Storefront Testing App

Planning and Estimation

Team 4 - Bugs are Features Too

University of Kentucky - CS 499 - Fall 2019

Customer: Ben Fox, Trissential

September 27, 2019

Authors:

Seanna Lea LoBue - Editor, Size Estimate, Project Schedule, Resource Allocation

Michael Murray - Editor, Size Estimate, Risks and Mitigation, Resource Allocation

1. Size Estimate

We have chosen to evaluate the size of our project in terms of the sum of the story points attributed to our user stories. By having each team member assign story points relative to an agreed upon estimate of 8 story points for the product list page we were able to produce an estimate of the overall size of our project. As such, we estimate the size of our project to be 90 story points. Our project schedule is divided into 5 sprints, each with a duration of 2 weeks, therefore we have a target velocity of 18 points per sprint. The larger size of these sprints is helpful to accommodate some of our larger stories that we would prefer to not subdivide, as well as, providing additional flexibility for the real-life constraints of our team. We will develop the product along functionality tracks such as login and account creation, product search and display, and product selection and order creation. This will allow us to create functional elements regardless of the scope of the ecommerce platform itself.

2. Risks and Mitigation

Risk: Unintended bugs

Comment: As the purpose of this project is to have tightly controlled intentional bugs for the purpose of training and client demonstrations, unintended bugs, even if minor, can be particularly detrimental to the perceived value of this product.

Mitigation: We will conform with good coding practices, take part in regular peer review of each other's code and we will perform both human and automated testing during our iterations.

Risk: Core Storefront design takes too much of the development time

Comment: The main purpose of this project is to have a platform for dynamic bug assignment on a per-user basis. However, as the Storefront does not already exist, designing one is a foundational element to this project, but alone is not satisfactory. Thus, it is important to maintain pace so that the main goal can be completed.

Mitigation: A well established hierarchy of element priority will mean that after a bare minimum of the Storefront has been created, parallel work can proceed on the bug framework. Also, by observing the schedule we have designed, we can continuously evaluate our production timeline and make further changes if needed.

Risk: Redoing Storefront design in order to accommodate bug framework

Comment: As mentioned above, the Storefront, in some form, must exist before continuing. As such, the risk exists that the first iteration of the Storefront will not be well designed to accommodate our needs for providing dynamic bug assignments.

Mitigation: After completion of the first components of the Storefront, work can begin on efficient methods introducing dynamic bugs. As the project progresses, these techniques can be evaluated in terms of applicability to the entire project scope. Testing early and monitoring progress will mean that if a problem does arise, we can minimize the impact.

Risk: Addition of new bugs is too hard

Comment: We will attempt to incorporate as many bugs as possible; however, the goal is to design the system in such a manner that future CS499 teams or Trissential could continue development on this project. As such, it's important to emphasize ease of making changes and extensions.

Mitigation: All bug tracking and indexing will be done with individual database entries; therefore, adding new bug entries and parsing them will be simple. More complex however, is the code implementation of new bugs. While some bugs will likely rely on unique code, our plan is to maximize the number of bugs that can be introduced by a common intermediate function. Ideally such that the introduction of some bugs could be as simple as a reference to a function id and the intended invalid return value.

Risk: Web server compatibility/issues

Comment: After speaking to our client, we learned that in the past, things worked in the student's environment but frequently encountered issues when attempting to deploy at the end of a semester.

Mitigation: While it would prove too cumbersome to attempt to develop immediately from the client's environment, we will test each of our major milestones in the client's environment.

Risk: Feature creep

Comment: Some client requests were open ended on how much variety was needed within a singular task. As such, some of our ideas we would like to implement, but were not explicitly required by the client, may not be feasible within the allotted time.

Mitigation: As acknowledged above, by having a hierarchy of element priority, we can focus on the requested deliverables, expanding the scope when time permits.

Risk: Bugs are too simple

Comment: One of the core goals of the dynamic bug framework is to test and train employees of the client's company. If the bugs we are able to produce are too simplistic, then it is possible that the product will not be of the quality that is expected by the client.

Mitigation: Our conversations with the client on what they want, including an emphasis on what kind of bugs they desire, has given us a good understanding of the variety and depth of the bugs expected. Nonetheless, we will regularly meet with our client, discuss progress and when applicable, demonstrate our current iteration. By doing so, we can receive feedback, allowing us to see if we have deviated from expectations and make corrections.

Risk: Multi-bug incompatibility

Comment: Our bugs are intended to cause particular elements of the Storefront to not function as expected, but not crash the website. It is possible that combinations of certain bugs may produce errors that if not handled or prevented will cause the application to terminate unexpectedly.

Mitigation: When possible, bugs will be designed to give a valid response to user input, just not the one they had intended. In this case, we can be confident, assuming our Storefront is functional under a no-bug configuration, that no error will occur, because the response will match an input the user could have made. When introducing bugs that we believe could result in compatibility issues we will resort to testing to ensure that either no error occurs or that we prevent the assignment of conflicting bugs at one time.

Risk: Not completing the project in time

Comment: The definition of completion for our project is fairly broad, but it is important for us to end the semester with a set of deliverables that have satisfied all of the core needs of the client.

Mitigation: Our project plan is designed in such a way that we will satisfy the pillar customer needs early in the production schedule. This means that the only elements at risk for not being completed by the end of the semester will primarily be quality of life features and the full intended breadth of pre-programmed bugs. That being said, we will adhere to our schedule, distribute work in a manner to take advantage of individual skills and communicate when issues arise in order to minimize the aforementioned risk.

3. Resource Allocation and Project Schedule

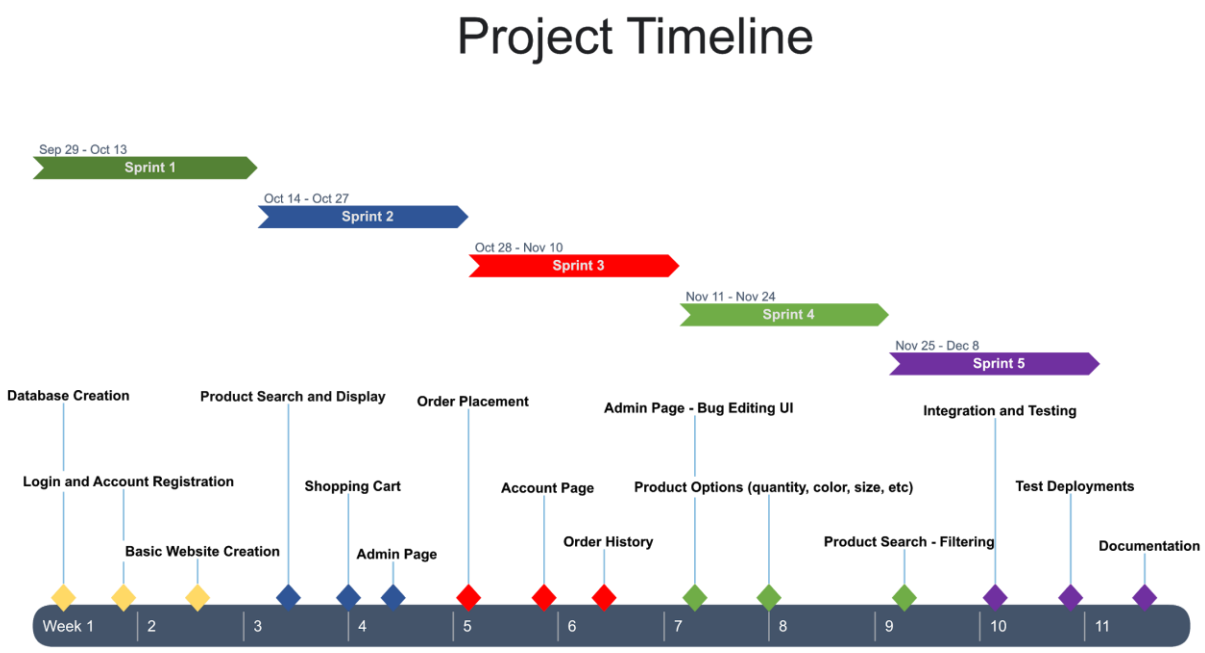
This project will rely on the existence of a web server. For developmental purposes, we will be able to host our project on the Multi-lab virtual machines belonging to the University of Kentucky, which have a pre-configured apache web server. For deployment, our client has an available apache web server we can use. However, when migrating to the client's infrastructure we will need to ensure compatibility between versions and libraries installed as compared to the development environment. No other resources are known to be required for our forecast of our project development.

For ease of implementation, we will be constructing a MySQL database, running a minimum version 5.7. Database versioning will be handled within the program in order to minimize potential risks associated with database incompatibilities introduced during the project. Installation of MySQL will be handled either within documentation, allowing the Trissential team to install the database server as best fits their needs.

3.1 Project Schedule

The project schedule provides a detailed outline of what functional areas will be concentrated on during each sprint. While there is an expectation that some elements may extend to another sprint or be worked concurrently, these items are logically arranged so that items that are prerequisites for further development work are done earlier within the project timeline.

Figure 3.1 Project Sprints and Milestones



3.2 Team Resource Allocation

Our tasks will be roughly allocated into group tasks, where work is divided equally and then project branches where one or two people take ownership of a single segment of the project.

All Members:
Sprint 1 - Basic Website Creation
Sprint 5 - Integration and Testing
Sprint 5 - Test Deployments
Sprint 5 – Documentation

Seanna Lea LoBue:
Sprint 1 - Database Creation
Sprint 2 - Admin Page
Sprint 4 - Admin Page - Bug Editing UI

Michael Murray:
Sprint 2 - Product Search and Display
Sprint 4 - Product Search - Filtering

Eric Prewitt:
Sprint 1 - Login and Registration
Sprint 3 - Order History

Nicholas Wade:
Sprint 2 - Shopping Cart
Sprint 3 - Order Placement

Kee West:
Sprint 3 - Account Page
Sprint 4 - Product Options

In addition to these timeline tasks, we will do code reviews and other planning tasks no less than biweekly during the course of the project. Customer meetings will be planned

around major milestones, but are expected at the end of Sprint 1, 2 and 4 in order to showcase major elements around the bug framework deployment.

4. Word Count

Seanna Lea LoBue - 346

Michael Murray - 1,197

Eric Prewitt - 0

Nicholas Wade - 0

Kee West - 0