PRODUCT BACKLOG

Sprint 2

NIMRA
MUHAMMAD FAIZAN
SHERHRYAR

Contents

1. Introduction:

1.1 Purpose:

This Prioritized Product Backlog document outlines the detailed user stories for the Event Management System (EMS), derived from the Software Requirements Specification (SRS). Each user story is assigned a priority level—High, Medium, or Low—to facilitate iterative development, ensuring that critical functionalities are delivered first.

1.2 Scope:

The backlog encompasses the complete range of EMS functionalities, including user signup, authentication, event creation and management, registration and cancellation processes, notification and reminder systems, and event reporting. Priorities are based on stakeholder needs (attendees, organizers, customers), system dependencies, and operational significance, ensuring a Minimum Viable Product (MVP) in initial sprints followed by subsequent improvements.

1.3. References:

- EMS (event management system) software requirement specification

2. Product Backlog:

2.1. Overview:

This section presents a complete list of user stories, providing detailed information such as descriptions, acceptance criteria, links to the SRS, estimated effort, potential risks, and reasoning for prioritization. Priorities are categorized as:

- **High (Red)**: Critical features necessary for basic system operation and user engagement, vital for the Minimum Viable Product (MVP).
- **Medium (Orange)**: Features that improve user experience and system efficiency, planned for implementation after core features.
- **Low (Green)**: Additional features that offer extra value but are not immediate priorities.

2.2. User Stories:

1. Event Creation:
    - ***Description:*** As an Organizer, I want to create an event so that I can schedule and manage it efficiently.
    - **Acceptance Criteria**:
        - The event creation form is accessible via the JavaFX interface.
        - The form requires event name, date, time, location, and capacity (numeric).

  - System validates that the date is in the future and capacity is positive.
  - Upon submission, the event is saved to the database, and a confirmation message displays.
  - Error messages appear for invalid inputs (e.g., past date, empty fields).
- **Effort Estimate**: 5 story points (moderate complexity due to form validation and database integration)
- **Risks**:
  - Dependency on database connectivity.
  - Potential issues with interface rendering across platforms.
- **Rationale**: Essential for the system's core purpose—without events, no functionality is possible.

2. **Login:**

- **Description:** As an Organizer, I want to log into the system so that I can securely access my event management tools.
- **Acceptance Criteria:**
  - The login screen is accessible via the JavaFX interface.
  - Requires email with and password.
  - System validates credentials against the database.
  - Successful login redirects to the organizer dashboard; failed attempts show an error message.
- **Estimate Effort:** 3 story points
- **Risks:**
  - Security vulnerabilities if passwords aren't hidden (using dots).
  - Database latency affecting login speed.
- **Rationale**: Critical for secure access to organizer features, foundational for system use.

3. **Signup:**

- **Description**: As an Attendee, I want to sign up for the system so that I can access event registration features.
- **Acceptance Criteria**:
  - The signup form is accessible via the JavaFX interface.
  - Requires name, email with valid pattern, and password.
  - System ensures email uniqueness and saves the account to the database.
  - Upon success, a confirmation message displays; errors show for duplicates or invalid inputs.
- **Effort Estimate**: 5 story points.
- **Risks**:
  - Database errors during account creation.

- **Rationale**: Essential for attendees to join the system, a prerequisite for participation.

4. Register for Event

- **Description**: As an Attendee, I want to register for an event so that I can secure my participation.
- **Acceptance Criteria**:
  - Customer/Attendee requests to register for an event.
  - Registration requires attendee register details and checks event capacity.
  - Upon success, attendee is added to the event's list; confirmation displays.
  - Errors show if the event is full or registration is closed.
- **Effort Estimate**: 5 story points
- **Risks**:
  - Race conditions if multiple attendees register simultaneously.
  - Database update failures.
- **Rationale**: Core feature for attendees, critical for the system's primary function.

5. View Events

- **Description**: As an Attendee, I want to view events so that I can choose which ones to attend.
- **Acceptance Criteria**:
  - Events are shown on screen.
  - Clicking an event shows additional details; no login required for basic view.
- **Effort Estimate**: 3 story points.
- **Risks**:
  - Slow performance with large event lists.
  - Event pictures path causing issues.
- **Rationale**: Fundamental for attendees to engage with the system, part of the MVP.

6. Update Details of Event

- **Description**: As an Organizer, I want to update event details so that I can keep attendees informed of changes.
- **Acceptance Criteria**:
  - Organizer can select an event and edit fields (e.g., date, location) via JavaFX.
  - Updates are validated (e.g., future date) and saved to the database.
  - Confirmation displays; errors show for invalid inputs.

- **Effort Estimate**: 3 story points.
- **Risks**:
  - Updates may fail mid-process.
  - Notification integration delays
- **Rationale**: Enhances flexibility but not critical for initial event setup.

7. Cancel Event:

- **Description**: As an Organizer, I want to cancel an event so that I can terminate it.
- **Acceptance Criteria**:
  - Organizer can select an event and mark it as cancelled.
  - System updates the event status and optionally notifies attendees.
  - Confirmation displays upon success.
- **Effort Estimate**: 3 story points.
- **Risks**:
  - Notification failure.
  - Database rollback issues.
- **Rationale**: Useful for management but not essential for core operation.

8. Close Event Registration:

- **Description**: As an Organizer, I want to close event registration so that I can limit attendance when needed and according to deadline date.
- **Acceptance Criteria**:
  - Organizer can toggle registration status to “closed” via JavaFX.
  - System prevents new registrations and updates the database.
  - Confirmation displays; attendees see “Registration Closed” on event view.
- **Effort Estimate**: 2 story points.
- **Risks**:
  - UI not reflecting status change immediately.
  - Database sync issues.
- **Rationale**: Improves control but can wait until registration is functional.

9. Send Event Notifications:

- **Description**: As an Organizer, I want to send event notifications so that I can update attendees on important details.
- **Acceptance Criteria**:
  - Organizer can compose and send a message to attendees via JavaFX.
  - System sends emails to all registered attendees.
  - Success message displays; errors show if email fails.
- **Effort Estimate**: 8 story points.
- **Risks**:

    - Email service down or authentication issues.
    - Email delivery delays.
- **Rationale**: Enhances communication but not required for MVP.

10. Cancel Registration by Attendee

- **Description**: As an Attendee, I want to cancel my registration so that I can withdraw if I can't attend.
- **Acceptance Criteria**:
    - Attendee can select an event and cancel registration via JavaFX.
    - System removes them from the attendee list and updates capacity.
    - Confirmation displays; errors show if cancellation fails.
- **Effort Estimate**: 3 story points (simple database update)
- **Risks**:
    - Database errors affecting capacity.
    - UI refresh delays.
- **Rationale**: Improves attendee experience but not critical for initial use.

11. Update Attendee Details

- **Description**: As an Attendee, I want to update my details so that my information remains accurate for communication.
- **Acceptance Criteria**:
    - Attendee can edit name and email via a JavaFX profile screen.
    - System validates email uniqueness and saves changes.
    - Confirmation displays; errors show for invalid inputs.
- **Effort Estimate**: 3 story points.
- **Risks**:
    - Duplicate email conflicts.
    - Database update failures.
- **Rationale**: Useful for accuracy but not urgent for basic functionality.

12. Generate Event Reports

- **Description**: As an Organizer, I want to generate event reports so that I can analyse pattern of attendees.
- **Acceptance Criteria**:
    - Organizer can select an event and generate a report via JavaFX.
    - Report includes attendees' info, displayed.
    - Success message shows; errors display if data is unavailable.
- **Effort Estimate**: 5 story points.
- **Risks**:
    - Inaccurate data if records are incomplete.
    - UI rendering complexity for reports.

- **Rationale**: Valuable for analysis but not needed for event execution.

13. Export Attendees List

- **Description**: As an Organizer, I want to export the attendees list so that I can use it for check-in or records.
- **Acceptance Criteria**:
    - Organizer can export the attendee list as a CSV file via JavaFX.
    - File includes name and email; export completes with a minute.
    - Success message displays; errors show if export fails.
- **Effort Estimate**: 3 story points.
- **Risks**:
    - File system access issues.
    - Large lists causing delays.
- **Rationale**: Adds convenience but isn't essential for core operation.

14. Send Reminders

- **Description**: As an Organizer, I want to send reminders so that I can ensure attendees don't forget the event.
- **Acceptance Criteria**:
    - Organizer can schedule and send reminders via JavaFX.
    - System emails attendees using JavaMail; success message displays.
    - Errors show if SMTP fails or no attendees are registered.
- **Effort Estimate**: 8 story points (complex due to scheduling and SMTP)
- **Risks**:
    - JavaMail integration delays.
    - Mail sending errors.
- **Rationale**: Boosts engagement but can be added later.

15. Provide Feedback

- **Description**: As an Attendee, I want to provide feedback so that I can share my experience with organizers.
- **Acceptance Criteria**:
    - Attendee can submit feedback via a JavaFX form post-event.
    - Form requires a rating (1-5) and optional comments; saved to database.
    - Confirmation displays; errors show if submission fails.
- **Effort Estimate**: 3 story points (simple form and storage)
- **Risks**:
    - Database write errors.
    - UI form validation issues.
- **Rationale**: Enhances future events but isn't critical for current functionality.

3. Prioritization Summary:

3.1 Priority Distribution:

- High Priority **(5 Stories)**: US1, US2, US3, US4, US5
    - **Total Effort**: 21 story points
    - **Details**: These stories establish the foundation of the EMS, providing essential organizer access (event creation, login) and attendee interaction (signup, registration, event viewing). Important for MVP.
- Medium Priority **(6 Stories)**: US6, US7, US8, US9, US10, US11
    - **Total Effort**: 22 story points
    - **Details**: These enhance event management (updates, cancellations, registration closure), communication (notifications), and attendee flexibility (cancellation, profile updates).
- Low Priority **(4 Stories)**: US12, US13, US14, US15
    - **Total Effort**: 19 story points
    - **Details**: Supplementary features (reports, attendee exports, reminders, feedback) offer analytical insights and user engagement but are not urgent for initial deployment. They are scheduled for later iterations to refine the system.

3.2. Development Strategy:

The development will proceed using an Agile methodology:

- ***Sprint 1:*** Deploy US1–US5 to establish a working system with organizer access and essential attendee features. Aim for completion in 2 weeks.
- ***Sprint 2***: Integrate US6–US10 to enhance event management, communication, and attendee flexibility. Projected timeline is 2–3 weeks, depending on effort and dependencies and complexity.
- ***Sprint 3:*** Add US11–US15 to include reporting, exports, reminders, and feedback, scheduled after verifying core system reliability, estimated at 2 weeks due to normal complexity.

3.3. Risk Management:

- **Dependencies**: Conduct early testing of database connectivity and JavaMail functionality to resolve risks in US1, US4, US9, and US14.
- **Performance**: Perform load testing during US5 and US12 development to confirm support for up to 1,000 attendees per event with minimal latency.

- **Security**: Implement regular checks during US2 and US3 development to protect against authentication weaknesses and ensure data security.

Arigatou gozaimasu