**EVENT IT!**

- **AN EVENT REPOSITORY**



**OOAD PROJECT ITERATION 2**

**SUBMITTED BY TEAM: Group9.getEvent( )**

**Date: 03/21/2016**

**Team Members:**

Namrata Singh

Naveenraj Palanisamy

Navya Vandanapu

Niveditha Varadha Chandrasekaran

Shravya Kuncha

**INDEX**

1. Introduction 3

1.1 Vision

1.2 Overview

2. Project scope for iteration 2 3

3. Domain Model 4

4. Design Specification 5

4.1 Functional requirements

4.2 Non Functional requirements

4.3 Constraints and Limitations

5. Use Case Diagram 6

6. Use Case Text 7

7. System Sequence Diagram 10

8. Operation Contract 13

9. Interaction Diagrams 14

10. Class Diagram 21

11. Supplementary Specification 22

12. Testing 22

13. UI Screenshots 24

14. Plan for Iteration-II 29

15. Glossary 30

16. Development Case 30

1. **INTRODUCTION**
   1. **Vision**

To create a web application – EventIt!, a one stop event repository that helps people to know about the various events happening in and around their neighborhood. It also serves as a platform for the event hosts to promote their ticket free events.

**1.2 Overview**

The following functionalities are included in this system.

* Provides a platform to create events and promote them on the website
* Helps people to get to know the various events taking place
* Allows users to browse the events based on different categories
* Provides login to users to host or to register for events
* Allows users to reserve tickets for events based on the availability
* Allows the event hosts to update or delete events
* Allows users who have reserved tickets to rate the events once the event has happened
* Ensures that notifications are sent for every update
* Eliminates any inappropriate content present

1. **PROJECT SCOPE FOR ITERATION 2**

The following core functionalities will be addressed in this iteration.

* Browsing the events based on category and date.
* Reserving tickets for events by the registered users.
* Cancelling the reserved tickets by the registered users.

1. **DOMAIN MODEL**

|  |
| --- |
| C:\Users\Niveditha\Desktop\EventIt domain model.jpg |

1. **DESIGN SPECIFICATION**

**4.1 Functional requirements:**

* Admin should be able to approve or reject event publish requests.
* Registered users should be able to create the events and submit them to the admin for approval.
* Registered users should be able to browse events, reserve tickets if needed and rate the events.
* General users should be able to browse events and sign up for the website if needed.

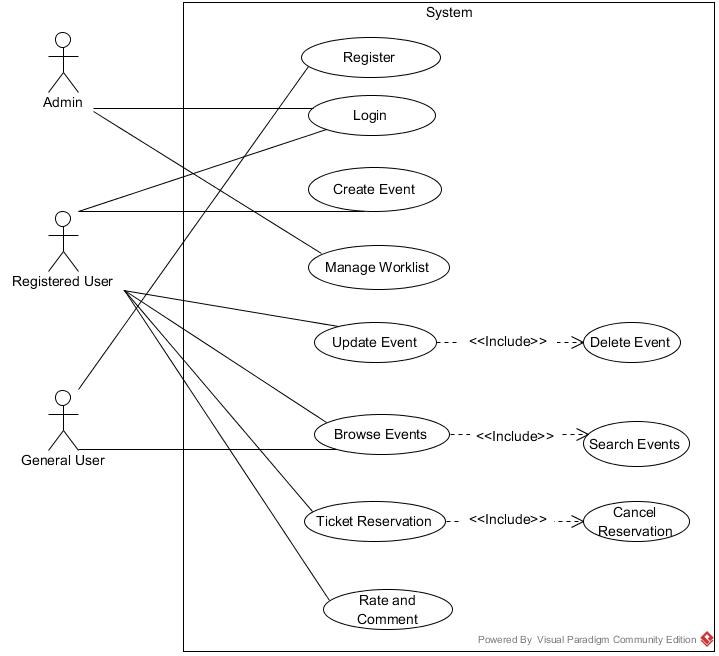
**4.2 Non Functional requirements:**

* *Usability:* This application provides usability by having a user friendly user experience. As the application is hosted in the web, anyone with Internet can access the website.
* *Security:* The application is password restricted for registered users and admin, so that general users and other anonymous users will not be able to see the controls that are available for the admin and registered users.
* *Deployment:* The application is deployed in web to get worldwide access.
* *Changeability:* The application changes its layout based on the screen size.

**4.3 Constraints and Limitations**

* The application will not let any user to create and promote their events until they sign up for the website.
* The application will not let any registered users to modify or delete the events which are not created by them.

1. **USE CASE DIAGRAM**



1. **USE CASE TEXT**

**ACTORS:** General User, Registered User, Admin

**SYSTEM:** The EventIt System

**6.1 USE CASE UC1: CREATE EVENT**

**Scope:** EventIt application

**Level:** User goals

**Primary Actor:** Registered user

**Stakeholders and Interests:**

**Registered user:** Can create, promote their events and browse and reserve tickets for published events

**General user:** Can browse published events

**Pre-conditions:** The user should be logged into the application

**Post-conditions:** The event created is sent for approval.

**Main Success Scenario (Basic Flow):**

1. The user selects Create event option.
2. The system displays a form to input event details like Event name, Event Location, Event Date, etc.
3. The user fills the form and submits it for the approval
4. The system sends an email notification to the admin for event approval request

**Extensions (or Alternative Flows):**

3a. The user enters an event name which is already present

1. The system prompts a message about the duplicate event name

1a. The user ignores the message and submits the form with same event name

1b. The user changes the event name and submits the form

**6.2 USE CASE UC2: MANAGE WORKLIST**

**Scope:** EventIt application

**Level:** User goals

**Primary Actor:** Admin

**Stakeholders and Interests:**

**Registered user:** Can create, promote their events and browse and reserve tickets for published events

**General user:** Can browse published events

**Pre-conditions:** The admin should be logged into the application

**Post-conditions:** The worklist is updated and the approved events are published on the website

**Main Success Scenario (Basic Flow):**

1. The admin clicks on the worklist
2. The system displays the event approval requests
3. The admin clicks on the event for its details
4. The system displays the details from the create event form submitted by the user
5. The admin approves or rejects the event
6. The system sends a notification to user regarding the event approval.

**Extensions (or Alternative Flows):**

5a. The admin approves the event

1. The system publishes the event on the website

**6.3 USE CASE UC3: BROWSE EVENTS**

**Scope:** EventIt application

**Level:** User goals

**Primary Actors:** General user, Registered user

**Stakeholders and Interests:**

**Registered user:** Can create, promote their events and browse and reserve tickets for published events

**General user:** Can browse published events

**Pre-conditions:** User must visit the web application

**Post-conditions:** List of all the events is displayed

**Main Success Scenario (Basic Flow):**

1. The user visits the web site and clicks on the browse events
2. The system displays the list of all the upcoming events
3. The user clicks on the event of his/her choice
4. The system displays the event page with the event details.

**Extensions (or Alternative Flows):**

2a. The system displays the message “No upcoming events”

2b. Includes UC6 Search events

**6.4 USE CASE UC4: TICKET RESERVATION**

**Scope:** EventIt application

**Level:** User goals

**Primary Actor:** Registered User

**Stakeholders and Interests:**

**Registered user:** Can create, promote their events and browse and reserve tickets for published events

**General user:** Can browse published events

**Pre-conditions:** The user should be logged into the application and select an event

**Post-conditions:** The event ticket is reserved for the user

**Main Success Scenario (Basic Flow):**

1. The user clicks on the reserve ticket option on the event page
2. The system displays a confirmation form for ticket reservation
3. The user confirms by filling his details in the form and submits it
4. The system sends an email to the user regarding the confirmation of ticket reservation
5. The system displays a message “Ticket is reserved”

**Extensions (or Alternative Flows):**

a. At any time system fails,

To support recovery, ensure the system’s state is not changed and all the events are recovered from any step of the scenario.

1a. User select cancel ticket option on the event page. Include use case ‘Cancel Ticket Reservation’ UC5

2a. The system displays a message “Tickets are sold out”

**6.5 USE CASE UC5: CANCEL TICKET RESERVATION**

**Scope:** EventIt application

**Level:** User goals

**Primary Actor:** Registered User

**Stakeholders and Interests:**

**Registered user:** Can create, promote their events and browse and reserve tickets for published events

**General user:** Can browse published events

**Pre-conditions:** The user should have reserved a ticket for an event

**Post-conditions:** The reserved ticket is cancelled for the user

**Main Success Scenario (Basic Flow):**

1. The user selects the Ticket reservation history
2. The system displays the list of reserved tickets
3. The user selects the particular ticket to cancel
4. The user clicks on the cancel ticket reservation button
5. The system sends an email notification to the user regarding the ticket cancellation
6. The system displays the message “Reserved ticket is cancelled”

**Extensions (or Alternative Flows):**

a. At any time system fails,

To support recovery, ensure the system’s state is not changed and all the events are recovered from any step of the scenario.

5a. The system shows a message “Ticket registration finalized. Cannot cancel the ticket”

**6.6 USE CASE UC6: SEARCH EVENTS**

**Scope:** EventIt application

**Level:** User goals

**Primary Actor:** General User,Registered User

**Stakeholders and Interests:**

**Registered user:** Can create, promote their events and browse and reserve tickets for published events

**General user:** Can browse published events

**Pre-conditions:** The user should select browse events tab

**Post-conditions:** The events are sorted and displayed based on the selected filter

**Main Success Scenario (Basic Flow):**

1. The user selects the filter (by category or by date) by which events should be sorted and displayed
2. The system displays the list of events based on the selected filter

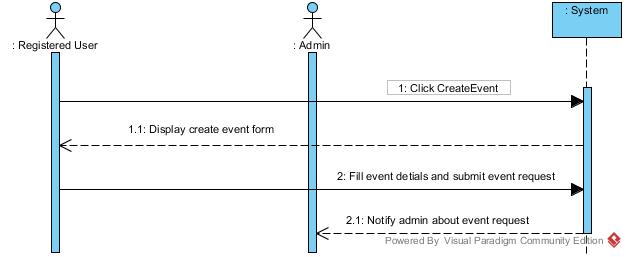
**Extensions (or Alternative Flows):**

a. At any time system fails,

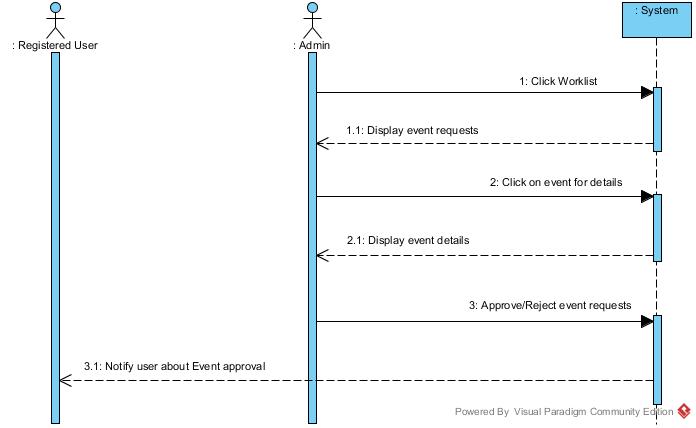
To support recovery, ensure the system’s state is not changed and all the events are recovered from any step of the scenario.

1. **SYSTEM SEQUENCE DIAGRAM**

**7.1 CREATE EVENT**

****

**7.2 MANAGE WORKLIST**



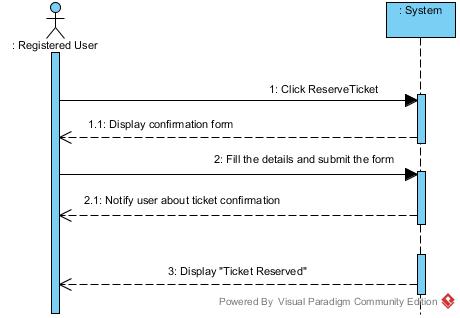
**7.3 BROWSE EVENTS**



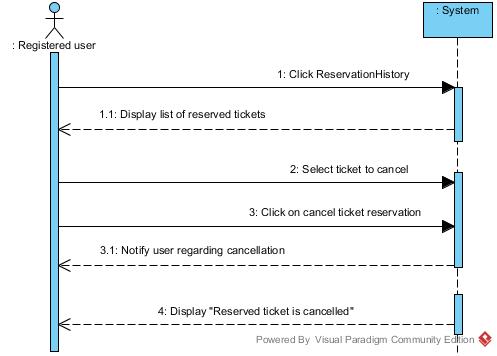
**7.4 SEARCH EVENTS**

****

**7.5 TICKET RESERVATION**

****

**7.6 CANCEL TICKET RESERVATION**

****

1. **OPERATION CONTRACT**

**Contract 1:** Create Event

**Operation:** createEvent (EventName: varchar, Decr: varchar, Venue: varchar, Date: date)

**Cross Reference:** Use case: Create Event

**Preconditions:** none

**Post conditions:** Event instance E was created

Attributes of E were initialized

Work List instance W was created

E and W were associated

**Contract 2:** Manage Work List

**Operation:** updateWorkList(Event: varchar)

**Cross Reference:** Use case: Create Event

**Preconditions:** There is aCreate event underway

**Post conditions:** Attributes of W were modified

Work instance W was updated

**Contract 3:** BrowseEvents

**Operation:** getallevents()

**Cross Reference:** Use case: Browse Events

**Preconditions:** none

**Post conditions:** E.display() became true

**Contract 4:** SearchEvents

**Operation:** getallevents()

**Cross Reference:** Use case: Search Events

**Preconditions:** There is a Browse event underway

**Post conditions:** Check instance C was created

Events E were associated with C

E.display() became true

**Contract 5:** ReserveTicket

**Operation:** Ticket()

**Cross Reference:** Use case: Ticket Reservation

**Preconditions:** There is at least one Ticket with T.status as false

**Post conditions:** Ticket instance T was created

T was associated with E

T.ticketId became Ticket ID

T.status became true

**Contract 6:** Cancel Ticket

**Operation:** getticket()

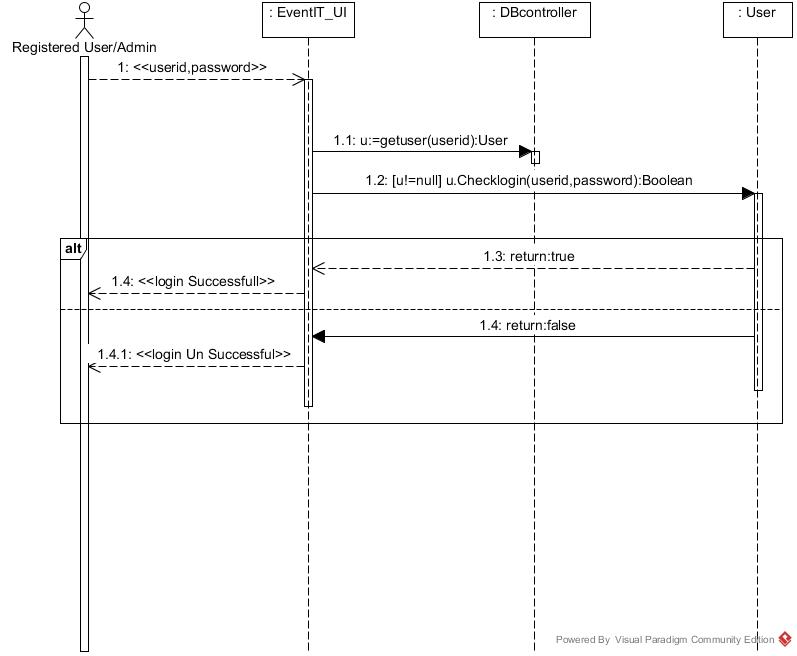
**Cross Reference:** Use case: Ticket Reservation

**Preconditions:** There should be a Ticket T for an event with T.status as true

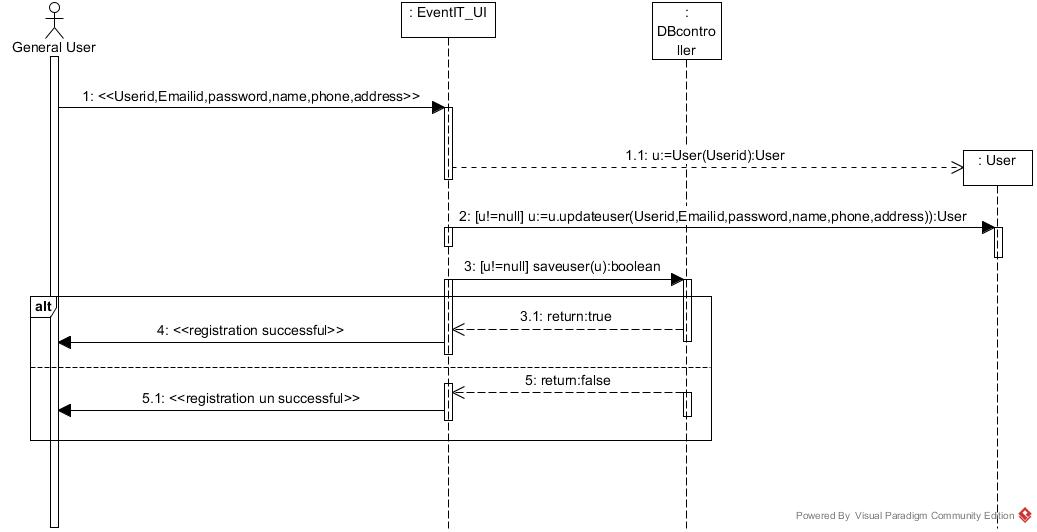
**Post conditions:** T.status became false

1. **INTERACTION DIAGRAMS**

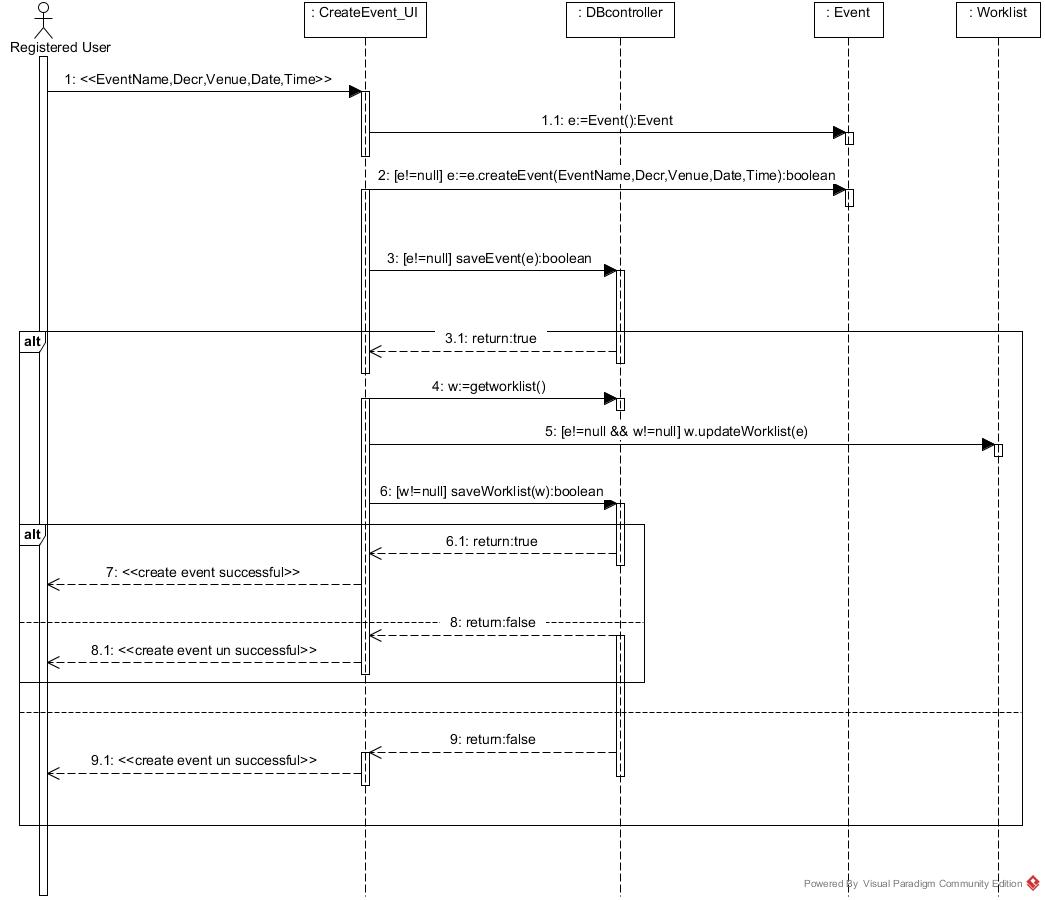
**9.1 LOGIN**



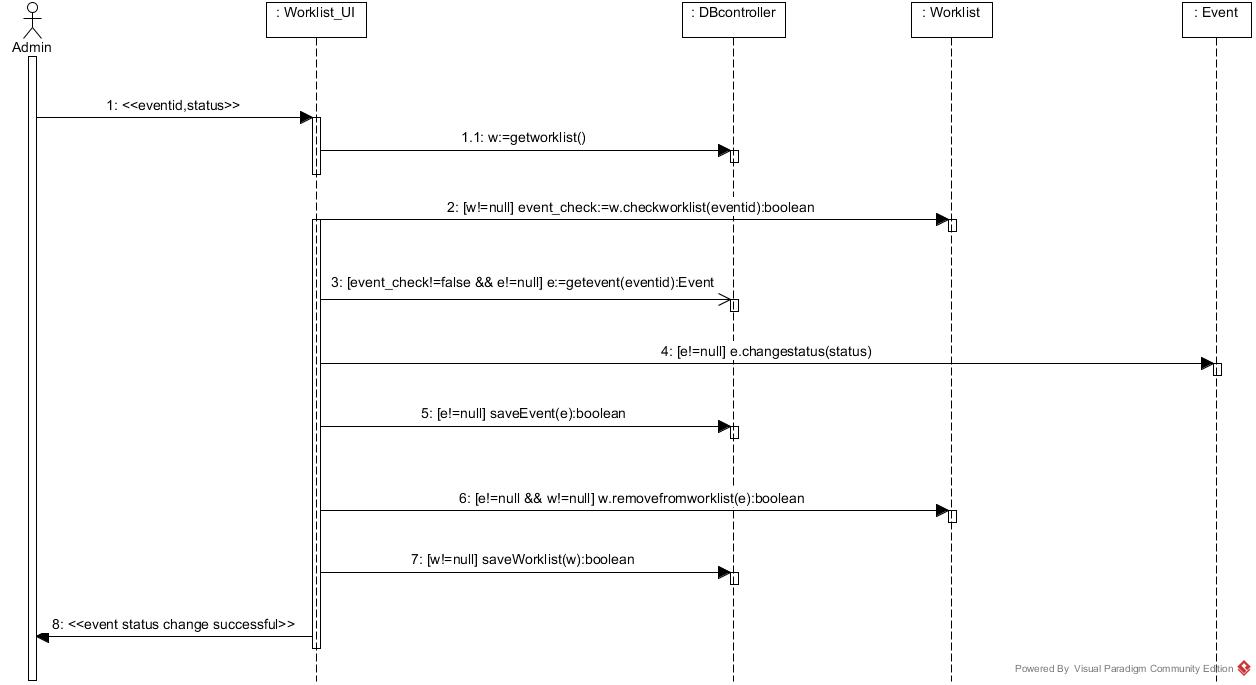
**9.2 REGISTER**

****

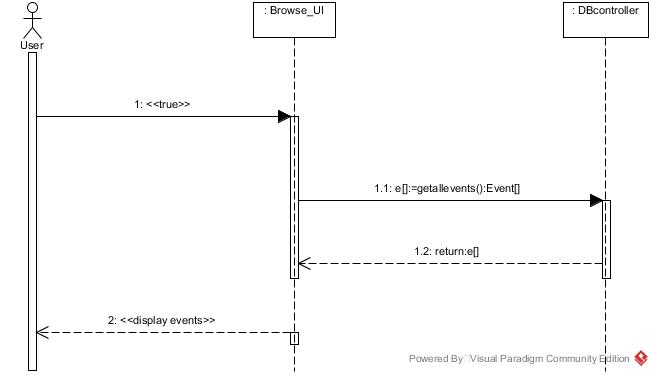
**9.3 CREATE EVENT**



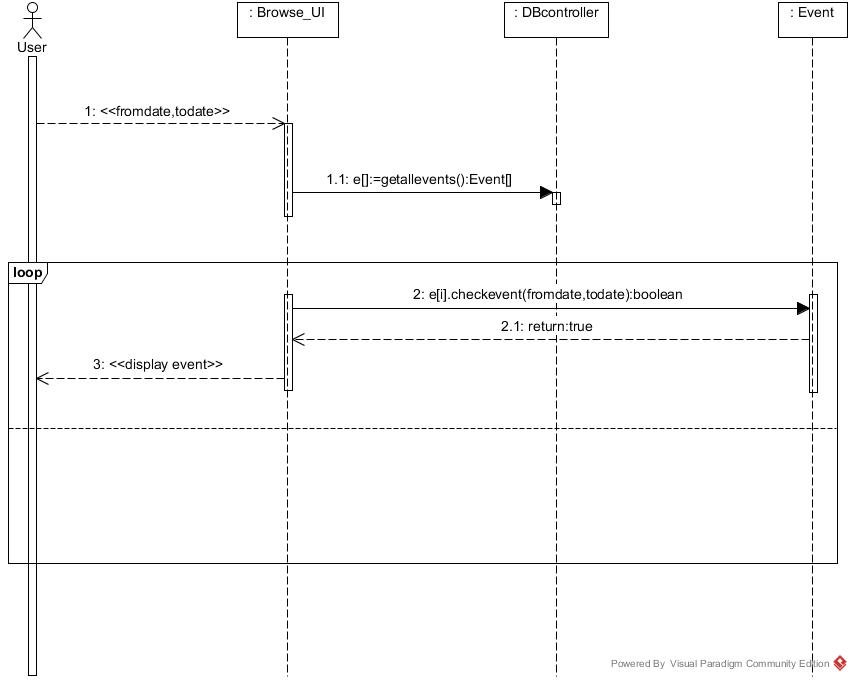
**9.4 MANAGE WORKLIST**



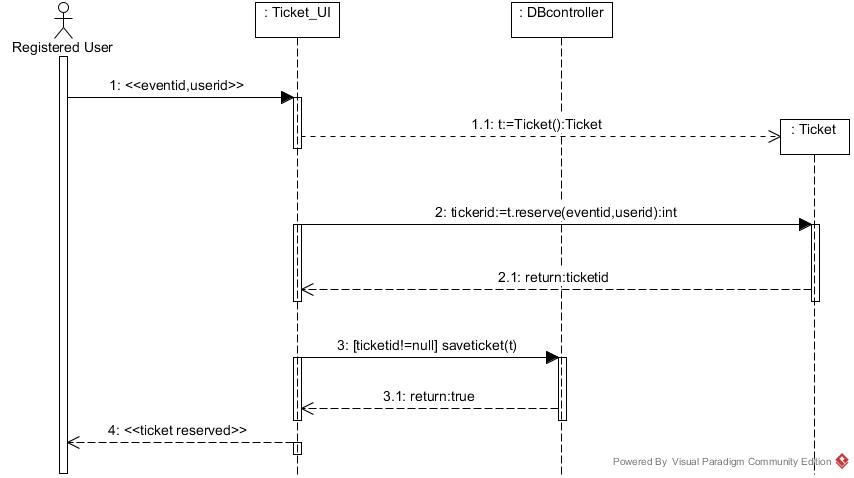
**9.5 BROWSE EVENTS**



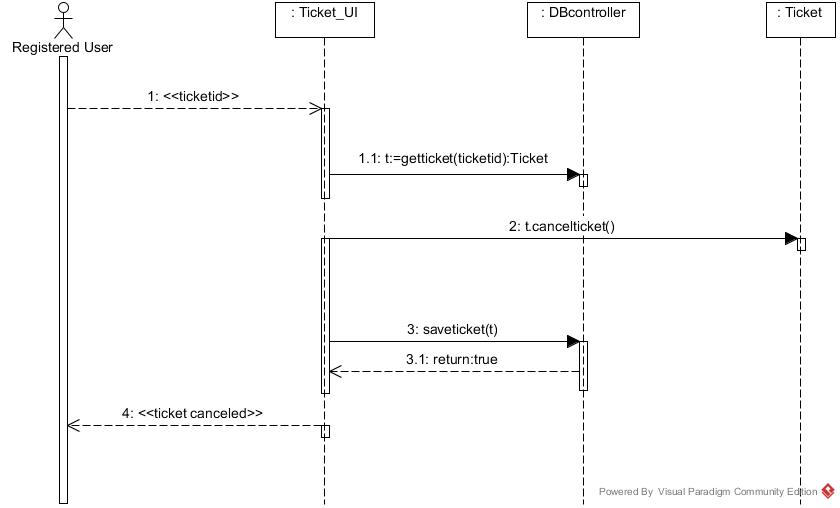
**9.6 SEARCH EVENTS**



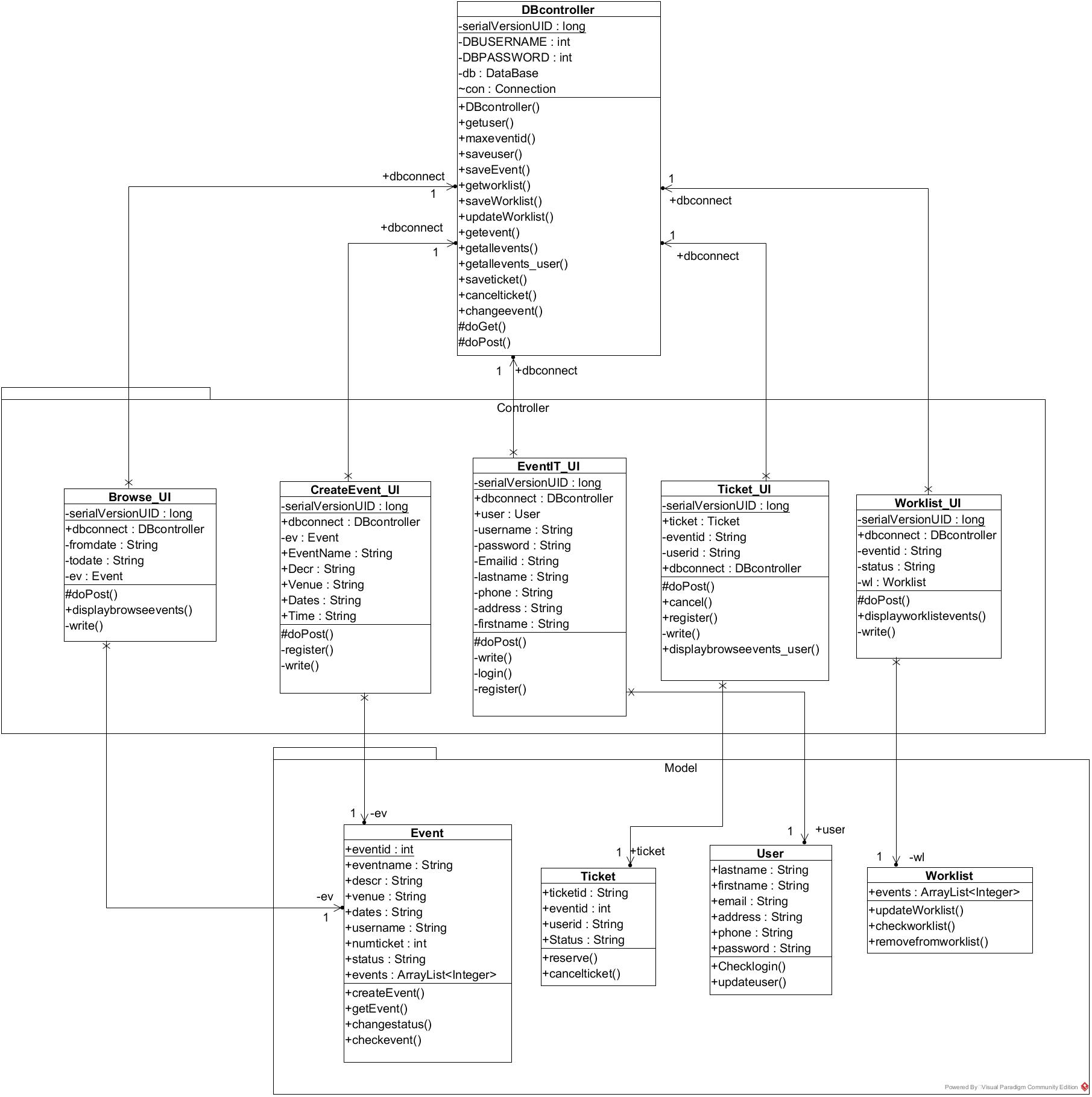
**9.7 TICKET RESERVATION**

****

**9.8 CANCEL TICKET RESERVATION**

****

1. **CLASS DIAGRAM**

****

1. **SUPPLEMENTARY SPECIFICATION**

* This program must run on all the platforms.
* This program must run on Internet Explorer, Google Chrome, Safari and Firefox.
* Developer will give a manual that explains how to use this program.

1. **TESTING**

Testing is done on the software to check if the quality is maintained and implementation is according to requirements stated earlier.

* Meets the requirements which guided design and development
* Works for correct inputs and handles incorrect inputs
* Get system response within time frame.
* Achieves the Stakeholder goal.

**11.1 Test Plans**

The following activities and test cases are used to check the quality of code and to map the implementation with the requirements of the system.

**11.2 Unit Testing**

* Unit testing is done for each use case.
* Testing is done by giving basic inputs.

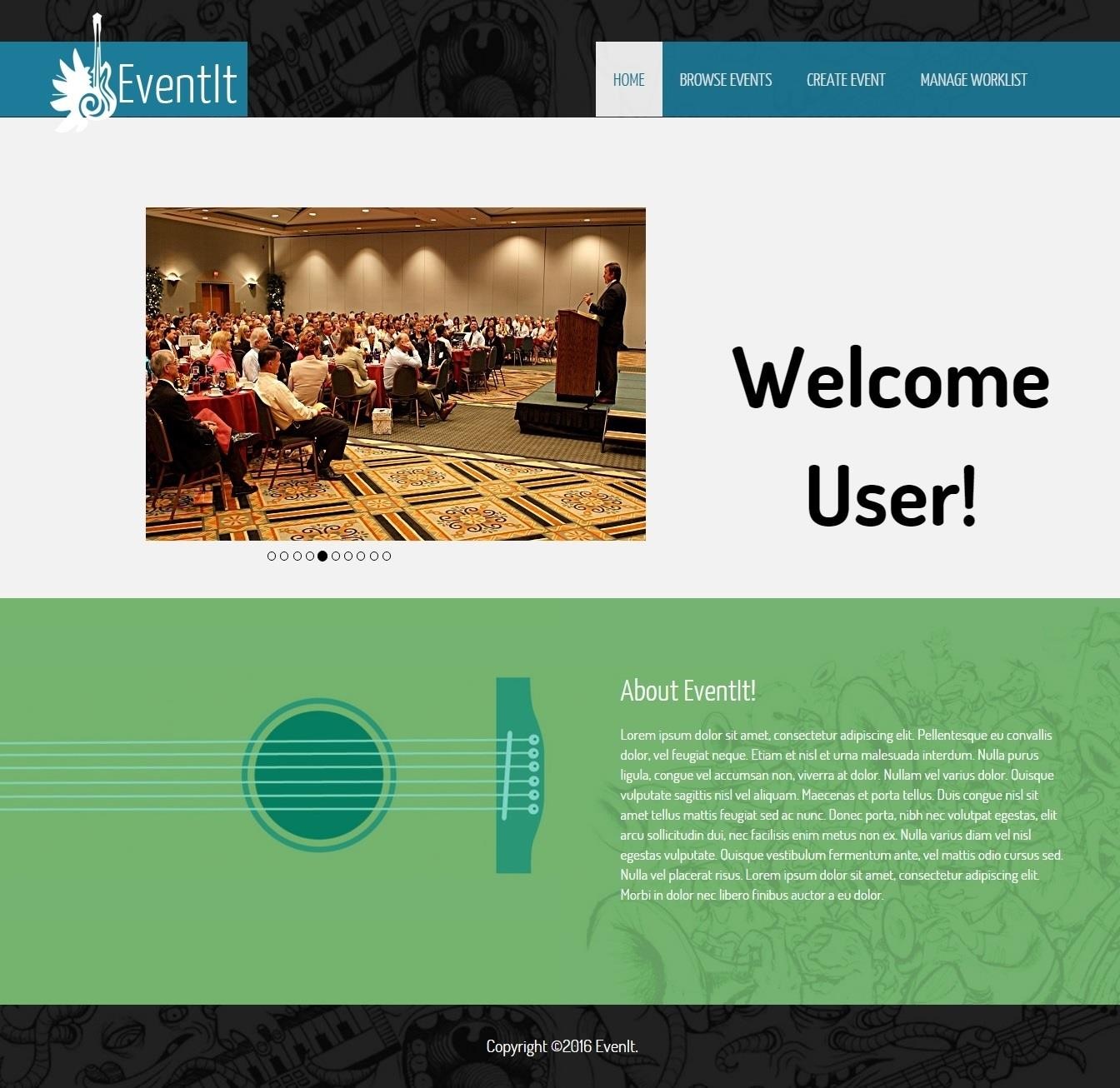
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No** | **Test Case** | **Test Case Description** | **Expected**  **Output** | **Observed**  **Output** |
| 1 | Login by using email  id/user id | 1.The user enters URL  and clicks enter.  2.User provides login details and clicks login | The user is able  to login in | The user is able  to login successfully and home page is displayed |
| 2 | Sign up to the system | 1.The user enters URL  and clicks enter  2.User clicks sign up  3.Open Model page  4.User provides sign up details and clicks submit | The user is able  to sign up | The user is able  to sign up successfully and login in page is displayed |
| 3 | Create Event | 1.The user enters URL  and clicks enter  2.Open create event page  3.Provide event details  4.Clicks submit | The user is able to create event | The user is able to create event and event is sent for approval |
| 4 | Manage Worklist | 1.The admin enters  URL and clicks enter  2.Open worklist page  3.Clicks event to approval and clicks approve or reject | The admin is  able to manage worklist | The user is able to approve event successfully and event is published. |
| 5 | Browse Events | 1.The user enters URL  and clicks enter  2.Click Browse events  3.Displays published events | The user is able  to view events | The user is able  to view event successfully |
| 6 | Ticket Reservation | 1.The user enters URL  and clicks enter  2.Clicks Ticket reservation  3.Displays reservation form | The user is able  to reserve tickets | The user is able  to reserve tickets successfully |
| 7 | Cancel Reservation | 1.The user enters URL  and clicks enter  2.Clicks reservation history  3.Clicks on ticket  4.Clicks cancel reservation | The user is able  to cancel reserved tickets | The user is able  to cancel reserved tickets successfully |

1. **UI SCREENSHOTS**

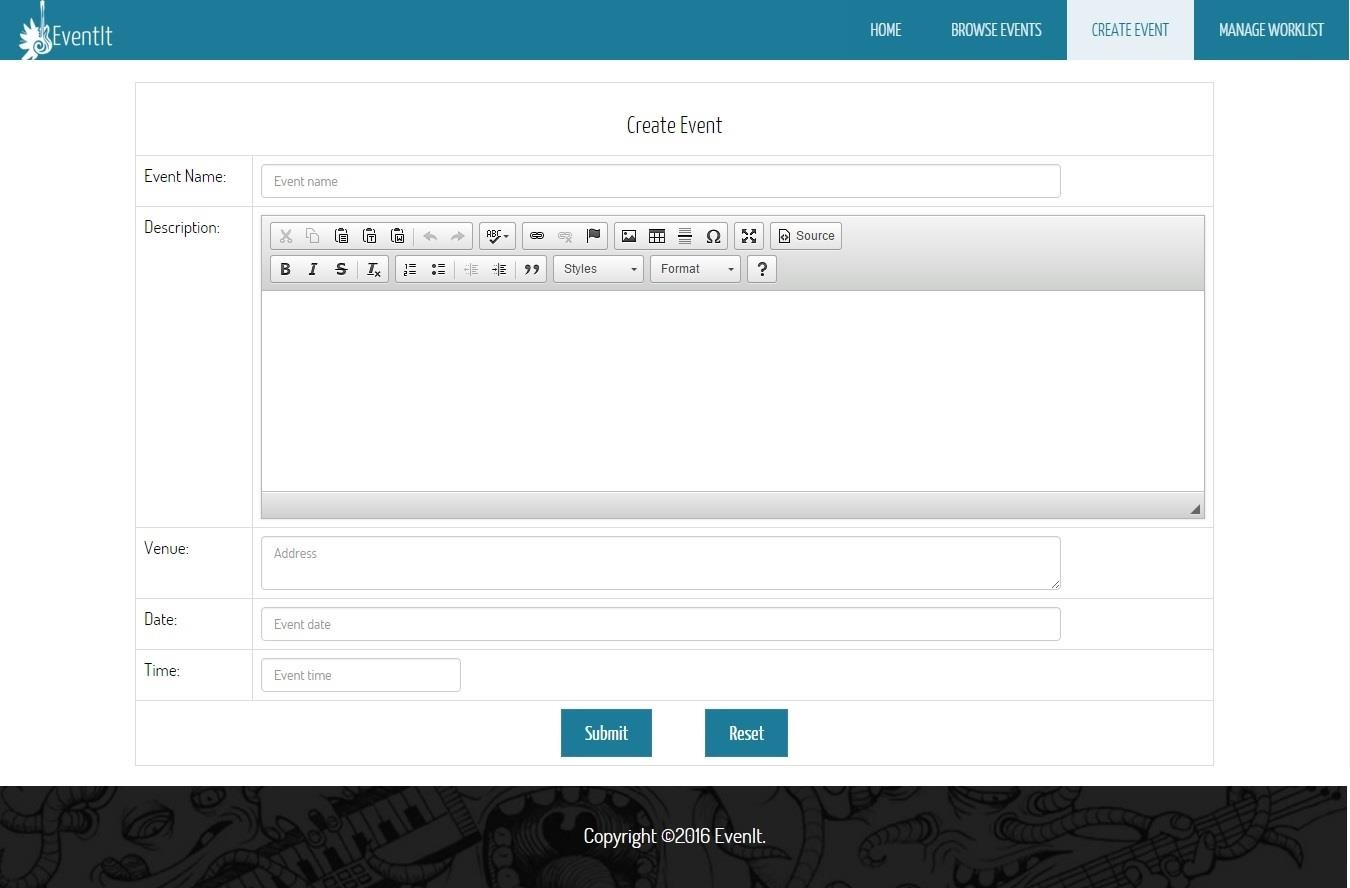
**Login Page**



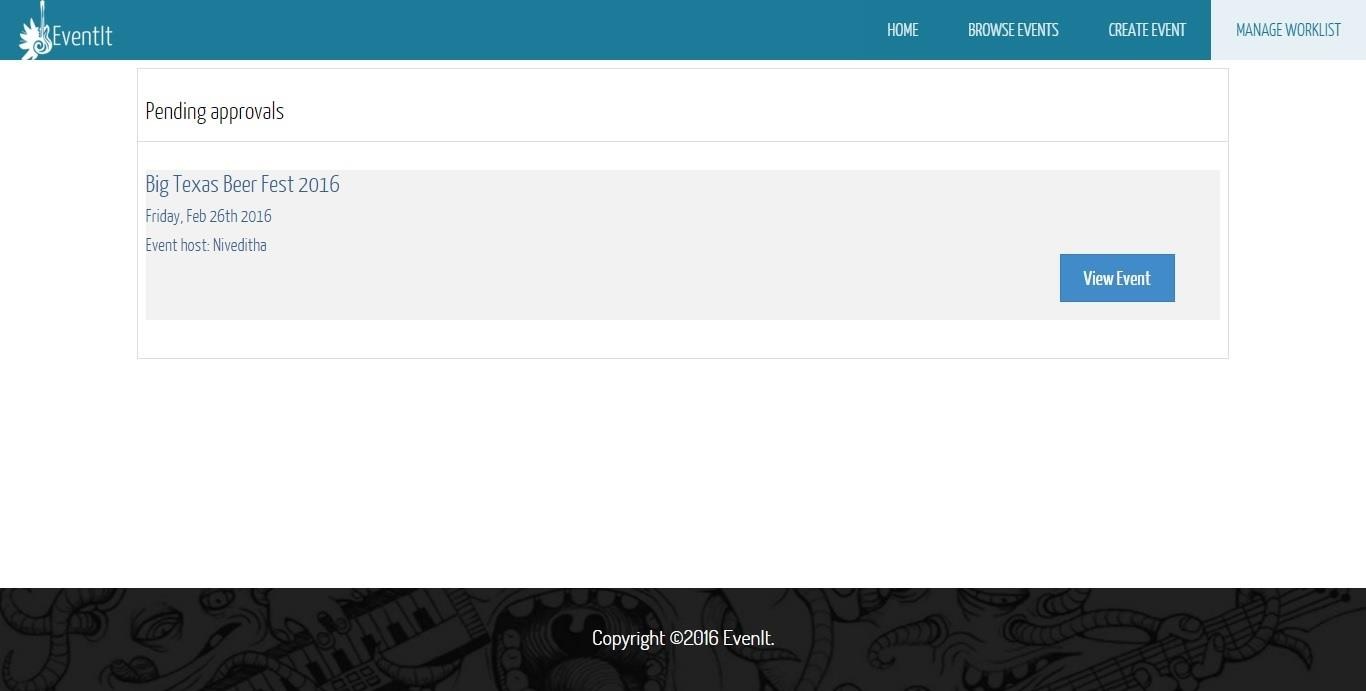
**Home Page**



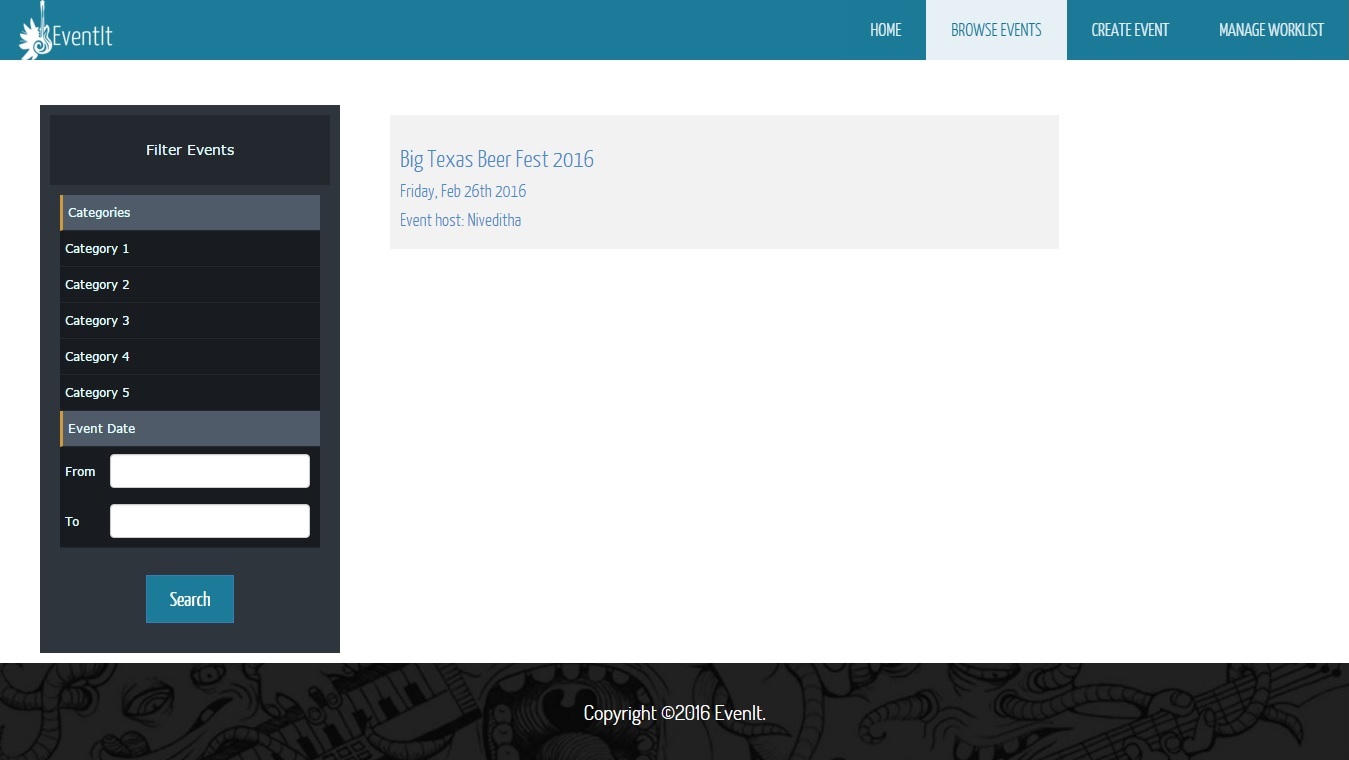
**Create Event**



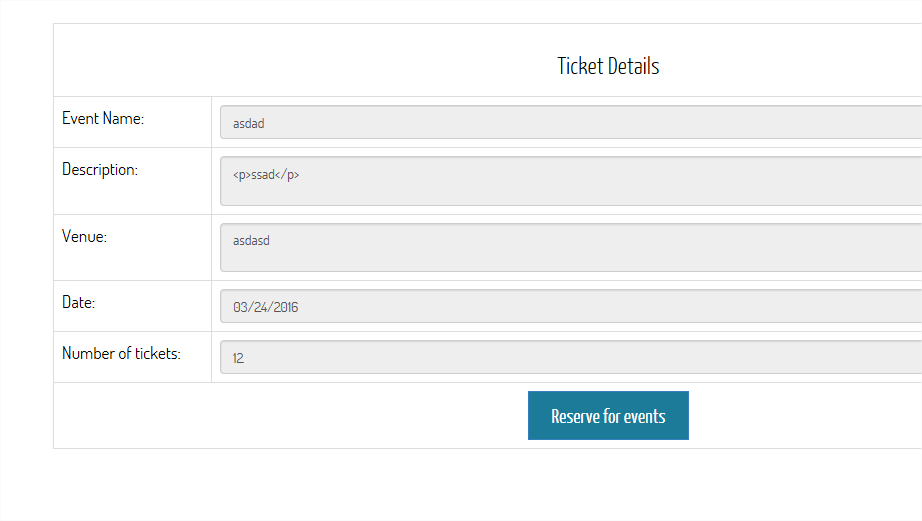
**Manage Worklist**

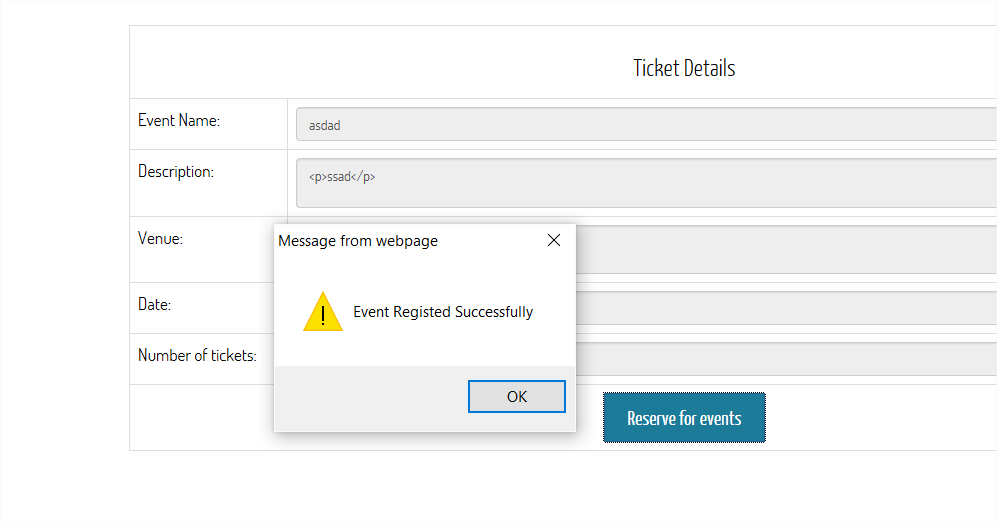


**Browse Events**

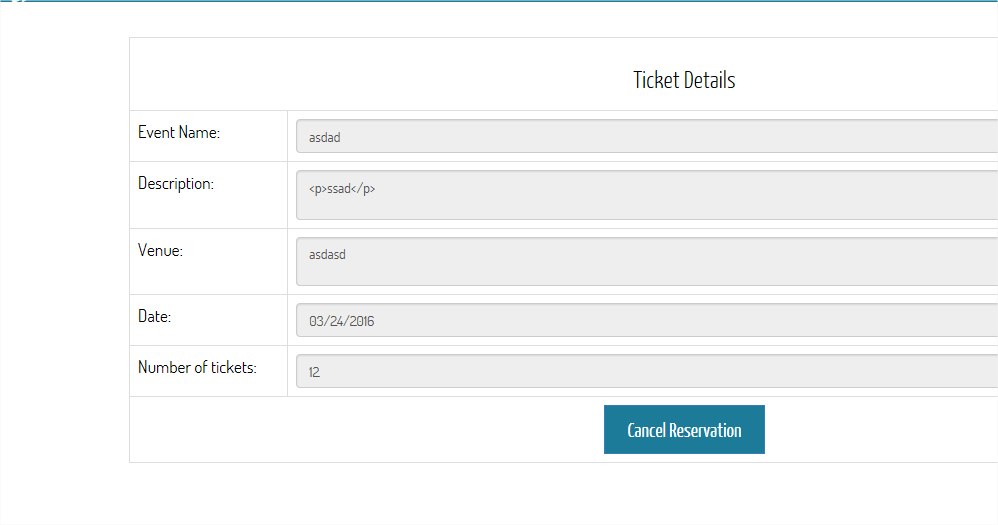
****

**Ticket Reservation**

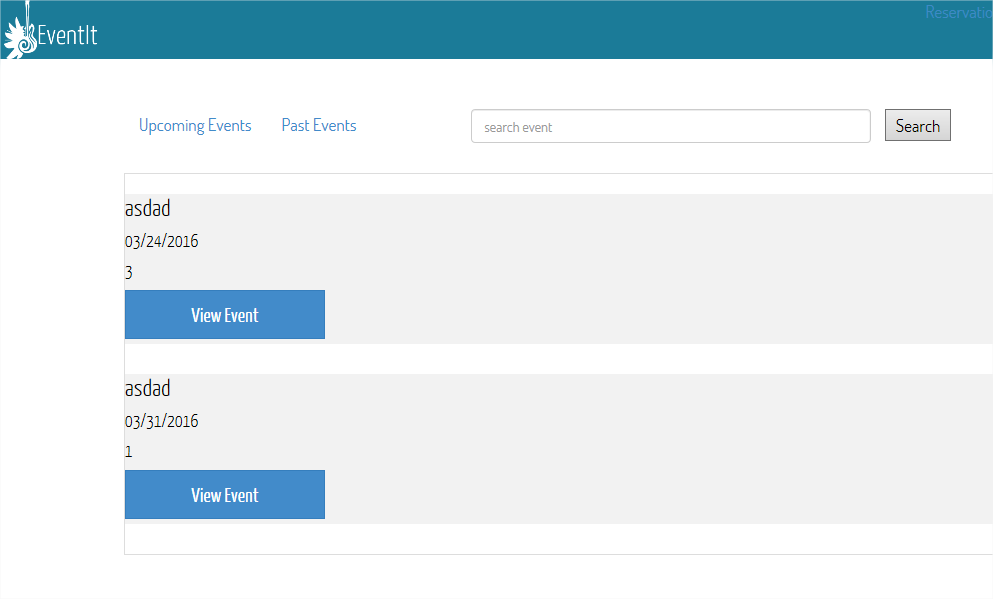




**Cancel Ticket Reservation**



**Search Events**



1. **PLAN FOR ITERATION - III**

* Requirement analysis
* Update event module
* Delete event module
* Rate and comment module
* Refine the system
* Documentation and Report
* Backlogs from iteration 2

1. **GLOSSARY**

|  |  |  |  |
| --- | --- | --- | --- |
| **Glossary** | | | |
| **Revision History** | | | |
| **Version** | **Date** | **Description** | **Author** |
| Iteration 1 document | 2/22/2016 | First draft to be refined in later iterations | Group 9 |
| Iteration 2 document | 3/20/2016 | First draft to be refined in later iterations | Group 9 |
| **Definitions** | | | |
| **Terms** | **Definition and Information** | | |
| Admin | Admin is responsible for the credibility of the events being published on the  website. Whenever a registered user creates an event, an event request is sent to the admin for approval, to check the authenticity of the event and its host, before being published on the site. This prevents the creation of fraudulent events on the site. | | |
| Registered User | Registered user is the user who has signed up for the website. These users have the privilege to create events and promote them after the event is approved by the admin. They can also edit the event details or delete the event. Not only that, but they can also reserve tickets for the events they wish to attend, which are hosted by other users. They will also have permission to rate the events they have attended. | | |
| General User | General users are the other users who visit the website. They can browse  events based on different event types and look into the details but they will not be able to reserve tickets unless they sign up | | |
| Username | User ID of the User to be used to login to the system. | | |
| Worklist | The list of events to be approved by the admin to be published on the web  site. | | |

1. **DEVELOPMENT CASE**

|  |  |  |  |
| --- | --- | --- | --- |
| **Discipline** | **Artifact** | **Iteration 1** | **Iteration 2** |
| **Business Modeling Requirements** | Domain Model  Use- Case Model | s  s | r  r |
| Vision  Supplementary Specification | s  s | r  r |
| **Design** | Glossary  Design Model | s | r  s |