

B.Comp.Dissertation

**"Online Judge" for
Data Structures and Algorithms Course**

by

Rose Marie Tan Zhao Yun

Department of Computer Science
School of Computing
National University of Singapore

2013/2014

B.Comp.Dissertation

**"Online Judge" for
Data Structures and Algorithms Course**

by

Rose Marie Tan Zhao Yun

Department of Computer Science
School of Computing
National University of Singapore

2013/2014

Project Number: H160090

Advisor: Dr. Steven Halim

Deliverables:

Report: 1 Volume

Abstract

VisuAlgo (currently hosted at <http://algorithmics.comp.nus.edu.sg/~onlinequiz/staging/>) is an online platform for learning about various data structures and algorithms through a series of visualisations and a newly developed “Online Judge” system that uses automated testing. This report describes the project in greater detail, focusing on the design of the user interface for the new test component as well as the redesign of the interface and user experience for the original visualisation component. It describes the process taken to accomplish these tasks, from defining the requirements to full development, with emphasis on refinement through prototyping. The evolution of the interface design at each stage in development is demonstrated through the various user workflow patterns and prototype screens. Finally, the report concludes with an evaluation of the end product against common interface design heuristics and recommendations for future development of the overall project.

Subject Descriptors:

Human-centered computing ~ User interface design
Human-centered computing ~ User interface programming
Human-centered computing ~ Web-based interaction
Applied computing ~ E-learning
Data Structures and Algorithms

Keywords:

User interface, user experience, web development, education, visualization, data structures and algorithms

Implementation software:

HTML5, CSS3, JavaScript 1.7, PHP 5.3.3, MySQL

Acknowledgements

I would like to express my sincere gratitude to Dr. Steven Halim, my project supervisor, for his constant valuable insight and guidance throughout the whole process, his great enthusiasm for using and advocating the use of the system, and his contribution to the project even at a development level. I would also like to thank my fellow project members for their valuable contributions: Ivan Reinaldo for developing the graph library for the visualisation tool as well as the system architecture for the “Online Judge” tool, and Nguyen Hoang Duy for his contributions especially to the graph drawing component. In addition, I would like to extend special thanks to all past developers of the visualisation tool for laying the foundations for this project, and the class of CS2010 (NUS AY13/14 Semester 1) for their valuable feedback. Finally, I wish to thank my family and friends for their unwavering support in the past year.

List of Figures

Figure 1: User workflow for visualisation component	15
Figure 2: Initial homepage design	16
Figure 3: Initial visualisation page design	17
Figure 4: Colour palette used	17
Figure 5: New home page design	18
Figure 6: New visualisation page design	18
Figure 7: Fully developed visualisation page	19
Figure 8: Final playback controls close-up	20
Figure 9: File includes and page relationships for the visualisation component	21
Figure 10: User workflow for Online Quiz 1 test page	24
Figure 11: Online Quiz 1 login screen	25
Figure 12: Online Quiz 1 test screen	26
Figure 13: Online Quiz 1 result page	26
Figure 14: User workflow for Online Quiz 1 answer key page	27
Figure 15: User workflow for Online Quiz 2 test page	29
Figure 16: Online Quiz 2 test page - interface modifications and additions	30
Figure 17: Online Quiz 2 answer page - interface modifications and additions	30
Figure 18: User workflow for training mode page	31
Figure 19: Training mode topic selection screen	32
Figure 20: Training mode test screen	32
Figure 21: Training mode result screen	33
Figure 22: Training mode answer screen	33
Figure 23: Test mode login screen	34
Figure 24: Test mode instruction screen	34
Figure 25: Test mode test screen	35
Figure 26: Test mode result screen	35
Figure 27: User workflow for answer key page	36
Figure 28: Answer key login screen	36
Figure 29: Answer key answer screen	37
Figure 30: Scoreboard page	37
Figure 31: User workflow for admin control panel page	38
Figure 32: Control panel login screen	39
Figure 33: Control panel scoreboard screen	39
Figure 34: Control panel test settings screen	39
Figure 35: Home page with links to “Online Judge” component	40

Table of Contents

Title Abstract	i ii iii
Acknowledgement	iv
List of Figures	v
1 Introduction	1
2 Project Background	2
2.1 Motivation	2
2.2 Development History	2
3 Project Objectives	4
3.1 Overall Project Objectives	4
3.2 Individual Project Objectives – My Role in this Project	6
4 Literature Review	7
4.1 The Original Project	7
4.2 Similar Algorithm Visualisation and Assessment Projects	8
4.3 Online Learning and Testing	8
4.4 UI Design	10
5 Design and Development Methodology	11
6 Design and Development of the Visualisation Component	13
6.1 System Requirements	13
6.2 User Workflow and Design Mockups	15
6.3 Prototyping, Refinement, and Full Development	19
7 Design and Development of the “Online Judge” Component	21
7.1 System Requirements	21
7.2 Preliminary Workflows, Designs, Prototyping and Refinement	23
7.3 Final Workflow, Design and Full Development	30
8 Conclusion	41
8.1 Evaluation	41
8.2 Recommendations for Future Development	45
References	vii
Appendix A: Selected Feedback from Online Quiz 1	ix
Appendix B: Selected CS2010 Teaching Feedback	xiii
Appendix C: Internal Interfaces for All Input Types	xiv
Appendix D: Internal Interfaces for All Answer States in Answer Key	xvi

1 Introduction

This report describes my contributions to the larger joint project currently known as VisuAlgo. The project's name, VisuAlgo, aims to capture its purpose and essence succinctly – visualising data structures and algorithms. There are two parts to the project – a teaching component and a testing component, the former comprising a set of interactive visualisations of various data structures and algorithms, and the latter a set of automatically generated questions on those data structures and algorithms, for self-practice as well as actual tests.

The teaching component of the project, also hereafter referred to as the visualisation component, began development in 2011, and the collection of visualisations was already fairly extensive before I joined the project. During the past year, the system has undergone a major redesign, both in terms of interface as well as underlying architecture. Currently, the full system consists of a home page with links to both components:

Visualisation Component

Visualisations using the new User Interface (UI) and Scalable Vector Graphics (SVG) animation – Binary Search Tree/AVL Tree, Binary Heap, Union Find, Bitmask, Minimum Spanning Tree, Single-Source Shortest Paths, Suffix Tree, Suffix Array, and Geometry, as well as visualisations not yet updated to use the new interface and still using HTML5 Canvas – Sorting, Linked List, Stack, Queue, Recursion, Graphs, Graph Traversal, Segment Tree, Binary Indexed Tree, Max Flow, and Graph Matching.

“Online Judge” Component

A **Training** page for students' self-practice, a **Test** page for formal regulated tests, an **Answer Key** page for students to review their answers after the test, a **Scoreboard** of test results, and an **Admin Control Panel** for the instructor to control, customise and monitor the test in progress.

My main responsibility was to improve the existing interface and develop the testing component, i.e. this report's eponymous "Online Judge" system, and integrate them into a single consistent and comprehensive system, with emphasis on improving and standardising the overall user experience. Therefore, my contributions to the code base for the above pages have been the HTML and CSS for the new visualisation pages and JavaScript for their interface (not including the graph library and visualisation widgets), as well as all the HTML, CSS and JavaScript for the test component.

2 Project Background

2.1 Motivation

Dr. Steven Halim first conceptualised the visualisation project in 2011, with the intention of enhancing the teaching of various data structure and algorithm classes at the National University of Singapore (NUS). He felt that the teaching methods typically employed in these classes were lacking, and could be greatly improved by both providing smoother in-class demonstrations and by allowing students to continue exploring the algorithms in their own time on a wider range of examples.

The motivation for the second part of the project, the "Online Judge", came from repeated setting and grading of test scripts for these algorithm classes. The same basic data structure and algorithm questions have to be set and graded for each examination every semester, which is repetitive, time-consuming and uninteresting. By creating random question generators and automatic graders for these topics, much valuable time could be saved and better spent on other less trivial academic pursuits.

2.2 Development History

This project has gone through various stages of development since its conception in 2011. From 2011 to mid-2013, visualisations for many data structures and algorithms were created, from basic topics such as linked lists to classical graph algorithms such as Kruskal's minimum spanning tree algorithm, and even including rarer algorithms such as the binary

indexed tree. All of them were coded using JavaScript and HTML5 Canvas elements with a roughly similar black-and-white interface.

There were a few drawbacks to the visualisations at this stage. Firstly, there was difficulty creating smooth animations using HTML5 Canvas elements, which hampered the effectiveness of many of the visualisations, especially when consecutive animation states were very different from each other. Without smooth transitions, students were liable to get confused easily at these steps, not being able to tell clearly which graph nodes or edges were changing and why. Secondly, the method used for animation playback was inefficient, causing the playback to lag often. Finally, the system's user interface (UI) was problematic, with awkward menu sequences, inconsistencies between the various visualisations, and an unattractive overall look-and-feel.

Prompted by the weaknesses of the existing system, it was decided in June 2013 that in order to improve on the algorithm animations, all existing and future visualisations would be coded using Scalable Vector Graphics (SVG) elements instead of HTML5 Canvas. The D3.js JavaScript library was selected to be used for this, as it allows for smooth animations and facilitates SVG object manipulation. It was also decided that the entire existing system would undergo a complete UI and User Experience (UX) redesign in order to make it easier and more enjoyable for students to use. Therefore, from June 2013 to November 2013, the focus was on converting the old HTML5 Canvas visualisations into SVG-based visualisations and incorporating them into the new UI design.

Development of the "Online Judge" system also began during this period. While the back-end random question generator and grader was not developed yet at this point, the UI for the test system was designed, developed, tested and refined in late 2013. From December 2013 onward, the focus then shifted to fully automating the testing component, from coding the question generators and graders to automating the UI to handle different kinds of input. The two components of the project (visualisations and "Online Judge") were integrated, and admin controls were added to enable the instructor to customise and control the test settings.

3 Project Objectives

3.1 Overall Project Objectives

3.1.1 *To Improve Teaching Methods*

When teaching data structures and algorithms, it is necessary for the instructor to illustrate at least one example of how the algorithm works. This has traditionally been done by either drawing them on a board or with a prepared set of presentation slides. Both methods have drawbacks – hand-drawn examples are slow and error-prone, and state transitions cannot be explained clearly without smooth animations, and while animations are possible with slides, preparing animated examples in a presentation is tedious. In both cases, a limited number of examples with fixed input can be shown in class, and it is difficult to illustrate more examples spontaneously, whether by the instructor’s choice or students’ requests. The visualisation component of the project aims to overcome this problem by providing a system that allows for a large range of self-provided input, so that the instructor can both provide a larger number of examples easily and handle spontaneous examples quickly without error.

3.1.2 *To Ease Testing Procedures*

With random question generators and instant machine grading, the “Online Judge” component of the project aims to ease testing procedures by giving the instructor a quick and simple way to test students on basic algorithm concepts, saving time spent on setting and grading papers. Besides saving time, having an automated testing system also allows for asking more interesting questions. Currently, certain questions that may have multiple to infinite answers (e.g. “Draw an AVL Tree such that inserting a new value of 5 will cause 2 rotations”) are not being asked in tests, as grading these questions would be very difficult since it is tedious to verify each unique answer. However, a machine can verify each answer practically instantaneously, thus these more complex questions can be asked.¹

¹ Note that while these questions are not yet supported in this version of the system, graph drawing capabilities will be added to the “Online Judge” system in its next iteration of development.

3.1.3 To Enhance Student Learning

Ultimately, the main purpose of this project is to enhance student learning via the two aforementioned goals. With better teaching and testing procedures, student learning can be improved as well. It is important to note that according to Neil Fleming's VARK model, 60 – 65% of all people are visual learners, meaning that they learn best through pictures and other visual aids (Fleming, 2006). Thus, visualisations of data structures and algorithms would be beneficial to the majority of students. It has already been stated above that instructors do usually use visual aids to teach these topics, but the visualisation component of the project aims to greatly enhance the visual experience by offering clearer and more detailed illustrations. This is accomplished with clean and consistent visuals (e.g. graph nodes and edges), and more importantly, with animations. With animated visualisations, students can have a better understanding of how the various steps in an algorithm work. For example, animation makes AVL tree rotations much clearer than only showing the before-and-after-rotation states and leaving students to figure out which pointers changed.

Web-based tutorials have been shown to very effective for student learning (Guy, Lownes-Jackson, 2012). In particular, besides allowing the instructor to provide smoother and more varied in-class demonstrations, catering to customised input in the visualisation component also helps students directly by allowing them to try out more examples on their own at their own pace, and with their own input. Without the visualisation tool, students only have the few examples shown in class to revisit and revise. If their understanding of the topic is not strong, they may make conceptual errors without realising it when trying out examples on their own. However, with a well-designed visualisation system, students can try out their own examples, or at least a greater number of examples on the system and view the outcome of the these examples without the possibility of errors.

The training mode of the test system also helps to enhance learning by providing instant feedback through automated assessment, which is beneficial to students (Benford, Burke, Foxley and Higgins, 1994). With this system, students can test their knowledge of the various data structures and algorithms through self-practice and find out which topics they have a good grasp of and which require more revision. As the test component generates random questions and data structures, students will be assured that they can have ample practice

before the actual test. The provision of self-practice tools also helps to raise the raw average marks of students (Benford et al, 1994). In this particular case, it ensures that as many students as possible understand the basics so that focus can be given to more advanced application of the algorithms. Furthermore, providing students with a training system similar to the test system will ensure that they are familiar with the test environment before the actual assessment, so that the only difficulties a student may face doing the actual online test are content-based, not technical or usability issues.

3.2 Individual Project Objectives – My Role in this Project

3.2.1 To Improve the UI and UX of the Visualisation Component

The original visualisation component that had been developed prior to June 2013 had many drawbacks in terms of its UI and UX (see Section 4.1), which resulted in poor usage patterns by past students. Dr. Halim had already used the visualisation component in his algorithm classes, and he found through survey feedback that most students only used the website sporadically – only right after class and just before exams. Even then, not many students were using it at all.

Therefore, one of my two main responsibilities was to re-design the entire UI and UX to increase the usage of the visualisation tool, so that students would be able to fully reap its benefits. It was my responsibility to ensure that the new UI was clear, consistent, cohesive, easier to learn, and that the entire user experience was enjoyable.

3.2.2 To Design and Implement the Interface of the “Online Judge”

My other main objective in this project was to design and implement the interface for the new component of the project, the “Online Judge”. I was to be responsible for the front-end mechanism of the new testing system, from determining the system requirements to ensuring that it runs smoothly for both the students and the instructor (the person administering the test), while ensuring that it connected cohesively with the visualisation component of the project, maintaining a similar look-and-feel. This was ultimately the main objective of this project.

3.2.3 *As a Supporting Role in Other Areas*

I was also responsible for supporting my fellow project-mates in supplementing libraries when needed, whether it is coding a visualisation for the visualisation component or the question generator for a particular topic in the “Online Judge” component. This was important so as to ensure that the project did not stagnate or get stuck in a production bottleneck at any point.

4 Literature Review

4.1 The Original Project

Before I could start improving and extending the project, I had to first gain an understanding of the original existing project, what it offered, what it lacked, and how it worked. I read Dr. Halim’s short paper on this project to gain a better understanding of the project’s goals (see Section 3.1) and the research that had already been conducted, and tried using the visualisations, recording notes about my user experience.

I noted multiple issues with the UI and UX. Firstly, the overall look-and-feel was bland and unattractive, with plain black and white boxes and a simple HTML table with default styling for the home page. Secondly, there were several aspects of the interface that were confusing, such as the menu system, where clicking on a first level menu button caused the first level menu to be replaced on screen by the second-level menu, and so on. This made it confusing for the user to follow the menu hierarchy and lose track of how to access certain menu options. Another confusing element was the title bar, as it also acted as the status bar. There were also inconsistencies between the different visualisations, such as different functions of the settings button in the top right corner or a different number of visualisation boxes used. The difficult learning curve added to poor usability, as the visualisations could be rather difficult to follow or understand for users who were completely new to the particular algorithm. The visualisations also had certain disadvantages that came from using HTML5 Canvas, such as difficulty in creating smooth animations. Furthermore, the inner mechanism of the system also caused certain issues with playback, such as laggy rewinding.

4.2 Similar Algorithm Visualisation and Assessment Projects

Several other data structure and algorithm visualisation sites exist, as is recognised in Dr. Halim’s paper (Halim, Koh, Loh and Halim, 2012). Some of them are the University of Maryland’s Interactive Data Structure Visualizations, Virginia Tech’s AlgoViz Portal (Ullrich and Fellner, 2004), and the University of San Francisco (USFCA)’s Data Structure Visualisations. The closest and most extensive such project is the last of the 3 listed above.

While the USFCA site has a rather large library of visualisations and some of the animations are better than those in our original project, our project is still worth pursuing for several reasons. Firstly, our project contains certain visualisations not included in the USFCA, such as the Binary Indexed (Fenwick) Tree. It also supports interactive graph drawing², which allows users to draw their own graphs and then run graph algorithms on them, compared to hardcoded samples in most visualisation sites. Finally, and most importantly, our visualisation tool goes beyond just teaching – it includes testing capabilities as well.

An existing tool that has similar aims of both visualisation as well as assessment of algorithm topics is TRAKLA2 (Nikander et al, 2004). However, it is not as easily accessible as it requires a sign-up or download, and its assessment has a different focus from our system’s assessments – it mainly tests users on replicating the steps of an algorithm. Moreover, the system does not seem to have been updated very recently (last update was in 2009), and its user interface reflects this as well.

Yet another tool that features random question generators and automated testing is Wolfram Alpha’s Wolfram Problem Generator. However, it only covers mathematical topics, and not data structure and algorithm topics.

4.3 Online Learning and Testing

Some research was also done on online learning and automated testing and grading to understand more about this area and the relevant concerns.

² The graph drawing capability of the current version of this project is buggy, but once refined, it will be made available again in the future.

I observed the trends in Massive Open Online Courses (MOOC), and by joining Coursera, I learnt first-hand that online learning is difficult to regulate, difficult for many students to complete, and that most grading and certificate-awarding is honour-based. As the scope of our project does not yet extend to dedicated online learning and accreditation, this is not really an issue for us, although it does highlight the issue of authentication for online testing.

An article from the New York Times discussed the pros and cons of a system offered by EdX that does automated essay grading (Markoff, 2013). The benefits of such a system are in saving time and providing instant feedback, but detractors state that automated grading will never be as good as human grading, as machines cannot actually “read”. However, while we are intending to do automated grading, this criticism does not apply to our system, as the online testing and grading that we are planning to do does not have subjective answers. Our system deals with algorithms, so the answers provided will be clearly correct or wrong.

Another article from the University of Virginia discusses the problem of test randomisation and maintaining equal standards (University of Virginia, 2013). This is relevant to our project, as we plan to generate random questions on our system for tests for modules actually taught in NUS. The reason offered in the article for randomising test questions is to prevent cheating, as if students are given different test questions, they cannot compare or share answers, and there are no question sets or answer keys that can be found before the test. However, the article points out that such randomisation creates the problem of maintaining equal standards across all the randomised tests, i.e. some students may get a set of easier random questions, while some get more difficult questions. Grades are then unfair.

We discussed this problem somewhat, and one idea we had was to classify certain types of problems at certain difficulty levels and choose an appropriate number of random questions from each difficulty level. This does not nullify the problem, but it does reduce the level of unfairness. However, feedback from students from the current CS2010 class has shown that some are unhappy with the idea of randomised questions at all. Another idea was to use a “random seed” to generate a set of questions that *all* students would do. They may then be able to copy from each other, but there is still no pre-defined question set or answer key, and time is still saved in setting and grading the test. We eventually decided to use the random seed method to generate fair test questions.

4.4 UI Design

Much of my knowledge of UI design has been obtained from design and UI classes I have taken previously, such as CS3240 (Human Computer Interaction) in 2012 and NM3229 (Interactive Visualizations) in 2013. From these, I understand the design principles such as the Gestalt Principles (Graham, 2008) and usability guidelines. I read up on web usability and interaction design, as well as the Nielsen usability heuristics (Nielsen and Molich, 1990), summarised here:

- 1) Visibility: the system should always inform the user of its current state through appropriate feedback, and with reasonable efficiency.
- 2) Match Between System and Real World (Metaphors): Actions and information should be presented to the user in a familiar way. Follow real-world conventions.
- 3) User control and Freedom: the system should allow the user the ability to stop and exit any operation without a lengthy process. Forgiveness should be supported.
- 4) Consistency: platform interaction standards should be adhered to, and similar words and elements should mean the same thing or behave in the same manner.
- 5) Error Prevention: pre-empt and prevent errors by eliminating error-prone conditions, or by performing a check.
- 6) Recognition: the user should not need to rely on recall as much as recognition. Reduce the user's memory load by making actions and options visible.
- 7) Flexibility and Efficiency: allow novice users obvious controls, but also cater to more advanced users by offering shortcuts or accelerators.
- 8) Minimalist Design: do not overload the user by showing irrelevant information.
- 9) Error-recovery: If an error occurs, the problem should be described clearly and succinctly to the user, and a solution suggested.
- 10) Help: Ensure that the user has access to help at all times, and that the help provided is clear and succinct.

I also gained much inspiration and ideas of good static/interaction design from <http://www.awwwards.com/>, which recognises good web design and development, and gained knowledge of responsive design from online sources.

5 Design and Development Methodology

What follows is a description of the approach I adopted in the design and development of the UI for both the visualisation and “Online Judge” components. The steps were not always undertaken sequentially, rather, certain sets of steps could be revisited iteratively, in the style of agile development. This approach can be applied to any other project that I work on in the future.

1. Determine System Requirements

In order to design a system, whether in terms of UI or system architecture, it is always necessary to first have a clear understanding of what the system intends to do and the requirements that it needs to meet. The scope of the project was agreed on, both in terms of functional requirements (what capabilities the system was to be make available to the user), and non-functional requirements (e.g. extensibility, compatibility, security).

2. Determine User Workflow

Once the functional requirements had been established, it was then determined how the content should be organised, which pieces of information or options were to be made available to the user at which point in time, and the sequence of steps the user should have to take to accomplish each task.

3. Initial Design Mockups

The next step was to draft several different design mockups based on the system requirements and determined user workflow. This was done initially as rough sketches on paper, and later in more polished forms in Adobe Photoshop. At this step, it was important to keep in mind important web and graphic design principles such as the importance of the grid, and interface design heuristics such as the Nielsen Heuristics – visibility, user control, consistency, error prevention, recognition, flexibility, minimalist design, error-recovery, and access to help. Due to the web-based nature of this project, it was also necessary to consider web responsiveness

(how to handle the page layout on screens of different resolutions) when designing the various web pages.

This step did not involve creating a single design; rather, it involved creating multiple differing designs concurrently, evaluating and modifying them along the way to find the best resulting approach. Some designs were discarded at certain points when it became clear they did not fit the requirements or did not seem to be working out. The result was a final set or at most two sets of mockups to be prototyped.

4. Develop Prototype

A prototype was built next, based on the mockups created previously. This was done in actual HTML/CSS/JavaScript, and hosted locally or on an available server. While this step used the same tools as the actual full development phase, it was still a prototyping phase, as it was done on a smaller scale (a single visualisation page), or with hard-coded functions instead of full back-end capabilities. In this way, the design could be tested in a layer of abstraction, without having to wait for the full set of functionalities to be coded completely.

The previous pure design stage, while taking into account multiple considerations, often does not consider all possibilities, something that becomes clearer and can be better considered during development. Thus, prototyping is an important step in evaluating the design. If multiple mockups were prepared at the previous stage, both were prototyped to examine how each would work in development, and a final decision made on which design to adopt. This design was then refined in the prototype, with more detailed choices being made, such as interaction-dependent behaviour (e.g. appearance changes on mouse hover and mouse click, or event transitions).

5. Test Prototype and Collect Feedback

As Dr. Halim was teaching CS2010 Data Structures and Algorithms II in AY13/14 Semester 1 (the first semester of this final year project), we were able to test our prototypes and gather feedback by letting the students in his class use the system for the module. Working as a Teaching Assistant for this class gave me the opportunity to work directly with the students

and gather feedback through personal interaction with the students, in-class observations of their usage patterns, and surveys.

6. Evaluate and Refine Design

Based on the feedback received, I made changes to the prototype design to fix certain problems faced by the students. The changes made at this step were usually not major, however, the user workflow model was sometimes tweaked at this point, based on prior observations of actual user interactions.

7. Full Development

Finally, the project went into full development. As the project developed and more functionalities were added, problems were sometimes encountered, or it was observed that the system needed to support other functions. When this occurred, I would consider the current system as a whole and determine how the new feature should be added in a way that fit in with the design. This is where agile development was necessary, and steps 4 to 6 were revisited iteratively.

6 Design and Development of the Visualisation Component

While the focus of this report is on the “Online Judge” system, the re-design of the visualisation component was first necessary so that the testing component could be designed in a consistent manner, in order to ensure the cohesiveness of the whole VisuAlgo project.

6.1 System Requirements

Based on the analysis of previous team’s work and discussions with Dr. Halim and my fellow project-mate Ivan Reinaldo, we came to a shared understanding of what the project aimed to do, and the functions that it had to support. The requirements of the visualisation project can be summarised in the following categories:

6.1.1 Functional Requirements

- 1) The visualisation tool should include representations of data structures and algorithms taught in the following NUS modules:
 - a. CS1020 Data Structures and Algorithms I
 - b. CS2010 Data Structures and Algorithms II
 - c. CS2020 Data Structures and Algorithms Accelerated
 - d. CS3230 Design and Analysis of Algorithms
 - e. CS3233 Competitive Programming
- 2) All visualisations must be accessible to the user from a central (home) page
- 3) The visualisation tool must be able to show the sequence of steps involved for each operation related to a certain data structure or algorithm
- 4) The visualisation tool must allow the user to control the playback of the sequence of steps for each operation, including the ability to pause, replay, move directly to a particular step in the sequence, and control the playback speed
- 5) The visualisation tool must provide an explanation or description of each step in each operation
- 6) The visualisation tool must provide pseudocode-tracing for each operation
- 7) The visualization tool should allow users to build instances of provided data structures and representations of algorithms using their own datasets, instead of relying on a hard-coded dataset

6.1.2 Non-Functional Requirements

- 1) Performance: all operations and animations should occur with minimal delay
- 2) Extensibility: the system should be able to accommodate additional secondary features in the future
- 3) Compatibility: the visualisation tool must be able to function fully in the latest version of Google Chrome (Version 33). If possible, it should work on Mozilla Firefox Version 25+ and Safari Version 6+ as well. It should be usable on tablets as well as laptops, on screens with a resolution of at least 1024x768 px.
- 4) Usability: the visualisation tool should be reasonably easy to learn, easy to use, and enjoyable to use

6.2 User Workflow and Design Mockups

The user workflow was drafted as follows:

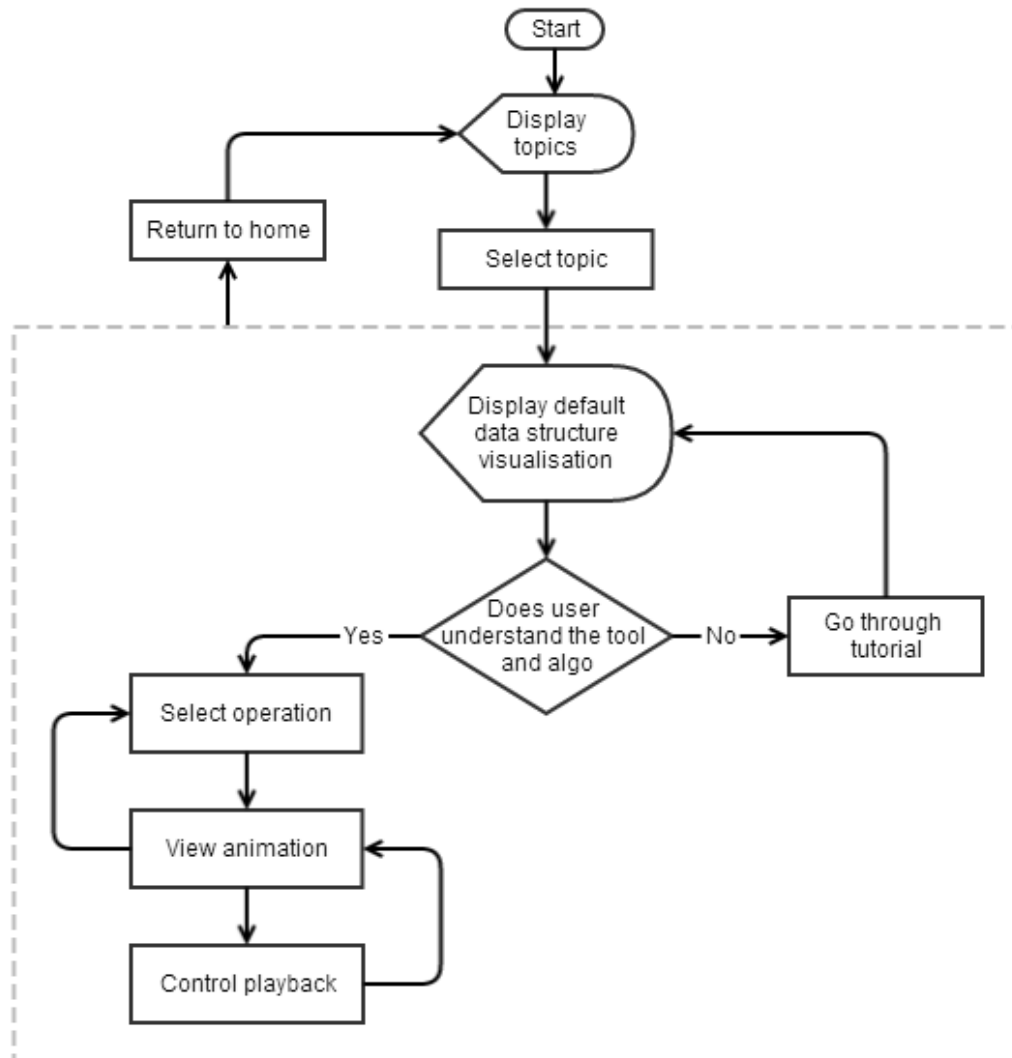


Figure 1: User workflow for visualisation component

Based on this design, it was evident that 2 page templates needed to be designed – a home page with links to all the different topics, and a visualisation page for each topic. These visualisation pages should all have the same structure, but with customised operations and animations. 2 modes were identified for the visualisation page – the default “exploration” mode for students to dive into trying the different available operations and view the animations, and a tutorial mode, for students to learn about the algorithm and the tool interface.

Each visualisation page had to have the following elements:

- 1) Link to return to the home page
- 2) Visualisation title and alternate versions (e.g. the Binary Search Tree visualisation requires 2 versions of the data structure – regular BST and balanced AVL tree. The 2 structures are too similar to each have their own visualisation page. Rather, there should be an option to toggle between these 2 structures on a single visualisation page.)
- 3) The visualisation area
- 4) Menu of available operations and their input forms
- 5) The current operation name
- 6) Operation step status description and codetrace, grouped together as they fulfil the same function of providing the user with a description of the current step in an operation. For consistency in design, it was agreed on that all pseudocode would be limited to 7 lines.
- 7) Playback controls and animation speed control
- 8) Exploration and Tutorial mode toggle
- 9) At this point, we also wanted to include a Facebook login as a social feature.³

6.2.1 *Initial Ideas*

After much experimentation, I arrived at a first draft:

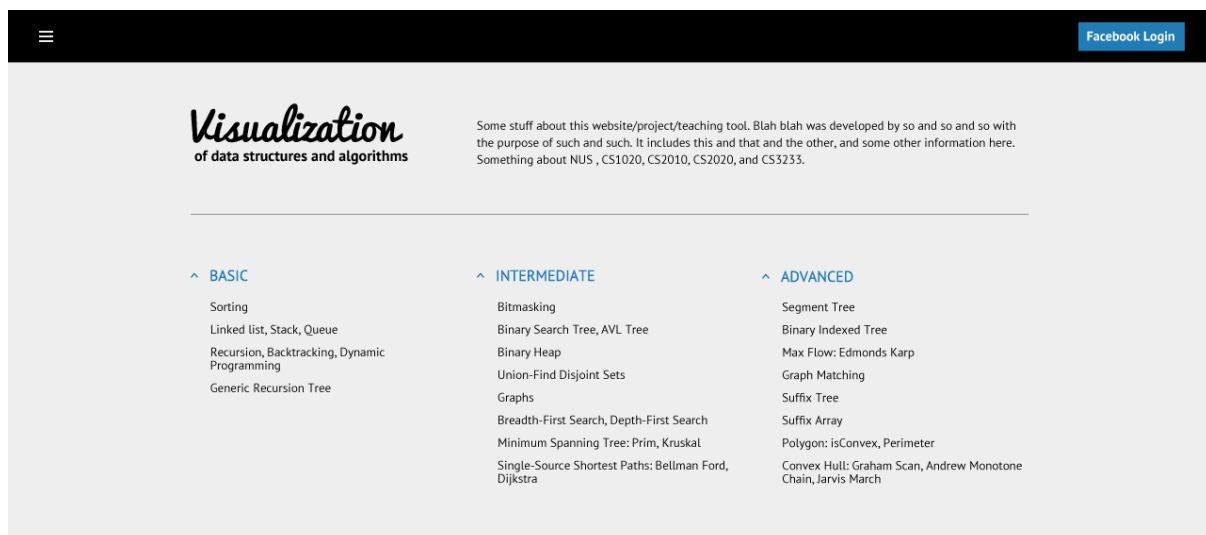


Figure 2: Initial homepage design

³ This feature is no longer a priority.

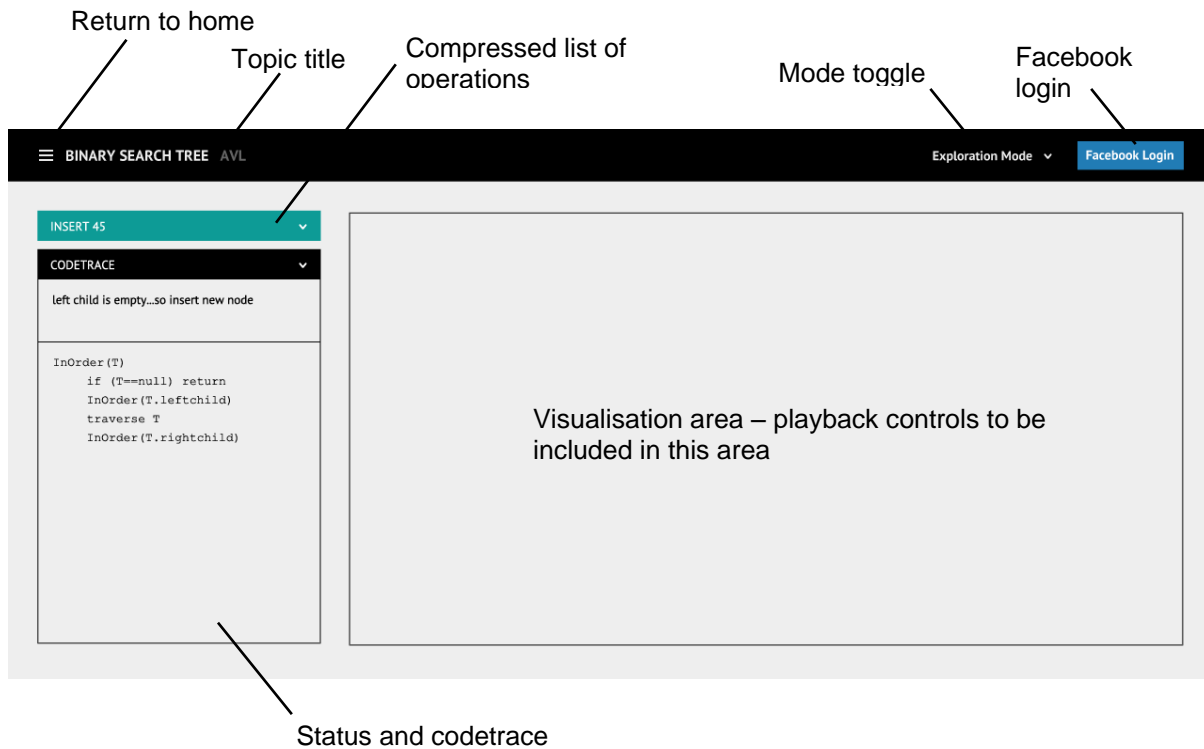


Figure 3: Initial visualisation page design

However, several problems were identified with respect to this design mockup. Firstly, the visualisation area felt very restrictive, especially with limited screen space on smaller screens. Secondly, it was difficult to display the menu of available operations in this design with each operation's function signature requiring different types of input boxes. It was also decided that to make the home page more visually appealing, each topic link should have a unique image thumbnail to depict that particular data structure or algorithm.

6.2.2 Improved Design

Another design was thus drafted, with additional attention paid to making the user experience enjoyable. I utilised a bright colour palette to try to induce a more playful and welcoming feeling.



Figure 4: Colour palette used

The new design looked thus:

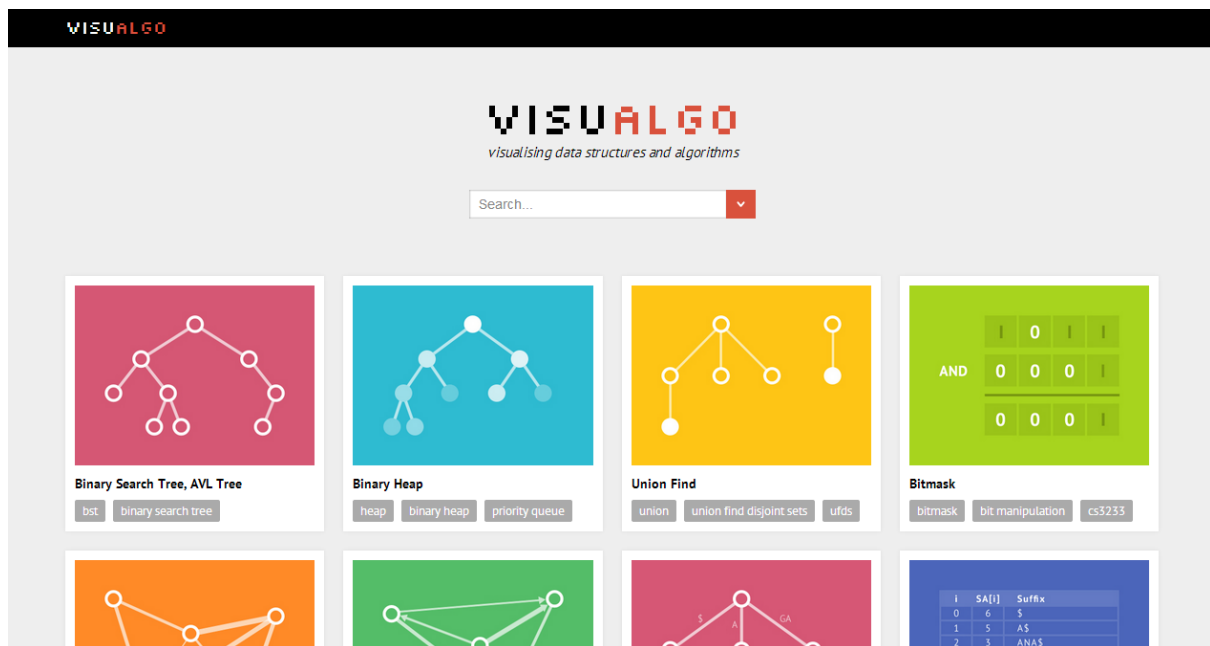


Figure 5: New home page design

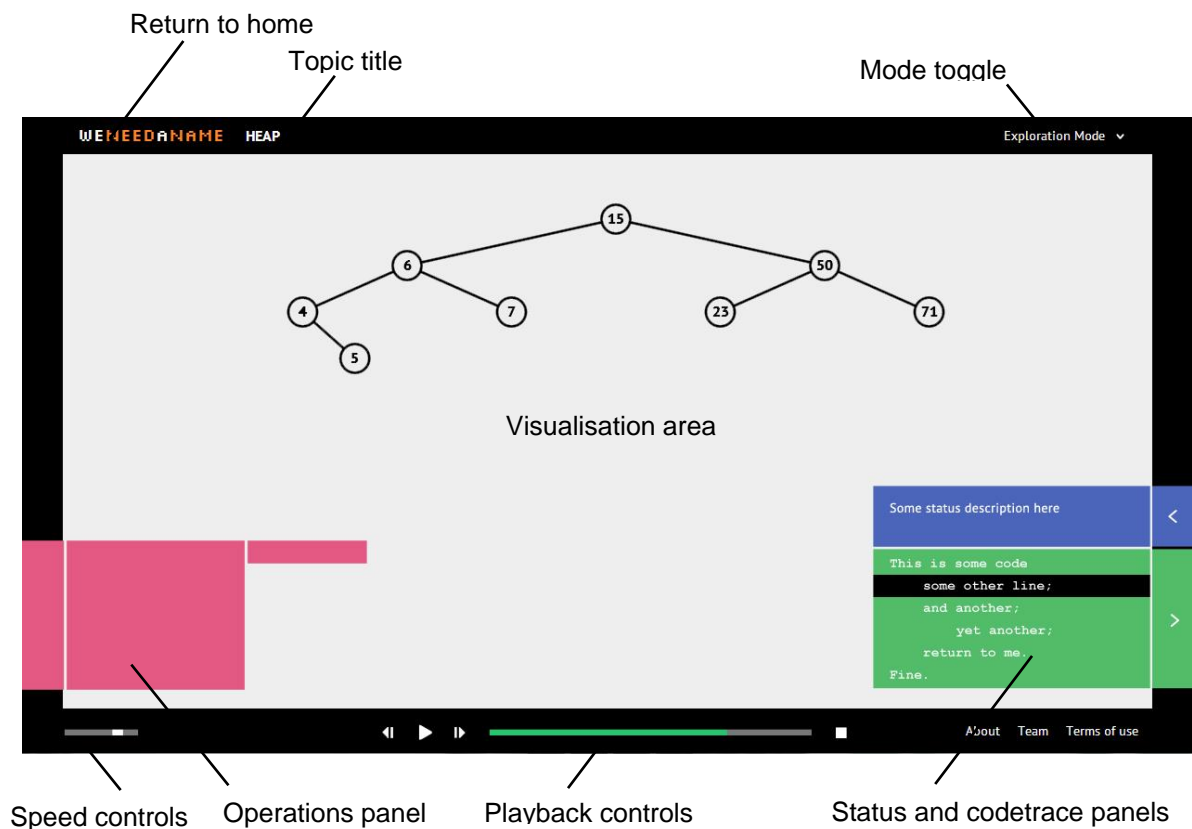


Figure 6: New visualisation page design

The new design solved the problem of restrictive visualisation space by dedicating the entire screen to the visualisation, while the panels could be hidden if necessary. The black frame gave focus to the visualisation area, and gave the entire tool a sandbox-feel, which was ideal for the idea of learning and exploration. The operations menu also worked much better with this design, because it could accommodate a 2-tier menu with different inputs for each operation.

6.3 Prototyping, Refinement, and Full Development

A prototype was built for the homepage and the Binary Search Tree visualisation page based on this design, and refinements were made as the prototype was developed.

Each thumbnail on the homepage was designed to have a static and an animated GIF version, played when the user hovers over that topic thumbnail. These small refinements were added to contribute to creating an interactive and fun user experience. The short animations also tie in with the concept of the visualisation tool, giving users an idea of what the system does. Another feature that was used to contribute to the sense of fun was the use of randomised colours (from the previously defined palette) for the various panels on the visualisation page. While not essential, the sense of surprise aims to make the experience more enjoyable.

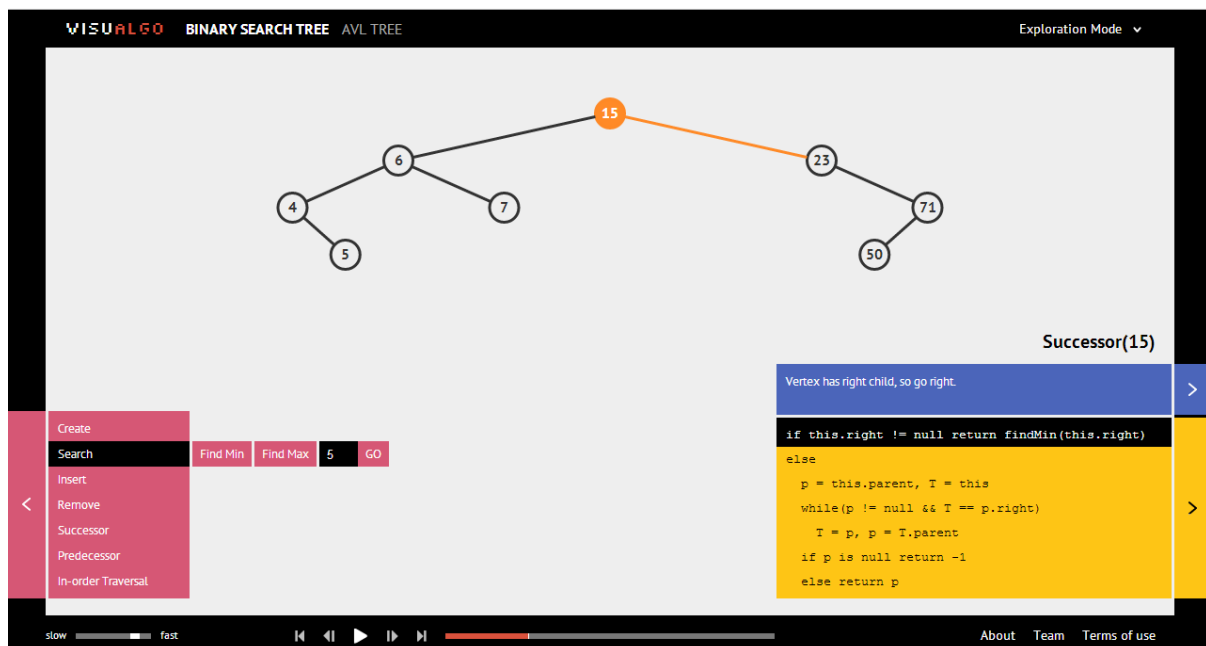


Figure 7: Fully developed visualisation page

After observing the lecturer use the tool in class to give algorithm demonstrations, the playback controls were refined to suit the controls he constantly needed. Some shortcut keys were also included so that users can use the tool more efficiently.



Figure 8: Final playback controls close-up

Playback control	Shortcut key
Play, Pause, Replay	Spacebar
Step forward	Right arrow
Step backward	Left arrow
Go to beginning	
Go to end	
Increase playback speed	+ (or =)
Decrease playback speed	- (or _)

Table 1: List of playback controls and their shortcut keys

As the development of the various operations for each topic progressed, their respective visualisation pages were built using the same UI template.

6.3.1 Summary of Code Structure

For a full description of the entire system architecture, refer to the project numbered H160080. For consistent styling, all HTML pages in this component share a common CSS file and a common JavaScript file where the colour palette and shared footer information is defined (About, Team and Terms of Use statement). The home page has its own CSS and JavaScript files to handle home page-specific functionality, while all visualisation pages share a common graph library, CSS file and JavaScript file controlling common visualisation functionalities such as panel styling and panel open/close transitions. Finally, each specific visualisation page has its own CSS file to style it's menu options for each operation, a JavaScript widget file which defines the animations for each operation, and another JavaScript file which defines the behaviour of the menu options for each operation (e.g. one input box for the BST search operation, and 3 options for the BST create operation).

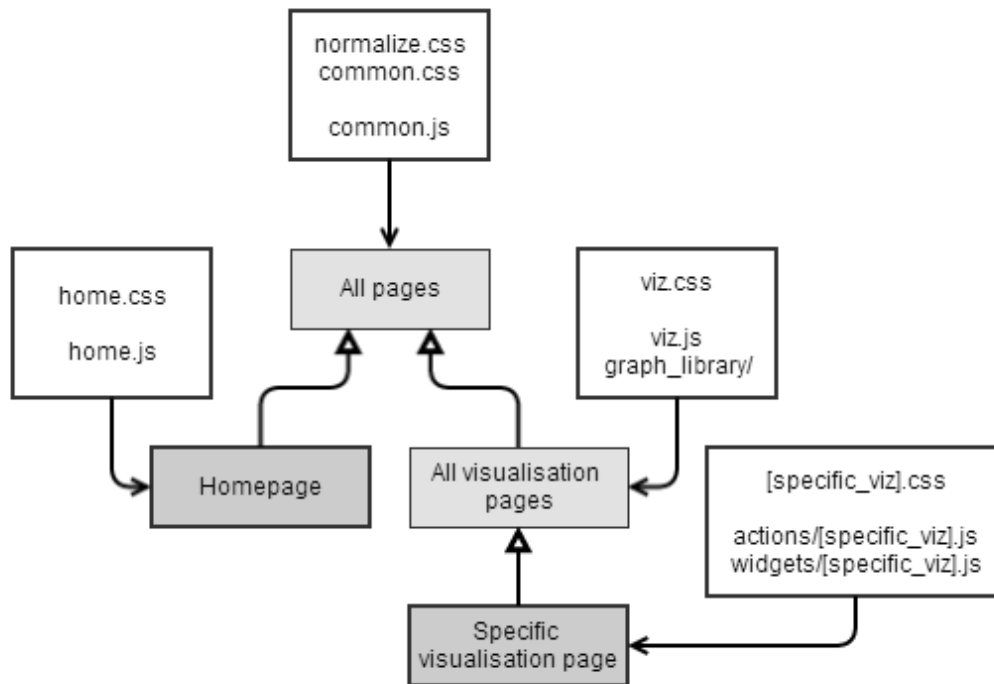


Figure 9: File includes and page relationships for the visualisation component

7 Design and Development of the “Online Judge” Component

With an established design for the visualisation tool, the “Online Judge” component could then be designed as well, with the same overall design concept and look-and-feel.

7.1 System Requirements

7.1.1 *Functional Requirements*

- 1) The Online Judge tool should be able to generate questions on data structures and algorithms taught in the following NUS modules:
 - a. CS1020 Data Structures and Algorithms I⁴
 - b. CS2010 Data Structures and Algorithms II
 - c. CS2020 Data Structures and Algorithms Accelerated
 - d. CS3230 Design and Analysis of Algorithms⁴
 - e. CS3233 Competitive Programming⁴

⁴ Future work

- 2) The Online Judge tool must be able to display the questions it generates to the user clearly, both in terms of question phrasing and visual display of data structures
- 3) The Online Judge tool must be able to handle a predefined set of types of user input, and ensure that user input is recorded and sent to the server with 100% accuracy. The types of user input it should handle include but are not limited to:
 - a. Vertex selection (single and multiple, ordered and unordered)
 - b. Edge selection (single and multiple, ordered and unordered)
 - c. Number input
 - d. Multiple choice options
 - e. “No answer” option where applicable
 - f. Graph drawing input⁴
- 4) The Online Judge tool must always show the user his/her current answer to each question where applicable
- 5) The Online Judge tool should allow the user to navigate between the test questions
- 6) For graded online tests, the Online Judge must constantly display to the user the time remaining for the test
- 7) The Online Judge tool must be able to verify the correctness of answers to the questions it generates with 100% accuracy
- 8) The Online Judge tool must be able to display the answers to the questions it generates
- 9) The Online Judge tool must be able to grade the responses for the question sets it generates and display the resulting score to the user
- 10) The Online Judge tool should allow the test administrator to generate a parameterised question set to be used in a graded online test
- 11) The Online Judge tool should allow the test administrator to control student access to the graded online test and answer key, as well as monitor and record the results of the test

7.1.2 Non-Functional Requirements

- 1) **Stability:** the Online Judge system must always be able to run without bugs, especially during graded tests
- 2) **Compatibility:** the visualisation tool must be able to function fully in the latest version of Google Chrome (Version 33). It should be usable on screens with a resolution of at least 1024x768 px.

- 3) Usability: the Online Judge tool should be instantly usable (practically no learning curve), and easy to use
- 4) Security: students should not be able to gain access to the test questions before the test, the test answers before or during the test, allow other students to take the test for them, give themselves more time, or change recorded test scores
- 5) Extensibility: the system should be able to accommodate additional secondary features in the future

7.2 Preliminary Workflows, Designs, Prototyping and Refinement

The NUS AY13/14 Semester 1 CS2010 module included 2 online quizzes as part of its overall assessment, which were used as prototype tests for the “Online Judge” tool. Therefore, there were 2 iterations of prototyping – Online Quiz 1 (19 Sep 2013) and Online Quiz 2 (31 Oct 2013).⁵ Both prototypes were built without random question generators and graders, rather, the questions and answers were hardcoded into the system. Each test had a single JavaScript file handling the test UI logic and AJAX calls to the server for login validation as well as question, answer and elapsed time data. Both the prototypes and the final development were hosted on the server at <http://algorithmics.comp.nus.edu.sg/>.

7.2.1 *Prototype 1: Online Quiz 1*

The requirements and scope for Online Quiz 1 were:

- 1) 10 questions selected randomly from 20 set questions, with 5 of them “easy”, 3 of “medium-difficulty”, and 2 of them “difficult”
- 2) The question order would be scrambled, i.e. a “difficult” question could be question 1 while an easy question could be question 10
- 3) Each student would be given 2 attempts, and the questions for each attempt may be different (see requirements 1 and 2)
- 4) A time limit of 10 minutes for each attempt
- 5) The student can choose to use both attempts or only 1, where the *final* score (note: not the higher score) would be recorded

⁵ Due to the beta nature of the testing system at this stage, as a safeguard, Online Quiz 1 only offered students 2 bonus marks to their Online Quiz 2 score, the latter of which was given an overall weightage of 10%.

Based on these requirements, the following user workflow was drafted:

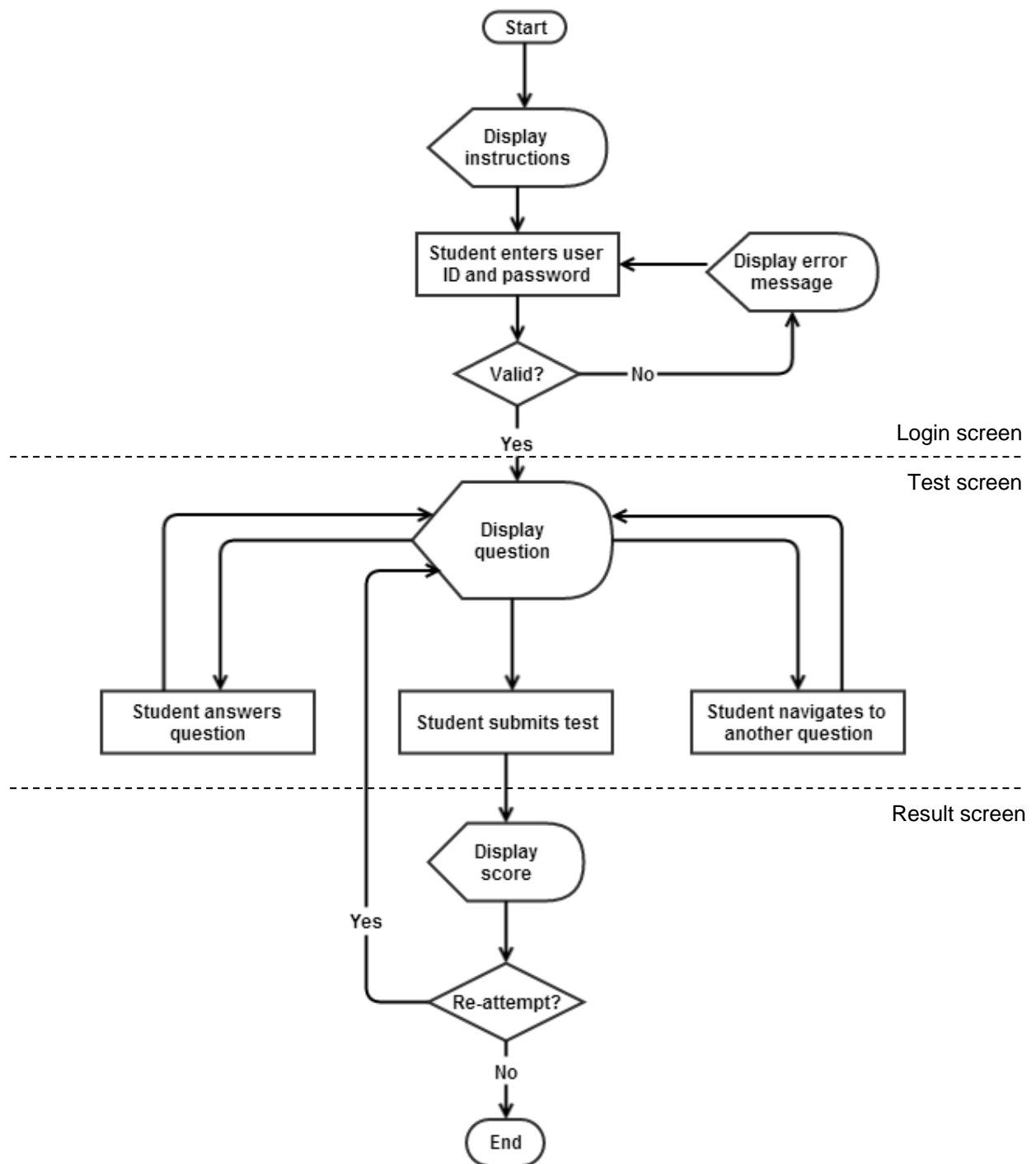


Figure 10: User workflow for Online Quiz 1 test page

From this workflow, it was evident that 3 screens had to be developed – the login screen, the main test screen, and the result screen. The login screen had to include the test instructions and a login form. The initial mockup was designed thus:

WENEEDANAME ONLINE QUIZ 1

INSTRUCTIONS

- + You are given 2 attempts. The result from your final attempt will be recorded.
- + Your first attempt begins when you log in.
- + If you choose to make a second attempt, you may get a different set of questions.
- + You are given 10 minutes for each attempt.
- + Do not refresh your browser, go back, or close the window while taking the test.
- + Your score is only valid if you are physically present in the lab.
- + You are not allowed to discuss the questions or answers until 4pm today, even with your fellow lab group mates.
- + In the case of any disputes, the decision of the lecturer is final.

user id

password

LOG IN

About Team Terms of use

Figure 11: Online Quiz 1 login screen

The test screen had to include the question text, question graph, question navigation, submit button as well as status information such as the student's name (for verification purposes), attempt count, and time remaining. The only user-input this prototype had to support was single vertex selection. The vertices were thus made clickable, and highlighted when selected by the student to indicate the student's answer.

A preview of the system to the students prompted the concern that the submit button could be confusing or clicked by accident, which would be very problematic. A simple fix was made for the test by adding a window confirm dialog on clicking the submission button. However, this problem was noted down for refinement in the next iteration.

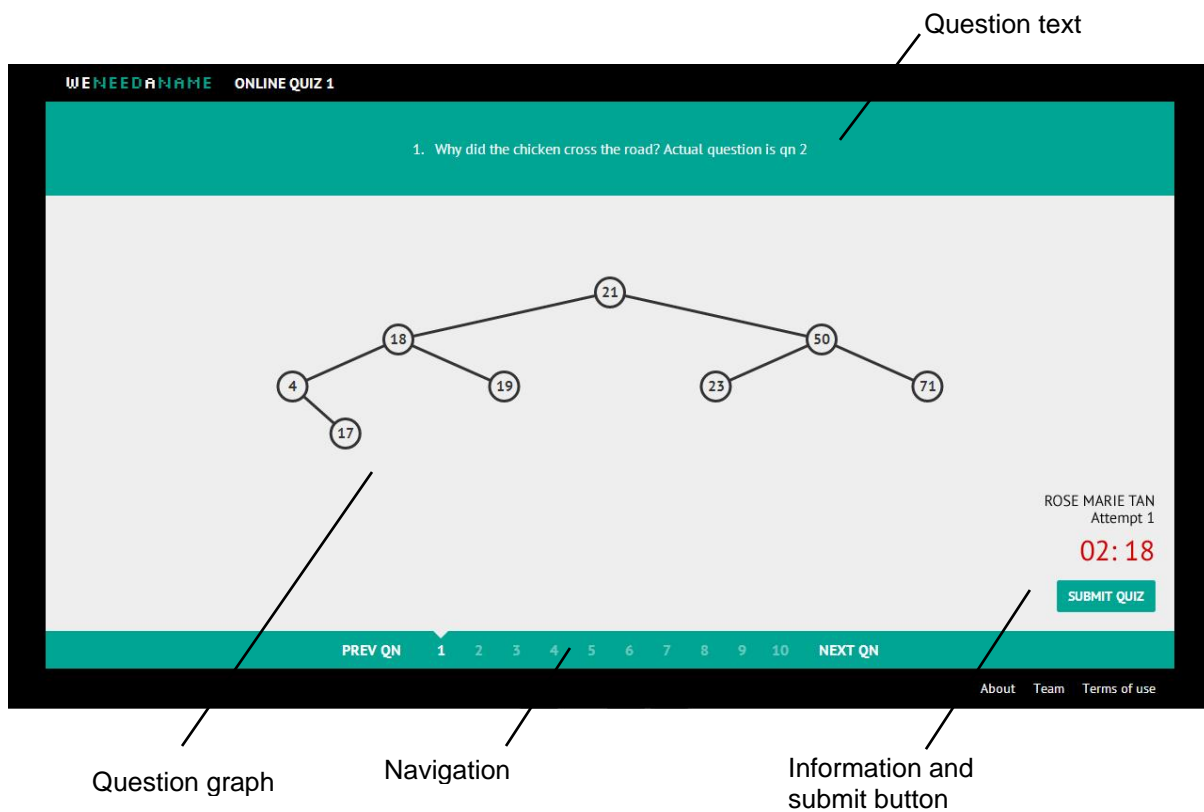


Figure 12: Online Quiz 1 test screen

On confirmation of submission, students were brought to the result screen:

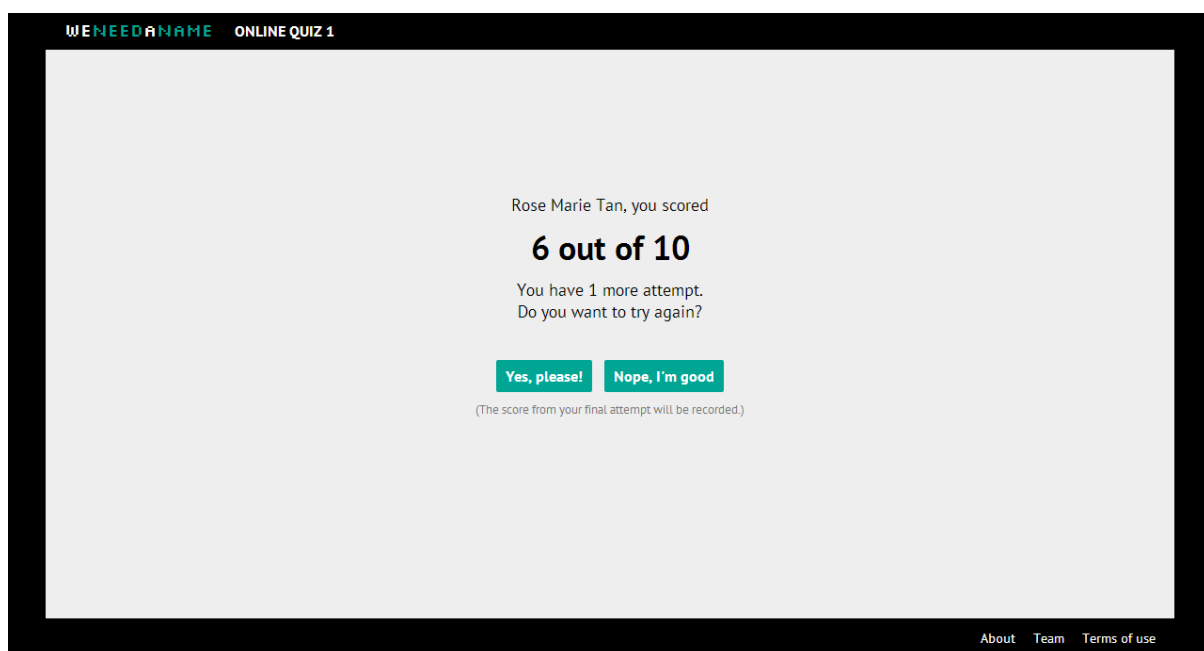


Figure 13: Online Quiz 1 result page

If they had attempted the test twice already, the message text would read differently, and the options would not be shown. Besides the test itself, an answer key was also needed, as stated in the functional requirements. The workflow for the answer key was simpler:

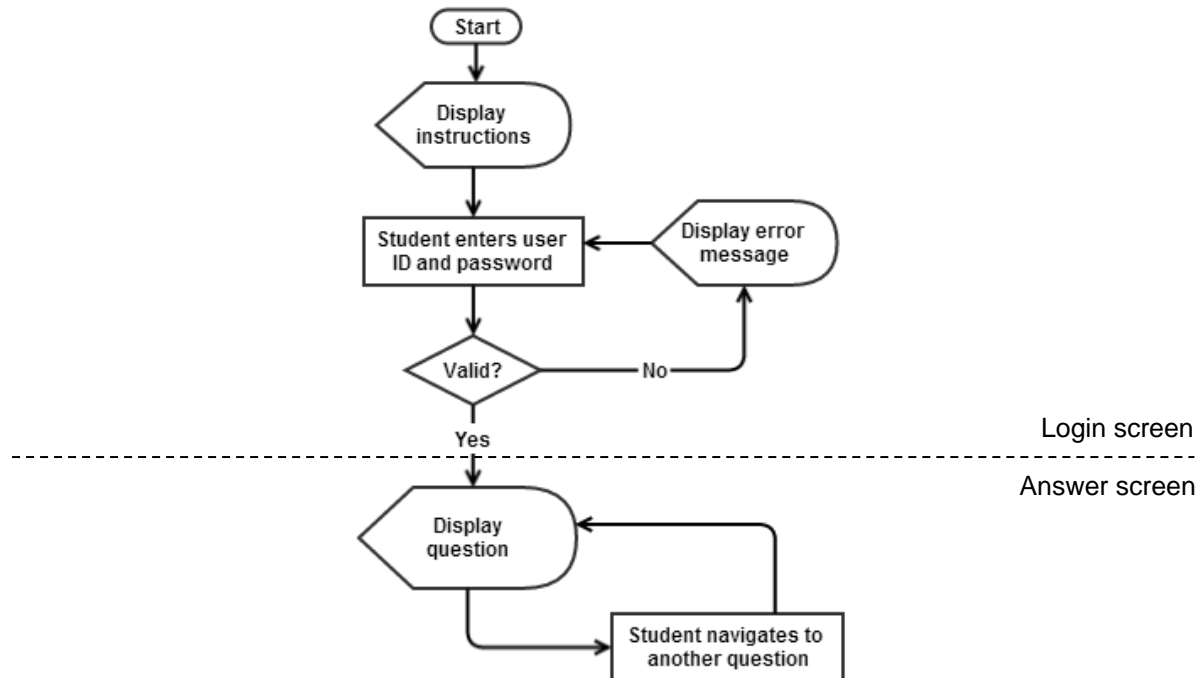


Figure 14: User workflow for Online Quiz 1 answer key page

The answer key thus had 2 screens – a login screen and an answer screen. These 2 screens were laid out in the same way as the login screen and test screen of the actual test respectively, so their screenshots will not be included.

7.2.2 Prototype 2: Online Quiz 2

Feedback on the UI/UX for Online Quiz 1 was collected from students verbally, via the CS2010 Facebook group, and from a CS2010 IVLE survey (see Appendix A). There were many positive comments, but there were also some negative impressions, the most common of which were:

- 1) Lack of mental preparation in starting the quiz, as the quiz starts immediately after login
- 2) Difficulty reading the white question text on the green background
- 3) Students thought the “submit” button was the “next question” button

These concerns were taken into account and the UI for the second prototype, Online Quiz 2, was refined accordingly (Fig. 16).

To address (1), an additional screen for the instructions was added in-between the log-in screen and the test screen, with a “Start” button so that students could get a feel of when they were really starting the test.

For (2), the question text was changed to black text on a white background, which is a much better colour combination for readability (Hall and Hanna, 2004). The text alignment was also changed from centre-aligned to left-aligned, as left-aligned text is more readable.

For (3), a few adjustments were made. Firstly, the submit button was moved to the top right of the screen, away from the question navigation bar, so that students would not confuse the submit button with question navigation. Secondly, 2 pseudo-popup checks were added to ensure that students were really aware that they are submitting the quiz rather than clicking on it accidentally. Finally, the button was made only visible when the student had answered all questions. Otherwise, as with Online Quiz 1, the quiz would submit automatically when time ran out.

There were also a few changes to the specific test requirements. Firstly, there were 20 fixed questions instead of 10 picked from 20. Secondly, only one attempt of 40 minutes was allowed. This caused a change in the workflow:

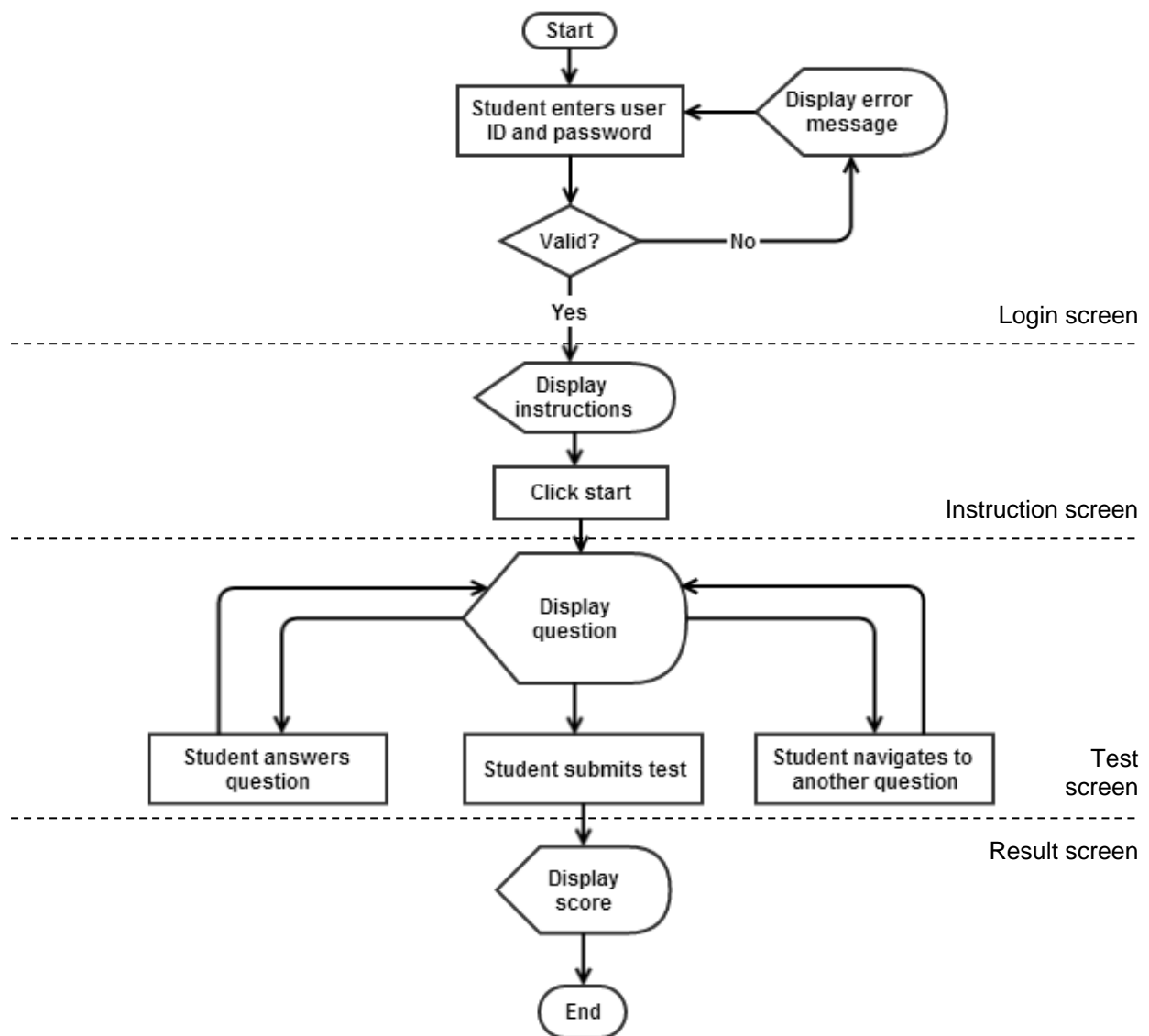


Figure 15: User workflow for Online Quiz 2 test page

This iteration also had to support a few other input types – in addition to single vertex selection, multiple vertex selection, single edge selection, and multiple edge selection were supported. Due to the newly supported input types that involved ordered selection of multiple edges or vertices, highlighting the selected elements was no longer enough. Thus, the selection for each of these questions was also printed on the screen under the question text. To cater to multiple element selection, “undo” and “clear” buttons were also added (Fig. 16).

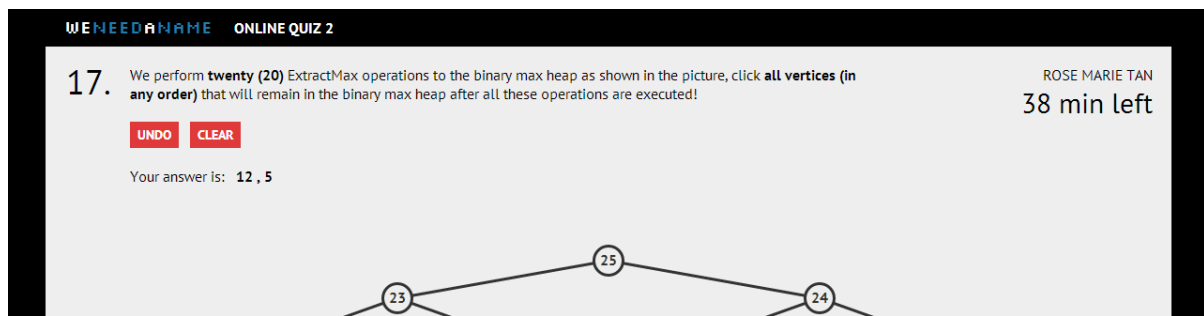


Figure 16: Online Quiz 2 test page - interface modifications and additions

The result screen was similar to that from Online Quiz 1, without the re-attempt option, and the answer key was updated accordingly as well, once again with 2 screens, similar to the Online Quiz 2 login and test screens, but without the “undo” and “clear” buttons, and with both the student’s answer and correct answer displayed.

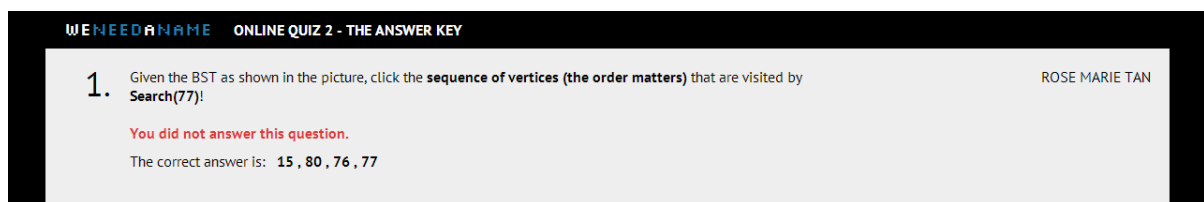


Figure 17: Online Quiz 2 answer page - interface modifications and additions

This prototype did not receive much negative feedback from the students, the only commonly-raised issue being a slight difficulty in selecting the edges. However, this was an artefact of the graph widget, which was difficult to change and outside the scope of my project. In all other aspects relating to the UI, this prototype was rather well-received, and students were able to adapt to the new input types without issue.

7.3 Final Workflow, Design and Full Development

While the question generators and graders were being developed concurrently, the interfaces for the full range of pages of the “Online Judge” component was developed. While the prototypes had been built for recorded tests, VisuAlgo also aimed to build a training mode, so that students would have the opportunity to practise the topics before the actual tests. Additionally, an admin panel had to be designed and developed, to make the entire automated system work more smoothly and so that the test administrator, while not having to set the questions specifically, could still have good control over the testing process.

7.3.1 Training Mode

This mode had a slightly different set of interface requirements from the test mode:

- 1) No user login or time limit
- 2) The user should be able to select the topics he/she wants to practise
- 3) The user should be able to submit the test at any point in time
- 4) The answers should be shown to the student immediately after the result screen

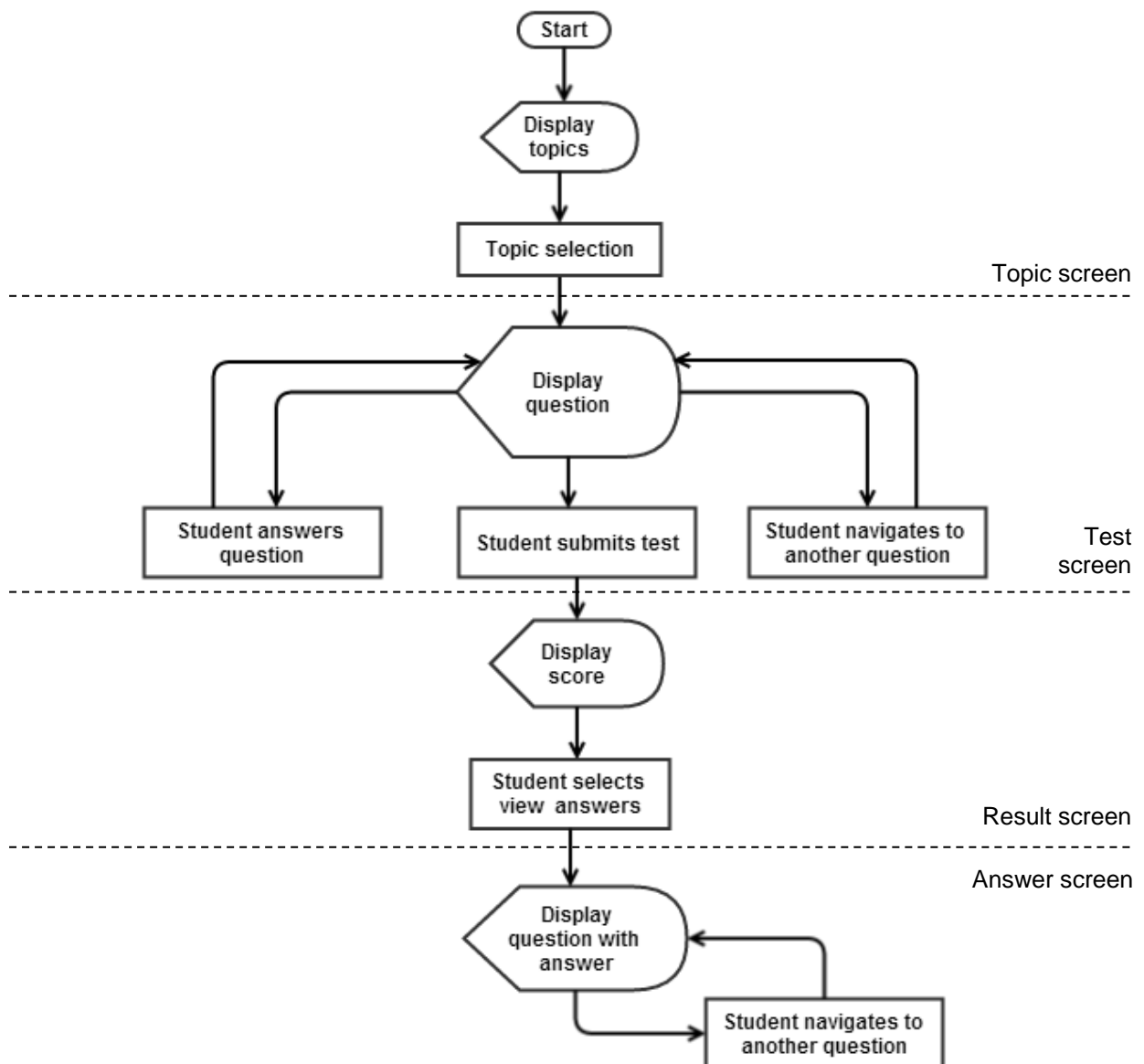


Figure 18: User workflow for training mode page

The final UI for the training mode is described in the following screenshots:

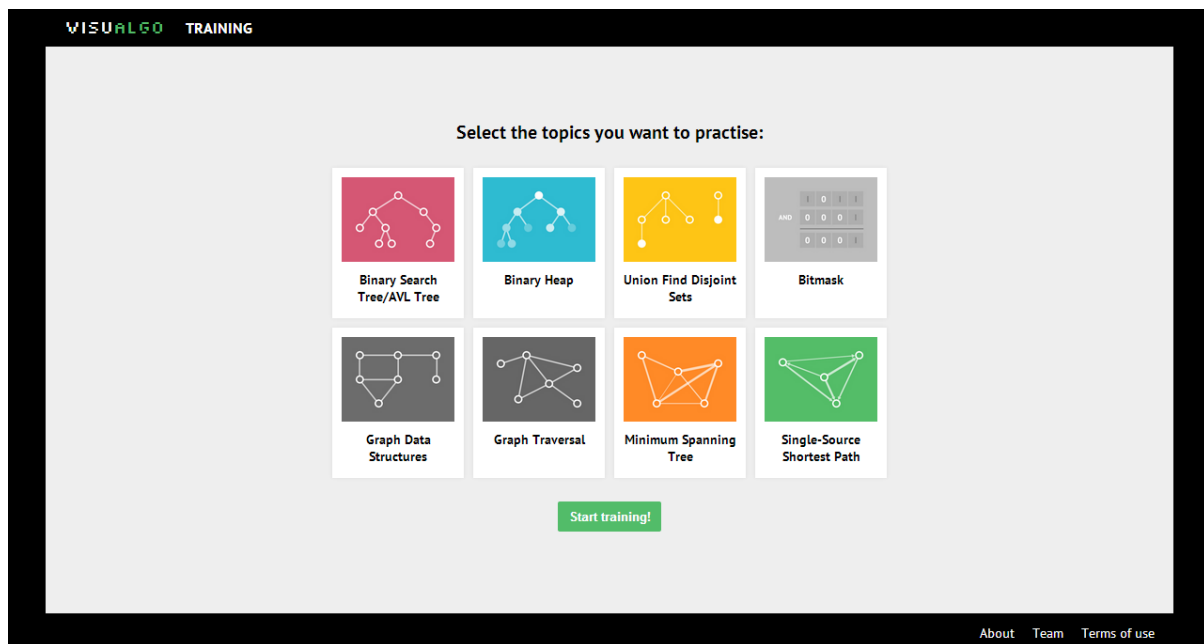


Figure 19: Training mode topic selection screen

All available topics are shown to the user with the same thumbnail images used on the home page of the visualisation tool for consistency. Selected topics appear in colour, while unselected topics are in greyscale. Initially, all topics are unselected. When the user decides to start, the test screen is shown.

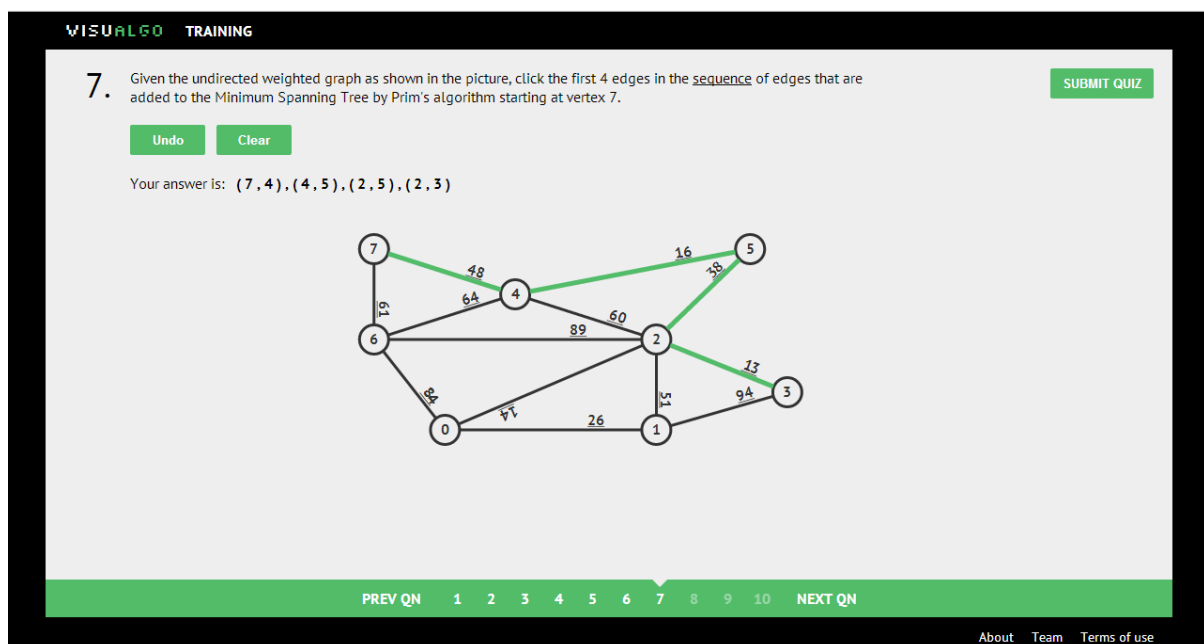


Figure 20: Training mode test screen

For the full list of different input interfaces, view Appendix C. After submitting the quiz, the user will be shown his/her result.

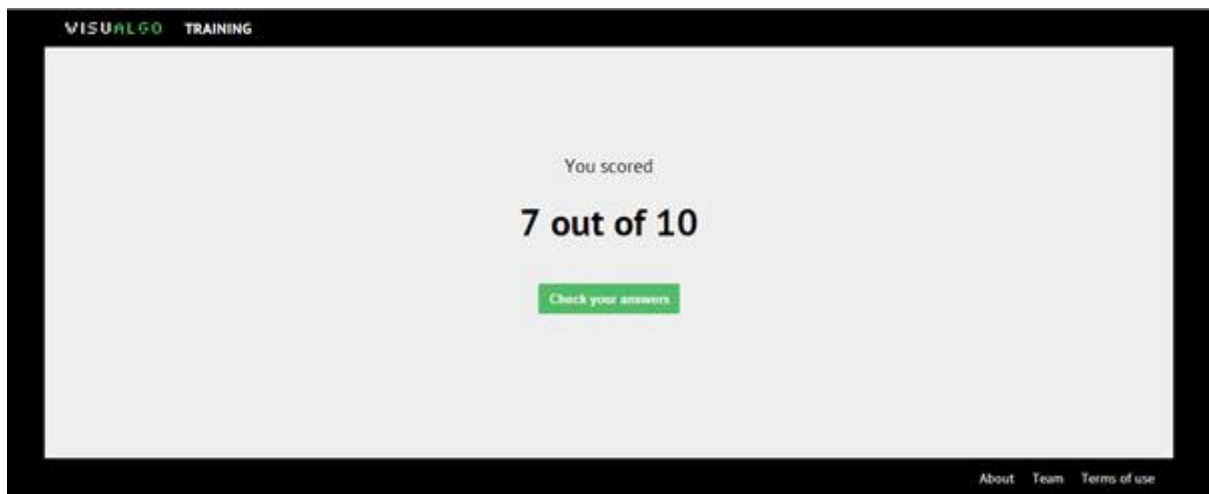


Figure 21: Training mode result screen

Finally, the user can check his/her answers and view the correct answers.

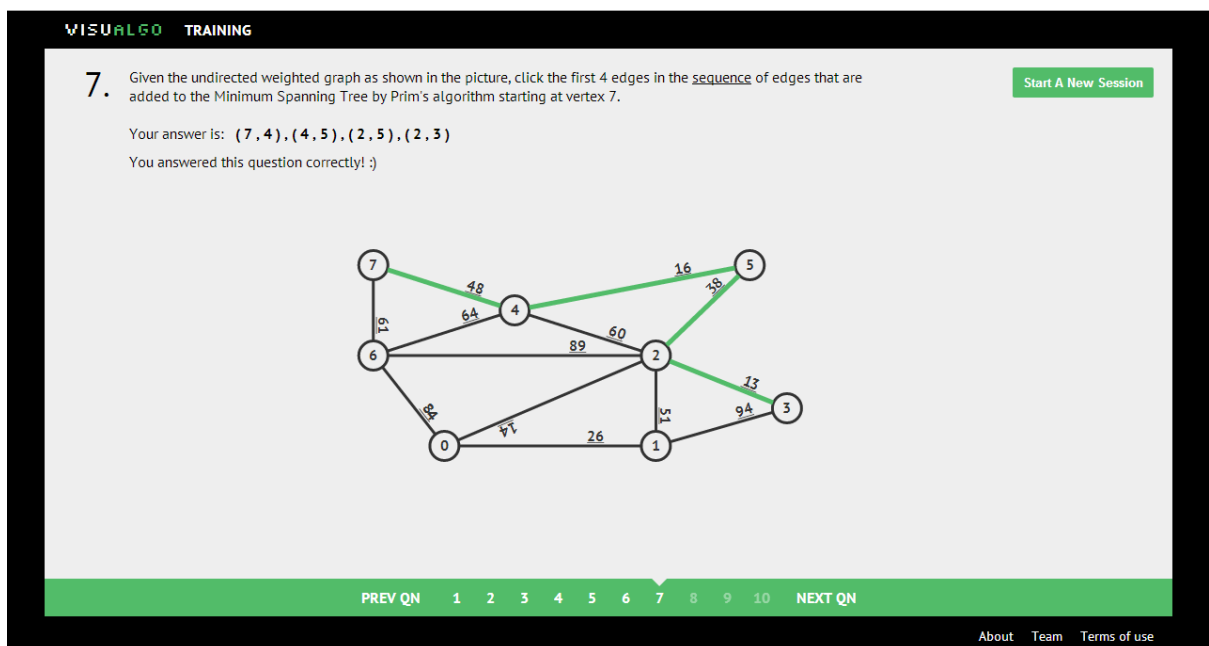


Figure 22: Training mode answer screen

Instead of the “submit” button on the top right, the user now has the option to start a new practice session.

7.3.2 Test Mode

The test mode is made up of 4 different presentation pages – the test itself, the answer key, the scoreboard, and the admin control panel. The test itself follows the same user workflow as the final prototype, Online Quiz 2 (Fig. 15). However, as the screens for Online Quiz 2 were somewhat customised to that particular test alone, some minor adjustments were made for generic tests in the future. A “log out” button was also added to the result screen to give students a sense of completion. The final test screens are shown below. The different internal interfaces for each input type are exactly the same for the test mode as for the training mode (refer to Appendix C for the full list).

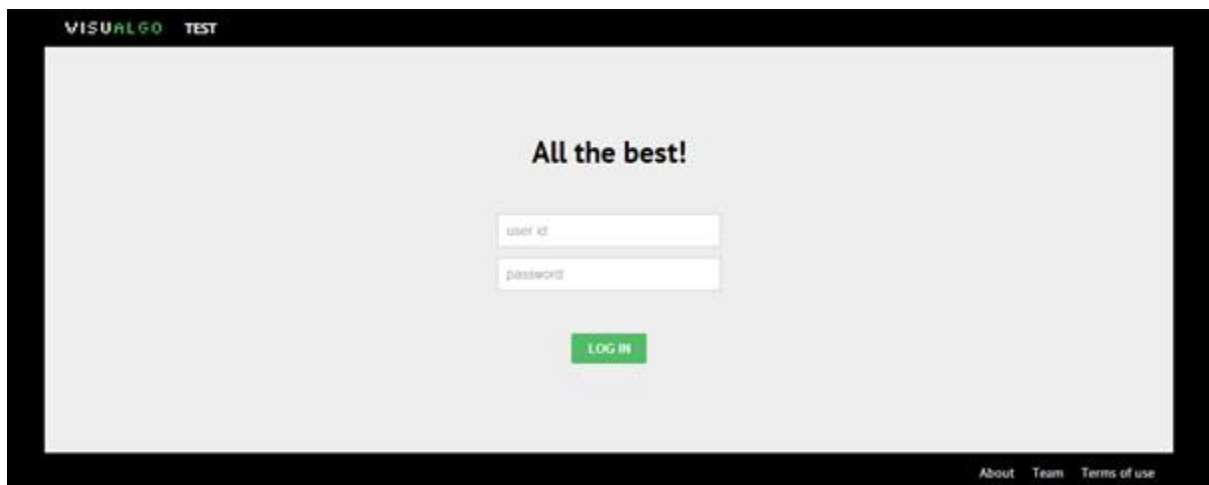


Figure 23: Test mode login screen

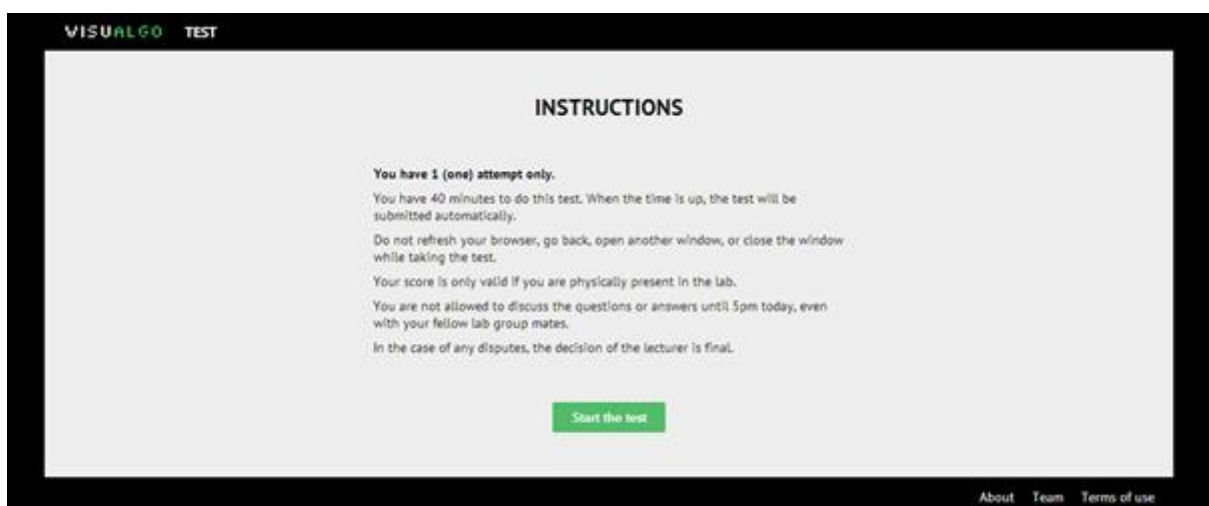


Figure 24: Test mode instruction screen

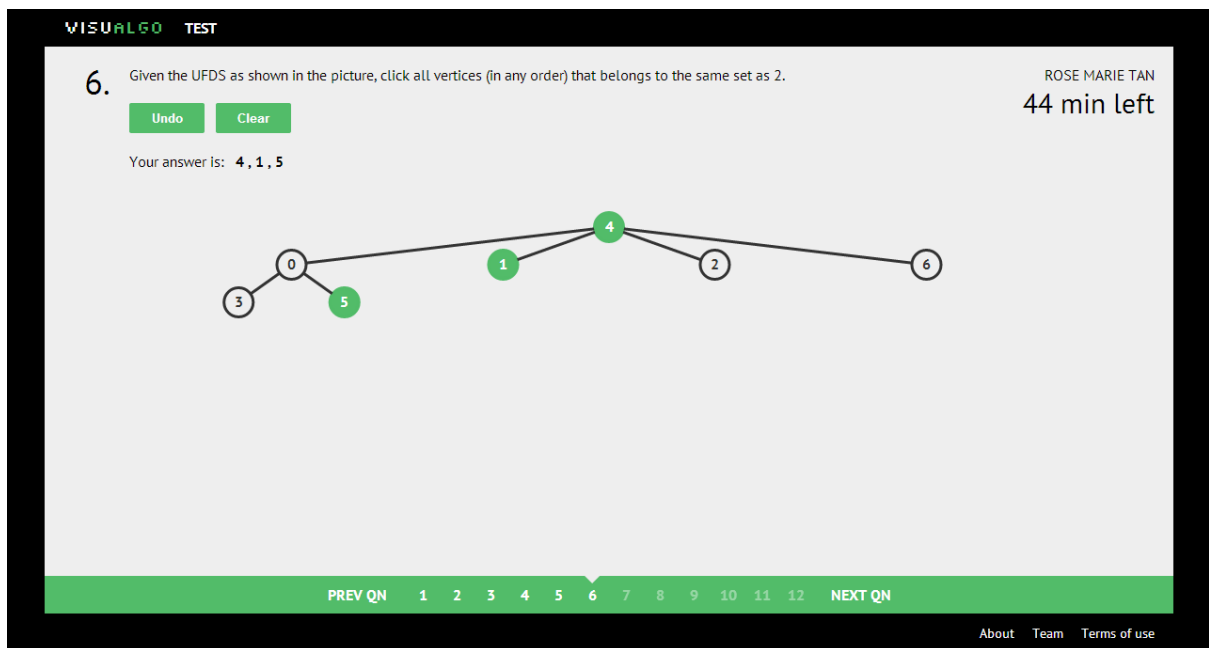


Figure 25: Test mode test screen

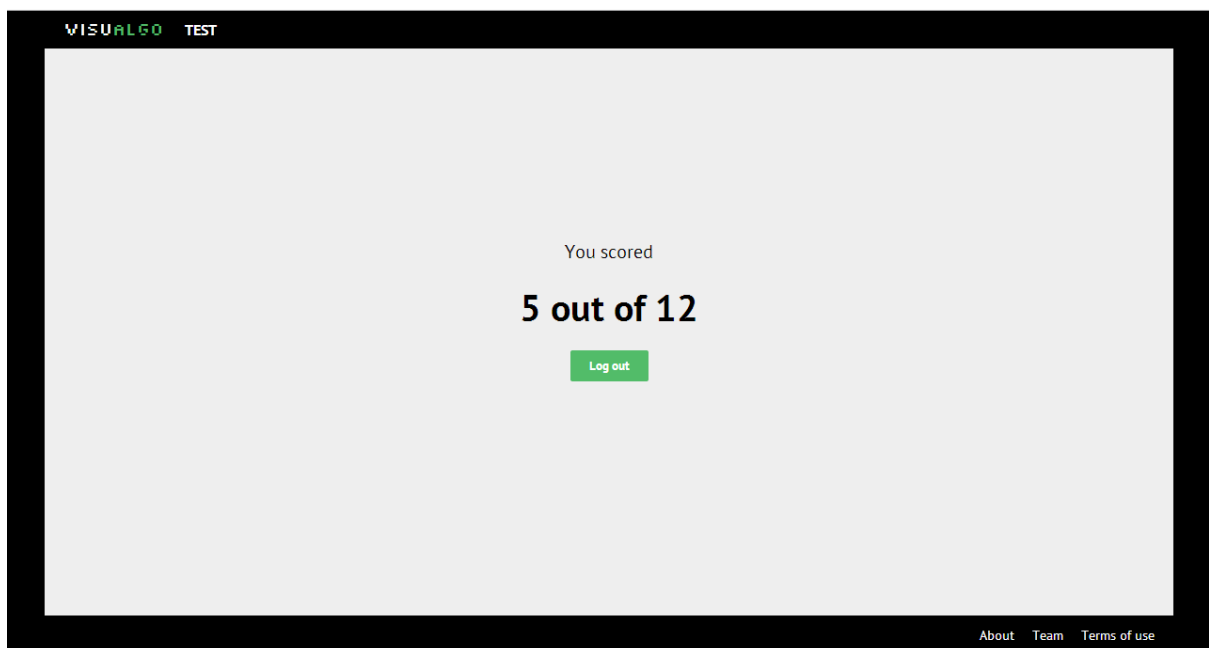


Figure 26: Test mode result screen

The answer key was modified slightly from its status for Online Quiz 2, as a scoreboard was made viewable to the students. It does not require login, and has been linked to the answer key page.

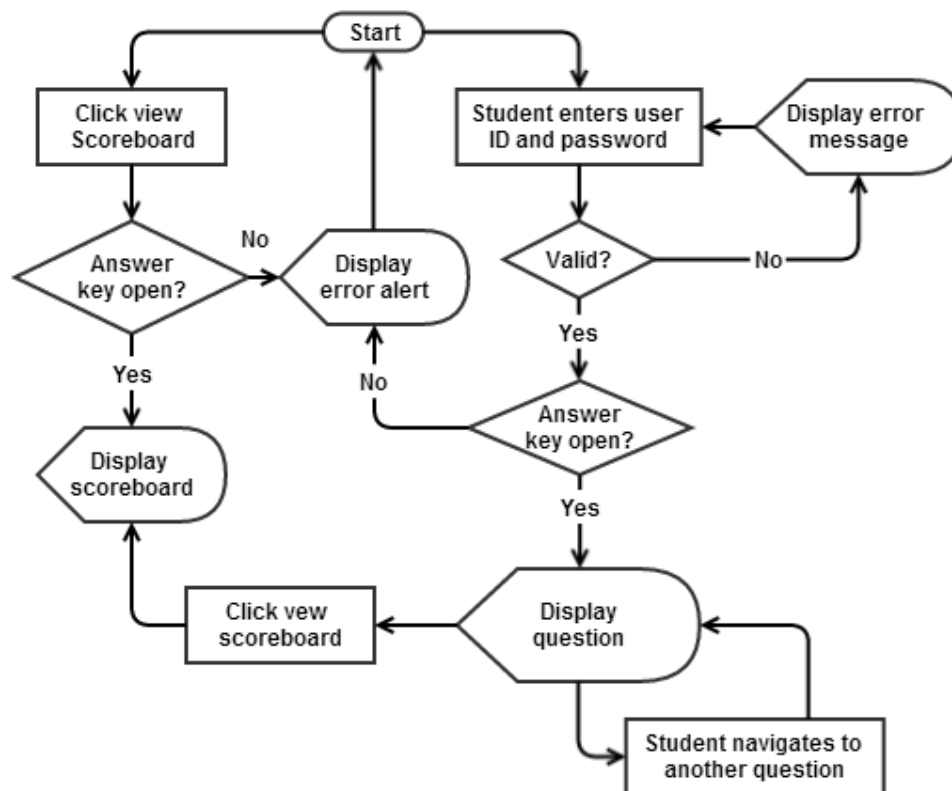


Figure 27: User workflow for answer key page

Figure 28: Answer key login screen

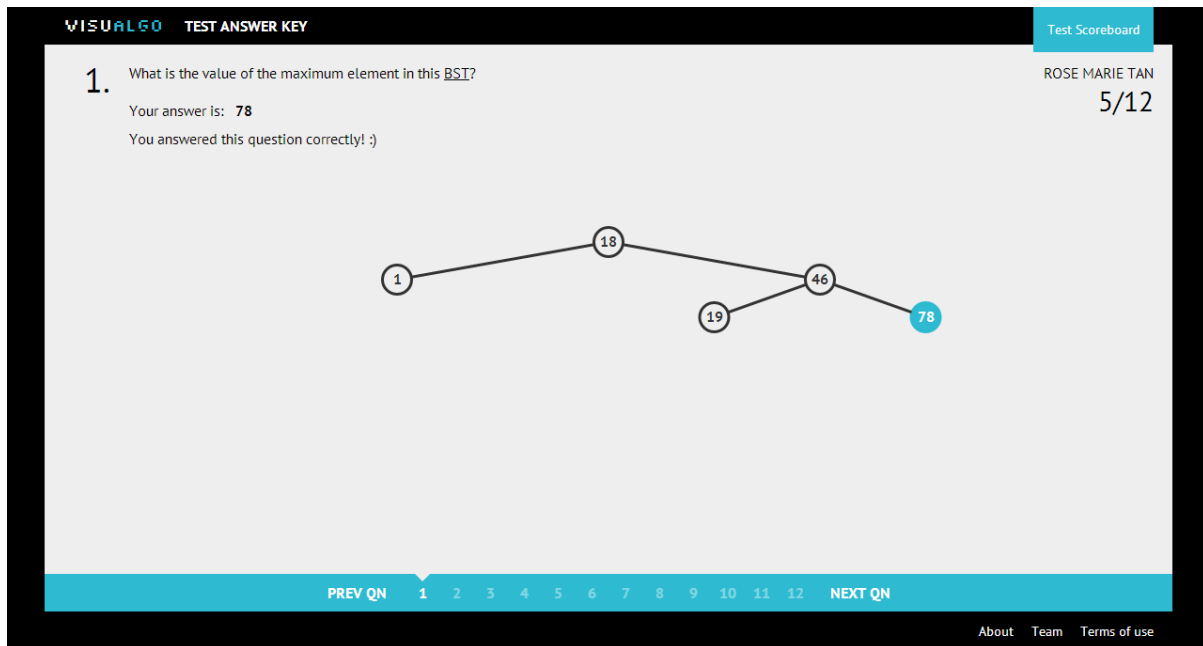


Figure 29: Answer key answer screen

VISUALGO TEST SCOREBOARD				
No.	Matric Number	Student Name	Score	Time Taken
1	A123*56*8	Student 4	11	1m 21s

Current average score: 3.20

Figure 30: Scoreboard page

For confidentiality purposes, full matriculation numbers are not shown. The list also only shows students who passed the test as well as the overall average. The list is ordered primarily by score, and secondarily by time taken.

Finally, the admin control panel was developed. The page allows the test administrator to open and close accessibility to the test and answer key, set the test parameters, included topics tested, time limit, and number of questions and to choose and try out the random set of questions based on a random seed. This page also allows the administrator to upload a list of students taking the test so that the system can use this list for test and answer key login verification. Finally, the administrator can also reset a student's attempt if necessary (as a last resort in the case of a glitch during the test for a particular student), and monitor the test results on an internal scoreboard. While the external scoreboard linked to the answer key only shows students who passed, this internal scoreboard shows the full list of students and their

scores. The default view is the scoreboard, as this screen is most important to the administrator during the test. As this page should only be accessible to the administrator, there are no links to this page from anywhere else in the VisuAlgo system. Only the administrator who knows the URL can access it. There is also a special admin login required.

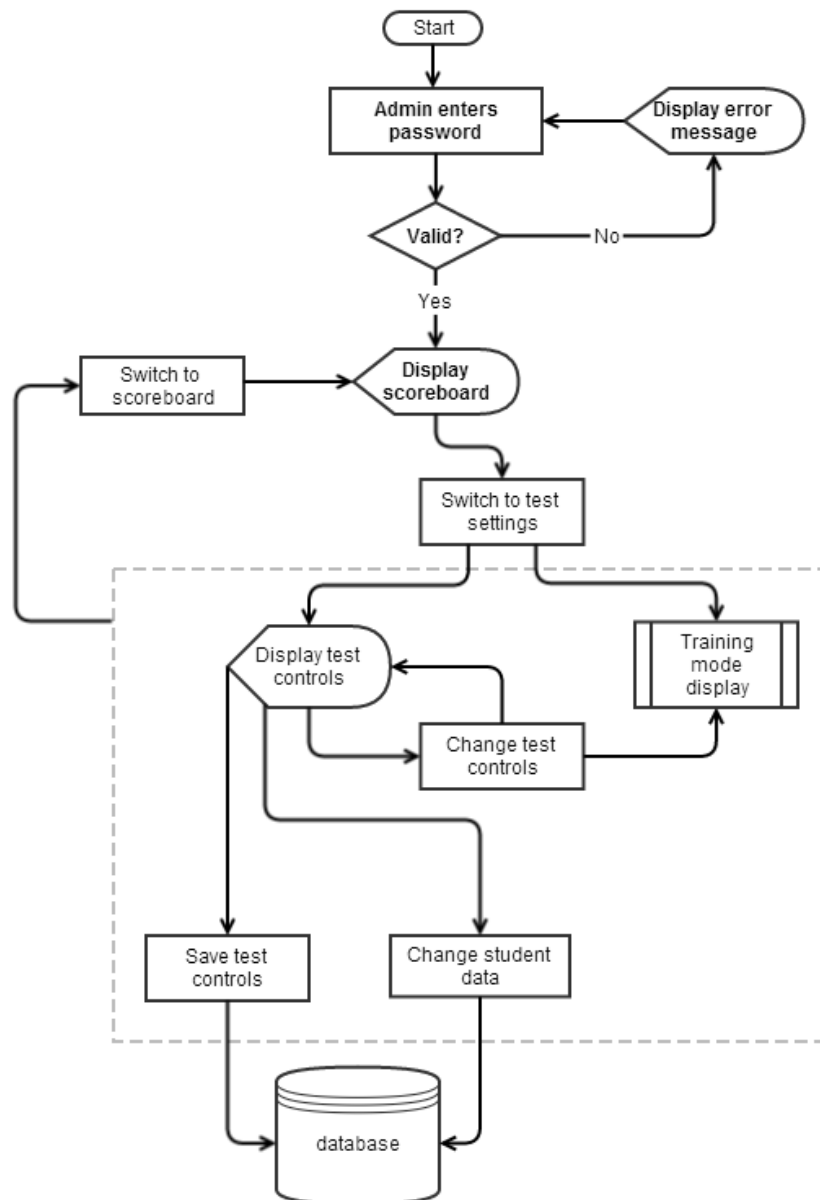


Figure 31: User workflow for admin control panel page



Figure 32: Control panel login screen

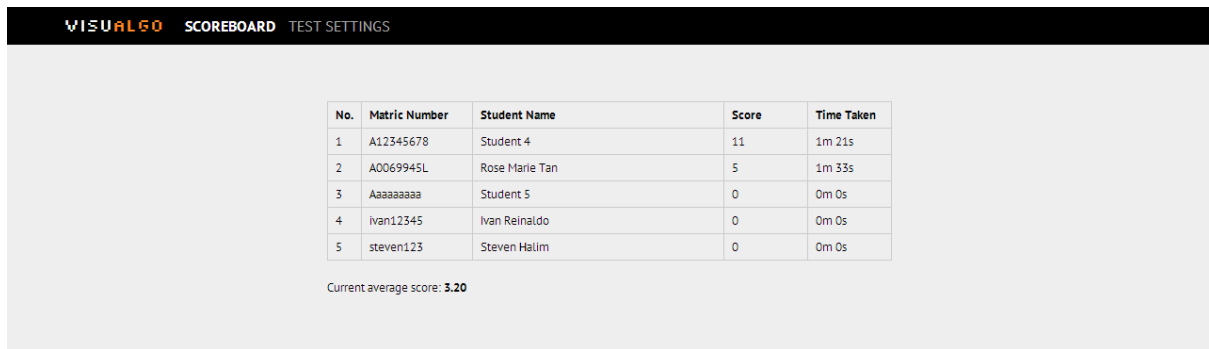
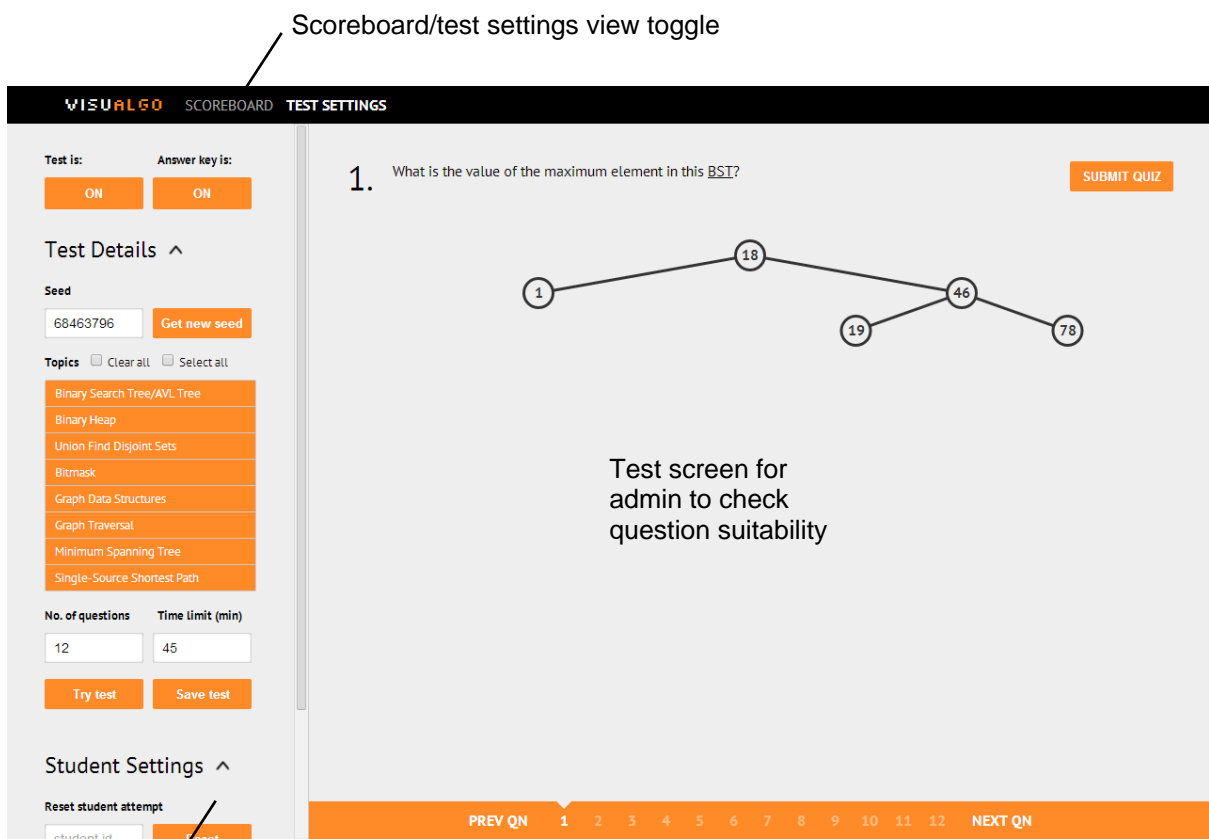


Figure 33: Control panel scoreboard screen



Test control settings and student access settings

Figure 34: Control panel test settings screen

Finally, the entire system was integrated, such that the applicable “Online Judge” components could be accessed from the homepage of the visualisation component. The training link is always available, but the Test and Answer Key links are only shown when each of them is open (as controlled by the administrator via the control panel). Both should not be available at the same time, but they are both shown here for demonstration purposes.

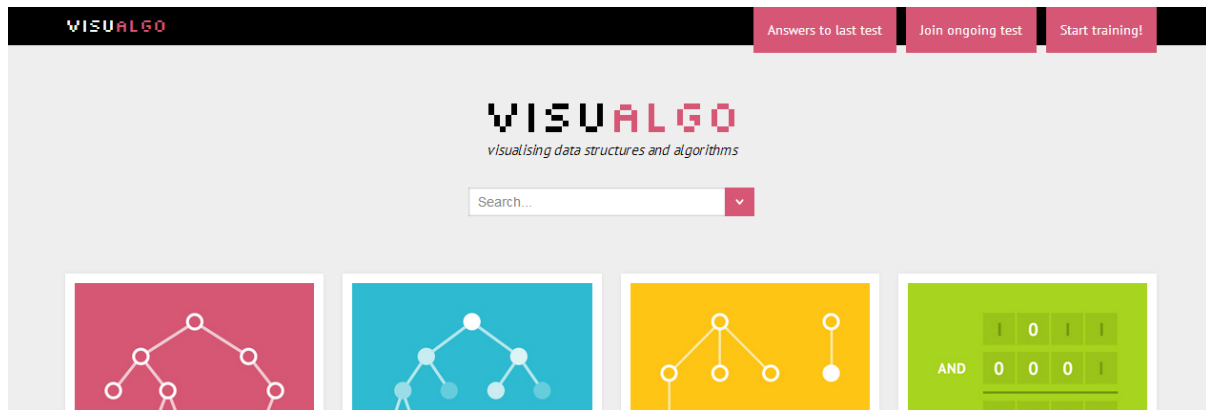


Figure 35: Home page with links to “Online Judge” component

7.3.3 Summary of Code and Control Structure

So far, the user workflow and UI/UX design have been shown and described. In this subsection, a short description of the front-end code structure is provided. For the full system architecture, refer to the project numbered H160080.

There are 5 separate HTML pages – the training mode, the real test, the answer key to the test, the scoreboard, and the Admin Control Panel. All the front-end logic is written in JavaScript, with DOM manipulation using the jQuery library. The training mode, test, answer key and control panel all share certain functionalities and styling, and thus all include a set of JavaScript and CSS files:

- 1) question_wordings.js: the full set of question text for every question in the question generators
- 2) test_mode_constant.js: synced with the PHP constants so that the client and server can communicate using constants, avoiding “magic number” situations
- 3) question_processing.js: parses and processes the information returned to the client by the server, and stores them in arrays for the interface logic to handle

- 4) `answer_interface.js`: the logic for displaying the different input interfaces for each type of user input (e.g. multiple vertices, multiple choice)
- 5) `test_common.js`: other miscellaneous shared functions, such as initialisation steps, and logic for the “undo” and “clear” buttons.
- 6) `test_common.css`: common stylings

However, since each of the pages has differences from each other as well (e.g. test requires login while training does not), each also has its own additional JavaScript and CSS files which contain page-specific styling and logic. These are `trainingmode.js` and `training.css`, `testmode.js` and `testmode.css`, `answerkey.js` and `answerkey.css`, `scoreboard.js` and `scoreboard.css`, and `controlpanel.js` and `controlpanel.css` respectively.

All question generators and graders are written in PHP, and all test settings and student test information are stored in a MySQL database. The interface interprets and records information such as students’ answers on the client-side, and sends these to the server using AJAX calls. It also retrieves validation permissions (for test, answer key and control panel), test configurations (for test, answer key and control panel), test questions (for training, test, answer key and control panel), answers to the test (for training, answer key and control panel), and test scores (training, test and control panel) via AJAX calls to the server. I was responsible for implementing the interface, doing client-side event-handling, as well as client-side communication and parsing of return data.

8 Conclusion

8.1 Evaluation

While the system has garnered a widely positive response through surveys conducted on the CS2010 class who used both the visualisation and “Online Judge” components of the system, more formal statistical usability studies have not been carried out to evaluate the usability of the system more deeply. This was unfortunately mostly due to time constraints and the lack of access to participants – during the full development of the “Online Judge” tool, Dr. Halim was not teaching the AY13/14 Semester 2 CS2010 class, thus we did not have access to a large enough sample of students to test the system.

Nevertheless, response to the interface so far has been very positive, with few usability issues raised especially with respect to the newly developed “Online Judge”.

Usability of the system was designed with close adherence to the Nielsen’s Heuristics:

8.1.1 Visibility

The title of the page on the top left indicates to the user where he/she is in within the entire VisuAlgo system, whether it be a visualisation topic (e.g. “Binary Heap”) or a page in the test component (e.g. “Training”). Within the visualisation component itself, the mode (exploration or tutorial) is always shown on the top right, and the current operation above the status description panel. On the test screens in the test system, users are informed of their location in the test by the arrow indicator on the question navigation bar. The opacity of the numbers on the navigation bar also indicates which questions have already been answered by the user. This, together with the timer shown, helps users gauge how much of the test they have completed and how they should apportion the remaining time to the remaining questions. The highlighting and printing of the user’s current answer for each question also acts as an indication of the current state of each question.

8.1.2 User control and Freedom

Within the visualisation tool, users can start a new operation at any time simply by selecting the new operation in the left panel. Within the test system, students are given the ability to navigate between questions in any order just as in a pen-and-paper test, giving him/her the freedom to answer the questions in any order. If the student makes a mistake in selecting the answer, the interface allows him/her to change the answer. The only prevention of full user control is the ability to submit the test before answering every question. However, this is done with good reason – as described earlier, in an actual test situation, the prevention of accidental submission is more important than full user freedom. The test will eventually submit when the time runs out even if some questions have not been answered. This is beneficial in another way as well – it encourages students not to give up, and to at least attempt each and every question. However, the submit button is enabled at all times in the

training mode, as it is not an actual test setting, and students may wish to progress directly to the answer key without answering the question when practising.

8.1.3 Consistency

The whole system has been designed to be internally consistent. This is why the visualisation component had to be designed before the “Online Judge” component. The overall look-and-feel is consistent, utilising the same design elements on all pages – the black framing, colour palette for buttons, panels and animation highlights, styling and interaction style for buttons, and so on. All visualisation pages, while supporting completely different operations, each with different function signatures, use an operations menu on the left that works in the same way. The playback controls on these pages are also consistent with widely used media controls, both in terms of icons used and keyboard shortcuts.

8.1.4 Error Prevention

In the “Online Judge” control panel, the administrator is prevented from modifying the test conditions while the test is running by a check. Note that the error messages are shown in a non-intrusive manner, as a pseudo-popup (actually an overlay) which fades away after a short time. Accidental submission in the test mode is also minimised by warning students with a similar pseudo popup. Students are similarly prevented from starting a training session without selecting any topics.

8.1.5 Recognition

On entering a visualisation page, the only panel open is the menu of operations. This is done to draw the attention of the user to the various operations immediately, so that users know what action they should take and what operations are available to them. The use of a consistent interface also aids in recognition rather than recall.

8.1.6 Flexibility and Efficiency

Novice users are able to control the playback on visualisation pages using the playback control buttons at the bottom of the screen. More advanced users, however, have the option of using keyboard shortcuts to control the playback more efficiently.

8.1.7 Minimalist design

Within the visualisation component, navigational links are kept to the frame of the screen, so that emphasis is given to the visualisation in the centre. No more information than is necessary is shown on all pages in the test component as well.

8.1.8 Error-recovery

If the user selects an operation that cannot be carried out due to incorrect selection of data structures (e.g. Breadth-First Search as a Single-Source Shortest Path algorithm on a weighted graph) or an irrelevant input (e.g. Prim's Minimum Spanning Tree algorithm from a vertex number that does not exist in the graph), the user is prompted with a non-intrusive error message next to the menu option that briefly indicates why the operation is not possible. The user is then able to correct the inputs to enable the operation.

8.1.9 Help

Within the visualisation tool, help is always provided via a tutorial mode that is accessible on every visualisation page.

That said, UX design is more than making a site easy to learn and easy to use – designing for the user's enjoyment in using the tool is an important goal as well (Badre, 2002). The system has achieved this by emphasising not just function but form as well. The aesthetics of the website has been enhanced by the use of colours throughout, pleasant but not distracting sliding and fading transitions, as well as adherence to the Gestalt laws, such as the law of proximity in placing the operation name, status description and codetrace panels together on the right as they share a similar function of explaining the state of the operation.

8.2 Recommendations for Future Development

While the improvement of the visualisation tool and the development of the “Online Judge” tool have shown good results, they only form the basis for a much larger system with stronger capabilities. The entire system can be strengthened in a number of ways:

8.2.1 More Detailed Usability Testing

As acknowledged earlier, statistical analysis has not been carried out to fully evaluate the system. To maximise the benefits of the system to the students, it is advisable that more detailed usability testing be carried out.

8.2.2 Supplementing Visualisation and Question Generator libraries

About half of the visualisations developed before June 2013 have not been converted to use the new SVG graph library and interface. These need to be migrated to the new system so that the entire visualisation component will be cohesive and up-to-date. New visualisations can also be added to make the tool more extensive (e.g. adding the Red-Black Tree data structure to the Binary Search Tree page). More questions for more topics should also be developed for the “Online Judge” tool to make it more comprehensive and allow for greater variation in question generation.

8.2.3 Re-incorporating graph drawing capabilities

While the original HTML5 visualisation system supported certain graph drawing capabilities, the new system does not fully support that functionality as the graph drawing tool had to be re-developed to use the new graph library. This graph-drawing functionality has been developed by a fellow project member, however, the presence of some remaining bugs and artefacts has prevented it from being released as part of the current system. These problems should be fixed, and the graph drawing functionality incorporated into all the graph algorithm topic pages, as well as integrated with the test system so that user input can include graph drawing. This will allow for many more interesting questions to be included in the question generators, giving the system in stronger testing capabilities.

8.2.4 Improving Learnability

While learnability of the visualisation tool has been improved by including short tutorials, the brief descriptor of the data structure or algorithm in the current tutorial style is likely not sufficient for users who are completely new to data structures and algorithms, especially in the case of the more advanced algorithms. Therefore, more can be done to improve this, such as providing a much more detailed description of the topic, and including an option to allow hints during the animations themselves.

8.2.5 Improving Test Security

While the system architecture of the current test component incorporates several strategies to minimise the possibility of students cheating, there are still security loopholes that may be breached. In particular, the client-server AJAX communication and database need to be secured more strongly. It is important to improve the security of the “Online Judge”, as the test results go towards final module grades and should not be tampered with.

8.2.6 Better Question Calibration

Currently, the questions generated by the automated generators are fairly random, meaning that certain questions can be very simple if the generated data structure is a very simple one (e.g. asking for the Binary Search Tree search sequence on a single-node tree). This is not ideal, as the purpose of testing the question is lost. Therefore, effort should be invested into improving the random generators so that the questions generated are always meaningful.

8.2.7 Greater Admin Control

It is also recommended that more detailed options be made available to the administrator via the control panel so that the test can be calibrated exactly to suit the assessment goals. For example, allowing the administrator to specify the number of questions or the level of difficulty for each topic.

References

- Badre, A. (2002). *Shaping Web usability: interaction design in context*. Addison-Wesley Professional.
- Benford, S.D., Burke, E.K., Foxley, E. and Higgins, C.A. (1994). Ceilidh: A courseware system for the assessment and administration of computer programming courses in higher education. In *Proceedings of the Interdisciplinary Workshop on Complex Learning in Computer Environments*.
- Fleming, N.D. (2006). *Teaching and learning styles: VARK strategies*. ND Fleming.
- Graham, L. (2008). Gestalt theory in interactive media design. *Journal of Humanities & Social Sciences*, Vol.2, No.1, 2008.
- Guy, R.S. and Lownes-Jackson, M. (2012). Assessing the Effectiveness of Web-Based Tutorials Using Pre-and Post-Test Measurements. *Interdisciplinary Journal of E-Learning & Learning Objects*, 8.
- Hall, R.H. and Hanna, P. (2004). The impact of web page text-background colour combinations on readability, retention, aesthetics and behavioural intention. *Behaviour & information technology*, Vol.23, No.3, 2004, pp. 183-195.
- Halim, S. and Halim, F. (2013). *Competitive Programming 3 The New Lower Bound of Programming Contests*. Lulu, Singapore, 2013.
- Halim, S., Koh, Z.C., Loh, V.B.H. and Halim, F. (2012). Learning Algorithms with Unified and Interactive Web-Based Visualization. *Olympiads in Informatics*, 6.
- Markoff, J. (2013, April 4). Essay-Grading Software Offers Professors a Break. *The New York Times*. Retrieved from <http://www.nytimes.com/2013/04/05/science/new-test-for-computers-grading-essays-at-college-level.html>
- Nielsen, J. and Molich, R. (1990, March). Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (pp. 249-256). ACM.
- Nikander, J., Korhonen, A., Seppälä, O., Karavirta, V., Silvasti, P. and Malmi, L. (2004). Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. *Informatics in Education-An International Journal*, Vol.3, No.2, pp. 267 – 288.
- Parr, C. (2013, May 9). Mooc completion rates ‘below 7%’. *Times Higher Education*. Retrieved from <http://www.timeshighereducation.co.uk/news/mooc-completion-rates-below-7/2003710.article>

Solomon, E.A. (2013, February 8). MOOCs: A Review. *The Tech*, 133 (2). Retrieved from <http://tech.mit.edu/V133/N2/mooc.html>

Ullrich, T. and Fellner, D. (2004). AlgoViz-a computer graphics algorithm visualization toolkit. In World Conference on Educational Multimedia, Hypermedia and Telecommunications, Vol.2004, No.1, pp. 941-948.

University of Virginia (2013, June 4). Strategy to clean up cheating in online courses. *ScienceDaily*. Retrieved from <http://www.sciencedaily.com/releases/2013/06/130604153342.htm>

Appendix A: Selected Feedback from Online Quiz 1

Source: CS2010 AY13/14 Semester 1 IVLE Mid-semester Survey

Total number of respondents: 97

Note that only responses relating to the user interface have been included here for brevity.

Likert Scale Questions

Question	Strongly disagree	Disagree	Neutral	Agree	Strongly agree
I like the online quiz format.	2 (2%)	9 (9.2%)	14 (14.4%)	42 (43.2%)	30 (30.9%)
The online test runs smoothly.	3 (3%)	6 (6.1%)	7 (7.2%)	56 (57.7%)	25 (25.7%)
The user interface is user friendly.	2 (2%)	6 (6.1%)	9 (9.2%)	53 (54.6%)	27 (27.8%)
I am fine with the idea that parts of my module grade is graded by a machine.	2 (2%)	0 (0%)	16 (16.4%)	40 (41.2%)	38 (39.1%)

Positive remarks

The interface looked cute. love the color:)
- very beautiful interface - there UI is very smooth - it gives us nice options, e.g. going back to a previous question with our answers highlighted - auto submission once time is up
-Easier to visualize
clear diagrams, user friendly interface. Fun and interesting way to take a quiz
The experience is smooth.
I like the user interface.
I think the system is very usable. The illustration is clear.
It is a fast and simple test system.
Amazing. Efficient. System allows you to spend more time on harder ones if you can finish the easy ones quickly.
The idea of using is clear whereby we just need to click on the part we feel it is correct. User interface is friendly and can find all the necessary button easily
The transition is nice. The overall colour scheme used is comfortable for my eyes
Instantaneous score feedback.
Simple and easy to use interface
Good for testing basic understanding
Good experience and forced to think fast. Nice UI

The interface is nice and the process was smooth.
Very well developed and good UI. Illustrations were clear. Positions of UI was appropriate. Timer was the right size, not causing any distractions. Saves a lot of time.
The interface is indeed impressive and runs very smoothly throughout the entire quiz.
I like the simplicity of just clicking the nodes in the diagram!
The count down timer is good. Let's us pace ourselves appropriately for each question.
The UI is nice and gives a better presentation than paper test
Fast marking, clear questions and answer format
<ul style="list-style-type: none"> - More beautiful than IVLE. - Automated grading.
Immediate marking and scores. System was clear and user-friendly. It was easy to understand and perform the quiz.
Easy to use
Clean and simple UI. The visualization tool is clearer than the one printed on the past year papers.
Its easy to use.
<ul style="list-style-type: none"> - faster to do - simpler - less tension - less stress due to using something you comfortable with (PC s)
Easy to use, just clicking the mouse (compared to written quiz). I like the visualisation graphics.
Nice user interface
it is very user-friendly and easy to use. it saved us a lot of time compared to writing or typing.
Interactive and more visual
user friendly
Very clean design, it is quite clear how we should answer the questions.
The whole test is intuitive enough.
Immediate result if good - no tension after exam.
The style of this test is simple and easy to understand. Also good that we can have immediate feedback.
easier for marking.. more interactive
The UI is really good. Simple enough to use. I think we should have more implementation of online quiz.
clear, easy to use.

The application itself has a really neat UI, and the quiz made full use of it, and I think it looked really great.

Love love the UI. Super nice design

Negative remarks

the moment I logged in the timer started, maybe we can have a button which says click to start

Maybe there could a practice version to help students speed up the process of doing quizzes.

Should not have a submit button right at the start of the quiz, instead make it appear only after all questions are attempted.

The colour of the text used in the questions can be different to make it more obvious and easier to read. The submit button can be placed somewhere else like on the same level as the question navigation to avoid unnecessary distractions.

Was abit confused about the UI at the start; clicked "submit" instead of the next question and submitted the quiz on the first try.

submit button is misleading.

The navigation buttons may not be so obvious. I ended up pressing the numbers to go to the next page because I couldn't find the 'next page' button and since it was graded I didn't want to risk pressing the wrong buttons.

Question text could be made larger and of black color.

Position of the submit button is in the wrong place. For the first few question my mouse instinctively thought that it is the next button and almost clicked on it. Perhaps put it in the top right corner.

I think the "submit" button was slightly confusing. I didn't know (but could guess anyway) if the submit button is meant for submission of the answer for a particular question, or for the whole test.

Counting down by the minute instead of second might alleviate some undue stress while taking the test. Can consider counting down by the second in the last minute or last 30 seconds.

-Is it possible to change the countdown ticker to another colour (Red??) maybe in the last 10 seconds, students(me) who need every second of the 10 mins might lose track of time.

I think it was left unclear whether to drag or to click.. for the first few minutes, hence there was time wastage. Otherwise, nothing else.

Providing practice sessions or opening up the quiz after the test would be great.

The submit button is a bit less intuitive, almost wanted to click on it to go to the next question, maybe changed it to next instead and then, the last question is "submit".

the next qns button can be better adjusted to be at the submit button as after qns 1 user will normally shift their mouse to the right to click on next qns

The submit button is placed in a position where users would intuitively thought it is lead to the next question.

Appendix B: Selected CS2010 Teaching Feedback

Source: CS2010 AY13/14 Semester 1 Teaching Feedback

Total number of respondents: 125

Note that only responses relating to the visualisation and “Online Judge” tool have been included here for relevance.

The best aspect of this module is:

Very fun online quizzes - Very nice visualizations, very good effort :D
Online quizzes are well done
Visualization tool is a great step forward for learning.
The new teaching aid gives a lovely visual walkthrough on how the algorithms covered work. Excellent utility. The module also uses the above teaching tool to help facilitate electronic examinations, a welcomed improvement that should have been done a long time ago. Quite frankly, as a computing society (let alone a SCHOOL of computing), we should be moving away from pen-and-paper based examinations. Other modules, such as CS1010 and CS1020, should utilize this tool to help teach their own respective data structures and algorithms.
Visualization systems to explain algorithms. It is difficult to understand an abstract concept without it, especially those who are really green.
Visualisation tools help a lot. Hope it's completed by next semester
online quizzes are unique and a nice addition.
An accompanying website for us to use to understand the content better and test ourselves
Visualization tool is extremely well made and useful for self-study.

This module could benefit most by:

The phrasing of certain questions in the online quiz could be improved, because of limited time, it is easy to misinterpret the questions and because of an unfamiliar format (For each time, quiz 1 and quiz 2 the format is really different for the student that is taking the exam. Not only in terms of the interface, but the overall feel. A minor change in the interface, coupled with new types/ ways the questions are phrased can make a huge difference.), some who are slower to adapt might be at a disadvantage. I suggest giving a mock/ practice quiz for each quiz before having the actual quiz on another day. Maybe you could release our year's questions for them to practice on and to get a feel for the format of the questions and the interface, so that they could focus on answering the questions itself, instead of trying to get used to the medium (the online quiz system) on the day itself. Afterall, main purpose of online quiz is to test knowledge on basics, not to test one's ability to get used to a foreign system of assessment. ⁶
A complete visualisation system
finishing the visualization tool

⁶ This problem has been addressed with the inclusion of a new Training mode.

Appendix C: Internal Interfaces for All Input Types

1. Single vertex

VISUALGO TRAINING

3. Click the root of this heap.

SUBMIT QUIZ

```
graph TD; 92((92)) --- 83((83)); 92 --- 65((65)); 83 --- 16((16));
```

2. Single edge

VISUALGO TRAINING

8. Click the edge that has the maximum edge weight along the minimax path from vertex 5 to vertex 2. The minimax path between two vertices is defined as the path that minimizes the maximum edge weight between the two vertices.

SUBMIT QUIZ

```
graph LR; 2((2)) ---|6| 1((1)); 1 ---|44| 0((0)); 0 ---|26| 5((5)); 0 ---|50| 3((3)); 3 ---|35| 1; 4((4)) ---|11| 2; 4 ---|56| 6((6));
```

3. Multiple vertices

VISUALGO TRAINING

6. Click the sequence of vertices that are visited by BFS from source vertex 8. The neighbours of a vertex are listed in ascending vertex number.

SUBMIT QUIZ

Undo Clear

Your answer is: 8, 0, 1, 3, 5, 9

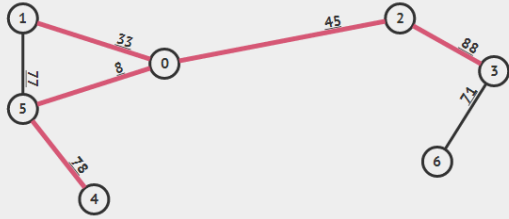
```
graph LR; 9((9)) ---|1| 5((5)); 5 ---|1| 0((0)); 0 ---|1| 8((8)); 8 ---|1| 1((1)); 0 ---|1| 3((3)); 3 ---|1| 5; 6((6)) ---|1| 7((7)); 2((2)) ---|1| 4((4)); 0 ---|1| 2;
```

4. Multiple edges

VISUALGO TRAINING

1. Given the undirected weighted graph as shown in the picture, click the first 5 edges in the sequence of edges that are added to the Minimum Spanning Tree by Prim's algorithm starting at vertex 4.

Your answer is: $(5, 4), (5, 0), (1, 0), (0, 2), (2, 3)$



5. Multiple choice question

VISUALGO TRAINING

7. Is the graph in the picture a valid AVL?

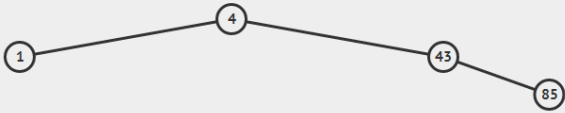
☒ Valid
☐ Invalid



6. Number input

VISUALGO TRAINING

3. What is the height of this BST?



No answer option

VISUALGO TRAINING

5. Click all vertices (in any order) that will cause the graph to be disconnected if deleted. Note that the vertex deletion is independent.

☐ No answer

Appendix D: Internal Interfaces for All Answer States in Answer Key

Question unanswered

VISUALGO TRAINING

6. Given the undirected weighted graph as shown in the picture, click the first 5 edges in the sequence of edges that are added to the Minimum Spanning Tree by Prim's algorithm starting at vertex 4.

You did not answer this question.

The correct answer is: $(4, 0)$, $(0, 2)$, $(2, 5)$, $(0, 3)$, $(0, 1)$

Correct Answer

VISUALGO TRAINING

1. What is the value of the minimum element in this BST?

Your answer is: **33**

You answered this question correctly! ;)

Incorrect Answer

VISUALGO TRAINING

2. Click all the leaf vertices (in any order) of this BST.

Your answer is: **31, 84, 98**

The correct answer is: **33, 84, 98**