

PROJECT MANAGEMENT AND DOCUMENTATION

By Oscar Liang

Contents of Document

**Contents of Document2**

**1. Defining and Understanding the Problem3**

**2. Identifying a Suitable Development Approach 5**

**3. Identification of Input processing and output6**

**6. Design and Function8**

**7. Evaluation10**

4. Gantt Chart and 5. User Manual see separate documents

1. **Defining and Understanding the Problem**

**Purpose of Project**

Life Simulator was created for the Year 12 Major Project Assessment Task for Software Design and Development (SDD); however, it will be released to the public as soon as version 1.0.0 has been completed.

**Defining the Problem**

As specified by the assessment notification, this project is a free choice project that is to be of an educational or commercial nature. Therefore, I have made it so that it would be a project of my interest.

**Inspiration**

This project was inspired entirely by BitLife. After playing the game for myself, I have discovered a few flaws in the game that I don’t like, such as the one-year iteration system. Therefore, I have taken this as a challenge for myself and decided to recreate the game with these flaws in mind, and create Life Simulator. I chose to recreate BitLife as opposed to other games that also have flaws as it would match up with the requirements and time limitations of the assessment task.

**Differences from BitLife**

As stated in the introduction, this project is entirely inspired by BitLife, however I have made it have significant differences in order for it to stand out from BitLife.

**The One-Day Iteration System**

The BitLife has a One-Year iteration system, which means that you can do actions at various times every year. This is not a fair system, as a lot of things can happen in your life in one year, and there are certain things you can or cannot do within a year. Therefore, I remade this feature into the one-day iteration system, however even this has flaws, as there are certain things you can only do during the daytime/night-time, and you cannot do infinitely many things during the day. If this project becomes successful enough, I may do a one-minute iteration system, with time passing as you do each action. This would be extremely hard, as there would be over 40 million minutes in an average lifetime, as opposed to over 30000 days or over 80 years.

**Lack of a Date System**

In BitLife, there is no system specifying the date or any time period you are currently in. Therefore, I have added this system in my Life Simulator so that the user can know what time period the player is currently in.

**2. Identifying a Suitable Development Approach**

The software approach I used was originally the prototype approach, as the amount of time I had to complete this project of 7 months meant enough time for me to gather users, have them try out my game and report whether there has been any issues with the code, however, due to the fact that I left this project to the last minute, as well as the effects of the COVID-19 pandemic, the amount of time got shorter, and hence I only had 2 months to complete it. Hence the software approach I had to use had to be switched to an approach most similar to the RAD (Rapid Application Development) approach.

Other approaches such as the structured approach and agile approach were not considered, as those approaches meant that I had to have a skilled team of programmers and a high budget in order to complete. I had neither of those. The end-user approach was also not considered, as that would mean that the project would have minimum functionality, not ideal for a major development project.

In my project, planning of any sort, including the creation of flowcharts, sketches and EBNF diagrams, was not used. All program ideas were either designed on the spot or thought of previously. Although RAD approaches usually do involve planning of some sort, I resorted to pure brain power, as this allowed me to save time and get more of this project done before the deadline. In the end it actually didn’t turn out so bad, as I was happy with the final design of the program.

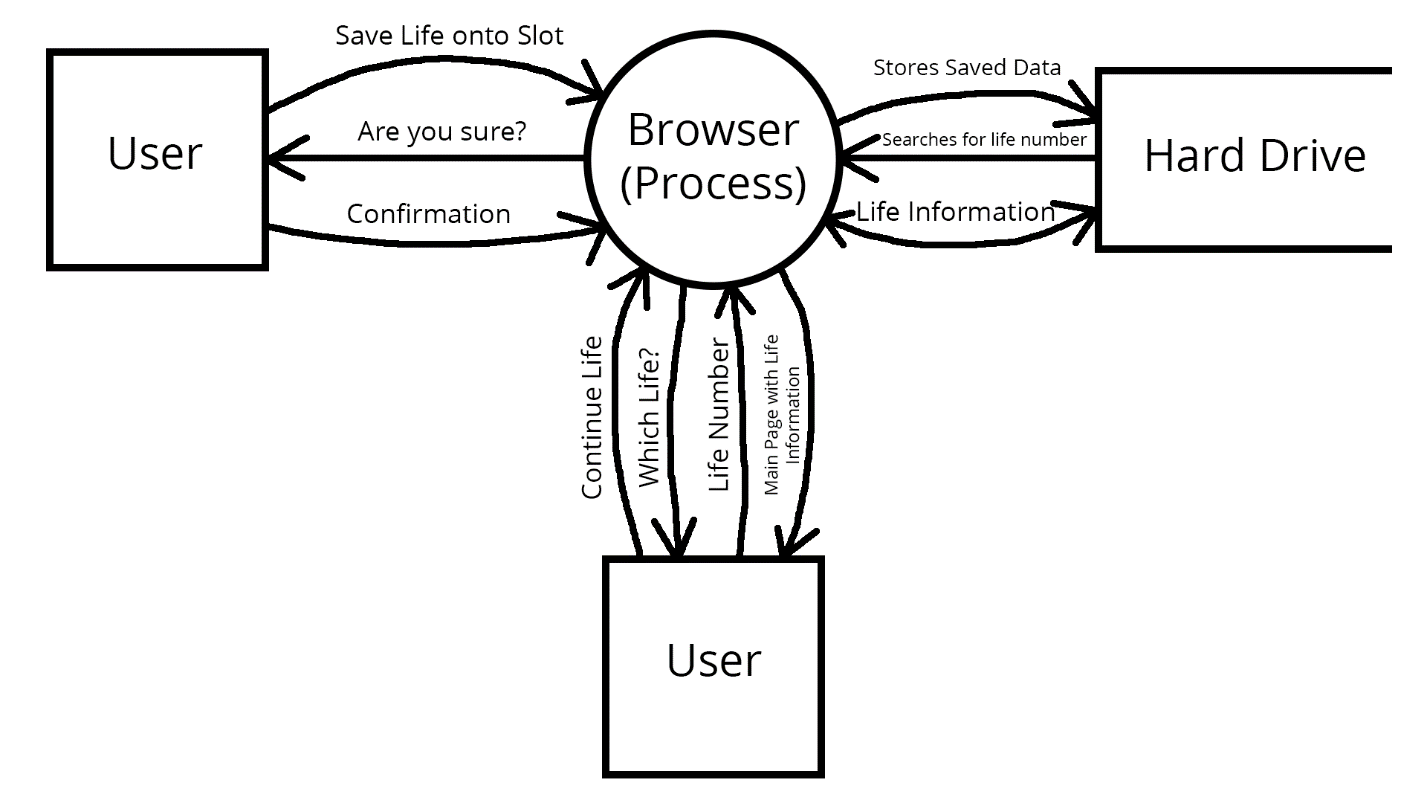
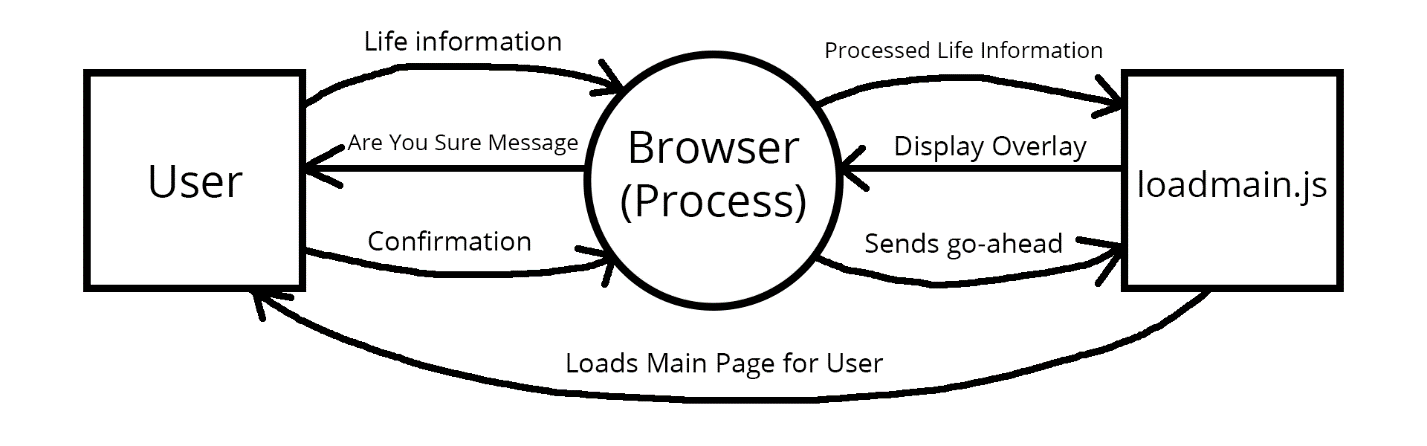
While implementing my program, I tested the program every time I add a new feature in, as I am the type of person that makes a lot of mistakes, no matter what. Therefore, testing at a frequent rate was vital, as that allows me to easily make out where I had made a mistake. This means that I had refreshed the page to load the new code on Firefox at least a thousand times while coding my program.

I also took advantage of the internet while implementing my program, as I had copy and pasted code from certain websites to help me further cut down the time it takes for me to complete the program. The jquery.fittext.js file (code that allows text to be the same size no matter how big the window was) and the js.cookie.min.js file (code that allows easy modification of cookies) was entirely copied off from the internet.

I resorted to using only intrinsic documentation, such as adding comments inside the code to identify what each block of code intends to do, as well as useful variable and control names to ease the comprehension of code throughout the program.

**3. Identification of Input Processing and Output**

Below are charts describing various processes in Life Simulator.

This Context Diagram describes the process in which the user starts their lives. This process happens during the beginning of the game, and starts the adventure of the player.

This Data Flow Diagram describes the process in which the user saves their lives. When this process happens, all of the information about the life of the user is copied onto the hard drive of the user, where they can access it the next time when they want to continue on with their life.

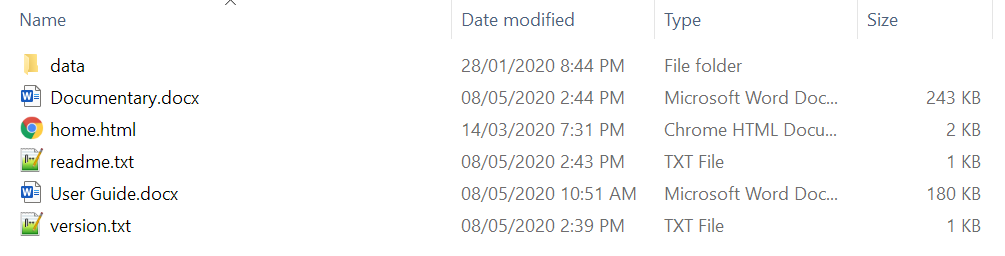
This IPO chart describes a process is known as time advancement. This process happens every time a day has been advanced. This is known as a function known as progress, and is identified in the code as progress(), which is lines 215-254 in loadmain.js. When the play button has been pressed, this function will repeat 50000 times, or advancing 50000 days (136.9 years), well beyond the average lifespan of a human. The function stops repeating whenever the pause button has been pressed or an event, such as death, happens in the player’s life. This is by far the most complex system in the entire game, yet it happens many times all throughout.

|  |  |  |
| --- | --- | --- |
| **Input** | **Process** | **Output** |
| * Break Variable * Days Since Birth * Date * Control Speed | * Determines whether process goes ahead based on break variable * Sets the date to be one day after * Sets the days since birth of the player to be one day older * Determines whether the player will die based on age and random number, if random number is greater than processed age, player will die, player will receive message and game will be ended * Waits a number of milliseconds based on the control speed before repeating process again | * Sets the calendar on bottom right to be one day after * Sets the players age on the info box to be one year older * Adds an event to the player’s diary, if an event occurred * Messages about major events |

**6. Design and Function**

The programming language used for Life Simulator is HTML, CSS and JavaScript. I have decided to choose this programming language as I have had plenty of experience with creating websites before, hence coding a game with HTML is relatively easy for me. This is also beneficiary because almost all devices could run HTML websites, including mobile devices such as a tablet or a phone.

Every single file that allows Life Simulator to run is located in one folder, which is the folder that this document is currently in, considering it wasn’t copied elsewhere. This folder may have varying names depending on the version of the simulator. By default, this folder contains 1 folder and 5 files. Below is a screenshot of the contents of that folder.



|  |  |
| --- | --- |
| data | All of Life Simulator’s files that allow it to run, including, HTMLs, CSSs, JavaScripts and images. Deleting this folder is not a good idea. |
| Documentary.docx | This word document you’re looking at right now. Contains the project management notes, documentation and explanation of code. |
| home.html | The main page of Life Simulator. Create a new life, continue your life, view your past lives, change the settings or view the credits here. |
| User Guide.docx | Word document that explains how to use the program, how to play and advanced tips and tricks of the game. |
| version.txt | Current version and date released of the game, as well as what’s new in this version. |
| readme.txt | Very basic information about Life Simulator. |

In order for Life Simulator to work, home.html and data must be in the same folder, otherwise the game will not work. It is also highly recommended that readme.txt to not be removed from the folder.

Every single html page is connected to three JavaScript files: jQuery, FitText, JSCookies and Unix to Date.

* jQuery is a feature-rich JavaScript library which adds certain features into HTML.
* FitText allows text to stay the same size regardless of the browser size. This is designed so that the user can manually change the GUI size. These JavaScript files are located in the data folder.
* JSCookies allows the programmer to easily change the value of cookies in the game.
* Unix To Date adds functions that are helpful to converting Unix timestamps into integer dates, or vice versa. This is extremely useful as the whole game is based around dates.
* Every html page is also connected to a CSS file named style.css in the data folder, which contains information on how to style each element of each html file.

Users are advised to open home.html upon downloading to begin the program. This will bring them to the home page, shown below, where they can create a new life to start their Life Simulator adventures.

**7. Evaluation**

Unfortunately, due to all the time I have wasted during the development of this project, I was only able to get the game up to version 0.2.4 during the deadline. This version of the game has very limited life functions, as the player could only be born, and could only die. However, the save and preserve life features were completed, which were the defining features of the game.

**Problems I faced**

Just like with all other development projects, there has been many problems that I faced during the development of this project.

One problem that I encountered is that of the storage of global variables in HTML. Global Variables were required in my program since there was two HTML files throughout the program, and I wanted both of them to share the same data. It was also required as I wanted users to be able to access their data if they quit out of their game.

The first thing I thought of using was writing all my save data on a separate JavaScript file, and have my HTML files read off that. However, I later realised that there was no way to do this, as HTML does not have a built-in file system to write and save files. Therefore, I resorted to using Cookies, which was a big blow for my project as it meant that Chrome can no longer be used to play the game.

Another problem that I encountered is that of the looping system in JavaScript. In JavaScript, the HTML only changes when the entire loop has finished. This means that when the play button is pressed in the main game, the screen wouldn’t update until after the loop has been broken. I didn’t want that, as I wanted the player to know their stats (such as current date and age) at all times.

**What I’ve learnt**

Although Life Simulator was never finished, whilst making this project I have learnt a lot of lessons, and has made my skills advance by a lot.

My understanding with the HTML, CSS and JavaScript languages have been advanced by a lot. While doing the project, I’ve ran into many problems that I didn’t know how to fix before, such as the aforementioned global variable problem as well as the JavaScript looping system. By doing research, I have now learnt how to solve these issues, and I believe the next time I create a HTML website it will be a lot easier, faster, as well as being better overall. This is extremely helpful as some jobs require their employees to have a fluent understanding on programming, which by doing this project I feel I have gotten closer to.

More importantly, I have also learnt to manage my time better, and that when assigned a major assignment, especially if my goal is to make it even better than what is expected, I need to put the time and effort into constantly doing the project and not to leave it to the last minute. I did not get to finish my game in time, and this has caused me to undoubtedly lose many marks on my report.

**How I could have Improved**

As previously mentioned, not leaving my assignment to the last minute would have made a huge difference in the outcome of my game, since I would have finished it. However, there are also many factors that could have also improved my design, such as simplifying my code, using shorter variable names as well as better organisation with all my files.

**Summary**

While this project was completely unfinished, and has completely no life features except being born and dying, I still considered this project to be a major success as it has taught me a lot about programming and time management.