

The Design of an Educationally Beneficial Immediate  
Feedback System

by

Justin Carvalho

A Thesis  
presented to  
The University of Guelph

In partial fulfilment of requirements  
for the degree of  
Master of Science  
in  
Computer Science

Guelph, Ontario, Canada

© Justin Carvalho, July, 2017

## ABSTRACT

### THE DESIGN OF AN EDUCATIONALLY BENEFICIAL IMMEDIATE FEEDBACK SYSTEM

**Justin Carvalho**  
**University of Guelph, 2017**

**Advisor:**  
**Dr. Judi McCuaig**

Student learning is positively affected by the quality, and timeliness of feedback they receive on their assignments and homework. Unfortunately, as class sizes increase, instructors find it increasingly difficult to produce meaningful feedback for students in a timely fashion. To maintain an environment in which a student can continue to receive beneficial feedback, an alternate scalable method of providing feedback needs to be established. This document presents the design of a system capable of providing automated feedback to students on their work immediately after it is requested. A prototype of the design that produces feedback on C programs was developed, and provided to computer science students. An analysis of the usage of the system, and the student work submitted to it shows the design can produce feedback that students find valuable, improves the learning experience, and improves programming ability. The positive results suggest that further research should be conducted.

## Acknowledgements

I'd like to acknowledge the following people for their support in the completion of this thesis:

Dr. Judi McCuaig

Dr. Daniel Gillis

George Carvalho

Rebecca Logan

Financial support was provided by the Ontario Graduate Scholarship (OGS) program, and the College of Physical and Engineering Sciences (CPES) of the University of Guelph.

# Contents

|   |            |
|---|------------|
| <b>List of Figures</b>                                    | <b>vi</b>  |
| <b>List of Tables</b>                                     | <b>vii</b> |
| <b>1 Learning, Feedback and Automation</b>                | <b>1</b>   |
| <b>2 Literature Review</b>                                | <b>4</b>   |
| 2.1 Measuring Changes In Learning Environments . . . . .  | 5          |
| 2.1.1 Satisfaction . . . . .                              | 5          |
| 2.1.2 Engagement . . . . .                                | 6          |
| 2.1.3 Self-efficacy . . . . .                             | 6          |
| 2.2 Feedback . . . . .                                    | 8          |
| 2.3 Automated Feedback . . . . .                          | 9          |
| 2.3.1 Student Work Products . . . . .                     | 10         |
| 2.3.2 The Parsing Process . . . . .                       | 10         |
| 2.4 Scalable Web Applications . . . . .                   | 12         |
| <b>3 System Architecture</b>                              | <b>15</b>  |
| 3.1 Architectural Requirements . . . . .                  | 15         |
| 3.2 Overview . . . . .                                    | 16         |
| 3.3 Interface . . . . .                                   | 17         |
| 3.4 Processor . . . . .                                   | 19         |
| 3.4.1 System API Module . . . . .                         | 19         |
| 3.4.2 Processing Module . . . . .                         | 22         |
| 3.5 Service Container . . . . .                           | 22         |
| 3.5.1 Service Container Configuration File . . . . .      | 23         |
| <b>4 Implementation and Results</b>                       | <b>26</b>  |
| 4.1 Implemented System Prototype . . . . .                | 26         |
| 4.2 Experimental Design . . . . .                         | 29         |
| 4.2.1 Data Collected From IFS . . . . .                   | 30         |
| 4.2.2 Data Generated By Code Analysers . . . . .          | 30         |
| 4.2.3 Data Collected Via Self-Assessment Survey . . . . . | 32         |
| 4.3 Results and Discussion . . . . .                      | 33         |

|          |   |           |
|----------|---|-----------|
| 4.3.1    | Does the designed system reliably produce feedback for students? . . . . .  | 36        |
| 4.3.2    | Does the designed system produce feedback students find valuable? . . . . . | 36        |
| 4.3.3    | Does the designed system improve the student learning experience? . . . . . | 36        |
| 4.3.4    | Does the designed system improve students' programming ability? . . . . .   | 37        |
| 4.3.5    | Summary of Results . . . . .  | 39        |
| <b>5</b> | <b>Limitations, Future Work and Conclusions</b>                             | <b>41</b> |
| 5.1      | Limitations . . . . .   | 41        |
| 5.2      | Future Work . . . . .   | 43        |
| 5.3      | Conclusion . . . . .  | 44        |
|          | <b>Bibliography</b>   | <b>45</b> |
| <b>A</b> | <b>Participant Survey</b>   | <b>50</b> |
| <b>B</b> | <b>Ethics Documents</b>   | <b>52</b> |
| B.1      | Ethics Application for IFS . . . . .  | 52        |
| B.2      | Data Privacy Plan . . . . .   | 62        |
| B.3      | Secondary Data Use Application . . . . .                                    | 65        |

## List of Figures

|     |  |    |
|-----|--|----|
| 2.1 | Self-Efficacy and Outcome Observations . . . . .         | 7  |
| 3.1 | System Architecture Overview . . . . .                   | 16 |
| 3.2 | Interface Wireframe . . . . .                            | 18 |
| 4.1 | IFS Login Page . . . . .                                 | 27 |
| 4.2 | IFS Upload Page . . . . .                                | 27 |
| 4.3 | IFS Upload Page: Configuration . . . . .                 | 28 |
| 4.4 | IFS Feedback Page . . . . .                              | 29 |
| 4.5 | IFS More Info Modal . . . . .                            | 30 |
| 4.6 | System Architecture Overview . . . . .                   | 35 |
| 4.7 | System Architecture Overview . . . . .                   | 39 |
| 4.8 | Median Errors Per Report By Number of IFS Uses . . . . . | 40 |

## List of Tables

|     |   |    |
|-----|---|----|
| 4.1 | IFS File Upload Data . . . . .                                      | 31 |
| 4.2 | IFS Feedback Item Data . . . . .                                    | 31 |
| 4.3 | IFS Feedback Item Interaction Data . . . . .                        | 31 |
| 4.4 | Post Analysis Data . . . . .  | 32 |
| 4.5 | Summary of Submissions . . . . .                                    | 34 |
| 4.6 | Top 20 Errors and Warnings . . . . .                                | 35 |
| 4.7 | P-Values Measuring Significant Difference In Usage Groups . . . . . | 38 |

## Chapter 1

# Learning, Feedback and Automation

Student learning is positively affected by the quality, and timeliness of feedback they receive on the by-products of their education. In the traditional educational setting, this feedback is most often in the form of a grade and verbal or written comment on the student's homework, assignments, and exams. In today's higher educational environments, instructional staff are finding it increasingly difficult to produce meaningful feedback in a timely fashion because of growing class sizes. To maintain an environment in which a student can continue to receive satisfactory feedback, alternative methods of providing it need to be considered. Research has shown that automated feedback can maintain, and even increase, the positive effect of feedback provided by instructional staff [1, 2, 3, 4, 5]. Computer Science is a learning domain which has seen a dramatically increasing number of students recently, and is a suitable candidate for an automated feedback solution.

A commonly occurring problem faced by Computer Science students is the feeling of helplessness, and ignorance of how to proceed when an error in their program code is encountered. While an experienced developer can quickly identify likely causes and potential solutions to a programming error by observing unexpected runtime behaviour or finding a compiler error, inexperienced developers often find the situation overwhelming and confusing. As a result, students are often reliant on feedback from the instructional team in helping them understand convoluted errors. These students could benefit from an immediate feedback system capable of automatically detecting, and providing feedback that explains mistakes in their code.

This paper presents the design and testing of a system capable of providing on-demand feedback to students. The design presents an extendible framework that can be used within any educational domain, provided the by-products of the students' education can be understood by a computer. To test the value of



the design a prototype was constructed with the ability to analyse C language programs.

**Thesis statement:** It is hypothesised that a scalable web application can be created to provide immediate feedback on a student's machine-parsable work product in a way that is beneficial to the student's learning experience.

Within this thesis statement, are several key points that deserve attention. The scalability and reliability of the system is important, particularly with growing enrolments and the emergence of Massively Open Online Courses. Without scalability, the system will eventually suffer the same difficulties instructors currently face with providing feedback. The system should be able to handle large computational loads, and provide administrators with an easy way to empower it with additional resources if the need arises. For the system to be adopted, it needs to maintain a greater time and cost efficiency than adding an additional instructor. Scalability, and reliability should be factored into the design of the system.

Feedback provided by the system needs to be immediate, supplying the student with a greater educational benefit than if they had to wait for delayed feedback, or received no feedback. For the system to generate context-sensitive and meaningful feedback that goes beyond generalities, the input needs to be restricted to items that are machine-parsable.

The system needs to provide a measurable benefit to the student's learning experience. It should be easy to use, provide value, and help them find and overcome errors in their programs. Most importantly, it should enable them to improve their programming abilities.

To analyse the hypothesis, the following four research questions will be discussed.

1. Does the designed system reliably produce feedback for students?
2. Does the designed system produce feedback students find valuable?
3. Does the designed system improve the student learning experience?
4. Does the designed system improve students' programming ability?

Chapter two of this document presents a review of relevant literature on how changes in the learning environment can be measured, the impact of feedback in education, the success of automated feedback and how it can be generated, and the attributes of a scalable system. Using the knowledge gained from the literature survey, chapter three describes the design and architecture of one possible immediate feedback system. Chapter four describes an implemented prototype, and an experiment that made use of it. Data collected from this experiment is analysed, and leveraged to answer the above research questions.

Chapter five summarizes the research, discusses the limitations of the study, and explains how future work can further explore the topic.

## Chapter 2

# Literature Review

The design and implementation of a scalable web application that provides immediate feedback on a student's machine-parsable work product in a way that is beneficial to the student's learning experience requires an understanding of:

1. The factors that foster a positive learning experience for a student, the metrics by which we can measure that experience, and the changes in those metrics that represent an improvement. With this understanding, the design of the system can focus on increasing the accessibility of a positive learning experience and an experimental prototype can be built to gather the data appropriate for building metrics that describe the impact the system made on learning experiences.
2. How feedback is used in education, and the context in which immediate feedback is most beneficial to learners. Using this information, the system can be designed so that it integrates with the appropriate learning activities, and produces effective feedback, equipping the student with all the benefits possible from an automated feedback system.
3. Techniques that can be leveraged to enable the automated generation of immediate feedback, and the subset of work products that they apply to. A working knowledge of these techniques, will allow the design of a system that leverages them.
4. Identify the characteristics of a scalable web application. The proposed architecture will need to provide a consistent, and reliable experience that enables students to receive feedback immediately, regardless of the number of concurrent users or the size of their work products.

This chapter presents a survey of relevant literature to situate the proposed system in the current state of research and development practises, beginning with a discussion on the methods of measuring changes in a learning environment.

## 2.1 Measuring Changes In Learning Environments

Researchers have identified many methods for measuring the effect of introducing new educational tools to the classroom. Lux et al. compared grades, and analysed satisfaction surveys when they introduced new technology to their introductory college chemistry class [6]. Cole and Todd measured student attitude during their experiment, and Kaila et al. considered engagement, and motivation [5, 2]. Laakso et al. and Kurvinen et al. both compared the results of similar tests taken before and after the introduction of their changes [3, 4]. A common method for measuring the effect of new educational tools is to measure student satisfaction with the tool, student engagement with their related courses, and their self-efficacy levels. The tool can be considered a positive addition to the learning environment if students are a) satisfied with the tool, b) increase or maintain their engagement with the related courses, and c) improve or maintain their self-efficacy levels. This section will explore the current research that measures these three characteristics of students.

### 2.1.1 Satisfaction

Student satisfaction is a gauge of a student's subjective evaluation of the activities, outcomes, and experiences related to their education [7]. Studies have shown a strong positive correlation to student satisfaction and retention, motivation, and success [8, 9, 10, 11, 12]. Astin boldly concludes "it is difficult to argue that student satisfaction can be legitimately subordinated to any other educational outcome" when he compares it to student attitudes, beliefs, self-concepts, behaviour, academic achievement, and career path. Given the proven effect of satisfaction on education, it is a common measurement used in pedagogical studies.

Traditionally, researchers have measured students' satisfaction through surveys using a simple binary response of *yes, I am satisfied* or *no, I am not satisfied* response to the question *are you satisfied with your overall educational experience?* While these responses are easy to collect, and simple to analyse, they often result in specific educational experiences and activities being incorrectly classified as satisfactory or unsatisfactory because of the weight of the others. Furthermore, responders may not thoroughly reflect upon all of their experiences without the survey prompting them [14]. To improve the recognition of the quality of the components that make up a student's satisfaction, various researchers have developed alternative methods that better measure the satisfaction of specific elements of the student's educational experience.

Gazza and Matthias had a satisfaction survey that included 36 Likert-type items to measure students' satisfaction with questions that specifically related to all the small components that make up their new online accelerated nursing education program [15]. Using the results of the survey, the researcher

discovered that students were not satisfied with the clarity of program requirements, assignments, responsiveness of requests for information, the amount of work required to succeed, or the availability of technical assistance. With this awareness, faculty of the school were able to develop an interdepartmental program for improving student satisfaction. Similarly, Douglas et al. made use of a questionnaire with 60 questions about specific educational elements the participants interact with [16]. The questions were written to provide quantitative results using a Likert scale. The results of the study informed the school that satisfaction rates for teaching and learning support materials across different instructors varied drastically. With this information, the school was able to provide training to instructors who needed it.

### **2.1.2 Engagement**

Student engagement is a measurement of the time and energy students invest in educationally purposeful activities [17]. It is often used as a key element in the prediction of a student's success, based on research that shows a positive co-relation between high engagement levels and high degrees of student success [18, 19]. In synthesis of research performed by Pascarella, a positive co-relation between engagement and students' educational aspirations, attitude towards college, academic achievements, intellectual and personal development, and institutional persistence was found [20]. As a result of the relations between engagement and positive learning attributes, it can be used to effectively audit improvements in the learning environment when it is measured accurately.

In an experiment performed by Hughes and Pace, responses from the National Survey of Student Engagement (NSSE) were taken and compared to withdrawal rates [18]. The NSSE was sent to random samples of freshmen and seniors, and questioned the responders using a Likert scale regarding their educational activities, participation, social interactions, and satisfaction. An analysis of the results showed students who leave college prematurely have lower engagement levels than those who remain enrolled. The NSSE was also used in a study performed by Kuh et al., where two key findings were made [21]. In their study on engagement in first year college students, results showed: 1) student engagement in educationally purposeful activities is positively related to academic outcomes, and 2) student engagement has a positive compensatory effect on first-year grades and persistence to second year, where an increased level of engagement correlated to higher first-year grades, and continuation of high achievement in second-year.

### **2.1.3 Self-efficacy**

Self-efficacy is the personal judgement of one's capabilities to organize and execute courses of action to attain designated goals [22, 23]. The level of self-efficacy is a measure of dependence on the difficulty of a specific task, the generality of self-efficacy refers to the transferability across different types of tasks, and the strength of self-efficacy measures the level of confidence a



Figure 2.1: Visual depiction of the influence self-efficacy has on outcome observation

person has about performing a specific task [24]. In a study performed by Linnenbrink and Pintrich, self-efficacy was found to be one of the most important factors that influence student engagement. They concluded that fostering strong self-efficacy beliefs is a pathway for all teachers to experience success in their classrooms, likely as a result of willing participation.

The human mind understands concepts much more effectively by observing the differential effects of our actions, rather than from examples provided by others. Furthermore, it remembers processes in greater detail when it is an active participant [26]. The teaching methodology, Action Learning, leverages this behaviour trait, and encourages one to perform an action, observe the results, and reflect [27]. When a subject finds the results of their actions to be beneficial, it is naturally understood that the same action should be repeatedly performed to receive similarly beneficial outcomes [28]. Otherwise, the action should be adjusted in a way that makes the results more preferable. Unfortunately, for this action learning to be effective, the person must be an active participant so that outcomes can be observed, and response expectations can be built. The willingness to participate is usually a result of one's self-efficacy. Bandura describes a self-efficacy to outcome model in their paper, shown in figure 2.1. In this model, self-efficacy directly affects the actions a person takes, and therefore their expectations and outcomes. This has a dramatic effect on the persons ability to reflect and learn from those outcomes. If a low self-efficacy debilitates behaviour, one may never be able to observe the outcome of their (non-existent) actions. To ensure changes in the classroom don't create unintended barriers to action learning by decreasing self-efficacy, researchers have established methods to measure it.

Sherer et al. used a modified version of the self-efficacy scale to measure the self-efficacy of their participants [29]. The self-efficacy scale is a survey composed of two subscales: the general self-efficacy subscale and the social self-efficacy subscale. Both are 10-item surveys, and task respondents with rating their agreement or disagreement with a statement on a symmetric numerical range. While the general self-efficacy subscale focuses on measuring the responders confidence in overcoming everyday tasks, the social self-efficacy

subscale focuses on their confidence in overcoming challenges in social situations. Using this scale, the researcher found a positive correlation between self-efficacy strength and educational success in their participants. In a survey, Zimmerman and Bandura recommends using the self-efficacy scale as a method for measurement as well [30]. Chemers et al. made use of the self-efficacy scale, and combined it with their own questionnaire for their study, inquiring about the students optimism, level of pressure and demand experienced, academic self-rating and expectations [31]. With the resulting measurement of self-efficacy, an analysis was performed that discovered self-efficacy and optimism were strongly, and positively, related to performance and adjustment, stress reduction, health, and overall satisfaction and commitment to remain in school.

A student's self-efficacy, engagement, and satisfaction are all indicators of the student's learning environment quality. Measuring, and tracking changes in these three metrics is an effective method of gauging whether the addition of an immediate feedback tool is beneficial to the student.

## 2.2 Feedback

Feedback is often considered to be one of the most powerful influences on learning [32]. Winne and Butler define feedback as “information with which a learner can confirm, add to, overwrite, tune, or restructure information in memory, whether that information is domain knowledge, meta-cognitive knowledge, beliefs about self and tasks, or cognitive tactics and strategies” [33]. There are two main categories of feedback: descriptive feedback, and evaluative feedback. Evaluative feedback, often in the form of grades and percentages, is an overall summary of how well or poorly a student completed a task. This feedback primarily serves to reinforce positive behaviour, and provide a measurement of the success of the student. Descriptive feedback takes a more thorough approach, providing the student with targeted descriptions and comments on what the student completed, and what they can do to improve [34].

A synthesis of over 500 meta-analyses, involving approximately 20-30 million students, found that feedback was the most powerful single factor that enhanced achievement [35]. The researcher states the most effective method of improving education in increasing the amount of feedback a student receives. Descriptive feedback is the most effective form, because it explains how and why the student understands or misunderstands a concept, and what actions they need to take to improve. While quality, and accuracy of feedback are known to be important, the timing of a student's feedback affects the student is still an open research question that studies are trying to answer. Some researchers argue immediate feedback is most important, as it enables a student to enter a flow state.

Flow is the mental state in which a person performing an activity is fully immersed in a feeling of energized focus (often referred to as “being in the zone”) [36]. Researchers have found that entering and maintaining a flow state has a positive effect on learning and knowledge retention [37, 38, 39, 40]. To encourage a person to enter a flow state, an activity should fulfil 3 conditions [41]:

1. The activity should have a clear set of goals, and a method of measuring progress towards those goals.
2. The activity should have clear and immediate feedback, enabling the person to adjust their actions and optimize performance.
3. The activity should be challenging, without seeming impossible. The participant must have confidence in their ability to complete the task at hand, without finding the activity mundane or boring.

While instructors have found providing effective feedback in education difficult and time consuming [42, 43], researchers found delayed feedback to be much less effective than immediate feedback [44, 32, 45]. As an alternative, methods of providing immediate feedback have been investigated to improve student engagement and performance with a focus on automatically generating feedback for students using automated computer systems. These methods of automated generation have shown to be an effective solution that frees up instructor time, and empowers students to get feedback on demand, despite the increasing ratio of students to instructors in modern classrooms.

## 2.3 Automated Feedback

Automated feedback is feedback that is created without the need for the instructional team to review the student’s work. Traditionally, automated feedback has been used to generate evaluative feedback, as it is relatively simple to check if the student’s result is correct or not. Math workbooks commonly provide this by listing the answers to questions, enabling students to check if their work was done correctly. Unfortunately, evaluative feedback is not effective with inexperienced learners, because they are unaware of where they went wrong [46]. With improvements in technology, researchers are investigating ways in which computer systems can provide descriptive feedback.

Kuikka et al. used VILLE, a computer program designed to validate student’s answers and provide immediate feedback during assessments, to teach students at the Turku University Mathematics and Business and found that grades of the users significantly increase [1]. Kaila et al. and Laakso et al. found similar results with their Computer Science courses [2, 3], and Kurvinen et al. found similar results in their primary Math course [4]. Cole and Todd have found a positive outcome in student attitudes towards General Chemistry when they provided their students with a series of web-based homework and tutorials that contained immediate feedback [5]. These automated systems all



provided formative feedback, a subset of descriptive feedback, that aims to help students identify their strengths and weaknesses, by evaluating student work products while they are working on them. This remainder of this section will define student work products, and identify the tools and processes needed to automatically produce feedback.

### 2.3.1 Student Work Products

A work product is a term for anything resulting from a process. In the context of education, a student work product is an artifact of a student's learning. These work products, usually in the form of assignments, tests, and homework questions, are used to both assess a student's understanding of the subject material and provide them with feedback. As established in section 2.2, receiving appropriate feedback while a student is learning is one of the key factors in their success. Although there has been work towards implementing systems capable of providing immediate feedback on *some* work products, to ensure students have access to consistent, immediate, and thoroughly explained feedback on *all* of their work products, instructors need to establish processes that encourage the production of machine parsable student work products or ones that can be easily translated into a format parsable by a machine.

Data that is machine parsable, or machine readable, is capable of being understood by a computer [47]. This understanding typically occurs in two stages: scanning, and parsing. In the past, the set of data capable of being scanned and parsed has been mostly restricted to written work that can be expressed with unambiguous symbols and tokens, such as mathematical expressions, highly structured data listings, or programming languages [48]. Today, with the increase in computing power and further understanding of language and computer algorithms, we've been able to expand the set to include more natural languages (human-like languages) [49, 50], visual representations (images, drawings, and handwriting) [51, 52], and sound [53].

### 2.3.2 The Parsing Process

Traditional algorithms for parsing non-trivial data into a format usable by computers take three distinct steps: a) scanning, b) parsing, and c) semantic analysis [54].

**a) Scanning:** The first step involves transforming the source data (input) into lexical tokens. A lexical token (referred to as token) is a structured representation of a sequence of characters taken from the input that explicitly states its categorization [54]. For example, the word *run* can be categorized as a present tense verb. The implementation of this step varies dramatically based on the format of the input, and could involve methods as simple as a finite-state machine or as complex as a neural network. The only restriction

is that the source material is taken, and reduced to ordered computer readable tokens (often represented with integer enumerations) with an acceptable accuracy. If a portion of input cannot be matched to a token, it is rejected as invalid input.

**b) Parsing:** After the input is broken down into a sequence of well-defined tokens, parsing can begin. The goal of this phase is to establish the syntactical correctness of the input. This is done to ensure that meaning can be found in the source material. The typical implementation checks this by comparing the order of the tokens with a defined language specification. If the order of the tokens does not fit within the language, it is rejected as meaningless input. If the input is considered meaningful then it is transformed into an abstract syntax more appropriate, and efficient, for processing, usually a tree or tree-derived data structure.

For example, if I wanted to create a program for listening to and recording simple voice logs, I could define my input language to accept sentences in one of the two formats:

- Date Subject Verb
- Date Subject Verb Subject

In this language, we have 3 unique tokens: Date, Subject, and Verb. When a scanner is run on our input, it would produce the date token when it hears two numbers, subject token when it hears a noun, and verb token when it hears an action. If I spoke the words “three twenty-five Frodo destroys ring”, our scanner would accept the input, and send “Time Subject Verb Subject” to our parser. The parser would match the ordered tokens to the language specification, and provide the input to the semantic analyser.

**c) Semantic Analysis** The final step of the process, similar to the the first, tends to differ based on the task that is being accomplished. In this stage, the abstract syntax is taken and the meaning of the input is established. The entire data structure is iterated over, and a specific action is taken based on each token. At this stage, the original value of the token is considered. Some programs, e.g. compilers, will continue with additional analyses. However, for most applications, the input has been decomposed enough for processing.

It should be noted that some parsing processes have deviated from this 3 step process. In the context of generating feedback on student work products, the majority of road blocks occur within the scanning stage because student work products tend to be hand-written notes in their native language, or drawings. To alleviate the difficulty of writing a scanner for those products, and defining a language capable of understanding every students minute difference in notation and language, some researchers are bypassing scanning entirely and having their students interact directly with the program while learning. This

approach has two significant benefits: a) the input is automatically tokenized with complete accuracy, and b) restrictions can be placed on the input ensuring it always has meaning, and removing the need for a language specification.

Rafferty et al. implemented this alternative process and had their chemistry students draw the arrangement of atoms before and after a chemical reaction within a computer program, dubbed WISE, that provided them with immediate, formative guidance as feedback [55]. Their results show that providing immediate automated feedback is equally as effective as teacher-generated feedback. Papastergiou built a digital game for teaching computer science students with immediate feedback inherently built-in, and found that it was more effective and more motivational than the traditional approach [56]. Corbett and Anderson used an computer-based problem solving environment, where students wrote small programs to answer questions. The control group that received immediate feedback, yielded the most efficient learning [57].

To provide consistent, immediate feedback to a large number of students, automated systems need to be scalable. The next section will define scalability, and identify the properties of a scalable web application.

## 2.4 Scalable Web Applications

An application’s scalability is a measure of its ability to complete a growing amount of work, as well as its ability to expand in a manner that accommodates a growth in work [58]. For example, if we observe that an application is capable of completing a maximum number of calculations per second, and we later observe an increase in the maximum number of calculations per second when additional resources are given to the application, that application has a high scalability on its resources. This type of scalability is referred to *load scalability* [58], and is one of the most important elements of web applications. Web applications that are not scalable tend to perform poorly when under heavy use, resulting in slow load times, server-client disconnects, and data anomalies.

Features of a scalable application need to be identified prior to the development of any web application. The design and implementation of the system, will need to incorporate and exploit each feature to optimize the system for growth. Based on the examination of existing systems, Tewari enumerates 7 design principles crucial to the implementation of a scalable web application: Decentralization, Caching, Offloaded Data and Computation, Hot-Spot Reduction, Minimization of Global Information, Dynamic Configuration and Availability Guarantee, and Trust Minimization.

**Decentralization** involves distributing the functions of a system among smaller distinct subsystems that exchange information by passing messages between each other. While a centralized design has a single system complete

a job, a decentralized design divides that job into smaller tasks and assigns each one to a different subsystem. The subsystems interact with each other by passing information through messages, and combine the results of each subtask completing the job. [60] Due to resource limitation, a centralized component will always have a finite growth potential and have a limit to the number of jobs, or size of job, it can complete at any given time. To achieve scalability, a system must be designed with the minimum number centralized components possible [59].

**Caching** is a well understood, and practised method for handling exponential growth in a user base by reducing the computation load on servers. A cache is a perishable store of requests, or calculations, that can be reused by a program in the event a similar request is received from a user. There are three key types of cache related to scalability: Shared Server Side Cache, Unshared Server Side Cache, and Client Cache. Server side cache is stored with the access point of the web application, and allows for the immediate return of results when requested by a user. It's typically used for expensive computations, and minimizing the need to access additional remote servers or databases. A shared cache is one that can be used by anyone that interacts with the server's access point. It is often implemented in tandem with multiple unshared caches which can only be used for specific users (as a result of the calculation containing user specific data). Client Cache is stored on a users local machine, and helps avoid unnecessary requests over the network to a server [61].

**Offloaded Data and Computation** from the server, to the client, allows for a massive reduction in server load and a decrease in user interface latency (as a result of reduced network access). The system achieves the reduction by storing user data, and performing computations on the data, on the user's machine instead of on the server [59]. This is becoming increasingly common with modern JavaScript frameworks (e.g. Angular, React, Ember) and the increasing popularity of public facing RESTful APIs. However, not all functionality can be offloaded, as any data that is stored or computed on a user's machine may be tampered with, and should never be considered a source of truth. Furthermore, data that is stored on a user's machine cannot be accessed from another machine, and therefore cannot provide users with a synchronized experience.

**Hot-Spot Reduction** involves pinpointing bottlenecks in your system, and reducing their impact. A bottleneck is a component, or communication path, in your system that has the lowest throughput [62]. By way of illustration, consider a highway with four lanes and consistent flowing traffic. If construction were to occur at one point on that road, and block 3 of the 4 lanes, traffic at that point will be slowed considerably. Vehicles that once used four lanes now have to converge and travel through one. While traffic on the rest of the road will continue to move faster than traffic at the construction point, the bottleneck

will eventually cause a queue of cars to build up before it, significantly reducing the throughput and speed of all cars on the road. Software bottlenecks have a similar impact on the throughput of data. To reduce their effect on program flow, the action causing the bottleneck should be performed in parallel with the remaining computations. If the task cannot be completed in parallel, then it should be moved to the end of the process (after the client receives a response), or its processor replicated to increase the throughput [62].

**Minimization of Global Information** is the removal of any global variables, or continuous properties. A global variable is a single storage location for data accessible by the entire system, and is primarily used to configure the behaviour of a system, or to broadcast information. To maintain data integrity, accessing global variables require the use of semaphores, a tool for coordinating reads and writes by limiting access to the variable to one process at a time. Requiring global coordination between  $n$  components becomes exponentially more expensive as a system grows and is therefore a limiting factor in scale [59].

**Dynamic Configuration and Availability Guarantee** are two key factors for ensuring a user has a consistent experience using your system, regardless of the number of other simultaneous users or system components. The former involves a design that enables your system to automatically allocate additional resources for specific components when under a heavy load, or create duplicate components to split the load. The latter typically involves a supervisor that checks the health of every component or service, and instantiates new ones to replace any that are failing. All of this should happen seamlessly behind the scenes, without any impact on a user of the system. The increasing availability of cloud computing has made implementations of this much more cost-effective than it has been in the past [63].

The scalability of a program used for providing immediate automated feedback must be considered through the entire design and implementation of the application. Without high scalability, there is no guarantee that feedback provided to students will be correct, consistent, or immediate, particularly around assignment deadlines when the system is likely to be used by many students.

## Chapter 3

# System Architecture

The purpose of this study is to establish whether a scalable web application can be created to provide immediate feedback on a student's machine-parsable work product in a way that is beneficial to the student's learning experience. In this chapter, a description of an architecture for the web application is presented. From the learner's perspective, this system will allow for the submission of a text based work product to be analysed. Immediately after analysis, the system will return feedback to the student with direct references to their input. The system can be customized with the addition of plugin services and configuration files, allowing it to be adapted for any form of parsable input, and any form of automated feedback including descriptive and evaluative feedback.

The implementation of this on-demand, immediate feedback system can be facilitated through a service-oriented architecture. A service-oriented architecture is a style of software design where functionality is provided as services across its components using a defined communication protocol. This design allows for loosely coupled, and adaptive components that can be duplicated for scalability. The description will begin by detailing the requirements the system must meet, and then describes the architecture and function of each component in the system.

### 3.1 Architectural Requirements

To fulfil the requisites of our thesis, the system architecture must enable scalability, extendibility, and ease of use. The following is a list of detailed requirements that a successful design needs to meet.

1. The system must be able to authenticate a user.
2. The system must provide access to linters configured for all of an authenticated user's classes.
3. The system must restrict access to files and reports to the related user.

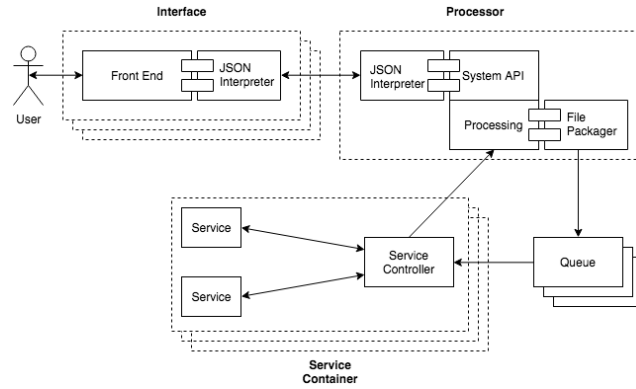


Figure 3.1: System architecture overview broken into three main components: the interface, the processor, and the service container.

4. The system must be able to customize feedback based on a user's preference.
5. The system must be able to handle a significant amount of load, without reducing a users experience.
6. The system must provide feedback in a reasonable amount of time.
7. The system must be easy to configure, ensuring instructors with various levels of expertise can provide their class with access.
8. The system must be extendible, allowing instructors to integrate linters and add their own services without modifying any existing code.
9. The system must resist student tampering, and prevent data theft in the event of a security breach.
10. The system must execute all code analysis in a separate, isolated environment, to prevent the code or any configured linters from gaining access to the system.

In the remaining sections of this chapter, we'll present a design capable of meeting all of the listed requirements, beginning with a general overview of the system.

## 3.2 Overview

The design splits the entire system into 3 main components: the Interface component, the Processor component, and the Service Container component. Each component operates as a black-box, requiring no access to the internals of another component. Interaction between each component occurs through strongly-defined input and outputs, accessed through public facing modules. Figure 3.1 shows the communication paths between all the components.

The user's interaction with the system happens through the front end, which is contained within an interface component. The entire system can be made of multiple interface components, allowing for multiple user interfaces. In addition to the front end, a JSON interpreter that is responsible for sending and receiving HTTP requests is contained within each interface component. The JSON interpreter interacts with the System API contained within the processor component.

The system contains a single processor component, and its responsibilities include flow control, queuing, and report packaging. When a report is requested, the processor inserts the request into a queue that feeds into the service container responsible for fulfilling the request. The system contains an equal number of queues and service containers, one for each configured type of report (typically one for each class).

The service container contains a service controller, that has services injected into it through their providers using dependency injection. This allows controllers to cherry-pick the services they require to generate a report, while excluding the ones they don't need, without any knowledge of their inner-workings.

This design allows for each one of the components to run on their own server, and even allows the interface to be offloaded to the user's computer. This makes the entire system distributable, modular, and highly scalable.

Sections 3.3, 3.4, and 3.5 will detail the features of each component.

### **3.3 Interface**

The interface component is the presentation layer (front end) of IFS, and contains very little business logic. Its most significant responsibility is to collect input from the user, and pass it off to the System API. Performing operations on the input is never done in the interface, and is instead offloaded to the processor. Input is strictly cleaned by the interface, and sent through the system API to the processor. The processor decides what operations need to be performed, and completes the work. When the work is complete, the system API returns the results of the operation to the interface, after which the interface performs its second responsibility: displaying the output to the user. All communications between the interface and the System API occur through a RESTful service, usually across a network. Packets are exchanged in JSON format, which is easily understood by both client-side and server side programming languages.



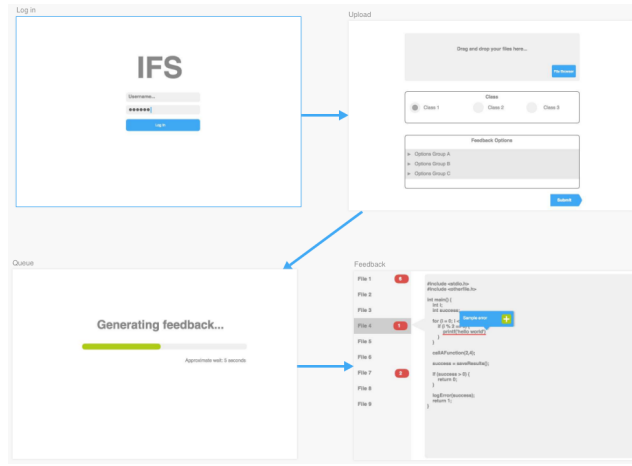


Figure 3.2: A wireframe drawing of one possible interface including 4 distinct pages: A login screen, a file upload and report configuration page, a queueing page, and a page displaying the generated feedback. Arrows describe the typical path through each page.

This design allows for the front end to be interchangeable, and deployed multiple times, establishing a many-to-one relationship between interface components and the processor component. This many-to-one relationship, combined with a remote server performing all operations, provides a architecture supported and inexpensive way to offer different interfaces for different inputs, configurations, courses, and schools, that can be implemented as native GUI clients, mobile applications, IDE plugins, command line programs, or web applications. For complete compatibility with the other components, the interface should provide the following features:

1. System status notifications, including updates and online/offline status.
2. User authentication using a username and password, with the ability to alert the user on both success and failure, and the ability to store a returned access key securely.
3. File uploads, including single files, multiple files, zip and tar archives.
4. Configuration options for generated feedback
5. Report status notifications, including displaying an estimated time until completion
6. Feedback message display, including both short and long form and the ability to vote on whether the feedback was helpful or confusing
7. Send https requests, and recieve https responses.

Figure 3.2 shows a wireframe drawing of one possible interface that meets the minimum requirements for complete compatability. Arrows are drawn depicting the traversal of a typical use case. Within the wireframe are 4 distinct

pages: A login screen, a file upload and report configuration page, a queueing page, and a page displaying the generated feedback. Features 1 and 2 are handled by the login page. Two textfields are shown, accepting the username and password. When the user attempts to log in, an API request with the contents of the fields is sent to the API and the returned access token is stored encrypted in local storage if authentication was successful, or an alert prompting the user to try again on failure. In situations where the system is found to be offline, discovered by the result of an API call, the fields can be replaced with a notice alerting the user that the system is currently unavailable. Features 3 and 4 are handled by the upload page, where a file, service selection, and linter configuration box is shown. The service selection allows the user to select which course they'd like to upload files for (which corresponds to service containers internally), and their associated configuration options in the box below. Feature 5 is handled by a simple progress bar page, which is automatically shown after the user submits their report request. Along with an easy to read bar in the middle of the page, an estimated time of completion is shown below. Feature 6 is handled by the feedback page, which shows a list of all the files and the number of feedback items belonging to that file on left navigation bar, along with the contents of the currently selected file on the right. The contents of the file are shown with red underline on portions that are associated with feedback. Hovering the mouse over the underlined text, will display a tooltip with the feedback, as well as an expand icon to see more information or to vote on the helpfulness of the feedback message.

### 3.4 Processor

The processor module's responsibilities include handling and replying to interface requests, supervising and scheduling reports, packaging files for their related service containers, and parsing service container responses so they can be returned to the interface. These responsibility are split between three key modules: the System API Module, the Processing Module, and the File Packager Module.

#### 3.4.1 System API Module

The System API Module is directly responsible for handling user authentication, interface data collection, and file uploads. All other requests are passed to the Processing Module once their input has been validated.

The System API Module provides a HTTP RESTful API to the systems interface. A RESTful API uses the representational state transfer (REST) communication protocol, to transfer data between clients and servers. It categorizes all communications into 5 verbs: *POST* (used when the client wants to create data on the server), *GET* (used when the client wants to read data from the server), *PUT* (used when the client wants to create or replace data on the

server), *PATCH* (used when the client wants to update or modify data on the server), and *DELETE* (used when the client wishes to remove data from the server). Along with these verbs, a RESTful API typically divides each of its possible actions into collections, which translates to a sub-path in the target URL. These actions are referred to as the collection's endpoints.

The IFS System API Module has five collections: *status*, *auth*, *file*, *report*, and *feedback*. The *status* collection is used to check the availability of the processor and service container components, as well as retrieve additional information on error messages. The *auth* collection is used to provide the interface with methods for user authentication, and a way to create service tokens. The *file* collection is used to upload and retrieve files that are used in reports. The *report* collection is used to submit report requests, check the status on reports, and retrieve the results of a report. The *feedback* collection is used to provide the system with data about how the report was used, and rate the feedback provided.

For complete functionality with the other components, the following endpoints should be implemented in the API:

- **Endpoint URL:** `api/`  
**Method:** GET  
**Input:** None  
**Returns:** An array containing the url of each endpoint, their input, and output  
**Usage:** To provide interface developers with a description of the API
- **Endpoint URL:** `api/status?error={code}`  
**Method:** GET  
**Input:** Optional, error code in url  
**Returns:** A high-level description of the current status of the system (online or offline), or provides an explanation of an inputted error code  
**Usage:** To allow the interface to check if the processor is capable of accepting service requests
- **Endpoint URL:** `api/auth`  
**Method:** POST  
**Input:** Username and password in body  
**Returns:** Returns an access token for a provided username and password. This token should then be used in the header of all future API calls  
**Usage:** To allow the interface to authenticate a user, and receive a token for identification in future API calls

- Endpoint URL:** api/file  
**Method:** POST  
**Input:** File to be uploaded, access token in header  
**Returns:** A file package id corresponding to the uploaded file  
**Usage:** To allow the interface to upload files to be analyzed in a future report
- Endpoint URL:** api/file?report={reportId}&filename={filename}  
**Method:** GET  
**Input:** reportId, filename in url, access token in header  
**Returns:** Returns the contents of a file belonging to a report, if the provided access token is allowed to view it  
**Usage:** To allow the interface to display the contents of a file in a report
- Endpoint URL:** api/report/{filePackageId}  
**Method:** POST  
**Input:** filePackageId in url, report configuration in body, access token in header  
**Returns:** Submit a report request for the specified package id  
**Usage:** To allow the interface to request a report for a previously uploaded file
- Endpoint URL:** api/report/{filePackageId}  
**Method:** GET  
**Input:** filePackageId in url, access token in header  
**Returns:** Returns the status of a report (not processed, in queue) or the results of a report if it has completed processing  
**Usage:** To allow the interface to check if a report has been requested for a file, if the request is currently queued, or to retrieve the results of a report
- Endpoint URL:** api/feedback/action  
**Method:** POST  
**Input:** Action information in body, access token in header  
**Returns:** None  
**Usage:** To record an action performed on the interface for data collection purposes
- Endpoint URL:** api/feedback/rating/{feedbackId}  
**Method:** POST  
**Input:** Action information in body, access token in header  
**Returns:** None  
**Usage:** To record a rating on a feedback item the user submitted (helpful or confusing)

\* Variables in the URL are represented with {Identifier}, where identifier is the name of the variable.

### 3.4.2 Processing Module

The Processing Module is designed to operate as a supervisor that overlooks the report generation process. It is responsible for four actions:

1. to translate report requests for the service containers, and translate report responses for the System API to return to the interface,
2. to submit file packages for processing to the correct service containers, and compile them so they met any prerequisites,
3. to ensure service containers do not become overwhelmed with requests,
4. and to collect data generated at the processing stage.

The following four paragraphs will explain the details of each action.

(1) When translating a request for the service container, the processing module works with the file packager to combine the students submission with a formatted file containing their report configuration settings and generates a single tar archive. When translating a report response, the processing module parses the output file and passes it to the system API to be formatted and returned to the interface.

(2) After translating an incoming report request, the processing module will review the report configuration and apply any necessary prerequisite transformations a service container requires. For example, if an instructor uses any preprocessing tools or has specific test harness for their courses, they're applied to or inserted into the file package.

(3) Once a report request is made, and a file package has been created, the processing module creates a entry into a first-in-first-out queue for the appropriate service container. This entry contains everything required for a service container to process a report.

(4) The processing module is also responsible for recording all data generated at this stage, including: time of report request, contents of file package, report configuration, time of entry into queue, time of removal from queue, and the results returned from the service container.

## 3.5 Service Container

The system can contain one or more service containers (one for each course configured in the system, along with the default container), that draw from their own unique queues. Service Containers are the components responsible for analysing file packages, and generating the report feedback for them. Within every container is a controller, and one or more service.

Each service within the service container comprises a stand-alone unit of functionality. They are autonomous, stateless, and can be instantiated within n-number of different service containers. Their responsibility is to accept a file package (items that feedback is being generated for), perform an analysis on it, and return the results. Some examples of possible services include: a syntax correctness checking service, a spelling checking service, and a memory leak locator service. The instantiation, input, and output of every service is managed by the service controller.

The service controller is responsible for bootstrapping the services within its container, providing them with the appropriate input, and calling them (individually or in parallel) in the configured order. Once each service has completed execution, the controller concatenates all non-hidden data returned from the services, and the output file is returned to the processor component. The combined results of these services are used to formulate a report on the input package.

### 3.5.1 Service Container Configuration File

Service Containers are automatically prepared when the system is started, based on their configuration files. This design allows for instructors to easily extend the capabilities of the system, and create a new service container by creating a new configuration file. Configuration files are in yaml format, and contain a list of services. Within the configuration file, each service must contain the following properties: **class**, **arguments**, **timeout**, **priority**, **aborts**, and **hidden**. A sample configuration file is shown in Listing 3.2.

The **class** property specifies the name of the main class for the service. This service must publicly provide the functions described in the interface in listing 3.1, and have a constructor that can accept all parameters listed inside the arguments array. A new service can be implemented and added to the system for immediate use in a configured container by creating a new class that implements the interface. When the service is called, the class will be constructed, and the execute function will be called. The value returned by the execute function will be stored in an array accessible by the service controller containing all data returned from the services. If the service's hidden configuration flag is false, the data included in the array for the service will be available as feedback in the report.

The **arguments** property contains an array of parameters that to be passed into the class constructor. These values can either be static, or dynamically passed in using the @ symbol. By default, `packageName`, `reportId`, and `return` are available, but additional variables can be specified in the configuration file. Specifying an empty array, will result in the constructor receiving no

parameters. The number and type of arguments must be compatible with the class' constructor.

The **timeout** property specifies the maximum time in seconds a service should spend executing before it is terminated, and the return value set to false.

The **priority** property specifies the priority level of the service. The property is optional, and will default to 100 if omitted. When the service controller begins processing a report, it will run all services in order priority from 0 to 100. If two services have the same priority, they will be executed in parallel. Services executed in parallel cannot use data returned from one another.

The **aborts** property specifies whether the service can abort the report, preventing future services from executing. The property is optional, and will default to false if omitted. If set to true, and during report analysis the service's return value is set to false, the report is considered complete and none of the remaining services are executed. This setting is useful for security scanners, or services that check if the input is capable of being parsed.

The **hidden** property specifies whether the service's return value will be returned to the interface as feedback. This property is optional, and defaults to false.

```
1 interface ServiceInterface {
2     LoadInput(integer ReportId) : boolean
3     ExecuteService(integer ReportId) : boolean
4     CheckExecutionStatus(integer ReportId) : Status Code
5     GetNextFeedbackItem(integer ReportId) : Feedback Item
6     Unload(integer ReportId) : boolean
7 }
```

Listing 3.1: Functionality required to be publicly implemented by every service loaded in the system.

```
1 containerName: ExampleContainer
2 services:
3     securityScanner:
4         class: SecurityScanner # class implements ServiceInterface
5         arguments: [@packageId, @reportId] # array of variables
6         timeout: 1 # in seconds
7         aborts: true
8         priority: 10
9         hidden: true
10    gcc:
11        class: Gcc
12        arguments: [@packageId]
13        timeout: 1
14        priority: 20
15    cppchecker:
```

```

16     class: Cppcheck
17     arguments: [@packageId, @return[services][gcc]]
18     # uses return value of gcc
19     priority: 30
20   cccc:
21     class: Cccc
22     arguments: [@packageId]
23     priority: 30

```

Listing 3.2: An example of a configuration file defining a Service Container

The proposed architecture makes use of a service-oriented design to provide an implementation of a scalable web application capable of providing immediate feedback to students. The system allows for customization by the instructional staff using plugin services to enable the acceptance of any form of parsable input, and the generation of any type of automated feedback. The design consists of three main components, the interface, processor, and service container. The student makes all their interactions through the interface, the processor receives input and prepares it for analysis, and the service container performs the analysis and returns the feedback.



## Chapter 4

# Implementation and Results

The thesis of this research is that *a scalable web application can be created to provide immediate feedback on a student's machine-parsable work product in a way that is beneficial to the student's learning experience*. This chapter will describe a prototype implementation of the architecture in section 4.1, and an experiment performed using the prototype in section 4.2. The chapter closes by analysing the data gathered in a discussion around the research questions, and draws a conclusion on the thesis in section 4.3.

### 4.1 Implemented System Prototype

For our experiment, a prototype of the system described in chapter 3 was built. The prototype was provided to all students enrolled in CIS:2500 Intermediate Programming (332 students), and CIS:3110 Operating Systems (226 students) at the University of Guelph during the winter semester of 2016. All three courses were approximately 16 weeks long, with weeks 15 and 16 dedicated to final examinations with no classes or assignments. The system, dubbed IFS, was implemented as a web application hosted at <https://ifs.socs.uoguelph.ca>. IFS was fully functional and provided authentication, feedback configuration, code submission, queueing, and a feedback page for displaying report results.

IFS's implementation included all 3 components of the architecture. The interface was built using PHP, the processor component with PHP and Perl, the queue made use of Redis, and the service container was written in C. To speed up development, a dynamic service controller factory was not implemented and configurations were hard-coded within the service container instead.

When a student navigated to the IFS website, they were greeted by a login page, shown in figure 4.1. IFS made use of the school's internal identification system (LDAP), allowing students to use their university username and password to log into the system. Once authenticated, the student was taken to the configuration and upload form.

Figure 4.1: The login page for IFS, containing a simple banner, and fields for the student to enter their University of Guelph username and password.

Figure 4.2: The upload page for IFS, containing a file upload form, a course selection, options to determine the type of analysis to perform, and an expandable advanced configuration shown in figure 4.3.

The configuration and upload form served two functions: 1) setting the configuration settings for the requested report, and 2) uploading the files to be analysed for the report. A screenshot of the page is shown in figure 4.2. The configuration options on the page defaulted to specific values previously set by the course instructor, however, they allowed changes to be made by the student. IFS accepted uploads as either a single source code file, or an archive of multiple source code files in zip, or tar format. The archives were restricted to 50 source files in any directory structure, totalling a size less than one megabyte.

The screenshot shows the 'Advanced' configuration section. It has three main parts: 'Syntax check' with buttons for 'ansi/c89', 'c99', 'c11', and 'gnu11'; 'Advanced Checks' with checkboxes for 'Style', 'Performance', 'Portability', and 'Unused Functions'; and 'Identifier naming convention' with checkboxes for 'Variables', 'Functions', and 'Types', each followed by a text input field containing '[a-z]\*'.

Figure 4.3: The expanded advanced configuration section available on the upload page of IFS. The default options are determined by a students course, but can be changed.

Once a report was successfully requested, the student was redirected to a page where they were shown a report generation progress bar, and an estimated wait time. During this time, IFS performed a basic security scan on the files while extracting them from the uploaded archive. The files were then sent to the service container, where they were run through the configured services. For the purposes of our experiment, services were built around cppcheck, gcc, ctags, and a custom built symbol parser (referred to as idcheck). The services were configured to run in parallel, and dumped their output to a file, which was returned to the interface in an array of normalized JSON objects (example in listing 4.1), and rendered on the feedback page.

```

1 {
2   'checker': 'gcc',
3   'msg': 'Missing semi-colon before 'int'',
4   'file': 'sample.c',
5   'line': 3
6   'keyw': 'missingsemicolon',
7   'scope': 1
8 }
```

Listing 4.1: An example of a normalized feedback item in JSON format. *checker* contains the service that outputted the item, *msg* contains the feedback message, *file* contains the file the feedback relates to, *line* contains the line number of the input that generated the feedback item, *keyw* contains the type of error, and *scope* contains the ID of the function or block the feedback relates to.

The feedback page consisted of a listing of all items returned from the services. Within each item, a message explaining the feedback, the file the item relates to, and the line number it occurred on was shown. A screenshot of the feedback page is shown in figure 4.4. If the student hovered their mouse over the feedback item, they were shown the input lines related to that item. When

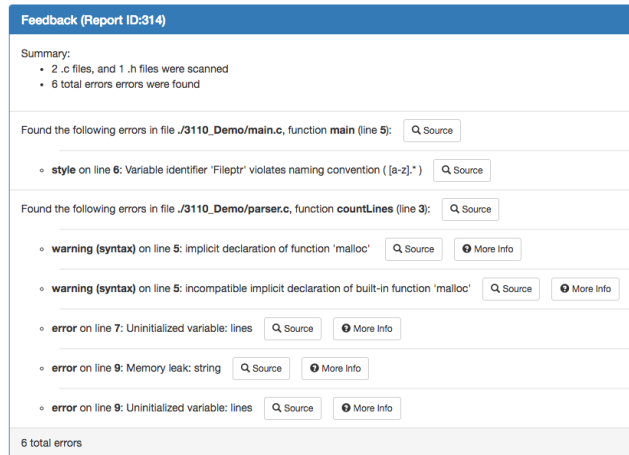


Figure 4.4: The feedback page for IFS, showing 6 feedback items from 3 files.

the student clicked on the *show source* button, a modal pop-up containing the entire source file related to the item was shown with all lines that produced feedback highlighted. On some common feedback messages, a *more info* button was available with the item, which provided the student with a simple explanation of the error, a sample input that would cause the error, a sample solution, and buttons to vote on whether they thought the additional information was helpful or confusing. An example of this is shown in figure 4.5.

Information for each submission was gathered and stored within IFS's database, including an analysis of the system, processing times, and interactions students made with the feedback. When aggregated with data collected from the experiment's survey, resulting data trends that support the thesis were found. In the following section, a summary of all data gathered during the experiment will be provided.

## 4.2 Experimental Design

The experiment, performed during the Winter 2016 (W16) semester at the University of Guelph, provided students with an implemented prototype of the designed system. The prototype collected data that described how the student used the system, the quality of a student's work product, and their engagement with the system. At the end of the experiment, a survey was provided to all participants asking questions that evaluated their self-measured satisfaction with our system, and their engagement with their computer science courses. A copy of the survey is provided in Appendix B. The aggregation of both datasets, along with data generated by analysing submissions post-experiment, was used to suggest answers to the following questions, providing evidence for our thesis:

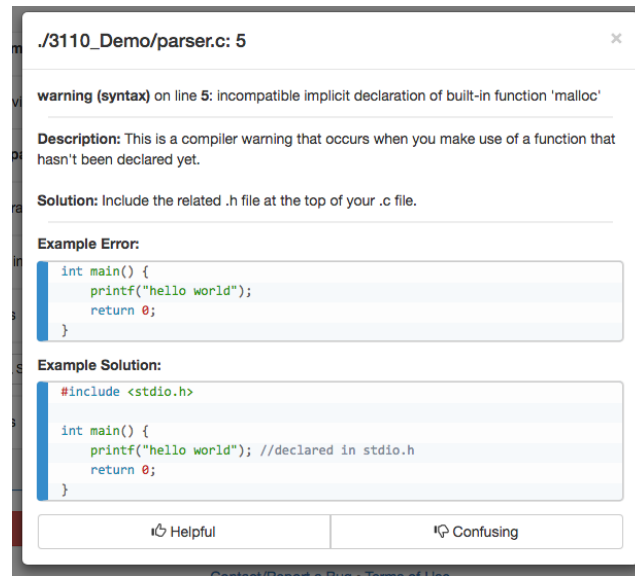


Figure 4.5: The more info dialog for IFS, shown when a user clicks the “more info” button on a feedback item.

1. Does the designed system reliably produce feedback for students?
2. Does the designed system produce feedback students find valuable?
3. Does the designed system improve the student learning experience?
4. Does the designed system improve students' programming ability?

#### 4.2.1 Data Collected From IFS

During the course of the experiment, IFS collected data related to its usage. During authentication, the students username, IP Address, and date of access was recorded. On successful file submission, the complete contents of the upload, and time were stored. During feedback generation, the details of each feedback item were stored. In addition, every interaction the student had with their feedback was logged. Tables 4.3), 4.1, 4.2 provide a complete listing of the data points collected during these three stages. Prior to analysis the collected data was anonymized: all student names and usernames were passed through a one-way cryptographic hash function, IP addresses were converted to either on or off school campus categories, and times were converted to a 6 hour window.

#### 4.2.2 Data Generated By Code Analysers

At the end of April, all submissions made to IFS were run through two code analysers: Frama-C and CCCC (C and C++ Code Counter). Frama-C and CCCC [64, 65] are developer tools designed to provide metrics on software

| Data Collected During Upload |   |   |
|------------------------------|---|---|
| Data Point                   | Description   | Hygiene / Anonymization Applied                     |
| User                         | School username of student who uploaded a file for analysis | Username hashed                                     |
| Uploaded File                | Complete contents of upload                                 | Names removed from all files                        |
| Datetime Upload              | Date and time of upload                                     | Converted to date and 6 hour window                 |
| Number of Files              | Number of files within the upload                           |   |
| Datetime Analysis            | Date and time of analysis completion                        | Converted to number of seconds to complete analysis |

Table 4.1: Data collected by IFS during a file upload.

| Data Collected For Each Feedback Item |   |   |
|---------------------------------------|---|---|
| Data Point                            | Description                               | Hygiene / Anonymization Applied                     |
| Associated Report                     | Report associated with feedback           |   |
| Feedback Message Identifier           | Feedback message associated with the item |   |
| File                                  | Name of file error occurs in              | Student name (if present in the filename) is hashed |
| Line                                  | Line in file the feedback item relates to |   |
| Service                               | Name of service that reported the item    |   |

Table 4.2: Data collected by IFS for each feedback item generated for a report.

| Data Collected For Every Interaction With Feedback |   |                                     |
|--|---|-------------------------------------|
| Data Point   | Description   | Hygiene / Anonymization Applied     |
| User   | School username of student who uploaded a file for analysis | Username hashed                     |
| Associated Report                                  | Report associated with feedback                             |                                     |
| Datetime Interaction                               | Date and time of interaction                                | Converted to date and 6 hour window |
| Type of interaction                                | One of:   |                                     |
|  | - Mouse hovered over feedback item                          |                                     |
|  | - Mouse hovered over More Info                              |                                     |
|  | - Mouse hovered over View Source button                     |                                     |
|  | - More Info button clicked on feedback item                 |                                     |
|  | - More Info rated helpful/confusing                         |                                     |
|  | - Contents of upload file viewed with feedback              |                                     |
|  | - Close using "logout" button                               |                                     |
|  | - Close using by closing browser window                     |                                     |

Table 4.3: Data collected by IFS when a student interacts with a feedback item.

| Data Generated By Frama-C and CCCC Per Submission |   |  |
|---|---|--|
| Data Point  | Description   | Hygiene / Anonymization Applied                      |
| Associated Report                                 | Report associated with feedback   |  |
| Source lines of code                              | Number of lines of code analyzed  |  |
| Source lines of comments                          | Number of comments in code  |  |
|   | Number of occurrences of each of the following:                           |  |
|   | - Lines of comments   |  |
|   | - Global Variables  |  |
|   | - If statements   |  |
|   | - Loop statements   |  |
| Counts  | - Goto statements   | Student name (if present in the filename) is hashed  |
|   | - Variable assignments  |  |
|   | - Exit points   |  |
|   | - Function declarations   |  |
|   | - Function definitions  |  |
|   | - Function calls  |  |
|   | - Pointer dereferences  |  |
| Cyclomatic complexity                             | Measure of linearly independent code paths                                |  |
| Function Names/Calls                              | Name of each define function, and the number of times each one was called | Student name (if present in function name) is hashed |

Table 4.4: Data generated by Frama-C and CCCC, post experiment, for each submission made to IFS.

source code written in C. Metrics are calculated by performing a static analysis of source code, which is a method of measuring code properties without executing it. While insufficient for pinpointing every bug that may appear during execution, static analysis is often relied on for measuring syntactic features (e.g. lines of code, vocabulary usage, or number of possible execution paths) [66]. As static analysis has a reliance on syntactically correct code, only code that could be completely compiled were used. To ensure accuracy in the results, both tools were applied to every submission. The results of both runs were checked for equality and in any situation where the counts were incorrect between the two programs, the statistics for that submission were considered inaccurate and discarded. Inaccuracies occurred in two percent of the submissions analysed by the tools.

For this study, Frama-C and CCCC were used to generate basic counts, function usage information, and the cyclomatic complexity of each submission, and within each code block in the source files. A complete list of all types of data generated is shown in table 4.4.

#### 4.2.3 Data Collected Via Self-Assessment Survey

When our experiment closed, an email was sent to all 285 participants requesting they fill out a survey, with a draw to win an Amazon gift card as incentive. The survey was not anonymous, but participants were assured their coursework and grades would not be affected by their survey responses, or survey participation. 113 participants responded to the survey. Questions in the survey collected both quantitative and qualitative responses, asking students to provide their answers to questions on a scale of 1-5 or provide written responses. Only the quantitative responses were used in this dissertation.

A Likert Scale is a psychometric scale often used for questionnaire responses. Questions related to IFS in the self-assessment survey made use of a Likert Agreement Scale to quantify responses. A Likert Agreement Scale survey tasks the respondent with specifying their agreement or disagreement with a statement, referred to as a Likert Item, on a symmetric series of options [67]. For the survey provided to IFS users, options included Strongly Agree, Agree, Neutral, Disagree and Strong Disagree. The following is a list of the Likert Items, whose ratings were used to support the thesis.

- The IFS system was easy to use.
- The information about my code provided by IFS was useful.
- I used IFS for most of my C coding assignments this semester.
- I used IFS to locate the code associated with the error or warning message.
- I used IFS to find out what error and warning messages meant.
- IFS helped me find solutions to errors in my code.

Similar to the questions related to IFS, questions related to a student's educational engagement were asked using a Likert Frequency Scale. The Likert Frequency Scale is a symmetrical scale used to measure the approximate number of times a respondent performs an action. Possible response options in the survey included Very Often, Often, Sometimes, Rarely and Never. The responses to the following questions were used to support the thesis:

- Over the past semester (Winter 2016), about how often have you done each of the following:
  - Asked questions in class or contributed to class discussions (in W16 semester).
  - Participated in forums or contributed to online discussions(in W16 semester).
  - Skipped class(in W16 semester).
  - Took detailed notes during class(in W16 semester).
- In semesters prior to 2016, about how often did you do each of the following:
  - Asked questions in class or contributed to class discussions (previous semesters).
  - Participated in forums or contributed to online discussions(previous semesters).
  - Skipped class(previous semesters).
  - Took detailed notes during class(previous semesters).

### 4.3 Results and Discussion

During the course of the experiment, IFS was used by 285 students. These students collectively made submissions to the system 2715 times. Table 4.5 shows counts of these submissions broken down by week, and figure 4.6 shows



| Week | Max Number<br>of Submissions<br>Per User | Average Number<br>of Submissions<br>Per User | Total<br>Submissions | Number of<br>Unique Active<br>Users |
|------|--|--|----------------------|-------------------------------------|
| 2    | 7  | 7  | 7                    | 1                                   |
| 3    | 14                                       | 2.84   | 230                  | 81                                  |
| 4    | 26                                       | 3.34   | 280                  | 84                                  |
| 5    | 9  | 3.47   | 52                   | 15                                  |
| 6    | 9  | 3.14   | 44                   | 14                                  |
| 7    | 15                                       | 3.93   | 287                  | 73                                  |
| 8    | 7  | 2.08   | 25                   | 12                                  |
| 9    | 8  | 2.5  | 40                   | 16                                  |
| 10   | 19                                       | 3.61   | 206                  | 57                                  |
| 11   | 22                                       | 4.42   | 420                  | 95                                  |
| 12   | 17                                       | 4.29   | 146                  | 34                                  |
| 13   | 30                                       | 5.97   | 537                  | 90                                  |
| 14   | 20                                       | 4.43   | 434                  | 98                                  |
| 15   | 2  | 2  | 2                    | 1                                   |
| 16   | 4  | 4  | 4                    | 1                                   |

Table 4.5: Summary of submission counts, and unique active users by week. A user was considered active if they made at least one submission that week, and looked at the feedback page. Weeks 2, 15, and 16 were excluded from analysis because they only had one unique active user.

that on average, users increased their use of the system every week. The submissions contained a collective total of 6267 C header files, and 10637 C source files. From these files, IFS accepted and analysed over 1.7 million lines of code producing 18239 different pieces of feedback. On average, this results in 9.53 system uses per participating student, and 6.7 pieces of feedback provided per use. Of the 18239 pieces of feedback generated, students requested more information from the system on 7614 of them. Table 4.6 shows the 20 most commonly occurring errors and warnings in submissions that were used to generate feedback items. Our survey had responses from 113 of the 285 students, with 91 completing all 39 questions. In this section, we'll leverage the vast amount of data collected to discuss answers to the questions derived from the thesis:

- Does the designed system reliably produce feedback for students?
- Does the designed system produce feedback students find valuable?
- Does the designed system improve the student learning experience?
- Does the designed system improve students' programming ability?

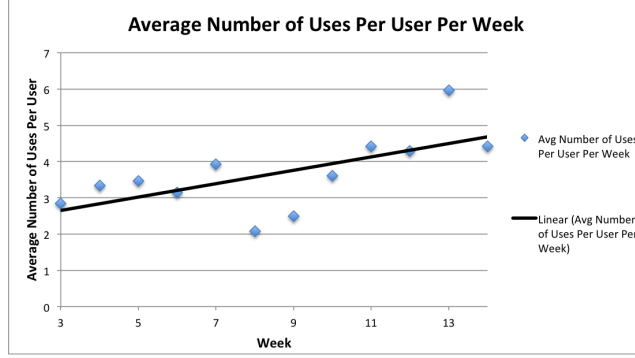


Figure 4.6: Average number of uses per user, per week. Week 2, 15, and 16 were excluded as they only had one active user.

| Error/Warning Type               | Total Number of Occurrences |
|----------------------------------|-----------------------------|
| Unused Function                  | 5528                        |
| Reducible Variable Scope         | 2236                        |
| Incorrect Variable Naming Scheme | 1747                        |
| Memory Leak                      | 1344                        |
| Dangerous Use Of Scanf Formatter | 1066                        |
| Including Non-Existent File      | 1009                        |
| Redundant Assignment             | 645                         |
| Array Index Out Of Bounds        | 393                         |
| Unread Variable                  | 354                         |
| Null Pointer Exception           | 352                         |
| Resource Leak                    | 305                         |
| Uninitialised Variable Used      | 300                         |
| Unused Variable                  | 278                         |
| Redefinition of Type             | 276                         |
| Useless Assignment               | 179                         |
| Variable Set But Not Used        | 176                         |
| Memory Leak On Realloc           | 173                         |
| Implicit Declaration of Function | 132                         |
| Variable Self Assignment         | 118                         |
| Duplicate Expression             | 106                         |

Table 4.6: Top 20 errors and warnings found by IFS, ordered by their number of occurrences.

#### **4.3.1 Does the designed system reliably produce feedback for students?**

The reliability of any educational tool is critical. It is crucial that students have access to resources when they need them, and that the resources presented are correct and predictable. Throughout the entire experiment, IFS provided feedback on every submission without error. Running the submissions through IFS a second time with no load, showed that results were consistent regardless of user capacity. On average, IFS was able to return feedback 2.5 seconds after upload, and the longest a student had to wait was 5.7 seconds. This longer time occurred during a window in which IFS had its largest number of concurrent users: 55. The number of queued reports reached 7. During the experiment, IFS maintained 100% uptime and required no maintenance. Usage of the prototype shows that the designed system can reliably produce consistent feedback for students at a small scale.

#### **4.3.2 Does the designed system produce feedback students find valuable?**

It is imperative that the designed architecture is capable of producing valuable feedback. If students do not find the feedback beneficial, then there is little incentive to use the system. To measure the value of the feedback IFS outputted, we relied on the student self-assessment survey. Eighty-eight percent of our respondents found the feedback to be useful, selecting either *agree* or *strongly agree* with the statement: “The information about my code provided by IFS was useful”. Using a one sample test of proportions, this result is statistically significant with a p-value less than 0.0001. As a result, we conclude that the architecture is capable of producing feedback students find valuable.

#### **4.3.3 Does the designed system improve the student learning experience?**

Even though literature shows that immediate feedback has a positive impact on a students learning experience, blindly introducing new educational tools carry of the risk creating negative learning experiences. While results show that IFS is capable of producing reliable, and consistent feedback that students find valuable, we also need to consider how it is used. Ideally, the feedback should enable students to progress, where they would otherwise get stuck, without having to rely on the traditional delayed feedback from their instructional support team. To measure this, we looked at IFS’s ease of use, and how the respondents used the tool. We also considered student engagement, as the literature review has shown a positive co-relation between satisfaction and engagement. An improvement in the student learning experience can be assumed if IFS is easy to use and it provides feedback that helps students progress. If the tool is unintuitive, it can easily cause frustration and breaks in a student’s flow experience when they try to get feedback. If the feedback

provided does not enable the student to progress when stuck, then it is simply additional noise that can overwhelm the student.

Ninety-three percent of the survey respondents agreed, or strongly agreed with the statement *the IFS system was easy to use*, with only 2.6 percent disagreeing. Based on this response, it is improbable for IFS to be a source of frustration and detract from a student's learning experience.

Seventy-five percent of the survey respondents used IFS for the majority of their C coding assignments during the W16 semester with 66 percent of them leveraging IFS to locate code in their projects that was the source of errors or warnings, 43 percent to find out what errors and warning messages meant. 56 percent of the users found IFS to be a helpful tool for finding solutions to errors in their code. These results suggest that approximately half of the respondents used IFS to receive immediate feedback to help them solve issues on their school work that would prevent project compilation, and therefore, delay completion of their programming tasks. Based on this response, the feedback produced by IFS does help a student progress with their school work.

Four percent of our survey respondents increased their in-class engagement in the W16 semester relative to previous semesters, 12 percent increased their online engagement, and 15 percent decreased the number of times they skipped classes. Unfortunately, these numbers are not large enough to be predictive of any effect IFS may have had on student engagement or satisfaction without a control group that had different outcomes.

#### **4.3.4 Does the designed system improve students' programming ability?**

Assessing a student's programming ability is difficult, however it is important to consider any changes IFS may make. To measure IFS's influence, changes in the participants' function modularity were analysed, as well as the median number of errors their programs produced. While these metrics focus on only two aspect of programming ability, they are able of providing an accurate quantification of our participants' capabilities.

Modularity was one of the key learning outcomes of CIS\*2500 (the course with the largest number of participants in the experiment) for the Winter 2016 semester. Therefore, it represents one of the few quantifiable aspects of growth in the programming ability of the majority of participants, with a reasonable expectation of change. To measure modularity, the number of lines per functions in every submission were calculated post-experiment with CCCC. For the purposes of analysis, a function with fewer lines was assumed to be more modular.

| Group           | Number of Uses   | P-Value | Size of Group |
|-----------------|------------------|---------|---------------|
| Rare Usage      | $1 < x \leq 10$  | 0.665   | 129           |
| Regular Usage   | $10 < x \leq 20$ | 0.01287 | 60            |
| Excessive Usage | $x > 20$         | 0.2948  | 40            |

Table 4.7: P-Values for the hypothesis supporting a significant difference between the average function modularity of submissions made during the first week a user used IFS and the last week, grouped by number of system uses. A significant difference in students who used the system 10 to 20 times was found, with an improvement in their function modularity.

To identify any significant differences in participants' function modularity, a delta of the average lines per function during the first week a participant used IFS and the average lines per function during the last week a participant used IFS was calculated. The deltas for each participant were then grouped into three categories based on their IFS usage: Rare Usage, Regular Usage, and Excessive Usage. For each group, a Wilcoxon Signed Rank Test was performed. Table 4.7 shows the number of uses of IFS for categorization, the size of each group, and the p-value resulting from the test. Only the regular usage group had results that represented a significant change in their function modularity.

The most interesting outcome of this analysis is that students who rarely used IFS showed no significant difference in their function modularity the first week they used the system and the last week, while the students who used IFS regularly did. In addition, the majority of the regular use group presented an improvement in their function modularity. These results suggest that students who incorporate moderate IFS use into their learning environments are more likely to show an improvement in their function modularity. Following the reasoning behind one of the learning outcomes of CIS\*2500, this represents an improvement in one aspect of the students programming ability.

The findings also show that students who used IFS excessively did not present any significant changes in their function modularity the first week they used the system and the last week. It is hypothesised that these students did not see any improvements, because they were either a) experienced developers who leveraged the tool to easily find errors, treating IFS as a software development tool rather than an educational tool, or b) weak developers who misunderstood key programming concepts, using IFS as a crutch to fix their programs without learning from the feedback provided. This hypothesis is backed by two facts. A subset of this group had little to no change in their already high (relative to the other participants) level of modularity. While the remaining students had very low modularity levels, consistently made the same errors in their code, and spent very little time interacting with the feedback page. Individual

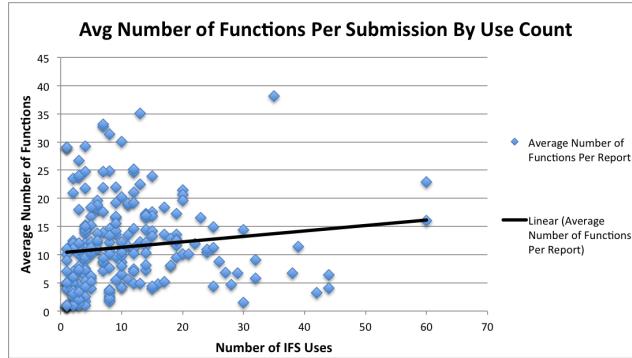


Figure 4.7: Average number of functions per submission by number of IFS uses. A linear regression is shown with a increasing trend.

submissions from the group either made use of an relatively low, or relatively high number of functions (see figure 4.7), providing further support for the large differences in this group. This exposes interesting problem our architecture was not designed to handle: overachieving and underachieving students often need customized feedback based on their skill level, to ensure they use that feedback to learn. It also suggests that IFS may be useful in a professional software development environment, in addition to the educational environment.

Errors can be a result of the programmer misunderstanding syntax, or constructs, of the language they are using and can therefore be referenced in a measurement of a programmer's ability. Errors were detected and recorded for every submission made to IFS. For the purposes of this experiment, a decreasing number of errors in a participant's submissions overtime, would represent an improvement in that student's understanding of programming.

Figure 4.8 shows the median number of errors per submission by use count. There is a clear downwards trend showing that as participants increase the number of times they use IFS, the number of errors found in their submission decrease. Based on these results, it is likely that IFS can improve a student's ability to find and fix errors on their own, reducing the need for external feedback in the future.

#### 4.3.5 Summary of Results

With the data collected during, and after, the experiment, all the research questions were answered. The reliability of the system is proven at a small scale. The design of the system does enable the production of feedback students benefit from. The designed system does improve the student learning experience. And students' programming ability does appear to be improved

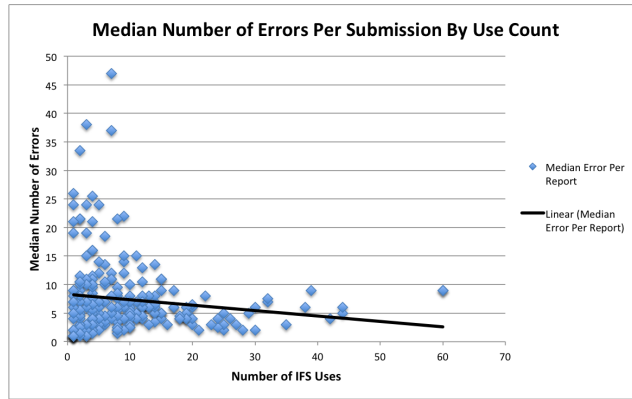


Figure 4.8: Number of median errors per submission by number of IFS uses. A linear regression is shown with a decreasing trend.

through the use of the system. The results of the experiment paint a positive picture for the use of automated immediate feedback systems in the educational environment.

## Chapter 5

# Limitations, Future Work and Conclusions

The results of the experiment chiefly show that a scalable web application can be created to provide immediate feedback on a student's machine-parsable work product in a way that is beneficial to the student's learning experience, with some limitations. The participants in the study, for example, were subject to selection bias. Even with these limitations, the positive findings present an welcome opportunity for future work that not only addresses these limitations, but further explores the effects of automated immediate feedback systems, their applications in other fields, and how they can be further integrated in learning environments. This section will explain the limitations of this study, introduce further research topics related to the thesis, and conclude with a summary of the entire project.

### 5.1 Limitations

The experiment performed suffered from three limitations that should be kept in mind:

1. Experiment participation was voluntary, and open to everyone in the selected courses. As result, there was an inherit selection bias in the user base.
2. *More Info* modals in the system were limited, and may not have presented everyone with equal opportunity to receive a complete learning experience with each feedback items.
3. Participation in the experiment was not large enough to test the reliability of the system at a large scale.

The following paragraphs provide an explanation for the occurrence of the limitation, the impact they have on the results, and how they can be avoided in future work.



1) As a result of the educational benefits a student may receive for participating in our experiment, ethical restrictions prevented us from establishing a control group. The system was instead made available to all students in the selected courses, and they were given the option to voluntarily use it. Students who chose to participate are likely students who are already moderately engaged with their course, and have a interest in improving their knowledge of programming or wish to supplement their school provided educational material with additional external materials and tools. This has the implication that at least some of participants would show improvement in their programming abilities over time whether or not they used the system. This bias has no relation to a students programming ability at the beginning of our experiment nor their satisfaction with the system. Since our analysis, and survey question responses looked at changes over time for each individual student based on their IFS usage, the bias has no impact on the findings except limiting the applicability of them to students who are willing to voluntarily use an educational tool.

2) The explanations and examples for the *More Info* modals related to feedback items for errors and warnings were manually written during development. Since it was not feasible to write one for every possible error, it was decided to only write them for some of the errors most common to novice programs. Unfortunately, there was very few studies previously performed with information specific enough to pinpoint errors likely to occur in beginner C programs. Instead, instructors and online forums were referenced. After analysing the results of the experiment, not all the of top errors had related explanations. Due to the missing explanations, it is likely the number of participants (43.1%) who agreed with the statement *I used IFS to find out what error and warning messages meant* was lower than it could have been. This may have also negatively affected a student's opinion on the usefulness of IFS as an educational tool, and instead caused them to categorize it as a development tool. Using the results of this experiment, future revisions of IFS may be able to target every single error a novice is likely to encounter.

3) Since the distribution of IFS to students was limited, and the number of volunteer participants small, the reliability of the system was not tested at a large scale. Nevertheless, the architecture will theoretical scale linearly with resources maintaining a consistent and reliable educational tool for students regardless of the size of deployment or user base. Future work can experiment with distributing the system to larger audiences, to find the point at which additional resources are required to maintain performance and determine the scalability of the platform. Another option, is to generate automated load tests using computer programs.

## 5.2 Future Work

The positive results of this experiment provide a clear indication that further research in automated immediate feedback systems should be conducted. In particular, further analysis on the data collected during this experiment should be performed, additional services targeting different educational domains should be developed, and the architectural design should be expanded for better integration with existing educational tools.

IFS produced an incredibly large dataset, and student submissions can be infinitely reprocessed by different code analysers to produce endless statistics about the participants programs. A relatively small subset of the gathered data was used to answer this thesis' research questions, but many other questions can be answered using the result set. For example, very few researchers have established common errors novice programmers make. Typical changes in a novice programmer's style could also be identified and used to create a framework for measuring a student's skill level based on their work products that is more accurate than grades. The analysis performed here completely ignored the impact our system had on student usage with other learning tools, or if there was a reduction in instructional staff support time. Assuming student satisfaction stayed the same, or increased, an automated immediate feedback system could result in a huge cost reduction in education, and may be worth exploring as a replacement to staff or other more costly learning tools.

The services built for use by IFS focused on the analysis of C language programs. Further work needs to be done to create services for other inputs, expanding the domains in which it is useful. Using existing technology and libraries, other programming languages, chemical structure drawings, math calculations and proofs, human language grammar and spelling, logic statements, and circuit diagrams are all inputs that can be analysed easily without any changes to IFS. These additional services would enable a thorough testing of the dynamic service controller configuration and creation, scalability, and generate data to answer questions about educational environments in other domains.

The architecture of IFS can be further expanded to provide integration points with school learning management systems (LMS), to better encourage use and provide feedback on work products that a student would not normally submit to IFS. For example, it may be useful to have IFS automatically generate feedback on an assignment submission for the student to review before confirming. This would dramatically reduce the number of unexpected grades, and make students immediately aware of any issues with their submission. The architecture could also be modified to decrease instructor workload by assisting the instructional team with grading or plagiarism detection.

### 5.3 Conclusion

Quality feedback provided in a timely manner has a positive impact on student education. However, as class sizes continue to grow, the provision of feedback from instructional staff becomes increasingly difficult. In order to ensure students have access to feedback, alternative methods of feedback generation need to be explored. One solution is to offload the workload of providing feedback from the instructional team, to an automated computer system.

In this thesis, we proposed that a scalable web application can be created to provide immediate feedback on a students machine-parsable work product in a way that is beneficial to the students learning experience. A technical design for an immediate feedback framework was created, and a prototype of the system was provided to computer science students at the University of Guelph. From the use of the system, it was discovered that the presented design is reliable, enables the production of feedback students benefit from, improves the student learning experience, and improves student programming abilities. The positive results of the experiment performed open the door to further research in the design, implementation, and application of automated feedback systems in educational environments as a solution for the provision of feedback.

# Bibliography

- [1] M. Kuikka, M.-J. Laakso, and M. Joshi, “The effect of the immediate feedback by the collaborative education tool ville on learning for business mathematics in higher education,” *Journal of Educational Technology Systems*, vol. 45, no. 1, pp. 34–49, 2016. [Online]. Available: <http://dx.doi.org/10.1177/0047239515625887>
- [2] E. Kaila, T. Rajala, M.-J. Laakso, R. Lindén, E. Kurvinen, and T. Salakoski, “Utilizing an exercise-based learning tool effectively in computer science courses,” *Olympiads in Informatics*, vol. 8, pp. 93–109, 2014.
- [3] T. R. M.-J. Laakso, E. Kaila, and T. Salakoski, “Effectiveness of program visualization: A case study with the ville tool,” *Journal of Information Technology Education*, vol. 7, pp. 15–32, 2008.
- [4] E. Kurvinen, R. Lindén, T. Rajala, E. Kaila, M.-J. Laakso, and T. Salakoski, “Automatic assessment and immediate feedback in first grade mathematics,” in *Proceedings of the 14th Koli Calling International Conference on Computing Education Research*. ACM, 2014, pp. 15–23.
- [5] R. S. Cole and J. B. Todd, “Effects of web-based multimedia homework with immediate rich feedback on student learning in general chemistry,” *J. Chem. Educ.*, vol. 80, no. 11, p. 1338, 2003.
- [6] K. W. Lux, M. J. D’Amato, B. Anderegg, K. A. Walz, and H. W. Kerby, “Introducing new learning tools into a standard classroom: A multi-tool approach to integrating fuel-cell concepts into introductory college chemistry,” *Journal of Chemical Education*, vol. 84, no. 2, p. 248, 2007. [Online]. Available: <http://dx.doi.org/10.1021/ed084p248>
- [7] R. L. Oliver, “Processing of the satisfaction response in consumption: a suggested framework and research propositions,” *Journal of Consumer Satisfaction, Dissatisfaction and Complaining Behavior*, vol. 2, no. 1, pp. 1–16, 1989.
- [8] J. M. Braxton, A. S. Hirschy, and S. A. McClendon, *Understanding and Reducing College Student Departure: ASHE-ERIC Higher Education Report, Volume 30, Number 3*. John Wiley & Sons, 2011, vol. 16.
- [9] W. E. Knox, P. Lindsay, and M. N. Kolb, “Higher education, college characteristics, and student experiences: Long-term effects on educational satisfactions and perceptions,” *The Journal of Higher Education*, vol. 63, no. 3, pp. 303–328, 1992. [Online]. Available: <http://www.jstor.org/stable/1982017>

- [10] P. Donaldson, L. McKinney, M. Lee, and D. Pino, "First-year community college students' perceptions of and attitudes toward intrusive academic advising," *NACADA Journal*, vol. 36, no. 1, pp. 30–42, 2016.
- [11] C. Gielow and V. Lee, "The effect of institutional characteristics on student satisfaction with college," in *American Educational Research Association Annual Meeting, New Orleans, LA*, 1988.
- [12] D. R. Garrison and M. Cleveland-Innes, "Critical factors in student satisfaction and success: Facilitating student role adjustment in online communities of inquiry," *Elements of quality online education: Into the mainstream*, vol. 5, pp. 29–38, 2004.
- [13] A. W. Astin, "Four critical years. effects of college on beliefs, attitudes, and knowledge." 1977.
- [14] K. M. Elliott and D. Shin, "Student satisfaction: An alternative approach to assessing this important concept," *Journal of Higher Education Policy and Management*, vol. 24, no. 2, pp. 197–209, 2002. [Online]. Available: <http://dx.doi.org/10.1080/1360080022000013518>
- [15] E. A. Gazza and A. Matthias, "Using student satisfaction data to evaluate a new online accelerated nursing education program," *Evaluation and Program Planning*, vol. 58, pp. 171 – 175, 2016. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S0149718916300477>
- [16] J. Douglas, A. Douglas, and B. Barnes, "Measuring student satisfaction at a uk university," *Quality assurance in education*, vol. 14, no. 3, pp. 251–267, 2006.
- [17] G. D. Kuh, "Assessing what really matters to student learning inside the national survey of student engagement," *Change: The Magazine of Higher Learning*, vol. 33, no. 3, pp. 10–17, 2001.
- [18] R. Hughes and C. R. Pace, "Using nsse to study student retention and withdrawal," *Assessment Update*, vol. 15, no. 4, pp. 1–2, 2003.
- [19] J. Burton, "Student engagement and success."
- [20] E. T. Pascarella, "Student-faculty informal contact and college outcomes," *Review of Educational Research*, vol. 50, no. 4, pp. 545–595, 1980. [Online]. Available: <http://dx.doi.org/10.3102/00346543050004545>
- [21] G. D. Kuh, T. M. Cruce, R. Shoup, J. Kinzie, and R. M. Gonyea, "Unmasking the effects of student engagement on first-year college grades and persistence," *The Journal of Higher Education*, vol. 79, no. 5, pp. 540–563, 2008.
- [22] A. Bandura, "Self-efficacy: toward a unifying theory of behavioral change." *Psychological review*, vol. 84, no. 2, p. 191, 1977.
- [23] I. B. Weiner, Ed., *Self-efficacy*. John Wiley and Sons, Inc, 2009, vol. The Corsini Encyclopedia of Psychology.
- [24] B. J. Zimmerman, "Self-efficacy: An essential motive to learn." *Contemp Educ Psychol*, vol. 25, no. 1, pp. 82–91, 1 2000.
- [25] E. A. Linnenbrink and P. R. Pintrich, "The role of self-efficacy beliefs in student engagement and learning in the classroom," *Reading & Writing Quarterly*, vol. 19, no. 2, pp. 119–137, 4 2003.
- [26] D. E. Dulany, "Awareness, rules, and propositional control: A confrontation with sr behavior theory," 1968.

- [27] K. Trehan and M. Pedler, "Cultivating foresight and innovation in action learning: reflecting ourselves; reflecting with others," 2011.
- [28] C. Rigg and K. Trehan, *Action Learning*. Boston, MA: Springer US, 2012, pp. 62–66. [Online]. Available: [http://dx.doi.org/10.1007/978-1-4419-1428-6\\_824](http://dx.doi.org/10.1007/978-1-4419-1428-6_824)
- [29] M. Sherer, J. E. Maddux, B. Mercandante, S. Prentice-Dunn, B. Jacobs, and R. W. Rogers, "The self-efficacy scale: Construction and validation," *Psychological Reports*, vol. 51, no. 2, pp. 663–671, 1982. [Online]. Available: <http://dx.doi.org/10.2466/pr0.1982.51.2.663>
- [30] B. J. Zimmerman and A. Bandura, "Self-efficacy and educational development," *Self-efficacy in changing societies*, pp. 202–231, 1995.
- [31] M. M. Chemers, L.-t. Hu, and B. F. Garcia, "Academic self-efficacy and first year college student performance and adjustment." *Journal of Educational psychology*, vol. 93, no. 1, p. 55, 2001.
- [32] J. Hattie and H. Timperley, "The power of feedback," *Review of Educational Research*, vol. 77, no. 1, pp. 81–112, 2007. [Online]. Available: <http://dx.doi.org/10.3102/003465430298487>
- [33] P. Winne and D. Butler, "Student cognition in learning from teaching," *International encyclopedia of education*, vol. 2, pp. 5738–5775, 1994.
- [34] C. R. Rodgers, "Attending to student voice: The impact of descriptive feedback on learning and teaching," *Curriculum Inquiry*, vol. 36, no. 2, pp. 209–237, 2006.
- [35] J. Hattie, "Influences on student learning," *Inaugural lecture given on August*, vol. 2, p. 1999, 1999.
- [36] M. Csikszentmihalyi, *Beyond boredom and anxiety*. Jossey-Bass, 1975.
- [37] J. Webster, L. K. Trevino, and L. Ryan, "The dimensionality and correlates of flow in human-computer interactions," *Computers in Human Behavior*, vol. 9, no. 4, pp. 411 – 426, 1993. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/074756329390032N>
- [38] K. Kiili, "Digital game-based learning: Towards an experiential gaming model," *The Internet and Higher Education*, vol. 8, no. 1, pp. 13 – 24, 2005. [Online]. Available: <http://www.sciencedirect.com/science/article/pii/S1096751604000776>
- [39] L. A. Custodero, "Seeking challenge, finding skill: Flow experience and music education," *Arts Education Policy Review*, vol. 103, no. 3, pp. 3–9, 2002.
- [40] E. Lee, "The relationship of motivation and flow experience to academic procrastination in university students," *The Journal of Genetic Psychology*, vol. 166, no. 1, pp. 5–15, 2005, PMID: 15782675. [Online]. Available: <http://dx.doi.org/10.3200/GNTP.166.1.5-15>
- [41] S. Abuhamdeh, J. Nakamura, and M. Csikszentmihalyi, "Flow," 2005.
- [42] M. G. Hewson and M. L. Little, "Giving feedback in medical education," *Journal of general internal Medicine*, vol. 13, no. 2, pp. 111–116, 1998.
- [43] D. Nicol, "From monologue to dialogue: improving written feedback processes in mass higher education," *Assessment & Evaluation in Higher Education*, vol. 35, no. 5, pp. 501–517, 2010. [Online]. Available: <http://dx.doi.org/10.1080/02602931003786559>

- [44] R. L. Bangert-Drowns, C.-L. C. Kulik, J. A. Kulik, and M. Morgan, "The instructional effect of feedback in test-like events," *Review of Educational Research*, vol. 61, no. 2, pp. 213–238, 1991. [Online]. Available: <http://dx.doi.org/10.3102/00346543061002213>
- [45] P. T. Sturges, "Information delay and retention: Effect of information in feedback and tests." *Journal of Educational Psychology*, vol. 63, no. 1, p. 32, 1972.
- [46] R. Singh, S. Gulwani, and A. Solar-Lezama, "Automated feedback generation for introductory programming assignments," in *ACM SIGPLAN Notices*, vol. 48, no. 6. ACM, 2013, pp. 15–26.
- [47] *American Heritage Dictionary of the English Language*, 5th ed. Houghton Mifflin Harcourt Publishing Company, 2016.
- [48] P. Shankar, "The evolution of compilers," *Resonance*, vol. 12, no. 8, pp. 8–26, 2007. [Online]. Available: <http://dx.doi.org/10.1007/s12045-007-0080-8>
- [49] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky, "The stanford corenlp natural language processing toolkit." in *ACL (System Demonstrations)*, 2014, pp. 55–60.
- [50] W. G. Lehnert and M. H. Ringle, *Strategies for natural language processing*. Psychology Press, 2014.
- [51] J. Ba, V. Mnih, and K. Kavukcuoglu, "Multiple object recognition with visual attention," *arXiv preprint arXiv:1412.7755*, 2014.
- [52] P. Doetsch, M. Kozielski, and H. Ney, "Fast and robust training of recurrent neural networks for offline handwriting recognition," in *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*. IEEE, 2014, pp. 279–284.
- [53] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.
- [54] A. V. Aho, M. S. Lam, J. D. Ullman, and R. Sethi, *Compilers: Principles, Techniques, and Tools*. Pearson, 2011.
- [55] A. N. Rafferty, E. Gerard, K. W. McElhaney, and M. C. Linn, "Automating guidance for students' chemistry drawings." in *AIED Workshops*, 2013.
- [56] M. Papastergiou, "Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation," *Computers & Education*, vol. 52, no. 1, pp. 1–12, 2009.
- [57] A. T. Corbett and J. R. Anderson, "Locus of feedback control in computer-based tutoring: Impact on learning rate, achievement and attitudes," in *Proceedings of the SIGCHI conference on Human factors in computing systems*. ACM, 2001, pp. 245–252.
- [58] A. B. Bondi, "Characteristics of scalability and their impact on performance," in *Proceedings of the 2nd international workshop on Software and performance*. ACM, 2000, pp. 195–203.
- [59] R. Tewari, "Architectures and algorithms for scalable wide-area information systems," 1998.

- [60] M. K. AL, M. S. TK, M. Kothiwale, and K. SS, “Real time and distributed computing systems.”
- [61] R. T. Fielding and R. N. Taylor, “Principled design of the modern web architecture,” *ACM Trans. Internet Technol.*, vol. 2, no. 2, pp. 115–150, May 2002. [Online]. Available: <http://doi.acm.org/10.1145/514183.514185>
- [62] B. Wescott, *Every Computer Performance Book*. CreateSpace Independent Publishing Platform, 2013.
- [63] Z. Xiao, W. Song, and Q. Chen, “Dynamic resource allocation using virtual machines for cloud computing environment,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1107–1117, June 2013.
- [64] P. Baudin, “Frama-c.”
- [65] T. Littlefair, “Cccc - c and c++ code counter.”
- [66] P. Louridas, “Static code analysis,” *IEEE Software*, vol. 23, no. 4, pp. 58–61, July 2006.
- [67] T. J. Maurer and H. R. Pierce, “A comparison of likert scale and traditional measures of self-efficacy.” *Journal of applied psychology*, vol. 83, no. 2, p. 324, 1998.



## Appendix A

# Participant Survey

An electronic survey was given to all participants via their school email addresses. This appendix lists every question on the survey.

**Page 1:** Questions on page one asked the respondent to rate the extent to which they agree with the following statements. Responses were selection boxes, with the following options: Strongly agree, Agree, Neither agree or disagree, Disagree, Strongly disagree, Not Applicable.

1. The IFS system was easy to use.
2. The information about my code provided by IFS was useful.
3. I used IFS for most of my C coding assignments this semester.
4. I used IFS to locate the code associated with the error or warning message.
5. I used IFS to find out what error and warning messages meant.
6. I used IFS to find out common causes of error and warning messages.
7. IFS helped me find solutions to errors in my code.

**Page 2a:** Questions on the first part of page two asked the respondent how often they have completed an activity in the context of learning to program during the current semester. Responses were selection boxes, with the following options: Very often, Often, Sometimes, Rarely, Never. The questions on this part of the page included the following:

1. Asked questions in class or contributed to class discussions (current semester).
2. Participated in forums online or contributed to online discussions (current semester).
3. Worked with classmates outside of class to prepare class assignments (current semester).
4. Discussed grades or assignments with the instructor or TA (current semester).
5. Skipped class (current semester).
6. Took detailed notes during class (current semester).
7. Applied material learned in a class to other areas (current semester).
8. Missed deadlines given for assigned work (current semester).

**Page 2b:** On the second part of page two, respondents were asked: *This semester, how many hours each week did you spend studying or doing assigned work outside of class?* Responses were selection boxes, with the following options: Zero, Less than one hour per week, 1 to 3 hours, 4 to 7 hours, 8 to 14 hours, More than 14 hours per week,

**Page 3:** Page three asked the same questions and used the same response selections as page two, with responses relating to the previous semesters. Each question emphasized that it was specifically for prior semesters, and the current semester should be excluded.

**Page 4:** Questions on page 4 asked the respondent to rate the extent to which they agree with the following statements. Responses were selection boxes, with the following options: Strongly agree, Agree, Neither agree or disagree, Disagree, Strongly disagree, Not Applicable.

1. I give up on tasks/assignments/programs before completing them
2. If a task/program/assignment looks too complicated, I will not even bother to try it
3. When I get a task/assignment to do, I go right to work on it
4. I avoid trying to learn new ways of programming when they look too difficult for me
5. Failure at tasks/assignments/programming just makes me try it harder
6. I feel insecure about my ability to do programming
7. I am a self-reliant person and I take full advantage of it when learning new programming concepts
8. I do not seem capable of dealing with most problems that come up while programming

**Page 5:** Page 5 consisted of three questions, each with free text responses. Each question is listed below:

1. Do you have any suggestions for improving our IFS system?
2. What semester of your university program are you in as of March 2016?
3. Do you have any other comments or information you would like to share?

## Appendix B

# Ethics Documents

This appendix contains all documents related to ethics, including the ethics application for IFS (not applicable pages excluded), the data privacy plan, and the secondary data use application. The ethics certificate number for this project is 16MY003.

### B.1 Ethics Application for IFS

## SECTION A: ADMINISTRATIVE INFORMATION



**A.1 Title of the research project:** Use of immediate feedback by novice programmers

### A.2 Investigator Information




Note that in the case of student research, the Principal Investigator is the faculty advisor for the purposes of this submission.

| Name & position | Principal Investigator | Faculty Co-Investigator             | Student Investigator     | Other Investigator       | Department                 | Phone No. | E-Mail               | Receives Communications                 |
|-----------------|------------------------|-------------------------------------|--------------------------|--------------------------|----------------------------|-----------|----------------------|---|
| Judi McCuaig    | Yes                    |                                     |                          |                          | School of Computer Science | 58534     | judi@uoguelph.ca     | Yes                                     |
| Justin Carvalho | n/a                    | <input type="checkbox"/>            | X                        | <input type="checkbox"/> | School of Computer Science |           | carvalhj@uoguelph.ca | <input checked="" type="checkbox"/> Yes |
| Dan Gillis      | n/a                    | <input checked="" type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | School of Computer Science | 52951     | dgillis@uoguelph.ca  | <input checked="" type="checkbox"/> Yes |
|                 | n/a                    | <input type="checkbox"/>            | <input type="checkbox"/> | <input type="checkbox"/> |                            |           |                      | <input type="checkbox"/> Yes            |
|                 | n/a                    | <input type="checkbox"/>            | <input type="checkbox"/> | <input type="checkbox"/> |                            |           |                      | <input type="checkbox"/> Yes            |
|                 | n/a                    | <input type="checkbox"/>            | <input type="checkbox"/> | <input type="checkbox"/> |                            |           |                      | <input type="checkbox"/> Yes            |
|                 | n/a                    | <input type="checkbox"/>            | <input type="checkbox"/> | <input type="checkbox"/> |                            |           |                      | <input type="checkbox"/> Yes            |


|  |     |                          |                          |                          |  |  |  |                                 |
|--|-----|--------------------------|--------------------------|--------------------------|--|--|--|---------------------------------|
|  | n/a | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> |  |  |  | <input type="checkbox"/><br>Yes |
|--|-----|--------------------------|--------------------------|--------------------------|--|--|--|---------------------------------|

Provide name of person who completed this form: Judi McCuaig

**A.3** Are there any issues or concerns regarding the timeline for approval that you would like to raise? ☒ N/A 

|  |
|--|
|  |
|--|

**A.4** Research Ethics Approval (other than University of Guelph)

**A.4.1** Will any other Research Ethics Board be asked for approval? ☐ Yes ☒ No 


If **YES**, please specify:

|  |
|--|
|  |
|--|

Copy of the clearance certificate or approval will be provided to the REB when available

☐ Yes


☐ Attached

**A.4.2** If you are undertaking research in a country other than Canada, submit a copy of the clearance certificate/approval from the Research Ethics Board in that country. 

☐ Attached

**OR** discuss what alternative measures are being taken (see information guide):

|  |
|--|
|  |
|--|

**A.5 Level of the Project:** please check all that apply 

|                                     |                  |
|-------------------------------------|------------------|
| <input checked="" type="checkbox"/> | Faculty Research |
| <input type="checkbox"/>            | PhD Thesis       |

|                                     |                               |
|-------------------------------------|-------------------------------|
| <input checked="" type="checkbox"/> | Master's Thesis               |
| <input type="checkbox"/>            | Masters Major Research Paper  |
| <input type="checkbox"/>            | M.Sc by Coursework            |
| <input type="checkbox"/>            | Undergraduate                 |
| <input type="checkbox"/>            | Honours Thesis                |
| <input type="checkbox"/>            | Class Project Specify course: |
| <input type="checkbox"/>            | Internship                    |
| <input type="checkbox"/>            | Practicum                     |
| <input type="checkbox"/>            | Independent Study             |
| <input type="checkbox"/>            | Administration                |
| <input type="checkbox"/>            | Contract – for profit sponsor |
| <input type="checkbox"/>            | Other – please specify:       |

**A.6 Funding of Project****A.6.1** Has funding been granted for this project?☐ Yes ☒ No☐ Pending**A.6.2** Agency or Sponsor

| Funder | Name of Program | Title of Grant | Grant Number |
|--------|-----------------|----------------|--------------|
|        |                 |                |              |
|        |                 |                |              |
|        |                 |                |              |
|        |                 |                |              |
|        |                 |                |              |

Comments:/funder

**A.6.3 Contract** – will there be an agreement with a research partner/funder (i.e. data sharing agreements, research funding agreements, confidentiality agreements etc.)?☒ N/A

Name of Research Partner/Sponsor:

|  |                              |
|--|------------------------------|
| Title of Research Project:   |                              |
| Has a copy of the contract been submitted to the Contracts Department of the Office of Research? | <input type="checkbox"/> Yes |
| Has the contract received final signatures?  | <input type="checkbox"/> Yes |
| Comments:  |                              |

**A.7 Peer Review**

|  |   |  |
|--|---|--|
| <b>A.7.1</b> Has this project undergone peer review for scholarly merit during the course of funding approval? | <input type="checkbox"/> Yes            | <input checked="" type="checkbox"/> No |
| <b>A.7.2</b> Has this project undergone peer review for scholarly merit by a graduate advisory committee?      | <input checked="" type="checkbox"/> Yes | <input type="checkbox"/> No            |
| <b>A.7.3</b> Comments:   |   |  |

**A.8 Disclosure of Conflict of Interest**

**A.8.1:** Will the researcher(s), members of the research team, and/or their partners or immediate family members receive any personal benefits This might include a financial benefit such as remuneration/income, intellectual property rights, rights of employment, consultancies, board membership, share ownership, stock options etc. ☐ Yes ☒ No

If YES, please describe the benefits below. Include details of all fees and/or honoraria directly related to this study, such as those for participant recruitment, advice on study design, presentation of results, or conference expenses.

|  |
|--|
|  |
|--|


**A.8.2** Describe any restrictions regarding access to or disclosure of information (during or at the end of the study) placed on the investigator(s), including those related to the publication of results. Note the nature of these restrictions and who is applying these restrictions. ☒ N/A

|  |
|--|
|  |
|--|


**A.8.3** Describe the possibility of commercialization of the research findings. ☐ N/A


|   |
|---|
| The software used by students to produce the data discussed in this proposal is novel and was built by a Guelph graduate student. If the research proposed here shows that students |
|---|

benefited from the software, there is a possibility of collaborating with a commercial partner to create a new version of the software. The data generated by the software as it was used in 2016 will not be shared with any partner nor will it comprise any portion of the new version of the software.


**A.8.4** Describe any personal or professional relationship between a member of the research team and any participants aside from the researcher/participant relationship. ☐ N/A 

Justin Carvalho taught CIS\*2500 in the Winter 2016 semester, and students from that course, as well as two other courses that Justin did not teach, were permitted to use the IFS system that generated the log files and data proposed for use in this research. Dr. Judi McCuaig will be the only person who handles the identified data, as she did not teach any of the students who used the system in 2016. Additionally, the research associated with this data will not begin until the S16 semester after all W16 grades have been submitted.

**A.8.5** Disclose any employment that research team members have outside the University of Guelph, if it is related in any way to the study (e.g. as the source of research participants.) ☒ N/A 

**A.8.6** Describe any consultancy or other contractual agreements, financial, partnership, or business interests within the last two years that might be perceived as a conflict of interest pertaining to this study. ☒ N/A 

## **A.9 Experience and Licensed Qualifications**

**A.9.1** What experience does the principal investigator have with the kind of research undertaken in this project and in this context, including the nature of the participants, methods of data collection, etc.? 

Dr. McCuaig has been investigating the effects of student engagement and self efficacy on success in university courses for the past five years. This study is part of the ongoing investigation. She has undertaken several studies using datamining techniques on data captured from student activity in courses. Dan Gillis is a statistician and computer science researcher who is looking at methods of collecting and analyzing data collected via interactions with online systems.



**A.9.2** What is the role of each member of the research team?



| Name            | Role                 | Contact with Identified Data            | Direct Participant Contact              | CORE Tutorial Completed                 |
|-----------------|----------------------|---|---|---|
| Judi McCuaig    | Primary Investigator | <input checked="" type="checkbox"/> Yes | <input checked="" type="checkbox"/> Yes | <input checked="" type="checkbox"/> Yes |
| Justin Carvalho | Student Researcher   | <input checked="" type="checkbox"/> Yes | <input type="checkbox"/> Yes            | <input type="checkbox"/> Yes            |
| Dan Gillis      | Co-Investigator      | <input checked="" type="checkbox"/> Yes | <input type="checkbox"/> Yes            | <input checked="" type="checkbox"/> Yes |
|                 |                      | <input type="checkbox"/> Yes            | <input type="checkbox"/> Yes            | <input type="checkbox"/> Yes            |

**A.9.3** How will the faculty with principal responsibility ensure that each team member has the expertise and experience necessary to carry out the research? How will s/he ensure that all team members are familiar with the contents of the ethics protocol? **Discuss for each team member.**




| Name            | Review confidentiality requirements | Review recruitment process          | Review consent process              | Direct oversight of procedure/process | Review of debrief post deception | Completion of graduate level method's course | Other                               | Details:   |
|-----------------|-------------------------------------|-------------------------------------|-------------------------------------|---------------------------------------|----------------------------------|--|-------------------------------------|--|
| Judi McCuaig    | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/>   | <input type="checkbox"/>         | <input checked="" type="checkbox"/>          | <input type="checkbox"/>            |  |
| Justin Carvalho | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input checked="" type="checkbox"/>   | <input type="checkbox"/>         | <input checked="" type="checkbox"/>          | <input type="checkbox"/>            | Justin's work will be reviewed weekly and examined to ensure protocol is being followed. |
| Dan Gillis      | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input type="checkbox"/>            | <input checked="" type="checkbox"/>   | <input type="checkbox"/>         | <input checked="" type="checkbox"/>          | <input checked="" type="checkbox"/> | Dan has extensive experience working with data and will be a huge benefit to this study  |
|                 | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>            | <input type="checkbox"/>              | <input type="checkbox"/>         | <input type="checkbox"/>                     | <input type="checkbox"/>            |  |

**D.4.4** Which member of the research team will administer consent?

☐ N/A 

**D.4.5** Has this individual had the necessary training to administer consent? Describe the training received or planned.

☐ Yes ☐ No ☐ N/A 

**D.4.6** Verify the following:




|                              |                              |  |         |
|------------------------------|------------------------------|--|---------|
| <input type="checkbox"/> Yes | <input type="checkbox"/> N/A | Copy of information letter will be available to participant (provide PRINT button for online survey) | Comment |
| <input type="checkbox"/> Yes | <input type="checkbox"/> N/A | Copy of information letter shows University of Guelph letterhead or logo                             | Comment |

**D.4.7** Will the participant have an opportunity to have questions about the project and their role as a participant answered? How will this opportunity be communicated to them?

☐ N/A 

We will publish the results of the study on the School of Computing website and invite students who took the course in the past to direct questions to the principle researcher.

**D.5 Proxy Consent:** Will you be obtaining **proxy consent** (e.g. of parent/guardian)?

☐ Yes ☐ NO 


If **NO**, please go to D.6

If **YES** to proxy consent please answer the following:

**D.5.1** Why is proxy consent necessary?



**D.5.2** How will competence of the participant be established, and who will determine this?

☐ N/A 

|                              |                              |   |         |
|------------------------------|------------------------------|---|---------|
| <input type="checkbox"/> Yes | <input type="checkbox"/> N/A | If participant declines to sign the second consent form, data will be removed from the study without penalty. Participant will still receive any incentives or reimbursement due. | Comment |
| <input type="checkbox"/> Yes | <input type="checkbox"/> N/A | Copy of second consent form attached to this application  | Comment |
| <input type="checkbox"/> Yes | <input type="checkbox"/> N/A | University of Guelph guideline on deception has been followed   | Comment |

**Submit each applicable document as an individual attachment with your application – do not merge the documents into one long file.**

**D.7** Is **community or institutional consent** required for your project? ☐ Yes ☐ No ☐ N/A ⓘ

Please discuss why this is required, how it will be managed.

**D.8** Will the participant be free to give consent, or refuse, without any undue influence or coercion? ☐ Yes ☐ No ⓘ

Explain any details you feel are relevant.

We are applying for a waiver of the consent process. Participants will not be coerced because they will not be contacted. Please see discussion in D.1

**D.9** How will you ensure that **consent is ongoing** throughout the project? How will you ensure that necessary information is provided to participants on an ongoing basis? ☐ N/A ⓘ

**D. 10** Discuss the likelihood that the **confidentiality** offered to participants may be **limited** by the legal obligation to “report information to authorities to protect the health, life or safety of a participant or third party” or that “a third party may seek access to information obtained and/or created in confidence in a research context” through either “voluntary disclosure” or “force of law”. [TCPS2, Article 5.1] ☐ N/A ⓘ

## SECTION E: DESCRIPTION OF THE RISKS AND BENEFITS

**E.1 Risks:** Itemize your response by each method/procedure employed during this research. ⓘ

Risk (check all that apply)

|                              |  |
|------------------------------|--|
| <input type="checkbox"/> N/A |  |
| <input type="checkbox"/> Yes | Physical (including bodily contact or administration of any substance)     |
| <input type="checkbox"/> Yes | Psychological (including feeling demeaned, embarrassed, worried, or upset) |
| <input type="checkbox"/> Yes | Social (possible loss of status or reputation)                             |
| <input type="checkbox"/> Yes | Economic (risk to livelihood or income)                                    |

**E.1.1** If you indicated **YES** to any of the above, are any of the risks indicated greater than the participant would encounter in their everyday life? ☐ Yes ☐ No

**E.1.2** For each risk identified above describe how the risk will be managed and include an explanation as to why alternative approaches could not be used. ☐ N/A ⓘ

|  |
|--|
|  |
|--|

**E.2 Possible Benefits** Describe any benefits to the participants/discipline/ society that would justify to participants why they should be involved in this study. ☐ N/A ⓘ

|                         |  |
|-------------------------|--|
| Benefits to Participant |  |
| Benefits to Discipline  | We will understand better how some instructional techniques and approaches effect student achievement. |
| Benefits to Society     | We will understand better how some instructional techniques and approaches effect student achievement. |

**E.2.1** Research results should be provided to participants where possible.

Will aggregate research results be provided to participants? ☐ Yes ☐ No ⓘ

If **YES**, explain what information will be provided to the participants upon completion of the project, and how will they receive this.

|   |
|---|
| <p>We will disseminate the results of the study via the school website and via announcements to alumni if possible. There is no way to provide results to individuals for this study because no analysis of individual participants will be done and no identifying information will be kept.</p> |
|---|

## B.2 Data Privacy Plan

## Identity protection measures

### 1. Time data

Log files from any computer system contain information about when actions occurred. For the purposes of this research all data that represents time will be converted to one of the following classes:

- a) 12:00 am – 4:00 am: **early morning**
- b) 4:01 am – 8 am: **morning**
- c) 8:01 am – 12:00 pm: **late morning**
- d) 12:01 pm – 4:00 pm: **afternoon**
- e) 4:01 pm -8: 00 pm: **evening**
- f) 8:01 pm – 11:59 pm: **night**

### 2. Location data

Web services (web sites, webmail, shopping sites, etc) routinely collect the internet address (IP address) of anyone who uses that service. That address can be used to learn the rough geographic location of the participant. For this research the IP address collected by the IFS system will be reduced to one of two values: on campus, or off campus. This abstraction will protect the identity of the participants by making it impossible to determine anything about the specific computer that the participant was using.

## Data Collected from IFS System

| Short Name            | Description   | Identity Protection   |
|-----------------------|---|---|
| Usage Count           | The number of times a participant has logged in to the system, the date/time of the login and whether the usage was from an on-campus or off-campus computer. | All time data will be recorded as belonging to one of the time classes noted above. Location data is converted from ip address as noted above.  |
| Program uploads       | The number of times a participant has attempted to upload their work to the system and information about whether or not that upload was successful.           | Participant information will be converted to a non-identifying participant ID prior to inclusion in this study.<br>All participant work will be abstracted to counts of features prior to inclusion in the data set for this study. |
| Measures and feedback | The feedback information provided to participants upon successful analysis of uploaded code as well as the date and time of that feedback.                    | This information does not pertain specifically to individuals and cannot by itself identify anyone. The participant information associated with the database  |

|                                   |  |  |
|-----------------------------------|--|--|
|                                   |  | entry will be converted to a non-identifying participant ID prior to inclusion in the study. Dates and time will be converted as described at the beginning of this document.  |
| Participant-feedback interactions | Some of the feedback information contain web hyperlinks leading to additional information. This data item is a count of which links participants followed and the time/date that the link was followed.  | The participant information associated with the database entry will be converted to a non-identifying participant ID prior to inclusion in the study. Dates and time will be converted as described at the beginning of this document. |
| Code Metrics                      | Participant uploaded source code will be analyzed to produce metrics that describe the code. Metrics include line counts, counts of the number of functions, input/output counts, cyclomatic complexity, and fan-in/fan-out. The process is similar to counting parts of speech, sentences, conjunctions, etc. from a written essay. | None of the metrics used will contain identifying information. The metrics will be connected to the participant id of the participant, but no other information about the participant will be kept as part of the metrics.             |
| Participant Experience Survey     | All participants who use the system were invited to complete a participant experience survey at the end of the semester to help us improve the system. The survey questions are attached.  | The participant information associated with the survey data will be converted to a non-identifying participant ID prior to inclusion in the study. Dates and time will be converted as described at the beginning of this document.    |

### B.3 Secondary Data Use Application



# University of Guelph Research Ethics Board (REB) Application to Involve Human Participants in Research (REB-App)

## SUPPLEMENT III SECONDARY USE OF DATA

Include this supplement only if your protocol discusses secondary use of a data set. In addition to this supplement, please fill out the REB-App, Section A, Section G and Section H.

### III.1

Is it your intention to reanalyze a data set you collected, for purposes other than described in the original application/consent form? ☐ Yes ☒ No

Is it your intention to reanalyze a data set collected by someone else? ☒ Yes ☐ No

III.2 Is it your intention to allow data to be reanalyzed by current or future colleagues, participants, or other researchers outside of the original research team? ☐ Yes ☒ No

III.3 Provide a summary below, of the research to be undertaken. Describe the purpose and background rationale for the proposed project, as well as any hypotheses and/or research question to be examined.

This proposed work is part of a larger investigation into the relationship between the quality of participant work products (their software source code) and participant self efficacy and engagement. The purpose of this study is to determine if participants' self efficacy, engagement and work product quality improves with regular access to a mechanism for obtaining immediate feedback about aspects of the quality of their work.

The Immediate Feedback System (IFS) was implemented by Justin Carvalho (an M.Sc student) to address problems that undergraduate participants have understanding the error messages and warnings provided by the traditional programming tools used at the School of Computer

Science. Over the Winter 2016 semester the IFS was made available to undergraduate participants taking SOCS courses. Participants securely submitted their partially complete programming assignments to the system and received immediate feedback about the quality of their work as measured by the system.

This proposed research is to create an anonymized data set from the log files, interactions, work product and participant feedback about the IFS system and then use that anonymized data to assess whether a relationship exists between participant use of the system and their self efficacy, engagement and work product quality.

#### III.4 Describe in clear and concise detail the data you will be working with, and provide a general discussion of the analysis you will undertake.

The raw data consists of log files and text files submitted by participants as they used the IFS system as well as the results of a survey that solicited feedback about the system at the end of the semester. The survey was accomplished using the university qualtrics installation. In total 277 participants used the system over the semester and 115 students provided feedback via the survey.

The data proposed for use in this research is logged as part of the normal operation of the IFS system. The data contains the central login id of the participant, the date and time of use, the information provided to the system by the participant, the system operations chosen by the participant, and the results of those operations.

The data from the qualtrics survey contains the participant's central login id as well as her/his answers to the survey questions (survey questions provided as an attachment).

The proposed work will create an anonymized data set using these two sets of raw data. The data will be linked using the central login id, which will be replaced by a randomized participant id as soon as the link has been accomplished. At no point will participant names, participant numbers or any directly identifying information be used in the analysis. Only Dr. McCuaig will have access to the data with central login identifiers and she will destroy the identified data as soon as the linking is done and the anonymized participant ids have been assigned. The data to be included in the data set is described in more detail in the attached data plan.

#### Analysis:

The data from the IFS system and the participant experience survey will be processed to remove the central login id that could identify participants. Each participant will be assigned a unique participant ID that cannot be traced back to that participant and the data from the survey and IFS system will be encoded to use only the participant ID when preserving relationships between pieces of data.

The resulting dataset will be analyzed using statistical data mining procedures such as clustering and classification to determine the absence or presence of relationship between the self reported answers in the survey, use of the IFS system, and characteristics of the work

uploaded to the IFS system by the participants. We hope to identify the relationship (if any) of use of the IFS system with improvement in work product submitted and higher self efficacy. In particular we are interested in identifying characteristics of participants who may have demonstrated weaker skills (via their work product) at the beginning of the semester and determining if their use of the IFS system is related to the degree of improvement seen in their work with the system over the semester.

### III.5 What would the impact be on the participant should **privacy** be **breached**?

If privacy were breached the usage history of the IFS system could be known as could the answers to the survey questions. Because the data will be anonymized with a participant ID, this information cannot be connected to the actual individual represented by the data.

This data contain no evaluations, grades, or personal comments and is entirely generated by the software so even if there were a breach of privacy, no details about the individual could be exposed.

The metrics associated with the practice exercises that a participant uploaded to the system could become visible to others if privacy were breached, but their actual work product is not part of the data for this research. The metrics could not be used to identify the individual.

Because the IFS data is computer generated, the only identifier it contains is the central login id of the participant and those will be removed during data cleaning- prior to any sort of analysis. It is possible that the work submitted for analysis to the IFS system will contain identifying information, and because of that the work itself is not being used for analysis. Instead we intend to use counts and metrics that are derived from the work submitted. These metrics will not include any copies or words taken from the work submitted by students. Dr. McCuaig will inspect the cleaned IFS data prior to analysis to ensure that no identifying information has been transferred to the data set for analysis

The answers to the survey questions contain no identifying information about the individual participants so a privacy breach of this data would not have an impact on the participant as the answers are not specific to any individual. There is a free-text answer field in the survey that Dr. McCuaig will manually go through and remove any identifying information that may be part of participants' answers. Removed information will be replaced with a random string of characters so that word counts are not affected.

### III.6 Possible Benefits Describe any benefits to your discipline or society that would justify the risk to the privacy of the original participants who generated this dataset. ☐ N/A

|                        |   |
|------------------------|---|
| Benefits to Discipline | The increase in computer science class sizes at the post secondary level has made it nearly impossible for instructors and teaching assistants to provide one on one assistance for novice programmers. Further challenge is found because many undergraduate participants commute to their course and are unavailable to come to office hours or |
|------------------------|---|

|                     |  |
|---------------------|--|
|                     | tutoring sessions. Systems like the IFS system can mitigate these difficulties by providing ways for participants to get immediate feedback at any time. This research will help us improve the design of the system to make it more effective.  |
| Benefits to Society | <p>Engaged participants are desirable in all disciplines. The findings from this work can be applied to other disciplines and domains and may even help participants become more self-regulated. If this research is successful we intend to pilot a similar system in a non-science discipline.</p> <p>We intend to use this research to help students understand the importance of self-motivation. We will host a seminar in which we present the early analysis of the data collected and will invite every SOCS participant. The invitation will be sent to all SOCS students, not just to participants who used the IFS system. During this seminar we will enumerate which data were used, how they were used, and what we have learned from the research. If participants can begin to understand how to help themselves, and how to stay engaged with learning, they will be much more successful in post secondary education, and in their subsequent work life.</p> |

**III.7** Does this data set contain identifiers of any kind?

☒ Yes ☐ No

If **NO**: **REB review is not required for secondary use of anonymous data but is required for data which can be linked (coded data) or anonymized data.**

Describe the type of data you will be using:

|                              |                                     |
|------------------------------|-------------------------------------|
| Type of Data: participant ID |                                     |
| Identified                   | <input type="checkbox"/>            |
| De-identified                | <input type="checkbox"/>            |
| Coded                        | <input type="checkbox"/>            |
| Anonymized                   | <input checked="" type="checkbox"/> |
| Anonymous                    | <input type="checkbox"/>            |

**For definitions see TCPS2; Chapter 5; P.56;**

**[http://www.ethics.gc.ca/pdf/eng/tcps2/TCPS\\_2\\_FINAL\\_Web.pdf](http://www.ethics.gc.ca/pdf/eng/tcps2/TCPS_2_FINAL_Web.pdf)**

Describe:

The data set that is analyzed will contain only anonymized identifiers. The raw data does contain the central login id but that will be removed before the data is used.

**III.8** Have participants from whom data was generated given permission for secondary use in the type of project you propose?

☐ Yes ☒ No

If **YES**: Attach a sample copy of the consent form used to obtain this permission.

☐ Attached

Discuss:

If **NO**, you are applying for secondary use of non-anonymous data without prior consent.

Verify the following:

|   |                              |  |  |
|---|------------------------------|--|--|
| <input checked="" type="checkbox"/> Yes | <input type="checkbox"/> N/A | Identifiable information is essential to the researcher  | We cannot conduct this research without identifying the multiple interactions of an individual as belonging to a single individual. The identity of that individual is unimportant, but we must keep all of the data from each individual separate from the other participants.  |
| <input checked="" type="checkbox"/> Yes | <input type="checkbox"/> N/A | The use of identifiable information without the participants' consent is unlikely to adversely affect the welfare of individuals to whom the information relates | There is practically no risk at all to participants in this research. The data used for this study will be obtained by anonymizing data in the log files of the instructional software used by the participants. The raw data will be pre-processed to ensure that identification of individuals within the study data is impossible. The data processing and management plan is provided as an attached document. |
| <input checked="" type="checkbox"/> Yes | <input type="checkbox"/> N/A | The researchers will comply with any known preferences previously expressed by individuals about any use of their information                                    | Comment  |
| <input checked="" type="checkbox"/> Yes | <input type="checkbox"/> N/A | The researchers will take appropriate measures to protect the privacy of individuals, and to safeguard the identifiable information                              | The plans for protecting the privacy of participants is noted in the data management document  |
| <input checked="" type="checkbox"/> Yes | <input type="checkbox"/> N/A | It is impossible or impracticable to seek consent from individuals to whom the information relates   | Participants are gone for the summer and there is no way to reliably contact them as they frequently do not read email during the summer.<br><br>We are studying the effect of engaging instructional practices on the self-efficacy and engagement of participants. When consent is   |

|   |                              |   |   |
|---|------------------------------|---|---|
|   |                              |   | sought from undergraduate participants, it is the already-engaged participants who give consent, and those with less self efficacy and engagement will simply ignore the request for consent. The results of this study will be unpublishable if the data used is limited to participants who are already engaged in their studies because we will be unable to reach any conclusions that can be generalized to the target demographic: disengaged participants. |
| <input checked="" type="checkbox"/> Yes | <input type="checkbox"/> N/A | The researchers have obtained any other necessary permission for secondary use of information for research purposes | Comment   |

**All of the above must be true for the REB to consider approving the research without requiring consent from the individuals to whom the information relates.**

**III.9** Who might have access to this data set while it is in the possession of the research team?

☐ N/A

Only the research team will have access. Only Dr. McCuaig will have access to the raw data that contains central login ids and even she will be unable to identify individuals from the data set because no mapping will exist between the identified log files and the anonymized data set.

**III.10** How will data security be established while the data set is in the possession of the research team?

☐ N/A

Describe:

The data will be located on a password protected server, physically located on the University Campus. The server is managed by a professional system administrator and is up to date with modern security measures.

**III.11** How long will this data set remain in the possession of the research team and how will it be disposed of after use?

☐ N/A

Describe:

This is a large, rich data set and this initial investigation is likely to suggest additional analyses that might be fruitful. We would like to retain the data for 5 years with the understanding that additional ethics approval is required for any additional analysis.

**III.12** Do you intend to link the data set with any other set of data?

☐ Yes ☒ No

Describe:

We do not intend to link the data produced by this investigation to any other data because this data will be anonymized and any connection with other data sets will be impossible.