Kent State Capstone Spring 2017

DRAFT PROJECT PLAN

DOCUMENT REVISION HISTORY

Major revisions of this document will be lettered (i.e. A, B, C).

Document Version	Description	Revised By	Revision Date	Reviewed By
A	Preliminary version for review	Conry	03-Jan-2017	
B				
C				
D				

Contents

1. Introduction

1.1. Purpose

The purpose of this document is to define the structure and requirements for Kent State University's Computer Science Capstone course. This document should help provide an outline to make the experience for the students educational and productive.

1.2. Overview

Students will be broken into teams to work on projects developing software solutions. Each team member will have a specific responsibility and deliverable. Since the course is a writing intensive course it will be expected that each responsibility has a considerable amount of writing throughout the semester. All documents

1.3. Intellectual Property

RoviSys is the owner for all works developed during the duration of the class. All work and Intellectual Property as a result of this course is the sole property of RoviSys.

1.4. Documentation

At minimum each group will be responsible for providing a User Requirement Specification, Functional Specification, and Functional Design Specification. Extra Documentation will be based on the roles for members on the team and scope of the project.

1.5. Schedule

- 1/17 – Classes Begin
- 1/20 – Student Apply for Roles
- 1/23 – Students are provided Roles
- 2/06 – Rev A of documentation Due
- 3/24 **– Engineering Cutoff**
- 4/03 – **Development Cutoff**
- 4/10 – Final Presentation

2. Student Teams

Students will split into project teams for the duration of the semester. Teams will consist of 4 members. It will be an expectation that every student will write code and complete some form of documentation. Roles necessary will vary based on the scope of the project.

2.1. Team Roles

Team roles should reflect current industry roles to give student experience on what they might experience on the job. Some roles will be required while others will be optional depending on the scope of the project.

2.1.1. Project Manager **(Required)**

The goal of the project manager is to translate requirements, both application and schedule, from the customer to the development team. This person will be the main point of contact from the development team to the customer (RoviSys). The Project Manager will be responsible to developing and maintaining the **User Requirement Specification (URS).** Projects Managers **must be in contact with their customer on at least a weekly basis,** providing updates and answering questions from their development team.

Deliverables: User Requirement Specification (URS) and weekly status updates to customer.

2.1.2. Technical Lead **(Required)**

Implementation details are managed by the Technical lead. Technical leads must work with the development team to create solutions to the requirements of the **User Requirement Specification**. All solutions are documented in a **Functional Requirement Specification(FS)**. Everything in the FS will need to be finished by the end of the projects duration. At the end of the project the Technical Lead will also be responsible for providing a **Functional Design Specification (FDS).**

Deliverables: Functional Requirement Specification (URS) and Functional Design Specification.

2.1.3. Quality Assurance Specialist **(Required)**

The QA specialists focus is making sure the system is working as expected throughout development. Applications are not customer ready until the QA team says so. QA specialists use automated tests to ensure consistency as well as manual tests plans that are run before a customer a customer can see the system. QA should try to implement **Unit Test**s whenever possible, otherwise all scope should be tested through a **User Acceptance Test (UAT)** plan.

Deliverables: User Acceptance Test and Unit Tests

2.1.4. User Experience Designer

User Experience has become increasingly more important for applications. User Experience Designers (UXD) are responsible for planning out user interactions with the system. This include creating **Mockups** before the creation of a page, and **sitemaps** to details user flow. More advanced techniques, like **User stories**, are not a requirement but will help the development team understand user requirements better.

Deliverables: Mockups, Sitemaps, and User Stories (Optional)

2.1.5. Database Administrator

Database Administrators (DBA) will be responsible for designing the data schema and maintaining the database. All changes made to the database must go through the DBA. The DBA is in charge of creating **schema diagrams** to document the approved and define structure of data. For deployment and release the DBA will be required to supply **scripts to create or migrate database**.

Deliverables: Schema diagrams and scripts to create or migrate database

2.1.6. Software Architect

Software Architects are responsible for the code quality aspects of a solution, making sure code adheres to the following principals:

- Readability
- Modularity
- Maintainability
- Portability

Architects will be responsible for making sure code is **properly commented** and documented. **Class Diagrams** will be necessary to show dependencies and attributes of a class.

Deliverables: Comments on code and class diagrams.

3. Communication

All students are expected to communicate issues and progress to the entire team**. Daily Standups** are not required but are strongly recommended. And uncertainties on scope should be communicated to the Project Manager ASAP.

3.1. Weekly Reports

Project Managers will be required to submit Weekly Reports to the Customer (RoviSys). These Reports must be emailed to Chris Conry by 10am every Tuesday. **Late or Missing reports will result in loss of credit.** These reports should include the following:

- Development Progress
  - Progress according to the schedule
- Designs for future development
- Issues During Development
- Questions or Clarification on scope.

3.2. Milestone Reviews

Teams will be required to schedule at least two Milestone reviews of the system with the Customer (RoviSys). More reviews are advisable because it will help to make sure the progress made is in the right direction. Milestone reviews will show the progress of the system to the customer. Everything demoed during the milestone review must be feature complete and working. Incomplete sections of the system or broken functionality during a demo will **result in loss of credit for the entire team.**

3.3. Escalation Path

Any good team can solve a problem on the **lowest level possible**. Team members are expecting to resolve issues between each other between themselves. Unresolved issues concerning interpersonal issues between members need to be escalated to the classes professor (Dr. Samba) ASAP. It is strongly recommended that members deal with the issues appropriately between themselves.

3.4. Final Presentation

During Finals Week students will give a final presentation of their application. During this presentation the team should lead the Application through the User Acceptance Test for the customer. The customer will have the option to pass or fail a feature based on its completeness.

4. Appendix

1. **Scope** – The scope of the project is everything detailed in the requirements documents (FS and URS). The project team should stay within the scope of the project to keep development efforts on schedule and in budget.
   a. **Scope Change** – During the course of a project scope can be added and removed by the customer and project manager. Scope added is generally within the existing goals of the system, but not defined as a requirement.
   b. **Scope Creep** – Any features implemented outside of the goals of the system. Scope Creep should be kept at a minimum if at all possible.
2. **Engineering Cutoff** – This Designates the last date where scope can be defined or changed.
3. **Development Cutoff** – This Designates the last date features can be developed. No code should be committed past this date, excluding bug fixes. The system should be tested and documented after this date.
4. **User Requirement Specification (URS)** - Document outlining the business and operational requirements from a customer. The requirements are not forms of implementation, but rather the expectations a customer has of the completed system. As customer requirements are defined with more detail or scope is added or removed those changes are detailed in the revisions of the document.
5. **Functional Requirement Specification (FS**) – Document containing implementation details to accomplish the user's goals and requirements laid out in the User Requirement Specification. As Implementation details change based on
6. **Functional Design Specification (FDS)** – Document detailing the architecture of the system. This should be a compilation of other user's functional documentation (Class Diagrams, Mockups, Site map, Schema Diagram)
7. **Unit Test** – Automated test that runs frequently to tests the validity of a system.
8. **User Acceptance Test (UAT)** – Document outlining the test procedure to validate integrity and completeness of a system. There should be a test for every requirement in the Functional Specification. Missing tests will result in a loss of credit.
9. **Mockups** – Rough draft of UI design.
10. **Sitemap** – Flow cart of site pages and access through navigation.
11. **User Story** – Use cases of an application based around the day to day activates of a user.
12. **Schema Diagram** – Diagram visualizing data dependencies and attributes.
13. **Daily Standup** – Daily meeting for team member to share development progress.