Software Design Document

for

Community Project Tracking

CS 472

Draft 0.2

Prepared by:
Tim Grediagin
Devin Jones
Brent Mello
Blake Eggemeyer

May 7, 2012

Contents

1 Introduction

1.1 Overview

This document is intended to be used by the developer of the TODO software.

1.2 Stakeholders

The stakeholder in the design is also the client.

1.3 Definitions

1.3.0.1 Should: Requirements with this marker are desired, but not crucial, and will be a part of the final deliverable contingent on time and progress.

1.3.0.2 User: The person, or persons, who operate or interact directly with the product.

1.3.0.3 Will: Requirements with this marker are guaranteed to be in the final delivered product.

1.3.0.4 Item: Item refers to a data object. This object is either an appointment or a task.

1.3.0.5 CSV: CSV is an acronym for Comma Separated Value. This is a standard and common format of simple tabular data.

1.3.0.6 GUI: Acronym for Graphical User Interface. Used to refer to the look and feel the user experiences.

1.3.0.7 Immediately: Immediately refers to actions that will begin as soon as the user has given the input for the action to occur.

1.3.0.8 Client: Julie Engfer, the Office Manager for Festival of Fairbanks.

1.3.0.9 Administrator: A user with special permissions as specified in section 3.1.2 User Management.

1.3.0.10 CPT: The name of this application.

1.3.0.11 Worker: The person(s) responsible for the hours worked in a Project. Users are also Workers, but there may be Workers that are not Users, e.g., "boyscouts."

1.3.0.12 Project: Seen at top of timesheet, e.g., "Bicycle Path."

1.3.0.13 Program: A project which occurs annually, e.g., "Clean Team."

1.3.0.14 Activity: The specific type of work a Worker does, e.g., "Ice Chipping."

1.3.0.15 Location: The place where an activity is done, e.g., "CORE 1st - 3rd."

1.3.0.16 Tool/Equipment: The implement used to complete an Activity. Corresponds to "Equipment Used" on the original timesheet.

1.3.0.17 Comment: A remark a User may optionally provide on the timesheet.

1.3.0.18 Timesheet: The name for the web page on which the various data are entered.

1.4 References

The `1998 - IEEE Standard for Information Technology - Systems Design - Software Design Descriptions` was referenced to produce this document.

1.5 Revision tracking

0.1	Feb 16	Empty document created.
0.2	April 26	Submitted for Review.

2 Design Considerations

2.1 Programming Languages

The project will be completed using Groovy on Grails to produce a java web application.

2.2 Database

The H2 database manager is the default database manager used by Grails and according to information from third parties is sufficient to handle the task necessary.

2.3 Project Management

This project will use `Git` version control in conjunction with `GitHub` to keep track of changes. The repository can be reached at `https://github.com/blake6489/Community-Project-Tracking`.

3 Domains

3.1 User

There will be two types of user accounts: admin and employee. Both types of accounts will have unique usernames and passwords. Employee accounts will have the following permissions: fill in and submit the current time sheet, and view past timesheets submitted by that specific account. Admin accounts will have the following permissions: create/delete admin or employee accounts, view usernames and passwords of all accounts, view and edit all submitted timesheets, create and submit timesheets on behalf of volunteer group, and access to automated report generation.

3.2 Activity

3.2.0.1 constraints Sets constraints for name and uniqueName variables.

3.2.0.2 name Max of 250 characters, alphanumeric and spaces.

3.2.0.3 uniqueName Max of 250 characters, alphanumeric and spaces set to lower case.

3.2.0.4 setName Sets both variables and makes uniqueName's characters lower case.

3.3 Location

3.3.0.1 constraints Sets constraints for name and uniqueName variables.

3.3.0.2 name Max of 250 characters, alphanumeric and spaces.

3.3.0.3 uniqueName Max of 250 characters, alphanumeric and spaces set to lower case.

3.3.0.4 setName Assigns values for name and uniqueName variables and makes uniqueName's characters lower case.

3.4 Project

3.4.0.1 constraints Sets constraints for project name variable.

3.4.0.2 name 250 characters, alphanumeric and spaces.

3.4.0.3 hasMany Initializes hasMany variable to the Timesheet template.

3.5 Role

3.5.0.1 constraints Sets constraints for authority variable.

3.5.0.2 mapping Sets the use of cache to true.

3.5.0.3 authority Specifies if worker account or admin account (ex: ROLE$_{A}DMIN$).

3.5.0.4 type A printable representation of authority (ex: admin).

3.6 Timesheet

3.6.0.1 constraints Sets constraints for date, project, template, and worker variables.

3.6.0.2 hasMany Initializes hasMany variable to the workRecord record.

3.6.0.3 date The date of the timesheet.

3.6.0.4 project The project the timesheet belongs to.

3.6.0.5 template The template for the timesheet.

3.6.0.6 worker The worker associated with the timesheet.

3.7 Tool

3.7.0.1 constraints Sets constraints for name and uniqueName variables.

3.7.0.2 name 250 characters, alphanumeric and spaces.

3.7.0.3 uniqueName 250 characters, alphanumeric and spaces set to lower case.

3.7.0.4 setName Assigns values for name and uniqueName variables and makes uniqueName's characters lower case.

3.8 User

3.8.0.1 beforeInsert Function called by insert function. Encodes the password as it is inserted.

3.8.0.2 beforeUpdate Function called by update function. Checks if the password needs to be encoded.

3.8.0.3 constraints Sets constraints for username, uniqueUsername, password, type, and worker.

3.8.0.4 createWithUserRole Creates a user with a userRole. Same purpose as the save() function in the controllers.

3.8.0.5 encodePassword Changes password variable to encoded form of the password.

3.8.0.6 password The password for the user account.

3.8.0.7 setUsername Assigns values for username and uniqueUsername variables and makes uniqueName's characters lower case.

3.8.0.8 type Either 'user' or 'admin'.

3.8.0.9 uniqueUsername The same value as the username variable, only set to lower case.

3.8.0.10 username Validates that username has a unique object id.

3.9 Worker

3.9.0.1 constraints Sets constraints for name variable.

3.9.0.2 name The first and last name of the worker.

3.10 Work Record

3.10.0.1 mapping Assigns identifiable names to the indices of the variables: project, timesheet, worker, and date.

4 Controllers

4.1 Activity

4.1.0.2 create Creates activity object with get request. Saves existing activity with post request.

4.1.0.3 delete Calls activity object's destructor.

4.1.0.4 index Redirects to list() function.

4.1.0.5 list Returns the parameters of the activity object and the activity's total count.

4.1.0.6 save Stores activity object in database and updates activity's total count.

4.1.0.7 show Fetches parameters of activity object with get request. Calls update() with post request.

4.1.0.8 update Changes activity object's parameters.

4.2 Home

4.2.0.9 grails Renders the appropriate home page for the user/admin.

4.2.0.10 home Redirects user to either adminHome or userHome based on the account/role being used.

4.3 Location

4.3.0.11 create Creates location object with get request. Saves existing location with post request.

4.3.0.12 delete Calls location object's destructor.

4.3.0.13 index Redirects to list() function.

4.3.0.14 list Returns the parameters of the location object and the location's total count.

4.3.0.15 save Stores location object in database and updates location's total count.

4.3.0.16 show Fetches parameters of location object with get request. Calls update() with post request.

4.3.0.17 update Changes location object's parameters.

4.4 Login

4.4.0.18 ajaxDenied Renders message for when the redirect url is denied.

4.4.0.19 ajaxSuccess Renders message for when the redirect url is accepted.

4.4.0.20 authfail Serves as the callback for a failed login. Redirects to the authentication page with the appropriate warning message (ex: springSecurity.errors.login.expired).

4.4.0.21 denied Redirects to the login denied page.

4.4.0.22 index Redirects to login() function.

4.4.0.23 full Renders login page for browsers with password remembering cookie.

4.4.0.24 login Renders login page for browsers without password remembering cookie.

4.5 Logout

4.5.0.25 index Redirects to the Spring security logout uri.

4.6 Project

4.6.0.26 create Creates project object with get request. Saves existing project with post request.

4.6.0.27 delete Calls project object's destructor.

4.6.0.28 index Redirects to list() function.

4.6.0.29 list Returns the parameters of the project object and the project's total count.

4.6.0.30 save Stores project object in database and updates project's total count.

4.6.0.31 show Fetches parameters of project object with get request. Calls update() with post request.

4.6.0.32 update Changes project object's parameters.

4.7 Tool

4.7.0.33 create Creates tool object with get request. Saves existing tool with post request.

4.7.0.34 delete Calls tool object's destructor.

4.7.0.35 index Redirects to list() function.

4.7.0.36 list Returns the parameters of the tool object and the tool's total count.

4.7.0.37 save Stores tool object in database and updates tool's total count.

4.7.0.38 show Fetches parameters of tool object with get request. Calls update() with post request.

4.7.0.39 update Changes tool object's parameters.

4.8 User

4.8.0.40 create Creates user object with get request. Saves existing user with post request.

4.8.0.41 index Redirects to list() function.

4.8.0.42 list Sets user parameters min to 10 and max to 100. Returns the parameters of the user object and the user's total count.

4.8.0.43 resetPassword Changes user's login password or displays appropriate error message.

4.8.0.44 save Validates user's password and sets instance variables (ex: accountExpired = false).

4.8.0.45 show If logged in, lists the parameters of the logged in user.

5 Views

5.1 Layouts

Layouts are application-wide views not tied to any specific controller. Layouts will be in the "views/layouts" folder.

5.1.1 application

This is the application-wide layout. It's responsible for deciding what layout templates to render based on user access levels. It must also render flash messages to alleviate other views from having to do so.

5.1.2 _login

This template renders when the user hasn't logged in. It'll contain an ajax form allowing the user to login. Once the user has successfully logged in, the page will refresh.

5.1.3 _admin

This template renders when an admin is logged in. It'll contain links to parts of the application admins can access.

5.1.4 _user

This template renders when a non-admin is logged in. Not sure what it'll show.

6 Interface

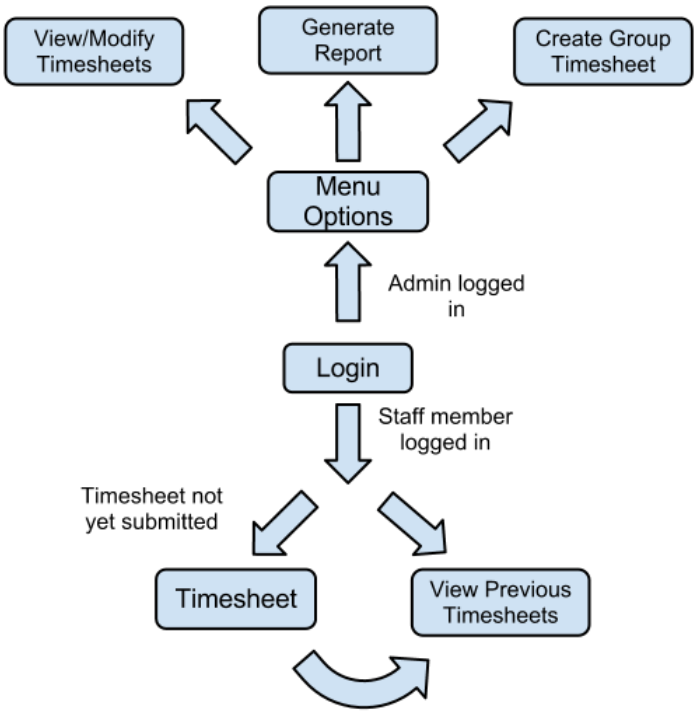

Figure 1: Diagram of user travel through site

7 Test Cases

8 Traceability matrix

	Functional						Non-functional	
	3.1.1	3.1.2	3.1.3	3.1.4	3.1.5	3.1.6	3.2.1	3.2.2
3.2								
3.3								
3.4								
3.5	✓							
3.6								
3.7								
3.8	✓							
3.9	✓							
3.10								
4.1								
4.2								
4.3								
4.4	✓							
4.6								
4.7								
4.8	✓							