**1. Specific requirements**

**1.1 External interface requirements**

**1.1.1 User interfaces**

**1.1.2 Hardware interfaces**

**1.1.3 Software interfaces**

**1.1.4 Communications interfaces**

**1.2 Components**

**1.2.1 Cards**

**1.2.2 Rules**

**1.2.3 Game Board**

**1.2.4 Players**

**1.2.5 Game Engine**

**1.1.1 User Interfaces**

Should this be all command line, or do you want to build a simple GUI? I'm not thinking any sort of animations, just a 2D graphical representation of the game state at any time. This will affect how we build the game engine, as it would have to handle these aspects. Personally, I don't see why we couldn't make a really simple GUI - I've done this by myself for school projects in much less time, but we all need to agree.

In any case, the game will need to poll the user for the number of players ( I think 2 - 4 per deck of 52 cards is standard). The name of the players, etc. It also has to have a way to say who's turn it is, display possible moves, and accept whatever move the player makes (if it is allowed by the rules).

**1.1.2 Hardware Interfaces**

A computer with the Java run time environment.

Specify Java version?

**1.1.3 Software Interfaces**

We aren't interacting with other software, I don't think...

**1.1.4 Communications Interfaces**

Should this be the file structure, as in, the component files and how the engine will read them?

**1.2 Components**

The really tricky thing about this project is the majority of the components will be text based (cards, players, board, rules) with the only the game engine being actual code. All of these parts can be modeled in Java as generic Objects, likely with other objects to help out, but the information to be applied to the objects during instantiation will have to come from a text file. Reading the file isn't a problem, or course, but we have to figure out how to organize the data, and what formal structure to use. I think the hardest part of that will be translating the rules into a formal logic which can be interpreted by the engine.

**1.2.1 Cards**

Kyle started working on this section. It looks good, except that I don't think we should specify any sort of code related requirements in the non-code components. For example, Kyle, you have this for the cards:

4.     A method needs to be created to “shuffle” the cards before the beginning of play

5.   Card order in the deck shall be stored in an array.  
6.      Each card will need a value assigned to it (e.g. A = 1, Q = 12).  Multi-dimensional array for the deck?

but the card components will be a plain text file, so there are no data structures like arrays to be used. Similarly, there aren't any requirements concerning methods at this point. This should be specified somewhere, most likely in the objects used in the game engine component. That way, the cards and such aren't couple to any particular programming language either. Also, describing the cards shouldn't include any rules on how the cards are actually to be used.

**1.2.2 Rules**

Should we just rewrite the known game rules here? If so, we should probably clean up the language a bit. We could attempt to use a formal method language, but I don't know if we will really have time to do that right now.

**1.2.3 Game Board**

Similar to the rules, I think we just need to write out how the playing field is arranged...

**1.2.4 Player**

- Can be Human or AI

- What else?

- Number of players or players per deck in use is more of a rule thing...

**1.2.5 Game Engine**

I think this part can be written out as class/objects and attributes/methods. I was seeing these as a few objects as part of the card game engine... (A UML diagram would probably be a good appendix to the SRS, which I can work on if you all agree)

Enum Suite {Diamond, Club, etc}

Playing Card (technically, we are using French Suite playing cards)  
- Integer value [A - K], where A, J-K are integer constants

- Suite suite

Deck

- Array of Cards, custom accessor methods <- this is much more efficient than using built in data structures since we know the max size at run time, and we won't need all of the built in methods of Lists and such

- Methods like shuffle, draw 1 card, draw X cards, exchange 1 card, exchange X cards

Enum PlayerType {human, AI, etc}

Player

- PlayerType

- Boolean isPlayersTurn

- String name

GameBoard

- Specific zones to specify the deck, discard, player area, but how to generalize this to work for different types of games with different card zones on the board?

- It's hard to conceptualize how this will work from a file. Maybe the text file can specify how many zones the board needs to be divided into, and what each zone is used for...? It's hard to say that nothing is coupled to the engine when the design of the engine has to be coupled to the formal method used to describe these components...

Other classes and methods for the engine will depend on the type of interface (Command Line vs GUI ).

**For GUI**

CardGameThread implements Runnable

- Run - run the game

- Method to paint if using GUI

MouseListener AND/OR KeyboardListener

ScreenFactory

etc