# Process Plan Documentation

## 1. Introduction

The software project we will be developing is a variation of the classic game *BattleShip* named *FleetDestroyer*. The goal of this project is to include several key features onto already captivating game to market it to the emerging young crowd. The customer base we are intending to target extends to anyone who is looking for entertainment or to relive their boredom.

This product will facilitate 2 player battleship, holds usernames as well as passwords, maintains leader board, integrate in-game chatting, and provides an appealing GUI (Graphical User Interface) with background music and highly interactive button employment.

The main feature we will be able to employ granted enough time is *Level-Ups*, unfortunately due to the time constraints we could not implement a working component onto the project.

## 2. Process Description

The driving goal throughout the whole lifecycle of the project was to mimic the Waterfall Model where we will have clear goals and soft due dates to progress towards project implementation and maintenance. As we delved deeper into project this approach did not synchronize well with the rest of the group. After attempting different models we agreed that the Iterative and Incremental model would be the most effective due to the time constraint and the level of our coding experience.

This approach improved our production as we could add key functional elements onto to the mainframe of the code and constantly test it until we were satisfactory of the results.

**Processes Modules**

- Most of our design modules didn't change but we did have to cut a few out due to a lack of time
    - Design Layout: We had our game planned before we started coding
    - Graphic Interface: The only thing that we could not include in our graphics were the animations for each action that you perform in game
    - Game Engine: The initial idea for this was just to have one class that was the game engine to deal with all of the computation for the game. We split this up into between the client and the server
        - The server deals with the numbers that we receive from the user
        - The client has the "engine" that runs, this includes all the graphics that happen on the actual computer and the client sends the user input to the server to interpret what happens next
    - Matchmaking Algorithm: This was one of the modules that we couldn't implement into the game, instead of an algorithm that matches players up with their current level, the game just sets up users at random
    - Account Configuration:
        - each user gets their own username and password so that only they can access their accounts
        - Each account is able to chat with any other account that is logged into the game
        - The main components of this module that we couldn't implement were:
            - Friends list
            - Recent Play History
            - High Score

Reward/power up System: This was another module we could not implement in the game. Users were supposed to level up based on how many games they played but we didn't accommodate enough time for this

## 3. Roles

| | | |
|---|---|---|
| Hadi Ghantous | Project Manager/System Architect | Took the responsibility of managing the group, setting up deadlines, backend infrastructure and server implementation |
| Sai Yarram | System Designer/Developer | Adding Graphics, sounds, images. Integration of Server and GUI of all the classes. |
| Nick Raymond | System Architect/Developer | Graphical Interface of the GameEngine. Integration of all the logic behind the game along with the GameEngine (The Board). |
| Michael Audi | Writer | Helped with some documentation. |

Our roles changed a little bit as we progressed throughout our project, it turned out to be a

## 4. Estimates

Our initial estimation was to finish the project within 4-6 weeks but we underestimated the complexity of the project. It took us longer to actually implement all the core features as well as have it display the GUI in an appealing fashion. In addition, the team ran into problems involving the server and client implementation, we were unsure of which approach was the most effective as well as easiest to implement.

| Estimation | Initial | Actual |
|---|---|---|
| Hours | 60-100 Hours | 90 Hours |
| Code | 1500 lines | Around 3970 |
| Defects | 50 | 150 |

## 5. Schedule

| Start Date | End Date | Task | Task Description | Required Roles |
|---|---|---|---|---|
| 10/16/16 | 10/19/16 | Create Board | Create a GUI where we present the board | Developers |
| 10/16/16 | 10/21/16 | Game Logic | Create basic Logic as to setting up the ships on the board | Developers |
| 10/19/16 | 10/23/16 | Server | Consider server implementation | Architect |
| 10/20/16 | 10/27/16 | GUI | Fix, add, edit, remove to make the GUI more presentable | Developers |
| 10/24/16 | 10/29/16 | Server | Starting the server, basic connection between them | Architect |
| 10/29/16 | 11/3/16 | GUI | Add other screen such as Login, UserName, Initial | Architect/Developers |
| 11/3/16 | 11/12/16 | Demo | Work on getting the basic version of the game | Architect/Developers |
| 11/17/16 | 11/29/16 | Server/GUI | Connecting the Server and the Client | Architect/Developers |
| 11/29/16 | 12/14/16 | Demo 2 | Attempt the second Demo and add all other components | Architect/Developers |

## 6. Reflection

The project gave us a delightful insight on how to approach and tackle a large Software Project as well as manage our time carefully. It forced us to go beyond our comfort zone as it gave us a vast array of options on how to go about it which was not what we were used to. Integrating so many components together and have them flow in a sequential manner was also a problem which we ran into quite often.

If we were given a chance to undertake this project or a similar project, there are new several approaches which we would implement as well as omit, to make sure the project would sail smoothly.

When we undertook this project, we did not use all the resources available to us and the key resource being GitHub. After our initial demo, we began to implement GitHub onto our development approach which greatly expanded our ability to communicate our code and progress instead of using Email or DropBox. This is something which would absolutely help us communicate more efficiently with the rest of the group and manage our time.

As we explore the concept of time, our initial phases and requirements were vastly underestimated and not clearly described. We assumed our implementation of features should not be too hard but as we progressed it was hard for us to differentiate the direction and the approach we were going to take. This led to miscommunication within the group with the programmers approaching each problem in their own way. We believe if we set concrete and clear goals as well the approach we should all take to arrive there then we could have easily avoided wasting time solving the same problem and coming up with different solutions.

Finally, we most definitely did not use all the libraries, packages, and code available to us. Rather, the team began to code everything from scratch which led to draining our most valuable resource which was time. Given a chance, we would approach the GUI in a more easy and applicable way through JavaNetBeans where we could draw out the components. Although we would implement a host of changes in our future projects, we strongly believe that our core principles were strong and with the minute changes described we could have easily put the project on FastTrack!