**Software Requirements Specification (SRS) for the Event Management System**

**1. Introduction**

**1.1 Purpose**

This document specifies the software requirements for the Event Management System. It serves as a foundational guide for the development team, outlining the system's functionalities, non-functional requirements, architectural design, and a preliminary testing plan. This SRS is intended to ensure all stakeholders have a clear and shared understanding of the project scope and expected outcomes.

**1.2 Scope**

The Event Management System is a web-based platform that facilitates the creation, management, and registration of events. The system will support core functionalities including user authentication, event browsing, event creation, and event registration.

**1.3 Intended Audience**

This document is intended for project managers, software developers, quality assurance teams, and other stakeholders involved in the system's development and implementation.

**2. Functional Requirements**

**2.1 User Management**

* **FR-UM-001:** The system shall allow new users to register by providing a username, password, and email.
* **FR-UM-002:** The system shall allow registered users to log in using their username and password.
* **FR-UM-003:** The system shall protect user passwords through secure hashing.
* **FR-UM-004:** The system shall provide a mechanism for users to log out.

**2.2 Event Management**

* **FR-EM-001:** The system shall allow authenticated users to create a new event by submitting a title, description, date, venue, and category.
* **FR-EM-002:** The system shall allow authorized users (the event creator) to update an existing event.
* **FR-EM-003:** The system shall allow authorized users (the event creator) to delete an existing event.
* **FR-EM-004:** The system shall validate event data to prevent the creation of events with invalid details (e.g., a past date).

**2.3 Event Viewing and Filtering**

* **FR-EV-001:** The system shall display a paginated list of all available events.
* **FR-EV-002:** The system shall allow users to view detailed information for a specific event.
* **FR-EV-003:** The system shall provide a search and filter functionality to find events by date and/or category.
* **FR-EV-004:** The system shall display a message ("No events found") if no events match the search criteria.

**2.4 Event Registration**

* **FR-ER-001:** The system shall allow authenticated users to register for an event.
* **FR-ER-002:** The system shall check for available slots before confirming a registration. If the event is full, it shall return a 400 Bad Request error.
* **FR-ER-003:** The system shall persist the registration data in the database, linking the user and the event.

**3. Non-Functional Requirements**

**3.1 Performance**

* **NFR-P-001:** Event retrieval and filtering should be responsive, with a page load time of under 3 seconds under normal load.
* **NFR-P-002:** The system should efficiently handle paginated requests for a large number of events.

**3.2 Security**

* **NFR-S-001:** User passwords must be stored securely using industry-standard hashing algorithms.
* **NFR-S-002:** All API endpoints must be protected against unauthorized access, using tokens or session management.
* **NFR-S-003:** Input validation must be implemented on all forms to prevent injection attacks (e.g., SQL injection, XSS).

**3.3 Usability**

* **NFR-U-001:** The user interface (UI) must be intuitive and easy to navigate.
* **NFR-U-002:** The system should provide clear and helpful feedback to users for all actions (e.g., successful registration, validation errors).
* **NFR-U-003:** The application should be responsive and accessible on various devices, including desktops, tablets, and mobile phones.

**4. Diagrams**

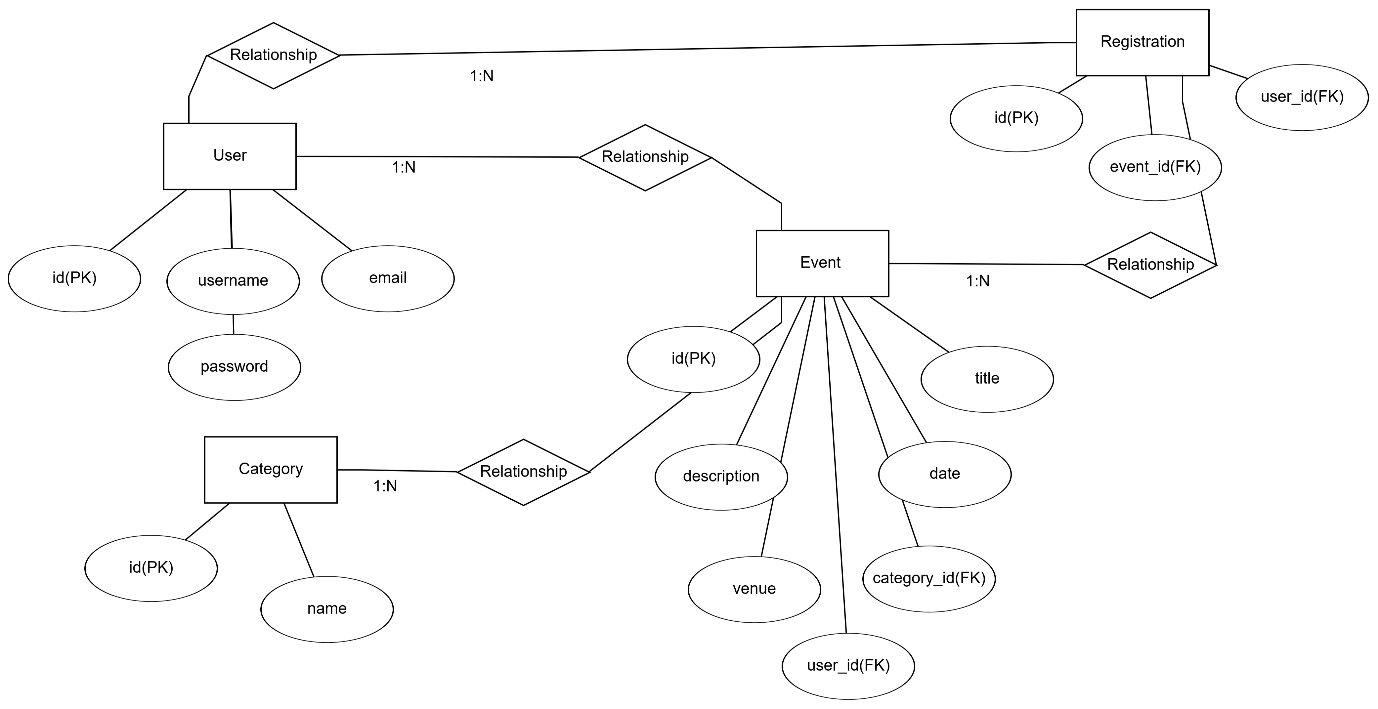
**4.1 Entity-Relationship (ER) Diagram**

The database schema will be based on the following entities and relationships:

* **User:**
  + Attributes: id (Primary Key), username, password, email.
* **Event:**
  + Attributes: id (Primary Key), title, description, date, venue, category\_id (Foreign Key), user\_id (Foreign Key).
* **Registration:**
  + Attributes: id (Primary Key), user\_id (Foreign Key), event\_id (Foreign Key).
* **Category:**
  + Attributes: id (Primary Key), name.

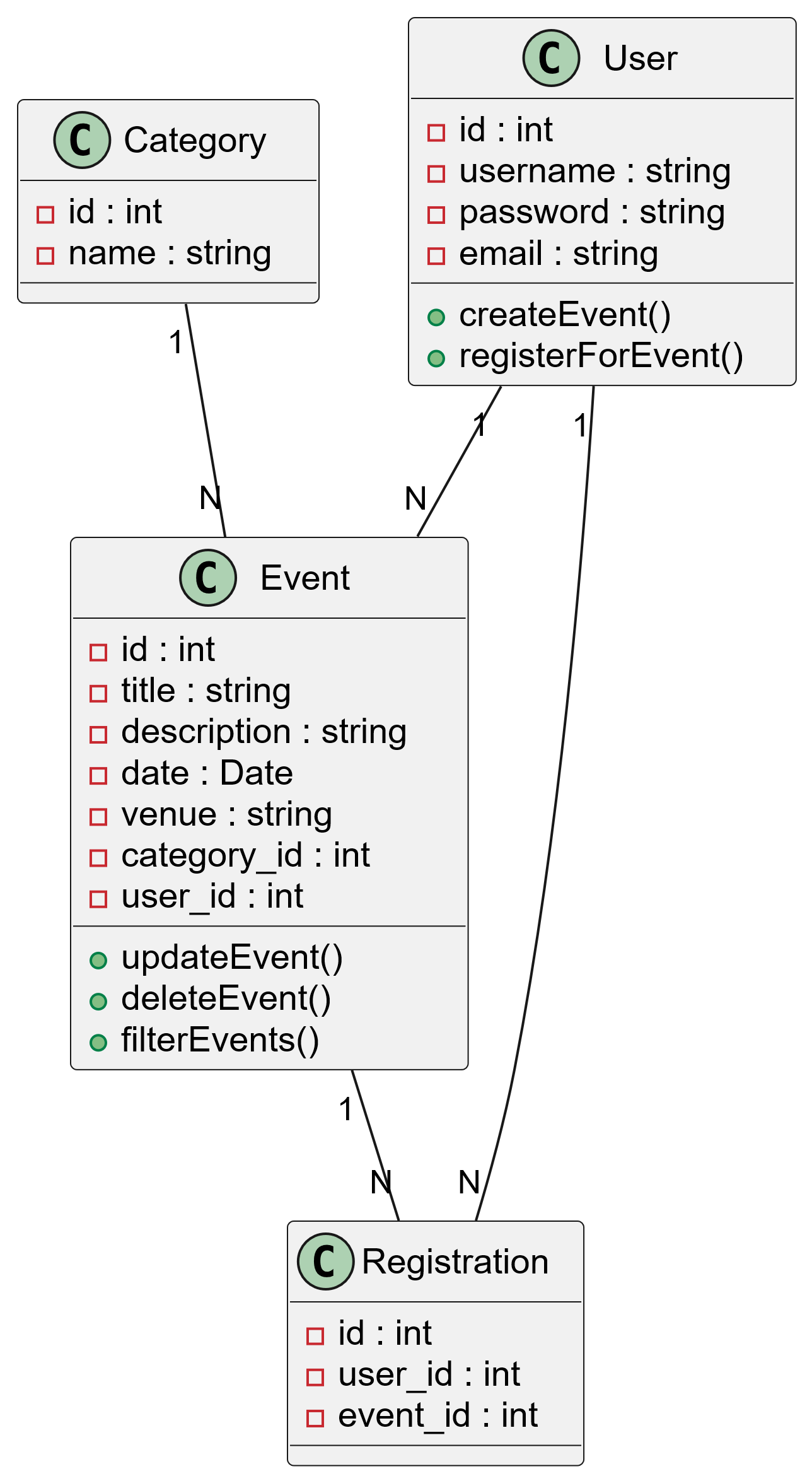
**Relationships:**

* **User 1:N Event:** A user can create many events.
* **User 1:N Registration:** A user can have many registrations.
* **Event 1:N Registration:** An event can have many registrations.
* **Category 1:N Event:** A category can be associated with many events.



**4.2 Use Case Diagram**

* **Actors:**
  + **User:** A person who interacts with the system to view, register, and create events.
  + **Admin (Optional):** An actor with additional privileges, such as approving events.
* **Use Cases:**
  + **Log In:** User authenticates to the system.
  + **View Events:** User browses a paginated list of events.
  + **Filter Events:** User searches for events by date or category.
  + **Create Event:** Authenticated user submits an event.
  + **Update Event:** Authenticated user modifies their own event.
  + **Delete Event:** Authenticated user removes their own event.
  + **Register for Event:** Authenticated user signs up for an event.
  + **View Event Details:** User sees the detailed information for a specific event.
  + **Log Out:** User exits their session.



**4.3 High-Level System Architecture**

The system will follow a three-tier architecture:

1. **Presentation Tier (Frontend):** Developed using **React**. This layer handles the user interface and interactions. It communicates with the Application Tier via a REST API.
2. **Application Tier (Backend):** Developed using **Spring Boot**. This layer contains the core business logic, handles API requests, processes data, and interacts with the database.
3. **Data Tier (Database):** A **MySQL** database will be used to store all persistent data, including users, events, and registrations.

**5. Testing Plan**

**5.1 Unit Testing**

Individual components and methods within the Spring Boot backend and React frontend will be tested in isolation. This includes testing service methods, controller endpoints, and React components to ensure they function as expected.

**5.2 Integration Testing**

Integration tests will verify that different modules of the system work together correctly. This will focus on the interaction between the frontend and backend, as well as the backend's interaction with the database.

**5.3 System Testing**

This phase will involve testing the complete, integrated system to ensure it meets all functional and non-functional requirements.

**5.4 Sample Test Cases**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case ID** | **Requirement(s)** | **Test Description** | **Expected Result** |
| **TC-001** | FR-UM-002 | Verify successful user login with valid credentials. | User is authenticated and redirected to the events page. |
| **TC-002** | FR-ER-002 | Attempt to register for an event that has no available slots. | The system returns a 400 Bad Request error with a message indicating the event is full. |
| **TC-003** | FR-EV-003 | Filter events by a specific category. | Only events belonging to the selected category are displayed. |
| **TC-004** | FR-EM-004 | Attempt to create an event with a date in the past. | The system returns a validation error and the event is not created. |

**6. Conclusion**

This SRS document provides a detailed blueprint for the development of the Event Management System. By adhering to the requirements and design principles outlined herein, the development team can build a robust, scalable, and user-friendly application that meets the needs of both organizers and attendees.