Development of a Multi-Campus Academic Event Listing and Management System for Berlin Universities

Understanding the Scenario:

The project entails developing a web-based platform for listing and publicising scholarly discussions and activities on three campuses of Berlin universities. Students, researchers, postgraduates, and outside interested parties are the main stakeholders.

1a. Main Scenario:

By offering a single platform for event discovery, the system seeks to promote cooperation and information exchange. Important components consist of:

- A platform that is easy to use and available to both university-affiliated users and the broader public.
- Integration with verification procedures to guarantee email validation for users affiliated with universities.
- Enabling users to plan, coordinate, and take part in events or discussion shows that are pertinent to their interests.
- The provision of transportation and travel information to make campus navigating easier.

1b. Sub-scenarios

1. University Affiliation Verification: Users claiming university affiliation must provide a university email for verification.
2. Event Management: Permit administrators or authorized users to plan events or a sequence of discussions.
    - Interest groups can use event curation tools to promote pertinent events in a targeted manner.
3. Subscription System: Allow people to sign up for notifications about events or topics that interest them.
   Offer choices to follow particular speakers, divisions, or themes.
4. Integration of Travel Information: Provide links to transportation platforms such as Google Maps or Translink for the locations of events.
   -Provide estimates of trip time according to the user's location.

1c. Specification Level 1 – Tables

User	Admin	Event	Location	Notification
Create account (Register)	Create & Approve event	Show events	Add location	Notify participants
Login to system	Manage user roles	View events details	check location	Event updates
Manage profile	Moderate interest groups	Subscribed events	Edit location	Subscription reminders
Subscribe to events	Upload Event details	Events details page	Return address	Alert changes
Search events	Assign moderators	Filter by topics	Show on map	Cancelation alerts
View subscribed events	Track event approvals	Overview of event	Generate directions	
Provide feedback	Review feedback	Rate event	Access transport API	Feedback request

User

Function	Description	Number of Interfaces and Interfaces
Register	Name, University , studies, email, password,age,gender	Registration form, user portal - 2
Log in	Email, password	Login page – 1
Mange profile	Address, subscription, interests	Profile settings page - 1
Subscribe to events	Choose the desires and subscribe to them	Event subscription page - 1
Search events	Search events, filter by topics, search dates	Event search page - 1
View subscribed events	See all events subscribed to,	Subscriptions page - 1
Provide feedback	Rate and comment on attended events.	Feedback submission form - 1

Admin

Function	Description	Number of Interfaces and Interfaces
Login	Email, password	Login page – 1
Create & Approve event	Add and approve events in the system	Event management dashboard - 1
Manage user roles	Assign roles and oversee user activities	User management interface - 1
Moderate interest groups	Approve or reject interest group applications.	Interest group management page -1
Upload Event details	Title,speaker, location,date , capacity	Event details page - 1
Assign moderators	Assign specific moderators to events.	Event subscriptions page - 1
Track event approvals	Monitor the status of events awaiting approval.	1
Review feedback	View and address user feedback for events.	Feedback review dashboard - 1

Event

Function	Description	Number of Interfaces and Interfaces
Show events	Display available events for users to browse	Event list page - 1
View events details	Provide detailed information about events; location,date , speaker , capcity, etc	Event details page - 1
Subscribed events	List events subscribed to by a user	Subscriptions page - 1
Filter by topics	filtering events by categories	Event filter options - 1
Overview of event	Present summary of the event's features	Event overview page - 1
Rate event	Display rate of the event and oter useres comments/ feedback	1

Location

Function	Description	Number of Interfaces and Interfaces
Add location	Input location details for events	Location entry form - 1
check location	Verify and display event location details	Location details page - 1
Edit location	Modify event location details	Location editing interface - 1
Return address	Address or a map link being added	Address display field - 1
Show on map	Visualize event location on a map	Map integration interface - 1
Generate directions and Access transport API	Provide directions to the event location	Directions page - 1

Notification

Function	Description	Number of Interfaces and Interfaces
Notify participants	Send updates to users about events	Notification interface – 1
Event updates	Inform users of changes to event details	Update notifications module – 1
Subscription reminders	Send reminders for subscribed events	Reminder notifications module - 1
Alert changes	Notify users of major changes or cancellations	Alerts and cancellations panel – 1
Feedback request	Request feedback from users post-event	Feedback request notifications and review page – 2

Initiating

2a. Feasibility Study

Verification (Input Processing)
SRS Format
A Feasibility Study (Requirement Gathering and Processing)

2a -Specification

- To be concluded collaboratively by the stakeholders, including:
  - University administrators.
  - Academic staff.
  - Event organizers and technical teams.
  - End-users like students, researchers, and external participants.

2b -Budget Estimated Cost: 40,000$ - 70,000$

  - Project Size: Multi-campus system covering three major Berlin universities.
  - Functionality: Core features like event listing, user authentication, notifications, and transport integration.
  - Quality Requirements: Ensure scalability, security, and usability.

Prioritisation:

  - Begin with essential features (event creation, browsing, notifications).
  - Gradually expand to advanced features (recommendations, analytics, multi-language support).

2c -Market and Industry Analysis (SWOT Methodology)

- **S**trengths:
  - Centralized platform reduces duplication of efforts across campuses.
  - Promotes cross-campus collaboration and knowledge sharing.
- **W**eaknesses:
  - Potential technical challenges in integrating with external APIs (e.g., Google Maps).
  - Initial user onboarding and adoption hurdles.
- **O**pportunities:
  - Expand to include more universities or regions in the future.
  - Generate revenue through premium features or external sponsorships.
- **T**hreats:
  - Competition from existing platforms or event management tools.
  - Data privacy concerns, especially under GDPR regulations.

-Competitors

https://gomomentus.com
https://www.eventbrite.com
https://www.eventbookings.com/en-de/

- Scope and Audience Definition

- Scope:
    - Focus on academic event management within Berlin universities.
    - Highlight unique features like cross-campus event visibility and transport integration.
    - Address operational gaps identified in current systems, such as fragmented communication or lack of event discovery tools.

- Audience:
    - Primary Users: Students, researchers, postgraduates, and external participants.
    - Secondary Users: Administrators, interest group leaders, and event organizers.

2d. Quantitative Study:

- Conduct surveys targeting end-users to gather insights on operational challenges.
- Examples: "What features would make it easier to discover academic events across campuses?"

Qualitative Study:

- Conduct interviews with technical experts and university administrators.
- Discuss scalability, technical feasibility, and integration challenges.

Feasibility Study Summary Table

Type	Main Points	Outcome
Technical Feasibility	Integration with Maps, Translink APIs.	Feasible with current tools
Economic Feasibility	Hosting and third-party API costs	Budget-friendly.
Operational Feasibility	User and admin adaptability to the new platform	High adaptability expected
Legal Feasibility	Compliance with GDPR and local regulations.	Requires privacy and data checks

2e. Functional Requirements Table

Feature	Description	User
User Registration	enables users to sign up and confirm their university email address.	Students
Event Management	Creating and managing events tools	Administrators
Subscription	users can subscribe to desired topics	Students
Event addresses	Integration with transport systems and map	Participants
Event Rating and Feedback	Allows attendees to rate and review events.	Participants
Advanced Search and Filtering	Filter by topic, location, or speaker.	Participants
Personalised Recommendations (notifications)	Suggests events based on user preferences	Administrators, Participants

Non-functional Requirements Table

Requirement	Description	Measure
Scalability	Manage a large number of concurrent users.	Support at least 10,000 users.
Security	Protect user data and authentication.	Testing Methods – Code review, penetration testing, scan
Usability	Make sure the user interface is easy to use.	Test interface intuitiveness and task completion time for key actions.
Performance	Ensure quick response times.	Page load time under 2 seconds
Accessibility	Accessible for users with disabilities.	Test with screen readers and other assistive technologies.
Localisation	Support multiple languages for diverse users. Provide English, German initially	Surveying participants language preferences
Maintainability	The system should be easy to update and fix.	Time required to implement updates or fix bugs.

Diagnosing

2f.Requirement Gathering

The main source of gathering requirements is:

- **Stakeholders**, the university students and staff members plus IT administrators from the three said universities.
- **Surveys/Questionnaire**, in-depth surveys to research about what user features would they prefer.
- **Observations**, currently we can conduct an analysis on how our target audience find and attend events.
- **Existing Systems**, look at similar current systems and find features that work well and gaps to improve on.

Goal: Using stakeholder input and methodical analysis, gather and identify technical, non-functional, and functional requirements.

- University Administrators: Establish specifications for reporting requirements, event approval, and system governance.
- Academic Staff: Indicate how you would like academic events to be created, run, and shared.
- Event Organizers: Specify requirements for user interaction, alerts, and event production.
- End users: (students, researchers, and outside participants) Emphasize elements that are needed to make finding events, subscribing, and providing feedback simple.

Requirements Gathering Success Criteria

1. Thorough Coverage
   Make sure that the demands of all parties involved students, administrators, and event planners, are recognised and recorded.
   Employ a variety of techniques, such as focus groups, interviews, and surveys, to record both explicit and implicit needs.
   All-important stakeholders concur that their needs are met, which is a success indicator.
2. Validation
   With stakeholder permission, confirm the requirements' correctness and viability.
   Make use of mockups or prototypes to confirm expectations.
   The formal approval of requirements and the absence of unresolved disagreements are indicators of success.
3. Setting priorities
   Sort requirements into Must-Have, Should-Have, Could-Have, and Won't-Have categories using the MoSCoW framework.
   Prioritise high-priority features like notifications and event management with your resources.
   Stakeholders agree on a clear list of priorities, which is a success indicator.

4. Traceability

   Use a traceability matrix to associate each need with a particular stakeholder or business necessity.

   Make sure that every feature has a documented purpose before it is implemented.

   A clear link between the criteria and the finished features is a success indicator.

5. Feasibility

   Verify that all needs are technically, financially, and operationally feasible.

   Examine complicated requirements (such API integration) on a regular basis for viability.

   Success Indicator: Every need may be carried out within the allocated spending limit and time frame.

6. Accessibility and Usability

   Prioritise WCAG accessibility standards compliance and user-friendly design.

   Prioritise mobile responsiveness and multilingual support.

   Positive usability testing feedback is a success indicator.

7. Continuous Review

   Adjust requirements to changing stakeholder needs by utilising Agile techniques.

   Refine requirements by reviewing them at regular intervals.

   Success Indicator: There are no significant disruptions to requirements, which are in line with project objectives.

Methods Used for Gathering Requirements

Method	Description	Purpose
Surveys	Staff and students were given the materials to help them find characteristics that enhance event management and discovery.	Quantitative insights into user preferences
Interviews	Carried out in collaboration with technical teams and university administration.	Qualitative input on operational challenges.
Focus Groups	Event planners talked about desired features and problems with the current system in groups.	Identified features for event creation and management
Observation	Examined current event systems to identify common issues and weaknesses.	Comparative analysis for improvements.
Document Analysis	Examined university rules and procedures to make sure they adhered to accessibility guidelines and GDPR.	Legal and compliance requirements.

Tools for Requirement Gathering and Validation

Tool	Purpose
Google forms	Surveys to collect feedback
Zoom / Microsoft teams	Conducting interviews and focus groups
Jira / Trello	Managing and prioritising requirement
Figma	Creating prototypes to validate requirements

Establishing

2g. Specification Level 2 – Tables

Specification	Requirements
Software Requirements	- Backend: Python framework. - Frontend: ReactJS/HTML/ CSS. - Database: MySQL /PostgreSQL
Hardware Requirements	- Server: AWS EC2 instances. - Storage: Minimum 100GB SSD for data. - Backup: Automated daily backups.
Quality Parameters	- Uptime: 99.9%. - Response time: Less than 2 seconds for core functionalities. - Error rate: <1%.
Security Requirements	- Two-factor authentication (2FA). - SSL/TLS for encryption. - Role-based access control.
APIs and Integration	- Maps API for travel info. - Email API for notifications. - OAuth for secure login.
Scalability	- Auto-scaling servers to handle up to 10,000 concurrent users. - CDN (e.g., Cloudflare) for content delivery.
Monitoring and Maintenance	- Tools: Grafana for performance monitoring, Prometheus for alerts. - Logging: Centralised logging with ELK Stack.
Testing Framework	- Functional tests with Selenium. - Load tests with JMeter. - Security tests with OWASP ZAP.

User Registration and Verification

Specification	Requirement	Scenarios	Interfaces
Software Requirements	User registration module with email verification (e.g., university email check)	New users signing up and verifying accounts	Registration form, Email API
Quality Parameters	Response time: <2 seconds for verification. Error rate: <1%	Email verification under peak traffic	Verification page, email server
Security Requirements	SL encryption for sensitive data. Two-factor authentication for login	Secure user data submission and login	Login module, Authentication API

Event Creation and Management

Specification	Requirement	Scenarios	Interfaces
Software Requirements	Event creation form, ability to list recurring talks. Integration with Google Maps API for location tagging	Organisers creating and updating events	Event form, Maps API
Hardware Requirements	Servers to handle file uploads for event images and media (minimum 10GB)	Events requiring multimedia uploads	Cloud storage, server infrastructure
APIs and Integration	Maps API for location data. Email API for event notifications. OAuth for organiser authentication	Transport and location-based event visibility	Maps and email APIs

Subscription and Notifications

Specification	Requirement	Scenarios	Interfaces
APIs and Integration	Email API for notification updates. Subscription management API for recurring notifications.	Users subscribing to events	Email and subscription module
Scalability	Support up to 10,000 simultaneous subscriptions. CDN for faster notification delivery	Users subscribing to events	Email and subscription module
Quality Parameters	Support up to 10,000 simultaneous subscriptions. CDN for faster notification delivery	Notifications for event cancellations/updates	Email and subscription module

Transport and Location Integration

Specification	Requirement	Scenarios	Interfaces
APIs and Integration	Maps API for route optimisation. Translink API for real-time public transport details	Users searching transport info for events	Maps and route visualisation
Monitoring and Maintenance	Regular monitoring of transport APIs to ensure accuracy. Fallback options in case of API downtime.	Transport API failures affecting users	Transport integration dashboard
Hardware Requirements	Server caching for frequently accessed routes (e.g., inter-campus travel)	Repeat users accessing transport info	Transport caching mechanism

Feedback and Analytics

Specification	Requirement	Scenarios	Interfaces
Software Requirements	Feedback submission form with ratings and comments. Analytics dashboard for event performance tracking	Post-event feedback submission by users	Feedback form, Analytics dashboard
Security Requirements	Anonymised feedback to protect user identities. Role-based access for analytics (organisers only)	Secure feedback collection and review	Feedback review panel

Accessibility and Usability

Specification	Requirement	Scenarios	Interfaces
Usability Requirements	WCAG 2.1 compliance for accessibility. Screen reader compatibility	Visually impaired users accessing the platform	Accessible UI for event pages
Quality Parameters	Multi-language support for German and English (extendable)	Users selecting their preferred language	Language selection dropdown

3a. Requirement Modelling

Clarity in functionality and alignment with business objectives are ensured by requirement modelling, which aids in the visualization and structuring of the system's requirements. The models utilized to depict and arrange the multi-campus event management system's requirements are shown below.

How user reacts with the software

Interaction	Description	Functionality	User Role
Login and Signup	Users log in or create an account to access the platform.	User authentication and verification.	Students, Administrators, Participants
Search Events	Users search for events by topic, date, or location.	Advanced search and filtering capabilities.	Students, Participants, External Users
Register for Events	Users register for events and receive confirmation notifications.	Event registration and notification system.	Students, Participants, External Users
Create and Manage Events	Administrators and presenters create and edit event details.	Event creation and management tools.	University Administrators, Event Presenters
Submit Feedback	Users rate and review events after attending.	Feedback and rating system.	Students, Participants, External Users
Receive Notifications	Users receive updates about new or upcoming events.	Notification system (email or in-app).	All Users
View Speaker Profiles	Users view detailed profiles of speakers.	Profile management and display system.	All Users

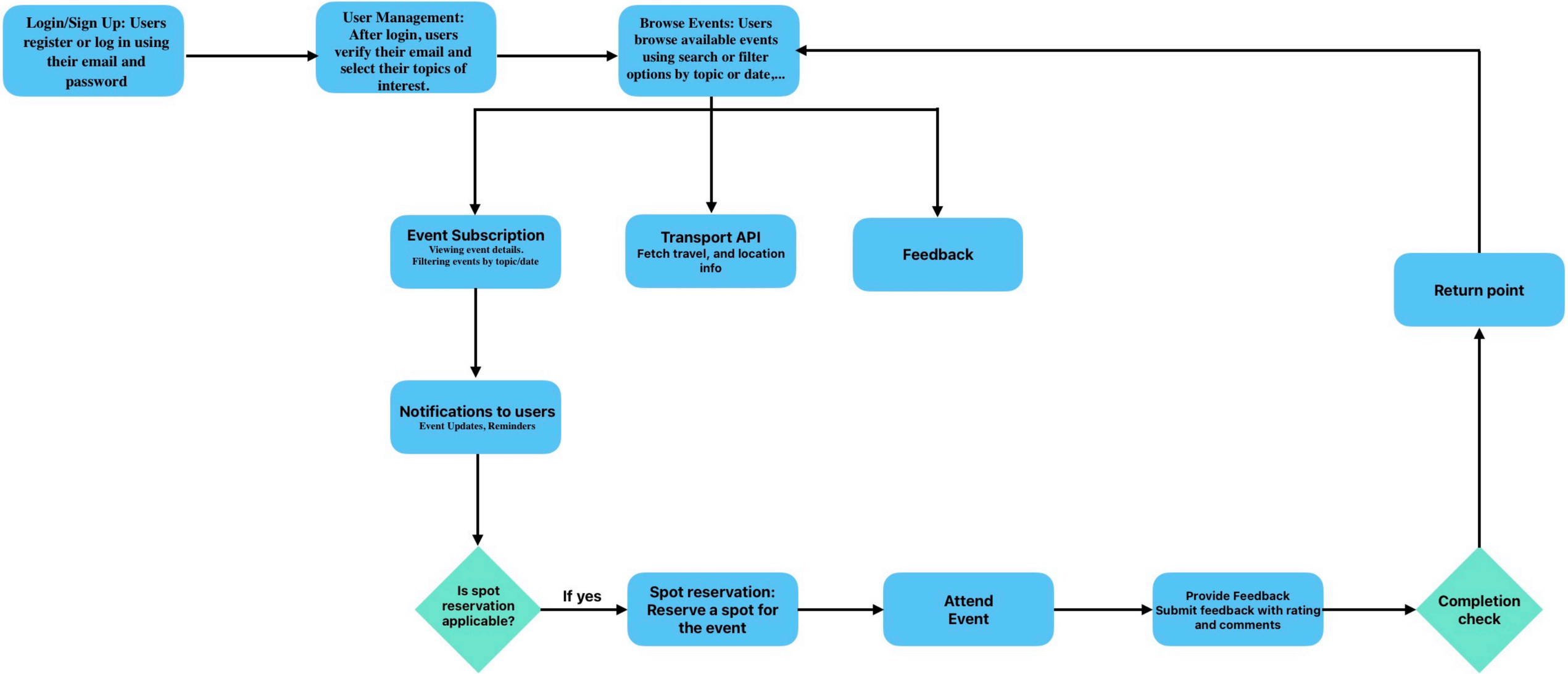
How the machine should respond back. Software behavior

User Action	System Response	Behavior Modelled
Login/Signup	Authenticate user, verify university email, and redirect to the dashboard.	Secure authentication and validation.
Search Events	Display filtered event results with details like location and time.	Quick database queries and results display.
Register for Events	Save registration, send confirmation email, and update event capacity.	Data persistence and notification handling.
Create Events	Validate input, save event details, and notify subscribers of interest groups.	Input validation and notification triggers.
Submit Feedback	Save feedback, update average event rating, and display it on the event page.	Real-time data updates and display.
Access Notifications	Display event reminders and updates in the notification centre.	Timely notification delivery.
View Speaker Profiles	Fetch and display speaker details, including uploaded materials.	Data retrieval and rendering.

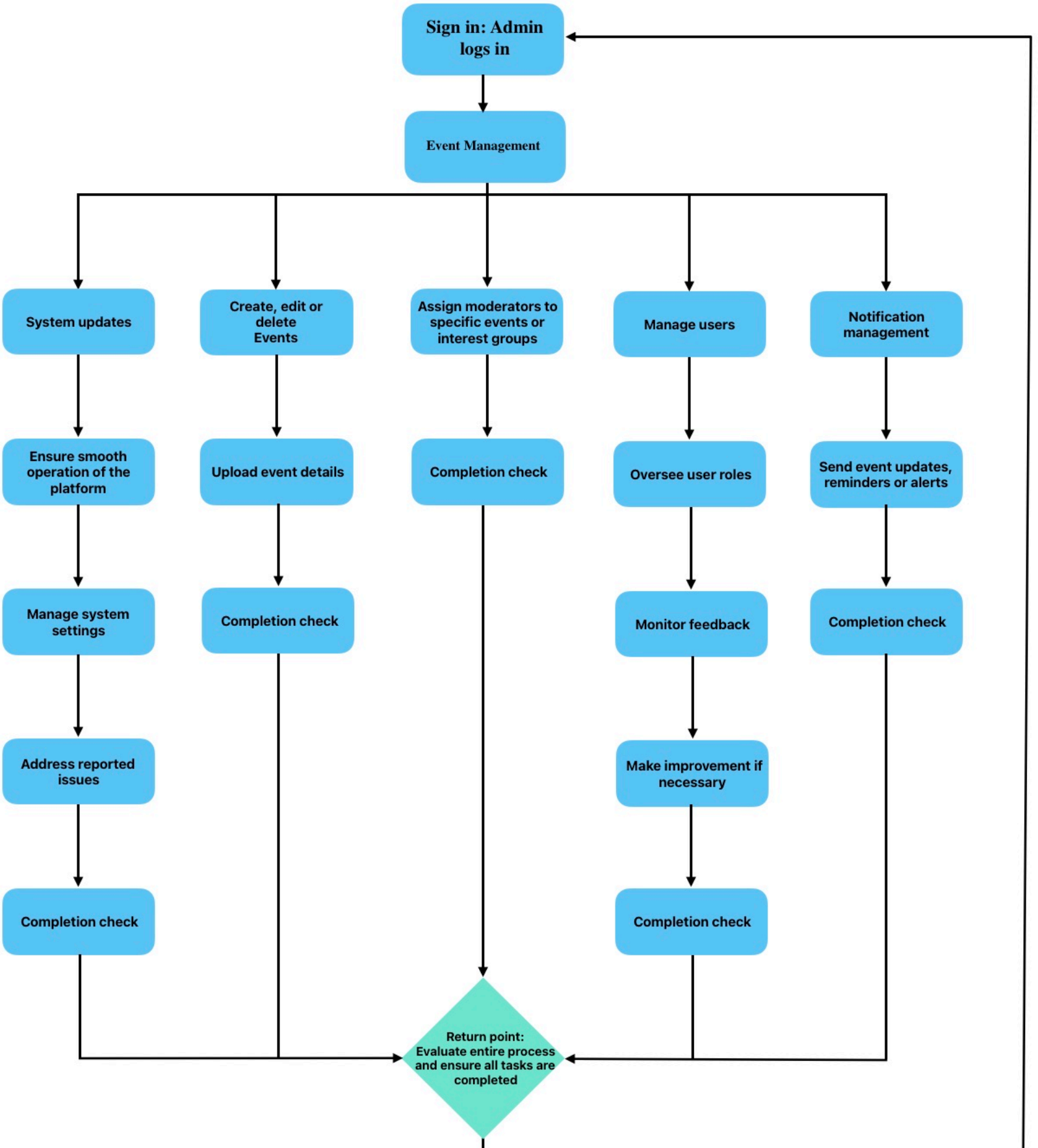
In order to measure the model, we define the metrics criteria.

Metric	Description	Measure	Quality Indicator
Page Load	Time taken for pages to load after a user action	Under 2 seconds for all actions.	System performance and responsiveness
Concurrent User Support	Number of users the system can handle simultaneously	Support for up to 10,000 concurrent users.	Scalability
Notification Delivery Time	Time taken to send event updates or reminders.	Within 1 second for all notifications.	Real-time communication.
Search Query Response Time	Time takes to display results for a search query	Under 1 second for all queries	Database optimization.
System uptime	Availability of the platform over	99.9% uptime	Reliability and maintainability.
Error Rate	Frequency of system errors or failures.	Less than 0.15 error rate.	Stability and robustness.
User Satisfaction	Feedback form users about ease of use and functionality	90% positive feedback in user surveys.	Usability and user experience.

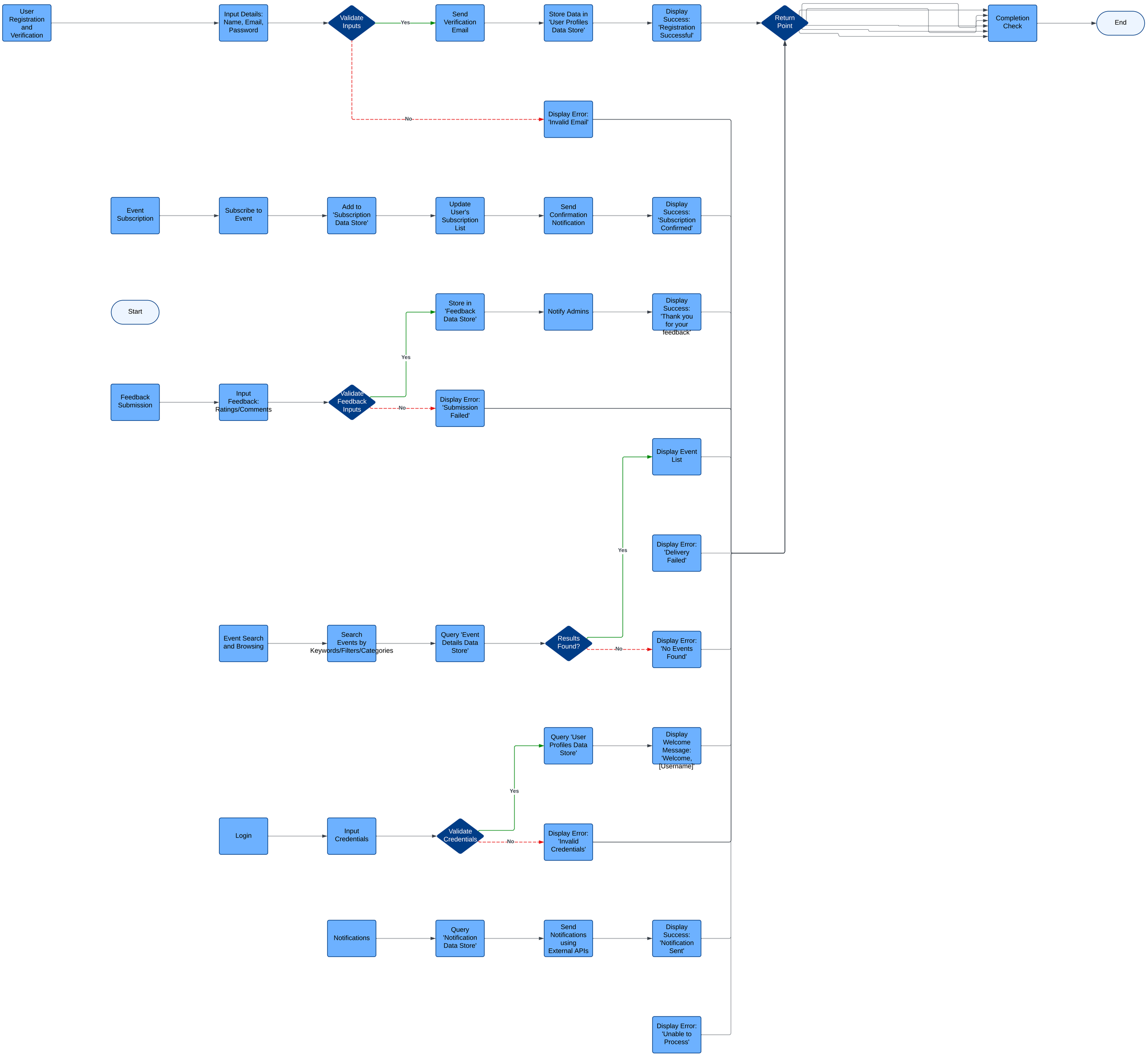
3b.User workflow



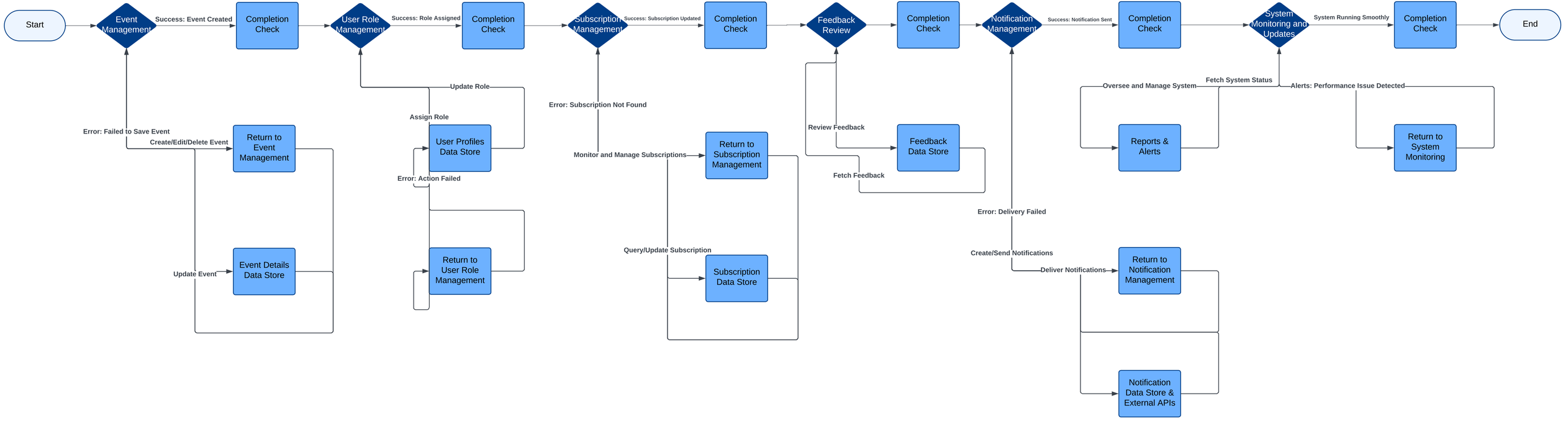
Admin workflow



3c.User DFD diagram



Admin DFD diagram



3e.Requirement Analysis

List of Users

Participant	Interests	Role in System
Students	Discover academic events and sign up for subjects.	Primary users
University Administrators	Oversee interest groups and create events.	Event and system managers.
External Users	Participate in events and explore academic interests.	Public audience with limited access
IT Administrator	Keep the platform up to date, handle technical problems	Monitor system uptime, Resolve technical bugs or failures, Manage API integrations
Event Presenter	Make their own event arrangements and oversee them.	Submit speaker profile., Upload supporting materials, Interact with attendees via Q&A

List of functions and Activities

Function/Activity	Description	User Role(s)
View Events	Browse a list of all upcoming events, including titles, dates, times, and locations.	All Users
Search Events	Use filters like topic, speaker, location, or date to find specific events.	Students, External Users, Participants
Register for Events	Sign up for events and receive confirmation of attendance.	Students, External Users, Participants
Create Events	Create and schedule new events, adding details like time, location, and speaker information.	University Administrators, Event Presenters
Edit/Manage Events	Update or delete existing events, including uploading materials or modifying details.	University Administrators, Event Presenters
Subscribe to Interest Groups	Follow specific interest groups to receive updates on relevant events.	Students, External Users
Provide Event Feedback	Submit ratings and reviews for attended events.	Students, External Users
Upload Event Materials	Upload supporting documents or slides for attendees to access.	Event Presenters
Access Speaker Profiles	View detailed profiles of event speakers, including their expertise and background.	All Users
Receive Notifications	Get alerts about new events, updates, or changes to subscribed events.	Students, External Users
Monitor System Health	Oversee system performance, uptime, and error tracking.	IT Administrators
Integrate with Maps	Access directions and travel times to event locations using mapping tools.	All Users
Participate in Q&A	Engage with event presenters through Q&A sessions.	Participants, Event Presenters

Measurements:

Behavioural Measurement

Metric	Description	Use Case
Event Engagement	Tracks the number of events viewed, subscribed, and average time spent per user.	Identify popular event topics and improve the recommendation engine.
Search Behavior	Monitors common search filters, advanced search usage, and search-to-click rates.	Optimize search functionality by analysing frequent queries and user preferences.
Feedback Activity	Measures the number of events rated or reviewed and the average rating per event.	Improve event quality by analysing feedback trends.
Subscription Trends	Tracks the most-subscribed interest groups and notification engagement rates.	Enhance subscription features and increase notification relevance to users.

Technical Measurement

Metric	Description	Use Case
System Performance	Monitors API response times, page load times, and server uptime.	Ensure quick response times and identify performance bottlenecks.
User Load	Tracks the number of concurrent users and daily/monthly active users (DAU/MAU).	Ensure scalability and system stability during peak usage.
Error Tracking	Logs failed API calls and incomplete registrations.	Debug frequent errors and enhance system reliability.
Notification Delivery	Measures the success rate and time taken for notification delivery.	Optimize real-time notification systems for event updates.

3f. Data Architecture in Requirement Definition

The goal is to create a scalable and reliable data architecture that can accommodate the academic event management system's multi-campus functional and non-functional needs. While preserving data confidentiality and integrity, the architecture guarantees effective data flow, retrieval, and storage.

Important Elements

- Fundamental Data Entities and Connections
    - The main items that the system will handle are referred to as entities. These include users, events, subscriptions, feedback, and notifications.

    - Relationships: Describe the interactions between these things.
      Relationship Examples: Users have the option to subscribe to several events.
      Feedback is associated with particular users and occasions.
      Event updates cause notifications to be sent out.

Entity	Attributes	Description
User	UserID (PK), Name, Email, Role	Represents system users (students, organisers, admins).
Event	EventID (PK), Title, Date, Location, OrganiserID (FK)	Represents academic events with associated details.
Subscription	SubscriptionID (PK), UserID (FK), EventID (FK), DateCreated	Links users to events they have subscribed to.
Feedback	FeedbackID (PK), UserID (FK), EventID (FK), Rating, Comment	Stores user ratings and comments for events
Notification	NotificationID (PK), EventID (FK), UserID (FK), Message	Represents alerts sent to users about event updates or reminders.

Data Flow

See how information flows through the system in reaction to user input.

-User Registration:

- Data: Name, Email, Role
- Flow: User inputs → Validation → Stored in User Table.

-Event Creation:

- Data: Event details (title, location, date)
- Flow: Admin inputs → Stored in Event Table → Linked to Notifications.

-Subscription:

- Data: UserID, EventID
- Flow: User subscribes → Stored in Subscription Table → Triggers Notifications.

-Feedback:

- Data: Rating, Comment
- Flow: User inputs → Stored in Feedback Table → Available to organizers.

Storage Design

-Database Model:

- Relational Database (e.g., MySQL or PostgreSQL) for structured data.
- Example Tables: Users, Events, Subscriptions, Feedback, Notifications.

-Cloud Storage:

- Use AWS S3 for large files, like event banners or multimedia.

-Indexing:

- Implement indexing for frequently queried fields (e.g., EventID, UserID

Data Integrity and Validation

-Constraints:

- Primary Keys: Uniquely identify rows in tables.
- Foreign Keys: Maintain relationships between entities (e.g., UserID in Subscriptions table).
- Unique Fields: Ensure no duplicate emails or event titles.

-Validation Rules:

- Email format validation for user registration.
- Date validation for event creation.

External API Integrations

Define how external systems contribute to the data architecture:

-Maps API:

- Provides event location data and transport directions.

-Email API:

- Sends notifications to users.

-OAuth Providers:

- Manage secure login and authentication via Google or Facebook

Non-Functional Considerations

-Scalability

- Equipped with an optimized database design, it can accommodate up to 10,000 concurrent users.
- For tables with heavy demand, such as Subscriptions and Feedback, partitioning or sharding is used.

-Security

- Encrypt sensitive fields (e.g., passwords, email addresses).
- Make sure users can only access their data by implementing role-based access control, or RBAC.

-Performance

- For frequently asked questions, such as event searches, use database indexing.
- Put caching in place for static information, such as event details or banners.

-Backup and Recovery

- Daily automated backups for all tables.
- Versioned backups for important information, such as subscriptions and event specifics.

Example Outputs

-ER Diagram

A high-level diagram showing entities and their relationships:

- Users → Subscriptions → Events → Feedback → Notifications.

-Data Flow Diagram (DFD)

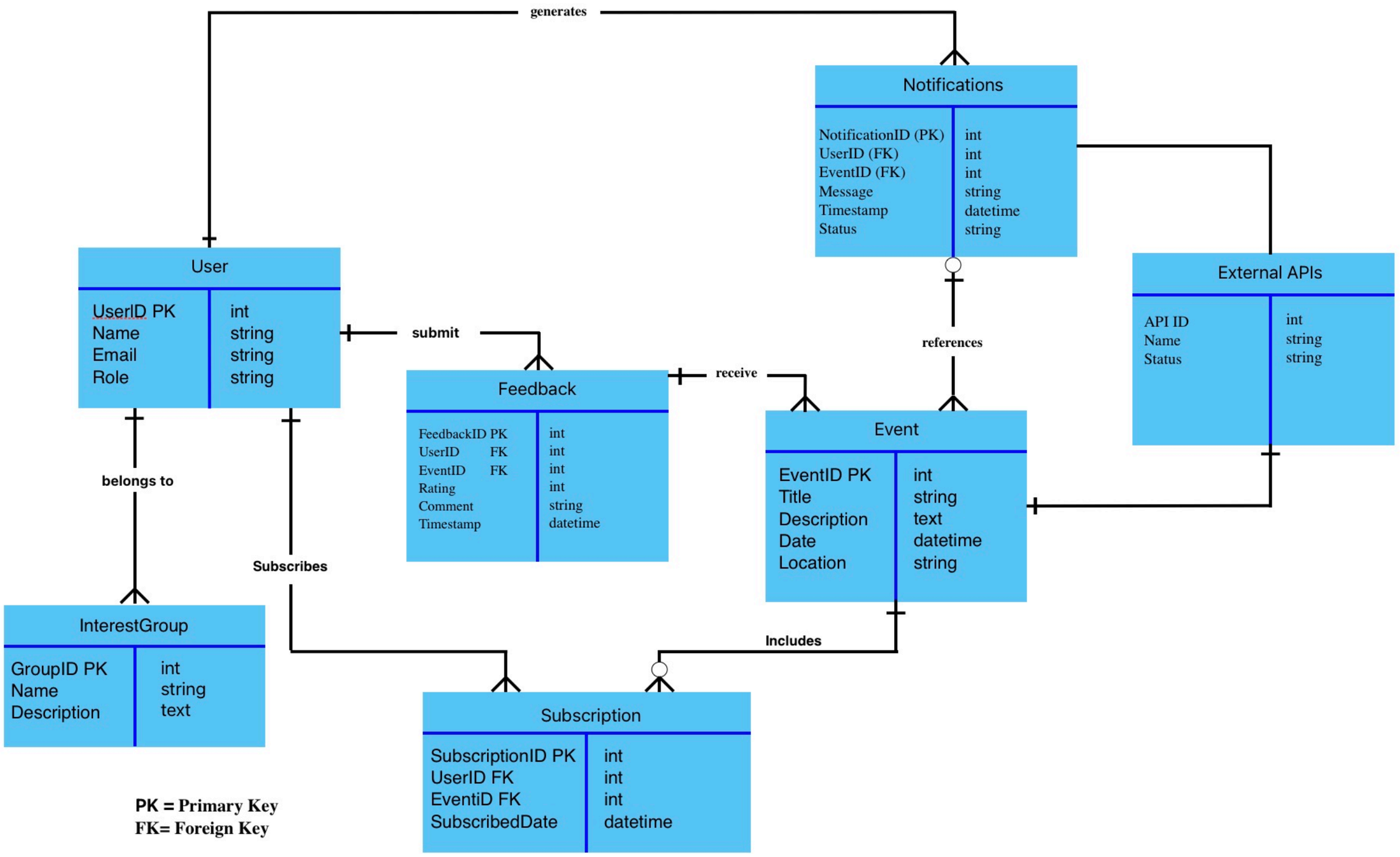
Illustrate how data flows across processes like registration, subscription, and feedback submission.

Database Schema

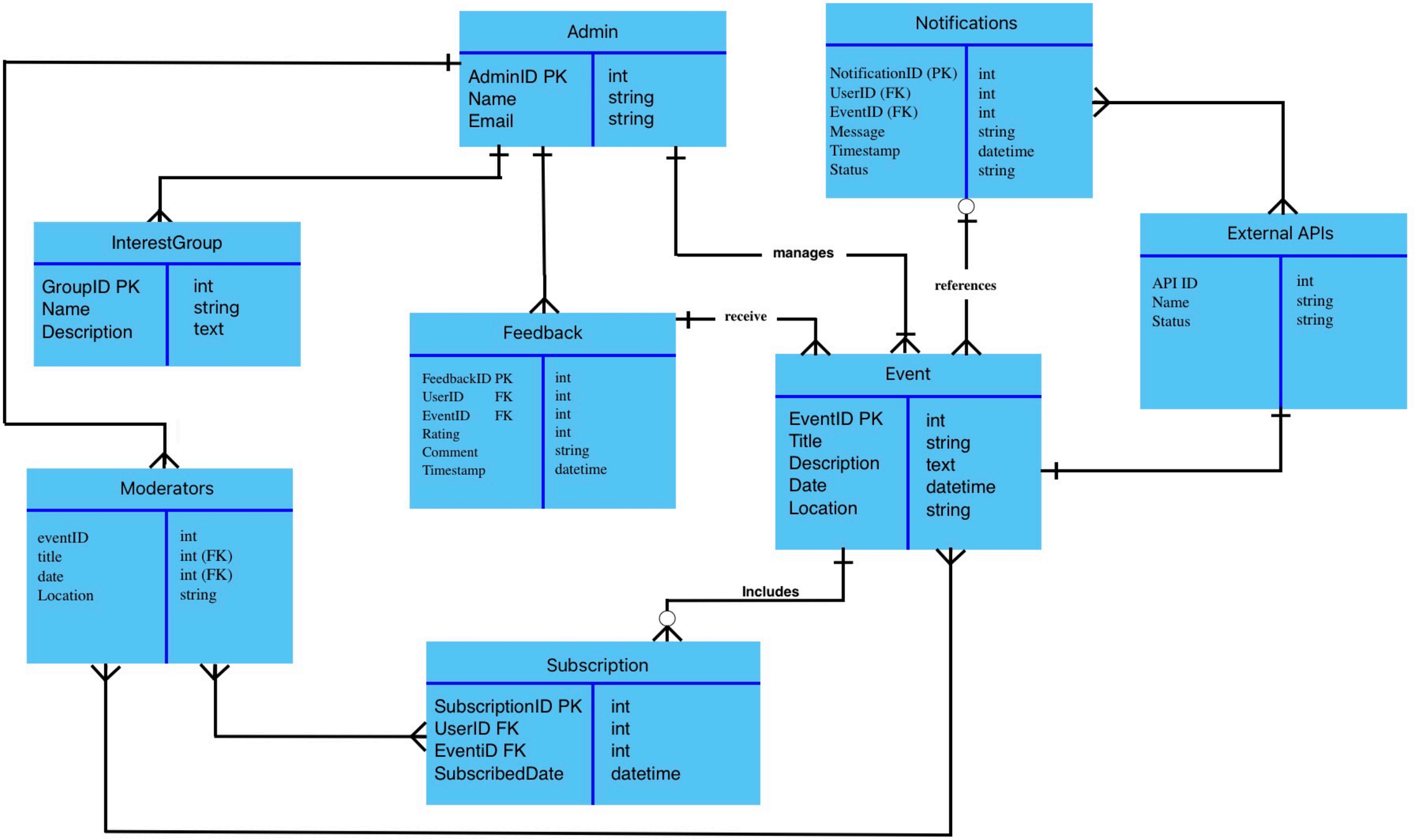
Table	Fields
User	UserID (PK), Name, Email, Role, Password
Event	EventID (PK), Title, Date, Location, OrganiserID (FK)
Subscription	SubscriptionID (PK), UserID (FK), EventID (FK), Timestamp
Feedback	FeedbackID (PK), UserID (FK), EventID (FK), Rating, Comment
Notification	NotificationID (PK), EventID (FK), UserID (FK), Message

Clarity in functionality and alignment with business objectives are ensured by requirement modelling, which aids in the visualization and structuring of the system's requirements. The models utilized to depict and arrange the multi-campus event management system's requirements are shown below.

3g.User ERD Diagram



Admin ERD Diagram



4a. Architecture Design

The goal is to create a comprehensive system architecture that combines databases, front-end, back-end, APIs, and infrastructure to make the system scalable, safe, and easy to use.

Presentation Layer (Frontend)

-Purpose: User-facing interface for interaction with the system.

-Technologies:

- ReactJS: For building dynamic and responsive interfaces.
- HTML5/CSS3: For structure and design.

-Components:

- User Portal: Event browsing, subscriptions, and feedback submission.
- Admin Dashboard: Event management, role assignment, and feedback review.

Business Logic Layer (Backend)

-Purpose: Processes user requests, handles business logic, and manages data interactions.

-Technologies:

- Django: A Python framework for scalable, modular development.
- RESTful API Services: Facilitates communication between frontend, backend, and external services.

-Key Functions:

- User authentication and authorization.
- Event management workflows.
- Notification generation and dispatch.

Data Layer (Database and Storage)

Purpose: Securely store and manage data for users, events, subscriptions, and feedback.

-Technologies:

- MySQL/PostgreSQL: For relational data storage (e.g., Users, Events, Subscriptions).
- AWS S3: For storing multimedia assets (e.g., event banners).

-Features:

- Indexing for fast event searches.
- Automated backups to prevent data loss.

Integration Layer (External APIs)

-Purpose: Connect the system to external services for extended functionality.

-APIs:

- Google Maps API: Provides transport information and event locations.
- Email API: Sends notifications to users.
- OAuth API: Enables secure login via Google orFacebook.

Infrastructure Layer

-Purpose: Supports hosting, scalability, and secure deployment of the system.

-Technologies**:**

- **AWS EC2**: Scalable cloud servers for hosting the application.
- Cloudflare CDN: Accelerates content delivery and reduces latency.
- Docker: Containerization for consistent development and deployment.
- Jenkins**:** Continuous integration and deployment pipeline.

Development Process

-Agile Methodology:

- Features like event creation, subscription, and feedback are developed using iterative sprints.
- Stakeholders should be reviewed frequently to make sure requirements are being met.

-Version Control:

- Use Git for source code management, with branches for features, testing, and production

Security Considerations

-Data Security:

- SSL/TLS encryption for all data transmission.
- Hashing user passwords before storing them.

-Application Security:

- Two-factor authentication for login.
- Role-based access control for sensitive data and admin actions.

-Infrastructure Security:

- Use firewalls and automated monitoring to prevent DDoS attacks.
- Regular security audits and penetration testing.

Scalability and Performance

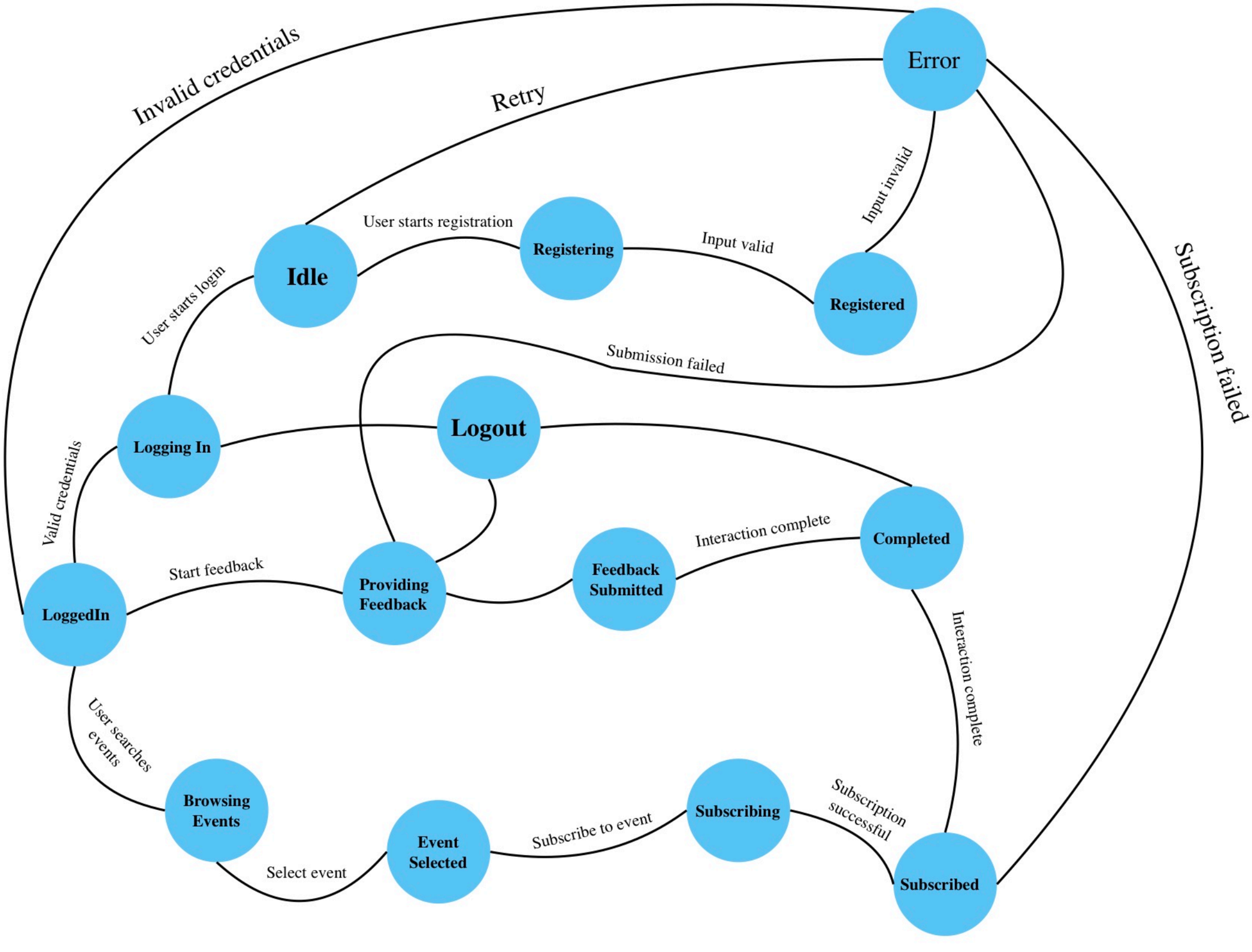
-Load balancing: Use an AWS load balancer to divide traffic evenly among several servers.
-Auto-Scaling: When demand is high, resources are automatically provisioned.
-Caching: For data that is often requested (such as event listings), use Redis or Memcached.

Deployment Pipeline

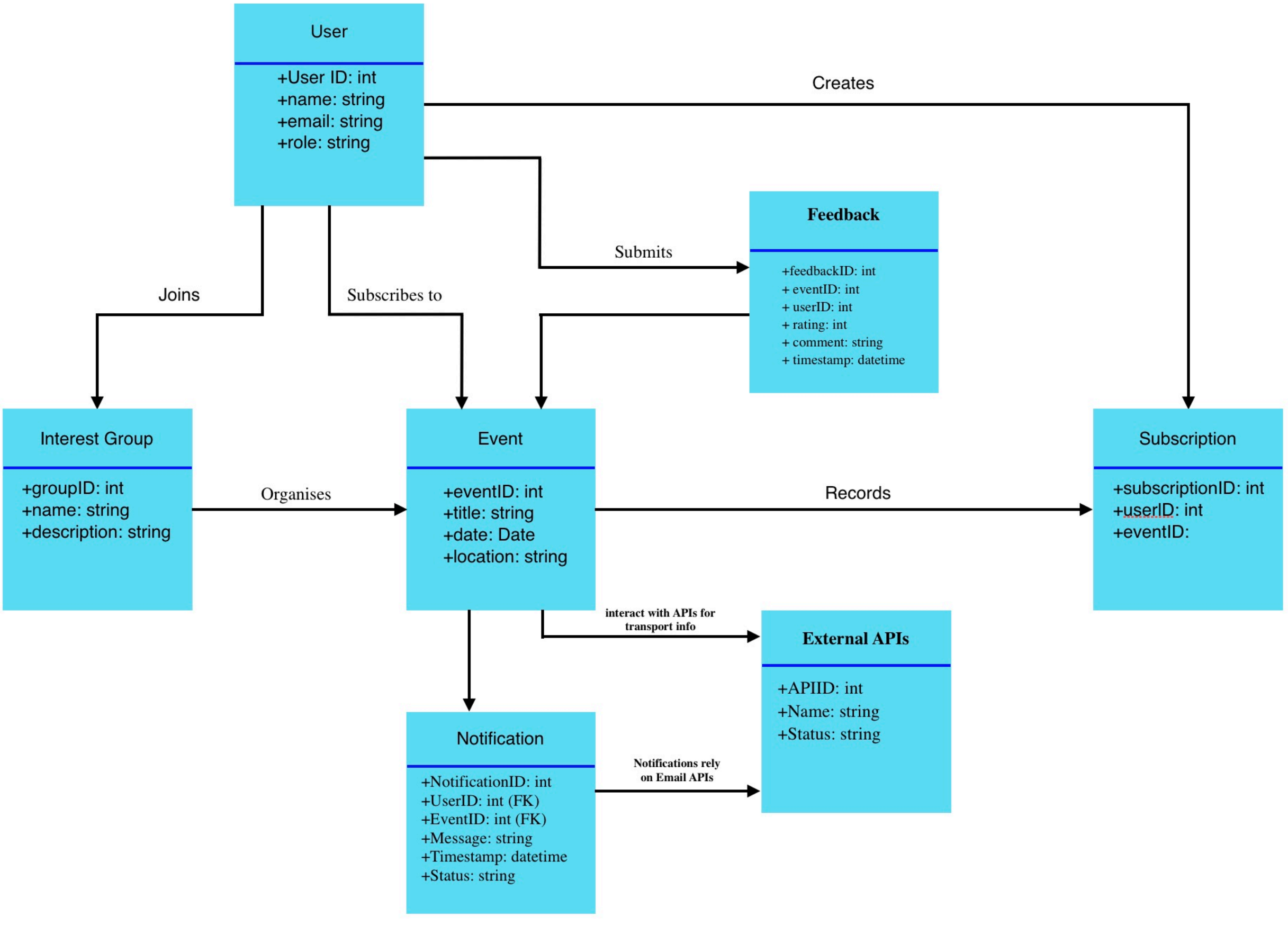
-Continuous Integration/Continuous Deployment (CI/CD):

- Code Integration: Jenkins pipeline triggers builds on every commit.
- Automated Testing: Run unit tests, integration tests, and security scans.
- Staging Environment: Deploy to a staging server for final testing.
- Production Deployment: Deploy verified builds to the live environment with rollback support.

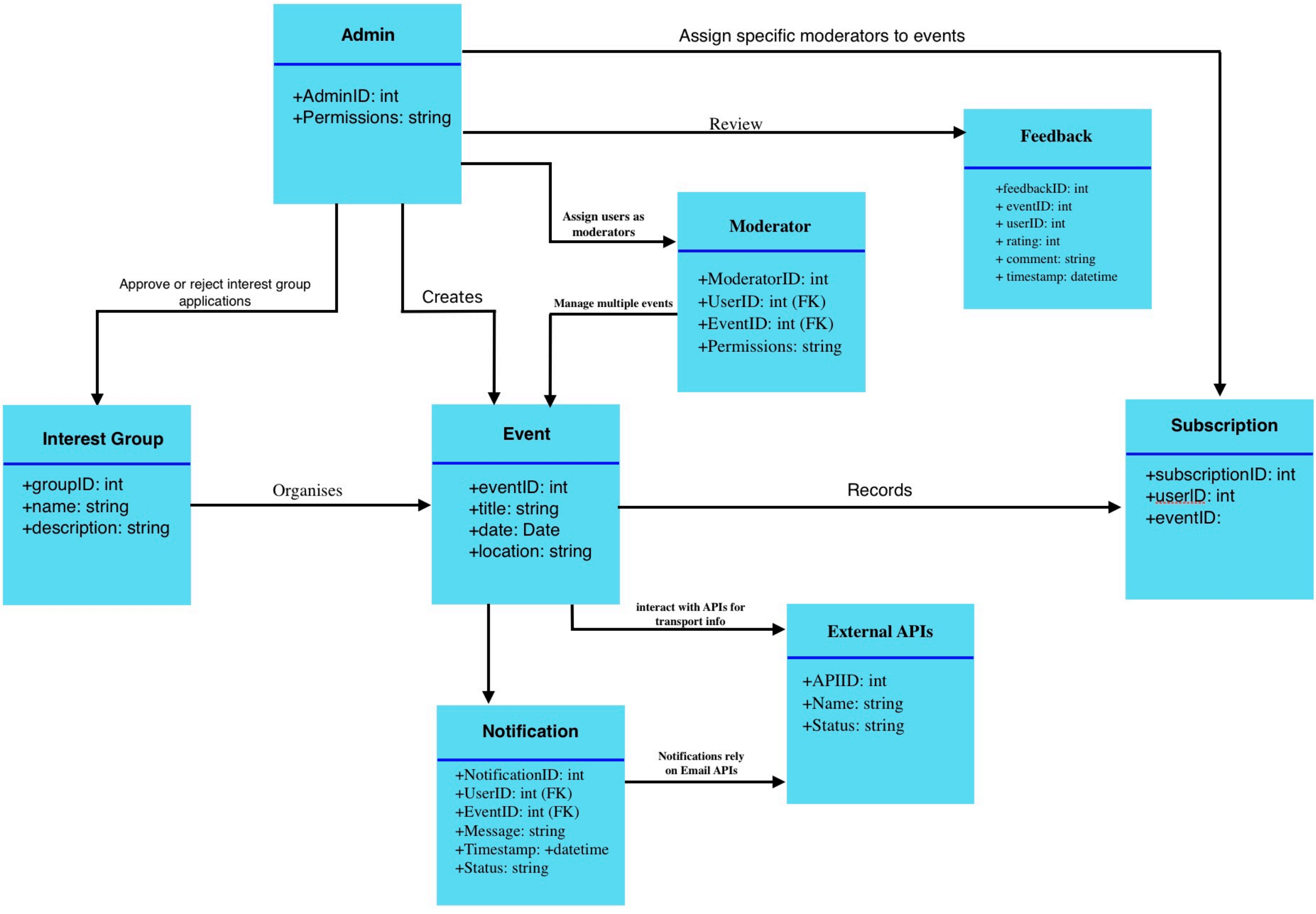
4b.FSM Diagram



4c.User UML Diagram



Admin UML Diagram



Acting

4d. CICD Pipeline

Continuous Integration and Continuous Deployment (CICD) Process

The creation, testing, and deployment of the multi-campus academic event listing platform are all made possible by the CICD pipeline. It supports automated monitoring, high-quality code, and regular updates.

Benefits of CICD:

- Quicker Delivery: System alignment with user requirements is maintained through frequent updates.
- Improved Quality: Automated testing guarantees stability and minimizes errors.
  Better Collaboration: There are fewer disputes when developers collaborate on shared repositories.
- Resilience: Downtime risks are decreased by automated rollback and monitoring.

-Key Stages in the CICD Pipeline

- **Code Integration**: Code is pushed to a shared repository by developers. With the aid of programs like Jenkins or GitHub Actions, automated builds are started for every commit.
- **Automated Testing**: Automatically carry out functional, integration, and unit tests. PyTest and Selenium are two tools that provide thorough testing coverage.
- **Build and Artifact Creation**: Build deployable artifacts like Docker containers and validate build.
- **Staging Deployment**: Before going into production, deploy to a staging environment for last-minute verification.
- **Production Deployment**: Automatically deploy verified builds to production with fallback options in case of problems.
- **Monitoring and Feedback**: Keep an eye on system faults and performance by using Grafana or Prometheus. Create alerts for irregularities to keep the system in good working order.

4e. Testing Plan and Quality Assurance

The testing strategy guarantees that the system satisfies both functional and non-functional requirements and operates as intended. Several testing approaches are included in the plan.
Functional Testing:

- Test individual features to ensure they work as intended.
- Key areas:
    - User registration and login.
    - Event creation and management.
    - Subscription and notification system.
    - Integration with external APIs (e.g., Google Maps).
- Tools: Selenium, Postman (for API testing).

Non-Functional Testing:

- **Performance Testing**: Ensure the system can handle 10,000+ concurrent users.
    - Tools: JMeter, Apache Bench.
- **Load Testing**: Test system behavior under peak usage.
- **Security Testing**: Verify user authentication, role-based access control, and input validation to prevent unauthorized access and vulnerabilities like SQL injection.
- **Usability Testing**: Make sure the system is simple to use and intuitive and collect user feedback by conducting surveys.
- **Integration Testing**: Validate data flow and interactions between components.
- **Regression Testing**: After updates, test the system to make sure the new features don't interfere with already-existing functionality. Test scripts that run automatically guarantee coverage.
- **Acceptance Testing**: Stakeholders' final verification that the system satisfies business needs.
- API Layer

The API layer is used as a bridge between the customer and the backend. It handles business logic, processes data requests, and communicates with the database or other services.

API will handle

- Validating user inputs (e.g. checking university email addresses).
- Handle, retrieve and display events, subscriptions, plans for interest groups.
- Provide location link for Translink.
- Provide location link for Google maps

Endpoints

The API will certain keywords called endpoints to take action.

- **GET&POST / Event** – this will retrieve a list of events
- **POST / Signup** – validation for signing up.
- **POST&DELETE / Subscription** – This is to subscribe to events
- **GET / Translink** – This is to retrieve the location link
- **GET / Google maps** – This is to retrieve the location link
- **POST / Verify** – This is to verify new details (User email especially)

Requirements

Functional Requirements**:**

- Ability to fetch and filter events based on location, date, and user preferences.
- Support for creating and managing recurring events.
- Integration with external services like Google Maps and Translink.

Non-Functional Requirements**:**

- High performance, to handle multiple users across campuses.
- Scalability, to accommodate growing numbers of users and events.
- Security, to protect user data and prevent security breaches.

Tools and frameworks for implementing the API layer:

- **Backend Frameworks**: Node.js (Express)
- **Database**: MongoDB for storing event and user data.
- **API Testing**: Postman for testing endpoints.
- **API Documentation**: Swagger for documenting the API.

Connection to the UI

How will the API connect to the User Interface:

- Use RESTful APIs for efficient data transfer.
- API returns data in a format easily consumed by the frontend, such as a JSON.
- Testing with simple example, such as fetching events and displaying them on the UI

Features	AP
User Registration	**Endpoints**: POST/signup (accepts user details, validates email, and stores in DB), POST /verify-email (sends verification link). **Explanation**: Verifies email format, sends verification email, and stores hashed passwords securely.
Event Management	**Endpoints**: POST /events (create new events), PUT /events/{id} (update event details), DELETE /events/{id} (delete events). **Explanation** Validates user permissions (admin-only actions), ensures event details (date, time, location) are accurate before storing in the database.
Subscription	**Endpoints**: POST /subscriptions (add subscription), DELETE /subscriptions/{id} (remove subscription). **Explanation** Links users to interest groups and sends notifications for new or updated events.
Event addresses	**Endpoints**: GET /events/{id}/location (returns campus address and transport links). **Explanation** Dynamically generates Google Maps and Translink URLs based on event location and user's address.
Event Rating and Feedback	**Endpoints**: POST /events/{id}/feedback (submit feedback), GET /events/{id}/feedback (fetch ratings and reviews). **Explanation** Validates user input, calculates average ratings, and stores user feedback in the database.
Advanced Search and Filtering	**Endpoints**: GET /events **Explanation** Parses filters, queries the database, and returns matching results.
Personalised Recommendations (notifications)	**Endpoints**: GET /recommendations (fetch personalized event suggestions). **Explanation** Uses machine learning or rule-based algorithms to analyse user preferences (e.g., past subscriptions, ratings) and suggests relevant events.

4f. Non-Functional Requirements Table

How the API integrates with the existing functional requirements

Requirements	API
Scalability	**Design**: Use a scalable backend (e.g., load-balanced servers, database sharding). **API Features**: Implement rate limiting and caching for frequently accessed data (e.g., event listings).
Security	**Design**: Encrypt sensitive data. Use HTTPS for secure communication. **API Features**: Implement token-based authentication.
Usability	**API Design:** Provide clean and intuitive endpoints (e.g., /events, /subscriptions). **API Features:** Return helpful error messages and use consistent response formats (e.g., JSON)
Performance	**Design**: Optimize database queries and use caching (e.g., Redis) for frequently accessed data. **API Features**: Ensure endpoints like GET /events are fast and lightweight.
Localisation	**Design**: APIs should accept a locale parameter to return localized content. **API Features**: Include date/time formats specific to the user's region.
Maintainability	**Design**: Follow RESTful principles with modular, reusable endpoints. **API Features**: Maintain comprehensive API documentation, implement clear error messages, and ensure backward compatibility for updates.

Functional Checklist for Quality Testing

Function	Expected Behavior	Pass/ Fail Criteria
User registration	Users can create an account with valid details (e.g., email, password).	Pass if account is created successfully.
User login	Registered users can log in with valid credentials.	Pass if the system grants access to valid users.
Event creation	Admins can create and save event details (title, date, description).	Pass if the event appears in the event listing.
Event Subscription	Users can subscribe to events they are interested in.	Pass if the user receives a confirmation.
Notifications	Users receive timely notifications about subscribed events and updates.	Pass if users receive notifications correctly.
Search Functionality	Users can search for events by title, topic, or date.	Pass if search results are accurate.
API Integration (Travel)	Fetch travel and location info using external APIs like Google Maps.	Pass if valid travel data is retrieved.
Feedback Submission	Users can submit feedback after attending events.	Pass if feedback is stored in the database.
Database Operations	System stores and retrieves user and event data correctly.	Pass if data operations are accurate.
System Performance	The system responds to user actions within 2 seconds.	Pass if the system meets performance criteria.

4g. Risk Assessment Framework

Risk	Likelihood (1–5)	Impact (1–5)	Risk Score (Likelihood × Impact)	Priority Level
API Integration Failure	3	4	12	High
System Downtime	4	5	20	Critical
User Adoption	2	3	6	Medium
Training Requirements	3	2	6	Medium
Data Breaches	5	5	25	Critical
Phishing and Fake Profiles	4	4	16	High
GDPR Infractions	3	5	15	High
Accessibility Standards	2	3	6	Medium
Delivery Delays	4	4	16	High
Budget Overruns	3	3	9	Medium

Risk Management

Technical Risks:

- API Integration Failure: Travel information may be affected by problems with the Translink or Google Maps APIs.
- System downtime: Sudden crashes brought on by heavy user traffic or server malfunctions.
- Data Loss: Important user or event data may be lost as a result of system failures or cyberattacks.
- Scalability Challenges: As usage increases, the platform may have trouble managing higher user traffic.
- Versioning Conflicts: Compatibility problems may arise when third-party APIs (like Google Maps) are updated.

Operational Risks:

- User Adoption: Opposition from users who are not familiar with the system.
- Training Requirements: In order to oversee the system, administrators and moderators might need more instruction.
- Event Management Errors: The credibility of the platform may be impacted by mistakes or delays made by administrators when generating or authorizing events.
- Communication breakdowns: Inadequate notification transmission or poor user-organiser communication.

Security Risks:

- Data Breaches: Unauthorized access to private user information.
- Phishing and Fake Profiles: Malevolent people trying to pose as affiliations of universities.
- Insider threats: when members of an internal team access or misuse data without authorization.
- Weak Passwords: People might make weak passwords, which raises the possibility of unwanted access.
- Distributed Denial of Service (DDoS) Attacks: Malevolent actors may try to prevent users from accessing the system.

Legal and Compliance Risks:

- GDPR Infractions: failure to adhere to data protection laws.
  Standards for Accessibility: Noncompliance with WCAG for users with disabilities.
- Event Liability: There may be fines or penalties for breaking local event-hosting laws.
  Intellectual property violations can lead to legal action if third-party content (such as text or photos) is used without permission.

Development Risks:

- Delivery Delays: Unexpected hold-ups in the development stages or a lack of resources.
- Budget Overruns: Expenses that are more than anticipated because of extra tools or integrations.
- Resource Limitations: Some improvements may be delayed due to a lack of access to qualified developers or tools.
- Requirement Changes: Modifications to the project's scope in the middle may make it more complicated and cause delays.
- Testing Deficiencies: Inadequate testing may lead to performance problems or defects that go unnoticed.

Risk Mitigation Strategies

Technical Risks:

- Implement fallback mechanisms for API failures (for example: cached data or alternate services).
- To manage spikes in traffic, use scalable cloud infrastructure.

Operational Risks:

- Organize administrator training sessions and user onboarding. To increase adoption, create FAQs and user-friendly guidelines.

Security Risks:

- Use strong authentication techniques, such as two-factor authentication. Perform audits and penetration tests on a regular basis.

Legal and Compliance Risks:

- To guarantee GDPR compliance, choose a compliance officer.
  Consider accessibility while designing interfaces and examine them on a regular basis.

Learning

Development Risks:

- To guarantee iterative development and early problem identification, apply Agile approaches.
- Set aside money in your budget for unforeseen costs.

Risk Mitigation Measurement

Risk	Mitigation Strategy	Measurement Criteria
API Integration Failure	Implement fallback mechanisms and use cached data	Percentage of API downtime managed by fallback
System Downtime	Use scalable cloud infrastructure	Number of traffic spikes handled without downtime
Data Breaches	Use two-factor authentication and regular audits	Number of audit findings and reported breaches
Phishing and Fake Profiles	Use two-factor authentication and regular audits	Number of audit findings and reported breaches
Phishing and Fake Profiles	Use two-factor authentication and regular audits.	Percentage of GDPR-related compliance checks passed
Delivery Delays	Use two-factor authentication and regular audits.	Percentage of GDPR-related compliance checks passed

Risk Monitoring and Review

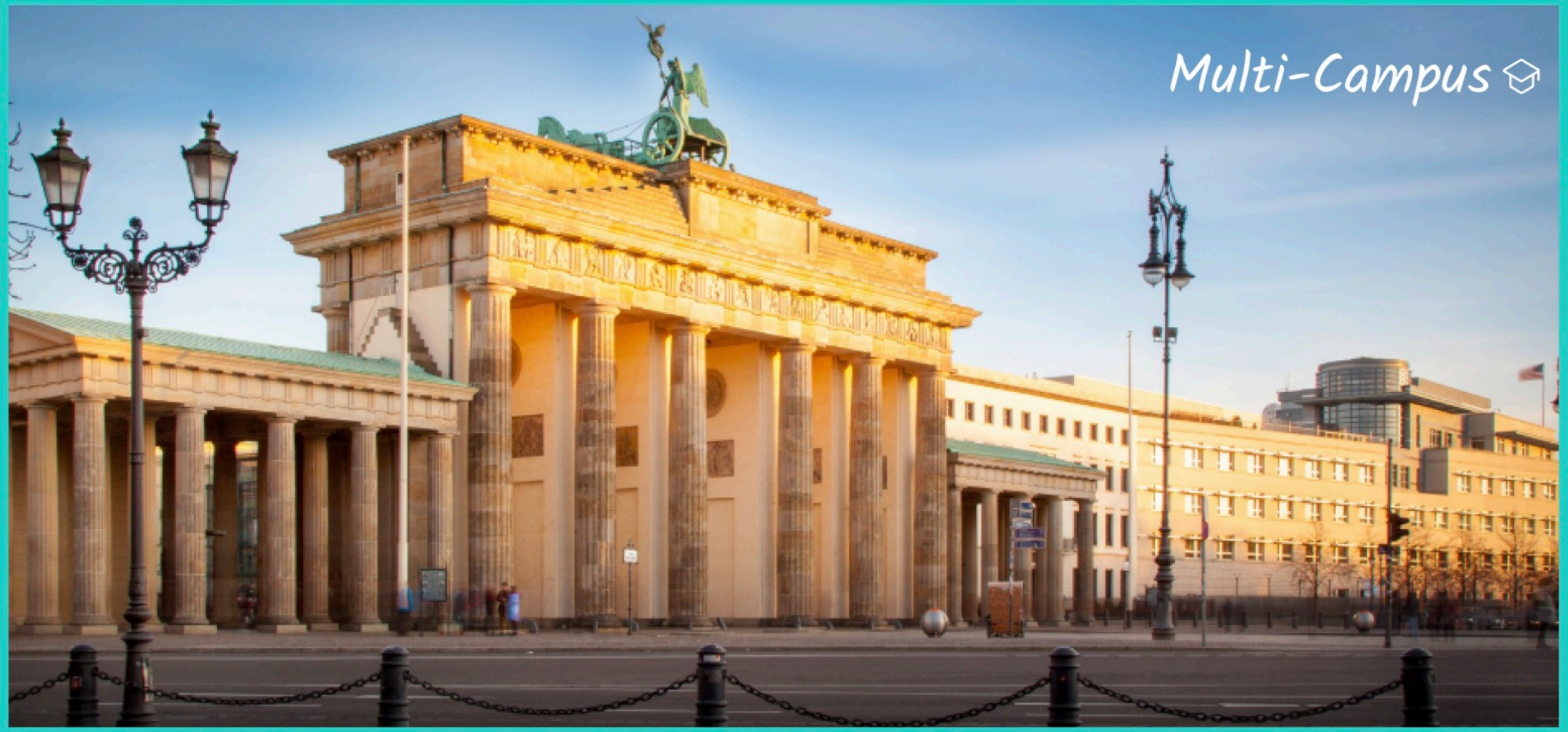
Review and revise the risk management strategy on a regular basis in light of user input and project progress.

- To keep track of risks and related activities, use Jira or Trello.
- Review Frequency**:** Conduct risk reviews monthly or during key project milestones.
- Responsibility**:** Assign risk ownership to team members (e.g., Compliance Officer for GDPR risks, DevOps for API issues)

Search events

Sign up / Login

Find My tickets | Events | Universities | Locations



Contact us | Newsletter | FAQs | About Us

Menu

Search events

Sign up / Login

Languages

Help

ickets | Events | Universities | Locations

ntact us

Newsletter

FAQs

About Us

Search events

Sign up / Login

Find My tickets | Events | Universities | Locations

Login

Username or email:

password:

Contact us

Newsletter

FAQs

About Us

Search events

Sign up / Login

Find My tickets | Events | Universities | Locations

Create an account

Name:

Surname:

Date of Birth:

University:

Program [select]:

Semester:

University Email:

Password:

Repeat Password:

Search events

Sign up / Login

Find My tickets | Events | Universities | Locations

Events

Contact us | Newsletter | FAQs | About Us

Search events

Log Out

Find My tickets | Events | Universities | Locations

Event X

Date: DD/MM/YYYY

Speaker: Mr. Y

Location: { Event address + Location's link }

I'll Reserve

Contact us | Newsletter | FAQs | About Us

Search events

Log Out

Find My tickets | Events | Universities | Locations

Spot Reservation

Full Name:

University email:

Phone number (optional):

Book

Contact us | Newsletter | FAQs | About Us

Search events

Log Out

Find My tickets | Events | Universities | Locations

Feedback

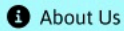
Location	☆ ☆ ☆ ☆ ☆
Speaker	☆ ☆ ☆ ☆ ☆
Content	☆ ☆ ☆ ☆ ☆
Meeting room cleanliness	☆ ☆ ☆ ☆ ☆
Timing	☆ ☆ ☆ ☆ ☆

Submit

 Contact us

 Newsletter

 FAQs

 About Us

Search events

Log Out

Find My tickets | Events | Universities | Locations

Subscriptions

Notifications

Enter Topic or speaker

Add

Contact us

Newsletter

FAQs

About Us

Login

Username or email:

password:

Search events

Log Out

Find My tickets | Events | Universities | Locations

Manage events

Create event

Edit existing event

Remove event

Contact us

Newsletter

FAQs

About Us

Search events

Log Out

Find My tickets | Events | Universities | Locations

Create events

Event Title:

Event Date:

Event description:

Event address:

Upload event poster

Create

Contact us | Newsletter | FAQs | About Us

Search events

Log Out

Find My tickets | Events | Universities | Locations

Edit Event

Event Title: Event X

Event Date: DD/MM/YYYY

Event description: TEXT

Event address: TEXT + Link

Confirm

Contact us | Newsletter | FAQs | About Us

Search events

Log Out

Find My tickets | Events | Universities | Locations

Remove Event

Event Title:	Event X
Event Date:	DD/MM/YYYY
Event description:	TEXT
Event address:	TEXT + Link

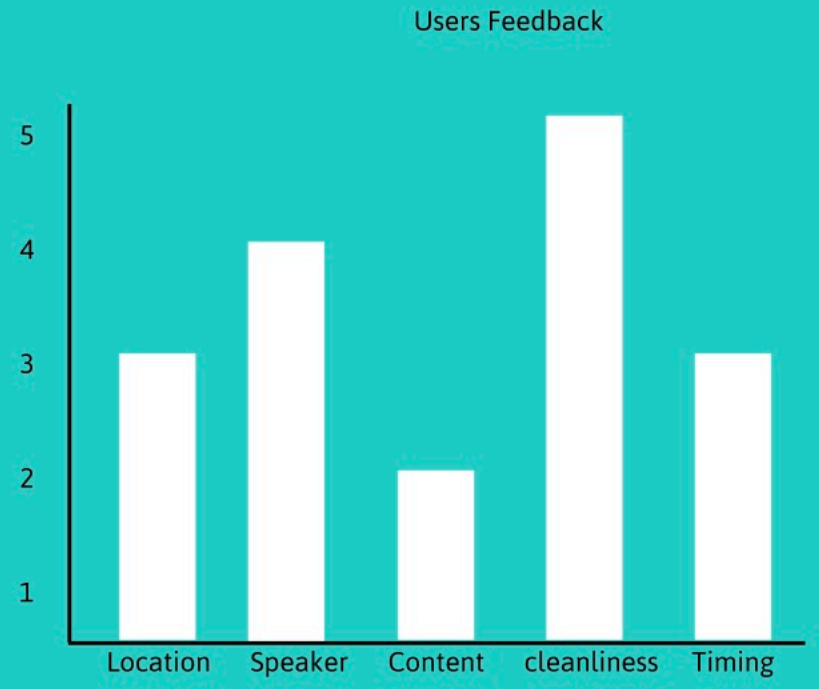
Remove

Contact us | Newsletter | FAQs | About Us

Search events

Log Out

Find My tickets | Events | Universities | Locations



Feedback of Event X

Date: { Date of event}

Contact us | Newsletter | FAQs | About Us