**Software Requirements Specification (SRS)**

**Darren Ross**

**Anthony Jack**

**Manikala Chevitipalli**

**Sai Achyuth Konda**

**GitHub Wiki:** <https://github.com/djgamekid/GDP-Group-I-bearcatmanager/wiki/Software-Requirements-Specification-(SRS)>

**Project Information**

**Project Charter**

*Summary of the problem:*

**Campus event organizers struggle to efficiently manage event creation, ticket sales, and attendee management, leading to disorganized events and poor attendee experiences.**

*Motivation for solving the problem (How this improves the client's workflow):*

**Implementing a comprehensive event management platform will streamline event organization, enhance attendee experience, save time with automated processes, provide valuable insights through real-time analytics, ensure data security and compliance, and support scalability for larger events. This will significantly improve the efficiency and effectiveness of campus event management.**

*A brief rundown of the required functionality:*

* **Event creation and management tools.**
* **Ticketing system with QR code generation.**
* **Attendee check-in via mobile devices through website.**
* **Notifications and reminders for event attendees.**
* **Real-time updates for attendee changes.**
* **Event dashboard with analytics.**
* **Ticket purchase interface with multiple payment options and options for student payments.**
* **API for ticketing and attendee management.**
* **Real-time data sync for check-in systems.**
* **Secure ticket storage and validation.**
* **GDPR compliance for attendee data.**

**About the Developers**

## Our Developers:

**Darren Ross**

GitHub Username: **djgamekid**  
Find my projects and repositories here: [Darren Ross](https://github.com/djgamekid)

Interest in Computer Science:

* Web Applications
* Backend Databases
* Mobile Applications
* IoT Devices
* AI

**Manikala Chevitipalli**

GitHub Username: **cmanikala**  
Follow me here: [Manikala](https://github.com/cmanikala)

Interests in Computing:

* Databases (SQL, MongoDB); Object-oriented programming languages
* Developing web applications using JavaScript, HTML, and CSS
* Mobile applications (iOS); Data structures
* AI & Web technologies

**Sai Achyuth Konda**

GitHub Username: **saiachyuth-pc**  
Link to GitHub page: [Sai Achyuth Konda](https://github.com/saiachyuth-pc)

Listing of strengths and interests in computing:

* Performance Testing Tools (Load Runner, Akamai, Dynatrace, Jenkins, Soap UI, Pinpoint)
* Database Management Systems (SQL); Object-Oriented Programming using Java
* Developing web applications using HTML, CSS, JavaScript
* Java, Python, AI

**Anthony Jack**

My GitHub Username: **DopeAnt25**  
Find my projects and repositories here: [Anthony Jack](https://github.com/DopeAnt25)

Interest in Computing:

* Front-end development
* Mobile development (iOS & Android)
* Game development / C++
* Theory and Implementation

**Problem Statement**

***The Problem***

*The current campus event system has inefficiencies for both users and administrators, particularly in ticket viewing, purchasing, and attendance verification. Organizers face challenges navigating multiple websites for event creation, including booking rooms and managing event details across various internal systems.*

*There is a lack of cohesive analytics for organizers, limiting their ability to track event performance and understand attendance demographics, especially for non-students. Students struggle to find and purchase tickets due to limited event visibility, while check-in processes using student IDs are inefficient and cause delays. Non-student attendees miss out on reminders and updates, creating unequal access to event information compared to students and potentially affecting participation.*

**Design**

**Functional Requirements**

***Bearcat Manager – A Web application that hopes to create an application that can be used to create and manage events and allow users to get tickets to events.***

***The system SHALL***

1. The system SHALL allow the user to get a ticket for an event.

2. The system SHALL allow the admin to create an event.

3. The system SHALL allow the user to search for created events.

4. The system SHALL allow the admin to delete events.

5. The system SHALL allow the admin to edit events currently created.

6. The system SHALL accept user credentials.

7. The system SHALL allow for account creation using email.

8. The system SHALL have a QR code generated with a ticket.

9. The system SHALL have an attendance manager to check in event attendees.

10. The system SHALL authenticate user credentials.

11. The system SHALL display a dashboard that hosts upcoming events and pre-selected events.

12. The system SHALL notify the user with an email confirmation of the selected event.

***The system SHOULD***

1. The system SHOULD provide some sort of feedback when the user is signing in.

2. The system SHOULD have an on-site notification system after a user logs in.

3. The system SHOULD be easily navigable.

4. The system SHOULD allow users to edit their profiles.

5. The system SHOULD allow participants to cancel the registration, freeing up the spots for others.

6. The system SHOULD be able to filter events based on location and event type.

7. The system SHOULD allow the user to submit feedback to developers.

8. The system SHOULD allow the user to receive help on-site via chatbox.

9. The system SHOULD have an API that controls ticketing to users and attendees.

10. The system SHOULD sync data in real-time to allow for a seamless experience in navigation.

11. The system SHOULD NOT store user credentials in plain text.

***The system MAY***

1. The system MAY play some sort of animation when logging in.

2. The system MAY have a mobile format.

3. The system MAY have pictures associated with each event.

4. The system MAY need to redirect to another website when a user clicks external links.

5. The system MAY have pictures associated with each event location.

6. The system MAY have a trending carousel on the main page.

**Non-Functional Requirements**

***Security***

1. Encryption: All login information will be encrypted in the database, ensuring user credentials are secure and inaccessible to unauthorized personnel.

2. Two-Factor Authentication (2FA): Email-based 2FA will be implemented to provide an additional layer of security, verifying user identity through a secondary method.

3. Access Control: The database will have restricted permissions, ensuring only authorized users can access sensitive information.

***Scalability***

1. Event Archiving: The system will maintain an archive of past events, enabling easy reintroduction of recurring events.

2. User Notification: The system will be capable of notifying all users about new potential events, accommodating an increasing number of users without performance degradation.

***Usability***

1. Simplistic Structure: The website's design will be straightforward and user-friendly, allowing easy navigation across different pages.

2. Interactive Design: Users will experience fluid navigation through the site, aided by interactive buttons and small animations.

**Performance**

1. Real-Time Synchronization: The system will synchronize data in real-time, ensuring that any changes made by users or admins are immediately reflected.

2. Concurrent Handling: The site will be optimized to handle multiple user requests simultaneously, ensuring smooth performance under load.

***Interoperability***

1. Data Format Support: The system will support common data formats (e.g., JSON, XML, CSV), facilitating seamless data exchange with other systems.

2. RESTful APIs: APIs will adhere to RESTful standards, ensuring efficient interaction with external systems.

3. Secure Authentication: The system will support OAuth 2.0 or equivalent secure authentication methods for safe integration with third-party systems.

***Portability***

1. Platform Independence: The system will be designed to run on various operating systems, including Windows, Linux, and macOS.

2. Deployment Flexibility: The system will be deployable across different environments, such as on-premises servers, cloud platforms (e.g., AWS, Azure), and hybrid cloud environments.

3. Containerization: Deployment via containerized solutions (e.g., Docker) will be supported to ensure consistency across development, testing, and production environments.

***Stability***

1. Error Handling: The system will include robust error handling to maintain stability and prevent crashes during unexpected situations.

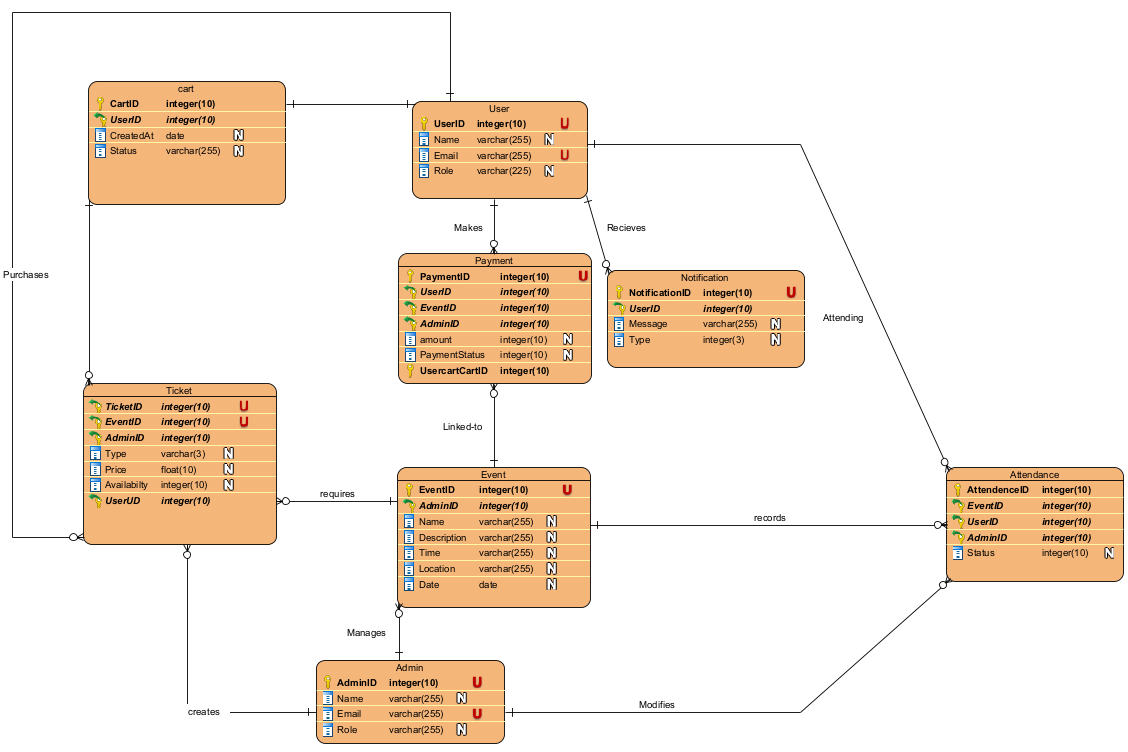
***Maintainability***

1. Code Documentation: Comprehensive documentation will be maintained for the codebase, facilitating easier updates and modifications.

2. Modular Design: The system will have a modular design, allowing individual components to be updated or replaced without affecting the entire system.

**Data Management Plan**

***Data Management Overview***

**

***Summary of Data to be Stored.***

***1. Admin Information***

*Admin ID: A unique identifier for each admin.*

*Name: The name of the admin.*

*Email: The email address of the admin.*

*Role: The role of the admin within the system.*

***2. User Information***

*User ID: A unique identifier for each user (student or staff).*

*Name: The name of the user.*

*Email: The email address of the user.*

*Role: The role of the user (either Student or Staff).*

***3. Event Details***

*Event ID: A unique identifier for each event.*

*Admin ID: The identifier for the admin managing the event.*

*Name: The name of the event.*

*Description: A brief description of what the event is about.*

*Time: The time the event starts.*

*Location: Where the event will take place.*

*Date: The date the event is scheduled for.*

***4. Ticket Information***

*Ticket ID: A unique identifier for each ticket type.*

*Admin ID: The identifier for the admin managing the event.*

*Event ID: The identifier for the event associated with the ticket.*

*Type: The type of ticket (e.g., NOM or VIP).*

*Price: The cost of the ticket.*

*Availability: The number of tickets available for purchase.*

***5. Attendance Records***

*Attendance ID: A unique identifier for each attendance record.*

*Event ID: The identifier for the event for which the user is attending.*

*User ID: The identifier for the user attending the event.*

*Status: The status of the user’s attendance (e.g., checked in or not).*

***6. Notification Details***

*Notification ID: A unique identifier for each notification.*

*User ID: The identifier for the user receiving the notification.*

*Message: The content of the notification message.*

*Type: The type of notification (e.g., Email, SMS, or App notification).*

***7. Payment Information***

*Payment ID: A unique identifier for each payment transaction.*

*User ID: The identifier for the user making the payment.*

*Event ID: The identifier for the event associated with the payment.*

*Admin ID: The identifier for the admin managing the event.*

*Amount: The amount paid.*

*Payment Status: The status of the payment (e.g., completed, pending, failed).*

*UsercartCartID: The identifier for the cart associated with this payment transaction, linking the payment to the specific items selected by the user.*

***8. Cart***

*Cart ID: A unique identifier for each cart.*

*User ID: The identifier for the user making the payment.*

*CreatedAt: The timestamp for when the cart was created.*

*Status: Indicates the current status of the cart (e.g., active, pending, checked out, or abandoned).*

***Data Security Plans***

***Initial Plans to Secure Data***

***1. Access Restrictions:***

* *User Roles: Implement role-based access control (RBAC) to ensure that users only have access to data necessary for their role (e.g., Admin, User, Event Organizer).*
* *Admins will have full access to manage events, users, and notifications.*
* *Users (Students/Staff) will have restricted access, allowing them to view and register for events, purchase tickets, and receive notifications.*
* *Authentication: Require strong authentication methods (e.g., password policies, two-factor authentication) to access the system and sensitive data.*

***2. Data Encryption:***

* *At Rest: Use encryption to secure sensitive data stored in the database, such as user emails, payment information, and notifications. AES (Advanced Encryption Standard) is recommended for this purpose.*
* *In Transit: Implement HTTPS to encrypt data transmitted between users and the server, preventing interception during data exchange.*
* *Sensitive Fields: Specifically encrypt sensitive fields like ‘Email’, ‘Payment Amount’, and ‘Notification Message’ to further enhance security.*

***3. Regular Security Audits:***

* *Conduct regular security audits and vulnerability assessments to identify and mitigate potential risks in the system.*

***4. Data Backup and Recovery:***

* *Implement a data backup strategy to ensure that all data is regularly backed up and can be restored in the event of data loss or a security breach.*

***Mapping of Functional Requirements to Data Storage***

***1. User Registration and Management:***

* *Related Data Storage: ‘User’ table*
* *Requirements: Store user information securely and allow admins to manage user accounts.*

***2. Event Management:***

* *Related Data Storage\*\*: ‘Event’ table, ‘Admin’ table*
* *Requirements\*\*: Admins need to create, update, and delete events, associating them with their profiles.*

***3. Ticket Sales:***

* *Related Data Storage: ‘Ticket’ table, ‘Payment’ table*
* *Requirements: Users should be able to purchase tickets, and relevant payment data should be securely stored and processed.*

***4. Attendance Tracking:***

* *Related Data Storage: ‘Attendance’ table*
* *Requirements: Track user attendance at events, enabling event organizers to manage participants effectively.*

***5. Notifications:***

* *Related Data Storage: ‘Notification’ table*
* *Requirements: Notify users about events, updates, and reminders securely without exposing sensitive information.*

***6. Payment Processing:***

* *Related Data Storage: ‘Payment’ table*
* *Requirements: Securely store transaction details, including user IDs and event IDs, to track payments made by users.*

**Proposed Prototypes**

*Aims and Wishes*

* Event Creation and Check-In: Our team aims to create a web application that provides tools for creating events and allows users to check in.
* Database and User Management: The website will include a database to store User & Admin login information and event details, ensuring secure and organized data management.
* User-Friendly Interface: We aim to design the website with a navigable and fluid interface, making it easy for users to browse and register for events.
* Admin Analytics and QR Code Check-In: The application will provide data and analytics to admins to monitor event success. Registered users will receive an email with a QR code for event check-in.

*Prototype*

* A basic website with the ability to create an event on the Admin site and register with an email notification on the User site.
* There will be a create button and a delete button for the Admin to modify events.
* The User will see the events and will be able to register and unregister.
* The Admin page will allow the generation of a ticket for each aforementioned created event and invalidate tickets for users if the event is deleted.
* The User will be able to click on a pre-registered event card to "check-in" where their ticket's ID will be validated.
* Both Admin and Users will log in through a simple login page that will be displayed before the site is shown.
* Users can add multiple events to a cart.

**Use Cases**

***Use Case 1: Ticketing System for Event Management:***

**A diagram of a company

Description automatically generated**

**Primary Actor:** User (Student or Staff)

**Scope:** Event Ticketing

**Brief:** A User (Student or Staff) can purchase or reserve tickets for an event through the system. The system manages ticket availability, processes payments if applicable, and sends a confirmation and digital ticket to the User.

**Stakeholders:** Users (Students and Staff), Admin, University Administration, Payment Processing Service.

**Postconditions:**

* Minimal Guarantees:
  + The system will update the ticket availability for the event.
  + The User will receive a confirmation notification and a digital ticket.
* Success Guarantees:
  + The User successfully purchases or reserves a ticket for the event.
  + The User receives a digital ticket with all necessary event information.

**Preconditions:**

* The system displays a list of events with available tickets.
* The User is logged into the system.
* The User has a valid payment method.

**Triggers:**

* The User clicks on an event they are interested in and selects the option to purchase or reserve a ticket.

**Basic Flow:**

1. *The system presents the User with a list of events that have available tickets.*
2. *The User selects an event and clicks the "Buy Ticket" or "Reserve Ticket" button.*
3. *The system displays the event details and ticket options (e.g., general admission, VIP).*
4. *The User selects the desired ticket option and quantity.*
5. *The system prompts the User to confirm their selection and proceed to payment (if applicable).*
6. *The User provides payment information (if applicable) and confirms the purchase or reservation.*
7. *The system processes the payment (if applicable) and updates the ticket availability.*
8. *The system sends a confirmation notification and a digital ticket to the User.*
9. *The system returns the User to the list of events, where they can see their ticketed events.*

**Extensions:**

* 2–3.
  + Event Filtering and Sorting
    - 1. The system allows the User to filter and sort events by categories such as date, type, or location.
    - 2. The system displays the filtered and sorted list of events based on the User's selection.
* 4-5.
  + Ticket Option Details
    - The system displays detailed information about each ticket option, including benefits and pricing.
    - The User can view and compare different ticket options before making a selection.
* 6-7.
  + Payment Processing
    - The system integrates with a secure payment processing service to handle payments.
    - The system displays a payment confirmation page after successful payment processing.
* 8-9.
  + Notification Preferences
    - The system checks the User's notification preferences (email, SMS, in-app notification) and sends the confirmation and digital ticket accordingly.
    - The system logs the notification and ticket sent to the User for auditing purposes.
* 10-11.
  + View Ticketed Events
    - The system provides a section or filter on the dashboard where Users can view their ticketed events.
    - The system allows the User to cancel their ticket if they can no longer attend (subject to event policy).
* Refund and Cancellation Policy
  + The system displays the refund and cancellation policy to the User before confirming the purchase or reservation.
  + The User can request a refund or cancel their ticket based on the event's policy.

**Use Case 2: *Event Check-In for Registered Users***

A diagram of a flowchart

Description automatically generated

**Primary Actor:** User (Student or Staff)

**Scope:** Event Check-In

**Brief:** A registered User checks into an event on the day of the event. The system verifies the User’s registration, marks them as "checked in," and updates the attendance list. The system can also allow the User to check in via a mobile app or through a Self-service System at the event venue.

**Stakeholders:** Users (Students and Staff), Admin, Event Organizers

**Postconditions:**

* Minimal Guarantees:
  + The system will mark the User as "checked in" for the event.
  + The system provides a real-time attendance report for event organizers, showing all checked-in attendees.
* Success Guarantees:
  + The User is successfully checked into the event and their attendance is recorded.
  + The system logs the attendance data for further analytics and reporting.

**Preconditions:**

* The User is registered for the event.
* The event is active (ongoing or about to start).
* The system has check-in options available (via app or Self-service System).

**Triggers:**

* The User arrives at the event venue and clicks "Check-In" from the mobile app or checks in using a self-service system at the event location.

**Basic Flow:**

1. *The system displays a "Check-In" button for the event on the User's dashboard or in the mobile app.*
2. *The User selects the event they want to check into and clicks "Check-In."*
3. *The system verifies the User’s registration status for the event.*
4. *Upon successful verification, the system marks the User as "checked in."*
5. *The system updates the event's attendance list to reflect the User's check-in.*
6. *The system sends a confirmation notification to the User, confirming the successful check-in.*
7. *The system updates the event organizer's dashboard to show the User's attendance.*

**Extensions:**

* 2–3.
  + Self-service Check-In
    - The User scans their event QR code or taps their ID card at the event.
    - The self-service system displays a confirmation screen, marking the User as "checked in."
* 4-5.
  + Error During Check-In
    - If the User is not registered, the system displays an error message and prompts them to contact event support.
    - The system logs the error for auditing purposes.
* 6-7.
  + Late Check-In
    - The system allows for a grace period after the event start time for late check-ins.
    - If the User checks in after the grace period, the system sends a notification indicating they are late.

**Use Case 3: *Creating Events for Students & Staff***

A diagram of a company event management

Description automatically generated

**Primary Actor:** Admin (Administrative Director)

**Scope:** Admin

**Brief:** The Admin inserts information regarding an event idea that details the idea itself, time, location, and date. This event is then monitored with analytics and a list of accepted invites from students and staff.

**Stakeholders:** Admin, University Administration

**Postconditions:**

* Minimal Guarantees:
  + The Admin will have a monitorable event with notifications sent to students and staff after one signs up.
* Success Guarantees:
  + An event card that is viewable by Users and updatable by the Admin.

**Preconditions:**

* The system lists event spaces available to the Admin

**Triggers:**

* The Admin clicks on an interface button to "Create a new Event"

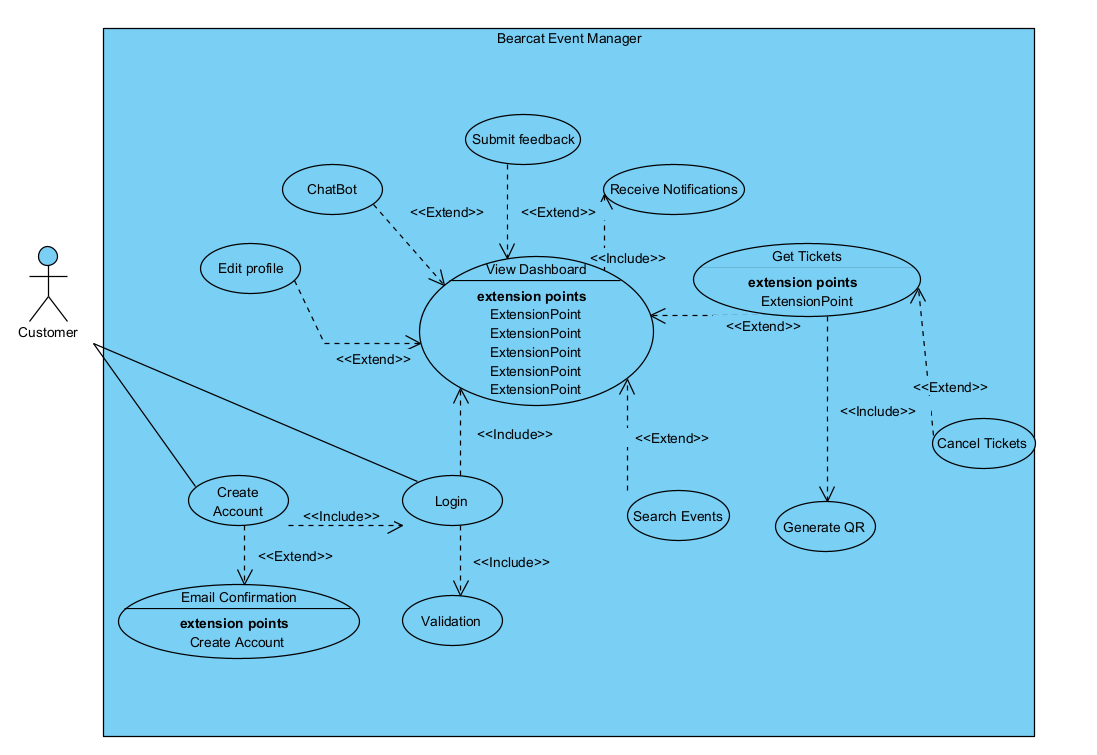
**Basic Flow:**

1. *The system will present a dashboard of analytics regarding any events previously created or a message to create one if none are created.*
2. *The system will have a button interface that links to a form page for the Admin to insert information.*
3. *The system will list a series of fillable tabs along with a generated list of locations to select from.*
4. *The Admin inputs the necessary information and clicks a save which will send the information to be stored and viewed.*
5. *The system returns the Admin to the dashboard with the viewable card that presents the information in a styled manner.*

**Extensions:**

* 1–2.
  + Welcome Message
    - The system has a simple greeting after an Admin login.
* Editable from Dashboard
  + The system displays a button for each created event.
  + The system does not display the aforementioned button if time has passed the set time.
* 4-5.
  + Alert Confirming Input
    - The system provides an alert pop-up ensuring the data entered is correct.
* Event Notification of Success
  + The system sends a notification to the Admin of a successful creation.

**Use Case 4: *A User (Student or Staff) Registers for an Event***



**Primary Actor:** User (Student or Staff)

**Scope:** Event Registration

**Brief:** A User views a list of upcoming events, selects an event they are interested in, and completes the registration process. The system then updates the event's attendee list and sends a confirmation notification to the User.

**Stakeholders:** Users (Students and Staff), Admin, University Administration

**Postconditions:**

* Minimal Guarantees:
  + The system will update the list of attendees for the event.
  + The User will receive a confirmation notification of their registration.
* Success Guarantees:
  + The User successfully registers for the event.
  + The User receives all necessary information about the event (e.g., date, time, location) in the confirmation notification.

**Preconditions:**

* The system displays a list of upcoming events available for registration.
  + The User is logged into the system.

**Triggers:**

* The User clicks on an event they are interested in from the list of upcoming events.

**Basic Flow:**

1. *The system presents the User with a dashboard or list of upcoming events.*
2. *The User selects an event they are interested in by clicking on it.*
3. *The system displays the event details, including date, time, location, and description.*
4. *The User clicks the "Register" button for the event.*
5. *The system prompts the User to confirm their registration.*
6. *The User confirms their registration.*
7. *The system updates the event's attendee list with the User's information.*
8. *The system sends a confirmation notification to the User, containing the event details and a confirmation message.*
9. *The system returns the User to the dashboard or list of events, where they can see their registered events.*

**Extensions:**

* 2–3.
  + Event Filtering
    - The system allows the User to filter events by categories such as date, type, or location.
    - The system displays the filtered list of events based on the User's selection.
* 5-6.
  + Registration Confirmation Alert
    - The system provides a pop-up alert to confirm the User's intent to register for the event.
    - The User has the option to cancel or confirm the registration in the pop-up alert.
* 7-8.
  + Notification Preferences
    - The system checks the User's notification preferences (email, SMS, in-app notification) and sends the confirmation accordingly.
    - The system logs the notification sent to the User for auditing purposes.
* 9-10.
  + View Registered Events
    - The system provides a section or filter on the dashboard where Users can view the events they have registered for.
    - The system allows the User to cancel their registration if they no longer wish to attend.

**Client Meetings**

**Meeting 1**

**Members Present:**

* **Darren Ross**
* **Anthony Jack**
* **Manikala Chevitipalli**
* **Sai Achyuth Konda**

**Meeting Agenda:**

*Work with the Client to discuss the project requirements of the event management application. This was also to establish what exactly the client wanted and what our team was capable of producing.*

**Questions Asked:**

* 1. **Sai Achyuth Konda:** *What specific tools and frameworks are recommended or required for effective testing of our application?*
  2. **Darren Ross:** *What is the process for creating a new wiki that is distinct from our existing one, and how should we approach its structure?*
  3. **Darren Ross:** *Can you provide detailed guidance on how we should title and organize the different wiki pages for clarity and accessibility?*

**Answers Received:**

* 1. *The required testing tools and frameworks will be listed and made available on the Canvas site. Ensure that you follow the provided guidelines for their use.*
  2. *Each page should be linked in the base wiki page, with comments in each to indicate what was done.*
  3. *The titles of the wiki pages should be listed as the assignment title.*

**Action Items:**

* + *The proposed design for the application*
  + *A detailed list of application requirements*
  + *A workflow diagram illustrating the user journey and interactions with the application.*

**Meeting 2**

**Members Present:**

* **Darren Ross**
* **Anthony Jack**
* **Manikala Chevitipalli**
* **Sai Achyuth Konda**

**Meeting Agenda:**

*Provide updates on Wiki Pages in terms of clarity and formatting. This meeting also aimed to provide a progress report on Bearcat Event Manager.*

**Questions Asked:**

* 1. **Anthony Jack:** *Are submission guidelines for each part of the client's requirements?*
  2. **Darren Ross:** *Since the client wants a detailed format for our wiki pages, should we revise our current iteration?*

**Answers Received:**

* 1. **The guidelines will be provided on the page.**
  2. **No there will be progress reports that will allow for more revised iterations.**

**Action Items:**

* 1. *Revise our problem statement.*
  2. *Create a skeletal outline of the web application.*