

CSCE 431- Software Engineering

Induction Proof Generator

Final Report



Team 2

January 28th, 2018

Table of Contents

I. Abstract	2
II. Introduction	2
III. Motivation	2
IV. Stakeholders	2
V. User Interface	3
A. Original Concepts Generation	3
B. Current Site Demonstration	4
VI. Team Roles	5
VII. Scrum Iterations & Users Stories	5
A. Iteration 0	5
B. Iteration 1	5
C. Iteration 2	6
D. Iteration 3	6
E. Iteration 4	6
VIII. Customer Meetings	6
A. Iteration 0 - January 2, 2018	6
B. Iteration 1 - January 8, 2018	7
C. Iteration 2 - January 9, 2018	7
D. Iteration 3 - January 10, 2018	7
E. Iteration 4 - January 11, 2018	7
IX. Testing	7
A. Behavior-Driven Design	7
B. Test-Driven Design	7
X. Configuration Management	8
XI. Issues	8
XII. Implementation Environment	9
XIII. Tools and GEMS	9
XIV. Github Repository	10
XV. Other Important Links	10
XVI. References	10

I. Abstract

This project was developed in CSCE 431 - Software Engineering. Its primary focus was to practice development using the agile methodology and utilize test and behavior driven development. The objective of the project was to develop a web application that students could use to practice writing induction proofs. The project used technologies such as Ruby on Rails, Github, Javascript, and others to accomplish this task.

II. Introduction

The induction proof generator is a platform for students to practice induction proof problems. The users will be able to log in to their account and view a profile page. The platform is able to generate different inductive proof problems as practice problems for users. The platform is also able to take in inputs from the users and convert it to standard text. The goal of this project is to assist the professor to manage a large amount of students by helping each individual students to practice doing inductive proof problems. Due to the scope and duration of the project, some features are yet to be complete, such as generating solutions for problems, comparing user input, and increasing types of problems,

III. Motivation

The customer, Dr. Ritchey, requested that our project team develop an interactive application that allows for students to practice inductive proofing and performs inductive proof checking. The customer also requested that the subsets of inductive proofs be split into the categories such as: strong induction, mathematical induction, and structural induction. The application will serve as a learning platform for the students of Dr. Ritchey's CSCE-222 class, With the platform, students will be able to practice inductive proofs. Meanwhile, the teacher will be able to keep track of each student's weaknesses and strengths without spending much time looking through each script and checking it.

IV. Stakeholders

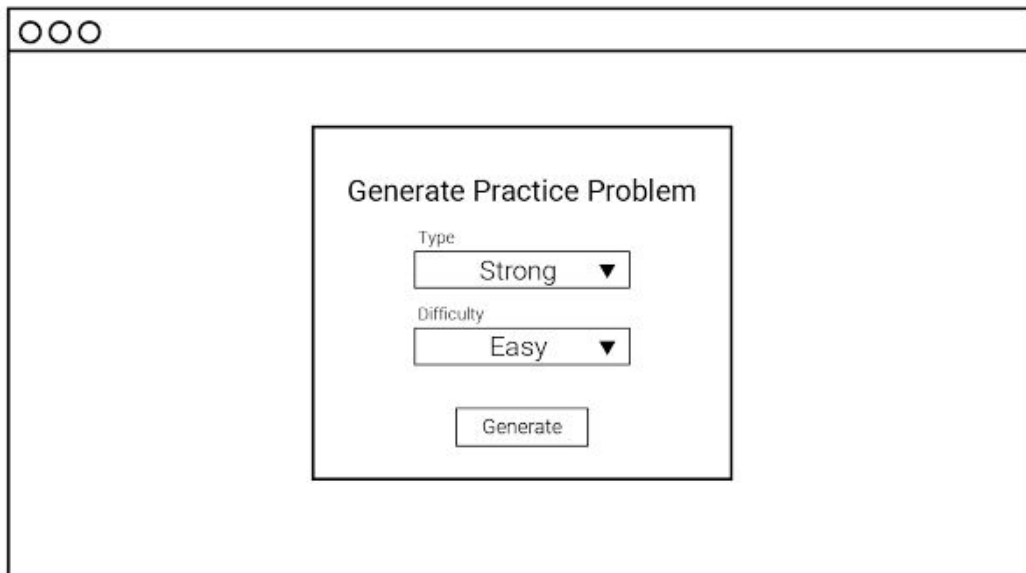
Client: Dr. Philip Ritchey

Advisor: Dr. Philip Ritchey

Team Members: Shawn Dolifka, Cheryl Goh, Marcus Heinonen, Amy Li, Nilay Patel, Tracy Yee

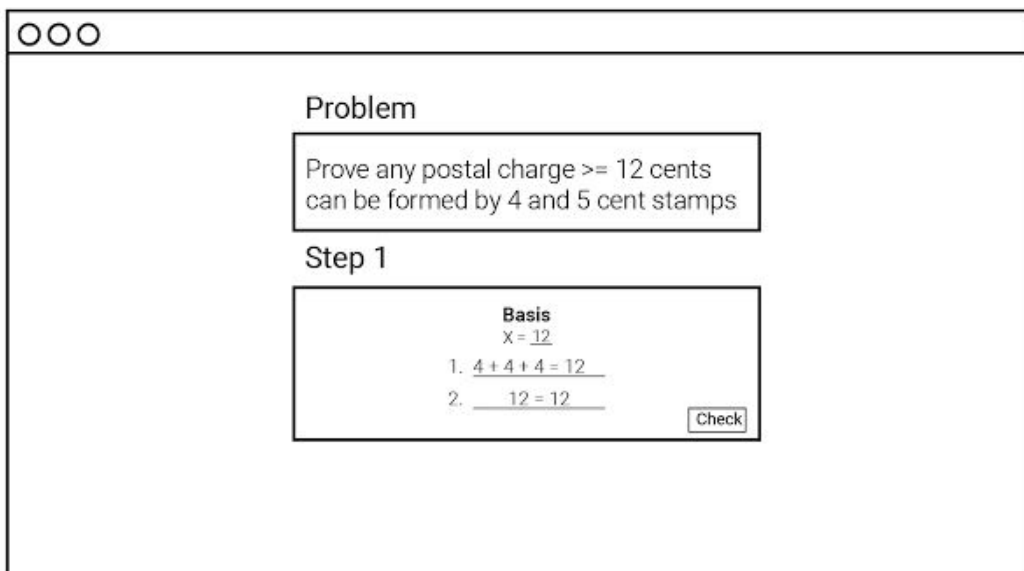
V. User Interface

A. Original Concepts Generation



A window titled "Generate Practice Problem" with three small circles in the top-left corner. Inside the window is a form with two dropdown menus. The first dropdown is labeled "Type" and has "Strong" selected. The second dropdown is labeled "Difficulty" and has "Easy" selected. Below the dropdowns is a "Generate" button.

Figure 1: User Interface Concept for Generating Problems



A window titled "Problem" with three small circles in the top-left corner. Inside the window is a text box containing the problem statement: "Prove any postal charge ≥ 12 cents can be formed by 4 and 5 cent stamps". Below the text box is a section titled "Step 1" which contains a box for the user's solution. The box contains the text "Basis" followed by "x = 12", then two numbered steps: "1. $4 + 4 + 4 = 12$ " and "2. $12 = 12$ ". To the right of the solution box is a "Check" button.

Figure 2: User Interface Concept for Completing Practice Problems

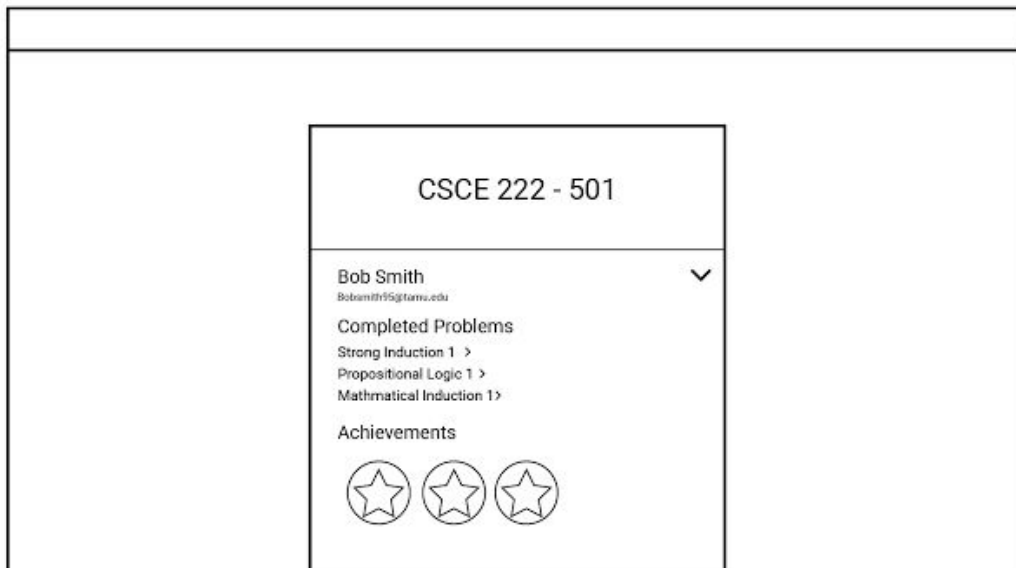


Figure 3. User Interface Concept for User Account Panel

B. Current Site Demonstration

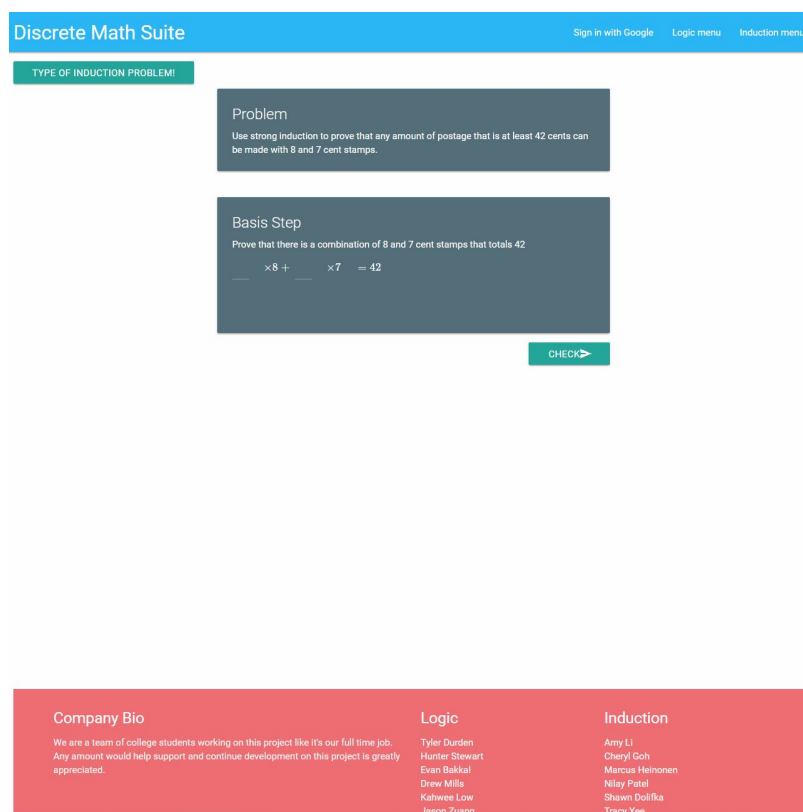


Figure 4: Final User Interface for Induction Proof Generator



Figure 5: User Interface for User Account

VI. Team Roles

Scrum master: Marcus Heinonen

Product owner: Nilay Patel

Developers: Shawn Dolifka, Cheryl Goh, Amy Li, Tracy Yee

VII. Scrum Iterations & Users Stories

A. Iteration 0

- We set up meeting times with our customer and created user stories
- Created low-fi user interface mockups to show to our customer
- Set up a Trello board and a Github repository

B. Iteration 1

- We implemented the following user stories:
 - Feature: Route for induction page - As a student, I want a page for induction problems, so that I can practice them.

- Feature: Format output in LaTeX - As a student, I want the output to be displayed in LaTeX formatting, so the page's output is clean.

C. Iteration 2

- We implemented the following user stories:
 - Feature: Allowing user to select induction problem type: - As a student, I want to be able to choose multiple types of problems, so that I can practice different types of induction problems.
 - Feature: Generating a stamp problem - As a student, I want to generate a stamp problem, so that I can practice strong induction.

D. Iteration 3

- We implemented the following user stories:
 - Feature: Google OAuth Login - As a student, I want to login with my google account, so that I can be identified securely.

E. Iteration 4

- We implemented the following user stories:
 - Feature: Showing and hiding cards - As a student, I want to show and hide the problem steps depending on the part of the problem, so that I can know which step I am on.
 - Feature: User Account Page - As a student, I want to view a user account page, so that I can see my profile and progress.

VIII. Customer Meetings

A. Iteration 0 - January 2, 2018

- We set up meeting times with our customer
- We created user stories
- We set up our tools and work environment for our project

B. Iteration 1 - January 8, 2018

- We implemented route for induction page
- We implemented formatting output in LaTeX

C. Iteration 2 - January 9, 2018

- We implemented features for allowing users to select different types of induction problems
- We implemented generating a stamp problem

D. Iteration 3 - January 10, 2018

- We implemented user login through Google account authentication
- We tested user account login through multiple real user accounts

E. Iteration 4 - January 11, 2018

- We implemented the feature to display only the relative step to the interface
- We implemented the user account page to view contents of profile and progress

IX. Testing

A. Behavior-Driven Design

Behavior-Driven Design (BDD) is a software development method in which the user behavior is defined prior to the construction of code. The user behaviors are written in common understandable languages and are then converted to automated test scripts. During the development process, the coding scripts are run against the test scripts to ensure that all the expected features are implemented correctly. [6]

B. Test-Driven Design

Test-Driven Design (TDD) is a software development method in which there are test cases written prior to the construction of code. During the development process, the coding scripts are run against the test cases to check the features are implemented correctly. Meanwhile, the development team can refactor the code to produce a validated deliverable at the end of sprint. [6]

The following figure demonstrates a BDD loop versus a TDD loop.

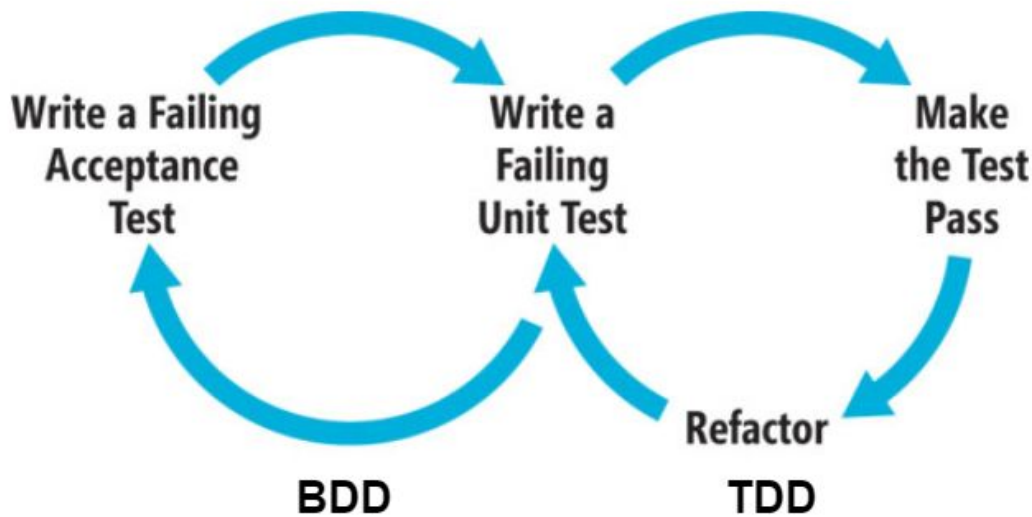


Figure 6: BDD versus TDD [7]

X. Configuration Management

In order to ensure code checking and efficiency we created a GitHub suite for our project along with several branches for the entire project which included an induction branch and some sub branches for specific major tasks in order to ensure that merge conflicts wouldn't arise. This was also crucial in order to keep code changes clean and easily viewable.

XI. Issues

Throughout the process of deploying our local application on to Heroku we were able to smoothly deploy most of the application in a functional capacity except for the Google Authentication part which was having issues allowing the Net ID/login info to be input and therefore in order to remedy the situation, a certificate was added allowing Heroku to be safely used by Google Authentication.

Another major issue run into was the induction step checking part of the algorithm as it was deemed extremely substantial to implement due to the requirement of a parser or keyword checker that could intuitively compare the user's input with the solution. That being said, in order to remedy this issue we just had the user directly input their basis and inductive hypothesis. However, to check the inductive step and to check different types of problems would be a significant undertaking that would not be manageable in the given time frame.

When implementing the behavioral tests using cucumber/capybara, we were unable to test our front end due to technical issues with the framework. Capybara was unable to find our html markup or javascript variables, meaning that it was unable to test our features. We were able to write .feature files for cucumber, but without the accompanying step definitions.

XII. Implementation Environment

The implementation environment of the project is UNIX. The releases of our product include the test release, the demo to client, and the final release. The product is released to Heroku app. The initial release date was on January 9th, 2018. The last release date was on January 28th, 2018. We used Slack and Google Drive as our communication channel, developed codes using different online databases, and contributed to the development of project through Github, personal programming, and pair programming.

XIII. Tools and GEMS

The following packages and tools were used in this project:

- Ruby 2.3.1
- Rails 4.2
- RSpec-Rails
- Heroku
- Google Authentication
- Materialize
- Bash
- GitHub
- Trello

In order to ensure that tasks were delegated and clarified, the software application known as Trello was used in order to assign tasks and complete them efficiently. A separate branch for our induction project was created when designing and implementing this idea due to the master branch consisting of the combination of both induction and propositional logic.

The following are additional unique GEMS that were used:

- Omniauth-google-oauth2 0.2.1
- JQuery-rails
- Tzinfo-data
- Sqlite3
- Cucumber-rails

- capybara

XIV. Github Repository

GitHub repository: <https://github.com/victoriaaa234/discrete-math-suite>

XV. Other Important Links

Trello Board: <https://trello.com/b/dw9glE9X/induction>

Heroku App: <http://arcane-sierra-40976.herokuapp.com/induction>

XVI. References

1. Jeff Sutherland, Ken Schwaber. “Chapter 1: The Crisis in Software: The Wrong Process Produces the Wrong Results”. Software in 30 Days. May, 2012.
2. Dror Feitelson, Eitan Frachtenberg, Kent Beck. “Development and Deployment at Facebook”. Facebook.
3. Joel Spolsky. “The Joel Test: 12 Steps to Better Code”. Joel on Software. August, 2000.
4. Fergus Henderson. “Software Engineering at Google”. Google. January, 2017.
5. Adam Pahlevi Baihaqi. “13 Scaling-related Lessons I Learned as a Software Engineer”. Techinasia. December, 2016.
6. Thejasree Prakash. “Test Driven Development vs. Behavior Driven Development”. Glow Touch Technologies. Web. January, 2018.
7. Agile Testing Framework. “Behavior Driven Development”. Web. January, 2018.