Goal:
develop the game "Morra Cinese" using:

- Node.js
- Angular.js

Description
of the mechanics of the implemented implementation:

Web Clients connect to the Node.js Web Server
An index.html is served by the server to the client when the client asks for the root URL '/'.
The index.html requires from the server the other necessary resources (and loads javascript files, images, html and css files). The index.html includes a ng-app angular.js directive which "starts" the single page Angular-js app

CLIENT
The client is required to signup with a username
It is then redirected to a new route where the client can challenge another player (selecting from the available ones).
The client can challenge a real player (if available), or a ROBOT player.
The match is played at the best 2 sets out of three.
Client and server communicate via Web Sockets.

SERVER
The client connects via Web Socket to the Server.
The server creates a Player for the new connected client. When the client challenges another client (or a Robot), the server creates a Match object to manage the game. The match object has a reference to both the challenger player and the challenged player and keeps updated status and statistics about the game. The server manages the execution of the matches.

KNOWN ISSUES AND IMPROVEMENT
improvement => Test definition to be completed
improvement => Use Rooms instead of bradcasting events
issue=> Set time-out to avoid that a player waits forever if the other one is not moving.
improvement => add a db layer to save statistics
improvement => move any configuration options out of source code
improvement => better User experience

SEE TABLE 1, TABLE 2 and TABLE 3 for more details about architecture, source code organization, test environment.

Topic	Description	Notes
Software Architecture	**- BackEnd in Node.js (Server)** **- FrontEnd in Angular.js (Web clients)**	**WebSockets** (socket.io based) have been used in order to make possibile: - real-time multiplayers experience - pushing of notifications from server to clients
FrontEnd Details	Angular.js based Introduced some a few other libraries in order to manage cookies, routing and websockets	**bower** (using the bower.json config file) used to install and mantain the required front end libraries With bower installed: cd to <morra_installation_dir>/client run *bower install*

		A dir bower_components will be created and pupulated with required javascript libraries
BackEnd Details	Based on Node.js a and Express Socket.io for WebSockets	**npm** (using the package.json config file) used to install and mantain the the required back end libraries With npm installed: cd to <morra_installation_dir> run npm install *A dir node_modules will be created and pupulated with required node librari*

TABLE 1- High Level Description

File/Folder	**Description**
client	Folder containing the Angular.js app In: <morra_installation_dir>/client/partials => partial htm file served when routing to new router.js <morra_installation_dir>/client/bower_componentes => javascript resources <morra_installation_dir>/client/main.js => main source file loaded by index.html <morra_installation_dir>/client/index.html => starting point for the application <morra_installation_dir>/client/public => static resources Other file are related to test. See TABLE 3 - Test Environment
config	Folder containing the file router.js which is used to serve the index.html file (which is responsibile for loading all the front end application on the client side requiring the necessarily libraries)
config.js	File keeping the main configuration params. Example: listening port
doc	Folder including this document
index.js	Main source file loading all the other node modules. To start the node application run: ***node index.js***
lib	Folder containing a file to manage static resources
node_modules	Folder populated executing the **npm install** command
package.json	Main development/deployment/test configuration file used by npm
README.md	README file
ws	Contains the **- ws/lib** folder which includes the following three source files: player.js: class modelling the Player match.js: class modelling the Match between two players morra_utils: includes a function used to compute a random selection (pietra, carta, forbici) - ws/test_client - ws/run_mc_withrobot.js - ws/run_mc1.js - ws/run_mc2.js See TABLE 3 - Test Environment for a description of these files.
test	See TABLE 3 - Test Environment for a description of these files

TABLE 2- Source Code Organization

Server Side test	Test implemented using mocha Tests are in <morra_installation_dir>/test One installed mocha, run **mocha** to execute the tests NOTES: only test in test_match.js and test_player.js implemented TODO: add missing tests
Front End	Test implemented using **protractor** **Test config file: protractor.conf.json** **Test files: <morra_installation_dir>/client/e2e-tests/scenarios.js** **Once installed protractor, to execute the tests move to <morra_installation_dir>/client/ and run** **protractor protractor.conf.json** NOTES: just a few tests implemented **TODO: add missing tests**
Integration and Isolation test	**<morra_installation_dir>/ws** **contains an alternative client that has been used to test the server with another command line client.** **For example to execute a test match of a player against a robot:** **- start the server: node index.js** **- move to ws: cd <morra_installation_dir>/ws** **- run the test: node run_mc_withrobot.js**

TABLE 3- Test Environment