1. Project's summary

a. Short description of the game

i. This is Space Wars! Build, upgrade and stay two steps ahead in order to defend your planet from relentless enemy threats. Crafted from the ground up by three young promises, this game challenges you to manage your metal and deuterium resources to build powerful fleets and planetary cannons. Your mission is as clear as day: protect your home planet from incoming, ever-growing enemy fleets. Raze the battlefield, think ahead and stay resolute. Good luck, commander!

b. Project's goals

i. The main goal of this project was to consolidate and apply everything we've learnt from the past year's class lessons in a comprehensive and unified manner. We have hardened our skills in every field, including programming, database management, web design as well as teamwork and communication.

c. Tools used (IDE, languages, libraries, frameworks...)

i. For this project we've made use of a handful of tools, which we will categorize based on their respective fields:

1. **Database Management:** We've used AWS (Amazon Web Services), specifically the **Aurora and RDS** services. Using these, we created a DB instance with open firewall rules to ensure seamless connectivity with other tools and across different locations, such as our home, the workplace, or school. To develop and monitor the database, we used MySQL Workbench.

2. **Programming:** The main programming language used in this project is Java. The two main libraries we used for the Java side are **javax.swing** and **java.sql**. With **javax.swing**, we developed the graphical user interface (**GUI**), while **java.sql** facilitated connections between Java and our database. We also incorporated java.io.**IOexception** to handle exceptions and manage potential bugs and errors in the game, as well as server-side issues.

3. **Web Design**: For the website, we used HTML and CSS to organize and stylize the layout, and Javascript to enhance the website's appearance. We also made use of some AI tools and Adobe Photoshop to create copyright-free images tailored to our needs.

2. Analysis and design

a. Functional and nonfunctional specs

i. The game can be controlled either on a console-based interface or a graphical one.
ii. On the graphical and console interface the user can build units, upgrade their technologies, view battle reports and visualize the enemy's army.
iii. The game is a tower defense-ish style of game, that means the game keeps playing infinitely until you lose
iv. An enemy's horde will spawn in 2 to 4 minutes, that's when you'll detect the horde. After that, you'll get 100 seconds until the horde attacks. You'll get to see the enemy's troops when the timer gets to 50 seconds.

b. Game's structure (screens, states, scenes...)

During initial startup, the game prompts the user to select between the console-based interface or the graphical user interface.

i. Structure common to both interfaces
In both implementations, resource generation and enemy army spawning are managed through scheduled tasks utilizing `java.util.Timer` and `javax.swing.Timer`, which track elapsed time to trigger periodic game state updates.

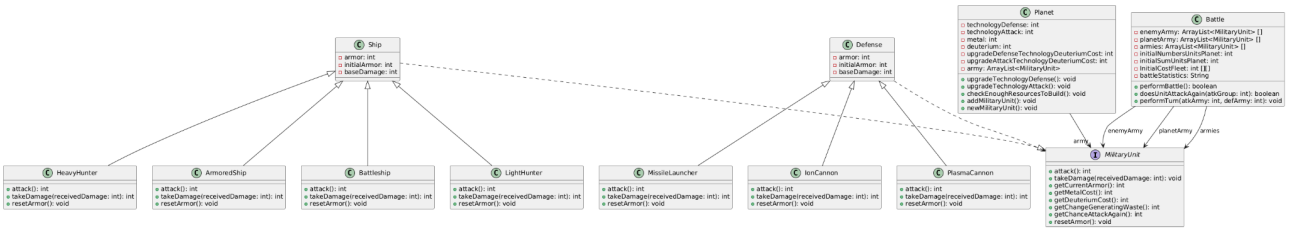
Additionally, both variants employ the class Game.java for the storage of game data, including planetary information, battle statistics and attributes of military units from both allied and enemy forces.

ii. Console-based Interface
The console interface enables user interaction through a main menu. Apart from terminating the program, it presents a total of five options, with sub-menus and contextual prompts displayed as necessary.

iii. Graphical User Interface
The GUI consists of a primary screen divided into four quadrants.
The upper-left quadrant displays information related to the planet, including the amount of resources available and the technology levels in offense and defense.
The upper-right quadrant shows our current army, specifying the amount of units of each type.

As for the enemy armies, the lower-right quadrant visualizes their approach toward the planet using a progress bar, as well as the count of each type.
Lastly, the lower-left quadrant contains a console-style log that provides the player with certain notifications, such as newly generated resources.
To display the battle reports, the user can open a secondary window by using the Options button in the top-left corner of the screen.

c. UML diagrams



3. Architecture and development

a. Source code's structure description

i. The source code is divided into different classes. With the interface **Variables** for every fixed value we need like the amount of metal you get every minute, we also have a class for each aspect of the game. The battle, the defense, the console and graphical UI... The game class manages the primary methods to start the game, while the main class initializes the UI and the game class to start the game itself.

b. Main components' explanation

i. The player is the commander of a planet which has its technology and units. The HUD facilitates the display of the stats the player needs to see, like the planet stats or the enemy's incoming army. The HUD also has a console so that any message that couldn't be naturally displayed (like the automatic generation of metal or an error if you can't build units) is inserted into the console.

c. Project/Files's diagram.

```
Proyecto_FinaldeCurso_CMC/
├── M1/
│   └── Project's documentation)
├── M2/
│   ├── Script_tablas.sql
│   └── Script_valores.sql
├── M3/
│   ├── ProgramaJava
│   └── imagenes/
│       └── imagenes del juego
├── M4/
│   ├── css/
│   │   └── estilos.css
│   ├── html/
│   │   ├── index.html
│   │   ├── about.html
│   │   ├── tutorial.html
│   │   ├── batallas.html
│   │   └── battles/
│   │       └── battleX.html, battleX.xml, battleX.xsl
│   └── imagenes/
│       └── imagenes para la web
├── M5/
        └── Diagrama de clases
```

4.Testing and cleansing

a. Tests done:

i. We created sample values to test the output in Java and SQL queries, and we used trial and error with the HTML and the game to see if everything was working properly.

b. Errors solved:

i. When it came to connecting it to the HTML to display the battle data, there was no way to make it so that when we generated the XML file and transformed it into HTML, it would appear in the battles HTML page.

ii. When we were adding units to the array from the GUI, it would only throw an exception or add said units, not the two altogether.

c. Improvements applied during development:

i. We wanted it so that instead of downloading an HTML file with everything that happened in the battles, the content would be displayed directly in the battles HTML page, without having to download a separate battleX.html file to view a specific battle. We wanted this to be automatically displayed when you click the HTML button that says View report.

5.Graphic and audio resources

a. Sources used (with copyright if applicable):

i. Most of the images used were found online and are originally from various video games. They are used here for educational and non-commercial purposes only. All copyrights belong to their respective owners.

b. How they have been integrated into the project:

i. You can implement it in the project using routes or folders, if you download the images.

6. User's manual

a. console mode:

i. Run the main class from IDE (eclipse).

ii. A simulation will launch the console mode where you can:

1. View your army and resources.
2. Create defensive and offensive units.
3. Upgrade your attack and defense technologies.
4. View upcoming enemy fleets.
5. Check reports from previous battles.

iii. Some actions happen automatically:

1. Resources generated every minute.
2. Enemies incoming

b. Graphical Interface:

i. The graphical version of the game mirrors the console functionality.

ii. All options are accessible through a user-friendly interface (not included in detail here).

c. Controls and objectives of the game.

Defend your planet from invading enemy fleets and become the most powerful commander in the galaxy.

You must:

i. Build and upgrade military units.

ii. Manage resources.

iii. Improve your attack and defense technologies.

iv. Win battles to collect space garbage and obtain more resources.

7. Reflection and Improvements

a. What difficulties were encountered and how they were overcome.

i. Its hard to be 3 persons and we have our differences like, we don't do the same for the same problem. This was one of the main conflicts that we have, however, we overcame this by:

Listening to eachother and explaining our solutions, dividing tasks based on our strengths and interests and using github to manage changes and reduce conflicts.