Antoine Gautreau,
Thomas Gautreau,
Alex MacDonald,
Carter Nesbitt.

Online Subscription for Java Games

December 14, 2020

Overview

A website is to be created. The administrators will be able to create accounts for users to login to the website with. Once logged in, the user can download the JAR file for the game. The user can then register the game with their created username and password. After playing, the player's score will be updated and recorded on the scoreboard on the website.

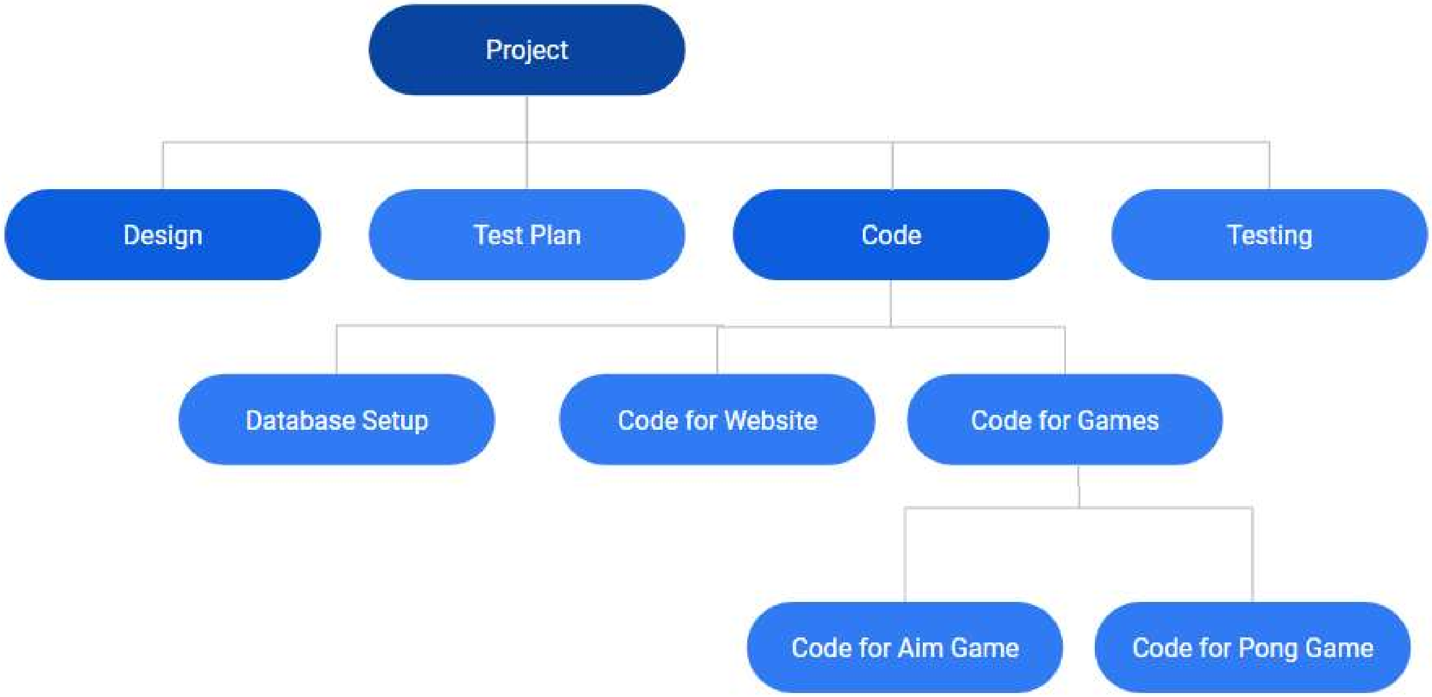
Goals

1. The admin can create user accounts.
2. A site is made where a user's login name and password can be used.
3. The user can download the game from the website after logging on.
4. The user can register the game with their username and password.
5. User's game score is saved and updated.

Planning

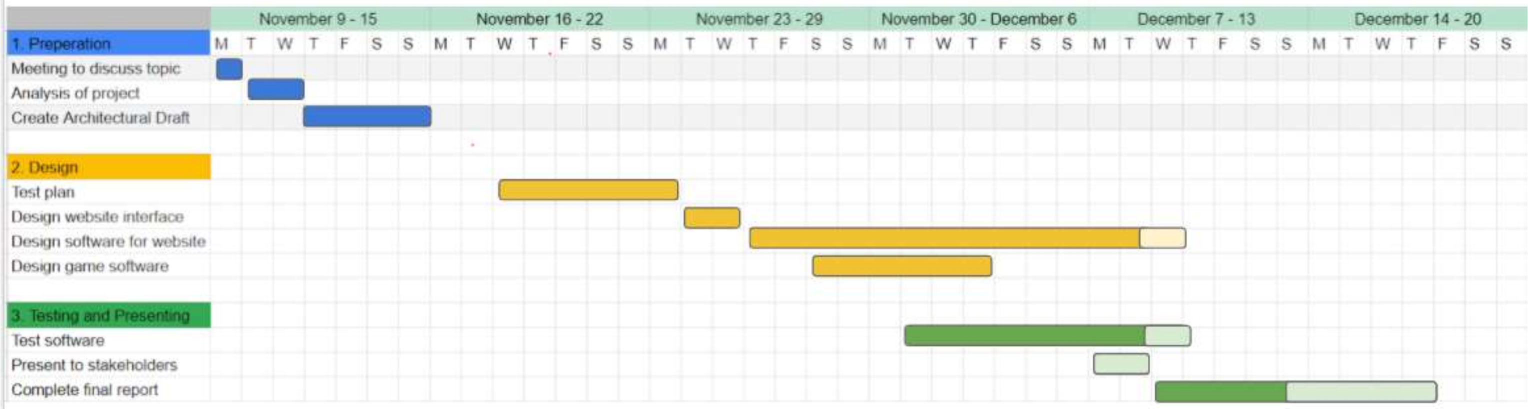
Work Breakdown Structure

A work breakdown structure was created to help visualize the tasks at hand. It helped our group to know all the different aspects required for the complete project, as well as helped show which phases of development stemmed from others.



2.Gantt Chart

A Gantt chart was used to plan out the different phases of development for the program. It was updated when different aspects of the program were completed, and overall used to keep the group in track and on schedule.

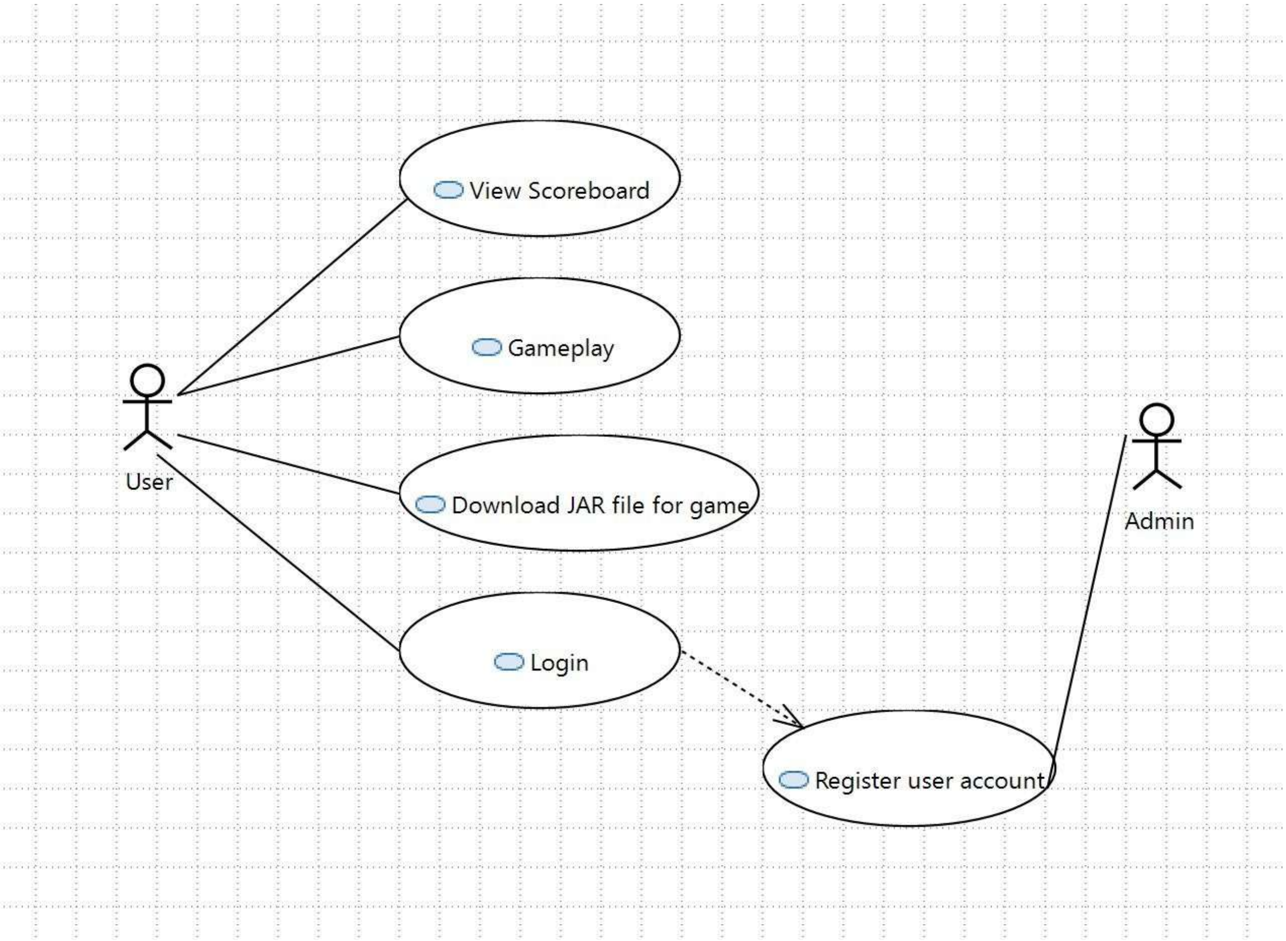


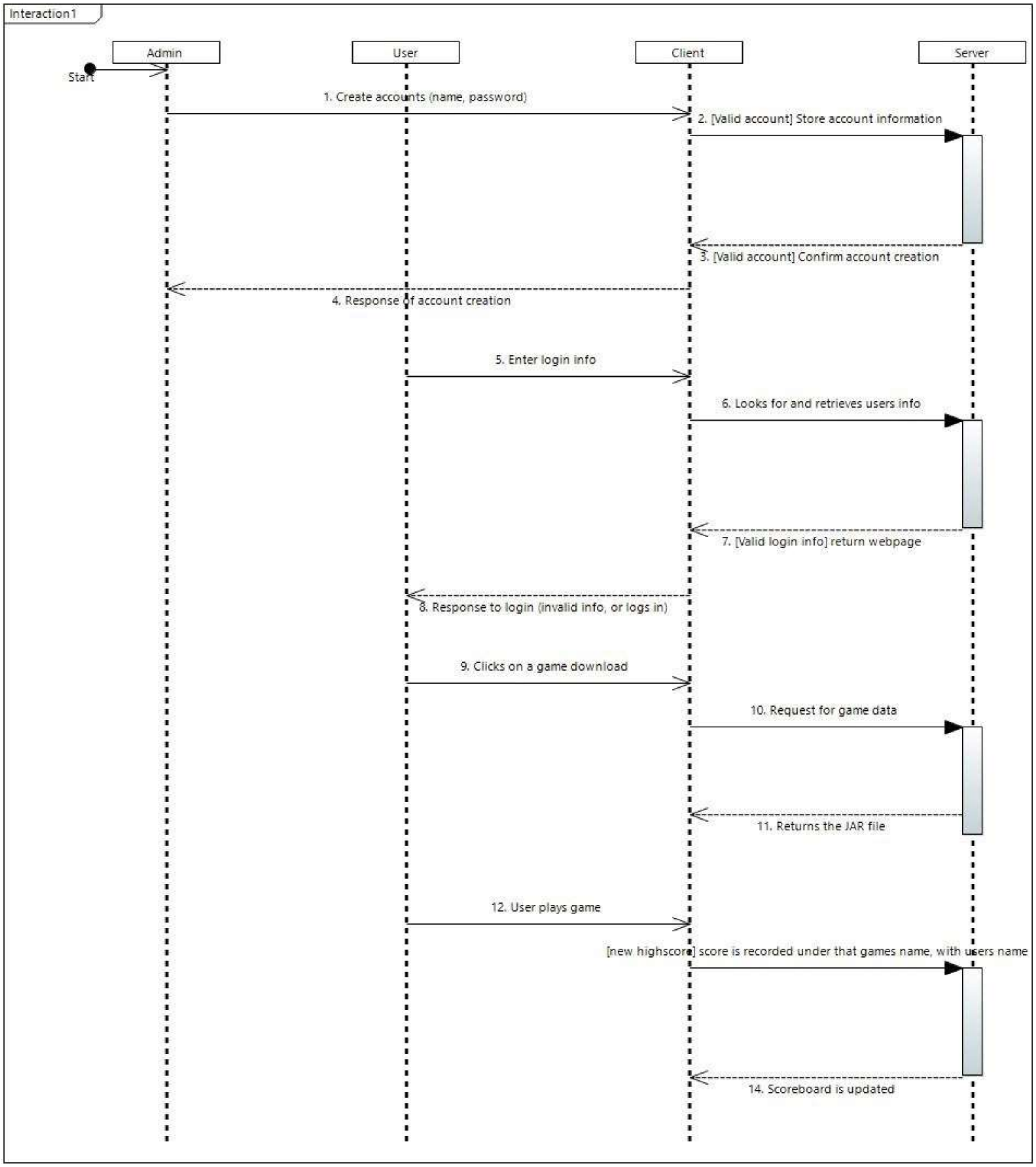
3. Identified Relationships

To help with the creation of the use case and sequence diagrams, the programs relationships were identified and recorded.

- Administrator creates users username and password.
- User logs in to the website.
- User downloads the game’s JAR file.
- User runs the game.
- The game updates the scoreboard.

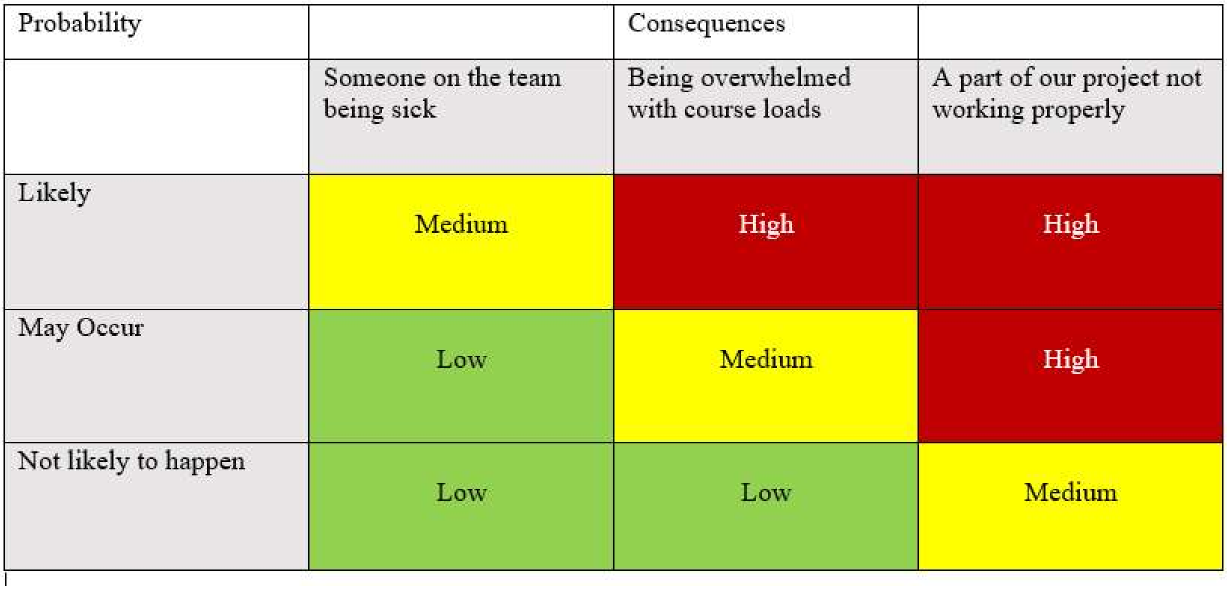
Diagrams


View Scoreboard
Gameplay
Download JAR file for game
Login
Register user account
User
Admin


Interaction1
Admin
User
Client
Server
Start
1. Create accounts (name, password)
2. [Valid account] Store account information
3. [Valid account] Confirm account creation
4. Response of account creation
5. Enter login info
6. Looks for and retrieves users info
7. [Valid login info] return webpage
8. Response to login (invalid info, or logs in)
9. Clicks on a game download
10. Request for game data
11. Returns the JAR file
12. User plays game
[new highscore] score is recorded under that games name, with users name
14. Scoreboard is updated

Risk Management

We decided the best way to represent our potential risks was with a risk management matrix. We first outlined the three potential risks and how likely they were to occur, and then determined how much a risk they were to our schedule.



Probability		Consequences	
	Someone on the team being sick	Being overwhelmed with course loads	A part of our project not working properly
Likely	Medium	High	High
May Occur	Low	Medium	High
Not likely to happen	Low	Low	Medium

Development Choices

- HTML, CSS and JavaScript were used to create the website's front-end interfaces.
    - We initially were going to use Java and SpringBoot to create the website and it's interface along with HTML, but eventually decided it would be easier to use Node.js over SpringBoot as some group members had past experience using JavaScript.
- Since the games needed to be made in Java, we used Javafx for the game design.
- Node.js was used to set up the server-side and the routing for the website.
    - We chose Node.js primarily for the reason that it provides a quick and easy way to create a simple web server that can be used to get our website displayed on our localhost in the browser.
- Node.js was used to connect the front-end (login inputs) of the website to the "users" table in our database and JDBC was used to connect the games to the "score" table.

- A localhost was used.
  - We used the localhost since it is a good way for testing web applications and since we are new to web development we decided to keep it as simple as possible.

- The “Aim Trainer” game was made in a way that the score was the time in which all thirty targets were hit. A reset button was implemented so that the user can play again after submitting their score, or try again if they did poorly or messed up during their run. The submit score button was used instead of directly opening the submit screen to give the user a choice to upload after seeing the score they obtained.

- The “Pong” game was made in a way where you keep a score until your opponent scores (in which the game ends) so that a more traditional scoring system could be utilized.
  - The opponent follows the Y direction of the ball at a different speed which changes throughout the game. This was chosen instead of a complicated A.I for this game as a more advanced A.I would be unnecessary.
  - The game increases the speed of the ball after it is hit up to a certain amount. The A.I increases aswell but their speed reaches its limit before the ball reaches the speed limit. This is so the A.I has a harder time hitting the ball which makes it fair for the player.

We decided to use an agile software development approach for this project, as we had a small team and a short period in which to plan. We made the choice to test our program as we created it. We made small prototypes for the website and when we were happy with the result we used that to build off of.

Task Management

- **Alex (Game Designer):** Working on “Pong” game code design and registration/validation of users when saving and uploading the user’s scores from the games to the database. Creation and organization of the final presentation.

- **Antoine (Website Front-end/Back-end):** Help on the setup of the front-end of the website (HTML and CSS of the webpages). Creating the routing for the websites pages. Creating the local database where the users and scores would be stored. Setting up the connection to the database and validation of user input to be inserted into the database.

- **Carter (Game Designer):** Working on “Aim Trainer” game code design and registration/validation when saving and uploading the user’s scores from the games to the database. Creation and organization of the final presentation.

- **Thomas (Website Front-end/Back-end):** Working on front-end design for the website (HTML, CSS and Bootstrap). Help on the database creation and accessing the database to update the websites scoreboards and user info.

Verification Checklist

The verification for this project was done by testing each of the key pieces extensively. This would be to protect our project from having an issue occur in multiple areas.

- Our Java games were tested for bugs and it was ensured that the scores were properly uploaded to the database if the score was considered a high score.

  Test Cases: The "Aim Practice" game was tested to ensure that the timer was correct and that when the thirty targets were hit the timer stopped and saved the current time as the score. It was tested to make sure the restart button worked at any time during the run. The mouse click event was tested through clicking different places on the screen, and ensuring that the score only went up when a target was clicked. It was also tested to make sure the submit screen with the test field for the username and password would appear when the submit score button was clicked. Lastly, the submit button was tested to ensure the score, username and password were saved to the database.

  The "Pong" game was tested to ensure the speed of the ball increased at a reasonable rate with each hit. After a lot of test trials, the speed of the ball would only increase up to a certain amount so that the ball didn't move through the players. The A.I was ensured to follow the ball so that it would hit it until the speed was increased, eventually making it more likely that the A.I would miss the ball. The score feature was tested to ensure the score only increased when the AI was scored on. It was tested to make sure the game ended and the submit screen with the username and password text fields was displayed when the AI scored on the player. Lastly, the submit button was tested to ensure the score, username and password were saved to the database.

  Both games have implemented a JDBC driver to connect to the database. The JDBC driver uses prepared statements to prevent SQL Injections.

- The website was verified by testing for bugs, user authentication, as well as during the user creation by admins. This was to make sure the website was receiving the data from the database properly.

  Test Cases: As a sort of user authentication, we programmed the website to redirect itself to the login page every time the login and password text fields had invalid login information. Also, when the admin creates users a message shows up saying whether the user creation was successful or unsuccessful depending on if the response given by the database after trying to insert the new user.

- The database was tested to ensure that the data was being saved and shared correctly by the games and website. We would also test to make sure the data was secure and that it was being sent to the website when the user wished to send their score.

  Test Cases: Confirmed that the entries from the website users, as well as the games scores were stored correctly in the database.

Validation Checklist

- We were able to make an admin page to be able to have an administrator create user accounts for them.

- We made a website able to take in a users login information, which allows them access to the rest of the website.

- We were unable to make a download system which allows the users to download a JAR file of the game. This was due to the fact that we used JavaFX to make our games. When we would produce the JAR file, it would not open since the file did not recognize the JavaFX library.

- We had a working copy of the games, which let you submit your scores to a database and only kept track of an accounts best scores for the games where they submitted a score.

- We were unable to produce a dynamic scoreboard which displays all the highest scores and with an associated account username. Part of this problem was our choice of using JavaScript/node.js to develop our website. Since JavaScript can sometimes run the program asynchronously, it can start to process certain sections of the code and continue along the program while other parts are being processed. This is problematic since we needed to use the information that was being processed, so that we could display the scores to the website. We found that the solution to our problem is to use callback functions.

Lessons Learned

- Better understanding of project management techniques.
  - The project management techniques examined in class were further explored and a better understanding was gained through implementation of these techniques, such as the use of a gantt chart for planning or a risk matrix for risk management.

- Better understanding of web design and uses of databases in working programs.
  - Our group for the most part were not very familiar with web design or the use of databases in java programs. Through working through this program we obtained a better comprehension of web design and how to implement a database as we developed our own website and implemented these different concepts into our own program.

- Identifying and finding solutions to problems when developing programs.
  - Many problems were faced when designing this program, such as a lack of knowledge on website creation. However, we were able to overcome most issues that we identified, through working as a team and using the resources available to us.

- Communication and effectively working together in a team.
  - Throughout working on the program, better communication skills were acquired. Our group members all had busy schedules and because of this, mutual time periods to work on the program were difficult to find. Thus we became more efficient in communicating effectively in the periods of time that we did have to ensure that as much work could be done as possible.