**Please note that, *contrary*to Iteration 1, here reqs for Tier2 and Tier3 are independent of Tier1: you need to choose upfront what kind of app you want to develop (and test!).**

**Also, this is a preliminary version of the  functional requirements for Iteration 2: it may be affected by the results of Iteration 1.**

**Additional requirements with respect to documentation, traceability and testing will be posted later.**

**Tier 1: for a mark between 50 and 76**

FIRST you must have or get to an **integrated**GUI and game that supports AT LEAST**tier 2** of the correction grid for Iteration 1.

SECOND you must turn your game into a networked (client/server) app.

THIRD you must complete as much as of  the game to get to tier 3 of of the correction grid for Iteration 1.

**Tier 2: for a mark between 76 and 89**

FIRST you must have or get to an **integrated**GUI and game that supports AT LEAST**tier 2** of the correction grid for Iteration 1.

SECOND you must turn your game into a web-app (using Spring if using Java).

THIRD you must complete as much as of  the game to get to tier 3 of of the correction grid for Iteration 1.

**Tier 3: for a mark of 90 or above**

FIRST, satisfy the requirements of Tier 2 of Iteration 2

THEN AND ONLY THEN:

a) add a cheat feature allowing a player to know how many cards there are in each stage of a quest

b) implement more complex AI strategies: I expect to come up with at least 1.

c) get rid of hot-seat playing when no AI players are involved:

   - the game puts in play the next story card

   - if it's a quest: players race to sponsor it within a certain deadline. Ie first to commit first to sponsor.

   - players have a deadline within which they decide to participate or not in a quest. If they do not respond in time, they do not participate

   - same idea for choosing cards to play in a stage: by default a player is not playing any...

   - same ideas for playing in a tournament: by default a player is not playing any...

   - tests allowing for totally asynchronous bidding. That is, there is no order for bidding and you can bid twice in a row.

Finally,as a fundamental non-functional requirement for ALL tiers: you MUST use patterns where relevant AND refactor to eliminate code redundancy as much as possible. (Be sure to label your commits as refactorings when that's what they do). You will need, in your second demo, to explain your choice of architectural pattern, and your use of other patterns, among other things...