Event Planner

SEIS 635 – Semester Project

May 14, 2013

Ross Anderson / Ben Passe - (Group 1)

Contents

[Project Plan 3](#_Toc356153834)

[Project Description 3](#_Toc356153835)

[Proposed Plan 3](#_Toc356153836)

[Actual 4](#_Toc356153837)

[Plan Comparison 4](#_Toc356153838)

[Requirements 5](#_Toc356153839)

[Use Case Diagram 5](#_Toc356153840)

[Fully Dressed Use Cases 5](#_Toc356153841)

[Manage Event 5](#_Toc356153842)

[Manage Wish List 6](#_Toc356153843)

[Send Invitations 7](#_Toc356153844)

[Return Attendance Response (RSVP) 8](#_Toc356153845)

[Analysis 9](#_Toc356153846)

[Domain Model 9](#_Toc356153847)

[System Sequence Diagram 10](#_Toc356153848)

[Design 11](#_Toc356153849)

[Class Diagram 11](#_Toc356153850)

[Implementation 11](#_Toc356153851)

[Technologies Used 11](#_Toc356153852)

[Organization 12](#_Toc356153853)

[Communication and Staying in Sync 12](#_Toc356153854)

[Technology 12](#_Toc356153855)

[Strategy 13](#_Toc356153856)

[Lessons Learned 13](#_Toc356153857)

[Things to do Differently Next Time 13](#_Toc356153858)

[Future Development 13](#_Toc356153859)

# Project Plan

## Project Description

The party planner application will allow a user to setup events, with the features including lists of invitees, wish lists (with items segmented to target specific users), wish list item information (including links to retail websites stocking the item), and vendor information (food and entertainment).  The application will also allow the user to setup an event’s RSVP list, location information, and the actual invitation.  This application will be developed as a desktop application in C# using Visual Studio, but ideally could be ported to a web application to allow more user interaction.

## Proposed Plan

|  |  |  |
| --- | --- | --- |
| **Event/Milestone** | **Start Date** | **End Date** |
| Project Proposal | 3/2/2013 | 3/23/2013 |
| Requirements Gathering | 3/23/2013 | 4/6/2013 |
| System Analysis and Design | 3/26/2013 | 4/13/2013 |
| Progress Report | 4/19/2013 | 4/20/2013 |
| System Implementation | 4/6/2013 | 5/15/2013 |
| Testing | 5/3/2013 | 5/17/2013 |
| Paper w/ All Artifacts | 5/10/2013 | 5/18/2013 |
| Class Presentation | 5/18/2013 | 5/18/2013 |

## Actual

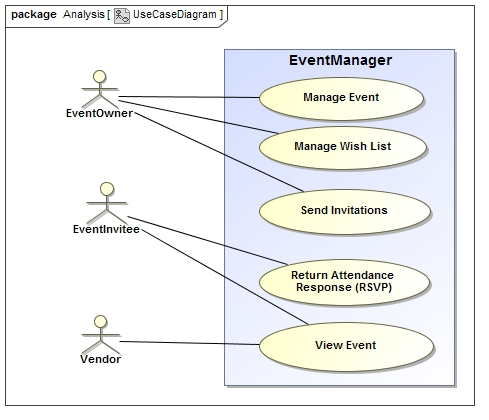
|  |  |  |
| --- | --- | --- |
| **Event/Milestone** | **Start Date** | **End Date** |
| Project Proposal | 3/2/2013 | 3/23/2013 |
| Requirements Gathering | 3/23/2013 | 4/6/2013 |
| System Analysis and Design | 3/26/2013 | 4/13/2013 |
| Progress Report | 4/19/2013 | 4/20/2013 |
| System Implementation | 4/13/2013 | 5/15/2013 |
| Testing | 5/10/2013 | 5/17/2013 |
| Paper w/ All Artifacts | 4/25/2013 | 5/18/2013 |
| Create Presentation | 5/5/2013 | 5/17/2013 |
| Class Presentation | 5/18/2013 | 5/18/2013 |

## Plan Comparison

Most of the project was executed according to plan, except for a few tasks. The development started about a week later, primarily due to the late choosing of the implementation technology. The testing also started about a week later, since it couldn’t start yet because of the late start on the development. An additional task was also added, the creation of the final presentation.

# Requirements

## Use Case Diagram



## Fully Dressed Use Cases

# Manage Event

Primary actor: Event owner

Preconditions:

1. User has previously registered

Stakeholders:

1. Event owner
2. Others invited by event owner

Success Condition:

1. Event is saved with all required information and other information as deemed relevant by owner

Primary Flow:

1. User signs in to application
2. User selects to create event
3. User adds location, time, and other information to event
4. User selects to save event

Alternatives

2. User selects to update existing event

4. User decides to cancel event before saving

Frequency of occurrence:

Nearly continuous

Special requirements:

None

# Manage Wish List

Primary actor: Event owner

Preconditions:

1. User has previously registered

Stakeholders:

1. Event owner
2. Others invited by event owner

Success Condition:

1. Wish list is saved with at least one item

Primary Flow:

1. User signs in to application
2. User selects to create wish list
3. User adds item to wish list
4. User selects to save wish list

Alternatives

2. User decides to update existing wish list

3. User adds item information including:

3a. Retail location

3b. Retail price

3c. Target invitee

Frequency of occurrence:

Nearly continuous

Special requirements:

None

# Send Invitations

Primary actor: Event owner

Preconditions:

1. User has previously registered
2. User has previously created event
3. Invitees have email addresses

Stakeholders:

1. Event owner
2. Others invited by event owner

Success Condition:

1. Invitations sent to other users

Primary Flow:

1. User signs in to application
2. User selects previously created event
3. User adds other users to an invitation list using email addresses
4. User creates invitation
5. User selects to send invitation to list of invitees

Alternatives

4a. User selects previously created invitation

4b. User selects template for invitation

5a. User selects to save invitation list and exit

Frequency of occurrence:

Nearly continuous

Special requirements:

None

# Return Attendance Response (RSVP)

Primary actor: Event invitee

Preconditions:

1. User has email address
2. User has received invitation

Stakeholders:

1. Event owner
2. Invitee

Success Condition:

1. Application stores the response for from the invitee
2. Event owner can view the response

Primary Flow:

1. User signs in to application
2. User selects to respond to invitation
3. User selects response to save

Alternatives

1a. User selects link in email to auto-respond

3a. User views wish list available for event

Frequency of occurrence:

Nearly continuous

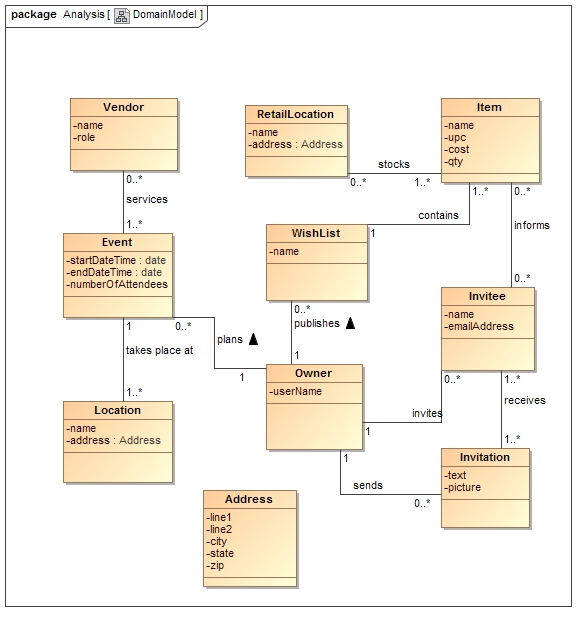
Special requirements:

None

# 

# Analysis

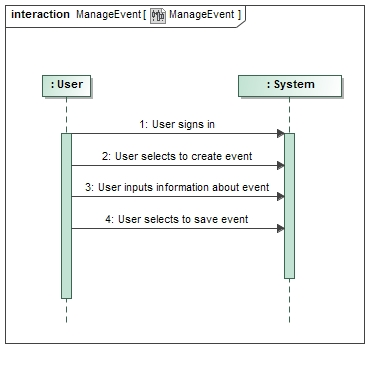
## Domain Model



## System Sequence Diagram

It was decided after starting the system sequence diagrams that most of them would look very similar, and thus would not provide any useful artifacts to aid in design or development. Thus, only one system sequence diagram was created, as seen below.

System Sequence Diagram for Use Case - Manage Event:



# Design

## Class Diagram

# 

# Implementation

## Technologies Used

For requirements, analysis, and design, MagicDraw was the primary tool used. MagicDraw definitely satisfied all of the specified needs for this project. The tool was easy to use, especially after completing the class assignments in MagicDraw.

When the project started, Java with Swing was the intended technology for the actual development of the application. However, due to division of tasks between the team and prior development experience, the decision was made to switch to using Microsoft C#. Since there was at least some experience with visual features of C# and no experience with Swing, it made sense to switch, especially since the team member with the C# experience would complete more development efforts.

The technology used for keeping documents and source code in sync was GitHub. More information on that is in the section Organization-Technology (Technology).

Since C# was the primary programming language used for the implemented, Visual Studio was the IDE utilized for the development. Visual Studio turned out to provide a relatively easy methodology for creating the GUI necessary, including tying actions to the implementation. In addition, the quick simulation of the application at any time proved to be very useful.

# Organization

There were several methods utilized for staying organized throughout the project. First of all, as mentioned above, GitHub was used for organizing and tracking work that was occurring. Using GitHub allowed the team members to see what the other person was doing. For division of responsibilities, both team members worked collaboratively on the requirements, Ben was the lead on the analysis and design, and Ross was the lead developer. By no means did either team member complete all tasks by himself, as both provided plenty of input and production in all phases. In addition, dividing up the tasks like this actually proved to be more effective since a team member could take the lead and make most decisions without having to discuss at length about each minor detail.

# Communication and Staying in Sync

## Technology

GitHub was used to keep artifacts and source code in sync. There was some experience with Git and GitHub before this project started, so there was not a large learning curve. However, using the GitHub client locally did provide some challenges as the merge tool was not good at all. Because creating a GUI in Visual Studio can indirectly create a lot of data behind the scenes, merges can end up being very difficult to perform. Thus, there was some copying and emailing of files done so that the merge could be done locally using different merge tools instead of the tool provided by GitHub.

Besides GitHub, we used Gmail for staying in sync throughout the project. We only needed to meet outside of class briefly due to the separation of tasks.

## Strategy

The strategy for staying in sync was to communicate often through email and to clearly comment what changes were made in the documents and source code. In addition, the time allowed in class for team project work was greatly appreciated and utilized, which allowed for the team members to not have to meet outside of class.

# 

# Lessons Learned

## Things to do Differently Next Time

First, we would pick the technology to use much earlier. Than we would be able to incrementally add features to the application at a steadier pace, instead of tail-heavy. We originally picked Java/Swing, but switched to C#. We didn’t know which technology to go with at the beginning because neither of us had much experience with desktop UI applications. It was difficult for us to choose a technology also because we are both s in different languages, Java and C#. Second, we would choose an application topic much sooner. It took us a few weeks just to pick a topic, and that may have slowed us down.

## Future Development

If we were to continue to develop the application, there are a few major features we would add initially. First, we would convert the application to be a web application. The application would be created as a web site that could additionally resize correctly for mobile applications. If there would be significant use, than we would create a mobile app, first for Android. Second, we would add email notifications for several events, including RSVP’s and wish list updates. Lastly, we would have links to retail stores that could potentially do automatic ordering. Then invitees for events could simply select to order an item from an event’s wish list, instead of having to go investigate and search for the item at stores themselves. This change and others would provide a much more seamless flow to the application so that users would feel as though they’re searching and ordering from a virtual registry at a store.