1. Project Overview

Team Bluebrick and the Software Engineering department at RIT were tasked with creating a project hub for the Center for Student Innovation. At RIT many students take part in projects both inside and outside of class. These projects span individuals, time at RIT, and disciplines. With the introduction the Center for Student Innovation, and the new ImagineRIT festival, there is a need for a place where students and faculty can display their projects to the community, and find people to help them with their projects.

The RIT Project Hub acts as a central location for members of the RIT community to upload their project information and display them to the world. People can access the site and browse the projects currently being worked on at RIT and hopefully find a project they want to contribute to. This system, because of its accessible and neutral nature is a place where students from multiple disciplines are able to collaborate and complete projects together. The site also, creates a point where the ImagineRIT and other project outreach groups can communicate with and encourage project teams to present their projects at events or other things of that nature.

2. Basic Requirements

On a high level, the system is responsible for allowing students and faculty to express their projects to the community and to also be able to browse and contribute to other projects. This high level goal is accomplished through the implementation of various more granular requirements.

Firstly, the system allows users to authenticate to the system in order to manage their projects and request to be added to other projects. This requirement is essential in order to combat vandalism and unwanted access to projects that you might not own.

Also, the system allows users to register their projects in the system with information about it, an image, and links to any resources that the team uses. This requirement is essential because without projects, the site becomes useless. By allowing users to create more than just a listing of their project, we increase the value of the site and increase the probability that a team will see value in the Project Hub and choose to use it.
Thirdly, the system allows users to create personal profiles. This requirement is important because it allows for the system to have knowledge of the skills and information about people in the project community. When you are looking for someone with a certain skill set, we are then able to provide a list of people with those skills. This requirement allows the system to begin to become more than just a project-listing tool, and turn into a tool where information flows both ways.

The Project Hub also deploys dynamic feeds to each project page, and a global feed on the main page to highlight updates on projects.

Also, the Project Hub classifies projects as either “New,” “Active,” “Inactive,” or “Orphaned.” These classifications allow us to meet our goals by encouraging users to be active in their projects (to avoid an inactive classification) and to allow projects to live on even after their creators have graduated. The orphaned projects can then be picked up by other students and carried on from where the other team left off. Also, since all of the resources for a project are stored on the project page, no previous information is lost.

These requirements constitute our high level functionality that allows the Project Hub to meet the needs of its customers. For a more granular and complete list of requirements, see Reference 1.

3. Constraints

Team Bluebrick had a couple of constraints that presented themselves in both personal and technical arenas. Initially, our customer delivered scope was very large and ended up flushing out constraints that we had in time and personnel in order to deliver a large system like they proposed. We ended up alleviating this constraint by determining what was actually needed by the customer and adjusting our scope down to a smaller set.

We also had a technology knowledge constraint. When we came into this project, we decided to use the Django Python Framework in order to build the application. With this selection came with the fact that only one team member had knowledge of Django. This constraint only had a profound impact at the beginning of the project, and was mitigated by a fast ramp-up time and keeping team members available to help each other technically.

Other than those two constraints, our project was fairly modular and allowed us to mitigate any other constraints, or have available to us many other ways to get things done, which mitigated any constraints we would have had.

4. Development Process

In reaction to our customers need for a rapid iterative process, and our need for a requirements focused release cycle, we decided to adopt SCRUM as our development process. During our time on this project, the SCRUM methodology was modified to meet our needs. Typically, in a SCRUM setting, certain members of the team have different SCRUM roles. We ended up not using any of the typical SCRUM roles and having all members of the team share the SCRUM-master role. We also had many changes to discuss each day so we ended up ditching the 5 minute stand-up meeting in favor of a longer meeting every three days through either class time or via Skype. Our process and all the changes that we made to it were put past our mentor and received the green light.
Since this was our first major project outside of co-op, we wanted to be sure that we were setting ourselves up for success, so plenty of sponsor process-related feedback was requested. On the other hand, the process was not discussed in depth with our customer. At the most, we described that we were going to be developing in an iterative process, and the customer was pleased with that decision.

5. Project Schedule: Planned and Actual
How did your team develop the project schedule? What were the key activities and milestones you identified? How did the actual schedule compare to the plan? What explains any discrepancies? How did your team adapt to these changes in the schedule?

In the first two weeks, when we were determining our schedule, we planned around our customer timeline and our Software Engineering department timeline. We took a calendar and mapped out the deliverables that we were responsible to the software engineering department for. We then sat down with our customers and determined what they would like to see by the end of the first quarter and so on. We then took those deliverables and began to fill in the blanks. We determined what was left and needed to be done in order to reach those deliverables and placed them into the schedule. This filled out our schedule and became our initial schedule.

During the two quarters, our schedule never really changed. The deliverables were small and very doable. The only real change that occurred was that we wanted to begin to deploy the system at the end of the quarter and we were not comfortable deploying it that early.

For a complete schedule, please see Reference 2.

6. System Design
When we began this project, we ended up using Django to provide our framework on this application. With this technology decision came many other design related decisions that were made for us. Django, as a Python framework, is conformed to a very strict Model-View-Framework architecture that extends itself down into the domain logic area for our system. This specific architecture is modeled in the following way.
The Django framework works in three distinct areas, which will be covered in individual sections below. The files and classes are separated into their model, view, and controller components. This architecture is structured in a way where it is very difficult to deviate from this pattern and put logic anywhere we would like. This presented us many advantages and some challenges in terms of design. We were restricted in the way that we architected the system, but on the other hand, it forced us to practice good programming practices that in the end made our program more modular, stable, and easy to interpret.

6.1 View
The View for our project lives within the view directory for our Django code path. The view in a Django application consists of HTML pages that deal with how the data from the underlying controllers is presented to the user through their web browser. Inside the view hierarchy, there is a folder for every group of view pages that are controlled by a certain controller. Inside those folders live each of the HTML pages for different functions performed on the controller. For instance, in the Person folder there are files for adding people, displaying people, etc. One file for each page is available. The HTML pages are then tied to their controllers, which live in the domain layer.

6.2 Controller

The controller for the Django application acts as the intermediary between the view and the data. This file is where the business logic exists and through this logic it is able to manipulate the object data that is made available to it through the model. From the controller’s perspective, when a user is navigating the system, they toggle something on the view. This change is then sent down to the controller. The controller then takes this input and decides what needs to be done. The controller calls whatever functions are necessary. These functions involve manipulating data objects that are persisted through the Model for that controller. When the functions are done running, the controller updates the view and displays to the user. The controller interacts heavily with the Model, which operates on the edge between the domain layer and the data layer.

6.3 Model

The model is responsible for taking the data from our MySQL database and persisting the data through object generation. There exists a model for each controller. That model corresponds to a certain table in our database. The model then, when invoked, creates a set of objects based on the table that it is bound to. The controller that imports the model that spawned them can then manipulate those objects. The model then takes the information stored in the objects it created and updates that information in the database. By accessing the information in this manner, the Model becomes the point of contact between the application and the data. This design choice allows us to perform data sanitization on the data that is input and check for other malicious markers.

6.4 Architectural Changes

For the vast majority of implementation decisions we used the Django included functions and worked within the architecture of the Django framework. There was one example where we wanted to do something outside of the scope of the Django framework, and it required a very different approach. Early in development we wanted to include a live feed that updates people on the status of projects, while allowing people to update their feed with information. We initially researched the problem and determined that Django supported feed objects that should do what we wanted to. Unfortunately, when we began implementation, the Django feed part was not working properly. We had to then begin to brainstorm how to architect this piece of functionality so that it did its job, but also worked well with Django. We ended up creating a new table of events that was populated when any functions occurred in a project. People were also able to just write their own events to the database and have them displayed on the site. This allowed us to access the data from anywhere and work within the MVC of the Django framework.
Process and Product Metrics

During the course of this project we employed two major process metrics. The first was requirements churn and the second was hours worked. Our requirements churn metric was the most interesting to a number of us because of the fact that requirements changed so frequently during the course of the project that we really weren’t able to grasp the scope of how much change we had to deal with until we began tracking it.
The second metric that we used to track progress and make sure that we were contributing to the project was hours worked. This time was tracked at the end of the quarter and allowed us to see where we were spending the most time, so that in the second quarter we would be able to determine what events we may need to spread out as to make everything easier on the team. This metric was essential to our team because we had tons of time to dedicate to the project in the Summer, but we were going to be under a full course-load during the Fall and needed to be sure that we could manage our time effectively.

**Product State at Time of Delivery**

The product is currently in a state that meets core functionality. The product is able to be deployed and used as was originally intended. With that said, there are still a couple things that we did not get around to implementing. Currently, there is no DCE authentication, even though we have figured out how to do it, and how to properly query
LDAP. All of that information has been composed and will be deployed to GitHub with the rest of the product artifacts. So, in the future, a co-op or other student could add on and get authentication working properly. This function was not completed because toward the end of this quarter we ran into deployment issues and those took precedent over secondary features that we had planned.

**Project Reflection**

A lot of things went right, and a few large things went wrong over the development cycle of this project. In the first quarter we ran into a scope issue. We quickly discovered that the originally proposed project was not what the customer necessarily wanted, and that the customer did not really know what they wanted. We attempted to mitigate this issue by releasing a number of prototypes and mock-ups of the application to determine what they wanted and what they didn’t. A lack of decision-making skills eventually lead to a radical requirements document. This document spanned ideas from a basic forum to discuss projects, to an all-encompassing project management suite / Facebook clone. This requirement churn then lead to our iterations not being focused and us not delivering what we wanted to at the end of the first quarter.

We ended up correcting the customer issue and nailing down our final requirements for the system. At this point, our project group really began to shine. We decided to deliver updates to the system every other week to our customers and they were satisfied with the direction the project was moving in. In those two-week increments, our product, we believe, evolved more rapidly and with a higher level of quality because of our newfound ability to meet concrete requirements.

At the beginning of the quarter we decided to go with Django to implement our system. This was most likely the largest gamble that we performed because only one person on our team had any previous knowledge about Django. We began to ramp-up on Django before coding had even begun and our one team member with the previous knowledge was available to us for assistance. This combination allowed us to overcome this technology gap and finish strong.

Overall, we learned that we need to force our hand in some customer-based situations and nail down requirements, and that risk mitigation plays a major part in long projects. Even the most expensive mistakes or gambles can be accounted for and mitigated if you plan further down the road. We were lucky that we saw some of the road bumps coming and were able to course correct or plan for them earlier. These skills allowed us to finish the project and deliver something we can be proud of.

**References**

1. Reference #1

<table>
<thead>
<tr>
<th>Feature ID</th>
<th>Summary</th>
<th>Status</th>
<th>Priority</th>
</tr>
</thead>
<tbody>
<tr>
<td>36</td>
<td>The system shall allow users to see a list of contacts for the website</td>
<td>Code Complete</td>
<td>5</td>
</tr>
<tr>
<td>35</td>
<td>The system shall allow users to search for projects.</td>
<td>Code Complete</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>Deleted</td>
<td>Notes</td>
</tr>
<tr>
<td>---</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>---------</td>
<td>-------</td>
</tr>
<tr>
<td>51</td>
<td>The system shall support a dynamic tool library, which supports check-in and check-out of items available in the Center for Student Innovation.</td>
<td>Deleted</td>
<td>1</td>
</tr>
<tr>
<td>29</td>
<td>The system shall allow users to communicate with video in personal messages.</td>
<td>Deleted</td>
<td>1</td>
</tr>
<tr>
<td>48</td>
<td>The system shall allow for projects to be linked to an external source for receiving donations and funding.</td>
<td>Deleted</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>The system shall allow people to import their information through SIS and Facebook.</td>
<td>Deleted</td>
<td>4</td>
</tr>
<tr>
<td>10</td>
<td>The system shall allow users to search for other users.</td>
<td>Deleted</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>The system shall allow users to add other users to their network (may mean in new scope that the system generates the network rather than allowing additions).</td>
<td>Deleted</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>The system shall perform recommendations on people.</td>
<td>Deleted</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>The system shall allow users to mark themselves as as looking for a project or not</td>
<td>Deleted</td>
<td>3</td>
</tr>
<tr>
<td>19</td>
<td>The system shall allow users to submit tips for their project to blogging staff.</td>
<td>Deleted</td>
<td>2</td>
</tr>
<tr>
<td>22</td>
<td>The system shall allow users to denote their interests.</td>
<td>Deleted</td>
<td>5</td>
</tr>
<tr>
<td>23</td>
<td>The system shall aggregate a history of recent events and posts for a project to display.</td>
<td>Deleted</td>
<td>3</td>
</tr>
<tr>
<td>24</td>
<td>The system shall have a system administration mode which allows for system administrative tasks.</td>
<td>Deleted</td>
<td>5</td>
</tr>
<tr>
<td>27</td>
<td>The system shall allow users to communicate through internal personal messaging.</td>
<td>Deleted</td>
<td>4</td>
</tr>
<tr>
<td>28</td>
<td>The system shall allow users to use external communication tools to communicate with other users.</td>
<td>Deleted</td>
<td>4</td>
</tr>
<tr>
<td>#</td>
<td>Requirement</td>
<td>Status</td>
<td>Percentage</td>
</tr>
<tr>
<td>----</td>
<td>-----------------------------------------------------------------------------</td>
<td>--------------</td>
<td>------------</td>
</tr>
<tr>
<td>30</td>
<td>The system shall allow users on a project to invite other users through personal messages.</td>
<td>Deleted</td>
<td>5</td>
</tr>
<tr>
<td>31</td>
<td>The system shall allow authenticated users to browse and create contests.</td>
<td>Deleted</td>
<td>5</td>
</tr>
<tr>
<td>32</td>
<td>The system shall allow for specially authenticated users to post to a Wordpress blog on the site.</td>
<td>Deleted</td>
<td>5</td>
</tr>
<tr>
<td>34</td>
<td>The system shall allow users to search for contests.</td>
<td>Deleted</td>
<td>5</td>
</tr>
<tr>
<td>37</td>
<td>The system shall allow users to see a calendar of events for RIT and the Innovation Center.</td>
<td>Deleted</td>
<td>5</td>
</tr>
<tr>
<td>38</td>
<td>The system shall allow users to see a list of courses being taught in the Innovation Center.</td>
<td>Deleted</td>
<td>5</td>
</tr>
<tr>
<td>39</td>
<td>The system shall allow users to access the Wordpress blog posts for the Innovation Center.</td>
<td>Deleted</td>
<td>5</td>
</tr>
<tr>
<td>45</td>
<td>The system shall provide RSS feeds for the main page updates, people updates, and project updates.</td>
<td>Deleted</td>
<td>1</td>
</tr>
<tr>
<td>49</td>
<td>The system shall allow for users to delete themselves from the system, removing their profile information.</td>
<td>Deleted</td>
<td>3</td>
</tr>
<tr>
<td>50</td>
<td>The system shall allow for users to disable their accounts on the system without removing their profiles.</td>
<td>Deleted</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>The system shall allow users to register on the site</td>
<td>In Progress</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>The system shall allow users to upload a description and picture for the project</td>
<td>In Progress</td>
<td>5</td>
</tr>
<tr>
<td>11</td>
<td>The system shall show a list of active and past users for a project.</td>
<td>In Progress</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>The system shall allow people to create new projects</td>
<td>In Progress</td>
<td>5</td>
</tr>
<tr>
<td>15</td>
<td>The system shall allow users to link their resources to the project site</td>
<td>Code Complete</td>
<td>5</td>
</tr>
<tr>
<td>40</td>
<td>The system shall require authentication for all creation and edits of data on the site.</td>
<td>Code Complete</td>
<td>5</td>
</tr>
<tr>
<td>41</td>
<td>The system shall allow users to submit feedback on the site via an on site form, which will forward the feedback to a development address.</td>
<td>In Progress</td>
<td>3</td>
</tr>
<tr>
<td>7</td>
<td>The system shall allow users to post skills they are looking for on the project page</td>
<td>Code Complete</td>
<td>5</td>
</tr>
</tbody>
</table>
The system shall allow users to tag projects with specific information like domain.  

18

2. Reference #2

Quarter 1

Week 1 (May 31 - June 6)

- Submit request to SE System Administrator to create team account providing: team name, login-ids of each team member and faculty coach. (Planned Week 2)
- Inform faculty coach and project sponsor of the URL for the project website. (Planned week 4)
- Submit project synopsis of approximately 200 words to sponsor and faculty coach for approval. (Planned week 4)

**Week 2 (June 7 - June 13)**
- Submit request to SE System Administrator to create team account providing: team name, login-ids of each team member and faculty coach. (Done)
- Start project website on an se.rit.edu machine. (Undetermined)
- Publish, on project website, team information including approved synopsis. (Undetermined)
- Publish, on project website, first tracking report of time/effort worked. (Undetermined)
- Complete project information survey form. (Undetermined)

**Week 3 (June 14 - June 20)**
- Submit first draft of project plan for review by faculty coach and sponsor. (Planned Week 4)
- Document development methodology on project website. (Planned Week 4)
- Document product/process metrics on project website. (Planned Week 4 or 5)
- Deliver high level prototype of feature requests and client needs. (Done)

**Week 4 (June 21 - June 27)**
- Inform faculty coach and project sponsor of the URL for the project website. (Tentative)
- Submit project synopsis of approximately 200 words to sponsor and faculty coach for approval. (Tentative)
- Start project website on an se.rit.edu machine. (Tentative)
- Publish, on project website, team information including approved synopsis. (Tentative)
- Publish, on project website, first tracking report of time/effort worked. (Tentative)
- Complete project information survey form. (Tentative)
- Submit first draft of project plan for review by faculty coach and sponsor. (Done)
- Document development methodology on project website. (Tentative)
- Document product/process metrics on project website. (Tentative)

**Week 5 (June 28 - July 4)**
- Mid-Term Peer Evaluations

**Week 6 (July 5 - July 11)**
- Coordinate Presentation Meeting for Week 9/10

**Week 7 (July 12 - July 18)**

**Week 8 (July 19 - July 25)**
- Draft interim presentation for week 9 or 10.
- End Sprint - June 21 to July 19 - Week 8: Deliver Functional Prototype

**Week 9 (July 26 - August 1)**
- Give interim presentation in week 9 or 10.

**Week 10 (August 2 - August 8)**
- Individual end-of-term peer evaluations.
- Interim team self-assessment.
- Project website and repository on an se.rit.edu machine up-to-date with all project artifacts, tracking reports, etc.
- Course evaluations.
- End Sprint - July 20 to August 2 - Week 10: Deliver Milestone "End of Quarter"

**Week 11 (August 9 - August 15)**
- Summary of reflection meeting.

---

**Quarter 2**

**Week 1**
- Project plan updated for second term.

**Week 2**
- 

**Week 3**
- 

**Week 4**
- Attend session on making a poster and writing the technical report.
- End Sprint - Week 4: Beta Release Milestone "Functionally Complete"

**Week 5**
- Mid-term peer evaluation
- Project poster concept

**Week 6**
- Preliminary project poster

**Week 7**
- Project poster

**Week 8**
- Draft final presentation for week 9 or 10.
- Technical report outline

**Week 9**
- Give final presentation in week 9 or 10

**Week 10**
- Draft technical report
- Individual peer evaluations
- Team final self-assessment
- Post-Mortem Curriculum Reflection
- Project website and repository on a se.rit.edu machine up-to-date with all project artifacts, tracking reports, etc.
- Course evaluations

**Week 11**
- Final project artifacts
- Final technical report
- Summary of Curriculum Reflection
- Senior survey