1. Project Description and intended use

1.1 Problem Statement

On average, wedding planners charge between $2,700 and $5,400 to plan a wedding from start to finish (1). The cost of a wedding planner often hovers around ten to twenty percent of the national average wedding cost of $27,000 in the U.S. today (2). Many wedding planners are hired solely for consulting purposes, which can cost couples between $50 and $150 an hour. Approximately 57 percent of couples who got married in 2009 used a wedding planner (1), which shows that couples are often willing to spend extra money to get help while planning their wedding. However, this expense only seems greater in the recent economy.

Collectively, Americans have lost nearly eight million jobs from the beginning of the recession in 2007 to 2010 (3). The unemployment rate in recent months, hovering around ten percent, is the highest since the early 1980s (4). The economic strain has been felt by all, and as a result, many couples are forced to cut extra spending and find alternative ways to help plan and budget their weddings.

Some of these alternative ways include using Internet tools. One popular site, theknot.com, is a wedding planning website that offers a variety of tools, guides, and information for couples who want to plan their own wedding (5). Another popular site, mywedding.com, is a website that provides resources to help couples plan and budget their wedding (6). Both of these sites are in a long list of similar tools that aggregate resources online and provide assistance to engaged couples working without a wedding planner. Couples planning their own wedding will often recruit a friend to help them as well. Often the maid of honor or another close friend of the bride will assist in planning, decorating, and other duties. Each of these examples shows that couples often seek help and utilize tools while planning their wedding.
1.2 Description of the Solution

The primary focus of our application is to emulate the services of a wedding planner. Our planner will guide and assist couples in their planning with timelines, budgets, guest lists and all the events that compose their wedding. It will have a simple, easy to use interface and will help couples track the essential details of their wedding. This web-based wedding planner will simplify the process of planning a wedding, by eliminating many of the stresses of planning a wedding.

Our solution is to create a tool that engaged couples can use every day to assist them in planning their wedding. It will provide couples a way to track essential details of their wedding by the use of a checklist type design. The checklist will guide couples through all the important decisions, reservations, and appointments that they must make. Couples will also have the ability to construct and maintain a guest list for the wedding, rehearsal dinner, wedding reception and any other events they want to include. Budget is important to any wedding, so our solution will provide couples a cost range for every detail of the wedding.

This tool is designed so couples can utilize all these tools in their own way so that they have the ability to plan every detail involved ensuring them that they will have a dream wedding.

1.3 User Profile

The primary user of this application would be engaged couples who seek assistance in planning a wedding by themselves. These couples will most likely have a limited budget and thus look to save money by finding an alternative to a wedding planner. These users will be looking for help to track costs and spending so they can stay within their budget. They might also be unsure of all the subtle details to properly plan their wedding.
2. Design Protocols

2.1 Case Diagram

Figure 1 is our Use Case Diagram. There are only two actors involved with this application, the user and the system. The user will interact with the application by editing, updating, creating and viewing content. The system will handle the user’s interactions in a fast and efficient manner. The system will also provide default data for each user to work from when starting to use the application, within the checklist and budget features. During the registration process, the user will login to Facebook and approve our application. The system will then query the Facebook Open Graph for the necessary information to identify the user within the application. This is used to populate the UserID as shown in the upcoming database diagram.
Figure 1: Case Diagram
2.2 Database Diagram

Figure 2 is our database diagram. The structure of the database is for the four main components of the application; checklist, budget, guest list, and events. The overall design of our database reflects each entity’s relationship with the base wedding entity. By following Third Normal Form, our database design allows flexibility without being subject to many update or insertion anomalies. The only tables that were not used are the User_Reservations and Reservation_Types tables. This feature was moved beyond the scope of our project during development.

Figure 2: Database Diagram
2.3 User Interface

User interface for our application focused on efficiency and simplicity. To accomplish this, we maintained good design and functionality practices throughout the application development lifecycle. Here are the key interface requirements we followed:

- No more than 1-2 clicks to get to any major portion of the application.
- Quickly and easily edit any text items that belong to the user.
- Intuitive and responsive UI elements (no UI stuttering or excessive POSTs that interrupt the user workflow)
- All aspects should allow users to enter the bare minimum of information at the start
- Don’t overwhelm with extraneous instruction
- Should enjoy using the application, not have to fight against it.
- Color scheme will be mostly white with subtle color highlights on key interface elements
- All pages will be uncluttered and have ample whitespace.

All of these requirements came together during both the Tech Expo and our user validation testing. Non-technical users were able to jump right into the application and start using it with no instruction.

2.4 Architecture Diagram

The architecture of our application follows the Model-View-Controller pattern. This minimizes the coupling between application logic and data presentation (11). This model divides the application into three layers, as shown below in Figure 3:

**Model:** This domain contains application logic and specific functions dealing with data.
**View:** Responsible for presentation of data and forwarding user interactions to controllers
**Controller:** Accepts user requests and control the models that fulfill these requests
Figure 3: MVC Architecture

- **Model**
  - Encapsulate Application State.
  - Respond to state query.
  - Notify View Changes
  - Expose Application Functionality

- **View**
  - Allow controller to select view.
  - Renders the model.
  - Sends User gestures to controller.

- **Controller**
  - Define Application Behavior.
  - Maps user action to model update.
  - Select View For Response.

- **State Query**
- **Change notification**
- **State change**
- **View Selection**
- **User gesture**
- **Method Invocation**
- **Events**
3. Deliverables

- Create, Read, Update, and Delete Events in a wedding
  - Save notes to the added events

- View and Complete Checklists
  - Save notes to checklist items

- Master Guest List
  - Export to Microsoft Excel for address label printing
  - Manage RSVPs

- Budget
  - High level and granular views
  - Range sliders to customize values

3.1 Wish list

- Mobile Views

- Vendor suggestions

- Integrate importing Facebook friends to Guest List

- Independent Guest List per Event
4. Project Planning

4.1 Project Schedule

Our project started in autumn 2011 and ended in spring 2012. In Figure 4, we only have listed items starting from early January 2012. This is when the work towards our final deliverables actually started. The autumn quarter was used as a high level and preliminary planning time, where we determined what we might actually work on in the following time.
4.2 Project Budget

One of our requirements for this project was to keep costs as low as possible, since this will be an entirely bootstrapped project. Through our planning, we have achieved that goal, as shown in Figure 5, through ample use of freely available software and currently owned hardware. We had a built in buffer early on in our project, which proved to be wise. In the end, we spent only $68 dollars towards this project. The bulk of this cost went into hiring an editor for our project reports.

<table>
<thead>
<tr>
<th>Software</th>
<th>Budgeted Cost</th>
<th>Actual Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aptana IDE</td>
<td>$0</td>
<td>0</td>
</tr>
<tr>
<td>Visual Studio 2010</td>
<td>0</td>
<td>? (MSDN)</td>
</tr>
<tr>
<td>MS SQL Server Mgmt Studio</td>
<td>0</td>
<td>? (MSDN)</td>
</tr>
<tr>
<td>Web Browsers</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>75</strong></td>
<td><strong>$68</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hardware</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Mouse &amp; Keyboard for Tech Expo (x2)</td>
<td>25</td>
<td>18</td>
</tr>
</tbody>
</table>

4.3 Software

The software that was used for the development of our project is shown in Figure 6 below.
<table>
<thead>
<tr>
<th>Software</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual Studio 2010</td>
<td>Code development for the application</td>
</tr>
<tr>
<td>SQL Server Management Studio</td>
<td>Database Management</td>
</tr>
<tr>
<td>GitHub.com</td>
<td>Distributed version control hosted on the web</td>
</tr>
<tr>
<td>Fiddler2</td>
<td>HTTP Request monitoring</td>
</tr>
</tbody>
</table>

Figure 6: Software

4.4 Hardware

The hardware that was used for the development of our project is shown in Figure 7 below.

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toshiba Satellite L355D Laptop</td>
<td>Development, testing and demos</td>
</tr>
<tr>
<td>Desktop PC</td>
<td>Development, testing and demos</td>
</tr>
</tbody>
</table>

Figure 7: Hardware

5. Proof of Design

The following section contains a breakdown of the different functionalities and design features that satisfies the stated deliverables and proofs this is a solution to the problem. The overall function of the application is to provide users the ability to keep track of their own weddings details, such as a guest list, events, budget and a check list that users can follow to ensure that every detail of their wedding is properly planned.

For users to begin using the application they must be Facebook authorized and accept the application. To begin planning their wedding each user must register by filling out a quick and simple form. Figure 8 is the Registration form which shows; that there is a minimal amount of information that is needed and it has a simple format.
After users are authenticated and registered their personal home page which is a page that allows users to quickly navigate, review, and update their wedding details. Figure 9 below is an example of a home page, as you will see everything is laid out consistently and only contains necessary content that is easy to read and straightforward. Each of three items in the middle of the page all our links that navigate the user to the corresponding item, giving the user quick access to different of their wedding plans.
The first feature of the application is maintaining a guest list, one of the most stressful parts of planning a wedding. So giving users the ability to add guests quickly and without any hassles was our first priority when developing the functionality of the guest list. As you will see in Figure 10 below we have a nice flowing form that is simple and easy to use. We developed it so that once a guest is created the form notifies the user that the guest was saved and the form is cleared so that multiple guests can be added very quickly.
Added guests are listed on the Guest List page, Figure 11 below which also displays to the user how many total guests have been added and total RSVPs. Next to each guest in the list is a edit link which gives the user the ability to edit their guests as they please. Another thing to point out is the design, the content is easy to read, it’s straightforward and there is no clutter.
Managing RSVPs is an important feature included in the application, which simply is a tool for the user to track which guests have RSVP for their wedding. The ability for our users to mark guests attending by a simple click of a check box saves time and effort. Figure 12 below is the layout of how users will be able to utilize this tool. The list of guest to easily scroll through, the one check box for attending, the row of buttons to control the content of the list is self explanatory. All these design elements provide the user the proper information and tools to manage the RSVPs easily without any extra stress.
The other important feature with the guest list is exporting. This feature allows users to export the guest list to MS Excel. This provides users with flexibility to use the guest list for reasons outside of the application, such as creating address labels for invitations, managing seating charts, or distributing to venue sites.

The next feature in our application focuses on tracking events that are commonly associated with planning a wedding such as important meetings or parties that users are attending. Figure 13 below shows how a user enters a new event, as you can see the design elements are similar to the rest of the application, very easy to follow and require minimal information. The note section allows users to enter any kind of information or details that are important to remember about the event.
Figure 13: Creating an Event

After an event is added it will appear on the Events page, Figure 14 below, here the user can see a list of all their events that they are planning on attending. Next to each event are a few functionality tools that the user can use to manage these events. Editing allows the user to add important notes or change dates, details is so the user can view the all details of the event, and the delete so events can be removed.

Figure 14: Events page
The next big feature of our application is the budget, here we are providing a tool that helps the user understand and know the type of expenses that are involved with a wedding. The page is broken up into ten main categories that are associated with a wedding and then are broken up into smaller sub categories. Each category contains a minimum and maximum amount which start at a US average cost these amounts determine the amounts of the sub categories. So this feature really breaks down the whole wedding allowing the user to what kind of costs are involved during planning. If you refer to Figure 15 below, it shows the top portion of the budget page, where you can see the first two main categories, Attire and Ceremony but also the range sliders.

![Figure 15: Budget page](image)

These sliders provide a simple to use tool so the user can adjust the amounts to closely reflect the users own budget. This feature will help users be able to better plan and manage their budget, saving them time and money.
The last feature of our application and the most essential part of our application is the check list. The check list that we are providing is a detailed list of items to ensure that the user’s wedding is planned correctly and that nothing is left out. The list was endorsed by a certified wedding planner so it is a significant tool that will give our users the ability to really plan their dream wedding. Under each checklist item we have included the functionality to add notes so users can really manage all the planning details involved. Figure 16 below, is a portion of the checklist, but it really shows how each item is clear, easy to read. The check boxes allow the user to mark off items as they complete them and then the navigation links at the top give the user the ability to view different portions of the list.

![Check List](image)

All of these features and design elements provide our users a simple easy to use way to manage, track and coordinate wedding details on their own, instead of hiring a wedding planner. The consistency throughout the application gives the user confidence and knowledge to use this application quickly and effectively. Every page contains only the most necessary content, so
there is no extra clutter which can add stress to our already stressed out users. All the features involved in the application are reasons why wedding planners are hired and what they solely help out with. So, ultimately this application is a tool that can be used, by engaged couples that can’t afford a wedding planner, to plan their own dream wedding.

6. Testing

There are a series of use cases that we have developed for testing each feature of the application. The results are outlined below. Table 1 contains the testing scenarios for procedures involved with events, such as creating, editing and deleting.

<table>
<thead>
<tr>
<th>Table 1: Scenarios for events</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step #</strong></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

Scenario 2: Edit an event

<table>
<thead>
<tr>
<th>Step #</th>
<th>Action</th>
<th>Data / Input</th>
<th>Expected Results</th>
<th>Pass/Fail</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Change the Date and Time of the event</td>
<td>Date/Time - &quot;June 4, 2013, 7pm&quot;</td>
<td>The screen should allow the users to begin editing details of the event</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Save the event</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Scenario 3: Delete an event

<table>
<thead>
<tr>
<th>Step #</th>
<th>Action</th>
<th>Data / Input</th>
<th>Expected Results</th>
<th>Pass/Fail</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Delete next to the event that was just saved.</td>
<td></td>
<td>A message should appear onscreen, verifying that the event has been deleted</td>
<td>Pass</td>
<td>There was no message verifying</td>
</tr>
<tr>
<td>2</td>
<td>Click OK</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2 contains the testing scenarios for the procedures involved with the checklist.

<table>
<thead>
<tr>
<th>Table 2: Scenarios for the checklist</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step #</strong></td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

Scenario 2: View checklist for a specific time period

<table>
<thead>
<tr>
<th>Step #</th>
<th>Action</th>
<th>Data / Input</th>
<th>Expected Results</th>
<th>Pass/Fail</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select 3-4 weeks on the Checklist page</td>
<td></td>
<td>Each item will have the details of what needs to be done and the ability to mark complete as well as a note section</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Scroll through the whole list of items</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Scenario 3: Mark a checklist item complete

<table>
<thead>
<tr>
<th>Step #</th>
<th>Action</th>
<th>Data / Input</th>
<th>Expected Results</th>
<th>Pass/Fail</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Check on the checkboxes next to a checklist item</td>
<td></td>
<td></td>
<td></td>
<td>No verifying message that change was saved</td>
</tr>
<tr>
<td>2</td>
<td>For the same checklist item click the checkmark</td>
<td></td>
<td></td>
<td></td>
<td>No verifying message that change was saved</td>
</tr>
</tbody>
</table>

20
Table 3 contains the testing scenarios for the guest list. It includes adding, viewing, updating and deleting a guest.

<table>
<thead>
<tr>
<th>Step #</th>
<th>Action</th>
<th>Data / Input</th>
<th>Expected Results</th>
<th>Pass/Fail</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1. Preconditions</td>
<td></td>
<td>Home page to display.</td>
<td>Pass</td>
<td></td>
</tr>
<tr>
<td></td>
<td>2. Navigate to the &quot;Guestlist&quot; page</td>
<td></td>
<td>The Guestlist page will load and user will see a list of names and a section to add additional names.</td>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>

**Scenario 1 - Add a new guest**

1. Enter name, address, city and state.
2. Click "Add"

A message should appear onscreen, verifying that the guest has been added.

**Scenario 2 - View main guestlist**

Scroll through the names that are displayed in the guestlist.

Each guests name should be displayed as a list. Next to each name there should be a 'X', which is used to remove names from the list.

**Scenario 3 - Update guest information**

1. Click on a guests name from the list.
2. Change their name.
3. Click "Add"

Guest information will appear in the textboxes in the add a guest section.

**Scenario 4 - Remove a guest**

Click the 'X' next to the guest that you just added/or one of the other guests.

A message should appear onscreen, verifying that the guest has been removed.

Table 3: Scenarios for the guest list

Table 4 contains the testing scenarios for the procedures involved with the budget, such as what is excepted when implementing and editing a budget.

<table>
<thead>
<tr>
<th>Step #</th>
<th>Action</th>
<th>Data / Input</th>
<th>Expected Results</th>
<th>Pass/Fail</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Preconditions</td>
<td></td>
<td>Home page to display.</td>
<td>Pass</td>
<td></td>
</tr>
</tbody>
</table>

**Scenario 1 - View Budget**

1. Navigate to Budget

All categories, amounts and range slider should display.

**Scenario 2 - Edit budget**

- Under Ceremony move the 1 range slider to change amounts.
- Type in a new minimum amount 2 for Reception.

Dollar amounts in the sub categories should change to relate to the changes made to the range amounts.

Amounts in the sub categories will change to reflect the changes made to the range amounts.

No verifying message that this change was saved.

No verifying message that this change was saved.

Table 4: Scenarios for budget

Each of these testing scenarios will be automated with scripts using the WATIR (Web Application Testing In Ruby) open source testing libraries. This will enable us to rapidly develop new features while ensuring existing functionality remains intact through regression testing.
7. Conclusions and Recommendations

7.1 Conclusion

Planning a wedding is a lot of work; making sure that every detail is planned, maintaining appointments and keeping track of guests all put considerable stress on the bride and groom. Weddings are also expensive, which adds to the stress and introduce more difficulty in planning a wedding. In order to have reliable and experienced assistance with the design, planning and management of their wedding, engaged couples will often hire a professional wedding planner. This adds another problem, since wedding planners are expensive. Wedding planning professionals charge between $2,700 and $5,400 (1), which is a considerable amount in this current economy. Our project aimed to solve this problem by providing an alternative, low cost set of tools for engaged couples to alleviate many of the stresses of planning their own wedding. The features we delivered for our project assist engaged couples in a similar fashion to an actual wedding planner. As shown throughout this report, this Wedding Planner web application saves money, reduces stress, all while assisting in planning a dream wedding.

7.2 Recommendations

At the beginning of our project, we decided that we wanted to create a simple web application that event planners could use to help plan the various events that they are hired for. The goal was to create something that would be both beneficial and logical for any event planner to use. With this goal in mind we started to think of what would be the deliverables to accomplish this goal. It was at this stage we when started to realize that weddings were always the focus of our meetings. After some discussions between ourselves and some guidance from our advisor, it was clear that we needed to narrow the scope of our project to just wedding
planning. We then discussed the impact on our initial use case and user profile artifacts. For our user profile, we settled on engaged couples planning their own wedding, that could not afford or did not want to hire, an expensive wedding planner could benefit from our project began the earlier stages of planning and development.

One of the things that we decided on rather quickly was the framework, we decided on ASP.NET MVC3 with Entity Framework. We felt that this was the best way to handle retrieving and manipulate data and to our views. Using the Razor View Engine gave us greater ability to give couples clean and stressful interfaces to use. Even though using this type of framework was fairly new to both of us, we found that it was very easy to learn and by the end of our project we felt that it was a good decision. Microsoft had several examples available on the web that allowed us to quickly get a solid understanding of the new Razor syntax for our views.

As for the deliverables that we included in our project, we really wanted to focus on the main reasons why wedding planners are hired and not just throw a bunch of features into it. We felt that this would give us the ability to develop a well defined application that is scalable, effective and solves the problem. With that being said there are some features that we feel could be added or changed to better assist couples while planning their wedding. For instance, the budget portion of our application didn’t turn out quite the way we envisioned it. So we have agreed to propose changing it, so there is a little bit more control for couples by allowing them to enter their own amounts to track their expenses. Giving couples the ability to see what type of costs our involved and to track what they are actually spending is a huge benefit. It would provide better planning and ultimately save them more money. Another feature that we have discussed and even suggested is the ability to import contact information from Facebook or
Outlook. The features that did not ‘make the cut’ for our final scope will give us items to work on when we desire to expand this application.

One thing we would have done differently given a chance to repeat this project would be build in the requirement to deploy the application to the web as soon as possible. Even though this would likely shrink our final feature set, releasing the application potentially at the end of the prototype phase would allow us to have actual users providing more legitimate feedback, rather than playing the guessing game of what might actually be useful. Another side effect of this would have been learning a wider breadth of skills around developing an application, such as search engine optimization, writing effective copy to convert visitors into users, and creating some sort of business strategy.

Overall, we both learned an incredible amount both about software development and about ourselves in such a short span of time. The lessons we learned while developing this application will stay with us both for many years to come as we venture forth as Information Technology professionals.
References


