Deployment Plan

Stay Fit | Towson University

COSc 412 - 101

Group 3

2017

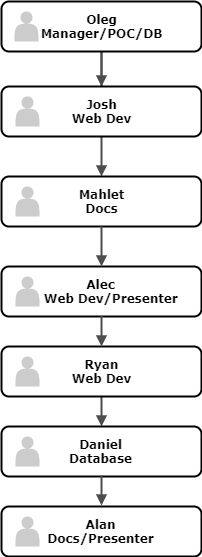
# Purpose

The purpose of the Deployment Plan document is to define a strategy and plan for how the software application will be implemented. It will include information on scheduling, deployment methods, development and production environment, support and testing considerations, training requirements, and evolution procedures.

# Scheduling and Roles

The following chart shows how the group progressed from receiving the requirements from the client to deploying the final application:

|  |  |
| --- | --- |
| Date | Progress |
| September 14, 2017 | Received requirements from client, planning and role assignment |
| September 21, 2017 | Created use case and sequence diagrams |
| September 28, 2017 | Created HLA, class diagram & interface spec. |
| October 12, 2017 | Created SPMP |
| October 19, 2017 | Presented preliminary documentation to client & began to development on application |
| October 24, 2017 | Client provided requirement change |
| November 2, 2017 | Code review criteria received |
| November 9, 2017 | Code review and requirement changes made |
| December 1, 2017 | Began testing |
| December 14, 2017 | Final application deployed and presented to client |
| Indefinitely | Support and management on the application |



# Development Environment

* Code is committed to GitHub, for ease of version control management
* Developers are using their own machines with such software installed: Ruby on Rails, Notepad++, localhost server for change testing.
* Ruby on Rails implementation
  + Model-View-Controller framework
  + Provides default structures for web service, database, web pages
  + Very good support and documentation on Heroku page, as well as third parties

# Test Environment

Every Rails application is automatically divided into three environments: development, testing, and production. There are several subdirectories in the test folder that store unit tests, integration tests, system tests, and others. You can implement your own skeleton tests in a separate environment, but Rails will automatically create skeleton tests for you as you add components to the application. Additionally, a dedicated sample database can be loaded into ‘config/databse.yml’ which can manipulate sample data without changing development, or production data. For larger scale system testing, Rails implements Capybara, which delivers web-based automation so you can simulate the application in the way a regular user would interact with it. Tests can be run in a command line environment, as shown by the following examples:

* “$rails test /path/to/specific/test.rb”
  + This will run a single ruby test case
  + adding “:(number)” after .rb will run only the test method on that line number
* “$rails test /path/to/directory”
  + This will run all test cases in the specified directory
* “$rails test:system”
  + This command runs a system test

# Production Environment

* Heroku application linked to the domain: stayfitt.us
* Application hosted on Heroku
  + Deployed to Heroku directly from GitHub repository; changes pushed with each commit to the master branch
* Database hosted on Heroku with PostgreSQL
  + Stores data securely and returns data in response to requests from the application

The website will be hosted on a server through Heroku, a cloud platform that provides developers with the tools to deploy, manage, and scale their applications. End-users will be able to interact with the application once it is running and perform the necessary tasks such as, creating an account, logging in, posting messages, etc. User data is stored on the database and is only accessible by administrators.

# Training

Training will be very straightforward and differ based on user role:

1. User
   1. A simple video guide will be shown on a student’s first log in that will take them through the application’s basic functions such as, editing their profile, posting messages, and creating schedules.
   2. Students can also be taught through hands-on demonstration by their teachers and trainers
   3. The Help section allows students to re-learn anything as needed
2. Moderators
   1. Moderators will also be given a video guide on log in that will take them through the application’s basic functions and the additional moderator functions such as, approve student posts, suspending students, and adding students to teams.
   2. Moderators will be able to learn hands-on from the administrators at their facility
   3. A Help section on the moderator account will allow them to re-learn anything as needed
3. Administrators
   1. Selected administrators will be taught the moderator functions with the same video. The additional administrator functions such as, adding and deleting users, and verifying vendors, will be explained in a tutorial document. Administrators can also contact the developers for any extra explanations.

# Evolution and Documentation

Updates to the application will be worked on and tested separately, and then pushed in and deployed as an updated version. Of course, users will be notified ahead of time when updates will be made and will not be able to use the service during the specified update period. Working demos and tutorial videos will be provided along with any new functionality for each user role.

All documentation will be stored and kept up-to-date. Documentation that will be kept includes:

* Tutorial documents
* Code
* Licensing & Support Documents
* Requirements & Changes