Hodaya Tabibian

Avi Negosa

Shai Kakiov

**Colony War**

Software Design Document

Name (s):

Lab Section:

Workstation:

Date: (03/16/2020)

|  |  |  |
| --- | --- | --- |
|  |  | Software Design Document |
| **TABLE OF CONTENTS** | |  |
| **1.** | **I**[**NTRODUCTION**](#page4) | **2** |
| 1.[1](#page4) | [Purpose](#page4) | 2 |
| 1.[2](#page4) | [Scope](#page4) | 2 |
| 1.3 | Overview | 2 |
| 1.4 | Reference Material | 2 |
| 1.[5](#page4) | [Definitions and Acronyms](#page4) | 2 |
| **2.** | **S**[**YSTEM**](#page4) **O**[**VERVIEW**](#page4) | **2** |
| **3.** | **S**[**YSTEM**](#page4) **A**[**RCHITECTURE**](#page4) | **2** |
| [3.1](#page4) | [Architectural Design](#page4) | 2 |
| [3.2](#page5) | [Decomposition Description](#page5) | 3 |
| [3.3](#page5) | [Design Rationale](#page5) | 3 |
| **4.** | **D**[**ATA**](#page5) **D**[**ESIGN**](#page5) | **3** |
| [4.1](#page5) | [Data Description](#page5) | 3 |
| [4.2](#page5) | [Data Dictionary](#page5) | 3 |
| **5.** | **C**[**OMPONENT**](#page5) **D**[**ESIGN**](#page5) | **3** |
| **6.** | **H**[**UMAN**](#page6) **I**[**NTERFACE**](#page6) **D**[**ESIGN**](#page6) | **4** |
| 6.1 | Overview of User Interface | 4 |
| [6.2](#page6) | [Screen Images](#page6) | [4](#page6) |
| [6.3](#page6) | [Screen Objects and Actions](#page6) | [4](#page6) |
| **7.** | [**REQUIREMENTSMATRIX**](#page6) | **4** |
| **8.** | **A**[**PPENDICES**](#page6) | **4** |

Software Design Document

1. **INTRODUCTION**

**1.1 Purpose**

This SDD document describes the architecture and system of the components of the mobile game “Colony War” and the precise implementation details required to satisfy the requirements as specified in the SRS document.

It is assumed that the reader has read the SRS, since this document also defines the implementation details of the desired behavior given the requirements within it.

Also, this document provides a description of the design of the system fully enough to allow for software development to proceed with an understanding of what is to be built and how it is expected to built. The Software Design Document provides information necessary to provide description of the details for the software and system to be built.

**1.2 Scope**

Description and Scope:

since this project is a mobile game, the software communicates with users- it takes care of users input: touches on screen and updates the game’s state accordingly.

The full description of game explained at the SRS document and at Vision document.

Also, the software communicates with Google Play Services in order to upload user’s score.

Goals, Objectives and benefits:

Goals-

1. Ability to respond to user input.
2. Ability to upload to Google Play Services the user’s score.
3. Ability make the game simple to use and easy to understand the rules.

benefits-

1. Make money from users purchases (in further release) and ads.

**1.3 Overview**

-System overview- (page: 6)

-System Architecture-architectural design, decomposition description and

And design rationale (page: 6-9)

-Data design- data description and dictionary (page 9-10)

-Compomonent design- (page 10-12)

-Human interface design- overview, screen images and screen objects and

actions (page 13-15)

-Requirments matrix (page 16)

**1.4 Reference Material**

-[SRS document](https://github.com/hodaya1995/proyekt_gmar/blob/master/SRS%20for%20Ariel%20Projects%202020.doc)

-[Vision document](https://github.com/hodaya1995/proyekt_gmar/blob/master/README.md)

**1.5 Definitions and Acronyms**

1. **SYSTEM OVERVIEW**

Colony War app is built on an open source framework. Source code is available using standard open source management tools such as Git. All source code is stored in a Github repository(https://github.com/hodaya1995/proyekt\_gmar). Any android user can contribute to the application. Colony War development language is Java. Github is used as the software development platform.

Also, at the app the user has a colony that in it he has soldiers, workers and buildings.

The user competes with enemy that has a colony too- this is generates by the software.

In summary, the system design includes the following sub-systems:

• Colony War mobile Application

•User Colony

•Enemy Colony

• Google Play Service Database

• Android Studio Development Platform

• Github Version and Source Code Control

There is no expectation that any of these systems will be changed or modified with the proposed system except Colony War mobile Application components.

1. **SYSTEM ARCHITECTURE**

**3.1 Architectural Design**

Login Page

Used as a main page, if the user is not connected The page includes:

* Login as a guest - Enter a nickname
* Login button
* Login button using google play account

If the input successfully passed the tests, the login button will execute

Server request for user authentication.

Once verified, the user will be taken to the main page and saved

The cookie we received for server-side authentication on requests

Future.

In failure, you will receive an error message as received from the server.

The error message will be displayed in the dialog window.

Links

* Home page link to main page
* New game
* Save game
* Change profile
* Logout will delete the cookies and return the user to screen
* Login

You can see below in illustration 1-the description of the log in.

Game structure

* Map: Selected by the user from a given selection
* Buildings:
  + Main building: This structure does the upgrading of the colony as well as creating a worker.
  + House: construction of residences for the population of the colony.
  + Military base: It is necessary to produce soldiers and to form an army, each base can contain a limited number of soldiers within it.
* Army: Consists of military bases and soldiers.
* Person: Each person has an amount of life and a number of a hit on the attack.
  + Soldier: - belong to the population who living in the military base.

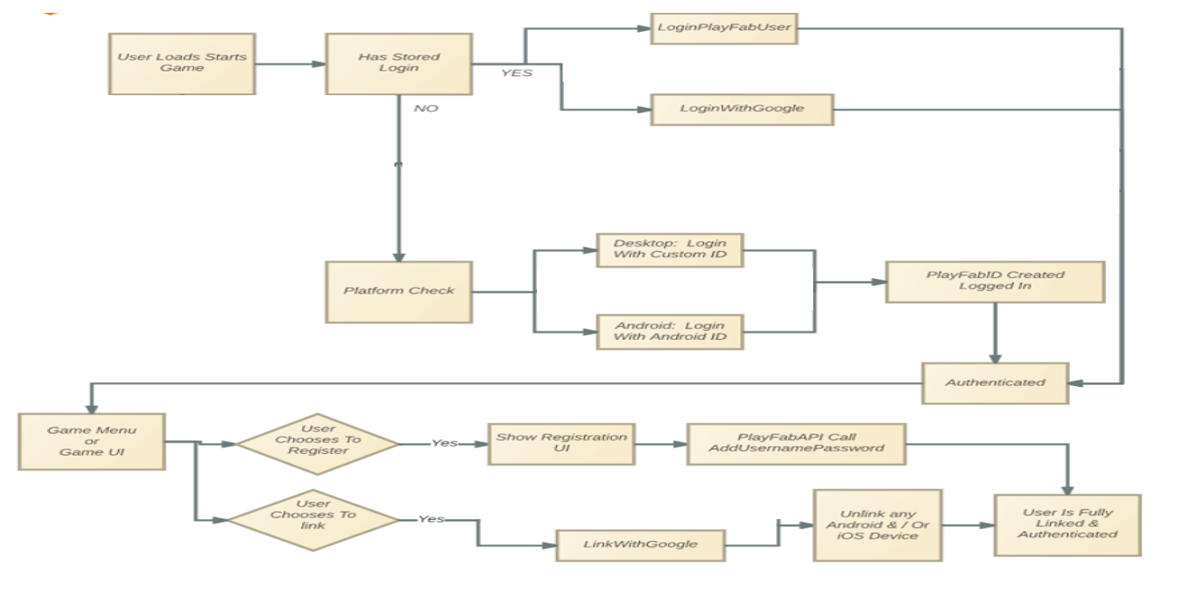
called when we go out for battles.

* + Worker: they role is reflected in the construction of buildings, and to work in the mines and trees.
* Resource:
  + Trees: which are scattered throughout the map.
  + Gold: provided by mines.

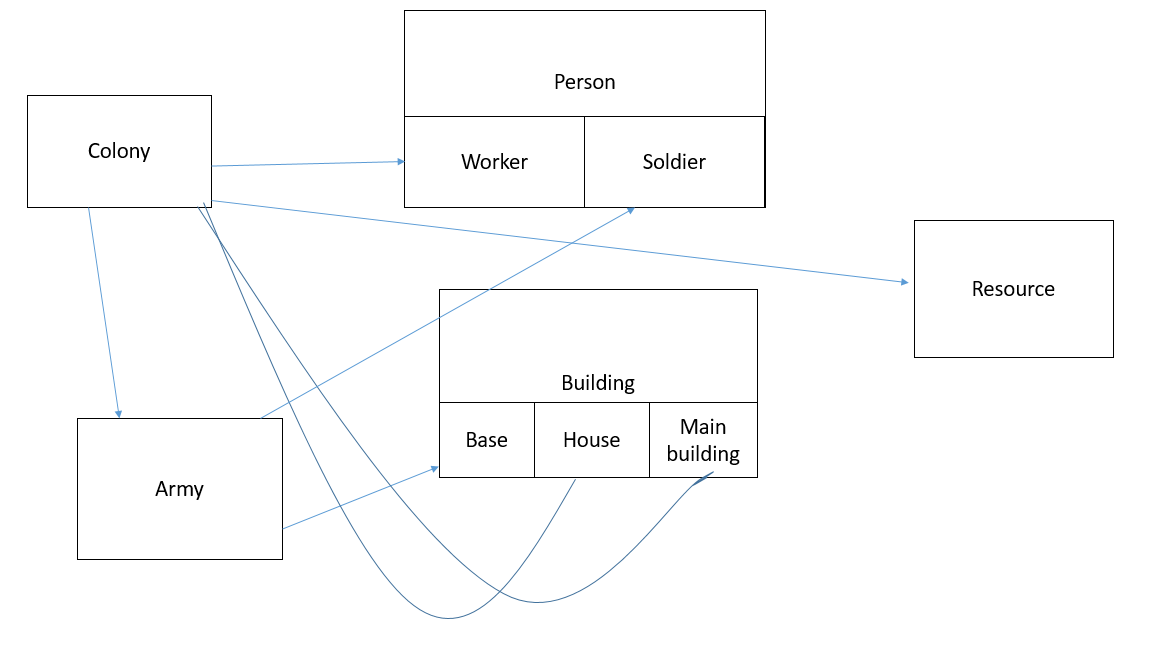
You can see below in illustration 2 the description of the objects of the game structure (in 3.2).

**3.2 Decomposition Description**

**Illustration 1-the process to connect to the game:**



**illustration 2: below we see illustration of the objects of the games and the dependency between them:** By the illustration we can see that Colony create worker.in addition Colony is a list of House and Main building and the army protect the Colony (existential dependency) and Resources create the Colony. Army is a list of soldiers and bases



**3.3 Design Rationale**

Person is a super entity.

Worker and Soldier are sub-entities inherited from Person.

In the same way a building entity is also a super entity of the other buildings.

Army is a list of soldiers meaning there is an existential dependency between Army to base.

The idea is that a feature cannot depend on a non-Super Key group to create a normalization level of BCNF(see illustration 2 for description).

1. **DATA DESIGN**

**4.1 Data Description**

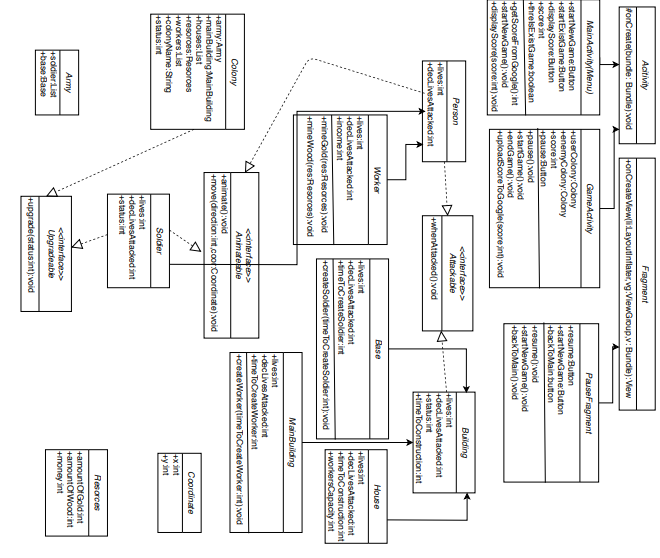
With firebase any registered user in google can be connected by

His google account and login his details are saved by using firebase auto

Also the user profile picture will be saved in firebase storage and also images, sound and other entity data(see illustration 3).

**4.2 Data Dictionary**

**Illustration 3: functional and objects description-class diagram:**



1. **COMPONENT DESIGN**

**public** **int** getScoreFromGoogle() {

**int** score=0;

Games.Leaderboards.loadCurrentPlayerLeaderboardScore(getApiClient(), getString(R.string.leaderboard\_id), LeaderboardVariant.TIME\_SPAN\_ALL\_TIME, LeaderboardVariant.COLLECTION\_PUBLIC).setResultCallback(**new** ResultCallback<Leaderboards.LoadPlayerScoreResult>() {

@Override

**public** **void** onResult(**final** Leaderboards.LoadPlayerScoreResult scoreResult) {

**if** (isScoreResultValid(scoreResult)) {

score = scoreResult.getScore().getRawScore();

}

}

});

**return** score;

}

**public** **void** startNewGame(){

startGame();

}

**public** **void** displayScore(**int** score){

**this**.score=score;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **void** pause(){

**this**.onPause();

}

**public** **void** startGame(){

startActivity(**new** Intent(MainActivity.**this**,GameActivity.**class**));

}

**public** **void** endGame(){

pause();

backToMain();

}

**public** **void** uploadScoreToGoogle(**int** score){

PendingResult<Leaderboards.LoadPlayerScoreResult> result = Games.Leaderboards.loadCurrentPlayerLeaderboardScore(gameHelper.getApiClient(),"LEADERBOARD\_ID",LeaderboardVariant.TIME\_SPAN\_ALL\_TIME, LeaderboardVariant.COLLECTION\_PUBLIC);

result.setResultCallback(**new** ResultCallback<Leaderboards.LoadPlayerScoreResult>() {

@Override

**public** **void** onResult(Leaderboards.LoadPlayerScoreResult loadPlayerScoreResult) {

Games.Leaderboards.submitScore(gameHelper.getApiClient(), "LEADERBOARD\_ID",loadPlayerScoreResult.getScore().getRawScore()+ score);

}

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **void** resume(){

**this**.inResume();

}

**public** **void** backToMain(){

startActivity(**new** Intent(GameActivity.**this**,MainActivity.**class**));

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **void** whenAttacked(){

**this**.lives--;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **void** mineGold(Resources res){

res.amountOfGold++;

}

**public** **void** mineWood(Resources res){

res.amountOfWood++;

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **void** createSoldier(**int** timeToCreateSoldier){

soldiers.add(**new** Soldier(timeToCreateSoldier));

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **void** animate(){

Manager.createPlayer( file.toURL() ); //gif file

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **void** moveTo(**int** direction,Coordinate coor){

**final** AnimatorSet mAnimatorSet = **new** AnimatorSet();

mAnimatorSet.playTogether(ObjectAnimator.ofFloat(**this**, "x", (**float**) coor.y, (**float**) coor.x),

ObjectAnimator.ofFloat(**this**, "y", (**float**) coor.y, (**float**) coor.x, ObjectAnimator.ofFloat(**this**, "rotation", 360)

mAnimatorSet.setDuration(1000);

mAnimatorSet.start();

mAnimatorSet.addListener(**new** Animator.AnimatorListener() {

@Override

**public** **void** onAnimationStart(Animator animation) {

}

@Override

**public** **void** onAnimationEnd(Animator animation) {

mAnimatorSet.playTogether(ObjectAnimator.ofFloat(**this**, "x", **this**.getX(), (**float**) coor.y),

ObjectAnimator.ofFloat(**this**, "y", **this**.getY(), (**float**) coor.x));

mAnimatorSet.setDuration(1000);

mAnimatorSet.start();

}

@Override

**public** **void** onAnimationCancel(Animator animation) {

}

@Override

**public** **void** onAnimationRepeat(Animator animation) {

}

});

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **void** createWorker(**int** timeToCreateWorker){

workers.add(**new** Worker(timeToCreateWorker));

}

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**public** **void** upgrade(**int** status){

**this**.status=status;

}

Software Design Document

1. **HUMAN INTERFACE DESIGN**

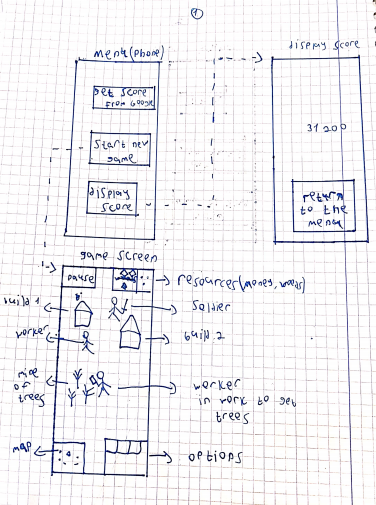
**6.1 Overview of User Interface**

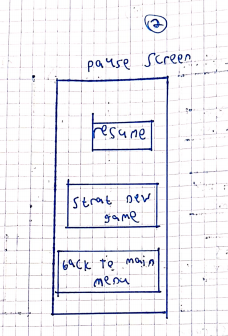
the game is on phone by touch-click.

the user needs to win in the game.

**illustration 4- Here I will show diagram of action to go to the game itself and return to the main Menu (here I show the features and feedbacks). after that I will explain the game itself on the screen game:**

dictionary: what in slanted writing is not button but regular display. in regular writing is button display.





the game screen (page 1-game screen) itself we build buildings and create soldiers by clicking on the options that we have on the game. we fight to defeat the enemy, collect resources by workers on the mines and more. The game is in generalness and in the future of this part i will explain in more details.in the end- the game will be achieved on click-touch to manage the game.

**6.2 Screen Images**

here the interface. the game is used by touch click on the phone. the game screen (below the Menu in the drawing-illustration 4 page 1) is example of the game that occur in one moment and not show the all game event.

dictionary of the game screen (illustration 4):

map-little screen in the left-down of the screen of the phone-with that we can to click on the little map to move on the coordinates on the big map(the phone itself).

options-little screen in the right-down of the screen of the phone-there we see kinds of options, for example-by clicking on building of soldier in the map we can see in the options kinds of options of the soldiers that we can create.in addition option of update the building.

**6.3 Screen Objects and Actions**

what I will write its rely on the drawing in the previously paragraph (illustration 4)

**screen Menu-**

**get score from google-**here we go to google see the score of our game.

**start new game-**we open the game screen and start to play.

**display score-**open new screen that show us the score of the last game.

**screen display score-**show us the score of the last game.

**return to the Menu-**option to return the screen of the Menu.

**screen game-screen-**the game itself. there we see all the map with our buildings, soldiers, enemy, resources and more .in the left up of the screen game we have pause button-by clicking that we are going to the pause screen.in addition we move on the map by clicking the little map on the left down of the screen(that transfer us to random point on the map-point that we choose to go).

we choose soldier/worker/building/more.. by click-touch of the phone and if we want to attack/work and more.. we just click on the enemy or mine to move the object to do his job.

the little screen in the up right of the screen are the resources that show us how much we have.

in the down-right of the screen we have little screen that show us options(like to create soldier-that by clicking on the building for example or upgrade the building and more..).

**screen pause-screen**

**resume-**this button allows us to continue the game we played last and return to the game screen.

**start new game-**to start new game and go to the game screen.

**back to main Menu-**by click on that option we return to the Menu.

1. **REQUIREMENTS MATRIX**

**Table-illustration number 5- the cross-­reference that traces components and data structures to the requirements in our SRS document:**

|  |  |  |
| --- | --- | --- |
| Links and game structure (Architecural Design-see 3) | User interface | 3.1.1 |
| - | Hardware interfaces | 3.1.2 |
| - | Software interfaces | 3.1.3 |
| - | Communication interfaces | 3.1.4 |
| All part of game structure and List data structure (Architecural Design and data structure-see 3 and 4) |  | 3.2.1-3.2.14 |
| Links(see 3) |  | 3.2.15-3.2.17 |

The requirements are suited to the parts 3 and 4.

The requirements list is in the end of the SRS document in page 15.

1. **APPENDICES**