Team project - cover page

Kaizen Team

GUID 1

2

3

4

5

Contents

[User roles 3](#_Toc535371910)

[General priorities 3](#_Toc535371911)

[User stories 3](#_Toc535371912)

[Non functional requirements: 5](#_Toc535371913)

[Sprint 1 (also for 2) review and report: 6](#_Toc535371914)

[Burndown chart 7](#_Toc535371915)

[Assumptions 7](#_Toc535371916)

[Test cases 7](#_Toc535371917)

[Deficiencies 7](#_Toc535371918)

[Screenshots: 7](#_Toc535371919)

[Architecture points: 7](#_Toc535371920)

# User roles

* User – includes player
* AI – automated player
* Robot – anything that is automated that is not a player including dealing cards

# General priorities

* Must have – the playing of the game
* Should have – persistent data
* Could have – JavaScript GUI
* WLTH – all other fancy features

# User stories

|  |  |  |  |
| --- | --- | --- | --- |
| NO | Story | Conversations | Priority |
| 1 | As a user I want to be able to choose cmd mode. | Done initially via cmd flag. -c Basic reading of command line is in package. | MUST (anything that is cmd is a must) |
| 2 | As a user I want to open online mode. | Done initially via cmd flag. -o | COULD |
| 3 | As a user I want to load deck of cards. | There is only one deck of cards used, no need for a file chooser. This is by positioning 'StarCitizenDeck.txt' in home folder of the program. No specification to choose the file. | MUST |
| 4 | As a user(hereby U) – store persistent data on game. | Done using a store database calss in java. | SHOULD |
| 5 | As U i want to retrieve persistent data on past games. | Done using retrieve database class. Prompted at the beginning of the program. | SHOULD. |
| 6 | As U I want to receive the program state log in cmd mode. | Involves a number of attributes to be shown. Can be implemented with extra -t flag in cmd mode. | SHOULD |
| 7 | As U or AI I want to receive fair share of cards. | Implemented in main gameplay – faily easy for 2, 4,5 players. In the case of 3 players we need special rule. | MUST |
| 8 | As a robot I want to randomize cards. | Done through array list collections shuffle method. | MUST |
| 9 | As a user I want to see details of top card. | Implemented by wrapping detail in card object, and card object in array list. Easy to retrieve. | MUST |
| 10 | As AI I want to choose best option for top card. | Choose an arraylist of details and choose max. Could implement hash set here. | MUST |
| 11 | As a robot I want to choose random first player. | Get random int from 1 to x, where x is the max number. | MUST |
| 12 | As a robot I want to generate relevant sql statements for the user for both input and output to the database. | Java class for postgres implemented. Each of the relevant input output info should be implemented in a variable in the classes. | SHOULD |
| 13 | As U I want to choose between online and cmd mode. | Easily done with a Boolean variable and cmd flags. | COULD |
| 14 | As U I want to see round, active player and card drawn (other details as well if needed). | Printed in standard out and in some way in html for web development. | MUST |
| 15 | As U I want to select category for the round. | Choose a number for specific category. | MUST |
| 16 | As a robot I want to determine who wins or loses the round. | Count cards is probably most reliable way but there should be other ways to do it. | MUST |
| 17 |  |  |  |
| 18 |  |  |  |

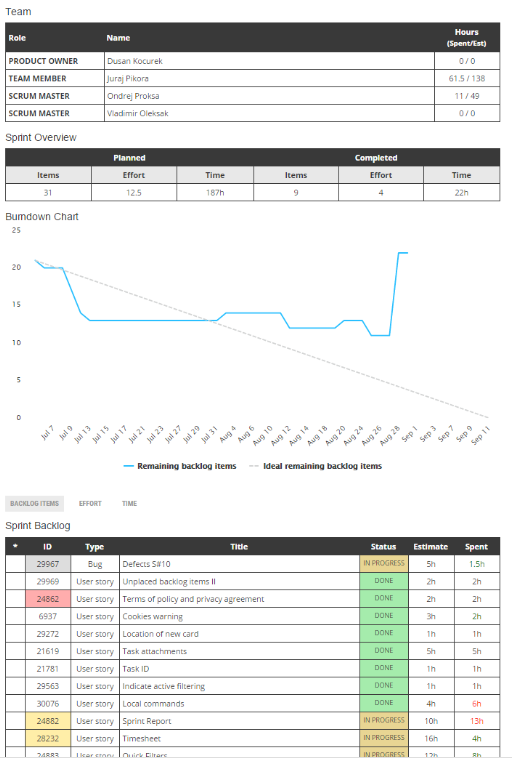
|  |  |  |  |
| --- | --- | --- | --- |
| X | As a user I would like to choose from gui or command line the relevant deck . | Can be implemented in command line as args[x] or using JFileChooser. | WLTH |
| Y | As a user I want to see card artwork in GUI. | Drawn and added as image or png on web. | WLTH |

Story points,estimated and actual time to be determined.

# Non functional requirements:

* Postgres sql database use
* Cli used as per template package;
* Implement maven as per template package;
* Java for main application, javascript for web development.
* Agile approach – scrum.
* Jar file method used for running.
* …

# Sprint 1 (also for 2) review and report:



Blablabla some comments on how it went must have planned and actual velocity.

Gantt chart.

# Burndown chart

How much is left as seen in diagram above.

# Assumptions

- compile from document

- get some of your own

# Test cases

- test per story card

- Junit may be redundant in this case

# Deficiencies

- will write at the end;

# Screenshots:

- will be done towards the end;

# Architecture points:

- model = this is storing data essentially from the relevant cards

- view = this is the gui or command line

- controller = anything that has to do with the non-static nature of cards

- player = object

- card = object

- everything else is implemented as we go – cards themselves may change