Software Design Document (SDD)

1. Introduction

1.1 Purpose

The purpose of this system is to create a command-line application where CS, CIS, and BINF majors, minors, and faculty can create, edit, remove, list, search, RSVP to, and receive alerts for events based on their profile.

1.2 Scope

This application provides event management and RSVP functionality through a terminal-based interface. Users can manage events, filter through available options, and receive notifications.

1.3 Definitions, Acronyms, and Abbreviations

- **User**: Represents a student or faculty user.
- **Event**: A scheduled item created by users that others can RSVP to.
- **RSVP**: User response indicating participation in an event.

1.4 Overview Overview

This document outlines the system architecture, components, workflows, interface design, data design, and testing considerations.

2. System Overview

The system is a CLI-based Python application that allows users to manage events and RSVPs. Users interact through text prompts in the terminal to create and manage events.

3. System Architecture

3.1 Architectural Design

A Python-based command-line system that uses object-oriented design. Interaction occurs entirely via user input entered in the terminal.

3.2 Subsystem Decomposition

- **Event Management**: Create, edit, remove, view, list, and search events.
- **RSVP System**: Allow users to RSVP, update RSVPs, and view RSVP status.
- **Notification System**: Provide event alerts or updates.
- **Filter/Search System**: Users can filter and browse events.

3.3 Design Constraints

- CLI-based interaction only; no GUI or web features.
- Python standard library preferred unless otherwise needed.

4. Detailed Design

4.1 Component Descriptions

- **Event Class**: Contains event title, date, time, description, creator, and list of RSVPs.
- **EventManager Class**: Handles creation, editing, deletion, listing, and searching of events.
- **RSVPManager Class**: Manages RSVP actions and tracks attendance.
- **NotificationManager Class**: Sends alerts or reminders for upcoming events.

4.2 Database Design

Initially, data will be stored in in-memory structures, with possible extension to file storage or simple databases.

4.3 Interface Design

Users interact via text prompts to:

- Create and edit events
- View and search events
- RSVP to events
- Filter events based on date or category
- Receive printed terminal notifications

4.4 Data Flow

User input → processed by managers/classes → event or RSVP updates → output displayed in terminal.

5. Testing Considerations. Testing Considerations

5.1 Testing Strategy

- **Unit Testing**: Test each class (User, Event, EventManager, RSVPManager, NotificationManager) individually.
- **Integration Testing**: Ensure flow between event creation → RSVP → notifications works correctly.
- **User Acceptance Testing (UAT)**: Validate that students and faculty can perform core tasks via the CLI.

5.2 Sample Unit Tests

- Creating a profile
- Adding, editing, and removing an event
- Listing and searching events
- RSVP flow (RSVP, update, remove)
- Notification trigger for upcoming events

6. Workflows

6.1 Event Lifecycle Workflow

1. User logs in or selects profile.
2. User selects **Create Event**.
3. User enters event details.
4. EventManager stores event.
5. Event appears in event list.
6. Other users may RSVP.
7. Notifications are sent before event start time.

6.2 RSVP Workflow

1. User views list of events.
2. User selects event to RSVP.
3. RSVP class records RSVP.
4. User may update or cancel RSVP.

6.3 Notification Workflow

1. System checks upcoming events.
2. If event is within predefined alert time → send notification.
3. Notification printed in terminal.

7. UML/Class Diagram (Text Description)

Class Overview

- **User**: name, major/minor, role, preferences
- **Event**: title, description, date, time, creator, RSVPs
- **Events class**: event list, operations (add/edit/remove/search/list)
- **RSVP class**: manage RSVPs per event
- **NotificationManager**: check time-based alerts

Relationships

- User → can create many Events
- Event → contains many RSVPs (User references)
- Events class → manages all Event objects
- Alert class → depends on Event data

8. MVP and MMP Scope

MVP (Minimum Viable Product)

- Create, edit, delete events
- View and search events
- RSVP to events
- Basic notifications (print to terminal)
- Users with role + major/minor

MMP (Minimum Marketable Product)

- Advanced filtering (date ranges, categories)
- File-based data persistence
- Multiple notification types