Team Members: Gregory Aiello, Alex Curtin, David Mayer,

Douglas Naphas, Jeff Ramspacher

Date: 05/09/2014

Version #2

REQUIREMENTS SPECIFICATION

Table of Contents

Overview

This document presents the requirements for the Random Fruit software project management system.

Abstract

Random Fruit is an open-source project management site for students developing software.

Like existing tools, it allows developers to break work down into tickets, assign the tickets to each other, view and update a page capturing all the information about a ticket, search tickets, and use the information captured in tickets to generate reports. Random Fruit is tailored to the needs of student projects using earned-value reporting, adopting reporting tools likely to be useful in an academic setting and enabling coordination among teams by an instructor.

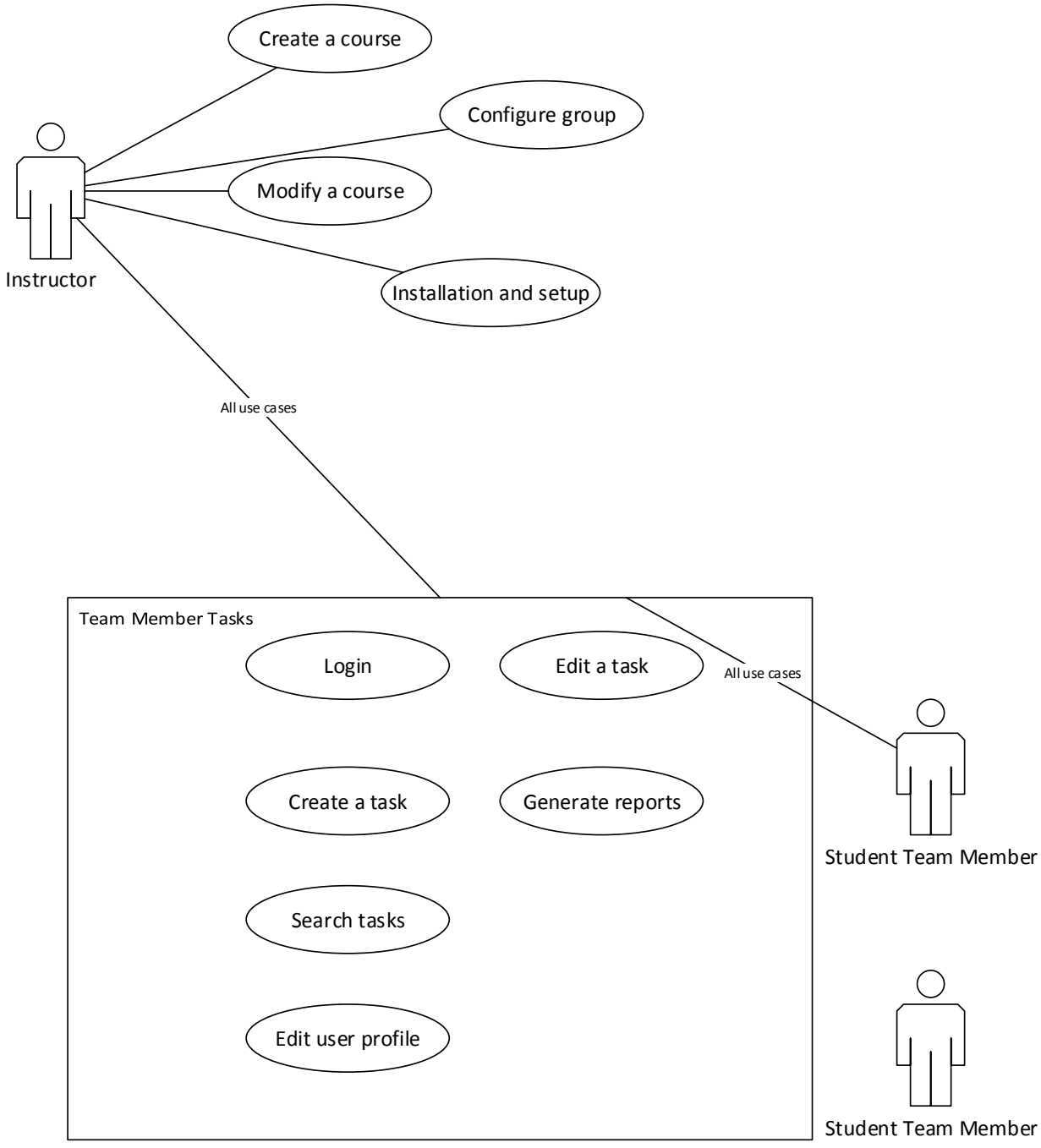
Random Fruit aims to help students manage their work while demonstrating their progress to an instructor. It tracks time budgets and time incurred for each ticket, as well as hours worked on the project overall. The reporting feature aggregates this information to produce earned-value charts showing the changes in the remaining work volume and highlighting the achievement of milestones. Students can use the system to visualize planned value, earned value, and actual value at any time.

In addition to acting as a superuser with team-member privileges over all the groups, the instructor can use the system to quickly view how the course as a whole is progressing towards its goals. The instructor can compare groups to spot potential problems before they become emergencies.

A dashboard landing page shows users a top-level view of their project's status. Markdown-enabled comments tell the story of a ticket on its view page. Users can filter and sort tickets, and save reports.

For distribution, Random Fruit's installer application places its file structure on the server of a course instructor or department at a university, where it serves a web page to student and instructor users. It is written in PHP, JavaScript, and HTML, and served by a MySQL database. It is accessed through the web.

Use Case Diagram



...

Use Cases

Instructor – Installation and setup

1. Instructor navigates to the installation directory of Random Fruit in a web-browser.
2. The configuration file containing the username, password, location, and name of the MySQL database to be used by Random Fruit is used to verify that the system can connect to the database. This step is cannot be skipped.
3. Once the database is accessible, instructor is required to provide:
    a. Username
    b. Password
    c. First name
    d. Last name

    In addition, the instructor can optionally provide office hours and office location.
4. The instructor is registered as the administrator. The database is adjusted/altered to accommodate Random Fruit (necessary tables are created and default data is set).
5. The instructor is then directed to the dashboard for the first time.

Instructor – Create a course

1. Instructor clicks “Create Course” from the dashboard.
2. System displays a prompt and the instructor enters the following information for the new course:
    a. A course code
    b. A description
    c. A start date
    d. A number of weeks for the semester
3. Instructor clicks button labeled “Create Course”
4. System stores course information in the database and the course is now visible from the “View Courses” page.

Instructor – Modify a course

1. Instructor clicks “View Courses” from the dashboard
2. System returns a page containing all courses in the database and their corresponding projects and project members.
3. Instructor can activate/deactivate a course by checking/unchecking the “Active” checkbox in the corresponding course panel.
4. Instructor can activate/deactivate planning mode for a course by checking/unchecking the “Planning” checkbox in the corresponding course panel.
5. Instructor can create projects in specific courses by clicking “Add Project” on the “View Courses” page and entering:
    a. A project name
    b. A project description

   c. An associated course
6. System stores project information in the database and the project is now associated with a specific course.
7. Instructor can add existing users to a project by clicking “Add User” on the “View Courses” page and selecting:
   a. A user
   b. A project to add the user to
8. System stores user-project relationship in the database and the user is now a member of both the course and corresponding course.
9. Instructor can remove users from a project by clicking the red “X” icon directly to the right of the corresponding users.
10. Instructor can delete projects from system by clicking the red “X” icon directly to the right of the corresponding project names.

Instructor – Create User

1. Instructor clicks “Create User” from the dashboard.
2. System displays a prompt and the Instructor enters the following information for the new user:
   a. Username
   b. Password
   c. Email address
3. Instructor clicks button labeled “Create User”
4. System stores user in the database.

Create ticket

1. User clicks the icon:  located in the upper-right corner of the page.
2. System displays a prompt and the user enters the following information for the new ticket:
   a. Project
   b. Assignee (default is the current user)
   c. Week due
   d. Title
   e. Description
   f. Amount of planned hours
3. System prompts user to check fields if user entered invalid data.
4. System closes prompt and stores ticket in the database once valid data is submitted.

Edit ticket

1. User navigates to a ticket page.
2. User can click on any of the following fields to edit them:
   a. Title

   b. Planned hours (if course is in planning mode)
   c. Owner
   d. Week due
   e. Week completed
   f. Description
3. System stores changes in database.
4. Users can also log work on a specific ticket by clicking the "Log Work" button on the ticket page and entering:
   a. The number of hours worked
   b. The week these hours were worked in
5. System stores actual hours in database and updates the ticket page to display the most current amount of actual hours.
6. Users can also comment on a specific ticket by clicking the "Comment" button on the ticket page and entering a comment.
7. System stores comment in database and updates the ticket page to display the new comment.

Generate reports

1. User logs in and system directs user to the dashboard.
2. System displays earned value charts for all of the projects that the user is a member of.
3. User can download any of the charts individually by clicking the "Save <Project Name> Graph" button under the specific chart.
4. System generates image file (.PNG) of chart and provides it to the user.

Login

1. User navigates to the home URL of the Random Fruit installation.
2. User enters:
   a. A username
   b. A password
3. When the "Sign In" button is clicked, the following can occur:
   a. If one or more of the text fields is left blank an error message is displayed noting the missing data.
   b. If both fields are filled, a query is made to the database looking for the username and password provided.
      i. If the values are found to be valid, then the User is redirected to the main page with additional options available based on their user permissions.
      ii. If the values are invalid, an error message is displayed.

Search tasks

1. User selects search box at top of page in the green navigation bar.
2. User enters search criteria and clicks eye glass button or presses enter.
3. System returns table of tickets with matching data (blank page if no results are found).
4. User can filter results by clicking on the filter icon on the far right of the table headings.

Edit profile

1. User clicks the icon:  located in the upper-right corner of the page.
2. System displays a prompt and the user enters the following information to change his/her password:
    a. Old password
    b. New password
    c. New password (again for confirmation)
3. System prompts user to check fields if the user entered invalid data.
4. System closes prompt and stores changes in database once valid data is submitted.

Non-Functional Requirements

1. The system must accommodate a reasonable variety of operating systems and web servers.
2. The system must fit in the amount of space likely to be available on a university computer science department's server.
3. The system must be highly available and quickly consistent.
4. The system must be stable after a single installation.