

Testing Report

Applicant Tracker

Client

Mosaic Learning - Carol Curley

Team 5 “Agile, Handle with Care”

Arvin Siva

Brian Wilson

Vihar Patel

James Baker

Steven Hargrove

Logan Rites

11/29/17

Table of Contents

[1. Introduction](#)

[1.1 Purpose of This Document](#)

[1.2 References](#)

[1.3 Coding and Commenting Conventions](#)

[1.4 Defect Checklist](#)

[1.4a Defect Categories](#)

[1.4b Defects](#)

[2. Code Inspection Process](#)

[2.1 Description](#)

[2.2 Impressions of the Process](#)

[2.3 Inspection Meetings](#)

[2.3.a Meeting Times](#)

[3. Modules Inspected](#)

[Code Description Table](#)

[4. Defects](#)

[Appendix A - Coding and Commenting Conventions](#)

[Appendix B – Team Review Sign-off](#)

[Appendix C – Document Contributions](#)

1. Introduction

1.1 Purpose of This Document

The purpose of this document is to provide an overview of the coding practices that were followed during the implementation of our Applicant Tracking System. Practices include coding and commenting conventions as well as any software defects, in addition to a review of all systems. Lastly, meetings are outlined.

1.2 References

- System Requirements Specification (SRS)
- System Design Document (SDD)

1.3 Coding and Commenting Conventions

We used a mix of UpperCamelCase naming convention and all lowercase naming for our classes, files and variables. For our files and classes we used UpperCamelCase and for our variables we kept them all lowercase. For example in UpperCamelCase we named our applicant controller file as “ApplicantController.php”, where there are no spaces in between each word and the first letter of each word is capitalized. For a name variable we simply kept it all lowercase as “name”.

1.4 Defect Checklist

1.4a Defect Categories

Category	Comments
Coding Convention Errors	Coding convention errors are when code deviates from the norm. This makes maintaining code difficult and is not good for application evolution.
Logic Errors	When code does not have the outcome originally intended.
Commenting Errors	Comments are not consistent or missing.
Security Oversights	Security oversights are vulnerabilities in code.

1.4b Defects

Category	Defect
Logic Errors	Interview questions did not appear at the top of the table to be ordered from most recent to latest
Logic Errors	Salary variable was a string, the wrong data type
Logic Errors	Allowed for duplicate entries
Commenting Errors	Missing Comments
Coding Convention Errors	Some variables did not follow chosen coding convention
Security Oversight	No time out function
Coding Convention Errors	Some variables hard-coded instead of being passed in
Coding Convention Errors	Code duplicated instead of being formatted as a function
Commenting Errors	Comments do not all match a uniform commenting format
Security Oversight	Password is visible when changing password

2. Code Inspection Process

2.1 Description

The code inspection was completed by both the primary and secondary coder. The primary coder would push updates as the secondary coder would record the results live. This process took longer but yielded more accurate results. The group used Github to push our changes and we drew the code and ran it on our local devices.

We did not host on a server so it presented difficulties as all members had to pull the code every time the developers made changes. This also required each member to make an identical databases, this was not ideal but worked for both the purposes of our code inspection and overall testing. Our primary coder handled the bulk of the implementation therefore it was imperative that more eyes got on the inspection to ensure that there weren't any overlooked issues.

2.2 Impressions of the Process

The code inspection process was mildly helpful, it made us try harder to make the code more simple to understand. Since most of us were available for meetings we were able to discuss verbally what parts of the code did what, and we did more of a code walkthrough between teammates in this process. To be honest, code inspection was one of the smaller pieces of our development process. Given our constrained amount of time to work on the project, implementation and testing became a much higher priority than inspecting the code. Simply put, we wanted the code to work well before we needed it to look good.

The best modular units of our code are involved with storing applicant data. All data is stored effectively and has little to no issues editing them. Some of the worst would be the user database in our system, since there is issues with access level and what they can see.

2.3 Inspection Meetings

During inspection meetings code was reviewed.

2.3.a Meeting Times

Date	Time	Attendance
11/27/17 M	4:00-4:45 PM	All Team Members
11/20/17 M	4:30-5:30 PM	All Team Members
11/17/17 F	2:30-4:00 PM	All Team Members
11/15/17 W	4:00-5:00 PM	All Team Members
11/13/17 M	4:00-5:30 PM	All Team Members
11/10/17 F	2:30-4:15 PM	All Team Members

3. Modules Inspected

Applicant Tracking System is a web application that utilizes the Laravel framework. Laravel uses a model view controller architecture that makes the code much simpler to write. A lot of the monotonous parts of our code is taken care of by Laravel already like handling url paths and controller logic for each view. Refer to the table below to view a brief description of the functionality of these files.

Code Description Table

#	File	Brief Description
1	AnswersController.php	Handles requests regarding data from the answers table. For example creating, editing answers.
2	ApplicantsController.php	Handles requests regarding data from the applicants table.
3	JobsController.php	Handles requests regarding data from the jobs table.
4	PagesController.php	Handles requests for certain pages and other miscellaneous operations.
5	QuestionsController.php	Handles requests regarding data from the questions table.
6	SourcesController.php	Handles requests regarding data from the sources table.
7	UsersController.php	Handles requests regarding data from the users table.
8	HomeController.php	Handles requests regarding the home page.
9	LoginController.php	Handles the login process for the users.
10	UpdatePasswordController.php	Handles the user's requests to changes their password.
11	RedirectIfAuthenticated.php	Checks if user logged in and redirects to homepage.

12	web.php	Define the urls and what function they are related to in one of the controllers. Known as routing.
13	applicants/create.blade.php	Provides user with a view to create an applicant. This is done in a form.
14	applicants/edit.blade.php	Provides user with a view to edit an applicant. This is done in a form.
15	applicants/index.blade.php	Provides user with a view that has a filterable table with data from the applicant table.
16	applicants/show.blade.php	Provides user with a view to specifically check the data of a particular applicant.
17	jobs/create.blade.php	Provides user with a view to create a job. This is done in a form.
18	jobs/edit.blade.php	Provides user with a view to edit a job. This is done in a form.
19	jobs/index.blade.php	Provides user with a view that has a table with data from the jobs table.
20	jobs/show.blade.php	Provides user with a view to specifically check the data of a particular job.
21	users/create.blade.php	Provides admin with a view to create a user. This is done in a form.
22	users/edit.blade.php	Provides admin with a view to edit a user. This is done in a form.
23	users/index.blade.php	Provides admin with a view that has a table with data from the users table.

24	users/show.blade.php	Provides admin with a view to specifically check the data of a particular user.
25	questions/create.blade.php	Provides admin with a view to create a question. This is done in a form.
26	questions/edit.blade.php	Provides admin with a view to edit a question. This is done in a form.
27	questions/index.blade.php	Provides admin with a view that has a filterable table with data from the questions table.
28	questions/show.blade.php	Provides admin with a view to specifically check the data of a particular question.
29	sources/create.blade.php	Provides admin with a view to create a source. This is done in a form.
30	sources/edit.blade.php	Provides admin with a view to edit a sources. This is done in a form.
31	sources/index.blade.php	Provides admin with a view that has a table with data from the sources table.
32	sources/show.blade.php	Provides admin with a view to specifically check the data of a particular source.
33	pages/createanswer.blade.php	Provides user with a view to create interview/phone screen answers for the particular applicant.
34	pages/editanswer.blade.php	Provides user with a view to edit interview/phone screen answers for the particular applicant.
35	pages/showanswers.blade.php	Provides user with a view that displays the interview/phone

		screen questions and answers for the particular applicant.
36	pages/index.blade.php	Provides user with a view that displays the front page of the application.
37	layouts/app.blade.php	Contains code that is used in all of the views (Ex: Navigation Bar) so it can be included instead of being rewritten constantly.
38	inc/messages.blade.php	Deals with error and success messages throughout the application. This is included in all the views.
39	home.blade.php	Provides user with a view of the application's homepage. View will be slightly different for admin as they have additional options.
40	user/change-password.blade.php	Provides user with a view that allows them to change their password.

4. Defects

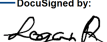
Category:Defect	Location	Comments	Fixed
Logic Error	ApplicantsController.php	The SQL query was retrieving the wrong data from the questions and answers table.	Yes
Logic Error	PagesController.php	The SQL query was retrieving the wrong data from the questions and answers table.	Yes
Security Oversight	LoginController.php	Users were able to access pages without being authenticated by manually entering the url.	Yes
Security Oversight	JobsController.php	Users were able to access this functionality without having admin privilege.	Yes
Security Oversight	QuestionsController.php	Users were able to access this functionality without having admin privilege.	Yes
Security Oversight	UsersController.php	Users were able to access this functionality without having admin privilege.	Yes
Security Oversight	SourcesController.php	Users were able to access this functionality without having admin	Yes

		privilege.	
--	--	------------	--

Appendix A – Team Review Sign-off

All team members have reviewed this document and agree on both the content and the format. Any concerns are addressed in team comments below.

Name: Logan Rites

DocuSigned by:

0C5A5B85D7104E1...

Signature

Date: 11/29/2017 | 17:42 PM EST

Comments: _____

Name: Vihar Patel

DocuSigned by:

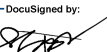
948F5B0E51CC49C

Signature

Date: 11/29/2017 | 17:48 PM EST

Comments: _____

Name: Steven Hargrove

DocuSigned by:

E2D6262EE82D4C0...

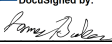
Signature

Date: 11/29/2017 | 17:47 PM EST

Comments: _____

(continued)

Name: James Baker

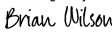
DocuSigned by:

8ABAA7A6F410402...

Signature

Date: 11/29/2017 | 17:42 PM EST

Comments: _____

Name: Brian wilson

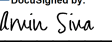
DocuSigned by:

E82CF307B59D402...

Signature

Date: 11/29/2017 | 17:54 PM EST

Comments: _____

Name: Arvin Siva

DocuSigned by:

A0211A55DE1C41A...

Signature

Date: 11/29/2017 | 17:50 PM EST

Comments: _____

Appendix B – Document Contributions

This document was worked on by all team members over two group meetings. Brian Wilson, James Baker, and Vihar Patel worked on the charts. Steven Hargrove, Logan Rites, and Arvin Siva worked on the longer paragraph descriptions and added screenshots.