VisuSort

Report

Prepared by VisuCorp (Team F3):

**Dogukan Baykan, Hannah Evans, Vicci Garner,
Seb Lewis, David Marlow and Phillip Oliver**

Table of Contents

1. Introduction

1.1 - Product Description

The VisuSort software visualises 4 different sorting algorithms; *Bubble sort*, *Insertion sort*, *Selection sort* and *Quicksort*. These will be contained separately in the program with the intent of showing the user how each of these algorithms work and in effect teaching them how to code something to achieve these methods. The visualisation and human computer interaction components integrated into the software which will be outlined later have been our focus in achieving this goal. VisuSort is targeted at first year computer science students learning sorting algorithms and we believe the software have been tailored specifically to these constraints.

Alongside teaching students, VisuSort offers a testing platform for helping to help students test their own knowledge of the sorting methods as well as be able to have their progress monitored by their tutors. This has been done via the use of an SQL database and multiple servers to translate a user's input into it as well as being a host for their user accounts. This allows users to register, log in and also view their grades within the software.

The visualisation component has been the main focus of our team and the final product represents our idea of what the best learning tool would be for these first year students. The design is simple and easy to follow with distinct colours but, with the way we have designed the software, it allows the user to increase the complexity of the visualisation.

1.2 - Report Description

The report that follows aims to describe the process in which we came about creating our software and the justifications behind our design decisions including our own opinions of how the team project has worked and reviewing our efficiency as a team throughout the development of VisuSort.
We will cover the following things:

- Software and Visualisation design
- HCI design
- Software engineering / Risk Management
- Evaluating of our software
- Evaluation of our teamwork
- Summary of our Project

2. Software Design

Looking at the design of our software, we must consider each component of the software as its individual components and evaluate each. These major components of our software are:

- Learning page
- Testing and HCI
- Server and database design

Each of these has had to be individually analysed for what will work best when it comes to creating the best product possible and has been a focus for the team ongoing throughout the project.

2.1 - Learning Page Design

The learning page is where the task of visualising an algorithm takes place. Everything about the design here is designed to be as easy to use as possible with clear and distinct buttons with clear purposes with the intention of making the software as simple as possible to use even for those who have, for instance, learning difficulties. Our original design, after much feedback, we discovered did not achieve this with too many buttons and text which was too difficult to read. We learnt a lot about the design needed for the page through this with buttons having a clear purpose and switching buttons such as *Pause* and *Unpause* for our visualisation that reduced the number of buttons we needed and this reflected well with user feedback.

The bulk of the page is taken by the visualisation, the main learning tool. Even without the text beside it, a user should be able to understand the algorithms well enough to describe their behaviour with this alone. However, we felt it was necessary for each algorithm to have a text based description showing the nuances of the sorting next to it.

The colour scheme within the page (without considering the visualisation) is set out to be as easy to read as possible. Initially, black text was used throughout the page but we quickly realised the blue theme of the VisuSort logo was too dark to have a high enough contrast to be easy to use. This meant we had to reconsider our use of colour which early on led us to a program wide colour palette. White text would be used atop and of the darker blue backgrounds and for any buttons or text screens we would have, a black font and a lighter shade of blue would be used to maintain an easy reading experience.

2.2 - Testing Page Design

The testing page of VisuSort is where the user will test their knowledge of sorting from what they've learned using the product; the design here stays with the theme of visual learning through reducing text and increasing interactivity.

The testing section is one JFrame with 5 question buttons that display the relevant question JPanels and a submit button that submits the score once all questions have been completed. The design of this page is consistent with the rest of the system in terms of the colour scheme, font, and logo.

For each question there is a question label at the top of the page displaying the question, and in the panel below is the actual question content. For questions 1 and 4 there are images/gifs of a particular algorithm with multiple choice answers below with radio buttons to make it clear what the user has to select. For question 2 the panel consists of text boxes and inputtable text fields for the purpose of filling in the missing stages, this was deliberately kept simple so that any ambiguity in how the question was to be answered was minimised. Question 3 of the tests is arguably the most interactive of the questions as it includes a drag and drop functionality for completing the answer- the panel is a little small for this question so a scroll bar had to be implemented which allows access to all of the question content, it isn't ideal but it still works reasonably well. Question 5 is again fairly simple consisting of images that the user has to number in the correct order, the first one is filled in to show the user the starting point, the rest of the boxes are clearly formatted as inputtable text fields so the user knows where to input their answer.

The JPanel for the question content was used to enable the overall layout and important buttons to remain accessible at all times- the Main Menu and Help buttons are there for user support and navigation while the individual question buttons allow the user to switch between questions before submitting their final answers. The submit button is purposely formatted to be greyed out and not selectable until all questions have been answered.

On the left hand side of the page is the progress information to tell the user how much of the test they have attempted so far and also in the case that they have previously taken the test they can see their previous best score. This is to encourage the user to continue and also improve on their previous attempts.

2.3 - Server / DataBase Design

This component of the program exists separately to the software and can be hosted on any machine dependant on the fact there is a correctly managed network interface (ie, all necessary ports are forwarded). There are 4 servers that run separately to manage traffic sent by the software and these run on the ports 25000-25003. We have chosen to implement the program like this to make maintenance and also the development process much easier for ourselves. When we want to amend one piece of code, all servers need not go down but just the server responsible for that traffic. These 4 servers can be separated into the login server, register server, admin server and test result submission server. The security needs for each have been considered and as such we can say with confidence that the server is secure. Our own attacks have been run on these as well to ensure this is the case including SQL injection, packet sniffing and brute forcing things such as password or session keys. What has been kept true across the software however is that any calls to the SQL table are through prepared statements to mitigate the chance of an SQL injection attack.

The security we have implemented is different for each of the protocols set out. For the register server, the password transmission is sent encrypted with its own 128 bit hex key, which will be kept secret, but only after it has been hashed using SHA-256. As such, we never actually at the server level will receive the raw plaintext of the password so if our database is compromised it will not be plainly readable. We do notice however that SHA-256 is generally not a great hashing system for passwords and can be brute forced. If we felt the software was of crucial importance to not be compromised, we would look into using a more intense hashing algorithm such as PBKDF2 or S-crypt. The security aspects of this section are relatively low though due to the lack of any data being returned.

The login server has been scrutinised more closely when it comes to security. Once again, passwords are being sent hashed then encrypted with a different 128 bit hex key. We have also used a preventative measure to prevent repeat attacks being used against us via the use of an encrypted random number transmitted from the server to the client. This will be decrypted, incremented then sent back to the server and if the number has been successfully incremented, then we know that the client is legitimate and the password has not been intercepted. Once logged in, the user receives a session key that changes every time they login. This will be used to verify the connection from this point on which too is encrypted when transmitted. We see these measures as suitably secure to protect the security of the users.

The admin server has been secured in a similar way but with the security check being on the session key and not the user's password. Once again we use a specific hex key to the admin server to verify the session as well as repeating this encrypted number verification technique to verify the source of the connection. If these measures have been met, we assume the user has a valid connection. The only way a user can access the admin check is by ensuring the user's permission level is above a certain value preventing unexpected access to other user's results and administrative features.

Finally, we have the results submission server. For this, the security once again comes from the encryption of the session key ensuring that the user is who they claim to be. Their result score will also be computed by sending a random encrypted integer to the client who is then to decrypt the message and send it back with their score added on top of this. This is in an attempt to prevent scores being intercepted and changed in the delivery process. As a result, we believe the server and the results on the other end are secure.

The SQL Database itself is comprised of a single table due to the fact that we are not storing that many columns and the table by itself is not difficult to read. We did consider separating the score and attempts columns, which are self explanatory, into their own table to keep the tables more consistent in theme but decided against this due to unnecessary complexity.

3. Visualisation Design and HCI

3.1 - Class Structure

This part of the software was the most likely part to have modifications made and new functionalities implemented over time. As such we had to use a robust, flexible class structure that enabled multiple members to make changes to some part of the code without affecting the rest.

The sorting classes are given the array of numbers to be sorted and they produce two arraylists, the first one contains position of numbers for every step and the second one holds the colour information for those steps. This arrangement makes it easier to test correctness of algorithm outputs when doing unit tests and separates program logic from the actual visualization.
Producing an arraylist of all steps instead of individual steps also makes slider implementation easier. All the sorting algorithms implement a sorting interface which defines abstract methods such as sort, swap and change colours

The sorting classes are used by the our sorting thread which loops through the steps and invokes the method for updating the screen for each one of the arrays.
Use of a separate thread for sorting makes it trivial to implement start/pause/stop operations as well as change visualization speed.

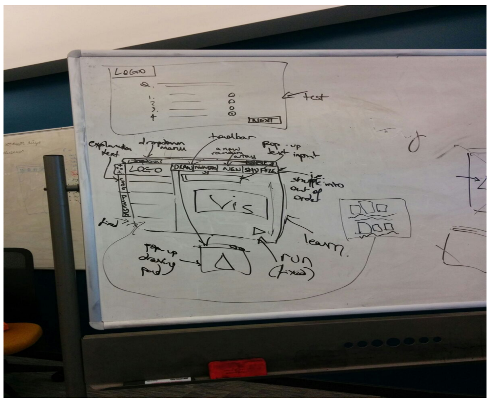
The method for updating screen passes the current number array and colour array to PaintPanel which does drawing/painting of the bars and background

The main class instantiates the other classes and stores variables.It also contains all the GUI elements as well as listeners and get/set methods.

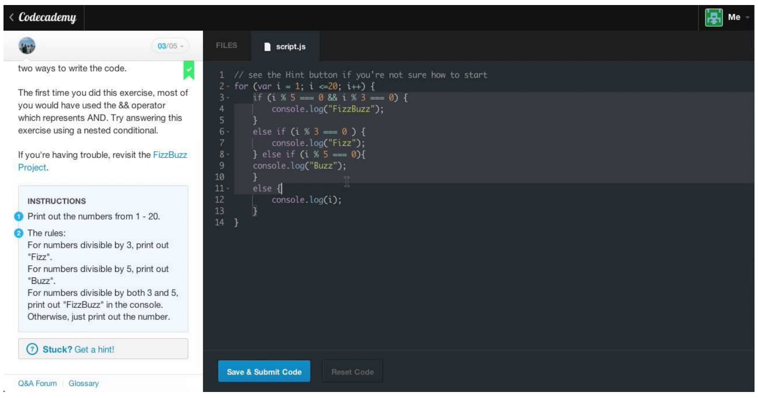
3.2 - GUI

As we were developing a piece of visualisation software the GUI was, naturally, a significant part of the project and went through several iterations following feedback from peers, team members and feedback from our tutor Xiaodong. Our target audience of school and university-age students not necessarily well-acquainted with computers also meant that the GUI had to be user-friendly and fairly self explanatory.

An early lo-fi prototype[1] of the visualisation is included here (below, left). We took inspiration from the design of online programming school Codecademy's lessons, where educational text is separated from an interactive/visual learning area (below, right). This was similar to our initial design, as included in the requirements specification.



A lo-fi prototype produced in a project meeting



two ways to write the code.

The first time you did this exercise, most of you would have used the && operator which represents AND. Try answering this exercise using a nested conditional.

If you're having trouble, revisit the FizzBuzz Project.

INSTRUCTIONS

1. Print out the numbers from 1 - 20.
2. The rules:
For numbers divisible by 3, print out "Fizz".
For numbers divisible by 5, print out "Buzz".
For numbers divisible by both 3 and 5, print out "FizzBuzz" in the console.
Otherwise, just print out the number.

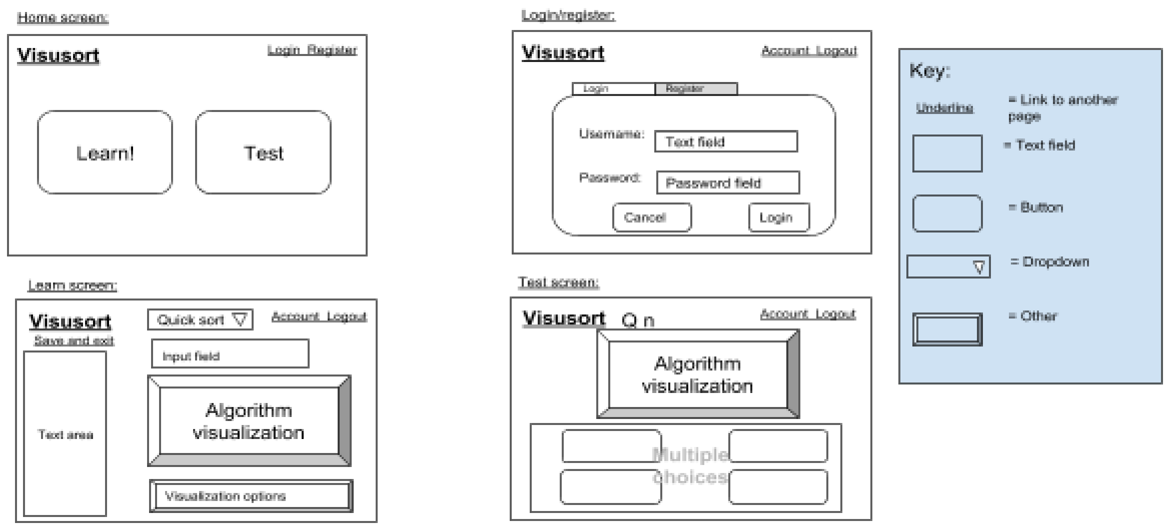
Stuck? Get a hint!

Q&A Forum | Glossary

```
// see the Hint button if you're not sure how to start
for (var i = 1; i <=20; i++) {
    if (i % 5 === 0 && i % 3 === 0) {
        console.log("FizzBuzz");
    }
    else if (i % 3 === 0 ) {
        console.log("Fizz");
    } else if (i % 5 === 0){
    console.log("Buzz");
    }
    else {
        console.log(i);
    }
}
```

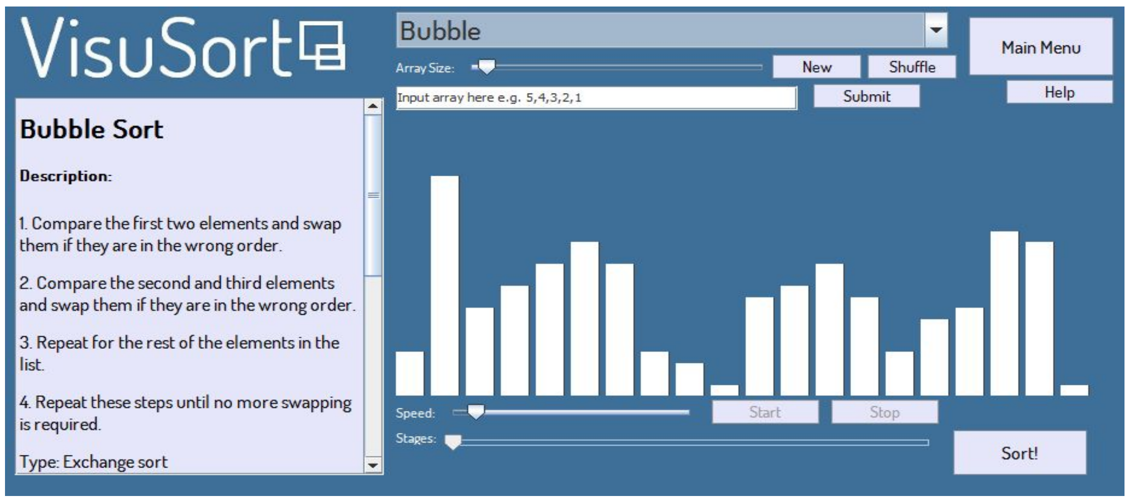
The layout of a Codecademy online lesson, from which we took inspiration[2]

[1] http://www.telono.com/en/articles/lo-fi-vs-hi-fi-prototyping-how-real-does-the-real-thing-have-to-be/
[2] Captured at http://www.codecademy.com



Initial design in the requirement specification

Although the design for the learn screen is slightly different to the implemented version (seen below) the general layout and design do retain many features devised during our initial brainstorming sessions.



Bubble Sort

Description:

1. Compare the first two elements and swap them if they are in the wrong order.

2. Compare the second and third elements and swap them if they are in the wrong order.

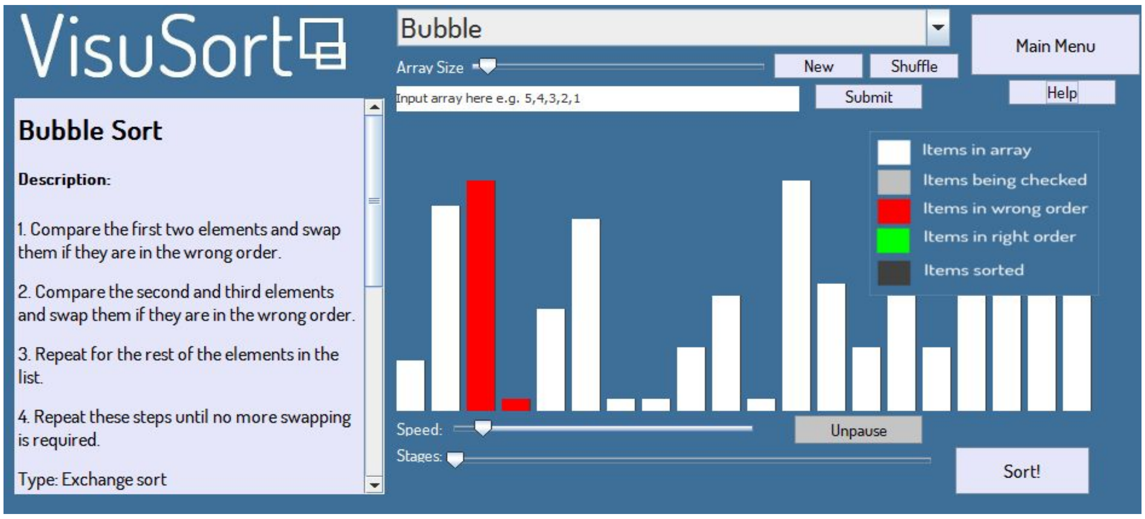
3. Repeat for the rest of the elements in the list.

4. Repeat these steps until no more swapping is required.

Type: Exchange sort

First static implementation

The next feature implemented stemmed from the feedback received from Xiaodong and also test users. After showing the above visualisation to Xiaodong he explained to us that as someone who did not know very much about sorting (our target audience) our visualisation was still quite confusing. Along with this feedback we received from our feedback form, people believed this could be aimed at a younger target audience ‘teenagers studying computing in school’ rather than our original aim of providing a service for first year undergraduates studying computer science. This convinced us to implement Xiaodong’s suggestion of using a color code system as a way to visually make it easier for the user to understand the sorting; Leading to our second prototype/learn screen:



Bubble Sort

Description:

1. Compare the first two elements and swap them if they are in the wrong order.

2. Compare the second and third elements and swap them if they are in the wrong order.

3. Repeat for the rest of the elements in the list.

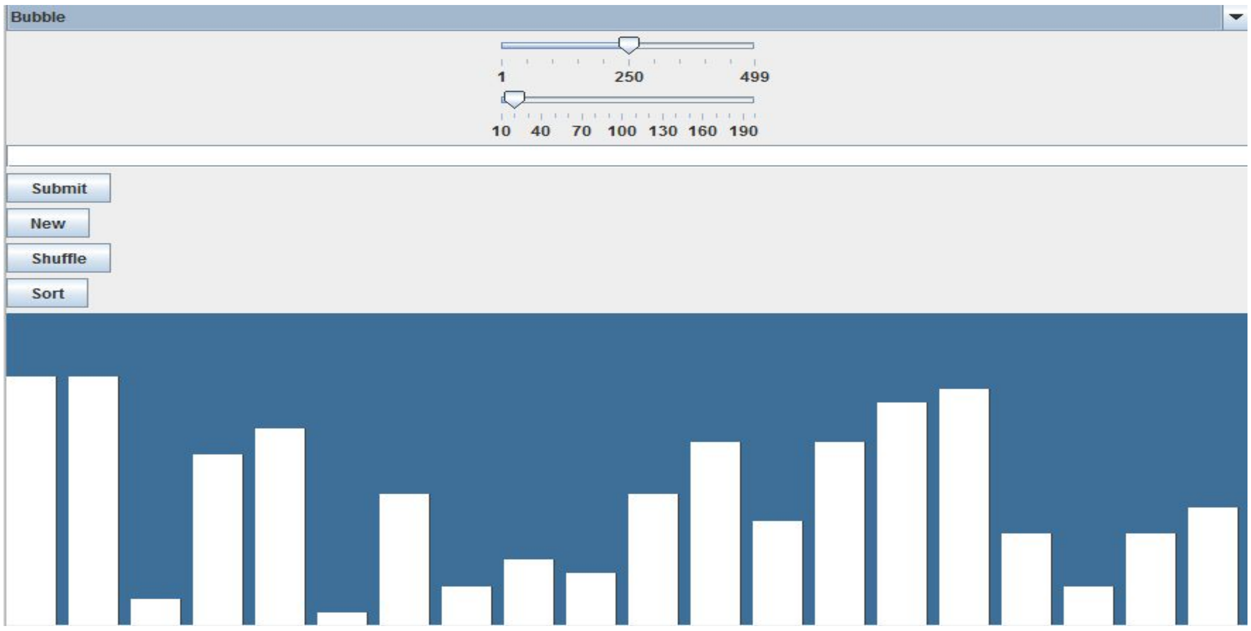
4. Repeat these steps until no more swapping is required.

Type: Exchange sort

⬜ Items in array
▨ Items being checked
🟥 Items in wrong order
🟩 Items in right order
⬛ Items sorted

Final design and implementation of VisuSort

As well as the inclusion of these new colors the visualisation by default was set to have a smaller array and a larger delay between steps. The visualisation itself without any layout or buttons for the users had a very simple implementation quite early on in the project:



Initial working sort prototype

The above version of VisuSort was extremely buggy and in no way user friend but was used as a start to what would become our finished product. The main change between the final/current VisuSort other than bug fixes would be the stages slider. This slider allowed the user to go back through the stages of the sort and proved to be quite complex to implement.

4. Software Engineering and Risk Management

4.1 - Software Engineering

We chose to use the agile software development methodology Scrumban (a fusion of the Scrum and Kanban methods) for our project.

Scrumban is a hybrid of two Agile development methodologies (Scrum and Kanban) which enable software to come to fruition via iterative feedback and improvement over the course of a project.

We found that Scrumban naturally suited our way of working and played into the various deliverables/hand-in dates we were expected to meet throughout the project. We were able to divide our workload into sprints corresponding to the dates of these deliverables: Specification, Prototype, Soft Launch (as presented to other groups in the cross-feedback sessions) and Hard Launch (the final presentation).

Crucially, we would not progress to the next task until all previous features had been implemented or, otherwise, removed. This brought in elements of Kanban methodology as it put a limit on the number of tasks/amount of work we were tackling at any given time, helping to maintain a consistent workload throughout the project.

4.2 - Risk Management

To ensure success in our project it was essential for all potential risks to be considered and lay out a strategy in the case of each one occurring. The initial risk assessment can be seen in the Appendix of this document.

The first risk we considered was that of there being bugs in the code that would result in the project breaking. As it’s difficult to prevent bugs from occurring the best strategy we could see was having back ups of past code without bugs available, we have done this by using the SVN to regularly store working versions of code so if an issue occurred we could revert back to the latest working version. Fortunately we’ve had very few issues of bugs over the course of our project with all errors being fixable and in the rare cases where they weren’t we had the backups available on the SVN.

Another risk we anticipated was that of time constraints in the project resulting in us having to sacrifice quality of our product to enable deliverables were ready on time.

Converse to our initial plan time constraints tended to be more of a help than a hindrance, with our best work being carried out in the lead up to deadlines such as the prototype presentation. We also managed to have basic software components completed early on as according to our strategy.

The risk of a team member being too ill to work on the project was seen as unlikely but a high threat, luckily no one did fall ill during the course of the project, at least not to the extent of being unable to perform their tasks. In the case of this happening we did have all code regularly uploaded and up to date on the SVN so another team member could take over if the need arose.

Given that the project was being created entirely over computers any computer malfunction would have been difficult to deal with and could have considerable slowed down the project. Fortunately no one experienced any computer malfunctions, but in the case that malfunction did occur we had the backups on SVN that any member could pull down to a different computer if necessary, the computers in the Computer Science labs were also available for use in such a situation.

Considerations also had to be made when concerning the SQL database. Initially when security measures hadn't been put into place, all data inserted into the database was considered test data so no 'real' data was entered (ie. fake names and blank passwords). The structure of the table used to store data was also documented so if any attacks were to take place either by us or by an external source which was to drop the table, we could get it back quickly. Once our security was implemented, the records were wiped and we began considering the database as live.

The final risk was that of an internet outage wherein we would lose internet access which would prevent our usage of the SVN, this wasn't too much of a risk though as editing of code and documentation could be carried out without any internet access. Internet outage was also not likely to occur and apart from a few days where the Wi-fi was down on campus and a powercut this wasn't an issue at all. Expanding upon this topic, our servers were also seen as a sign of risk as much we relied on having the server online consistently in order to test our product and have all members working on what they needed to be. This was mitigated though with a stable wired only connection at one of our team members houses and a permanently on machine to handle the traffic being sent even at unsociable hours.

5. Evaluation

Throughout the project we have been reflecting on our achievements and learning from any mistakes we might have made.

5.1 - Teamwork

Throughout the project our team met three times a week to discuss the project, assess our progress and determine new tasks to be completed before the next meeting.While gaining the experience of working in a team environment, we were able to use our knowledge of designing, implementing testing and documenting a software in practical terms to produce a working product. By self analysing our problems via effective communication we were able to tackle professional issues similar to what we would experience in a real working environment. Overall, utilizing self directed learning as a team, we were able to explore software engineering techniques and come up with creative solutions to difficulties encountered in full cycle software project development.

5.2 - Learning Visualisation

Possibly the most successful, and with the amount of time and people we dedicated to it, rightfully so portion of the VisuSort product is the visualisation and learning page. This was the area where the bulk of the thought process went into including multiple weeks worth of getting the details of how it would work down before implementing this in code. However, this is possibly a major flaw in our development process. Too much time was spent on ideas that were overly complicated and involved too much work which hindered our progress early on. We do not feel like we were impacted by this though and in fact it worked out for the best to have this long thought process as it meant we could maintain a clear vision for the entire span of the project.

Our design is very much centric to the target audience of teenagers in college or high school. The colours are not overly bright but are subdued and have a clear and distinct theme. The design is also basic without being too basic as to insult the intelligence of our potential users. Furthermore, we decided it would be helpful for students to see a pseudocode example of the visualisation being demonstrated to them. Although we had higher plans for this, in it’s current iteration it introduces the idea of coding the sorting algorithms to users without giving them any understanding of what is going on. Time constraints here prevented us from implementing any HCI components that would have been useful in giving them a better understanding of the code and potentially improving their learning.

5.3 - Testing Suite

Testing a user to see what they had learnt through VisuSort remains a key principle of what the product is about. Especially if the product is going to be used as credit in students courses, we wanted to make it as diverse and editable as possible. Although the base functionality of it achieves many of our initial goals, we still fell short in some key areas which we hope would be revised in later product iterations.

Firstly, the static and unvaried nature of the test currently means that each time a user takes a test, they are presented with the same questions each time with the same answer in each case. This came down to a level of complexity having to be introduced which was not feasible within the time constraints that we had. What we would hope to achieve with the product once released are tests that even if the same question is asked, there would be different possibilities or variants on the question that make the test dynamic in nature. For instance, one of our questions

5.4 - Security

This report describes the security measures we took to protect the privacy of our users and the ways we have combatted people from exploiting the program with malicious intent. Although for this early prototype product, we wanted to enforce these key principles early and for a large part we believe we got this correct. Unless resorting to brute force methods, our encryption and hashing on our traffic means that VisuSort is secure to the most common of attacks.

However, with the time constraints we had and whilst still not being too paranoid regarding the security of our system, there are design changes we could have chosen to better secure our system and help to ensure users data is safe. Primarily, this comes down to the aforementioned brute force attacks the data transmitted is liable to. Our AES keys once discovered make secure transmissions useless and although replacing keys can alleviate this problem temporarily, the key can repeatedly be compromised. Furthermore, our hashing algorithm for passwords, as mentioned in section 2.3, is based on SHA-256. This is not secure to brute force methods. A different hashing method or one that involves a random salt and then hash with a number of bits to the encryption would allow for much longer bruteforce times. With more time and perhaps more important data, this would be a valid concern. In the time we had though, compromises had to be made and a rudimental level of security in this regard had to be introduced.

5.5 - Not implemented

5.51 - Constraints

With the submission date approaching we believe we have created a complete product that fulfills all the aims we set out with from the start of the project. However, some aspects of the project that were either functional requirements from the start or just ideas we would have wanted to add were not implemented in the final version of VisuSort; the primary reason for this being time constraints. The first of these features was the inclusion of bucket sort, this was initially one of the four sorting algorithms we intended to implement (later replaced by selection sort) early on in the visualisation implementation it was obvious bucket sort would be much more complex to implement than the other sorting algorithms. The complexity/difference of bucket sort in comparison to the other three sorting algorithms led to the idea of its inclusion being scrapped.

The second instance of something like this happening was the inclusion of tutor accounts being available to users. Tutor account would have been able to create groups and view the scores of the students in their groups. A version of the score viewing was implemented but only made available to admins; if we made later updates tutor accounts could be implemented fairly simply. The learn screen was affected once again by the decision to not include a forward and backward button to allow the user to go back and forth through steps of the sorting. This was an idea from quite early on which never got implemented and once the idea arose again the time constraints meant it did not feature in the final product; these time constraints were mainly due to needing to get the test screen working. A final feature that didn't make it into the final product was the inclusion of colors in quick sorting, the main issue with implementing this was the complexity it also was overshadowed by implementation of the test screen(part of the project that took the entire team to complete) in the final weeks of the project.

5.52 - Scrapped

Some ideas that we came up with as a team were scrapped during development of VisuSort. The first and biggest of these was an idea we had in the beginning of implementing the learn screen, this was to include sorting shapes that the user would draw. This part of the project did have a prototype developed by Dogu which allowed users to draw shapes. The project however started to focus more on sorting as a learning tool for teenagers and we believed that the shapes were no longer required. There was a second idea that was scrapped was a similar idea to the previous shapes idea that we came up with. This was an idea to have a database of users who could be sorted based on attributes such as height or age; these ideas were thought of in a phase where we had not yet decided on sorting bars. After we decided to use bars to sort these ideas were never revisited and we believe without them the project still meets its goals.

6. Team Work

6.1 - Organisation

We quickly found that meeting regularly on Mondays, Wednesdays and Fridays suited us as a team. Notes were taken at each meeting and uploaded to SVN afterwards. This served the dual purpose of recording our own progress and making sure that we did, in fact, progress from one week to the next.
We used WhatsApp to stay in contact outside of team project hours and all made significant efforts to stay connected and improve our communication as the weeks went on. Roles were established as soon as possible to focus members and organise the modules of the software around them. This allowed each of us to have a degree of expertise in each area and worked well when it came to interfacing the different modules of VisuSort in an efficient manner.

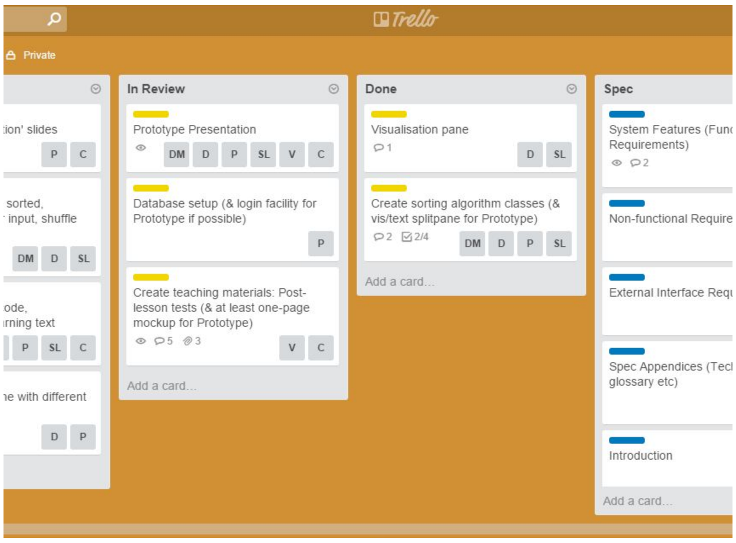
6.2 - Challenges

Challenges initially presented themselves as communication deficits in the early weeks. We would often find ourselves working on the same tasks in parallel without knowing what other team members were doing. This would prove to be a valuable lesson going forward and we decided to regularly verbalise what we were working on at each meeting. This would be the equivalent of our Scrum 'stand-ups' where members summarise their current work situation for others to know about. If a member was not present for these meetings, progress would be recorded through our whatsapp discussion and their tasks could be located through the meeting notes thereby circumventing any problems arising from their absence.

6.3 - Trello and Agile methods

We used Trello to help enforce our chosen software development methodology; Scrumban[3].

Trello enabled us to colour-code the sprints (see screenshot, right) which made it easier to discern our



ion' slides
P C

sorted,
input, shuffle
DM D SL

ode,
rning text
P SL C

ne with different
D P

In Review

Prototype Presentation
DM D P SL V C

Database setup (& login facility for Prototype if possible)
P

Create teaching materials: Post-lesson tests (& at least one-page mockup for Prototype)
5 3
V C

Add a card...

Done

Visualisation pane
1
D SL

Create sorting algorithm classes (& vis/text splitpane for Prototype)
2 2/4
DM D P SL

Add a card...

Spec

System Features (Func
Requirements)
2

Non-functional Require

External Interface Requ

Spec Appendices (Tecl
glossary etc)

Introduction

Add a card...

[3] http://kanbantool.com/kanban-library/scrumban/what-is-scrumban

progress, at a glance. We referred to Trello as much as possible and it was invaluable in helping us to get our product finished in a timely manner.

6.4 - Gantt chart

Our final Gantt chart is included in the **Appendix** of this report.

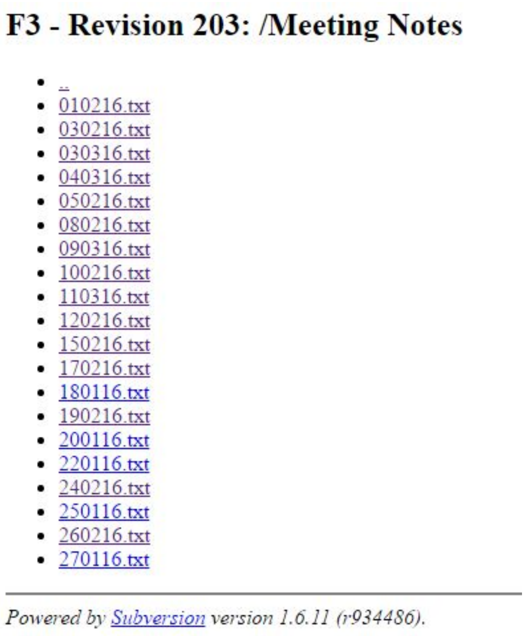
6.5 - SVN

Our team repository is located at: https://codex.cs.bham.ac.uk/svn/team-project/F3/

We found SVN to be useful for hosting meeting notes in particular, although it would have been beneficial if they had automatically been stored in reverse-chronological order.

Once a package structure had been established for the project we began using SVN more efficiently and found it easier to stay up-to-date with each other's versions.

We found merging our work to be troublesome as SVN is not as well-equipped to do this as Git. However, we did find SVN invaluable for quickly and easily sharing our work. Especially since we chose an Agile development methodology, the speed and ease of the sharing progress and



F3 - Revision 203: /Meeting Notes

- ..
- 010216.txt
- 030216.txt
- 030316.txt
- 040316.txt
- 050216.txt
- 080216.txt
- 090316.txt
- 100216.txt
- 110316.txt
- 120216.txt
- 150216.txt
- 170216.txt
- 180116.txt
- 190216.txt
- 200116.txt
- 220116.txt
- 240216.txt
- 250116.txt
- 260216.txt
- 270116.txt

Powered by Subversion version 1.6.11 (r934486).

A screenshot showing meeting notes in SVN

7. Summary

Our team project sought to visualise a fundamental principle of computer science and make it into an accessible learning product. In particular, we looked to be accessible to those of a younger age looking to develop their knowledge and those who may suffer from dyslexia or disabilities such as dyscalculia.

Through our time spent developing our product we identified the need not only to provide the learning mechanism for students but also a means for them to test themselves and have a learning tutor be able to review their progress. This has become a major component of the final VisuSort product. As we approached the final submission date, we started to allow users to use the system and see if we accomplished what we set out to. This came in the form of a testing environment set up for them with a post use questionnaire. Based on this feedback we believe we have achieved our goal of providing a visual learning tool.

The learning mechanism however was our primary concern as it directly tackled the task set out, visualise an existing algorithm. Sorting algorithms are a very apt algorithm to be visualised and we decided that the way we would differentiate ourselves was to use colours in a way that would alleviate the need for numbers and lots of code for the user to understand what is going on. With human computer interaction being integrated into this with customisable sliders and the user being able to change practically every parameter regarding the visualisation, we think that the product fully encompasses what a learning tool for sorting should do and has done this with a simple yet powerful visualisation.

8. Individual Summaries and Reflection

8.1 - Dogukan Baykan

In the first few weeks of the project, like the other team members, I contributed to the specification by writing the parts assigned to me, using trello for managing tasks.Afterwards we decided on the parts that needed to be implemented for a working prototype .Me and other members began working on the visualization however not much progress was being made in this area because we lacked a clear idea of how the visualization would look like.I decided to take the initiative and wrote a a simple bar sorting visualization and two of the algorithms, taking modularity into consideration to make it possible to re-use the classes for other types of animation.I then finished implementing rest of the algorithms with help from Seb and worked on creating the interactive control elements of the learning section until the prototype presentation.

After the presentation I listened to the feedback I received from other team members as well as our tutor and went onto improve the visualization and increase the robustness of the methods used by the control elements.While I was working on the visualization and the model used by the GUI, other members have improved the design of this section and at the end we have largely met our goals for this part of the project so I started documenting and commenting my work .Few weeks before the final presentation I started working on the report and testing section.I did two of the questions and went on to help other team members in fixing the problems encountered in piecing the questions together and getting the scoring system to work.

8.2 - Hannah Evans

In the initial stages of the project I focused on coordinating the organisational and teamwork aspects of team F3. We had regular meetings each Monday, Wednesday and Friday, and also found it convenient to stay in contact using WhatsApp.

As we began working on our first deliverable (an initial Specification) I introduced the group to Trello and the Agile software development methodology Scrumban. I suspected that this would be of help throughout the project, helping to keep work divided into manageable ‘sprints’ with flexible end-dates and also helping to spread workload throughout the project so as to avoid any unnecessary spikes. Our three weekly meetings were our ‘stand-ups’ and I was also responsible for assigning and monitoring Trello tasks and sprints. I am proud that I helped to establish a solid foundation for the group at this early stage.

After our teamwork and communication had reached homeostasis I began work on the teaching aspects of the project. I suggested that we take inspiration from Codecademy.com, an online coding course, whose various lessons have minimal amounts of on-screen text with the focus being on interactive learning in a central pane. My suggestion resulted in a lo-fi prototype of the main GUI being drawn up in one of our meetings and this served as a template for the rest of the software.

Once the team had decided on which (sorting) algorithms were to be implemented, Vicci and I researched and developed teaching materials for these. For each, I stored the distinct ‘steps’ (of the algorithm’s action), along with a short but informative description of its uses, in HTML format. This made for easy reading into the Java classes and also helped to satisfy our design aim of reducing the amount of text onscreen, generally.

As the ‘Learn’ section of the software began to be implemented by other members of the team I began work on implementing the ‘Test’ (i.e. quiz) section. This involved setting up a GUI ‘Test Menu’ with navigable question buttons, a score calculation/test submission button and questions being loaded in from an external HTML. As other members of the team began implementing individual test questions I began integrating them into the Test Menu.

After successful demonstration of our prototype I designed a feedback form for VisuSort which asked for users’ opinions on various goals we had for the software, including; educational objectives, HCI objectives and ease of use.

I enjoyed the module and learnt a lot about group work and my own working style.

8.3 - Vicci Garner

In the first few weeks of the project my role, similarly to the rest of the team members,was working on completing the initial project specification. At this stage I was responsible for laying out the time scale of our project in the form of a Gantt chart- this involved deciding specific tasks that needed to be completed in our project (such as creating a database, visualising, and testing at various stages) and how long each task should take to ensure that each required deliverable was ready at the correct time. This chart has been useful for keeping us on track over the course of the project.

As we moved on to the implementation of our product, I along with Hannah started work on the written teaching materials required for the different sections of the system; this entailed writing how each sorting algorithm worked along with each one's advantages and disadvantages. I also created and implemented the initial multiple choice quizzes for the test section using basic text and radio buttons; these tests were ultimately not used as we decided to go in the direction of more visual tests, however they were used for the prototype presentation to give an idea of what to expect from the end product.

After the prototype presentation the focus moved to visualisation- at this point I started work on improving the test section of our system, making the questions more visual in nature so that the user could test their knowledge in a more interactive way. I helped construct the questions by providing ideas in our group meetings for five different questions to visually test the user; my role from here was to create the last two questions. Question 4 proved fairly simple as it was asking the user to identify the sorting algorithm from three chronological pictures of an array being sorted and was just a matter of importing images and assigning radio buttons for each correct answer. Question 5 proved more difficult as our initial idea was to ask which algorithm would be fastest at sorting a particular array and then show GIFs of each algorithm sorting that array once the user had answered to compare- however I had a lot of trouble getting GIFs to work in java and even then getting each algorithm to sort at the same speed would be difficult. Also, due to the nature of our test section only allowing submission of questions after each one had been filled in, it was ambiguous as to when the comparison GIFs would be displayed. As it got to the end and we were approaching the deadline I made the decision- with the agreement of the rest of the team - to swap the question to ask the user to number the stages of a particular algorithm. This proved much simpler and allowed the product to be finished in advance of the deadline.

8.4 - Seb Lewis

At the beginning of the project I helped focus on ideas on what we could do as a group, after we decided on sorting algorithms I tried to help come up with ideas for how to visualise the algorithms. We then all worked on completing the first requirement specification. My main focus in the requirement specification was to complete the visualisation GUI designs and the flow of all the screens. Once we had decided on the algorithms we would cover I implemented the first implementation of the algorithms. Dogukan worked on the visualisation and got bubble, insertion and quicksort working, I then helped to implement selection sort after we scrapped the idea of showing bucket sort.

After we had a working prototype of the visualisation my main focus for a large section of the project was the learn screen. The first stages of making the learn screen were to implement some buttons for array manipulation allowing for the user to interact with the visualisation. We made an initial/prototype screen with features such as randomising an array, shuffling an array, adjusting size of the array, adjusting speed of the sort, and selecting which sort would run. This prototype was shown to Xiaodong and demonstrated to Ela during our first prototype meeting.

The next part of the project which was implemented was to move everything from the prototype and put it into a more professional looking screen along with making any appropriate changes. After this I continued to work mainly on the learn screen which included all the features from the prototype along with adding text descriptions. After this was implemented I tried to help Dogukan implement colors (a suggestion from Xiaodong) after the colors were implemented me and Dogukan spent roughly a week fixing bugs with the interaction between the visualisation and the buttons on the learn screen.

Once all the team members were happy with the learn screen everyone began work on the testing screen. I implemented Question 2 and helped Hannah move it into the Test screen. The final part of the project was me and Dogukan fixing the getScore() methods for all the questions once we fixed this it worked with the score submission(to the server) which Phil implemented. Throughout the project as well as coding I have tried to provide input and my opinion/ideas during team meetings. I also have contributed to all documentation whether it be functional testing, user testing or this report; my main focus being visualisation GUI design. With the team project coming to a conclusion I have thoroughly enjoyed working with the team on the project and have learnt a lot.

8.5 - David Marlow

During the first two or three weeks of the project, I was working on putting together the System Requirement Specification document. This was where the whole team worked together to set out the requirements and specifics of the project, and planned how the project should run throughout the term. Along with my teammates, I involved myself in suggesting ideas for the product, and I was part of the team dialogue when we settled on a final concept.

The first role I took on in the team was to work on designing and implementing an interface for the sorting algorithms to be used in the VisuSort program. This was done in a somewhat inefficient manner; a number of members of the team created an idea of how this should work, and then one was kept and the other ideas were discarded. It is important to keep the visualisation separate from the algorithms themselves, so that if any changes are made to the visualisation classes, algorithms can remain unchanged. This means that unit tests which have already been passed remain valid when visualisation changes.

My main contribution to the team is through the testing and quality control of the project. I created the test plan early in the project, in order to set out what the most vital tests to run would be. This was amended throughout the project as the requirements of the project evolved. When we decided on what the tests in the quiz section should be, I had to consider what the best test cases for these tests would be. The final version of this test plan is part of the test report, which includes planned test items for all different questions in the “Test” section of the program.

Another contribution I made to the testing of the VisuSort program was that I was the main driving force to ensure that functional testing for the program was carried out, both through completing large parts of it myself, as well as by encouraging my teammates who have better knowledge of specific sections of the program to carry out functional tests and record them in the Functional Testing Log, which is supplied as an appendix to the test report.

For the project, I developed the unit tests which were designed to ensure correctness of the sorting algorithms. This took a few weeks of work, due to changing implementations of sorting algorithms, but in the end the unit tests show that the sorting algorithms are correct. The unit tests are documented in the test report.

8.6 - Phill Oliver

During the first few weeks of the project, the group were all working on developing ideas for what we wanted to have the software do and draw up a specification for what it was to include and do. My role within this like everyone else’s was to partake in the groups discussion and critique each other's ideas in order to get the best possible product. I from this point started the writing up of the specification.

The bulk of my work was integrating our software into an internet connected resource handling all connections to external resources. This included setting up and hosting the java servers and also an SQL database. More on this can be found in section 2.3. As part of this, I solely handled all security concerns when it came to the software. Although we didn’t have to, we decided to take the security of our users very seriously when they are using VisuSort. As such, everything from SQL injection attacks and hashing of passwords had to be considered in order to ensure there were no vulnerabilities. I also ran attacks upon our own system including penetration testing and also monitoring the traffic to find any potential weaknesses (documented in test report).

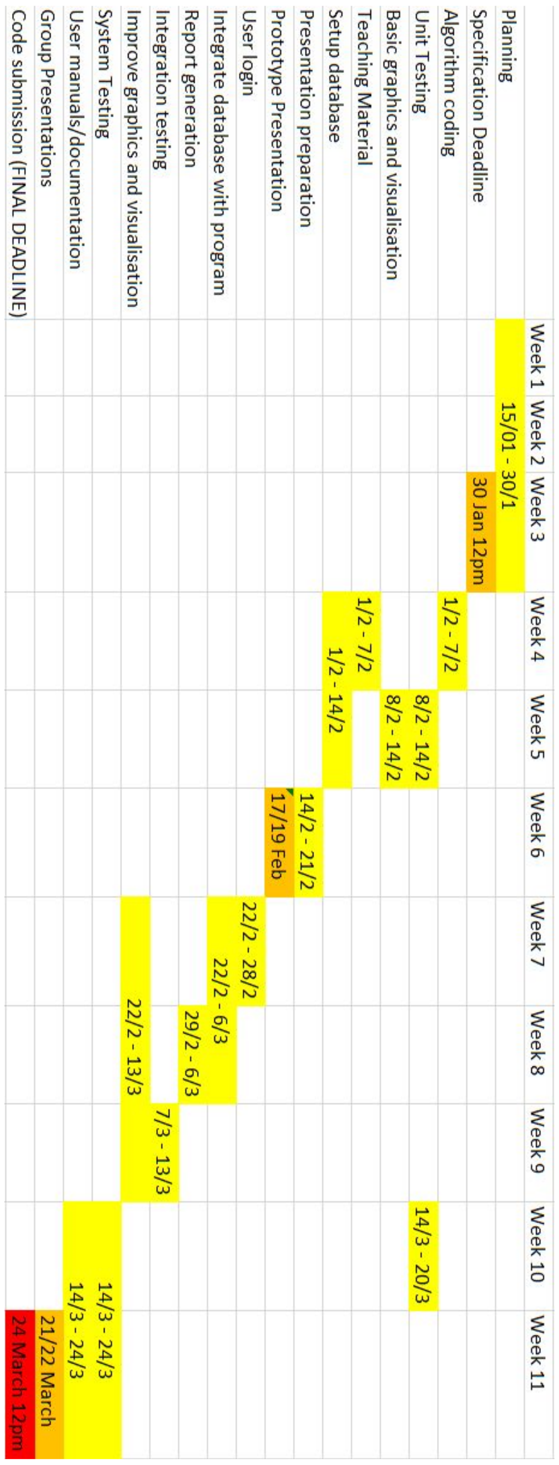
My contributions were also on the software itself however being the original designer for the opening screen. From here, elements like register window design and login had to be formatted as well as the underlying system to connect the different segments together. The importance of this screen meant I had to consider ease of use for users and thematically ensure our software is consistent. Leading on from this, I also designed the admin panel and mark viewing facility which is available to tutors and also the registration section. These panels again required thought into the best way to display them to users and I believe they are designed in a very clear and natural way. Leading on from this, within the testing section of our software, the ability to view previous scores and also submit your score once you complete a test was part of my design.

The whole design philosophy behind the user accounts section is that there is massive room for expandability so everything I have done can be expanded and has been even over the course of the software’s development cycle. The plan for VisuSort is that it will be a suite of programs for learning so this gives us lots of room to achieve this.

Final pieces of work I completed for the project were detailed Java documentation on classes as well as pushing myself and other members coming up to the final deadline to get work done.

9. Appendices

9.1 - Gantt chart, v2



	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11
Planning	🟨	15/01 - 30/1	🟨								
Specification Deadline			30 Jan 12pm								
Algorithm coding				1/2 - 7/2							
Unit Testing					8/2 - 14/2					14/3 - 20/3	
Basic graphics and visualisation					8/2 - 14/2						
Teaching Material				1/2 - 7/2							
Setup database				1/2 - 14/2	🟨						
Presentation preparation						14/2 - 21/2					
Prototype Presentation						17/19 Feb					
User login							22/2 - 28/2				
Integrate database with program							22/2 - 6/3	🟨			
Report generation								29/2 - 6/3			
Integration testing									7/3 - 13/3		
Improve graphics and visualisation							🟨	22/2 - 13/3	🟨		
System Testing										14/3 - 24/3	🟨
User manuals/documentation										14/3 - 24/3	🟨
Group Presentations											21/22 March
Code submission (FINAL DEADLINE)											24 March 12pm

9.2 - Risk Assessment

Risk	**Threat Level**	**Likelihood**	**Strategy**
Bugs in code	High	Medium	● SVN must be used to regularly commit (i.e. backup) code to the remote repository, so that a rollback can easily be implemented if necessary
Time constraints ***(i.e. sophistication or quantity of features is compromised)***	High	Medium	● We must focus on getting basic software components done first ● The software can then be fleshed out to include smaller details/features ● Quality over quantity
Illness	High	Low	● All code must be heavily documented and shared via SVN. ● Any jobs which have been assigned to the person(s) affected will be reassigned immediately
Computer malfunctions	High	Low	● In case of personal computer problems, the school's lab machines should be used as a first reserve ● Work should regularly be committed via SVN so that more than one copy of each file/resource exists at all times

Compromised SQL Database (Security)	Medium	Low	● All sensitive data must be encrypted ● No actual (i.e. plain text) passwords should be stored anywhere ● A strong password should be used for database access
Internet outage	Low	Low	● We do not require internet access to implement features of the software, but access to our notes/resources/repository will be affected ● We should focus on other tasks such as mock-ups, presentation/demonstration preparation