

    - name: Extract Docker metadata
      id: meta
      uses: docker/metadata-action@v5
      with:
        images: gcr.io/${{ secrets.GCP_PROJECT_ID }}/${{
github.repository }}

    - name: Build and push Docker image
      uses: docker/build-push-action@v5
      with:
        context: .
        push: true
        tags: ${{ steps.meta.outputs.tags }}

    - name: Deploy to Cloud Run
      run: |
        gcloud run deploy event-management-system \
          --image gcr.io/${{ secrets.GCP_PROJECT_ID }}/${{
github.repository }}:${{ steps.meta.outputs.tags }} \
          --platform managed \
          --region us-central1 \
          --allow-unauthenticated
```

## Steps in the Workflow

1. **Checkout Code**: Retrieves the latest code from the repository.
2. **Set Up Docker Buildx**: Enables multi-platform builds.
3. **Authenticate to Google Cloud**: Uses the service account key to authenticate.
4. **Set Up Google Cloud SDK**: Installs the necessary components for gcloud.
5. **Configure Docker**: Sets up Docker to use gcloud for authentication.
6. **Extract Docker Metadata**: Extracts metadata for tagging the Docker image.
7. **Build and Push Docker Image**: Builds the Docker image and pushes it to the Google Container Registry.
8. **Deploy to Cloud Run**: Deploys the Docker image to Google Cloud Run.

## Conclusion

- **Wrap Up**: Recap the main features and benefits of the Event Management System.
- **Future Enhancements**: Mention any planned future improvements or features.
- **Q&A**: Offer to answer any questions if this is part of a live presentation.