Software Design Specifications for UniEvent

Prepared by: Team 14

Document Information

Title: Software Design Specifications for UniEvent
Document Version No: 1.1
Document Version Date: 09-04-25
Prepared By: Team 14

Version History

Ver. Date	Revised By	Description	Filename
03-04-25	Team 14	Initial draft	UniEvent_SDD_1.0.docx
10-04-25	Team 14	updated	UniEvent_SDD_1.1.docx

Table of Contents

1 Introduction

This Software Design Specification (SDD) outlines the architectural design and detailed component structure of UniEvent, a comprehensive web-based university event management system. The system is specifically designed to streamline the event approval process at Mahindra University while promoting student participation through simplified event discovery and gamified engagement tracking.

1.1 Purpose

The purpose of this document is to provide a detailed blueprint of the UniEvent application's technical architecture and implementation strategy. It serves multiple critical purposes:

- Provides technical guidance to developers by outlining system components, their interactions, and implementation requirements
- Establishes clear design patterns and architectural decisions to ensure consistency across development
- Serves as a reference for stakeholders to validate design choices and verify alignment with business goals
- Documents the system's structure, behavior, and data models for future maintenance and enhancement
- Creates a foundation for testing strategies by defining system boundaries and component interactions

This document builds upon the requirements established in the Software Requirements Specification (SRS) and translates those requirements into a technically viable solution.

1.2 Scope

UniEvent provides comprehensive functionality for:

- Role-based access control system: Distinct interfaces and permissions for Students, Club Heads, Approval Authorities, and Logistics teams
- Event proposal and multi-stage approval workflows: Automated routing of approvals with status tracking
- Venue booking and logistics integration: Calendar synchronization and resource allocation

- Notification engine: Multi-channel alerts (email, in-app, optional SMS)
- Analytics and reporting dashboard: Event metrics, participation statistics, and engagement reports
- Mobile-responsive interface: Accessible on various devices with optimized layouts

The system is designed to handle the complete lifecycle of university events, from conception through approval, execution, and post-event analysis.

1.3 Definitions, Acronyms, and Abbreviations

- UI/UX: User Interface / User Experience
- API: Application Programming Interface
- DB: Database
- RBAC: Role-Based Access Control
- SSO: Single Sign-On
- COMET: Concurrent Object Modeling and Architectural Design Method
- JWT: JSON Web Token
- REST: Representational State Transfer
- CRUD: Create, Read, Update, Delete
- DTO: Data Transfer Object
- MVC: Model-View-Controller
- QA: Quality Assurance

1.4 References

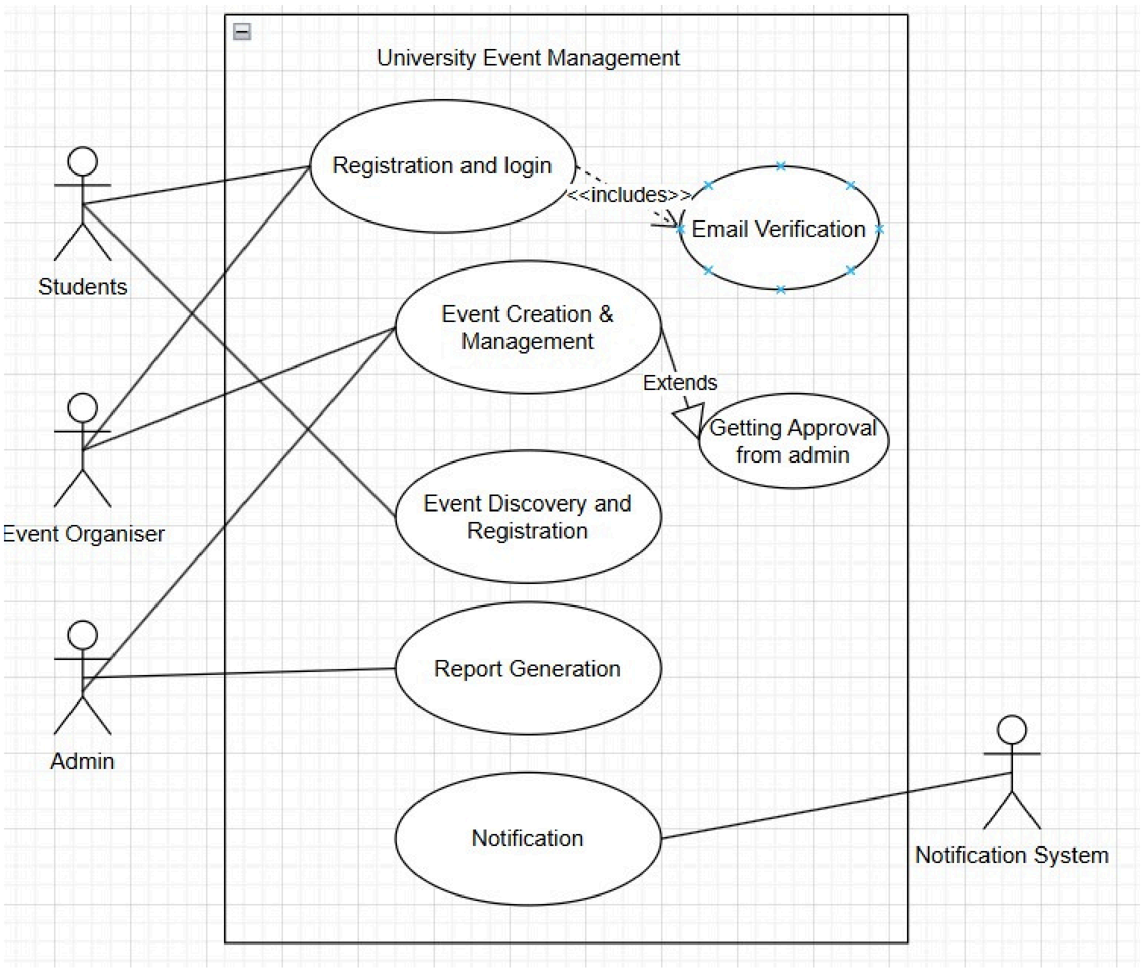
- Software Requirements Specification (SRS) for UniEvent, Version 1.0, dated 10-03-24
- Statement of Work (SOW) for UniEvent, dated 07-02-25
- Project Planning & Costing Document, Version 1.0
- Mahindra University Event Management Policies and Procedures
- COMET Design Methodology Reference Guide
- Node.js and Express.js Documentation
- MongoDB Documentation
- JWT Authentication Best Practices Guide

2 Use Case View

This section details the key use cases that drive the system design, highlighting interactions between actors and the system. The use cases represent significant, central functionality of UniEvent and demonstrate how various design elements collaborate to fulfill user requirements.

2.1 Use Case

The primary use cases involve various actors interacting with the UniEvent system to facilitate event management at Mahindra University:



2.1.1 Event Proposal and Approval Flow

Description: A student club head initiates an event proposal which then goes through multiple levels of automated approvals. Once approved, it gets listed in the event calendar accessible by students.

Actors: Club Head, Approval Authorities (Faculty Coordinators, Department Heads, Administrative Staff)

Usage Steps:

1. Club Head logs into the system using university credentials
2. Club Head navigates to "Create New Event" and completes the proposal form
3. System validates input and submits for first-level approval (Faculty Coordinator)
4. System notifies Faculty Coordinator of pending approval
5. Faculty Coordinator reviews and approves/rejects/requests changes
6. If approved, proposal automatically advances to next approval level
7. Final approver confirms logistics and resources
8. System updates event status to "Approved" and publishes to calendar
9. Club Head receives notification of approval

2.1.2 Student Event Discovery and Registration

Description: Students browse upcoming events, register for participation, and receive confirmation and reminders.

Actors: Students, Notification System

Usage Steps:

1. Student logs into system using university credentials
2. Student navigates to "Upcoming Events" dashboard
3. Student filters events by interest, date, or organizing club
4. Student selects an event to view details
5. Student registers for the event
6. System confirms registration and sends confirmation
7. System adds event to student's personal calendar
8. System sends reminders before the event
9. Student receives QR code for check-in

2.1.3 Participation Tracking and Badge Award

Description: Students attend events, get attendance verified, and earn participation badges based on engagement levels.

Actors: Students, Event Organizers, System Badge Engine

Usage Steps:

1. Student arrives at event and presents QR code
2. Event organizer scans code using the system
3. System verifies student registration and marks attendance
4. After event completion, system calculates participation points
5. System evaluates badge eligibility based on participation history
6. System awards appropriate badges to student profile
7. Student receives notification of badge awarded
8. Badge appears on student's profile and achievement dashboard

3 Design Overview

This section provides a high-level overview of the UniEvent software design.

3.1 Design Goals and Constraints

The design of UniEvent is guided by several critical goals and constraints that shape architectural decisions and implementation strategies.

Goals:

- Efficient multi-role workflow handling: The system must seamlessly manage different user roles and their specific permissions, ensuring appropriate access control while maintaining usability.

- High system responsiveness and uptime: Response times under 2 seconds for standard operations and 99% uptime to ensure reliable access during critical event periods.
- Secure and scalable deployment: Robust authentication mechanisms and scalable infrastructure to handle varying loads during peak usage periods.
- Intuitive user experience: Role-specific interfaces that present relevant functions clearly without overwhelming users with unnecessary options.
- Extensible architecture: Modular design allowing for future expansion of features without significant restructuring.
- Data integrity and consistency: Ensuring that event information, approvals, and student records remain accurate throughout the system.

Constraints:

- University IT infrastructure compliance: Must operate within the existing IT ecosystem at Mahindra University, including security policies and network restrictions.
- Academic calendar alignment: Development, testing, and deployment schedules must account for academic breaks and examination periods.
- Low-cost/free hosting solutions: Budget limitations require efficient use of resources and preference for open-source technologies.
- Browser compatibility: Must support standard browsers used within the university environment (Chrome, Firefox, Safari, Edge).
- Mobile responsiveness: Must function effectively on various screen sizes without requiring native mobile applications.
- Limited development timeframe: The project must be completed within a single academic semester.

3.2 Design Assumptions

The design is based on several key assumptions that guide implementation decisions:

- Stakeholder feedback: Department heads, club representatives, and administration will provide timely feedback during development iterations.
- University SSO integration: Authentication will rely on the university's existing Single Sign-On system, assuming API access will be granted.
- Calendar API access: Integration with student calendars assumes access to Outlook/Google Calendar APIs
- Notification capabilities: System assumes access to standard notification APIs (e.g., SendGrid for email, optional Twilio for SMS).
- Consistent internet access: Users will have reliable internet connectivity within the university campus.
- Browser standards compliance: Users will access the system using modern, standards-compliant web browsers.
- Data privacy consent: Students are assumed to have provided consent for their participation data to be stored and analyzed within the system.
- IT department support: University IT will provide necessary server access and deployment assistance.

3.3 Significant Design Packages

The UniEvent system is structured into the following major design packages, each encapsulating specific functionality:

- Authentication & Session Management:
  - User authentication
  - Session handling
  - Role-based access control
  - Profile management
- Event Workflow Engine:
  - Proposal creation and submission
  - Multi-level approval routing
  - Status tracking and notifications
  - Version control for proposal revisions
- User Dashboard & Event Discovery:
  - Personalized event recommendations
  - Advanced search and filtering
  - Calendar integration
  - Registration management
- Venue & Resource Management:
  - Room booking system
  - Equipment allocation
  - Availability checking
  - Conflict resolution
- Participation & Badge Tracking:
  - Attendance verification
  - Badge award rules engine
  - Achievement progression
  - Participation history
- Notification Scheduler:
  - Email notifications
  - In-app alerts
  - Reminder scheduling
  - Batch processing
- Analytics & Reporting Module:
  - Event performance metrics
  - Participation statistics
  - Department/club activity reports
  - Trend analysis

3.4 Dependent External Interfaces

The table below lists the public interfaces this design requires from other modules or applications.

External Application	Using Module and Interface Name	Functionality/Description
University SSO System	Authentication & Session Management / SSO API	Used for validating student and faculty credentials, retrieving basic user profile information, and confirming university role
University Calendar System	Event Workflow Engine / Calendar API	Integration to check venue availability, book resources, and add approved events to institutional calendars
Email Gateway Service	Notification Scheduler / Email API	SendGrid or university SMTP server for sending event notifications, approval requests, and registration confirmations
QR Code Generation Service	Participation & Badge Tracking / QR API	Generation of unique QR codes for event check-ins and attendance verification
Google/Microsoft Calendar	User Dashboard / Calendar Sync API	Optional integration allowing students to sync registered events with personal calendars

3.5 Implemented Application External Interfaces

The table below lists the implementation of public interfaces this design makes available for other applications.

Interface Name	**Module Implementing the Interface**	**Functionality/Description**
Event API	Event Workflow Engine	RESTful API providing event creation, retrieval, and management operations for potential future mobile app or third-party integrations
User API	Authentication & Session Management	Secure endpoints for user authentication, profile management, and role validation
Analytics API	Analytics & Reporting Module	Data endpoints providing aggregated analytics on events and participation for potential integration with university dashboards
Badge API	Participation & Badge Tracking	Endpoints for checking and awarding badges, retrievable by other university systems for potential integration with broader student achievement platforms

4 Logical View

This section presents the architectural organization of the software system, detailing how components interact to implement the required functionality. The logical view provides a comprehensive understanding of the system structure from high-level architecture down to key class collaborations.

4.1 Design Model

UniEvent follows a layered architecture with clear separation of concerns between presentation, business logic, and data access components. The system is built using modern web technologies with emphasis on maintainability, scalability, and security.

4.1.1 Architecture Overview

The system employs a modified MVC (Model-View-Controller) architecture with additional service layers:

- Presentation Layer:
  - Frontend: Vanilla JavaScript with role-based dynamic routing
  - Responsive UI components with minimal dependencies
  - Client-side validation and state management
- Application Layer:
  - RESTful API endpoints using Node.js with Express
  - Service modules for business logic implementation
  - Authentication middleware for security enforcement
  - Workflow orchestration services
- Data Access Layer:
  - MongoDB database connectors
  - Repository pattern for data operations
  - Data validation and transformation
- Cross-Cutting Concerns:
  - Logging and monitoring services
  - Error handling middleware
  - Caching mechanisms
  - Security utilities

4.1.2 Key Modules and Classes

User Management Module:

- UserController: Handles authentication requests and profile operations
- UserService: Implements user-related business logic
- UserRepository: Manages data access for user information
- RoleManager: Enforces role-based permissions

Event Module:

- EventController: Processes event CRUD operations
- EventService: Implements event-related business logic
- EventRepository: Manages data access for events
- ApprovalWorkflow: Orchestrates the approval process

Badge and Participation Module:

- BadgeController: Handles badge-related operations
- ParticipationService: Tracks and validates event attendance
- BadgeRulesEngine: Determines badge eligibility
- AchievementRepository: Manages badge and achievement data

4.2 Use Case Realization

This section details how the system's components collaborate to implement the key use cases identified in Section 2.

4.2.1 Event Proposal and Approval Realization

The sequence diagram below illustrates how the event proposal and approval flow is implemented:

1. Club Head fills out the event proposal form via the UI
2. Form data is validated client-side and submitted to the EventController
3. EventController forwards request to EventService for business validation
4. EventService creates a new event record with "Pending" status via EventRepository
5. ApprovalWorkflow identifies required approvers based on event type and department
6. NotificationService alerts the first approver of pending action
7. Approver reviews event details and submits decision
8. ApprovalWorkflow updates event status and routes to next approver if approved
9. Once final approval is received, EventService updates event status to "Approved"
10. NotificationService alerts Club Head of approval and publishes to calendar
11. Students can now view and register for the event

4.2.2 Student Registration and Participation Realization

The activity diagram below illustrates the student registration and participation process:

1. Student browses events through EventController (filtered query to EventRepository)
2. Student selects event and initiates registration via RegistrationController
3. RegistrationService validates eligibility and creates registration record
4. NotificationService sends confirmation with QR code
5. At event, QR code is scanned by AttendanceController
6. ParticipationService validates attendance and records participation
7. After event, BadgeRulesEngine evaluates participation for badge eligibility
8. If criteria met, BadgeService issues new badge to student profile
9. NotificationService alerts student of new achievement

5 Data View

This section describes the persistent data structures underlying the UniEvent system, including entity relationships and data dictionary definitions.

5.1 Domain Model

The domain model represents the key entities in the UniEvent system and their relationships:

Core Entities:

- User: (id, name, email, role, department, profile_image)

    - Represents students, club heads, faculty, and administrators
    - Contains authentication and profile information
- Event: (id, title, description, start_date, end_date, location, capacity, creator_id, approval_status, visibility)
    - Represents events organized by clubs
    - Maintains status throughout approval workflow
    - Links to venue and resources
- Approval: (id, event_id, approver_id, status, comments, timestamp)
    - Tracks approval decisions and comments
    - Maintains history of approval workflow
- Registration: (id, event_id, user_id, registration_date, attendance_status, feedback)
    - Records student registrations for events
    - Tracks attendance and participation
- Badge: (id, name, description, criteria, image_url)
    - Defines available achievement badges
    - Establishes criteria for awarding
- UserBadge: (id, user_id, badge_id, date_earned)
    - Links users to earned badges
    - Records achievement history
- Venue: (id, name, capacity, facilities, availability)
    - Catalogs available event spaces
    - Tracks booking status and conflicts
- Notification: (id, user_id, message, type, is_read, created_at)
    - Stores system notifications
    - Tracks delivery and read status

5.2 Data Model (Persistent Data View)

The data is stored in MongoDB, a NoSQL database that provides flexibility for complex document relationships while maintaining good performance characteristics.

5.2.1 Data Dictionary

This section defines the key data elements within the UniEvent system:

User Collection:

- user.role: Enum(Student, ClubHead, FacultyCoordinator, DepartmentHead, Admin, Logistics)
- user.department: String (Engineering, Arts, Science, Business, etc.)
- user.status: Enum(Active, Inactive, Suspended)

Event Collection:

- event.approval_status: Enum(Draft, Pending, ApprovalLevel1, ApprovalLevel2, Approved, Rejected, Cancelled)

- event.type: Enum(Academic, Cultural, Technical, Sports, Workshop, Seminar)
- event.visibility: Enum(Public, DepartmentOnly, InviteOnly)
- event.priority: Enum(Low, Medium, High)

Badge Collection:

- badge.type: Enum(Participation, Organization, Achievement, Special)
- badge.level: Enum(Bronze, Silver, Gold, Platinum)
- badge.category: Enum(Technical, Cultural, Leadership, Academic)

Registration Collection:

- registration.status: Enum(Registered, Waitlisted, Cancelled, Attended, NoShow)
- registration.feedback_status: Enum(None, Submitted, Featured)

Notification Collection:

- notification.type: Enum(Approval, Rejection, Reminder, Badge, AnnouncementGeneral)
- notification.priority: Enum(Low, Normal, High, Urgent)
- notification.delivery_channel: Enum(InApp, Email, SMS, All)

6 Exception Handling

UniEvent 2.0 implements a comprehensive exception handling strategy to ensure system stability, provide meaningful feedback to users, and facilitate troubleshooting. This section outlines the exception types, handling mechanisms, and recovery strategies.

Authentication Exceptions:

- LoginError: Thrown when login credentials are incorrect or invalid
  - Logged with timestamp and IP address (no credentials logged)
  - User receives friendly message suggesting password reset
  - After 5 consecutive failures, account is temporarily locked for security

Authorization Exceptions:

- PermissionDeniedError: Thrown when users attempt to access unauthorized resources
  - Logged with user ID, requested resource, and timestamp
  - User receives appropriate message explaining access limitation
  - For security, specific permission details are not disclosed

Resource Conflicts:

- EventConflictError: Thrown when attempting to book already reserved venues
  - Includes conflicting event IDs and time slots
  - User offered alternative venues or time slots when possible
  - Logs conflict patterns to identify scheduling optimization opportunities

External Service Failures:

- APIError: Thrown when external service integration fails (e.g., calendar sync)
    - Implements graceful degradation - system continues with reduced functionality
    - Automatic retry mechanism with exponential backoff
    - Admin notification for persistent failures

Data Validation Errors:

- ValidationError: Thrown when submitted data fails validation rules
    - Returns specific field errors to assist immediate correction
    - Client-side validation prevents most common errors
    - Server-side validation ensures data integrity

7 Configurable Parameters

UniEvent is designed with flexibility in mind, allowing administrators to adjust system behavior through configurable parameters without requiring code changes. This section details the key configuration options available.

Configuration Parameter Name	Definition and Usage	Dynamic?
SESSION_TIMEOUT	Duration in minutes before user sessions expire due to inactivity (default: 30)	YES
EVENT_APPROVAL_THRESHOLD	Minimum number of approvals required based on event size and type	YES
MAX_REGISTRATIONS_PER_EVENT	Default maximum registrations allowed for standard events	YES
NOTIFICATION_LEAD_TIME	Hours before event when reminder notifications are sent	YES
BADGE_POINT_THRESHOLDS	Point requirements for each badge level (Bronze, Silver, Gold)	NO
WAITLIST_AUTO_UPGRADE	Whether to automatically upgrade waitlisted registrations when spots open	YES
FEEDBACK_COLLECTION_WINDOW	Days after event when feedback can be submitted	YES
REGISTRATION_CUTOFF_HOURS	Hours before event when registration closes	YES
REPORT_GENERATION_SCHEDULE	Cron expression for when to generate automated activity reports	YES
EMAIL_TEMPLATE_PATHS	File system paths to customizable email notification templates	NO

Configuration is managed through a combination of:

- Environment variables for deployment-specific settings
- Database-stored configurations for dynamically updatable parameters
- Configuration files for static parameters

Administrators can modify dynamic parameters through an administrative interface, while non-dynamic parameters require application restart to take effect.

8 Quality of Service

This section outlines the quality attributes of UniEvent , including availability guarantees, security measures, performance expectations, and monitoring capabilities.

8.1 Availability

UniEvent is designed to be highly available, particularly during peak usage periods such as club registration drives and major event days.

Availability Targets:

- 99% uptime guarantee during academic semesters
- 99.5% availability during business hours (8 AM - 8 PM)
- Scheduled maintenance windows communicated at least 48 hours in advance

High Availability Mechanisms:

- Redundant database backups with daily snapshots and point-in-time recovery
- Automated service health checks every 5 minutes
- Auto-restart mechanisms for application components
- Graceful degradation for non-critical features during load peaks

Maintenance Strategy:

- Scheduled maintenance during off-hours (typically 2 AM - 4 AM)
- Rolling updates to minimize downtime
- Maintenance history tracking for audit purposes
- Emergency maintenance procedures with defined SLAs

8.2 Security and Authorization

Security is a critical aspect of UniEvent, protecting sensitive university data and ensuring appropriate access control.

Authentication Strategy:

- Role-Based Access Control (RBAC) with 6 distinct roles
- JWT token-based authentication with appropriate expiration
- Integration with university Single Sign-On where available
- Secure password storage using bcrypt hashing algorithm
- Multi-factor authentication for administrative accounts

Data Protection:

- All sensitive data encrypted at rest
- Data access logging for audit purposes

Security Controls:

- CAPTCHA on login and proposal submission to prevent automated attacks
- Comprehensive audit logging of security-relevant actions

Compliance:

- Alignment with university data protection policies
- Regular security reviews and vulnerability scanning
- Security incident response plan

8.3 Load and Performance Implications

UniEvent is designed to handle varying loads, particularly during peak registration periods and popular events.

Performance Requirements:

- Support for 100+ concurrent users during normal operation
- Capacity for 500+ concurrent users during peak events
- Page load times under 3 seconds for standard operations
- Search query response under 2 seconds

Optimization Strategies:

- MongoDB indexing for frequently queried fields
- Caching for static content and common queries
- Asynchronous processing for non-critical operations
- Optimized database queries with proper indexing

Scalability Approach:

- Horizontal scaling capability for API servers
- Connection pooling for database efficiency
- Resource monitoring with automatic alerts at 70% threshold
- Load testing for 500 concurrent users before deployment

8.4 Monitoring and Control

Comprehensive monitoring ensures system health and performance, allowing proactive issue resolution.

Monitoring Infrastructure:

- Application performance monitoring dashboard

- Real-time error tracking and alerting
- Database query performance analysis
- API endpoint response time tracking

Operational Controls:

- Admin dashboard for system status visualization
- Configurable alerting thresholds
- Error log aggregation and analysis
- Performance trend reporting

Metrics Collection:

- Error rates and types
- Database query performance
- User activity patterns
- Event popularity metrics
- Approval workflow efficiency