The Princeton Computer Science LabTAs and How an Academic Can Learn to Code

**Alexander Daifotis, B.S.E. Computer Science Class of 2014**

**Advisor: Brian Kernighan, Professor of Computer Science**

**COS 497, Fall 2013 – Senior Independent Work**

Abstract: In this report, we discuss the goals, accomplishments, and conclusions of the research conducted this semester in the fulfillment of COS 497 and departmental requirements. This independent work consisted primarily of designing, building, and maintaining a newly developed online queuing system for use during the introductory computer science tutoring sessions sponsored by the department, in which undergraduate TAs are available to assist with the students’ coursework. In addition to presenting the reader with the design and use of this system, we will also discuss the lessons learned from its development that are relevant to an academic who has little experience with real world web development. We also relate the pedagogical insights derived from statistically analyzing the data generated by the tool.

Table of Contents

Introduction 3

Selection of Web Stack and Platform 4

Design 4

Backend 4

Frontend 4

Observed Usage 4

The Problem Solving Mindset 4

The Mentality of Agile Design 4

When is Technology *Not* the Solution? 4

Building Something That Will Last 5

Data Analysis 5

Conclusion 5

# Introduction

Like many of its peer institutions, Princeton offers various services for academic tutoring outside of class in many introductory courses. For the Introductory Computer Science Sequence (IntroCS), the department runs tutoring sessions 7 days a week that are organized and staffed by undergraduates. This sequence consists of three classes that are meant for majors, and one that is meant for non-majors and those looking for a broader and gentler introduction to the field. The three major courses are COS 126 (Introduction), COS 226 (Data Structures & Algorithms), and COS 217 (Systems Programming). The non-major course is COS 109 (Computer In Our World). Henceforth, we will refer to the classes by their course codes rather than the colloquially naming for the purpose of consistency.

As the chart shows, enrollment over the last four years as accelerated at a staggering pace.[[1]](#footnote-1) COS 126 in particular has been a major driver of these increases, though even the 200 level classes have seen enrollments surge. Along with this increase have of course come increasing demands upon the undergraduate tutors, known in the department as LabTAs, to provide assistance.

As the manager of this program over the last two semesters, I can tell you from first hand experience that it is a bit of a handful. The Head TA is responsible for hiring, coordinating with instructors, staffing the shifts, and many other administrative tasks. And, being the engineering student that I am, I have spent a considerable amount of time thinking about ways of improving and simplifying the way the program is run. Some of these have been simple improvements—putting the schedule in a GoogleDoc instead of writing the HTML for a website, surveying more often, consolidating basic information for the TAs, etc. The main portion of this independent work, however, was a project a bit grander in scale.

One thing that has always been a particularly annoying is the fact that I cannot be in two places at once. The reason this frustrates me is that I cannot monitor what happens in the lab in any more than a semi-regular basis, since that would entail me being there every night of the week. What occurred to me, was that like some other schools, a way of getting the next best thing would be replacing the ad hoc system of asking for help in the lab with an electronic help queue. Historically, if the lab was not busy, a student would just flag down a free TA for assistance. If a lot of people started asking for help, somebody would simply start a list of names on the blackboard that TAs would check off as they were attended to. This is a perfectly functional solution to the problem, but it obviously does nothing to help with the problem of actually gaining insight into what is occurring in the lab. Furthermore, from a UI point of view, a webpage is a bit nicer of a solution.

Thus, for this independent research, I have implemented just such a service. In the rest of this report, I will begin by describing in more detail some of the motivations and fringe benefits of undertaking this project. Then, I will discuss the process of designing, building, and iterating on the project. After this exposition, I will move onto a discussing the lessons learned from both the process of building a design and from the data collected over the semester. Finally, I will end with a brief conclusion and ideas for further work.

# Motivations

Blah blah learn how to build

# Selection of Web Stack and Platform

Soifjwojf

Oiwjfwef

# Design

Wef

## Backend

Awefwf

## Frontend

Awefwaf

Wefwef

# Observed Usage

Maybe this is more relevant to discuss with Data Analysis? But this is more of a UI and feedback perspective I suppose. Might not be enough here to make it a section.

# The Problem Solving Mindset

Wefwefwefwef

## The Mentality of Agile Design

Be fast, be simple, avoid complicated things, focus on the goal

## When is Technology *Not* the Solution?

Understand the costs of technology are great, time, maintenance, learning, etc. You have to ask, is this getting you close to the goal?

## Building Something That Will Last

Difference between a hack and a project, not a one-time thing. How to ensure things continue being used that are meant for posterity. What are important considerations?

# Data Analysis

Many subsections here, break out further when we come to it

# Conclusion

Blah blah awesome blah

1. Data for the Spring 2014 semester is as of 12/18/2013. Final numbers were not attainable at the time of submission. Note they are likely inflated since students typically drop. [↑](#footnote-ref-1)