Strategic Web Based Management Game

Group 12

Per Eriksson

Per Strand

Simon Ragnar

Ingemar Markström

Max Walter

Design Document

1 Preface

1.1 Version history

Version	Comment (reason for / summary of changes)	Date	Author(s)
1.0	First version	2008-03-09	Per Eriksson Per Strand Simon Ragnar Ingemar Markström Max Walter

1.2 Expected readership of the Design Document

The expected readership of this document are software developers.

Table of Contents

2 Introduction

This document provides a thorough basis for how the system of this web based game will work (Requirements Document section 3) on all levels. This includes the presentation, logic and data layers. Having that said, this document will take into account the technical aspects of the system but only a rough design of the Graphical User Interface.

The document's intended audience is the developers of the system and their supervisor and the purpose is to give them a clear picture of the system design for the implementation phase.

Any reader of this document should have access to the Requirements Document to be able to follow up on references in order to better understand this document.

2.1 Abbreviations and acronyms

RD	Requirements Document, a document describing all functionality
GUI	Graphical User Interface
PHP	A web development language
Sec	Secondary
Rec	Resource
AKA	Also Known As

2.2 Important terms

Term	**Description**
Space ship, AKA The Ship	This is the vessel that the game revolves around. Everything you can do is done by using the Ship.
Wormhole	The goal of the game. Enter the wormhole first and you are the victor.

Term	Description
Modules	The only mean of gaining new abilities and bettering your old ones is to build and upgrade your modules or research their efficiency. There's a total of eight modules: - Missile battery module - Cannon module - Teleportation module - Missile decoys module - Storage module - Engine module - Powerplant module - Repair module
Web browser	An application that has the ability to show web-pages e.g. Firefox.
Web forms, AKA forms, web-page, page	The GUI of the game. Can contain PHP code.
Game logic, AKA logic	Pure PHP forms. Used to separate the game logic from the GUI or the web forms.
Gameround	A gameround starts when the developers choose to start it. A gameround ends when a players reaches the Wormhole.
Game map	The Game map is a part of the GUI and is also an own class in the class structure of the game.
Session	A session is determined and handled by the user's web-browser. In essence it is variables saved by the the user's web-browser for a specific period of time or for as long as they are necessary.
Session data	Data that are stored in the server memory about the user using PHP objects.
Game object	Game object data is data about the player or his/her surroundings that can be altered by other players in the game and thus always confers to the database for all data usages.

Term	Description
User data	User data is data that only the user can alter. User data confers the database the first time the user logs in and when the user changes his/her User data information.
Web-based	Something that can be accessed through the internet using a web-browser or similar program.
Server	Is called through internet to return information to the caller (client)
Client	Calls a server to get and show information.
Database	Stores information on hard drive.
Control flow	Issues orders or calls to other parts of the application
Data flow	Sends data to other parts of the application.
Massive multiplayer game	A game that has the ability to serve more than 32-64 players at a time (up to several thousand) in the same game, i.e. on one and the same server and in one and the same game universe.
Strategy game	Typically a strategy game is a game where there exists a map of some sort where the player can change the content of the map in some way.
Role-playing game	Typically a role-playing game is a game where the player is in control of a character that has the ability to evolve. The character can take the shape of a person, monster or even a space ship with crew and all.
Forum	An application where players can post messages in order to discuss various topics online.
Highscore	A list sorted on the most successful participants in falling order.
Escape points	A point that all players get to give a rough estimate on their ships abilities (or power).
Close-to-center	In this case a list sorted on players closest to the wormhole in ascending order.
Module slots	A ship can not house an infinite amount of modules. When the module slots are all occupied no more modules can be built.
Ship condition status, AKA condition status	An indication on how damaged the ship is.
Stars	A star is gained each time the player passes a pre determined amount of Escape points. Stars can be used to boost research on different modules.

Term	Description
Ranks	A rank consists of a number and a player name where the number represents the players position in comparison to other players in ascending order.
(Map) Focus	Focusing means that the player has targeted another player in the game with his/her weapons or teleport.
(Map) Search	Searching on the map means that the player types in another players name and that player is shown on the map.
(Weapon) Capacity (Storage)	The storage capacity of different weapons munitions.
.logic	Is a class containing code that decides the events in the game.
.presentation	Is a class that has code that determines the GUI of the game.
.data	Is a class that has code that sets and gets game information for the game logic using a database.
Map objects	Are objects that can be shown in the game map.
Map square	The game map consists of squares called map squares.
(Map) Stackable	If a square is stackable, the game can put a new object on that square.

2.3 Abstract

Section	Description
2. System Overview	Provide an overall and detailed overview of the system architecture
3. Design Considerations	Describe issues which need to be addressed or resolved before attempting to device a complete solution
4. Graphical User Interface	Provide a rough Graphical User Interface of the system
5. Design Details	Provide a detailed design of the system
6. Functional Test Case	Provide Test Cases for the system

3 System Overview

3.1 General Description

3.1.1 General technical description

The strategic web based management game is as the name suggests, a game that you can play on the web. This means that wherever you are, all you need to play this game is a computer with an internet connection and a web browser (in our case Firefox). The system uses a client-server architecture. All information about the participating players are stored in a database.

3.1.2 General game description

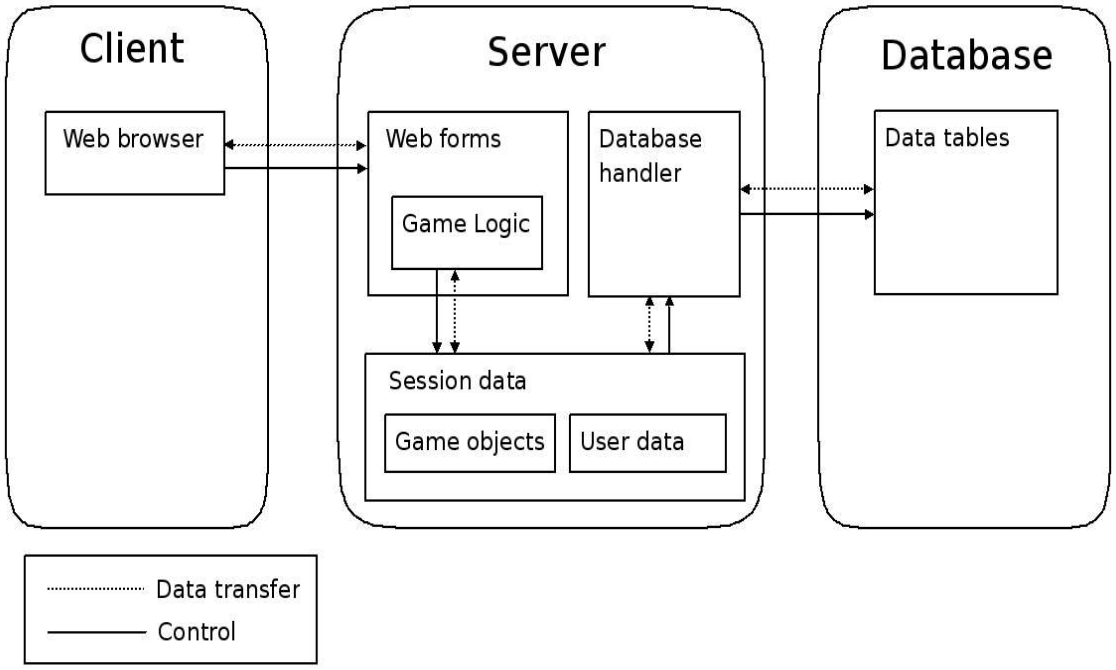
The game is set in a futuristic world where players find themselves trapped in an other dimension with a space ship, where the only means of escaping the dimension is by entering a wormhole. All players will start in a circle and the wormhole will be located in the center of the circle. The players will compete over who will reach the wormhole first, and the first player to reach the wormhole is declared the winner. It is possible for new players to enter the game even after the game has started. Once a winner is declared the game restarts.

3.1.3 Detailed general game description

As mentioned, all players will be in control of a space ship. At the start of the game the space ship has three basic properties, i.e. the ability to generate power, the ability to gather resources from space and the ability to move. The ability to gather resources is a quality that all ships have and can not be changed in any way, however the other two attributes are directly a result of the ships corresponding “modules”. In this case the ship starts with an Engine module, Powerplant module and a Storage module.

All ships have the capability to build and upgrade modules and to research module efficiency. There are 8 modules in total. Also, all ships have a maximum amount of slots on where to build modules.

3.2 Overall Architecture



The system will be a web-based game consisting of a server that clients can connect to through a web-browser. Users will play the game by navigating through, and interacting with, the web pages. In turn, the server will be connected to a database.

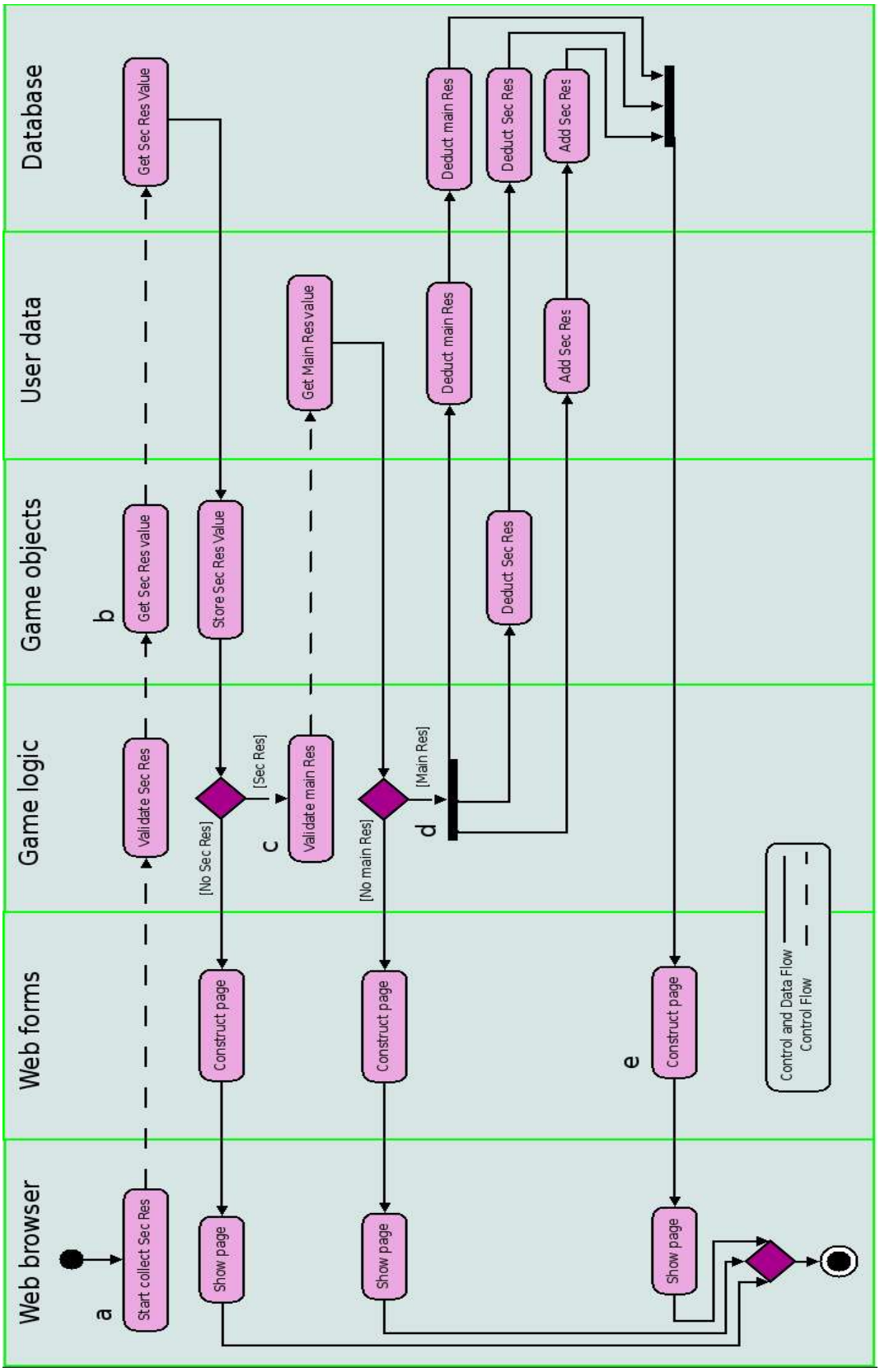
The Game logic section on the server creates, and/or changes Session data based on the actions requested by the user. The Session data will be stored on the server and will contain information regarding the user (User data) and other necessary game information (Game objects). All Database inquiries will be handled by the Database handler, which consults the Database. Eventual data needed for the inquiries will be fetched from the Session data.

Other than the graphical user interface (web-pages), each form will consist of Game logic. The game logic governs all actions in the system, apart from the Client's actions and the Database inquiries.

3.3 Detailed Architecture

This section will describe the control- and data- flow between the system's components using Use case Gather Resource (Gather Resource: Requirements Document, page 43) as an example. The Use case will be presented by an Activity Diagram defined in the Unified Modeling Language (UML). The alphabetical letters in the picture relates to the corresponding sections in the event description further down.

In this picture we've fused the Database handler with the session data objects. All database queries will be handled through the Database handler, which consults the Database.


Database
User data
Game objects
Game logic
Web forms
Web browser
Get Sec Res Value
Deduct main Res
Deduct Sec Res
Add Sec Res
Get Sec Res value
Store Sec Res Value
Get Main Res value
Deduct main Res
Deduct Sec Res
Add Sec Res
b
Validate Sec Res
[No Sec Res]
[Sec Res]
c
Validate main Res
[No main Res]
[Main Res]
d
Construct page
Construct page
Construct page
e
a
Start collect Sec Res
Show page
Show page
Show page
Control and Data Flow
Control Flow

(Sec = Secondary, Res = Resource)

Now follows a detailed description of all Data transfers and Control flows in the Use case.

a

Control flow: The user sends a request through the web browser to the server that he/she wants to gather the secondary resource (Secondary Resoure: Requirements Document, page 23). The Game logic validates the amount of secondary resources that are available at the resource Square (Resource Square: Requirements Document, page 23).

Data flow: None.

b

Conrol flow: The Game object loads and then stores the amount of secondary resources available at the resource square from the database and returns the data to the Game logic. The Game logic validates if there are resources to gather. We will assume that that is the case.

Data flow: The Database sends data to the Game object of interest and in turn to the Game logic.

c

Control flow: The Game logic loads the user's amount of main resources (Main Resource: Requirements Document, page 23) from User data. This data will be independent from everyone except for the user. Therefore no database enquiries are necessary. The Game logic then validates if the user has enough resources. We will assume that that is the case.

Data flow: User data sends data to the Game logic.

d

*Control flow:*The Game logic deducts a specified amount of main resources from the user. It also deducts a specified amount of secondary resources from the Resource square and adds it to the User data. The Game object, User data and the Database performs the operations described in **b** and **c**.

Data flow: Values that are to be deducted and added.

e

Control flow: The Game logic generates a web-page and sends it to the user's Web browser.

Data flow: The web-page.

4 Design Considerations:

- Works with the latest version of Firefox
- The system shall be implementable in PHP

Assumption on the main user:

- Likes to play web-based massive multi player, strategy and/or role-playing games on PC or console
- Likes to play games that are played for short time intervals
- Is competitive

Possible and/or probable changes in functionality:

- Make it possible for a player to obtain several spaceships
- Add additional kinds of resources/weapons and so on
- Add more wormholes
- Have parallel game rounds
- Add map obstacles
- Make it possible for players to loot other player's ships
- Donate money
- Buy special features
- Write messages on missiles
- Add more modules

5 General user interface information

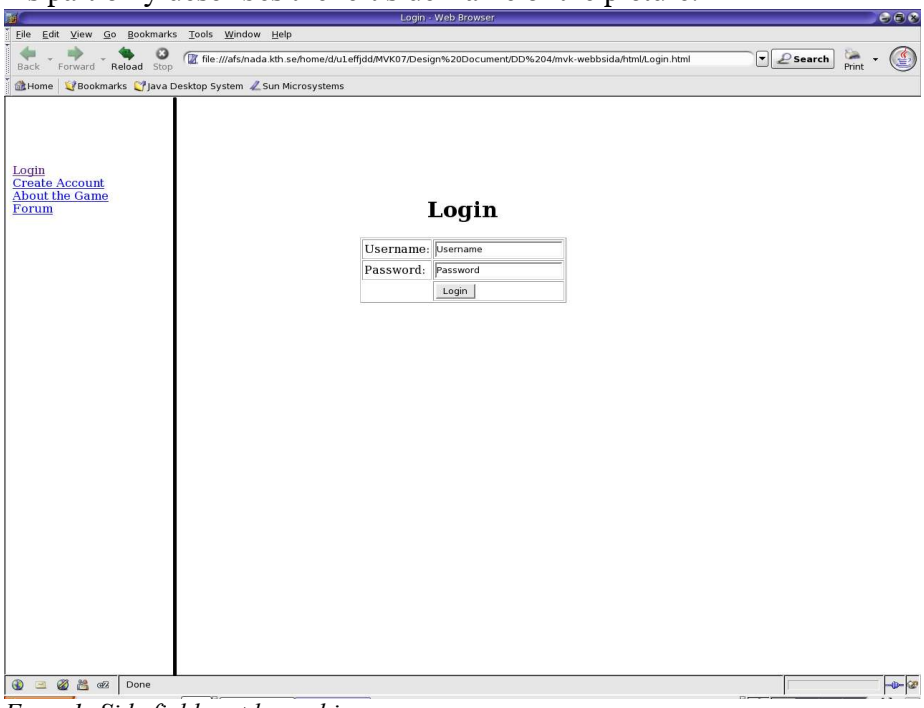
About the forum: The forum itself is outside our development scope, which means that we will not develop a forum ourselves, but rather use a ready made forum. Therefore there are no form sections or any other pictures describing the forum itself although there are lots of references to it.

The user interface in this game is divided in two sections, logged in and not logged in. When the user has not logged in, he/she can only use the forum, read about the game, create a new account and login.

When in the Logged in state of the game the user has a multitude of choices. Many of the choices are presented as links in the top and left areas that are always visible as long as the user is logged in. There is also an area that always displays general information about the player and his/her ship. Other than this there are many pages that can only be reached through other pages. A few examples of such pages are the different module pages that can only be reached by clicking on different areas of a picture on the Ship page.

5.1 Form 1: Side field, not logged in

This part only describes the left side frame of the picture.



Form 1: Side field, not logged in.

List of references to RD:

Create an account, section 7.1.1.1

Log in, section 7.1.1.2

Game instructions, section 7.1.1.3

Forum, section 7.1.1.4

The names of the controls and fields:

Links:

Login

Create account

About the game

Forum

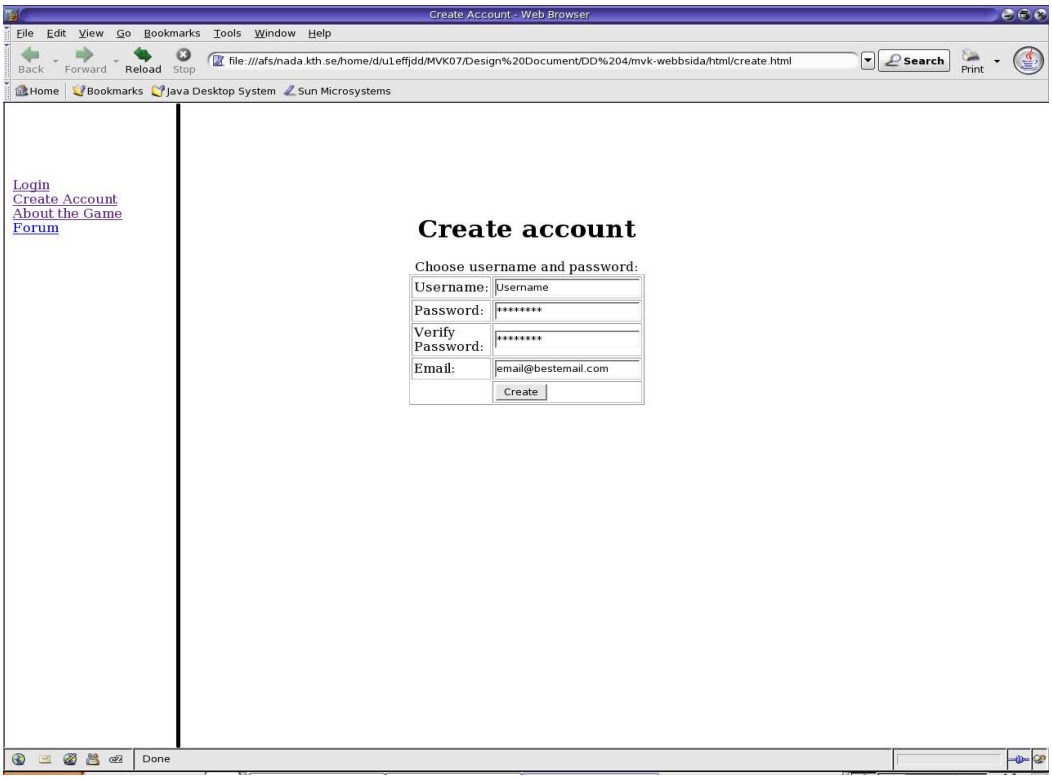
The names of the events, methods, or procedures that cause this form to be displayed:

This form describes the side fields that are always displayed in all Forms when not logged in.

The names of the events, methods, or procedures triggered by each control:

Login:	Calls Hyperlink logIn
Create account:	Calls Hyperlink createAccount
About the game:	Calls Hyperlink aboutTheGame
Forum:	Calls Hyperlink forum

5.2 Form 2: Create account



Form 2: Create account

List of references to RD:

Create an account, section 7.1.1.1

The names of the controls and fields:

Text fields:

Username

Password

Confirm Password

Email

Buttons

Create Account

The names of the events, methods, or procedures that cause this form to be displayed:

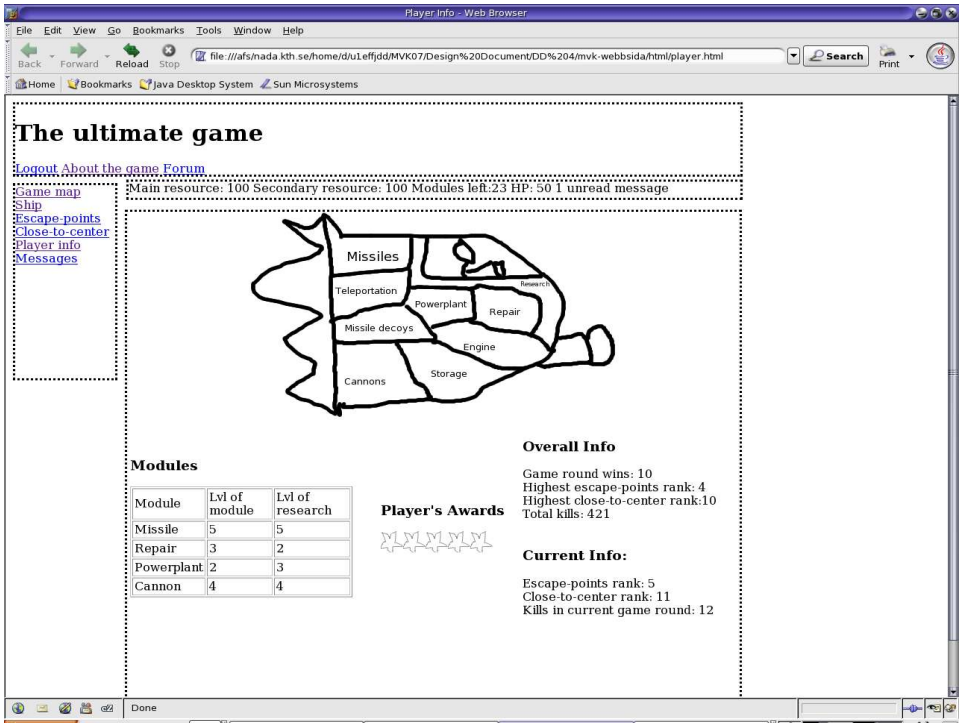
Click ”create new account” on the Form 1: Side field, not logged in.

The names of the events, methods, or procedures triggered by each control:

Create Account:	Calls function createAccount()

5.3 Form 3: Side fields, logged in

This part only describes the side and top frames of the picture



Form 3: Side fields, logged in

List of references to RD:

Game instructions, section 7.1.1.3

Forum, section 7.1.1.4

Game map, section 7.1.2

The ship, section 7.1.4

High-score list, section 7.1.6

Main resource, section 7.1.3.1

Secondary resource, section 7.1.3.2

Text messages, section 7.1.8.1

Names of controls and fields:

Links:

- Log out
- About the game
- Forum
- Game map
- The Ship
- Escape points Highscore list
- Close-to-centre Highscore list
- Player info

Information fields:

- Main resource
- Secondary resource
- Remaining module slots
- Ship condition status
- Messages

The names of the events, methods, or procedures that cause this form to be displayed:

This form describes the side fields that are always displayed in all Forms when logged in.

The names of the events, methods, or procedures triggered by each control:

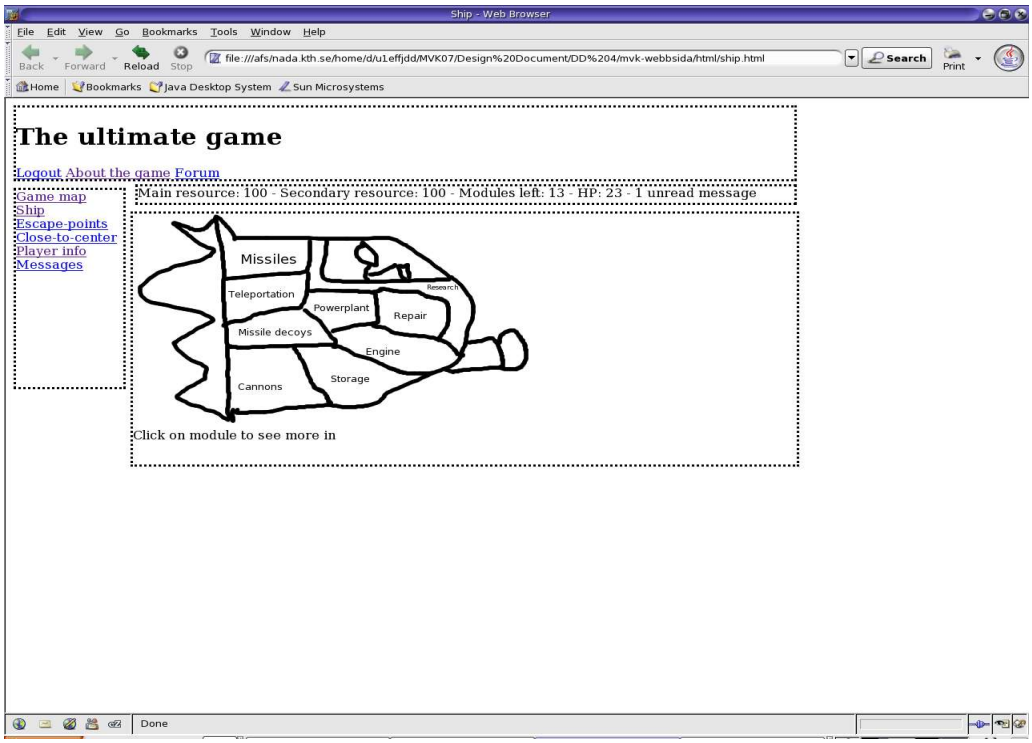
Log out:	Calls function logOut()
About the game:	Calls Hyperlink aboutTheGame
Forum:	Calls Hyperlink forum
Game map:	Calls Hyperlink gameMap
The Ship:	Calls Hyperlink theShip
Escape points Highscore list:	Calls Hyperlink escapePointsHighScoreList
Close-to-centre Highscore list:	Calls Hyperlink closeToHighScoreList
Player info:	Calls Hyperlink playerInfo
Main resource:	Calls function getMainResource()
Secondary resource:	Calls function getSecondaryResource()

Remaining module slots: Calls function getRemainingModuleSlots()

Ship condition status: Calls function getConditionStatus()

Messages: Calls function getNewMessages()

5.4 Form 4: The Ship



Form 4: The Ship

List of references to RD:

Modules, section 7.1.5

Research, section 7.1.9

Names of the controls and fields:

Links:

Missile Batteries Module

Teleportation Module

Missile Decoys Module

Cannons Module

Storage Module

Engine Module

Power Plant Module

Repair Module

Research

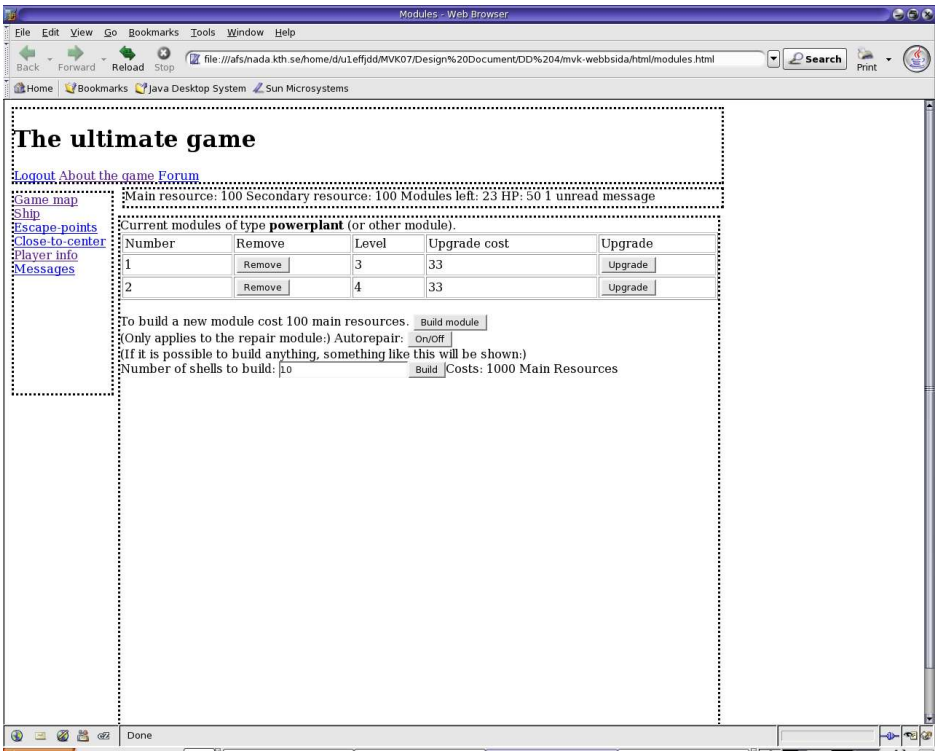
The names of the events, methods, or procedures that cause this form to be displayed:

Clicking on the playerShip Hyperlink located in the side field, logged in (form 3).

The names of the events, methods, or procedures triggered by each control:

Missile Batteries Module	Calls Hyperlink missileBatteriesModule
Teleportation Module	Calls Hyperlink teleportationModule
Missile Decoys Module	Calls Hyperlink missileDecoysModule
Cannons Module	Calls Hyperlink cannonsModule
Storage Module	Calls Hyperlink storageModule
Engine Module	Calls Hyperlink engineModule
Power Plant Module	Calls Hyperlink powerPlantModule
Repair Module	Calls Hyperlink repairModule
Research	Calls Hyperlink research

5.5 Form 5: All module forms



Form 5: All module forms.

List of references to RD:

Modules, section 7.1.5

Names of controls and fields:

Information fields:

List modules

Upgrade module cost

New module cost

Buttons:

Build new module

Upgrade module

Removemodule

Text fields:

(Missile module) Amount of missiles to be built

(Cannon module) Amount of missiles to be built

(Missile decoy module) Amount of missiles to be built

Information fields:

(Missile module) Cost to build missiles

(Cannon module) Cost to build shells

(Missile decoy module) Cost to build missile decoys

Buttons:

(Missile module) Build missiles

(Cannon module) Build shells

(Missile decoy module) Build missile decoys

(Repair module) Auto repair ON/OFF

The names of the events, methods, or procedures that cause this form to be displayed:

Clicking on the respective links inside the "The Ship" form (form nr 4) for each module.

The names of the events, methods, or procedures triggered by each control:

List modules:	Calls function listModules()
Upgrade module cost:	Calls function getUpgradeModuleCost()
New module cost:	Calls function getNewModuleCost()
Destruct module cost:	Calls function getDestructModuleCost()
Build new module:	Calls function buildNewModule()
Upgrade module:	Calls function upgradeModule()
Destruct module:	Calls function destructModule()
(Missile module):	
Cost to build missiles:	Calls function getBuildMissilesCost()
Build missiles:	Calls function buildMissiles()
(Cannon module):	
Cost to build shells:	Calls function getBuildShellsCost()
Build shells:	Calls function buildShells()
(Missile decoy module):	

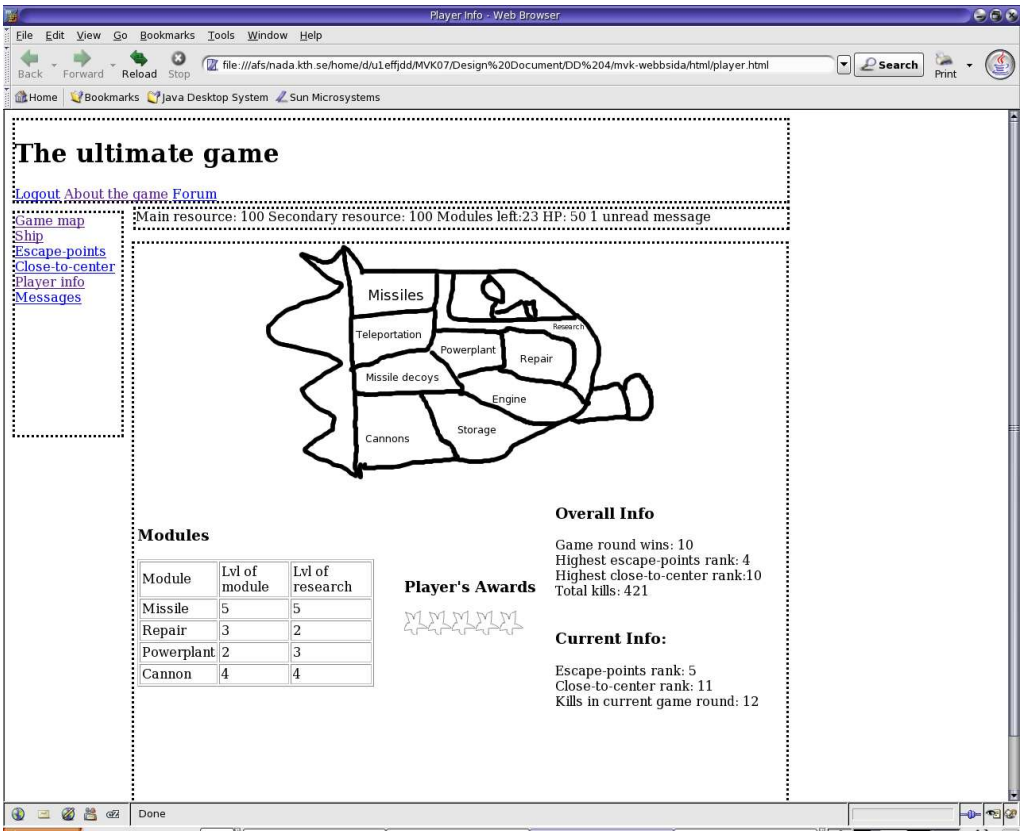
Cost to build missile decoys: Calls function getBuildMissilesDecoysCost()

Build missile decoys: Calls function buildMissilesDecoys()

(Repair module):

Auto repair ON/OFF: Calls function autoRepairONOFF()

5.6 Form 6: Player Info



1.1 Form 6: Player Info

List of references to RD:

None

The names of the controls and fields:

Information fields:

Player's Module Information

Player's Award Information

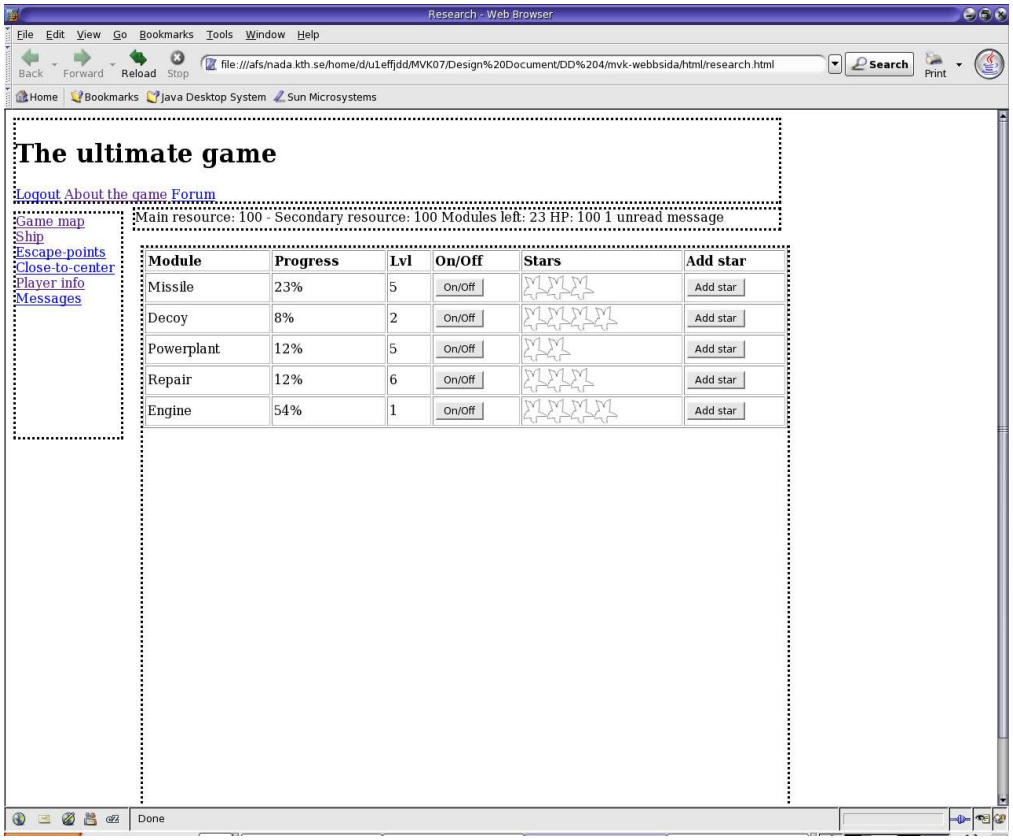
Player's Overall Information

Player's Current Information

The names of the events, methods, or procedures that cause this form to be displayed:

Clicking the playerInfo Hyperlink in form 3; “Side field, logged in”.

5.7 Form 7: Research page



Form 7: Research page

List of references to RD:

Research, section 7.1.9

The names of the controls and fields:

Information fields:

Available Stars

Research Type

Progress bar

Level box

Number of Stars

Buttons:

Research On

Research Off

Add Star

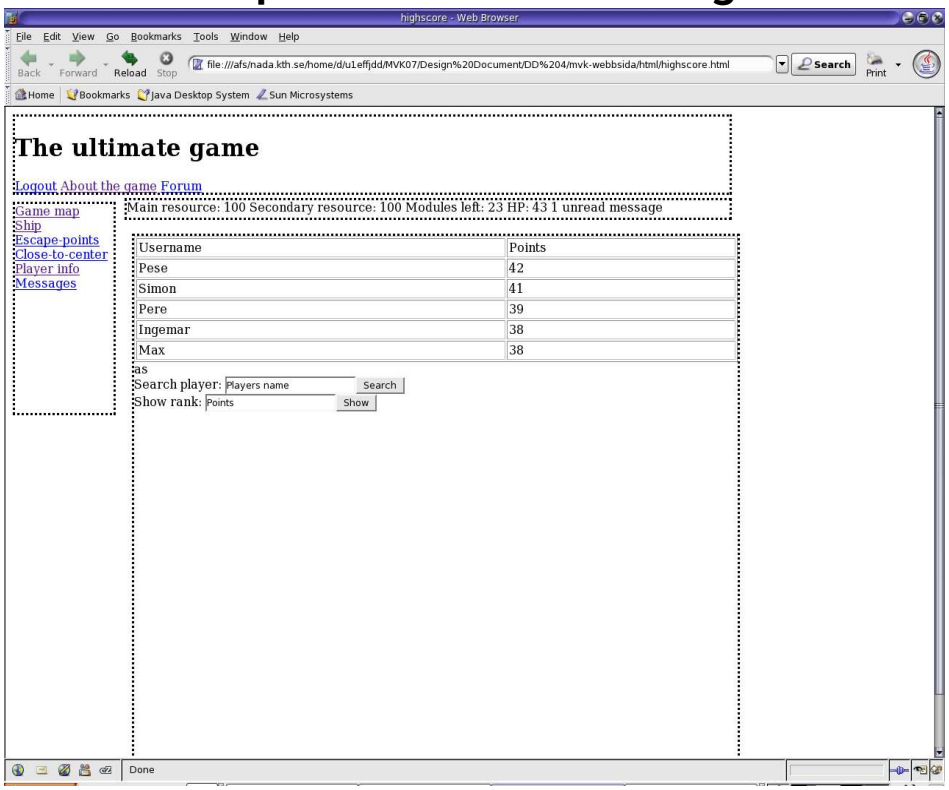
The names of the events, methods, or procedures that cause this form to be displayed:

Clicking the research Hyperlink in form 3; “Side field, logged in”.

The names of the events, methods, or procedures triggered by each control:

Available Stars	Calls function getAvailableStars()
Research Type	Calls function getResearchType()
Progress bar	Calls function getResearchProgressBar()
Level box	Calls function getResearchLevel()
Number of Stars	Calls function getNumberOfStars ()
Research On	Calls function activateResearch()
Research Off	Calls function deactivateResearch()
Add Star	Calls function addStar()

5.8 Form 8: Escape Points-/Close To- High Score List



Form 8: Escape Points-/Close To- High Score List

List of references to RD:

High-score list, section 7.1.6

We are here describing two different pages, the Escape Points High Score List and the Close To High Score List, this due to that they will be almost identical.

The names of the controls and fields:

Information field:

Display ranks

Buttons:

Search

Show

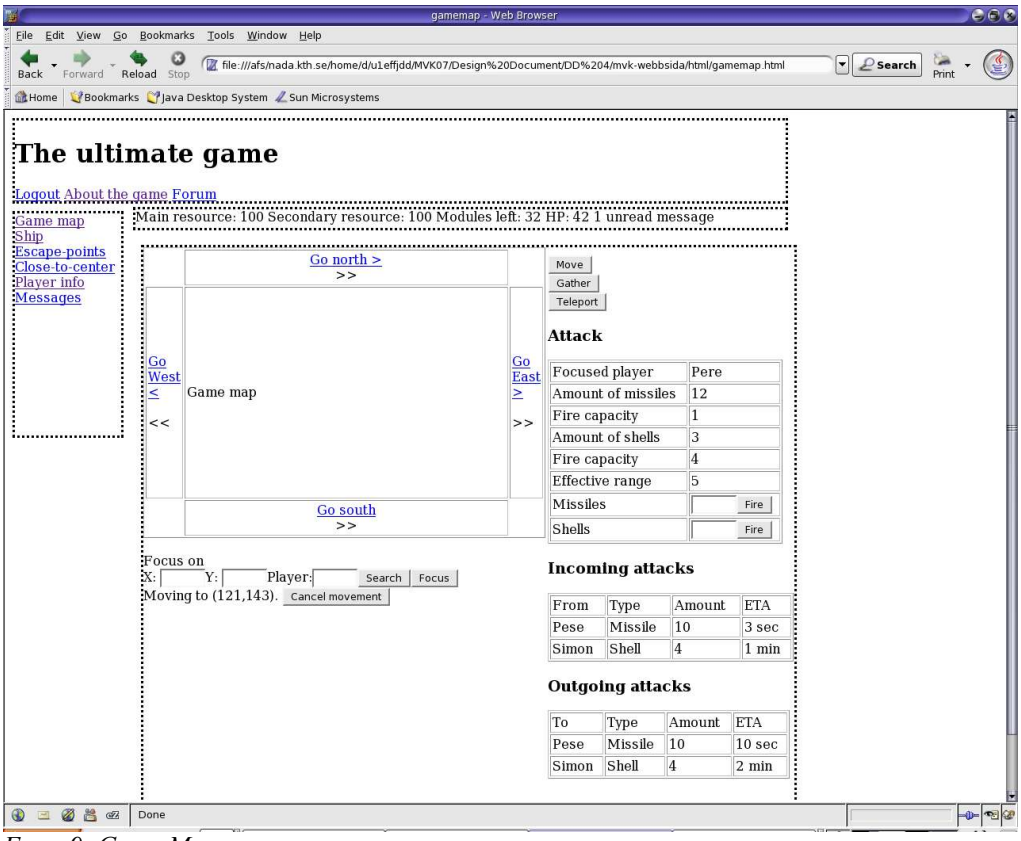
The names of the events, methods, or procedures that cause this form to be displayed:

escapePointsHighScoreList or closeToHighScoreList Hyperlink located in form 3; “Side field, logged in”.

The names of the events, methods, or procedures triggered by each control:

(Escape List)	Display ranks	Calls function getEscapeList()
(Escape List)	Search	Calls function searchPlayerByEscape()
(Escape List)	Show	Calls function showRankByEscape()
(Close to List)	Display ranks	Calls function getCloseList()
(Close to List)	Search	Calls function searchPlayerByCloseTo()
(Close to List)	Show	Calls function showRankByCloseTo()

5.9 Form 9: Game Map



Form 9: Game Map

List of references to RD:

Game Map, section 7.1.2.1

Movement of Ship, section 7.1.4.5

Launch Missiles, section 7.1.5.1.1.3

Firing Cannons, section 7.1.5.1.2.3

The names of the controls and fields:

Buttons:

Move ship

Move ship cancel

Fire missiles

Fire cannons

Move View of Map North: Short(displayed by a single arrow)

Move View of Map North: Far(displayed by a double arrow)

Move View of Map South: Short(displayed by a single arrow)

Move View of Map South: Far(displayed by a double arrow)

Move View of Map East: Short(displayed by a single arrow)

Move View of Map East: Far(displayed by a double arrow)

Move View of Map West: Short(displayed by a single arrow)

Move View of Map West: Far(displayed by a double arrow)

map Focus

map Search

Information fields:

Focused Player

Amount of Missiles

Fire CapacityMissiles

Amount of Shells

Fire CapacityShells

Effective RangeCannon

Incoming Attacks

Outgoing Attacks

Text fields:

Missiles to fire

Shells to fire

Map X coordinate

Map Y coordinate

Map focus on player

Complex Type:

Map frame

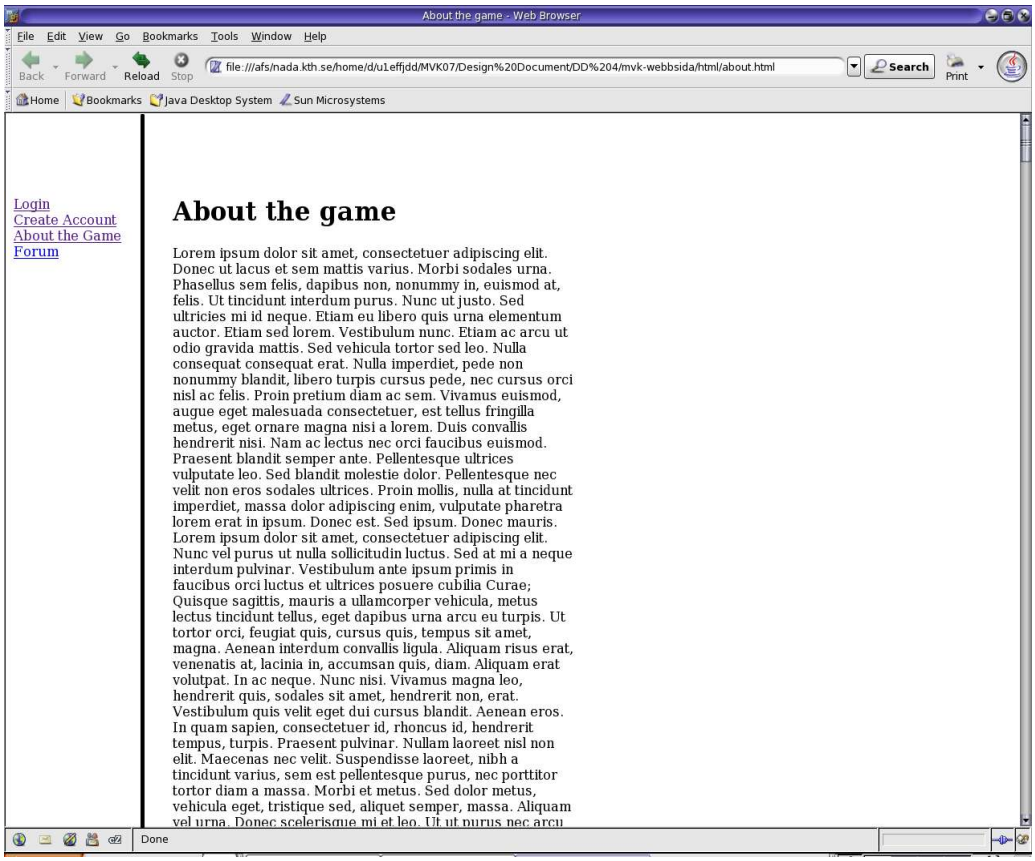
The names of the events, methods, or procedures that cause this form to be displayed:

Pushing the ”Game Map” Hyperlink in form 3; “Side field, logged in”

The names of the events, methods, or procedures triggered by each control:

Move ship	Calls function triggerMoveShip()
Move ship cancel	Calls function stopMovingShip()
Fire missiles	Calls function fireMissiles()
Fire cannons	Calls function fireShells()
Move View of Map North: Short	Calls function moveMapViewNorthShort()
Move View of Map North: Far	Calls function moveMapViewNorthFar()
Move View of Map South: Short	Calls function moveMapViewSouthShort()
Move View of Map South: Far	Calls function moveMapViewSouthFar()
Move View of Map East: Short	Calls function moveMapViewEastShort()
Move View of Map East: Far	Calls function moveMapViewEastFar()
Move View of Map West: Short	Calls function moveMapViewWestShort()
Move View of Map West: Far	Calls function moveMapViewWestFar()
map Focus	Calls function mapFocus()
map Search	Calls function mapSearch()
Focused Player	Calls function getFocusedPlayer()
Amount of Missiles	Calls function getAmountOfMissiles()
Fire CapacityMissiles	Calls function getMissileFireCapacity()
Amount of Shells	Calls function getAmountOfMissiles()
Fire CapacityShells	Calls function getShellFireCapacity()
Effective RangeCannon	Calls function getEffectiveCannonRange()
Incoming Attacks	Calls function getIncomingAttacks()
Outgoing Attacks	Calls function getOutgoingAttacks()
Map frame	if triggerMoveShip() has been called, call moveShip(), else call focusCoordinates()

5.10 Form 10: About The Game



Form 10: About The Game

List of references to RD:

Game instructions, section 7.1.1.3

The names of the controls and fields:

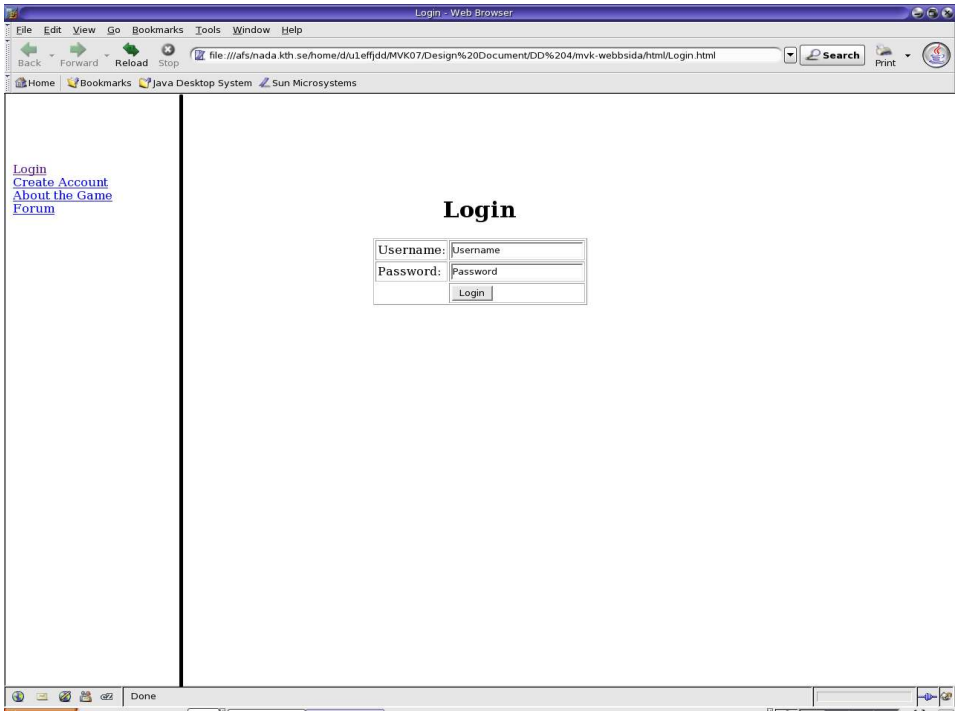
Information field:

About The Game

The names of the events, methods, or procedures that cause this form to be displayed:

Clicking the aboutTheGame Hyperlink in form 3; “Side field, logged in” or clicking on the aboutTheGame Hyperlink in form 1; “Side field, not logged in”.

5.11 Form 11: Login



Form 11: Login

List of references to RD:

Log in, section 7.1.1.2

The names of the controls and fields:

Buttons:

Login

Text fields:

Username

Password

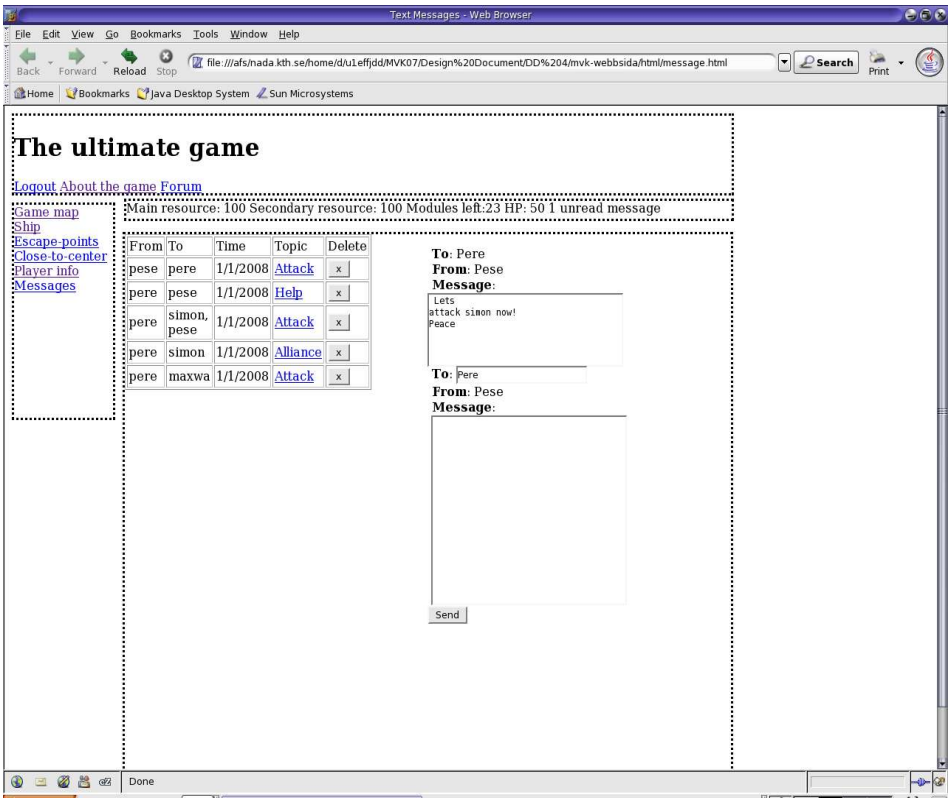
The names of the events, methods, or procedures that cause this form to be displayed:

Clicking the logIn Hyperlink in form 1; “Side field, not logged in”.

The names of the events, methods, or procedures triggered by each control:

Login Calls function logIn()

5.12 Form 12: Messages



Form 12: Messages

List of references to RD:

Messages, section 7.1.8.1

The names of the controls and fields:

Links:

Topic

Buttons:

Delete

Send

Information fields:

From

To

Time

Message

Text fields:

Send message

To

The names of the events, methods, or procedures that cause this form to be displayed:

Clicking the Messages Hyperlink in form 1; “Side field, not logged in”.

The names of the events, methods, or procedures triggered by each control:

Topic	Calls function showMessage()
Delete	Calls function deleteMessage()
Send	Calls function sendMessage()

6 Design Details

6.1.1 Class Responsibility Collaborator (CRC)cards

GameRound.logic	
Responsibilities	**Collaborators**
Create new map Create map objects AddNewPlayer Place map objects End game round	Player.logic Map.logic

Highscore.logic	
Responsibilities	**Collaborators**
Create escape point list Create close-to list	Highscore.data Player.logic

Highscore.data	
Responsibilities	**Collaborators**
Store escape point list Store close-to list	

Alliance.logic	
Responsibilities	**Collaborators**
Create new alliance Remove player from alliance Add player to alliance	Alliance.data Player.logic

Alliance.data	
Responsibilities	**Collaborators**

Represent name of alliance Represent all players of the alliance	

Map.logic	
Responsibilities	**Collaborators**
Draw map objects Create map object Validates if a square is stackable. Perform add incoming missiles. Perform add incoming shells.	Ship.logic ResourceSquare.logic Wormhole.logic Map.data

Map.data	
Responsibilities	**Collaborators**
Knows the location of all ships on the map Knows the location of all resource squares on the map Knows the location of all wormhole on the map Knows the time of impact of all incoming missiles Knows the time of impact of all incoming shells	

Ship.logic	
Responsibilities	**Collaborators**
Perform update main resource Perform update secondary resource Perform get coordinates. Perform change coordinates. Perform calculate damage Perform gather resource.	Modules.logic Ship.data

Ship.data	
Responsibilities	**Collaborators**
Represent amount of main resource Represent amount of secondary resource Knows condition status Knows its set of coordinates on the map Boolean isStackable	

ResourceSquare.logic	
Responsibilities	**Collaborators**
Subtract resource from square	ResourceSquare.data

ResourceSquare.data	
Responsibilities	**Collaborators**
Represent amount of resource Knows its set of coordinates on the map Boolean isStackable	

Wormhole.logic	
Responsibilities	**Collaborators**
Notify the system about a winner	GameRound.logic Wormhole.data

Wormhole.data	
Responsibilities	**Collaborators**
Knows its set of coordinates on the map Boolean isStackable	

Player.logic	
Responsibilities	**Collaborators**
Calculate Escape Points Be able to create a new account Be able to login Perform get all messages for a player. Perform send message. Perform remove message.	Player.data Ship.logic Alliance.logic

Player.data	
Responsibilities	**Collaborators**
Knows the player's closest distance to the wormhole Knows the highest escape point gained by the player Knows which alliance the player belongs to (if any) Knows all sent and received text messages Knows player Id Knows player user name Knows player password Knows player e-mail Knows the Id of the ship the player owns Knows all awards given to the player Knows how many kills the player has throughout the current game round Knows the overall total amount of kills the player has Knows how many game rounds the player has won	

Module.logic	
Responsibilities	**Collaborators**
Perform ship movement. Perform cannon attack. Perform missile attack. Perform build shells. Perform build missile decoys. Perform build missile. Perform toggle auto repair. Perform teleportation. Perform update resources. Perform build module. Perform calculation of all storages. Perform validate enough resources. Perform toggle research on/off for a specific module. Perform upgrade module. Perform remove module.	Module.data Ship.logic Map.logic ResourceSquare.logic

Module.data	
Responsibilities	**Collaborators**
Knows the quantity of each module Knows the level of each module Knows the build, upgrade and research cost for each module. Knows the maximum amount of storage.	

CreateAccount.presentation	
Responsibilities	**Collaborators**
Present the GUI for the create account web-page. Handle data input from the user.	Player.logic

Login.presentation	
Responsibilities	**Collaborators**
Present the GUI for the login web-page Handle data input form the user	Player.logic

TheShip.presentation	
Responsibilities	**Collaborators**
Present the GUI for the the ship web-page Handle data input form the user	

PlayerInfo.presentation	
Responsibilities	**Collaborators**
Present the GUI for the player info web-page Handle data input form the user	Player.logic Module.logic

EscapePointsHighscore.presentation	
Responsibilities	**Collaborators**
Present the GUI for the escape points highscore web-page Handle data input form the user	Player.logic

CloseToHighscoreList.presentation	
Responsibilities	**Collaborators**

Present the GUI for the high score list over the players that are closest to the wormhole, web-page Handle data input form the user	Player.logic

GameMap.presentation	
Responsibilities	**Collaborators**
Present the GUI for the game map web-page Handle data input form the user	Map.logic Module.logic Player.logic

AboutTheGame.presentation	
Responsibilities	**Collaborators**
Present the GUI for the about the game web-page Handle data input form the user	

Messages.presentation	
Responsibilities	**Collaborators**
Present the GUI for the messages web-page Handle data input form the user	Player.logic

MissileBatteryModule.presentation	
Responsibilities	**Collaborators**
Present the GUI for the missile battery module web-page Handle data input form the user	Module.logic

Teleportation.presentation	
Responsibilities	**Collaborators**
Present the GUI for the teleportation module web-page Handle data input form the user	Module.logic

MissileDecoyModule.presentation	
Responsibilities	**Collaborators**
Present the GUI for the missile decoy module web-page Handle data input form the user	Module.logic

Cannons.presentation	
Responsibilities	**Collaborators**
Present the GUI for the cannons module web-page Handle data input form the user	Module.logic

StorageModule.presentation	
Responsibilities	**Collaborators**
Present the GUI for the storage module web-page Handle data input form the user	Module.logic

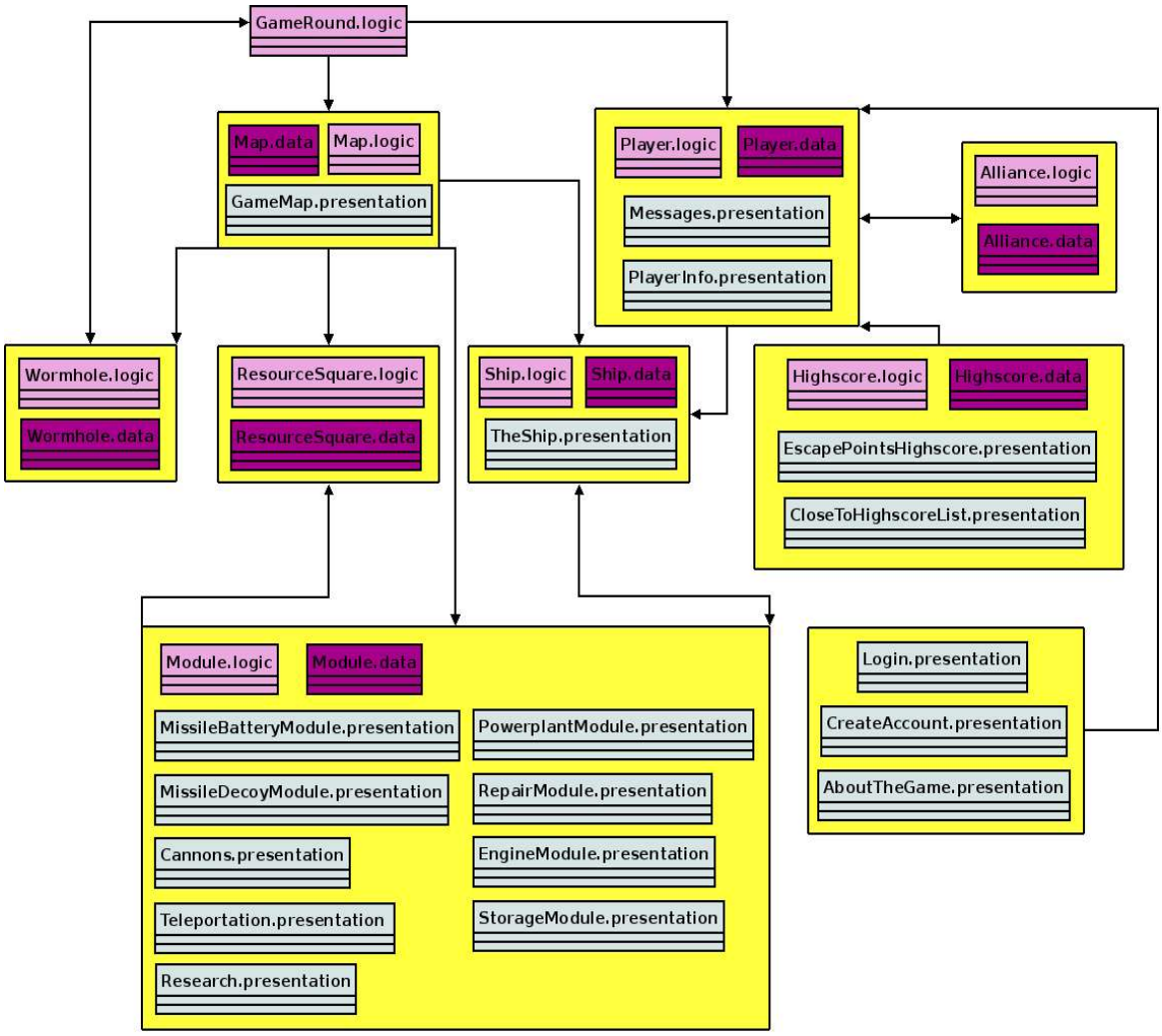
EngineModule.presentation	
Responsibilities	**Collaborators**
Present the GUI for the engines module web-page Handle data input form the user	Module.logic

PowerplantModule.presentation	
Responsibilities	**Collaborators**
Present the GUI for the login web-page Handle data input form the user	Module.logic

RepairModule.presentation	
Responsibilities	**Collaborators**
Present the GUI for the repair module web-page Handle data input form the user	Module.logic

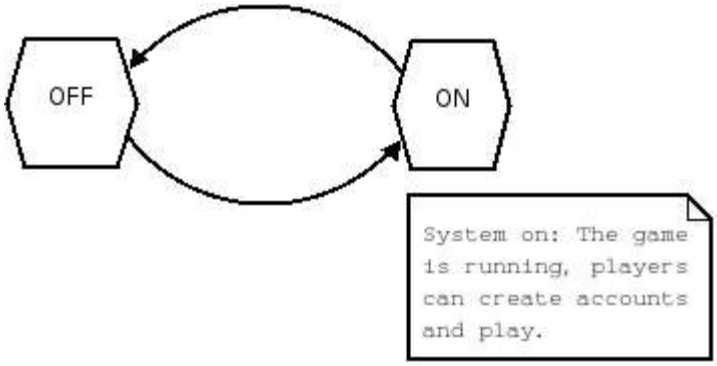
Research.presentation	
Responsibilities	**Collaborators**
Present the GUI for the research web-page Handle data input form the user	Module.logic

6.2 Class Diagram



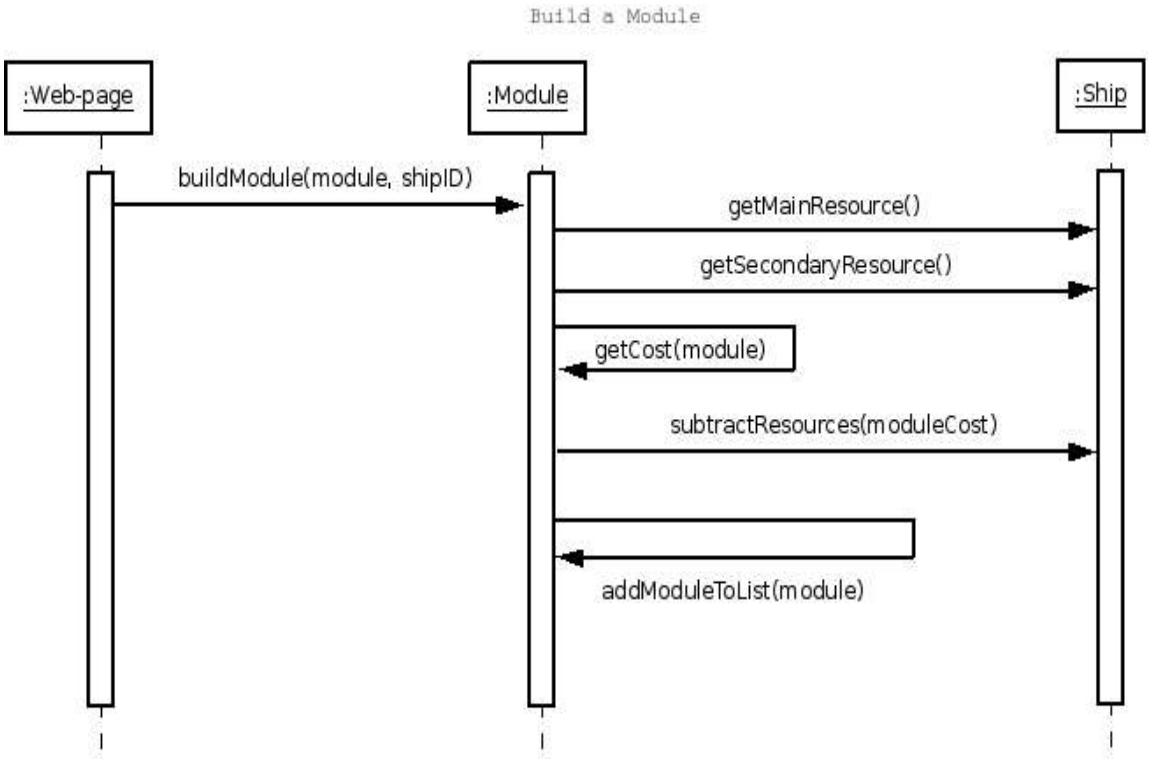
6.3 State Charts

The states of the system are very simple. Once the system is turned on there are no specific states it can enter.

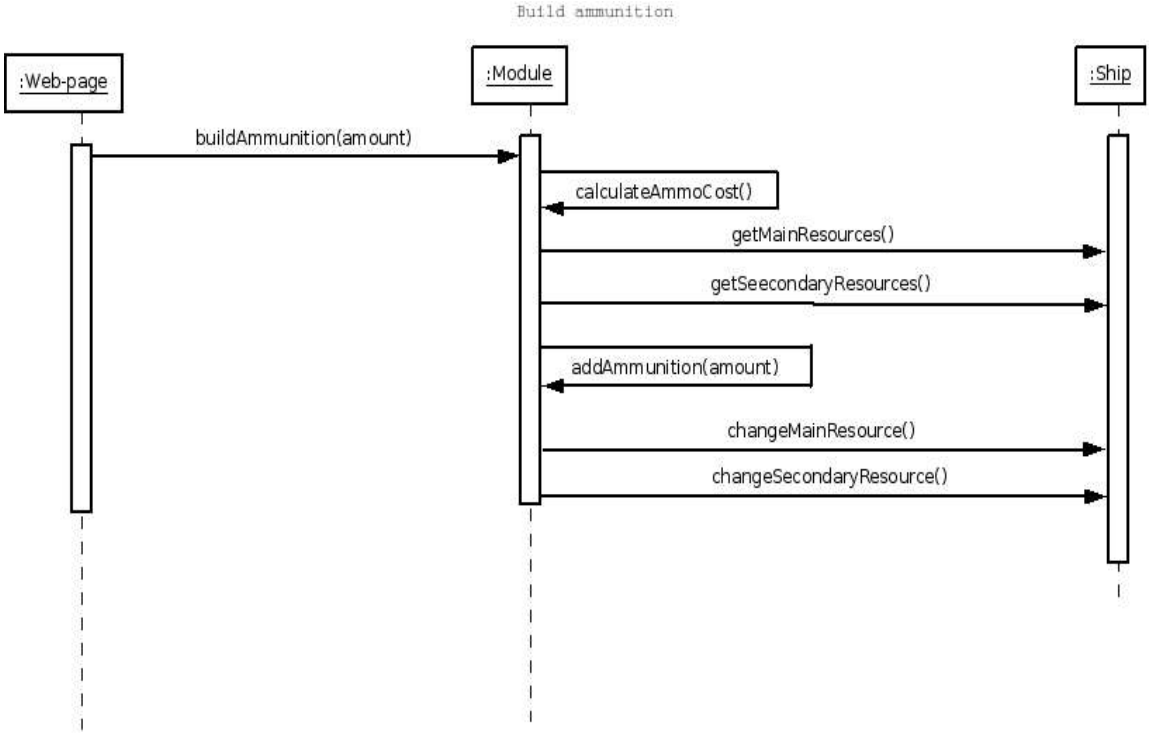


6.4 Interaction Diagrams

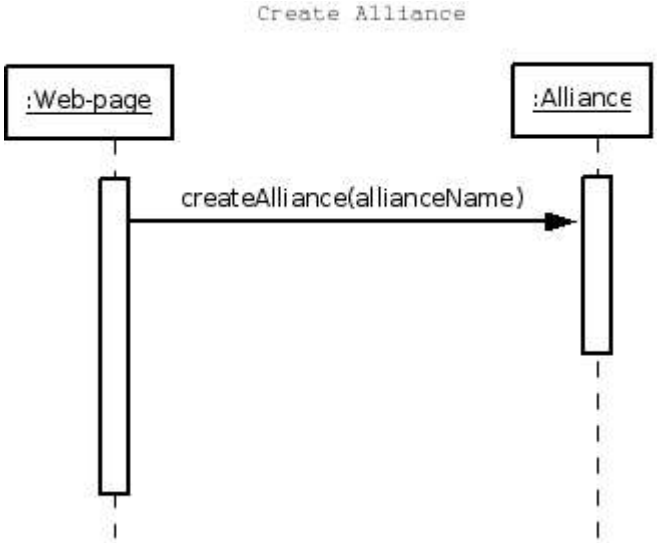
6.4.1 Build a module



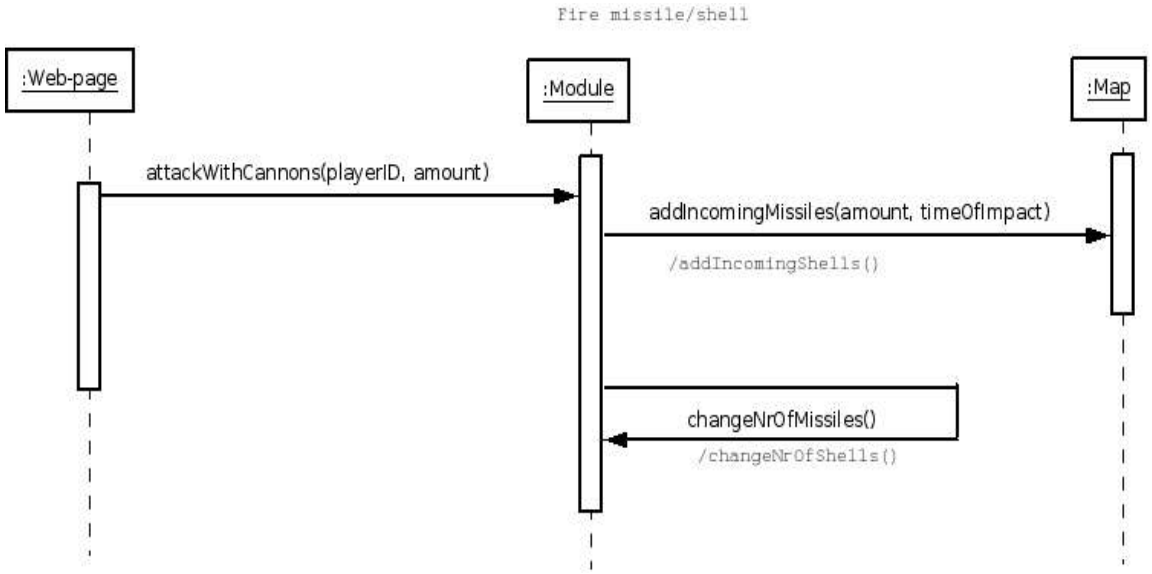
6.4.2 Build ammunition



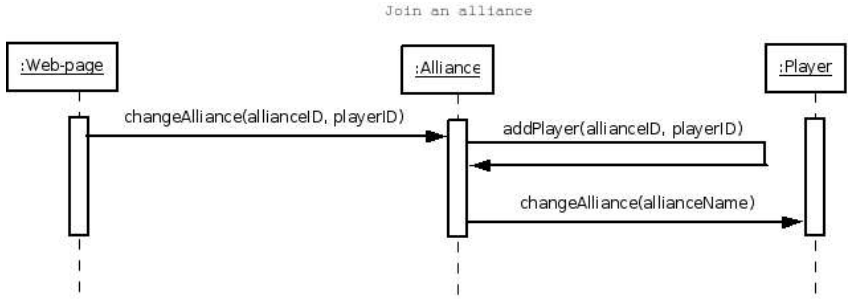
6.4.3 Create Alliance



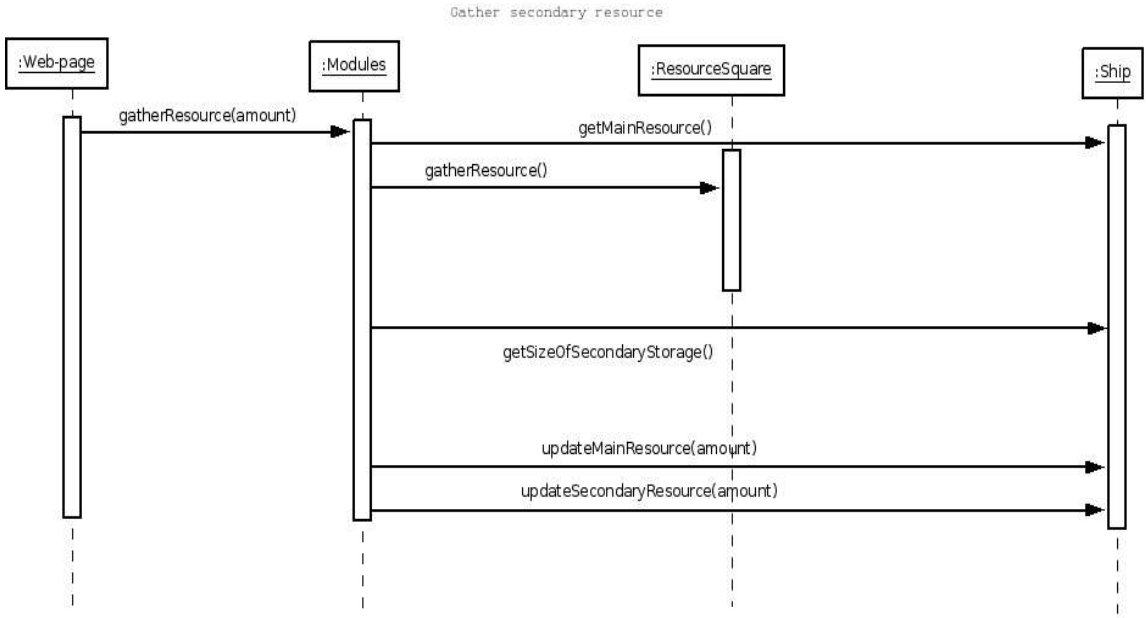
6.4.4 Fire missile/shell



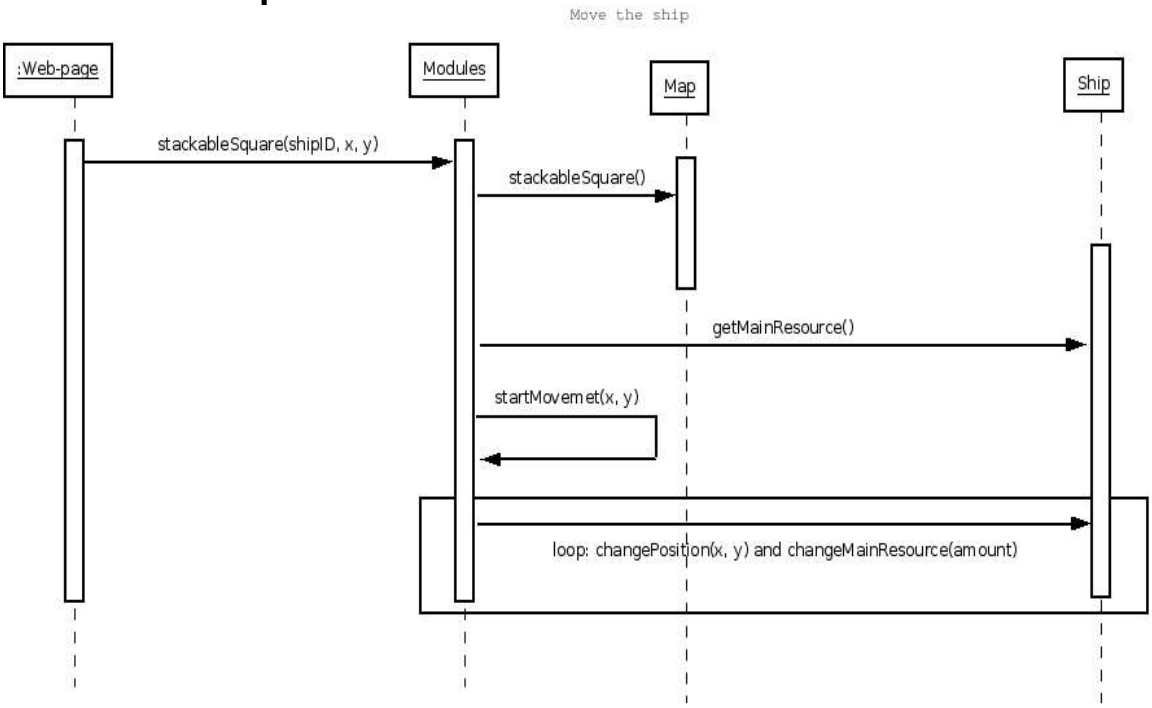
6.4.5 Join an alliance



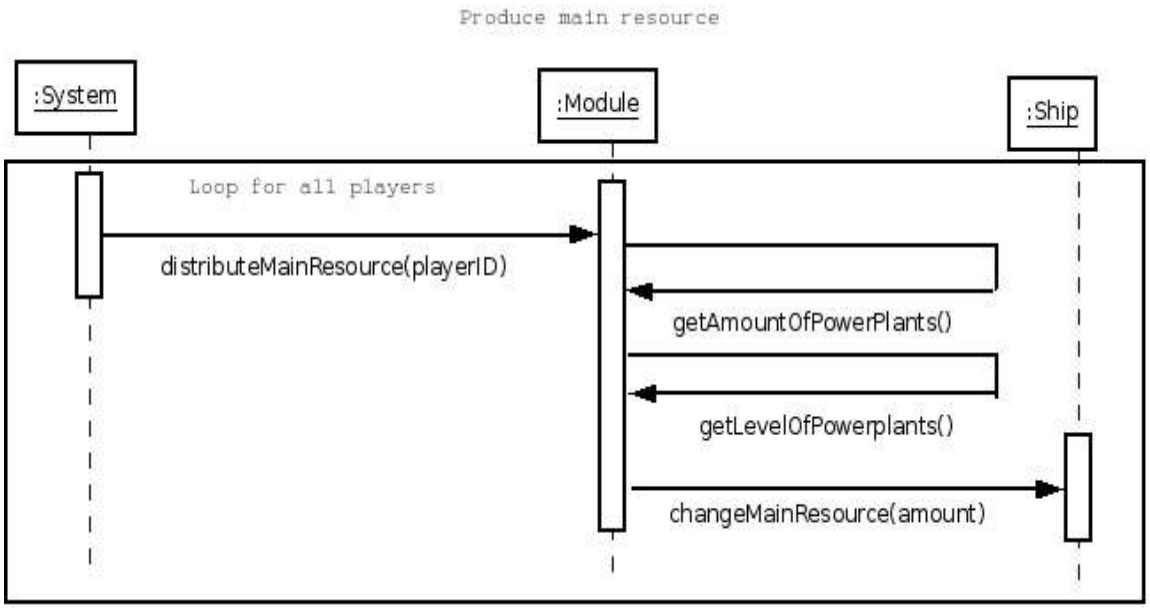
6.4.6 Gather secondary resource



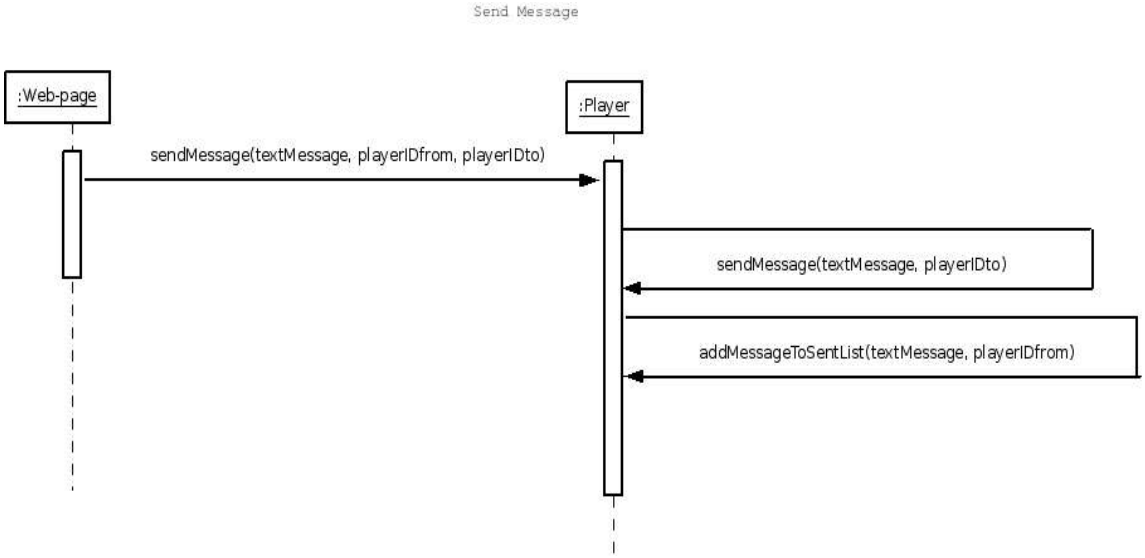
6.4.7 Move ship



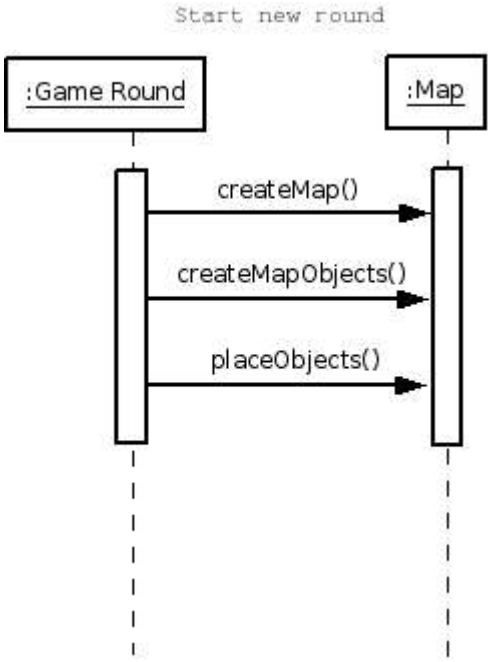
6.4.8 Produce main resource



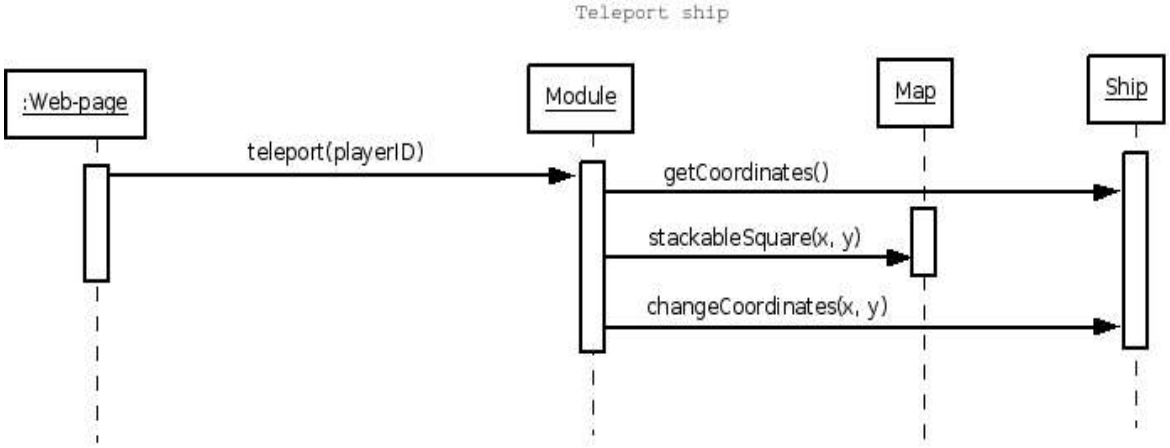
6.4.9 Send message



6.4.10 Start new round



6.4.11 Teleport ship



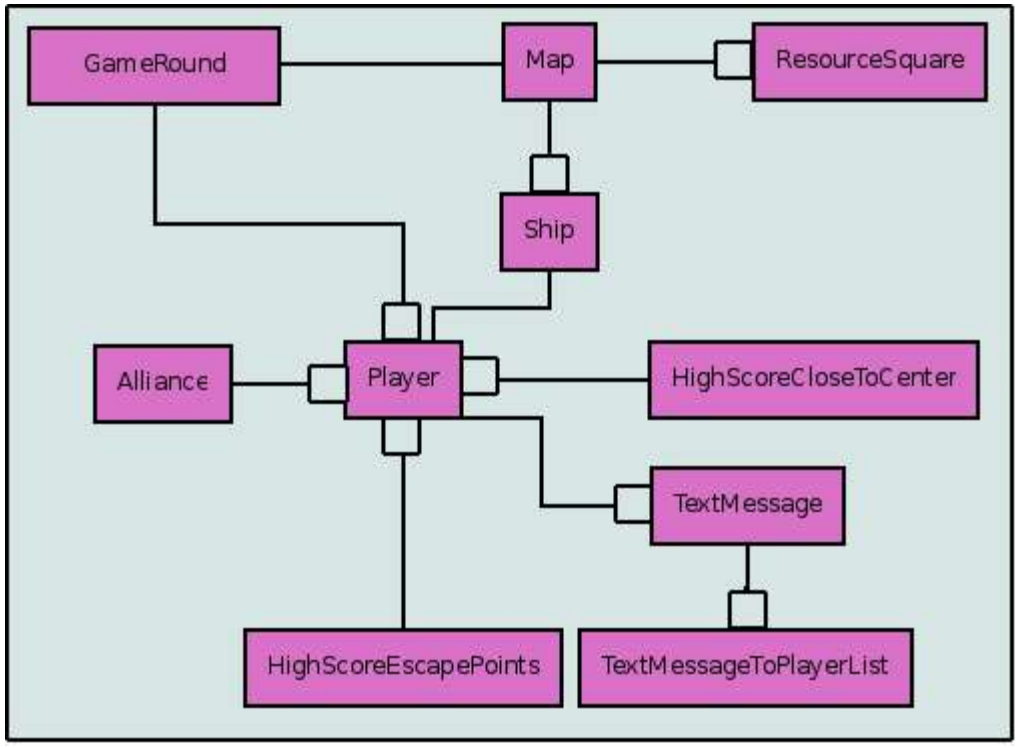
6.4.12 Upgrade module

6.5 Detailed Design

Table of Contents

6.5.1 Database

Structure:



Tabels:

GameRound
- RoundID
- MapID

TextMessage
- PlayerID(from)
- Date
- Subject
- Message

Player
- PlayerID
- PlayerName
- PlayerPass
- PlayerEmail
- ShipID
- AllianceID
- RoundID
- BestEscapePoint
- Kills
- Wins

Ship
- ShipID
- ModuleMatrix
- Coordinates
- MainResource
- SecondaryResource
- ConditionStatus
- MapID

ResourceSquare
- Coordinates
- Amount
- MapID

Map
- MapID
- SizeOfMap
- WormholeCoordinates
- AmountOfPlayers

TextMessageToPlayerList
- TextID
- PlayerID (to)

HighScoreEscapePoints
- Rank
- PlayerID
- Points

HighScoreCloseToCenter
- Rank
- PlayerID
- Distance

6.5.2 General Detailed Design information:

To ensure that the variables in our project are capsuled in a consistent and clear way, each logic-class will only be able to get data from it's corresponding data-class. This means that if A.logic needs data from B.data, it has to go through B.logic to get that information.

Due to this, every logic and data-class will have “getters” and “setters” for all the variables stored in the data-class. A “getter” is a method that picks a variable up from a data-class and returns it, and in the same way a “setter” is a method that changes the value of a variable in a data-class.

The presentation layer in our project consists of several presentation-classes. The purpose of these presentation-classes is to generate HTML code for the corresponding page which is to be sent to the user. These classes will only contain one method each, buildPage, which consults the logic-classes if needed, in order to construct the HTML code and present the page for the user.

With the explanations of “getters”, “setters” and presentation-classes above they will not be described further in this section. The reason for this is that it'd be very redundant to explain the same thing over and over.

6.5.2.1 Highscore.data

Fields

Attribute: escapePointList
Type: String[][]
Usage: This is used to keep track of the escape point highscore list. The matrix shall be sorted by points. It shall be filled with playerID's and points.

Attribute: closeToList
Type: String[][]
Usage: This is used to keep track of the close to center highscore list. The matrix shall be sorted by points. It shall be filled with playerID's and points.

6.5.2.2 Alliance.data

Fields

Attribute: allianceName
Type: String
Usage: Every alliance has a unique name.

Attribute: allianeceMembers
Type: String[]

Usage: This is used to keep track of all members in an alliance. The array shall be filled with playerID's.

6.5.2.3 Map.data

Fields

Attribute: allShipsLocation
Type: String[][]
Usage: This is used to know the location of all ships on the map. The matrix shall be filled with shipID's and coordinates.

Attribute: allResourceSquaresLocation
Type: int[][]
Usage: This is used to know the location of all resource squares on the map. The matrix shall be filled with x and y coordinates.

Attribute: wormholeLocation
Type: int[]
Usage: This is used to know the location of the wormhole on the map. The array will have two elements, x and y coordinates for the wormhole.

Attribute: incomingMissiles
Type: String[][]
Usage: This is used to keep track of all incoming missiles and when they will reach their targets. The matrix shall be filled with playerID's (both attacker and receiver), amount of missiles, time until impact and accuracy.

Attribute: incomingShells
Type: String[][]
Usage: This is used to keep track of all incoming shells and when they will reach their targets. The matrix shall be filled with playerID's (both attacker and receiver), amount of shells, time until impact and damage.

6.5.2.4 Ship.data

Fields

Attribute: mainResource
Type: int
Usage: Every user has an amount of the main resource at his/her disposal.

Attribute: secondaryResource
Type: int
Usage: Every user has an amount of main resource at his/her disposal.

Attribute: conditionStatus
Type: int
Usage: This is used to keep track of the ships condition status.

Attribute: mapCoordinates
Type: float[]
Usage: This is used to keep track of the ships location. The array will have two elements, x and y coordinates for the ship.

6.5.2.5 Resource.data

Fields

Attribute: amountOfResource
Type: int
Usage: This is used to keep track of the amount of resources existing in the resource square.

Attribute: mapCoordinates
Type: int[]
Usage: This is used to know the resource squares location on the map. The array will have two elements, x and y coordinates for the resource square.

6.5.2.6 Wormhole.data

Fields

Attribute: mapCoordinates
Type: int[]
Usage: This is used to know the wromholes location on the map. The array will have two elements, x and y coordinates for the wormhole.

6.5.2.7 Player.data

Fields

Attribute: playerID

Type: int
Usage: Every player has an unique player id which is used to get and set information about the player, the player's ship etc.

Attribute: userName
Type: String
Usage: This is used for a player to log in.

Attribute: password
Type: String
Usage: This is used for a player to log in.

Attribute: email
Type: String
Usage: This is used for the ability to send information about the game to the players.

Attribute: shipID
Type: int
Usage: This is used to connect a player to his/her personal ship.

Attribute: playerAwards
Type: String[]
Usage: This is used to keep track of all awards a player has obtained throughout all game rounds.

Attribute: nrKillsCurrentGameRound
Type: int
Usage: This is used to keep track of how many kills a player has scored during the current game round.

Attribute: nrKillsAllGameRounds
Type: int
Usage: This is used to keep track of how many kills a player has scored throughout all game rounds.

Attribute: gameRoundsWon
Type: int
Usage: This is used to keep track of how many game rounds a player has won.

Attribute: closestDistance
Type: float
Usage: This is used to keep track of how close a player has come to the wormhole

throughout all game rounds

Attribute: highestEscapePoints
Type: int
Usage: This is used to keep track of the highest escape point a player has obtained throughout all game rounds.

Attribute: alliance
Type: String
Usage: This is used to keep track of which alliance a player belongs to.

Attribute: sentMessages
Type: String[][]
Usage: This is used to keep track of all messages that a player has sent to other players. The matrix shall be filled with playerID's and text messages.

Attribute: recivedMessages
Type: String[][]
Usage: This is used to keep track of all messages that a player has received from other players. The matrix shall be filled with playerID's and text messages.

6.5.2.8 Module.data

Fields

Attribute: moduleInformation
Type: int[][]
Usage: This is used to keep track of all module information for a player, including how many modules of each type the player has, what level they are, how much they can store and any other eventually needed information.

Attribute: buildCost
Type: int[][]
Usage: This is used to keep track of the build cost for all different kinds of modules.

Attribute: upgradeCost
Type: int[][]
Usage: This is used to keep track of the upgrade cost for all different kinds of modules.

Attribute: researchCost
Type: int[][]
Usage: This is used to keep track of the research cost for all different kinds of modules.

6.5.2.9 ResourceSquare.logic

6.5.2.9.1subtractResources

Parameters: int ResourcesToSubtract
Return value: boolean resourceSubtracted
Description: the function subtracts the drawn resources from the total resource of the square and returns true. If it is not enough resources to withdraw it returns false.
Database: select the old value of ResourceSquare and update with the new value.
Pre-condition: none
Validity checks: enough resources have to exist in the resource square.
Post-condition: the function will return whether or not the resources was drawn from the square.
Calls: "getters" and "setters"
Called by: the presentation layer
RD: Functional Requirement 7.1.3.3 Resource squares, Use Case 8.3.2 Gather resources

6.5.2.10 Module.logic

6.5.2.10.1 moveShip

Parameters: int endX, int endY
Return value: boolean startMoving
Description: moves the ship from the current position to the coordinates given as parameters and returns true. If it is not possible to move the ship to the given coordinates, return false.
Database: update the coordinates of Ship
Pre-condition: none
Validity checks: Check if it is enough resources to move the ship and that the square the ship is moving to is not occupied buy something
Post-condition: the function will return whether or not the movement has started.
Calls: "getters" and "setters", updateResources, validateEnoughResources
Called by: the presentation layer
RD: Functional Requirement 7.1.4.5 Movement of ship, Use Case 8.4.1 Move ship

6.5.2.10.2 cannonAttack

Parameters: Ship attackShip
Return value: boolean attackStarted
Description: starts the attack on the ship and returns true. If it is not possible to attack the ship return false. If the attack is possible draw ammunition from the ship.
Database: update the ammunition shells of the ship.

Pre-condition: none
Validity checks: check if it is enough resources to perform the attack.
Post-condition: the function will return whether or not the attack was started.
Calls: “getters” and “setters”,
Called by: the presentation layer
RD: Functional Requirement 7.1.5.1.2 Cannons module, Use Case 8.5.3 Fire shells

6.5.2.10.3 missileAttack

Parameters: Ship attackShip
Return value: boolean attackStarted
Description: starts the attack on the ship and returns true. If it is not possible to attack the ship return false. If the attack is possible draw ammunition from the ship.
Database: update the ammunition missiles of the Ship.
Pre-condition: none
Validity checks: check if it is enough resources to perform the attack.
Post-condition: the function will return whether or not the resources was drawn from the square.
Calls: “getters” and “setters”
Called by: the presentation layer
RD: Functional Requirement 7.1.5.1.1 Missile module, Use Case 8.5.2 Fire missiles

6.5.2.10.4 buildShells

Parameters: int numberOfShells
Return value: boolean shellsBuilt
Description: If it is not possible to build a new shell it will return true. If is not possible to build the shell it will return false.
Database: update the ammunition of shells of the Ship.
Pre-condition: none
Validity checks: -
Post-condition: the function will return whether or not the shells was built.
Calls: “getters” and “setters”, validateEnoughResources, updateResources
Called by: the presentation layer
RD: Functional Requirement 7.1.5.1.2 Cannons module, Use Case 8.5.1 Build ammunition (missiles/shells)

6.5.2.10.5 buildMissileDecoys

Parameters: int numberOfDecoys
Return value: Boolean decoysBuilt
Description: If it is not possible to build a new missile decoys it will return true. If is not

possible to build the missile decoys it will return false.
Database: update the ammunition of missile decoys of the Ship.
Pre-condition: none
Validity checks: -
Post-condition: the function will return whether or not the missile decoys was built.
Calls: “getters” and “setters”, validateEnoughResources, updateResources
Called by: the presentation layer
RD: Functional Requirement 7.1.5.2.2 Missile decoys module, Use Case 8.5.1 Build ammunition (missiles/shells)

6.5.2.10.6 buildMissile

Parameters: int numberOfMissiles
Return value: Boolean MissileBuilt
Description: If it is not possible to build a new missiles it will return true. If is not possible to build the missiles it will return false.
Database: update the ammunition of missiles of the ship.
Pre-condition: none
Validity checks: -
Post-condition: the function will return whether or not the missiles was built.
Calls: “getters” and “setters”, validateEnoughResources, updateResources
Called by: the presentation layer
RD: Functional Requirement 7.1.5.1.1 Missile module, Use Case 8.5.1 Build ammunition (missiles/shells)

6.5.2.10.7 toggleAutoRepair

Parameters: -
Return value: Boolean toggleTo
Description: start and stop repairing the ship returns what the research is toggled to.
Database: update ModuleMatrix in Ship
Pre-condition: none
Validity checks: enough resources
Post-condition: the function will return the state that auto repair has been toggled to.
Calls: “getters” and “setters”, validateEnoughResources, updateResources
Called by: the presentation layer
RD: Functional Requirement 7.1.4.4 Repair ship, Use Case 8.4.3 Repair the ship

6.5.2.10.8 teleport

Parameters: int distance

Return value: Boolean teleported
Description: If the ship has enough resources the teleportation will be done depending on the distance.
Database: update position of the Ship
Pre-condition: none
Validity checks: enough resources
Post-condition: the function will return if or if not the teleportation has been done.
Calls: "getters" and "setters", validateEnoughResources, updateResource
Called by: the presentation layer
RD: Functional Requirement 7.1.5.2.1 Teleportation module, Use Case 8.5.5 Teleport the ship

6.5.2.10.9 updateResources

Parameters: int resource
Return value: -
Description: Update the primary resources for the ship.
Database: update resources for ship
Pre-condition: none
Validity checks: -
Post-condition: new value for the resources of the ship
Calls: "getters" and "setters"
Called by: the presentation layer
RD: -

6.5.2.10.10 buildModule

Parameters: int type
Return value: boolean built
Description: If the ship has enough resources the module will be built. In other case, the method will return false.
Database: update ModuleMatrix in Ship
Pre-condition: none
Validity checks: enough resources
Post-condition: the function will return if or if not the module has been built.
Calls: "getters" and "setters", validateEnoughResources, updateResources
Called by: the presentation layer
RD: Functional Requirement 7.1.5 Modules, Use Case 8.4.5 Build a module

6.5.2.10.11 calculateAllStorages

Parameters: -

Return value: int totalStorage
Description: Return the total storage for the primary and secondary resources.
Database: select MainResource and SecondaryResource
Pre-condition: none
Validity checks: -
Post-condition: the function will return
Calls: "getters" and "setters", validateEnoughResources, updateResources
Called by: the presentation layer
RD: Functional Requirement 7.1.5.3 Storage module

6.5.2.10.12 validateEnoughResources

Parameters: int resource
Return value: Boolean enough
Description: Validate if the given number of resources is enough, then return true. Otherwise, false.
Database: select MainResource and SecondaryResource
Pre-condition: none
Validity checks: enough resources
Post-condition: the function will return if or if not it is enough resources.
Calls: "getters" and "setters",
Called by: the presentation layer
RD: -

6.5.2.10.13 toggleResearch

Parameters: int type
Return value: Boolean toggleTo
Description: start and stop research for a module and returns what the research is toggled to.
Database: update ModuleMatrix in Ship
Pre-condition: none
Validity checks: -
Post-condition: the function will return what the module has been toggled to.
Calls: "getters" and "setters", validateEnoughResources, updateResources
Called by: the presentation layer
RD: Functional Requirements 7.1.9 Research, Use Case 8.9.1 Research a research field

6.5.2.10.14 upgradeModule

Parameters: int type
Return value: boolean upgraded

Description: If the ship has enough resources the module will be upgraded. In other case, the method will return false.
Database: update ModuleMatrix in Ship
Pre-condition: none
Validity checks: enough resources
Post-condition: the function will return if or if not the module has been built.
Calls: "getters" and "setters", validateEnoughResources, updateResources
Called by: the presentation layer
RD: Functional Requirement 7.1.5 Modules, Use Case 8.5.6 Upgrade a module

6.5.2.10.15 removeModule

Parameters: int type
Return value: -
Description: Remove the module from the ship.
Database: - update ModuleMatrix in Ship
Pre-condition: none
Validity checks: -
Post-condition: -
Calls: "getters" and "setters"
Called by: the presentation layer
RD: Functional Requirement 7.1.5 Modules, -

6.5.2.11 Player.logic

6.5.2.11.1 calculateEscapePoints

Parameters: int playerID
Return value: int amountOfEscapePoints
Description: the function calculates the amount of escape points that the user has based on his/her modules on the ship and his/her level of research on each module
Database: returns all information regarding the modules on the player's ship
Pre-condition: none
Validity checks: the amount of escape points cannot be negative
Post-condition: the function will return the amount of escape points that the user has
Calls: "getters" and "setters"
Called by: the presentation layer, createEscapePointList()
RD: Use Case 8.6.1 Calculate Escape Points

6.5.2.11.2 createNewAccount

Parameters: String userName, String password, String email

Return value: boolean accountCreated
Description: the function validates if an account with the desired user name already exists. If not, a new account gets created given the desired user name, password and email
Database: selects all players given the desired user name. If there are no results from this database query, the function stores the player account in the database
Pre-condition: none
Validity checks: user name and password must be between 3-12 characters. Email must contain an '@' and '.' character.
Post-condition: the function will return whether or not the account was created
Calls: "getters" and "setters"
Called by: the presentation layer
RD: Functional Requirement 7.1.1.1 Create An Account, Use Case 8.1.1 Create An Account

6.5.2.11.3 login

Parameters: String userName, String password
Return value: boolean login
Description: the function validates if the given password corresponds to the password associated with the given user name in the database.
Database: selects the user name in the database where the user name and password corresponds to the given parameters
Pre-condition: none
Validity checks: none
Post-condition: the function will return whether or not the login was successful
Calls: "getters" and "setters"
Called by: the presentation layer
RD: Functional Requirement 7.1.1.2 Login, Use Case 8.1.2 Login to an account

6.5.2.11.4 sendMessage

Parameters: int fromPlayerID, String toPlayer, String subject, String message
Return value: boolean messageSent
Description: the function looks up if a player with a user name called toPlayer exists. If it does it sends the given message with the given subject to the given player and returns true.
Database: selects the playerID for the String toPlayer, if there are no database results from this query the function stores the message as sent from the fromPlayerID in the database and the message as sent to playerID for toPlayer.
Pre-condition: none
Validity checks: none
Post-condition: returns whether or not the message could be sent
Calls: "getters" and "setters"
Called by: the presentation layer

RD: Functional Requirement 7.1.8.1 Text messages, Use Case 8.8.1 Sending a short text message

6.5.2.11.5 removeMessage

Parameters: int messageID, boolean isSent (if true, the message has been sent by the player)
Return value: none
Description: the function removes the message with the messageID
Database: removes the selected message
Pre-condition: none
Validity checks: none
Post-condition: removes the selected message
Calls: "getters" and "setters"
Called by: the presentation layer

6.5.2.12 GameRound.logic

6.5.2.12.1 createNewMap

Parameters: int amountOfPlayers
Return value: none
Description: the function creates the game map in regard to the amount of players.
Database: stores all information regarding the created map and its content in the database
Pre-condition: none
Validity checks: none
Post-condition: creates a game map based on the amount of players
Calls: "setters"
Called by: the presentation layer
RD: Use case 8.2.3 Create a game map

6.5.2.12.2 createAndPlaceMapObjects

Parameters: int mapID
Return value: none
Description: the function extracts the size of the map and the amount of players on the map from the database and creates and places all objects on the map
Database: returns the amount of players on the map and the size of the map, then stores information regarding all content on the game map.
Pre-condition: a map has been created
Validity checks: none
Post-condition: creates map objects and places them on the map

Calls: “getters” and “setters”
Called by: the presentation layer

6.5.2.12.3 addNewPlayer

Parameters: int playerID
Return value: none
Description: the function adds a new player to the current game round and places his/her ship on the map.
Database: stores that the new player has been added to the map and the location of his/her ship
Pre-condition: a game round exists
Validity checks: none
Post-condition: adds a new player to the game round
Calls: “getters” and “setters”
Called by: the presentation layer

6.5.2.12.4 endGameRound

Parameters: int roundID
Return value: none
Description: ends the current game round
Database: deletes all information that won't be needed in future game rounds
Pre-condition: there is a winner to the game round
Validity checks: none
Post-condition: ends current game round
Calls: none
Called by: the presentation layer

6.5.2.13 Highscore.logic

6.5.2.13.1 createEscapePointsList

Parameters: none
Return value: none
Description: the function calculates and updates the Escape points highscore list in the database.
Database: This function completely recalculates the entire HighScoreEscapePoints table in the database.
Pre-condition: none
Validity checks: The list must represent the players in the game exactly.
Post-condition: none

Calls: “getters” and “setters” in Highscore.data
Called by: the presentation layer
Requirements document: Functional requirements 7.1.6, Use case 8.6.1

6.5.2.13.2 createCloseToList

Parameters: none
Return value: none
Description: the function calculates and updates the Close to highscore list in the database.
Database: This function completely recalculates the entire HighScoreCloseToCenter table in the database.
Pre-condition: none
Validity checks: The list must represent the players in the game exactly.
Post-condition: none
Calls: “getters” and “setters” in Highscore.data
Called by: the presentation layer
Requirements document: Functional requirements 7.1.6

6.5.2.14 Wormhole.logic

6.5.2.14.1 notifyWin

Parameters: none
Return value: boolean hasWon, Player winningPlayer
Description: the function is run to control of anyone has won the game. This happends if anyone is inside the wormhole square.
Database: Controls if the wormhole square is occupied.
Pre-condition: none
Validity checks: none
Post-condition: the function will return whether or not the wormhole square was occupied.
Calls: “getters” and “setters” in Wormhole.data
Called by: the presentation layer
Requirements document: Functional requirements 7.1.2.2

6.5.2.15 Map.logic

6.5.2.15.1 returnMapObjects

Parameters: int x, int y
Return value: float[][] mapObjects
Description: the function takes the posistion on the map that shall be displayed and returns all the objects in that area and their coordinates. For example, if the function returns one

ship then it will return a one column and 3 rows matrix with type, x and y as values.
Database: This function gets all values from ResourceSquare and Ship tables.
Pre-condition: The map must have been created.
Validity checks: The returned objects must be correct in terms of type and location.
Post-condition: enough information is returned to create a map pane.
Calls: "getters" and "setters" in Map.data
Called by: the presentation layer
Requirements document: Functional requirements 7.1.2.1

6.5.2.15.2 addIncomingMissiles

Parameters: int amount, int level, Date timeToArrival
Return value: none
Description: the function uses insert sorting to place all attacks in the game in a array sorted on timeToArrival. A thread is used to handle this array.
Database: none
Pre-condition: none
Validity checks: none
Post-condition: the missileAttack is placed in the array
Calls: none
Called by: the presentation layer
Requirements document: Functional requirements 7.1.5.1.1, Use case 8.5.2, 8.5.6

6.5.2.15.3 addIncomingShells

Parameters: int amount, int level, Date timeToArrival
Return value: none
Description: the function uses insert sorting to place all attacks in the game in a array sorted on timeToArrival. A thread is used to handle this array.
Database: none
Pre-condition: none
Validity checks: none
Post-condition: the shellAttack is placed in the array
Calls: none
Called by: the presentation layer
Requirements document: Functional requirements 7.1.5.1.2, Use case 8.5.3, 8.5.7

6.5.2.16 Alliance.logic

6.5.2.16.1 Create new alliance

Parameters: String nameOfAlliance
Return value: boolean allianceCreated

Description: the function takes the wanted name of the
Database: searches for the wanted name of the alliance, and adds the new alliance if it don't already exist
Pre-condition: none
Validity checks: name of the alliance must be at least 3 characters long.
Post-condition: the function will return whether or not the alliance was created
Calls: "getters" and "setters"
Called by: The presentation layer
RD: Functional Requirement 7.1.7, Use Case 8.7.1

6.5.2.16.2 Remove player from alliance

Parameters: String playerName
Return value: boolean playerRemoved
Description: Takes the name of the player that should be removed, and removes him from the alliance
Database: Searches for the name of the player, and removes him from the alliance
Pre-condition: none
Validity checks: none
Post-condition: the function will return whether or not the player was deleted
Calls: "getters" and "setters"
Called by: The presentation layer
RD: Functional Requirement 7.1.7, Use Case 8.7.5

6.5.2.16.3 Add player to alliance

Parameters: String nameOfPlayer
Return value: boolean playerAdded
Description: The function adds a not already existing player to the given alliance
Database: Searches for the players name in the alliance, and if it not already exists, adds him to the alliance
Pre-condition: none
Validity checks: the player is not in more alliances
Post-condition: the function returns whether or not the player is added.
Calls: "getters" and "setters"
Called by: the presentation layer
RD: Functional Requirement 7.1.7, Use Case 8.7.2

6.5.2.17 Ship.logic

6.5.2.17.1 Perform update main resource

Parameters: Int ammountOfMRes
Return value: none
Description: Adds upp the main resources.
Database: store new value for main resource
Pre-condition: none
Validity checks: there is enough storagemodules for the add up.
Post-condition: the function returns whether or not the player is added.
Calls: "getters" and "setters"
Called by: the logic layer
RD: Functional Requirement 7.1.3.1

6.5.2.17.2 Perform update secondary resource

Parameters: Int ammountOfSRes
Return value: none
Description: Adds upp the secondary resources.
Database: store new value for secondary resource
Pre-condition: none
Validity checks: there is enough storagemodules for the add up.
Post-condition: the function returns whether or not the player is added.
Calls: "getters" and "setters"
Called by: the logic layer
RD: Functional Requirement 7.1.3.2, Use Case 8.3.2

6.5.2.17.3 Perform change coordinates

Parameters: Vector Coordinates
Return value: none
Description: the function changes the location of the ship by updating the coordinates.
Database: searches for the ship, then updates the coordinates vector for the ship.
Pre-condition: none
Validity checks: the coordinate is inside the boundaries of the map.
Post-condition: none
Calls: "getters" and "setters"
Called by: the logic layer
RD: Functional Requirement 7.1.4.5, Use Case 8.4.1

6.5.2.17.4 Perform calculate damage

Parameters: String typeOfWeaponHit, String shipID
Return value: float damageDone
Description: the function calculates the damage another ship has done to the actual ship
Database: searches the ship in the database, then updates the energy of the ship
Pre-condition: none
Validity checks: none
Post-condition: none
Calls: "getters" and "setters"
Called by: the logic layer
RD: Functional Requirement 7.1.4.3 Use Case 8.5.7, 8.5.7, 8.4.2

6.5.2.17.5 Perform gather resources

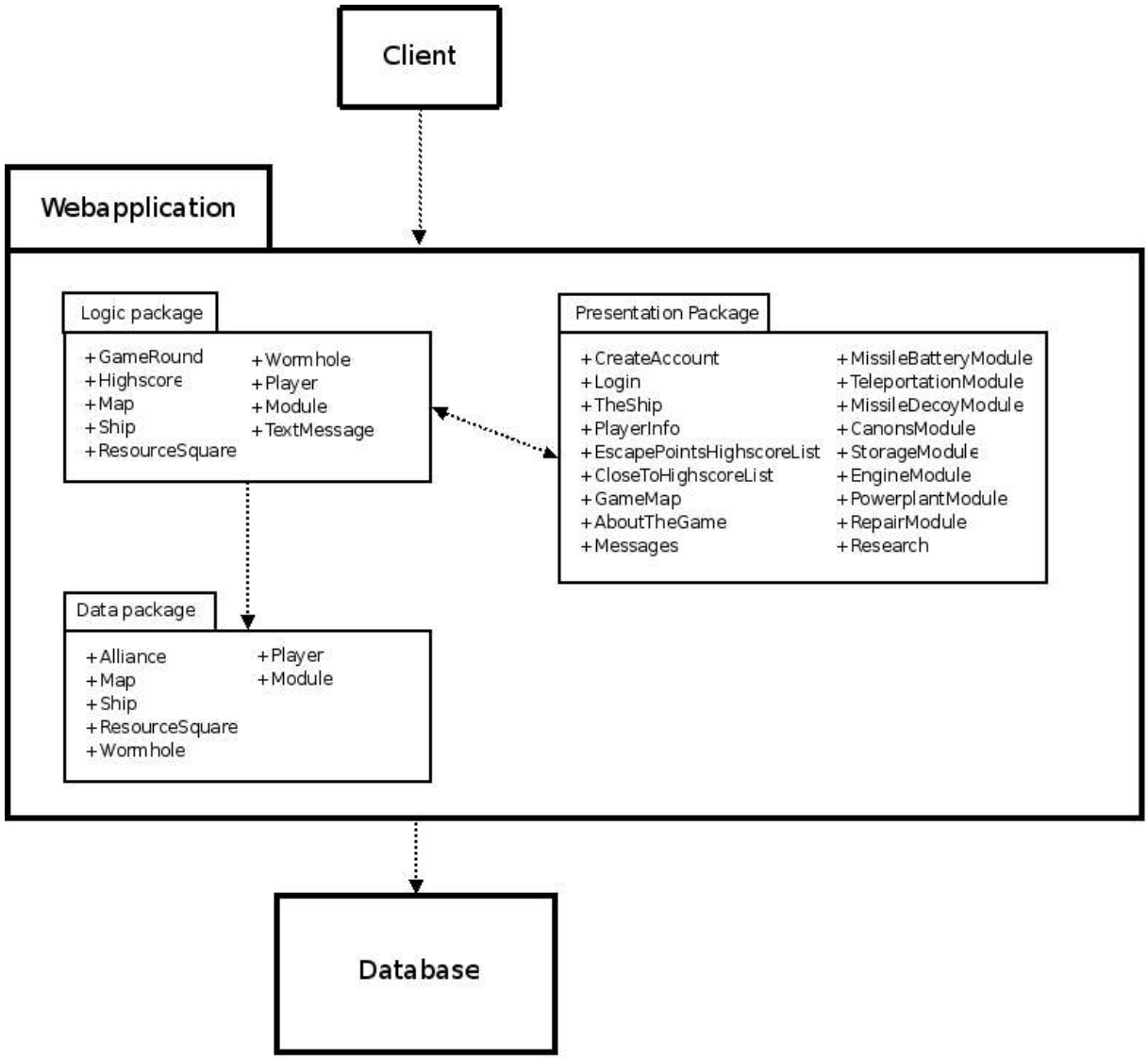
Parameters: none
Return value: boolean gathering
Description: the function sets the ship to start gather resources from a field.
Database: none
Pre-condition: the ship is in a resource square
Validity checks: none
Post-condition: the function returns whether or not the ship starts gathering resources
Calls: "getters" and "setters"
Called by: the presentation layer
RD: Functional Requirement 7.1.3.3, 7.1.3.2 Use Case 8.3.2

6.5.3 Requrements document cross referencing table

Functional requirements	Methods
7.1.1 Web page	Presentation layer
7.1.1.1 Create an account	DD 5.5.11.2
7.1.1.2 Log in with an existing account	DD 5.5.11.3
7.1.1.3 Read about the game	Presentation layer
7.1.1.4 Visit the forum	Presentation layer
7.1.2 Game map	DD 5.5.15
7.1.2.1 Function of the map	DD 5.5.15.1
7.1.2.2 The wormhole	DD 5.5.14
7.1.2.3 Startup placement of players	DD 5.5.12.2
7.1.2.4 Viewing the map	DD 5.5.15.1
7.1.3 Resources	DD 5.5.17.1, 5.5.17.2
7.1.3.1 Main resource	DD 5.5.17.1, 5.5.17.2
7.1.3.2 Secondary resource	DD 5.5.17.1, 5.5.17.2
7.1.3.3 Resource squares	DD 5.5.9
7.1.4 The ship	DD 5.5.17
7.1.4.1 Default ship	DD 5.5.12.2
7.1.4.2 Destroying ship	DD 5.5.17.4
7.1.4.3 Damage ship	DD 5.5.17.4
7.1.4.4 Repair ship	DD 5.5.10.7
7.1.4.5 Movements of ship	DD 5.5.10.1
7.1.5 Modules	DD 5.5.10
7.1.5.1 Offensive Weapons	Presentation layer
7.1.5.1.1 Missile batteries module	DD 5.5.10.3, 5.5.10.6
7.1.5.1.2 Cannons module	DD 5.5.10.2, 5.5.10.4
7.1.5.2 Defensive Weapons	Presentation layer
7.1.5.2.1 Teleportation module	DD 5.5.10.8
7.1.5.2.2 Missile decoys module	DD 5.5.10.5
7.1.5.3 Storage module	DD 5.5.10.11, 5.5.10.12
7.1.5.4 Engine module	DD 5.5.10.1

7.1.5.5 Power Plant	DD 5.5.10.9
7.1.5.6 Repair module	DD 5.5.10.7
7.1.6 High-score list	DD 5.5.13.1, 5.5.13.2
7.1.6.1 Escape points	DD 5.5.13.1
7.1.6.2 Skills star awards.	Presentation layer
7.1.7 Alliances	DD 5.5.16
7.1.7.1 Players in an alliance	Presentation layer
7.1.7.2 Benefits from being in an alliance	Presentation layer
7.1.8 Communication	Presentation layer
7.1.8.1 Text messages	DD 5.5.11.4, 5.5.11.5
7.1.9 Research	DD 5.5.10.13
7.1.9.1 Researching missiles	DD 5.5.10.13
7.1.9.2 Researching cannons	DD 5.5.10.13
7.1.9.3 Researching missile decoys	DD 5.5.10.13
7.1.9.4 Researching teleportation	DD 5.5.10.13
7.1.9.5 Researching engines	DD 5.5.10.13
7.1.9.6 Researching repair	DD 5.5.10.13

6.6 Package diagram



7 Functional Test Cases

7.1 Create an account

Description	Create an account
Reference	Functional requirement: 7.1.1.1 Create an account
Precondition	User name does not exist.
Input	User name, password, and email address
Expected Output	A new account is created
Instructions	1. Go to the games web page 2. Input the user name, password and email address in the specified text boxes 3. Click the button **Create Account**. 4. The text **Account created** is shown

7.2 Login to account

Description	Log in to account
Reference	Functional requirement: 7.1.1.2 Log in
Precondition	The account already exist
Input	The user name and the password to the account
Expected Output	The user will be logged in
Instructions	1. Go to the games web page 2. Input the user name and the password for the account 3. Click the button **Login**

7.3 Enter the wormhole

Description	Enter the wormhole and win the game
Reference	Use case: 8.2.1 Win a game round Functional requirement: 7.1.2.2 The wormhole
Precondition	Enough main resources to move into the wormhole
Input	The coordinates for the wormhole
Expected Output	The player wins the game round
Instructions	1. Be the first one to enter the wormhole 2. A text displaying the message **You Have Won** is shown

7.4 Gather resources

Description	Gather resources from resource square
Reference	Use case: 8.3.2 Gather resource Functional requirement: 7.1.3 Resources
Precondition	The ship has moved to a resource square and has enough main resources.
Input	-
Expected Output	The player will have more secondary resources.
Instructions	1. Click on the link **Game map** 2. Click the button **Gather** 3. A text displaying **Gathering resources** is displayed

7.5 Move the ship

Description	Move the ship from one location to another
Reference	Functional requirement: 7.1.4.5 Movement of ship
Precondition	Enough main resources.
Input	The location wanted to move to.
Expected Output	The ship starts to move to the location specified
Instructions	1. Click on the link **Game map** 2. Click the button **Move** 3. Click on the location on the map 4. A text **Movement is initialized** is displayed

7.6 (Auto) Repair the ship

Description	Automatically repair the ship
Reference	Use case: 8.4.3 Repair the ship Functional requirement: 7.1.4.4 Repair ship
Precondition	The ship is damaged
Input	-
Expected Output	The ship starts to repair
Instructions	1. Click on link **Module** 2. Click on the **On/Off** button on the repair module 5. A text **Repair is initialized** is displayed

7.7 Choose module

Description	Choose a module to adjust
Reference	Functional requirement: 7.1.5 Modules
Precondition	The module is built
Input	
Expected Output	The module page is displayed.
Instructions	1. Click on link **Module** 2. Click on the link in the wanted module

7.8 Build module

Description	Build a certain module
Reference	Functional requirement: 7.1.5 Modules
Precondition	Enough resources and module slots.
Input	
Expected Output	Module is built.
Instructions	1. Click on link Module 2. Click on the **Build new module** button 3. The text **Module is built** is displayed

7.9 Upgrade module

Description	Upgrade a certain Module
Reference	Functional requirement: 7.1.9 Research
Precondition	The module has been built.
Input	
Expected Output	The module is upgraded.
Instructions	1. Click on the link Module 2. Click the **Upgrade** button on the module line 3. The text **Upgrade is complete** is displayed

7.10 Remove module

Description	Remove module
Reference	Functional requirement: 7.1.5 Module
Precondition	The module has been built.
Input	
Expected Output	The module is removed.
Instructions	1. Click on the link Module 2. Click the **remove** button on the module line 3. The text **Module removed** is displayed

7.11 Build ammunition (missiles/shells)

Description	Build ammunition
Reference	Use case: 8.5.1 Build ammunition Functional requirement: 7.1.5.1 Offensive Weapons
Precondition	The weapon is built, there is enough room for the produced ammunition and enough resources.
Input	Number of shells
Expected Output	The given amount of shells is built
Instructions	1. Click on the link Map 2. Write the number of shells wanted in the text box named **Shells** 3. Click the **Build** button 4. The text **Shells built** is displayed

7.12 Fire shells - Includes Hit with a shell

Description	Fire shell
Reference	Use case: 8.5.3 Fire shells, 8.5.7 Hit with a shell Functional requirement: 7.1.5.1.2 Cannons module
Precondition	The shells exist
Input	Number of shells to attack with

Description	Fire shell
Expected Output	The focused player is attacked
Instructions	1. Click on the **Game map** button 2. Click on your targeted ship in the map 3. Specify how many shells wanted to attack him with by entering the wanted number of shells 4. Click the **Fire** button. 5. A text **Attack initialized** is displayed

7.13 Fire missiles - Includes Hit with a missile

Description	Fire missile
Reference	Use case: 8.5.2 Fire missile, 8.5.6 Hit with a missile Functional requirement: 7.1.5.1.1 Missile batteries module
Precondition	The missiles exist
Input	Number of missiles
Expected Output	The focused player is attacked
Instructions	1. Click on the **Game map** button 2. Click on your targeted ship in the map 3. Specify how many missiles wanted to attack him with by entering the wanted number of missiles 4. Click the **Fire** button. 5. A text **Attack initialized** is displayed

7.14 Teleport the ship

Description	Teleport the ship to another position
Reference	Functional requirement: 7.1.5.2.1 Teleportation module
Precondition	The teleport module is built
Input	
Expected Output	The ship is teleported

Description	Teleport the ship to another position
Instructions	1. Click on the **Game map** button 2. Hit the **Teleport** button 3. The ship changes coordinates on the map

7.15 Search for player in high score list

Description	Find a certain player in the high score list.
Reference	Functional requirement: 7.1.6 High-score list
Precondition	Player created.
Input	Name of the player
Expected Output	The position in the high score list.
Instructions	1. Click on the link **Escape-points** or **Close-to-center** to go to the high score lists. 2. Enter the player name in the text field **Search player**. 3. Click on **Search**. 4. The position on the high score list is shown.

7.16 Show player by rank

Description	Find players with a certain rank.
Reference	Functional requirement: 7.1.6 High-score list
Precondition	Player created.
Input	The rank.
Expected Output	All players with the certain rank.
Instructions	1. Click on the link **Escape-points** or **Close-to-center** to go to the high score lists. 2. Enter the rank in the text field **Show Rank**. 3. Click on **Search**. 4. The players with the certain rank is shown.

7.17 Create an alliance

Description	Create a new alliance and show a confirming message.
Reference	Use case: 8.7.1 Create an alliance
Precondition	Player created and not in an alliance.
Input	Name of alliance.
Expected Output	That the alliance has been created.
Instructions	1. Click on the link **Alliance**. 2. Enter the name of the alliance in the text field **Name of alliance**. 3. Click on **Create**. 4. The text **The alliance has been created** is shown.

7.18 Invite to an alliance

Description	Invite a player to join the alliance and show a confirmingmessage.
Reference	Use case: 8.7.2 Join an alliance Functional requirement: 7.1.7.1 Players in an alliance
Precondition	Alliance created and the player is not in an alliance.
Input	The name of the player to be invited.
Expected Output	The player has joined the alliance.
Instructions	1. Click on the link **Alliance**. 2. Enter the name of the player to invite in the text field **Name of player**. 3. Click on **Invite**. 4. Wait on the response. 5. The message **Player has joined the alliance** is shown.

7.19 Disband an alliance

Description	Disband an alliance and inform all players in alliance about it.
Reference	Use case: 8.7.3 Disband an alliance Functional requirement: 7.1.7.1 Players in an alliance
Precondition	Alliance created.
Input	-
Expected Output	A message is sent to all players in the alliance about the disbanding of the alliance and the alliance is disbanded.
Instructions	1. Click on the link **Alliance**. 2. Click on **Disband alliance**. 3. The message sent to players is shown.

7.20 Leave alliance

Description	The player wants to leave the alliance.
Reference	Use case: 8.7.4 Leave alliance Functional requirement: 7.1.7.1 Players in an alliance
Precondition	Player has joined the alliance.
Input	-
Expected Output	A message is sent to all players in the alliance that the player has left the alliance and the player is not in the alliance any more.
Instructions	1. Click on the link **Alliance**. 2. Click on leave **Alliance**. 3. The message sent to all players in the alliance is shown.

7.21 Dismiss player from alliance

Description	A players is dismissed from an alliance by the leader of the alliance.
Reference	Use case: 8.7.5 Dismiss player Functional requirement: 7.1.7.1 Players in an alliance
Precondition	Player has joined the alliance.
Input	Name of the player.
Expected Output	A message is sent to all players in the alliance that the player has been dismissed from the alliance and the player is kicked.
Instructions	1. Click on the link **Alliance**. 2. Enter the name of the player in the text field **Player to dismiss**. 3. Click on **Dismiss**. 4. The message sent to all players in the alliance will be shown.

7.22 Send text message

Description	Send text message to another player
Reference	Use case: 8.8.1 Sending a short text message Functional requirement: 7.1.8.1 Text messages
Precondition	Player created.
Input	Name of player, subject and text message.
Expected Output	Message is added to the player's message list.
Instructions	1. Click on link **Messages**. 2. Enter the name of the player to send a text message to in the text field **To**. 3. Enter the subject of the text message in the text field **Subject**. 4. Enter the message in the text field **Message**. 5. Click on **Send**. 6. The text **The message has been sent** is shown.

7.23 Read text message

Description	Read a new incoming message
Reference	Use case: 8.8.1 Sending a short text message Functional requirement: 7.1.8.1 Text messages
Precondition	A message is sent to the player.
Input	-
Expected Output	The text message is displayed.
Instructions	1. Click on the link **1 unread message**. 2. The message will be shown.

7.24 Delete text message

Description	Delete a text message.
Reference	Use case: 8.8.1 Sending a short text message Functional requirement: 7.1.8.1 Text messages
Precondition	Text message exists.
Input	-
Expected Output	The message will disappear from the list of text messages.
Instructions	1. Click on link **Messages**. 2. Click on **x** for the message you want to delete. 3. The text **The message has been deleted** is shown that the message has been deleted.

7.25 Start research

Description	Start research for a module.
Reference	Use case: 8.9.1 Research a research field. Functional Requirement: 7.1.9 Research
Precondition	Enough resources.
Input	-
Expected Output	The research for the module will be on.
Instructions	1. Click on the link **Research**. 2. Click **On** for the module to do research on. 1. The button is showing the text **Off** instead.

7.26 Stop research

Description	Stop research for a module.
Reference	Use case: 8.9.1 Research a research field Functional Requirement: 7.1.9 Research
Precondition	The module must be under research.
Input	-
Expected Output	The research for the module will be off.
Instructions	2. Click on the link **Research**. 3. Click **Off** for the module to stop research. 4. The button is showing the text **On** instead.

7.27 Add star

Description	Add a star for a certain research field.
Reference	Functional requirement: 7.1.6.2 Skills star awards.
Precondition	The maximum number of stars has not been reached for the research field.
Input	-
Expected Output	A star will be added for the research field in the column **Stars**.
Instructions	1. Click on the link **Research**. 2. Click on **Add star** for the module. 3. A star will be added for the module in the column **Stars**.

7.28 Focus on the map

Description	Focus on a set of coordinates on the map.
Reference	Functional requirement: 7.1.2.1 Game map
Precondition	Game round started.
Input	Coordinates on the map.
Expected Output	The map is centered on the input coordinates.
Instructions	1. Click on the link **Game map**. 2. Enter the x coordinate in the text field **X**. 3. Enter the y coordinate in the text field **Y**. 4. Click on **Focus**. 5. The map is centered on the specified coordinates.

7.29 Search player on the map

Description	Find a player on the map and center the map where the player is.
Reference	Functional requirement: 7.1.2.1 Game map
Precondition	Player is created.
Input	Player's name
Expected Output	Map that is centered on the player.
Instructions	1. Click on the link **Game map**. 2. Enter the name of the player in the text field **Player**. 3. Click on **Search**. 4. The map is centered on the player.

7.30 Cancel movement

Description	Cancel the movement of the ship and change the map view.
Reference	Use case: 8.4.1 Move the ship Functional requirement: 7.1.4.5 Movements of ship
Precondition	The ship is moving.
Input	-
Expected Output	The ship has stopped.
Instructions	1. Click on the link **Game map**. 2. Click on **Cancel movement.** 3. The movement is now shown on the map anymore.

7.31 Pan map view

Description	Pan the map in any direction.
Reference	Functional requirement: 7.1.2.1 Game map
Precondition	-
Input	-
Expected Output	The map will pan in the chosen direction.
Instructions	1. Click on the link **Game map**. 2. Click on the link **Go left>**, **Go right>**, **Go south>**, **Go north>** to pan the map view. 3. Click on the link **>>** for any direction to pan more than with >. 4. A different part of the map will be panned.