

Combat-Cherrypy Server

COMPSYS 302 - Phase II



Group 1

SEung Bin (jack) choi - scho594

WAN CHUNG (BENJAMIN) TONG - WTON874

Contents

[Table of Figures 1](#_Toc453038086)

[Introduction 2](#_Toc453038087)

[Requirements 2](#_Toc453038088)

[Top-level view of System 3](#_Toc453038089)

[Screenshots 4](#_Toc453038090)

[Significant Issues and Solutions 5](#_Toc453038091)

[Features 6](#_Toc453038092)

[Network model Discussion 7](#_Toc453038093)

[Protocol Discussion 7](#_Toc453038094)

[Tools 8](#_Toc453038095)

[Future Improvements 8](#_Toc453038096)

# Table of Figures

[Figure 1: Top-Level Diagram of the Server 3](#_Toc453038097)

[Figure 2: New In-Game UI 4](file:///C:\Users\Choiboy\Desktop\COMPSYS_302_Group1_FinalReport.docx#_Toc453038098)

[Figure 3: Old In-Game UI 4](file:///C:\Users\Choiboy\Desktop\COMPSYS_302_Group1_FinalReport.docx#_Toc453038099)

[Figure 4: Front main page 4](file:///C:\Users\Choiboy\Desktop\COMPSYS_302_Group1_FinalReport.docx#_Toc453038100)

[Figure 5: Challenge/Online user list 4](file:///C:\Users\Choiboy\Desktop\COMPSYS_302_Group1_FinalReport.docx#_Toc453038101)

[Figure 6: About Developers page 4](file:///C:\Users\Choiboy\Desktop\COMPSYS_302_Group1_FinalReport.docx#_Toc453038102)

[Figure 7: Pop-up Notification 4](file:///C:\Users\Choiboy\Desktop\COMPSYS_302_Group1_FinalReport.docx#_Toc453038103)

[Figure 8: Challenge responding page 5](file:///C:\Users\Choiboy\Desktop\COMPSYS_302_Group1_FinalReport.docx#_Toc453038104)

[Figure 9: Network model (Hybrid) 7](#_Toc453038105)

# Introduction

The first phase of the project was to recreate the game, ‘Combat’ (by Atari) which would be suitable for 8-years-old children to play. The game was developed using Java and other software including GIMP to make the game more appealing for the younger players. In the second phase of the project, the task was to create a web server using Cherrypy for our combat game’s multiplayer feature. The aim of the second phase was to allow communication between the game (Java) and Cherrypy sever using the recommended tools that were recommended to the students. HTML, CSS and JavaScript were also used to get through the front-end development and using Cherrypy (Python) and SQL for the back-end development of the web server. Reasonable adjustments were made in the Java files to be able to connect the jar file to the Cherrypy server which will be described in more detailed later through the document.

# Requirements

Since the first phase, minor changes and addition to the existing code was made to improve the gameplay. Firstly, more requirements for phase 1 were met (which weren’t met by the end of first phase) this time by fixing up a lot of the collision detection functions. The only problem in the Java part of the project is that bullet-wall physics is not perfect and the bullets may bounce off the wall in abnormal ways half the time. Other than that, better user interface was implemented by adding text next to each tank to indicate which tank is which player. Secondly, a clearer score and power-up indication text on each side for the players was also implemented to help the player know their score and the power-up they currently have.

In the second phase, there are a number of requirements that are needed to be met to receive certain grades; at least 4 tasks of the specific grade’s task list are needed to obtain that grade. C grade tasks are mostly met where logging into the server and seeing who is current online was part of them. Sending challenges to other online users/responding to challenges and opening the Java application (Combat) through the Cherrypy server was also implemented successfully. JSON was used to send data through within the other web servers and also had encryption/hashing for the login information. The password is also masked so other people cannot see the password when it is entered into the server for security purposes. The HTML and CSS codes were separated from the Cherrypy code for simplicity and to show the modular and Pythonic structure, which also has short comments explaining the functions implemented into the code. A fairly simple but attractive design was implemented using HTML/CSS (front-end) to not overcomplicate the web server where simplicity was a key point when designing it. When the Java Application is closed (Combat game), it will automatically log the user out from the web server. Threading was used carefully to automatically report to the server every 30 seconds to keep the player logged on and the web server was tested on both web browsers that were brought up in class (Google Chrome and Mozilla Firefox- Linux Ubuntu) and saw no problems between in the server in both browsers. SQL database was used to store other COMPSYS 302 online user’s details so the information can be accessed through the database when we needed information about a specific user etc. The online user list would refresh every 10 seconds to prevent the list becoming out-dated, which could cause potential errors; connecting to an offline user. Lastly, a lot of time was spent towards the class protocol in and out of lecture. A lot of the time was spent in UG3 discussing about the possible class protocol multiple times with other team members and talking about the advantages/disadvantages of the possible options that were brought up. The final class protocol came close by to our protocol document which made it fairly comfortable to work with as there weren’t so many dramatic changes from what was initially brought up from the discussions.

# Top-level view of System

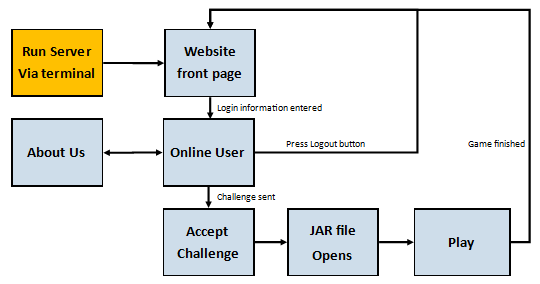


Figure 1: Top-Level Diagram of the Server

The diagram above is the top-level diagram of the web server that was developed. To start the web server, the player is required to run terminal on Ubuntu and find the directory of the server code to run the website. After the server is successfully running, the program will automatically direct the player to the front main page with the ‘IP: Port no#’ (e.g. 10.103.137.68:10001) entered into the address bar of Google Chrome or Mozilla Firefox (default browser) automatically by the Cherrypy code. Then the player will have to login to the server using their UPI and ID number as the username and password respectively. After login, the player will have an option to logout of their account or search for online users to play against. When the player sends a challenge to the opposition by clicking on the challenge button underneath their opposition’s UPI/information, the application will run using the socket communication implemented into the server. After the game ends, the player would be automatically logged out of the web server. This would require the player to log back into the server to be able to play with other users; all for security reasons. One of the biggest security problems that could occur is that the player could forget to log off the server after playing a game, which could allow other users who have access to that same computer to use the previous player’s account to play which is representing poor security of the system. An ‘About Developers’ page was added into the website where it will briefly introduce the two developers of this server/game (Group 1 members) and give a little insight on the developers.

# Screenshots

Figure 2: New In-Game UI

Figure 3: Old In-Game UI

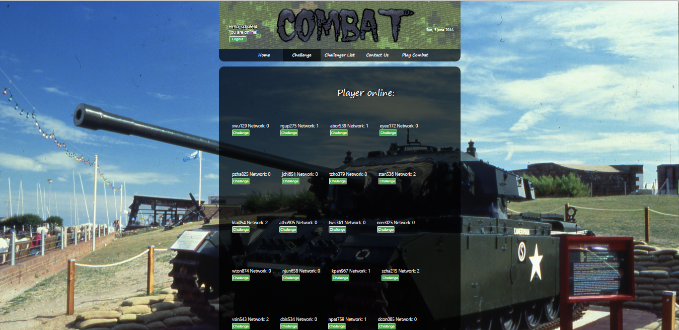
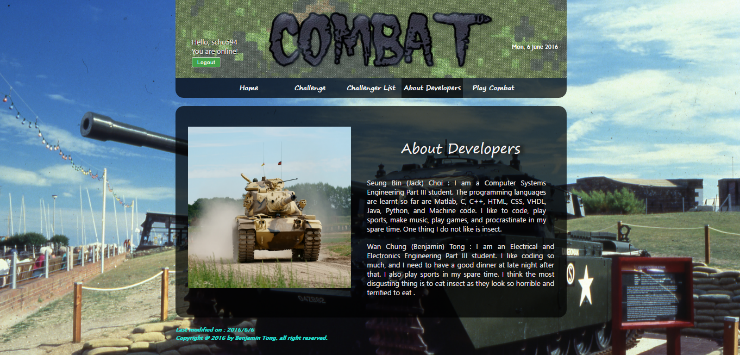
Just a short comment on the new in-game UI of our game, a lot of features were improved in our game. First one is that it shows which tank is which player but also shows the score and power up state at the bottom for each player for better gaming experience. The texts are also half transparent to not disturb the players during the game, but still showing essential information (HUD).

Figure 4: Front main page

Figure 5: Challenge/Online user list

Figure 6: About Developers page

Figure 7: Pop-up Notification

Figure 4 shows the first page the user will get to when they first run the terminal. The webpage will automatically load to this page then when the user logs in and it will show the UPI the user is logged in with (should be the user’s own account) and a logout button under it to be able to return back to the front page. The page will advance to the online user list right after login (figure 5), which will show the people who are also logged into the server (Do keep in mind, challenges can only be sent to the people with the same location as us: University Desktop, University WIFI, Rest of the world). The last screenshot is the small pop-up notification for sending challenges to other users. It will appear after the challenge is sent and it will show the user if it was challenge sent was successfully or not (figure 7). Challenger list page (figure 8) is used to check for any challenges from other online users. The user then could accept the challenge and the Java application will be launched on both computers.

Figure 8: Challenge responding page

# Significant Issues and Solutions

* First of all, as stated in Phase 1, both of us are in different specialisation so there were problems meeting up and progressing with the work. Ben was busy with ELECTENG 310 second group project and Jack was busy with COMPSYS 305 VHDL group project which both finished around Week 12/13. Due to the specialisations and group project due dates, it was hard to start and progress on the second phase. This was slightly solved by working harder when both of us were done with our other course projects.
* Secondly, the project itself was extremely difficult to implement and the minimum requirements for C grade was insane itself. It was very hard to implement the Java application into the Cherrypy server and through API to other users, yet alone having to make a perfectly working multiplayer system was overwhelming. This issue wasn’t overcome; hence the last minimum requirement was not met.
* For the given time, this project was very difficult to get through. Just to mention, the class protocol was roughly finalised in Week 11 and having to finish the tasks given in 2-3 weeks was insane, especially where most of the COMPSYS 302 students also had COMPSYS 305 project due with a presentation and report as well in Week 12. Timing was one of the most critical factors of this phase and the difficulty of the project also was more than what was expected. Timing issue wasn’t really solved by the end of the project as well.
* Lastly, the majority of the class was really worried when they started the phase 2 coding and realised that it was really hard. Due to this, quite a number of groups were giving up and turned our class mood into a disaster. During our coding sessions in the UG labs, half of our course members were demotivated and were worrying about the project deadlines, which also affected us and were very discouraged from such working environment.

# Features

* A good amount of time was taken up to produce an impressive design for the web server, which was made through HTML/CSS (and JavaScript) files (separate from the Cherrypy server). HTML code was separated instead of mixing it into the Cherrypy code to explore the full functionalities of HTML/CSS by itself and also for the simplicity of the code as it is easier to read when they are separated.
* Just an addition to the design part of the server, a few images here and there were added throughout the web server so the user experience can be enhanced. Only having words all over the page would not draw attention of the user.
* A simple and easy to use navigation system was implemented on our web server which will allow the player to navigate through the website with no big issue and everything will be clear and self-explanatory.
* SQL database was coded to generate every time the server runs to store information of current online users and will keep them in the database. This will allow faster and efficient use of information needed as the database is saved locally. This will eliminate unnecessary waiting time and resources while trying to obtain information through the server every time.
* JavaScript was used to develop a pop-up notification feature for the challenge function and also for the date feature that was added on the top-right side of the page.
* Extra navigation tabs were implemented which has brief introductions of developers, and there is a main page (Home) which will briefly explain what the web server is for.

# Network model Discussion

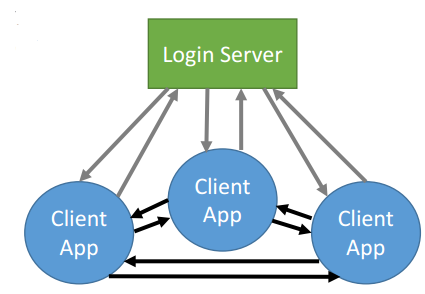


Figure 9: Network model (Hybrid)

According to the class protocol for the Cherrypy server, it was finalised to use the hybrid network model where the login server (made by Andrew Chen) would contain all the user’s account information. The users would have to login to Andrew’s server using their UPI and password to be able to access the information of other users in the course. Once logged in, the user needs to send a ‘challenge’ to another user to start the game with the opposition. Once the game starts, the information between the two users will no longer go through the login server (Andrew’s) but will operate in a P2P manner. This will be done by the two users communicating to each other directly using API where it will send key presses and object positions among each other. The P2P suitability for this project was a good protocol as it will be faster to operate with no ‘lag’. If all the information were to go through the main server, it might cause an overload of information for the server to handle, which could cause slow/inefficient data transfers. No other problems or disagreements were detected on any part of the network model as it was exactly what was preferred before the finalised protocol was released.

# Protocol Discussion

In terms of login protocol, no problems were found with any of the proposed points as they were fairly straight-forward and easy to understand. The login server was made by Andrew Chen, so it was reasonable that he would create the suitable login protocol for us. It was suitable for our project as it covered all the information that was needed and looked like most other groups were also happy about it as most groups (including us) finished the login protocol. The class protocol was also finalised to use the Master-Slave system where the user sending the challenge will become the master and the one receiving the challenge will become the slave. The only difference between the two is that the master will be in charge of all the computation of the game logic and sending the object positions to the slave where the slave will only send the key presses to be computed on the master code and render the objects received by the master. Nonetheless, the application protocol was reasonably fine with not many issues or big discussion worthy points. A problem with the application protocol other than the suitability, would be that it wasn’t finalised even till Week 12/13, where it was constantly changing (minor changes) throughout the weeks, which were confusing a lot of us in general and making the students overcomplicate the changes. But in reality, the changes were necessary and it was crucial that those changes were made in order for us to progress on with the server.

# Tools

The tools that were used in Phase 2 were: Java (minor), Cherrypy (Python), HTML, CSS. Java was used for the game and also for the connection between the game and the web server. To enable the communication between Java and Cherrypy, the Java code needed minor changes and additional code to be able to accept data in and out with Cherrypy. The main tools used for Phase 2 were the other three. HTML and CSS was used for the front-end development of the web server, which was in charge of the design and layout of the web server. On the other hand, Cherrypy was used entirely for the back-end development of the web server, where it was in charge of the server functionality. Cherrypy was a good tool to use for the web server as it was not too difficult to use and a lot of the complicated parts of the server network was done automatically by Cherrypy itself, leaving us to only code for the functionality of the server. The only problem with using Python was actually connecting the server and the Java together. It was very difficult in doing so; hence not being able to achieve the last minimum requirement. HTML and CSS was really simple and straight-forward to use and there were no problems using those at all, but some JavaScript was included to increase the functionality of the pop-up notifications and date features.

# Future Improvements

* One thing that should be developed furthermore in the future from our currently finished project is actually finishing all of the given specification on the web server. Due to lacking time and insane difficulty, it was really tough finishing even a quarter of the listed tasks.
* Including more attractive designs on the website and better functionality between the two users during gameplay would also be considered. For example, having a messaging and live score system on the browser is something that would be developed if given more time.
* Finally, making the web server completely P2P was also something that was brought up for future improvements as relying on the login server is too risky. The server may go down if not maintained properly or it might not be operated anymore; which would make our web server useless as they cannot obtain each other’s API/information anymore.