**April**

8

National University of Singapore (NUS)

08

**Fall**

**Team Number:** 44

**Team Members:**

Melvin Lee Wei Ming A0087914U

Muhammad Hizam Bin Kamis U095858L

Tioh Wei Lun U096988R

Veeraya Pupatwibul A0069637N

Event Manager Pro

CS2103 – Developer’s Guide

**Contents**

**1. Introduction**

* 1. Source Code
  2. Compiling Instructions
  3. Login Instructions

**2. Project Scope**

* 1. Product Specifications
  2. Gantt Chart

1. **Requirements Modeling**
   1. Use Case Model
2. **Design Models**
   1. Architecture
   2. Class Diagram
   3. Sequence Diagram
   4. Database Model
3. **Developer Notes**
   1. Overview
   2. Model-View-View-Model (MVVM)
   3. Database Server and Framework
   4. Testing Framework
   5. Project / Directory Structure
   6. Third-Party Resources
   7. Coding Conventions
4. **Test Plan and Test Cases**
   1. Overview of Test Strategy
   2. Unit Testing
   3. General Testing
5. **Future Development Iterations**
6. **Appendix**

**1. Introduction**

This developer’s guide describes V0.2 of Event Manager Pro. This document is an update to V0.1 and outlines improvements made to the application. It also provides the framework for the application.

* 1. **Source Code**

The development team uses Github for version control. The source code is open source and can be obtained from the following link:

<https://github.com/sagittaros/EventManagerModel>

You may get instructions on how to fork the repository from the following link: <http://help.github.com/fork-a-repo/>

**1.2 Compiling Instructions**

Using Visual Studio IDE:

* Open the “EventManagerPro.sln” file inside the main folder
* Once inside the visual studio IDE, click on Build Solution (F6)
* Lastly click Start without Debugging (ctrl + F5) to run the program

**1.2 Login Instructions**

For V0.2, the database contains the following accounts. Use any of the following accounts to log in:

**Username: U095858L Password: asdqwe**

**Username: A0087914U Password: asdqwe**

**Username: A0069637N Password: asdqwe**

**Username: U096988R Password: asdqwe**

**2. Project Scope**

**2.1 Product Specifications (deliverables from Iteration 1)**

In V0.2, we have improved the framework of V0.1 Event Manager Pro and added more functionality to the application. The list of functionality and specifications present are as follows:

At the Main Screen:

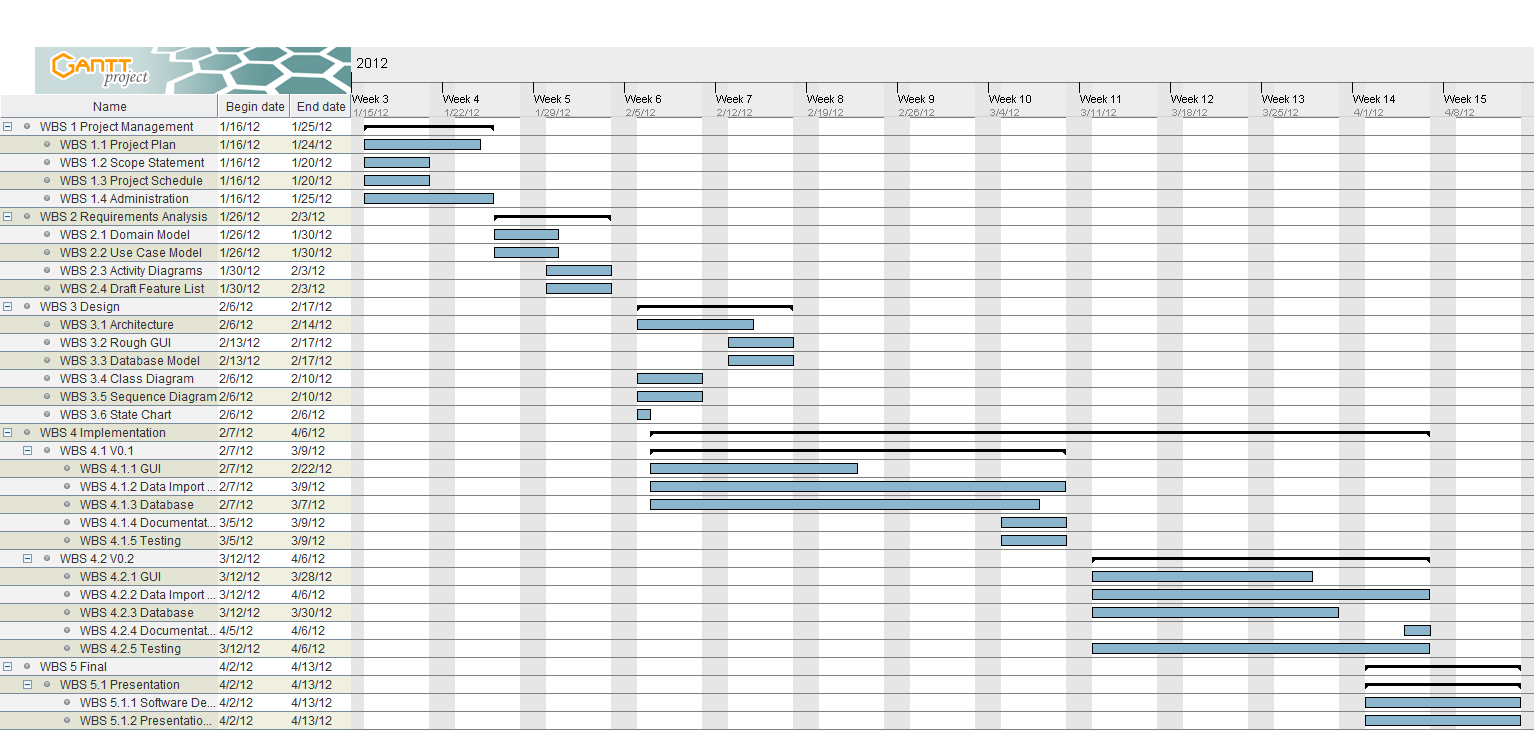
* The list of public events are displayed
* Any current NUS Student is able to log in into the system by using their account

There will now be two types of Logged In user; Event Planner and Participant. At the Home and New / Edit Event Screen:

* Event Planner is able to create a new event and input the following details and attributes for the event:
  + Name
  + Capacity
  + Description
* Event Planner will be given an option to promote and publish their event at the Front Page
* Event Planner is able to delete and edit his events
* Event Planner is able to edit the itinerary for his created events and choose a suitable venue. He is also able to choose the set the start and end time for a programme in the itinerary
* Event Planner is able to use the built in budget manager to plan the expenses
* Both Event Planner and Participants are able to view all upcoming events as well as the events that he signed up for at the Home Screen
* Participants are able to register and de-register for the events
* Event Planner can view and edit the guest list

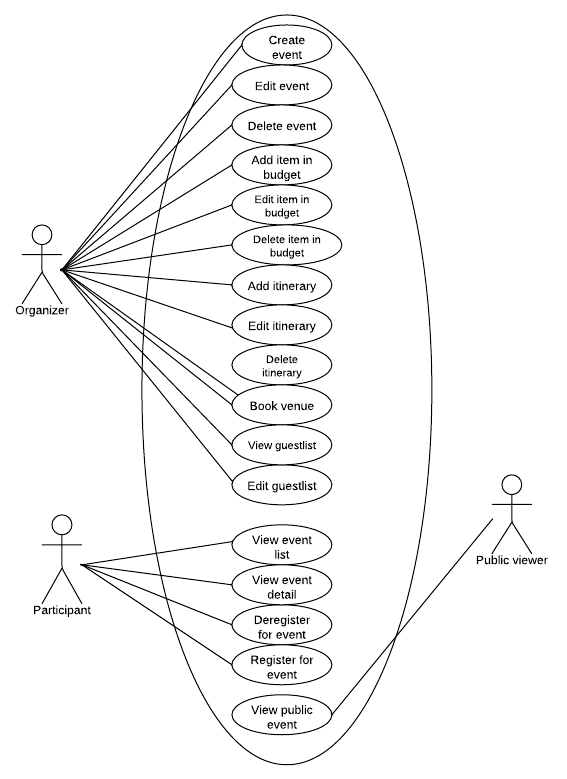
Other Features:

* Some information for an event cannot be edited three days from the date, which includes Event Name, Capacity and Venue.
* The system will indicate whether the venue is currently booked on the requested timeslot, as well as if the venue is unable to accommodate for the specified event capacity
* The system will indicate if the registration for any event is full
* The system will indicate to the Event Planner that he has exceeded his budget

**2.3 Gantt Chart (Work Breakdown Structure)**

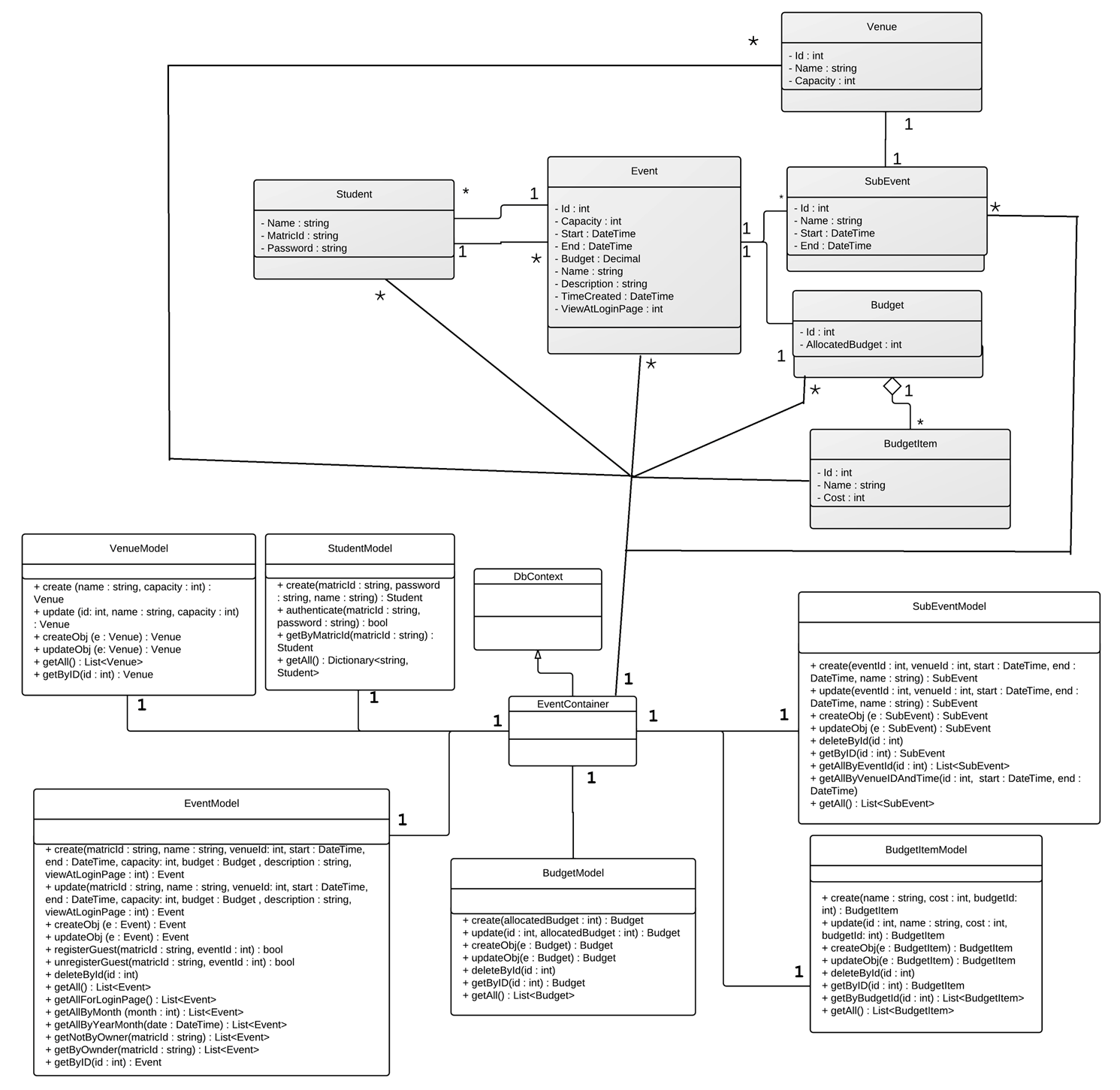
**3. Requirements Modeling**

**3.1 Use Case Model (Revised)**



**4. Design Models**

**4.1 Class Diagram**

****

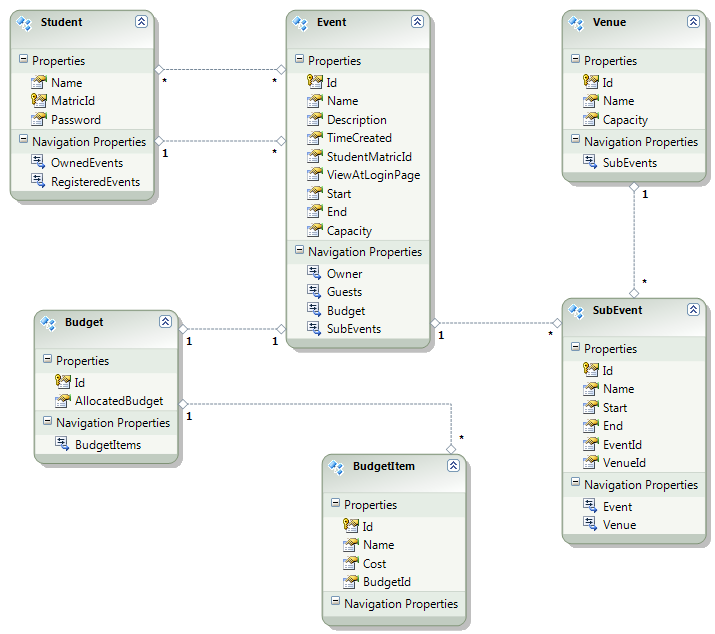
**4.2 Architecture**



**4.3 State Machine Diagram**

*To be updated(?)*

**4.4 Database Model**

**

**5. Developer Notes**

**5.1 Overview**

The development team has decided to use Windows Presentation Foundation (WPF) as the main platform, with C# as the primary programming language, for the development of this application. Visual Studio 2010 is used in almost every aspect of the development of the application, including database modeling and unit testing. Microsoft Expression Blend 4 is used occasionally for the development of the user interface (UI).

**5.2 Model-View-View-Model (MVVM) Pattern**

To fully leverage on Windows Presentation Foundation’s binding capabilities, the application is developed under the Model-View-View-Model (MVVM) pattern. This pattern separates the UI logic and the data logic, which are handled by the View and the Model respectively, and are connected with the ViewModel.

The application does not use a full framework (eg. [PRISM](http://compositewpf.codeplex.com/)) to implement the design pattern. Instead, it uses a simple framework derived from an [example](http://tsells.wordpress.com/2010/06/02/wpf-model-view-viewmodel-m-v-vm-example/) which creates base classes for each layer, for example: ViewModelBase and ModelBase, to be inherited by their respective layer classes.

As the domain object classes generated by the database layer are only usable for the database entities, new Model classes is created to encapsulate these existing object classes, and at the same time, derive new properties from existing ones to work with the application UI, via the ViewModel.

This code example shown below is in violation of the MVVM pattern, which involves creating and accessing a Model object in the View:

**Wrong Code Example (Views/EventView.xaml.cs):**

public partial class EventView : Page

{

// Constructor

public EventView()

{

EventModel myEvent = new EventModel();

this.eventNameBox.Text = myEvent.Name;

The correct approach is to create the Model object under the ViewModel, create a public class property which exposes the object, and eventually binding the object to the View.

**Correct Code Example (ViewModels/EventViewModel.cs):**

public class EventViewModel : BaseViewModel

{

private EventModel \_event;

public EventModel Event

{

get { return this.\_event; }

set

{

this.\_event = value;

this.NotifyPropertyChanged("Event");

}

}

}

**Correct Code Example (Views/EventView.xaml):**

<TextBox Text="{Binding Path=Event.Name, UpdateSourceTrigger=PropertyChanged, Mode=TwoWay, ValidatesOnDataErrors=True, ValidatesOnExceptions=True}”>

**5.3 Database Server and Framework**

The application uses SQL Server CE (Compact Edition 4.0) to store information within the database. Developers are required to have SQL Server CE 4.0 installed to develop and deploy the application. To communicate between the database and the WPF platform, the development team has implemented a Database Abstraction Layer under the ADO.NET Entity Framework 4.0 to utilize Object-Oriented concepts as a database.

**5.4 Testing Framework.**

Unit testing is automated using Unit Testing Framework provided by Visual Studio Ultimate Edition. Unit testing is carried out for all models in EventManagerPro.DBLayer.

**Test code example taken from Test/StudentModelTest.cs :**

/// <summary>

///A test for authenticate

///</summary>

[TestMethod()]

// authenticate with correct matricId and password

public void authenticateTest()

{

string matricId = "test";

string password = "test";

bool expected = true;

bool actual;

actual = StudentModel.authenticate(matricId, password);

Assert.AreEqual(expected, actual);

}

**5.5 Project / Directory Structure**

Our application is developed over three Visual Studio projects, followed by folders to organize the classes:

* **EventManagerPro.DBLayer** (Database Abstraction Layer)
  + **DomainModels**  
    Contains classes which fetches data from the database
* **EventManagerPro** (Main Application Project)
  + **Controls**Custom [User Controls](http://msdn.microsoft.com/en-us/library/cc294992.aspx) used by the application
  + **Converters**  
    Classes implemented from the [IValueConverter](http://msdn.microsoft.com/en-us/library/system.windows.data.ivalueconverter.aspxoFbzYVStg) interface to parse binding data to be suitable for use in the UI
  + **Models**  
    Model classes from the MVVM pattern
  + **ViewModels**ViewModel classes from the MVVM pattern
  + **Views**View classes from the MVVM pattern, which contains both XAML and C# files
* **Test** (Unit Testing)

**5.6 Third-Party Resources**

The application uses the following third-party resources for development:

* [Elysium Theme](http://elysium.codeplex.com/) (Free)  
  To implement the Metro design across the application
* [Microsoft Entity Framework 4.0](http://nuget.org/packages/entityframework) (Free)  
  Microsoft's recommended data access technology for new applications
* [Metro Studio 1](http://www.syncfusion.com/downloads/metrostudio) (Free)  
  Icon library for WPF applications

**5.7 Coding Conventions**

The development team has adapted Microsoft’s [C# Coding Conventions](http://msdn.microsoft.com/en-us/library/ff926074.aspx) as primary guidelines for the development of this application, as well as the following additional guidelines:

* Private class members have an underscore before its variable name   
  eg. \_myEvent
* Class methods can be named either by Camel Casing or Pascal Casing  
  eg. GetEventName() or getEventName()

In addition to the normal commenting, the development team has also used the code comment [Web Report](http://msdn.microsoft.com/en-us/library/tkxs89c5.aspx) format to document class’ properties and methods.

**6. Test Plan and Test Cases**

**Design of test cases**

The principle of equivalence partitioning is used in designing the test cases to make sure that possible range of inputs is covered yet the test can be done efficiently.

**Unit testing**

Unit testing is carried out for all models in EventManagerPro.DBLayer. This is to make sure that the interaction with the database works as it should and to ensure that they work when integrated with other functions. The database of the Test project is pre-populated with a specific set of data just for the purpose of testing and it is separated from the database used in the main program.

**System testing**

System testing is done manually via interaction with the UI. We strive for branch coverage in system testing and to cover all possible edge cases that may happen in real usage of the program.

Below are the different cases used in system testing:

|  |  |  |  |
| --- | --- | --- | --- |
| Use Case | Test Case No. | Test Case Description | Expected Results |
| Log in | LI1 | User attempts to log in with the correct username and password  **username:** test  **password:** test | this.\_validateUser(matricId, password) == true  Log-in successful and user is automatically redirected to EventsScreen |
| Log in | LI2 | User attempts to log in with incorrect username and password  **Username:** abc  **Password:** abc  **Username:** test  **Password:** (blank)  **Username:** (blank)  **Password:** test  **Username:** (blank)  **Password:** (blank) | this.\_validateUser(matricId, password) == false  Log-in unsuccessful and an “Authorization failed” message box pop up with a text saying “Your matriculation number and password were not recognized. Please check and try again.” |
|  |  |  |  |
| Log out | LO1 | Attempt to logout from EventsScreen page | Log-out and redirect to log-in page |
| Log out | LO2 | Attempt to logout from Add Event page | Log-out and redirect to log-in page |
| Log out | LO3 | Attemp to logout from Edit Event page | Log-out and redirect to log-in page |
|  |  |  |  |
| View upcoming events on Log-in Page | VL1 | Toggle month view using next and previous button | Next: this.upcomingEventsFilter.SelectedIndex++  Previous:  this.upcomingEventsFilter.SelectedIndex—  Drop down (event filter by month) changes to the desired month  Panel show lists all events for the desired month or if there are no upcoming events for that month, the panel shows the message “There are no upcoming events happening this month.” |
| View upcoming events on Log-in Page | VL2 | User attempts to view events from previous month by pressing “Previous” while the page is showing events for current month | this.upcomingEventsFilter.SelectedIndex == 0  Nothing happens. |
| View upcoming events on Log-in Page | VL3 | User attempts to view events that will happen more than 1 year from now by pressing “Next” | this.upcomingEventsFilter.Items.Count == this.upcomingEventsFilter.SelectedIndex  Nothing happens. |
| View upcoming events on Log-in Page | VL4 | User creates an event that is not visible to public. | Events not shown in Upcoming Event panel on the Log-in page |
| View upcoming events on Log-in Page | VL5 | User creates an event that is visible to public. | Event shown in Upcoming Event panel on the Log-in page. |
|  |  |  |  |
| Add Event | AE1 | User attempts to add event with name “Test Event” (will be called “Test Event” from now on) with itinerary, valid budget value, budget item and capacity > 0. | User redirected to EventsScreen page. The created event will be listed under “My Event” with the correct details. |
| Add Event  \* Event must have itinerary | AE2 | User attempts to add “Test Event” but without itinerary. | The field containing wrong/required value is highlighted in red. The save button is disabled. The event can’t be saved. |
| Add Event  \* Budget cannot be < 0 | AE3 | User attempts to add “Test Event” but with budget < 0 | The field containing wrong value is highlighted in red. The value is not accepted and will not be shown in the field. A default positive value will be displayed instead. The save button is disabled. The event can’t be saved. |
| Add Event  \* Capacity cannot be <= 0 | AE4 | User attempts to add “Test Event” but with capacity <= 0 | The field containing wrong/required value is highlighted in red. The save button is disabled. The event can’t be saved.  If the input value is a negative number, the value is not accepted and will not be shown in the field. A default positive value will be displayed instead. |
| Add Event  \* Budget item price cannot be < 0 | AE5 | User attempts to add “Test Event” but with budget item price < 0 | The field containing wrong/required value is highlighted in red. The save button is disabled. The event can’t be saved. |
| Edit Event  \* Budget can be exceeded. | AE6 | User attempts to add budget item such that the total costs exceed the budget | Event successfully edited in the database. When the user goes to the edit event page again, a warning label will notify user that the budget has been exceeded. |
| Add Event  \* Cannot book venue that is already booked | AE7 | User attempts to add “Test Event” but with a venue that is already booked | The field containing wrong/required value is highlighted in red. The save button is disabled. The event can’t be saved. |
| Add Event | AE8 | User attempts to add “Test Event” with two itinerary both at the same venue and time. | This is allowed. User redirected to EventsScreen page. The created event will be listed under “My Event” with the correct details. |
|  |  |  |  |
| Edit Event | EE1 | User changes the name of the “Test Event” to “Dummy Event” | User redirected to EventsScreen page. The name displayed for that event will be “Dummy Event”  Correct event details will be displayed on the EditEventScreen when the user tries to edit again. |
| Edit Event | EE2 | User changes the time of the itinerary program “Test Event” to another day | User redirected to EventsScreen page. The date displayed for that event on EventsScreen will be updated.  Correct event details will be displayed on the EditEventScreen when the user tries to edit again. |
| Edit Event | EE3 | User changes the capacity of the “Test Event” to 100 | User redirected to EventsScreen page.  Correct event details will be displayed on the EditEventScreen when the user tries to edit again. |
| Edit Event | EE4 | User changes the venue of the “Test Event” to LT16 | User redirected to EventsScreen page.  The venue displayed for that event on EventsScreen will be LT16.  Correct event details will be displayed on the EditEventScreen when the user tries to edit again. |
| Edit Event | EE5 | User changes the budget of the “Test Event” to 100 | User redirected to EventsScreen page.  Correct event details will be displayed on the EditEventScreen when the user tries to edit again. |
| Edit Event | EE6 | User delete registered the participant from the guest list | User redirected to EventsScreen page. Correct event details will be displayed on the EditEventScreen when the user tries to edit again. |
| Edit Event | EE7 | User adds more program to the itinerary | User redirected to EventsScreen page. Correct event details will be displayed on the EditEventScreen when the user tries to edit again. |
| Edit Event  \* Event must have itinerary | EE8 | User deletes all itinerary | Event cannot be saved. Save button is disabled. |
| Edit Event  \* Budget cannot be < 0 | EE9 | User attempts change the budget to < 0 | The field containing wrong value is highlighted in red. The value is not accepted and will not be shown in the field. A default positive value will be displayed instead. The save button is disabled. The event can’t be saved. |
| Edit Event  \* Capacity cannot be <= 0 | EE10 | User attempts to change capacity to 0 or < 0 | The capacity field is highlighted in red. The save button is disabled. The event can’t be saved.  If the input value is a negative number, the value is not accepted and will not be shown in the field. A default positive value will be displayed instead. |
| Edit Event  \* Budget can be exceeded. | EE11 | User attempts to add budget item such that the total costs exceed the budget | Event successfully edited. When the user goes to the edit event page again, a warning label will notify user that the budget has been exceeded. |
| Edit Event  \* Cannot book venue that is already booked | EE12 | User attempts to add “Test Event” but with a venue that is already booked | The field containing wrong/required value is highlighted in red. The save button is disabled. The event can’t be saved. |
| Edit Event  \* Some details can’t be change <3 days before the actual event. | EE13 | User attempt to change capacity and itinerary of the event that will happen within 3 days from now. | Capacity and itinerary fields are disabled. These can’t be changed. |
|  |  |  |  |
| Delete Event | DE1 | User clicks Delete Event followed by confirming the delete action (clicks Yes when prompted) | Events deleted from database and from My Events List.  A label “Your event has been successfully deleted.” is made visible. |
|  |  |  |  |
| View Event | VE1 | User is on the EventsScreen page and navigate between Upcoming Event and My Events | List of events that are created by others are shown on the Upcoming Event tab while list of events created by the users are shown on My Events tab. Events that has already passed are not shown under Upcoming Event. |
|  |  |  |  |
| Register Event | RE1 | User is on the EventsScreen page and navigate between Upcoming Event and My Events | List of events that are created by others are shown on the Upcoming Event tab while list of events created by the users are shown on My Events tab. Register button changes to an unregister button. |
| Register Event | RE2 | User is viewing event detail (click “info” button) and click register | List of events that are created by others are shown on the Upcoming Event tab while list of events created by the users are shown on My Events tab. Register button changes to an unregister button. |
|  |  |  |  |
| Unregister Event | UE1 | User is on EventScreen page under Upcoming Events and click unregister. | The event no longer shows on My Events tab for the student. And the student is no longer listed under Guest List. Unregister button changes to register button. |
| Unregister Event | UE2 | User is on EventScreen page under My Event and click unregister. | The event no longer shows on My Events tab for the student. And the student is no longer listed under Guest List. Unregister button changes to register button. |
| Unregister Event | UE3 | User is viewing event detail (click “info” button) and click unregister | The event no longer shows on My Events tab for the student. And the student is no longer listed under Guest List. Unregister button changes to register button. |
|  |  |  |  |
| View event detail | VE1 | User click “info” on EventScreen page under My Event and Upcoming Event. | User re-directed to a page containing the correct event details. |

**7. Future Development Iterations**

**Additional features to be implemented in future iterations:**

* Break down the Start and End time options in itinerary to half hourly instead of hourly
* Implement a built in “Note Taking” feature to enable Event Creator to write reminders and notes about their events
* Implement a built in “Scheduling and Timeline” feature to allow Event Creator to plan and schedule the development of his event
* Enable Event Creator to upload Event Poster or Picture for their events
* Implement a notifications system to better notify users on the changes for the events
* Allow more than one user to edit an event
* Improve performance by reducing the number of calls to the database

**8. Appendix**