# Canvas LMS Testing Framework Report


## Don’t Test Me

Kenneth Dela Cruz, Kyle Glick, Sam Word

# Table of Contents

# Chapter 1 - Choosing a Project

---

## I. Canvas LMS

### Language(s):

Ruby, Javascript (Coffeescript), HTML+CSS

### Framework(s) and Tools:

Ruby on Rails, Node.js, PostgreSQL

### About:

Canvas is a well-established (circa 2010, used in many colleges, universities, and K-12 schools), open-source LMS by Instructure Inc. It is released under the AGPLv3 license for use by anyone interested in learning more about or using learning management systems.

### Why we picked it:

- ➢ It has extensive documentation on Github
- ➢ The Github repository is being actively maintained and still has active development
- ➢ Because Canvas uses Ruby on Rails, it allows us to write tests and script in Ruby
- ➢ Canvas provides a script to automatically set up the application from scratch (though, it did not work in our experience on Ubuntu 16.04)
- ➢ It is a massive web application with a clear MVC architecture and extensive test suite written in Ruby.
- ➢ It cleverly combines Node.js and Ruby on Rails, meaning each team member has something to gain from the project.

# II. Installation Report

Problems and difficulties were encountered when attempting to build Canvas on an Ubuntu image. On their GitHub page, Canvas provided a Quick-Start Guide which included a set of instructions for using their quick-start script and another set without using the script. Our first hurdle came from attempting to use their automated script. The script failed at some point during its execution, and being unaware of the details of the script, we chose to set up Canvas manually. Most of the issues we encountered while manually installing Canvas were a result of our inexperience with installing Ruby on Rails projects and their dependencies. For instance, two of us installed an incorrect version of Ruby, causing problems later on in the process, and we all encountered issues with permissions in PostgreSQL. We often resolved these issues by searching online for a way to undo our mistakes and trying again.

Unfortunately not all of the obstacles we encountered were as trivial as installing the wrong Ruby version. In fact, the most difficult issue we encountered surfaced while running a built in Rake task to install all of the required assets. Canvas uses a custom fork of jQuery 1.7.2 with a very, very small change. When Canvas’s asset management tool, Yarn, tried to retrieve the code from that jQuery repository, it complained about an Invalid Tar Header coming from the source of the repository. We were originally unable to get this version of jQuery to run, so we tested various jQuery releases, eventually finding that version 2.2.4 would compile. However, when running the application, nothing would load on the page. It turned out that jQuery wasn’t being defined using version 2.2.4. After struggling with this issue for awhile, we decided to write an Issue on Canvas’s Github page. While writing the issue, we reset the jQuery version used by Yarn to get a screenshot of the error to find that the error no longer appeared. Surprised, once we finished compiling the assets again and ran the application, everything functioned as intended. We’re still not entirely sure what fixed this issue, but we believe that adding several asset folders and files, a step only mentioned in the Production Start Guide, may have resolved our issue.

On our team Github wiki, we provided step-by-step instructions for settings up Canvas. Many of the steps were taken from Canvas’s Quick Start Guide, though a few additions were made that were required to get the application running. Some of these were manual edits that we believe are Ubuntu-only issues, such as providing access to the PostgreSQL server, while others were steps in the Canvas Production Start Guide that were not in the Quick Start Guide.

# III. Custom Installation Guide

This is a guide on setting up Canvas LMS by Instructure Inc. in an Ubuntu environment based on our experiences. The official development and production start guides can be found on the Canvas GitHub:

- Official Quick Start Guide
- Official Production Start Guide

## Set Up Git

```
sudo apt-get install git
git clone https://github.com/instructure/canvas-lms.git canvas
cd canvas
```

## Install Curl

```
sudo apt install curl
```

## Install Node.js and PostgreSQL

```
curl -sL https://deb.nodesource.com/setup_6.x | sudo bash -
sudo apt-get install nodejs postgresql
```

## Install Ruby and Ruby Dependencies

```
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:brightbox/ruby-ng
sudo apt-get update
sudo apt-get install ruby2.4 ruby2.4-dev zlib1g-dev libxml2-dev
libsqlite3-dev libpq-dev libxmlsec1-dev curl make g++
```

## Create PostgreSQL Superuser

```
sudo -u postgres createuser $USER
sudo -u postgres psql -c "ALTER USER $USER WITH SUPERUSER" postgres
```

## Configure Gem Installation Directory

Note: You will have to export the GEM_HOME directory for each new bash session. If you don't want to do this, the Quick Start guide explains how to permanently link to the gem installation directory.

```
mkdir ~/.gems
export GEM_HOME=~/.gems
```

## Install Bundler

```
gem install bundler -v 1.15.3
```

## Install Yarn

```
curl -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -
echo "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee
/etc/apt/sources.list.d/yarn.list
sudo apt-get update && sudo apt-get install yarn
```

## Install Python (for Yarn)

```
sudo apt-get install python
```

## Install Gems

```
$GEM_HOME/bin/bundle install
```

## Install CoffeeScript

```
sudo npm install -g coffee-script@1.6.2
```

## Set Up Config YML Files

```
for config in amazon_s3 delayed_jobs domain file_store outgoing_mail
security external_migration; do cp -v config/$config.yml.example
config/$config.yml; done
```

## Set Up Database Config YML File

```
cp config/database.yml.example config/database.yml
```

## Create Tables and Populate Data

Note: The email and password you enter here will be your login credentials later.

```
$GEM_HOME/bin/bundle exec rake db:initial_setup
```

## Create Test Database Admin

```
psql -c 'CREATE USER canvas' -d postgres
psql -c 'ALTER USER canvas CREATEDB' -d postgres
```

## Give PostgreSQL Permissions

```
sudo gedit /etc/postgresql/9.5/main/pg_hba.conf
```

Change `peer` to `trust` where it's used.

## Create Test Database

```
createdb -U canvas canvas_test
psql -c 'GRANT ALL PRIVILEGES ON DATABASE canvas_test TO canvas' -d
canvas_test
psql -c 'GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO canvas' -d
canvas_test
psql -c 'GRANT ALL PRIVILEGES ON ALL SEQUENCES IN SCHEMA public TO canvas'
-d canvas_test
psql -c "ALTER USER canvas WITH PASSWORD 'password'" -d canvas_test
```

## Configure Test Database Credentials

```
gedit config/database.yml
```

Add `password: password` inside of the test environment YML `test:`

## Set Up Test Database Tables

```
RAILS_ENV=test $GEM_HOME/bin/bundle exec rake db:test:reset
```

### Create Asset Files and Directories

```
mkdir -p log tmp/pids public/assets app/stylesheets/brandable_css_brands
touch app/stylesheets/_brandable_variables_defaults_autogenerated.scss
touch Gemfile.lock
sudo chown -R canvasuser config/environment.rb log tmp public/assets
app/stylesheets/_brandable_variables_defaults_autogenerated.scss
app/stylesheets/brandable_css_brands Gemfile.lock config.ru
```

### Compile Assets

```
yarn install
$GEM_HOME/bin/bundle exec rake canvas:compile_assets
$GEM_HOME/bin/bundle exec rake brand_configs:generate_and_upload_all
sudo chown -R canvasuser public/dist/brandable_css
```

### Run Server

```
$GEM_HOME/bin/bundle exec rails server
```

# Chapter 2 - Test Specifications

## I. Test Report

### The testing process

Our test suite is testing methods from various model files in the Canvas LMS system.

### Testing schedule

10/03/2017 - 5 Test Specifications
10/10/2017 - 10 Test Specifications
10/17/2017 - 15 Test Specifications
10/24/2017 - 20 Test Specifications
10/31/2017 - 25 Test Specifications / 5 Running Tests
11/07/2017 - 15 Running Tests
11/14/2017 - 25 Running Tests
11/21/2017 - 5 Faults Implemented

### Test recording procedures

Results of the tests will be recorded in an HTML document and opened in a Web Browser at the end of the suite. The individual test reports will include the test number, requirement being tested, method tested, inputs, expected outcome, test outcome, and pass/fail.

### Hardware and software requirements

* Other versions may be used, but have not been tested

- Ubuntu 16.04 *
- Ruby version 2.4.2 *
- PostgreSQL 9.5.10 *
- Ruby Bundler 1.5.3

### Constraints

Our team is not familiar with the Canvas LMS system. This makes it difficult for us to find and confirm test cases and requirements. Various requirements are not specifically documented for us to use. We also have to determine how to use Canvas’s code in our tests.

### Tested Items

1. Course#score_to_grade(score)
2. LatePolicy#missing_points_deducted(points_possible, grading_type)

# II. Test Cases

## Example Test Case

1. Test ID
2. Test Requirements
3. Test Model
4. Test Method
5. Test Parameters
6. Expected Value
7. Test Executable Name

## Test Case 1

1. 01
2. Converts a score of 0 to a grade when the grading scale is [[“Pass”,50],[“Fail”,0]]
3. Course
4. score_to_grade(score)
5. 0 [[“Pass”,0.50],[“Fail”,0.0]]
6. Fail
7. course_score_to_grade_test

## Test Case 2

1. 02
2. Converts a score of 25 to a grade when the grading scale is [[“Pass”,50],[“Fail”,0]]
3. Course
4. score_to_grade(score)
5. 25 [[“Pass”,0.50],[“Fail”,0.0]]
6. Fail
7. course_score_to_grade_test

## Test Case 3

1. 03
2. Converts a score of 49.999 to a grade when the grading scale is [[“Pass”,50],[“Fail”,0]]
3. Course
4. score_to_grade(score)
5. 49.999 [[“Pass”,0.50],[“Fail”,0.0]]
6. Fail
7. course_score_to_grade_test

## Test Case 4

1. 04
2. Converts a score of 50 to a grade when the grading scale is [[“Pass”,50],[“Fail”,0]]
3. Course
4. score_to_grade(score)
5. 50 [[“Pass”,0.50],[“Fail”,0.0]]
6. Pass
7. course_score_to_grade_test

## Test Case 5

1. 05
2. Converts a score of 100 to a grade when the grading scale is [[“Pass”,50],[“Fail”,0]]
3. Course
4. score_to_grade(score)
5. 100 [[“Pass”,0.50],[“Fail”,0.0]]
6. Pass
7. course_score_to_grade_test

## Test Case 6

1. 06
2. Calculates the points to be deducted when the missing submission deduction is 100%, the points possible is 80 points, and the grading scale is Pass or Fail
3. LatePolicy
4. missing_points_deducted(points_possible, grading_type)
5. 100 80 pass_fail
6. 80.0
7. late_policy_missing_points_deducted_test

## Test Case 7

1. 07
2. Calculates the points to be deducted when the missing submission deduction is 50%, the points possible is 80 points, and the grading scale is not Pass or Fail
3. LatePolicy
4. missing_points_deducted(points_possible, grading_type)
5. 50 80 string
6. 40.0
7. late_policy_missing_points_deducted_test

## Test Case 8

1. 08
2. Calculates the points to be deducted when the missing submission deduction is 0%, the points possible is 100 points, and the grading scale is not Pass or Fail
3. LatePolicy
4. missing_points_deducted(points_possible, grading_type)
5. 0 100 string
6. 0.0
7. late_policy_missing_points_deducted_test

## Test Case 9

1. 09
2. Calculates the points to be deducted when the missing submission deduction is 100%, the points possible is 50 points, and the grading scale is not Pass or Fail
3. LatePolicy
4. missing_points_deducted(points_possible, grading_type)
5. 100 50 string
6. 50.0
7. late_policy_missing_points_deducted_test

## Test Case 10

**Test File**

1. 10
2. Calculates the points to be deducted when the missing submission deduction is 25%, the points possible is 9000 points, and the grading scale is not Pass or Fail
3. LatePolicy
4. missing_points_deducted(points_possible, grading_type)
5. 25 9000 string
6. 2250.0
7. late_policy_missing_points_deducted_test

## Test Case 11

1. 11
2. Calculates points when the missing submission deduction is 100%, the points possible is 80 points, and the grading scale is Pass or Fail
3. LatePolicy
4. points_for_missing(points_possible, grading_type)
5. 100 80 pass_fail
6. 0.0
7. late_policy_points_for_missing_test

## Test Case 12

1. 12
2. Calculates points when the missing submission deduction is 50%, the points possible is 80 points, and the grading scale is not Pass or Fail
3. LatePolicy
4. points_for_missing(points_possible, grading_type)
5. 50 80 string
6. 40.0
7. late_policy_points_for_missing_test

## Test Case 13

1. 13
2. Calculates points when the missing submission deduction is 0%, the points possible is 100 points, and the grading scale is not Pass or Fail
3. LatePolicy
4. points_for_missing(points_possible, grading_type)
5. 0 100 string
6. 100.0
7. late_policy_points_for_missing_test

## Test Case 14

1. 14
2. Calculates points when the missing submission deduction is 100%, the points possible is 50 points, and the grading scale is not Pass or Fail
3. LatePolicy
4. points_for_missing(points_possible, grading_type)
5. 100 50 string
6. 0.0
7. late_policy_points_for_missing_test

## Test Case 15

1. 15
2. Calculates points when the missing submission deduction is 25%, the points possible is 9000 points, and the grading scale is not Pass or Fail
3. LatePolicy
4. points_for_missing(points_possible, grading_type)
5. 25 9000 string
6. 6750.0
7. late_policy_points_for_missing_test

## Test Case 16

1. 16
2. Assigns an incomplete grade when the score provided is nil for a pass-fail assignment
3. Assignment
4. score_to_grade(score=0.0, given_grade=nil)
5. pass_fail 0.0 nil
6. incomplete
7. assignment_score_to_grade_test

## Test Case 17

1. 17
2. Correctly calculates a grade on a letter-grade assignment
3. Assignment
4. score_to_grade(score=0.0, given_grade=nil)
5. letter_grade 10 8.7
6. B+
7. assignment_score_to_grade_test

## Test Case 18

1. 18
2. Correctly calculates a grade on a letter-grade assignment when provided a long decimal grade
3. Assignment
4. score_to_grade(score=0.0, given_grade=nil)
5. letter_grade 10 8.69999
6. B
7. assignment_score_to_grade_test

## Test Case 19

1. 19
2. Correctly calculates a grade on a percent assignment with nil points possible
3. Assignment
4. score_to_grade(score=0.0, given_grade=nil)
5. percent nil 5.0
6. 5%
7. assignment_score_to_grade_test

## Test Case 20

1. 20
2. Correctly calculates a grade on a percent assignment and non-nil points possible
3. Assignment
4. score_to_grade(score=0.0, given_grade=nil)
5. percent 100 59.78
6. 59.78%
7. assignment_score_to_grade_test

## Test Case 21

1. 21
2. Ignores ungraded assignments
3. LatePolicy
4. points_deducted(score: nil, possible: 0.0, late_for: 0.0, grading_type: nil)
5. 5.0 hour true 10 nil 1 day points
6. 0.0
7. late_policy_points_deducted_test

## Test Case 22

1. 22
2. Ignores pass-fail assignments
3. LatePolicy
4. points_deducted(score: nil, possible: 0.0, late_for: 0.0, grading_type: nil)
5. 5.0 hour true 10 nil 1 day pass_fail
6. 0.0
7. late_policy_points_deducted_test

## Test Case 23

1. 23
2. Ignores assignments that are not meant to be graded
3. LatePolicy
4. points_deducted(score: nil, possible: 0.0, late_for: 0.0, grading_type: nil)
5. 5.0 hour true 10 nil 1 day not_graded
6. 0.0
7. late_policy_points_deducted_test

## Test Case 24

1. 24
2. Correctly deducts points on a per-hour basis
3. LatePolicy
4. points_deducted(score: nil, possible: 0.0, late_for: 0.0, grading_type: nil)
5. 5.0 hour true 1000 1000 12 hours points
6. 600.0
7. late_policy_points_deducted_test

## Test Case 25

1. 25
2. Correctly deducts points on a per-day basis
3. LatePolicy
4. points_deducted(score: nil, possible: 0.0, late_for: 0.0, grading_type: nil)
5. 15.0 day true 1000 1000 3 days points
6. 450.0
7. late_policy_points_deducted_test

# Chapter 3 - Building a Framework

## I. Framework Architecture

Our test framework is run by executing the *runAllTest.sh* script in the *Don-t-Test-Me/Canvas/TestAutomation/* directory. The script iterates over test case files which contain details for each test case, then executes the corresponding Ruby test executable, providing the inputs and expected result for the test. Each test inherit from a class we developed called *Testatron*. This class provides generic I/O handling for each test case. The class validates parameter and result inputs and returns the result of the test to our script. Each test executable contains a class which inherits from *Testatron*, providing an initializer and a *run* method. After executing a test, *runAllTest.sh* gathers the test information, including an id, the model being tested, method being tested, inputs, and result of the test, and creates an HTML table to be displayed in a web browser at the end of the script.

# II. How-To Guide

## Install Git

```
sudo apt-get install git
```

## Clone our Repo

```
git clone https://github.com/CSCI-362-02-2017/Don-t-Test-Me.git
cd Don-t-Test-Me/canvas/
```

## Install Ruby Libraries

```
sudo add-apt-repository ppa:brightbox/ruby-ng
```

➢ Press <return> when prompted

```
sudo apt-get update
sudo apt-get install ruby2.4 ruby2.4-dev zlib1g-dev libxml2-dev
libsqlite3-dev libpq-dev libxmlsec1-dev curl make g++
```

➢ Press <y> when prompted

## Install PostgreSQL

```
sudo apt-get install postgresql
```

➢ Press <y> when prompted

```
sudo -u postgres createuser $USER
sudo -u postgres psql -c "ALTER USER $USER WITH SUPERUSER" postgres
```

## Install Bundler and Ruby Gems

```
mkdir ~/.gems
export GEM_HOME=~/.gems
gem install bundler -v 1.15.3
$GEM_HOME/bin/bundle install
```

## Set Up Database and Data

```
cp config/database.yml.example config/database.yml
createdb canvas_development
$GEM_HOME/bin/bundle exec rake db:initial_setup
```

- ➢ Provide an email when prompted for an admin email
- ➢ Verify the email
- ➢ Provide a password when prompted for an admin password
- ➢ Verify the password
- ➢ Enter a name when prompted for an organization name (We used Don't Test Me)
- ➢ Press <3> when prompted if you want to log data for Canvas

## Run the Script

```
cd TestAutomation
./scripts/runAllTests.sh
```

# III. Team Experiences

In general, implementing the first five test cases to be run by our script was much easier than expected. Since no one on our team had any experience with scripting, we began working on our script early to give ourselves ample time to get it working. Our biggest hurdle was not writing scripts, but installing dependencies so that we could call the model methods we wanted to test. In a typical Ruby on Rails application, unit tests are run with access to the entire code base. Our test suite, however, is external to Canvas. We could have loaded all of Canvas’s source code into our test suite, but the application is massive with many of its own dependencies that have no effect on our test suite. To make it easier for others to run, we decided to install *activerecord-tableless*, a gem to essentially bypass Rails’s *Active Record* framework and avoid setting up a test database (the methods we are testing do not involve calls to a database). Unfortunately, *activerecord-tableless* is a somewhat outdated gem with poor documentation, which made it difficult to set up and use properly.

# Chapter 4 - Refining the Framework

## I. Example Report

# II. Team Experiences

In our initial work on the test suite we were worried about the size of Canvas’s code and the large number of dependencies required to run it so we decided to use only the necessary model files for our tests. However, this required us to add many commented out lines of code, as well as include a gem which would allow us to use the models with a tableless application. We decided to try and find a way to run a limited version of Canvas’s code (for sake of memory space) which includes all of the Ruby code except for their tests. After thorough research, we found that by placing our *TestAutomation* files within Canvas’s repository, we could access the *rails runner* component which allows you to run Ruby scripts in the native Rails environment. By doing this, we were able to run our tests using completely unmodified code from Canvas. This did cause an increase in set-up work for the test suite, but we believe that it makes more sense for the framework to run natively. We hope to try to write a script that will allow a new user to automatically install dependencies and set up the PostgreSQL database with ease.

After adding the condensed Canvas source code to our repository, we found that writing our final four tests was relatively simple. Running the Rails environment natively with *rails runner* gave us access to all of the models we needed, so most of our work was simply providing attributes and parameters. The Canvas source code and *Testatron* did the rest.

# Glossary of Terms

---

## JavaScript:

A high-level object-oriented programming language developed by Brendan Eich and released in 1995. It is used primarily in World Wide Web content to make interactive pages and online applications.

## jQuery:

A JavaScript library built originally by John Resig and released in 2006. It’s primary purpose is to make manipulating HTML documents easier and provide improved interactivity with document elements.

## PostgreSQL:

A popular Database Management System (DBMS) released in 1996.

## Rake (Ruby Make):

A build utility for Ruby, similar to the “make” command.

### Rake Task:

A Ruby script built to run using Rake syntax.

## Ruby:

A dynamic, general-purpose programming language developed in the mid-1990s by Yukihiro Matsumoto. It was designed with ease-of-use and flexibility in mind. It can be used for functional, object-oriented, and imperative programming.

## Ruby on Rails (Rails):

A server-side MVC web application framework in Ruby developed by David Heinemeier Hansson in 2005. It uses HTML, CSS, and JavaScript for user interfacing and JSON and XML for data transfer. It also provides a built-in test suite and a variety of external libraries for developers.

### Active Record:

A Ruby library for working with relational databases. It handles the creation and use of business objects whose data requires persistent storage to a database.

### Gem:

An external Ruby on Rails package source. They’re compiled and installed by defining the gem source in the Gemfile and running bundle install.

### Rails Runner:

A component of the Ruby on Rails framework which allows a plain Ruby script to be executed in a native Ruby on Rails environment with full access to the environment and its components

### Spec (Specification):

The name given to Ruby on Rails automated tests using the built-in Gem, RSpec.

## Yarn:

A Dependency Management System (DMS) used to compile and control packages and assets.