CSC 440 Stage C Deliverable

- Timespan: 11/02/2017 - 11/22/2017
- Submitted: 11/20/2017
- GitHub Repo

Team Name: Redbeard

Team Members: Nick Tope, Eric Spahr, Katie Bolen

Application of Theories

(Note: While only a few of the principles below applied to the current stage, we thought it beneficial to write more principles in advance that we believe will apply to later stages)

1. KISS - Keep it simple stupid.
    - I used this rule when trying to explain Flask and Python code to my team members to show them what I learned from the class that was given on Friday night a couple of weeks ago. This helped make things easier to understand for my team members since they didn't attend. [Eric 11/1/17]
2. DRY - Don't repeat yourself. [Eric]
3. Always design for your audience. [Eric]
4. Stockholm Syndrome - Confront the brutal truth of the situation, yet at the same time, never give up hope. [Eric]
5. Adding human resources to a late software project will make it later. Create a procedure for delays and make all team members aware of this procedure; what is expected of them, how they should respond, and any potential consequences that may result from failure to follow. - Fred Brooks, Brooks' Law [Katie]
6. Focus project on a concept at which you can be the best. Formulate a list of potential concepts and attempt to explain why the team believes they can be the best. - Good to Great, Hedgehog Concept [Katie]
7. Team members should be in constant communication. In the earliest stages of the project, everyone should exchange contact information, and agree upon a regular meeting schedule. Some means of emergency contact should also be agreed upon and checked regularly. - Fred Brooks, manage complexity by simplifying communication [Katie]
    - Communication allowed us to determine that we did not have enough time to complete all tasks on time. As a result, we were able to adjust our deadline, figure out what needed to be done, and get it accomplished ahead of schedule.
8. Establish a design concept before beginning coding. Begin with a list of product requirements, and meet as a group to discuss any additional ideas. - David Parnas, Rational Design Process [Katie]
9. Create documentation for all stages and update frequently. A standardized form will ensure that all necessary information is provided. - David Parnas, Rational Design Process [Katie]

- The Wiki diagram and analysis document helped us to get a systematic look at structure, and to keep track of the code.

10. Design tests early in the planning process to help focus the architectural structure of software. - Advice from professor [Katie]
11. Utilize the principle of encapsulation to streamline debugging/testing, and to maintain a high level of adaptability of product. - David Parnas, Rational Design Process [Katie]
12. Assess team members' individual strengths to ensure that they are in assigned the most appropriate role within the team. - Good to Great, First Who... Then What [Katie]
    - We attempted to divide work up amongst members of the team based on their individual strengths, but also worked together to address issues or figure out problems.
13. Require the use of comments in coding to clarify program logic, thus simplifying communication between team members. Have team members meet up in pairs/small groups and explain their newly added code using only comments to verify its effectiveness. - Advice from professor [Katie]
14. UIP Principle - Understand, Iterate, Practice. [Nick]
    - Reviewing existing code helped us to better understand Python, and provided guidance as to how our code could/should be structured.
15. Peer reviews catch 60 percent of the defects. - Barry Boehm and Victor R
    - This principle will apply to every stage. Much of the mistake that are made can be caught through peer review. While we haven't written code for this stage, we did have the team peer review this report to look for mistakes. [Nick 11/1/17]
16. Disciplined personal practices can reduce defect introduction rates by up to 75 percent. - Barry Boehm and Victor R [Nick]
17. If you don't know history, you're doomed to repeat it. [Nick]

Schedule & Checklist

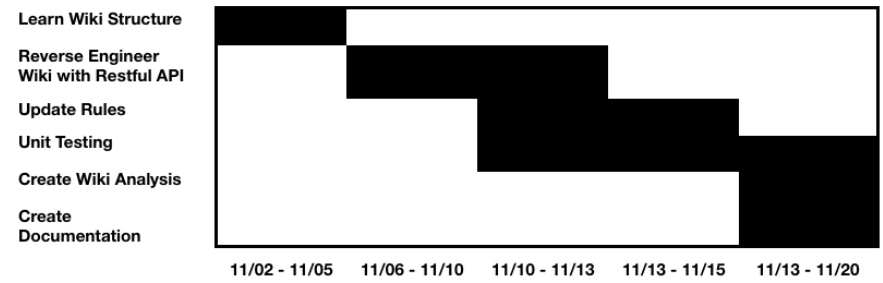

Figure 1: Initial Gantt Chart

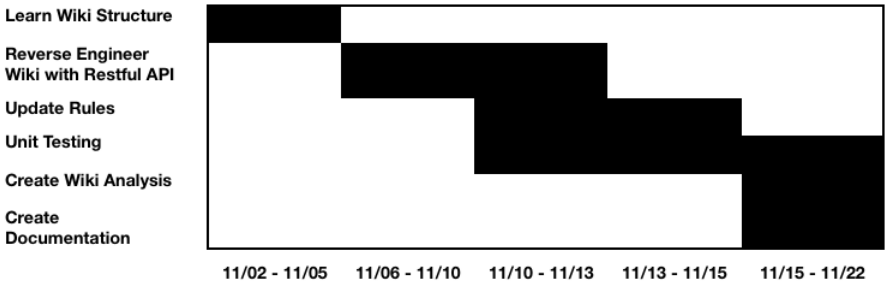

Figure 2: Updated Gantt Chart

	Delivered?
Stage D (10/26 - 11/01)	
Documentation	✔
Stage C (11/02 - 11/22)	
Schedule Reverse engineer wiki (GitHub) Wiki unit tests (GitHub) Wiki analysis Agile Retrospective	✔ ✔ ✔ ✔ ✔
Stage B	
TBA	
Stage A	
TBA	

Figure 3: Check List

Team Meetings

- [11/08/2017, 68 minutes] Eric and Nick met up after class to begin reverse engineering of Wiki; Eric updated Katie on changes to code.
- [11/13/2017, 74 minutes] All team members met up after class to update unit tests.
- [11/15/2017, 62 minutes] All team members met up after class to work on unit tests and update software rules.
- [11/20/2017, 68 minutes] All team members met up afterclass to review updates to code, finalize Stage C documentation, and complete Stage C retrospective. Stage B planning began, including schedule and checklist.

Wiki Analysis

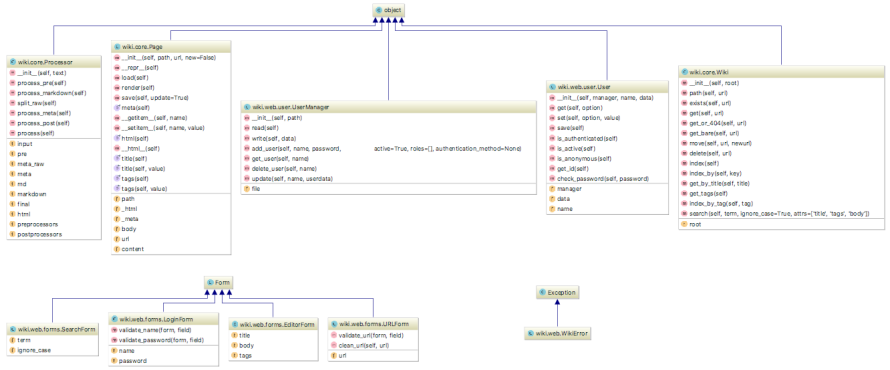

Figure 4: Wiki Diagram

Class	Objective	Input to Action
'wiki.core.Processor'	Processes content (markdown /metadata) for rendering	Passes content information from core classes to render content
'wiki.core.Page'	Creates page in Wiki (including setting its URL and content) and saves/loads content	Updates saves/modifies page content and passes to Processor for rendering
'wiki.web.user.UserManager'	Creates new users, updates/retrieves user information, and deletes current users	Input on user data is stored/updated in JSON for system use
'wiki.web.user.User'	Retrieves user information to assist in the creation /deletion/updating of user information	Passed information to verify contents in assistance of UserManager methods
'wiki.core.Wiki'	Allows the retrieval, organization, searching, and verification of content within the Wiki	Uses input from other core classes to retrieve information, and to update Wiki content and structure (indexing, URL relocation, etc.)

Class	Objective	Input to Action
'wiki.web.forms.SearchForm'	Searches specified form for input provided by user	Passes user's search input to find occurence or non-occurence of that search criteria; returns its occurence or notifies that it cannot be found
'wiki.web.forms.LoginForm'	Allows users to login to Wiki	Passes user's login input to verify their user information; login is allowed or notifies that it is not recognized
'wiki.web.forms.EditorForm'	Allows user to edit content title, body, and/or tags of specified form in Wiki	Passes user's input to modify form that is passed in; modifications to form are made
'wiki.web.forms.URLForm'	Validates existing URL and return clean URLs for use	Passes URL in to verify that it exists/does not exist; clean URLs

How Wiki Works: Wiki works by first allowing Users to sign-in. Authorization of user credentials is verified, allowing user to view existing content by navigating the Wiki's pages. User credentials can be updated. Page content on the Wiki can be retrieved through site navigation or located using search functions. Content can be edited; edited content is updated in the Wiki system for use.

How Restful API Used: Restful API appears in this code through the modularization of its code. User input and actions are broken down into their simpler parts, and implemented "behind the scenes," making use much cleaner and simpler for users.

Markdown to HTML Processing: Markdown files are called through the Processor class ["process", "process markdown"] and the "convert" method runs pandoc on the contents of the Markdown content to convert markdown symbols to their HTML equivalents. The final HTML content is produced.

Deadline Extension Report

Original Deadline: 11/15/2017 **Updated Deadline:** 11/22/2017

Issues: Multiple exams and assignments due the week of November 15 prevented the team from scheduling team meetings to complete the final components of Stage C. Unit tests were not complete as the deadline approached.

How Issues Were Addressed: The team collectively agreed to push back the deadline of Stage C and continued to work on the unfinished aspects of the current stage. We agreed to meet the following Monday to assess our progress, at which point all aspects were agreed to be satisfactorily completed.

Changes to Group Policy: Previously, the team agreed to meet on Wednesdays to work on project. It has been decided that one meeting per week was not sufficient; shorter meetings will now be held more frequently, including before and after class sessions on Mondays and Wednesdays. Facetime, email, and/or text messages will be used to keep all team members up-to-date if all members cannot attend.

Agile Retrospective

Team Leader: Eric Spahr

Team Members: Katelin Bolen, Nick Tope

Facilitator: Dr. Samuel Cho

Meeting: * Date: November 20, 2017 * In attendance: Nick Tope, Katelin Bolen, Eric Spahr * Goals: Finalize all documents, complete retrospective * Total meeting length: 68 minutes

Documents: * Stage C Application of Theories: Software Engineering Rules [ver2] * Stage C Schedule & Checklist * Stage C Team Meetings * Wiki Analysis * Stage C Deadline Extension Report * Stage C Agile Retrospective

Things That Went Well

- Reverse engineering of wiki
- Creation of wiki analysis

Things That Could Be Improved Upon

- Class and work schedules made it difficult to plan meetings as a group
- Unit testing fell behind

Lessons Learned

- Shorter, more frequent meetings would be better than longer, infrequent group meetings

Final Thoughts

- Planning for Stage B begins immediately; team will be in contact over Fall Break [11/22 - 11/26]
- Retrospective meeting concluded with a discussion over Stage B, and the drafting of Stage B Schedule & Checklist.