Assignment 4

“The Showdown”

We see you made it. You’re here at last, facing the finale of the software engineering workshop. As you already know by now, the final phase in Texas Hold ‘em is called “the showdown”, and it is time to see your hand. In this assignment, you have one objective: make your hand a winner, i.e. make everything work.

That being said, as you already know, we prefer quality over quantity, so prioritize your work accordingly. God is in the detail, we aren’t.

There are no new functional or non-functional requirements.
For your convenience, at the end of this document you will find a list of all the requirements and use-cases you were given during the semester.

Preparation

We expect that you come prepared to the showdown. The main objective of this preparation is not wasting a single attosecond on silly setup issues.

Checklist

- There are, at least, two computers **already running** and **connected**:
    - Computer 1 runs the server and the desktop client
    - Computer 2 runs both the desktop and web clients
- On each computer (2 or more) you have the latest versions of:
    - Server
    - Desktop client
    - Web client
    - Unit, integration, and acceptance tests
    - Documents and models (detailed below)
- Your database is empty.
- All models are already open in your modeling tool, or printed out.
- You did not push a new version in the last 10 minutes before the showdown.
- You did not drink from a puddle.
- You had a good night’s sleep, and are moderately sober.
- Your backup database is ready, just in case (detailed below).

Backup Database

In order to avoid “show stopping” errors in the early parts of the showdown, you need to prepare either a script, a separate database, or some other strategy, to allow quick population of your database with representative data, i.e. bring your database to a non-trivial state. For example:

- Multiple users in each league
- Multiple games with different preferences
- Replay
- Enough statistics for the web client to show

Required Models and Specifications

Make sure you have an up-to-date version of the following:

- Glossary
- Use-case descriptions (**not** use-case diagrams)
- Acceptance tests descriptions
- System sequence diagrams (or activity where applicable) for all use-cases, and detailed sequence diagrams for the following use-cases:
    - Spectate active game
    - Store all the information from a game, such as: actions performed by all players in the game, the cards dealt at each round, round beginning and end, etc.
    - Support playing a Texas Hold'em game: dealing cards, placing blind bets for players, allowing players to check (NOP), fold or bet according to the game rules, etc.
    - Find all active games which the user can join
- Class diagram of the entire system
- Component diagram of the entire system
- Desktop GUI statechart

Class Presentation

Congratulations, your Texas Hold ‘em application is a huge success, and you are ready to take on the world. Unfortunately, the world doesn’t speak English. It’s up to you to support new languages and locales.

A team member will prepare a 10-minute discussion on the topic of L10N/G11N/I18N. Please refer to the following points:

- What is each of L10N, G11N, I18N?
- How are they different from each other?
- Aside from translation, what other changes should be made during the processes above?
- How does one support these processes? Show examples in your implementation language.

Please refer to the presentation guidelines at the Assignment page.

NOTE: Unlike previous presentations, this presentation must **NOT** exceed 10 minutes, and the topic presented is in no way a new requirement. As mentioned at the beginning of the document: there are no new functional or non-functional requirements.

Appendix

Functional Requirements

1. Register new user
2. Logging a registered user in and out of the system
3. Edit user profile: password, email, personal avatar
4. Create new Texas Hold ‘em game, according to customizable preferences:
    - Game type policy: limit | no limit | pot limit
    - Buy-in policy: the cost of joining the game
    - Chip policy: determine the amount of chips each player is given
      (a value of zero means the game is played with real currency)
    - Minimum bet (equals to the big blind)
    - Min/Max number of players
    - Enable/Disable spectating
5. Support playing a Texas Hold 'em game: dealing cards, placing blind bets, allowing players to raise/call/check/fold according to the game rules, etc.
6. Join an existing game
7. Spectate an active game
8. Leave a game

9. Replay games that are no longer active
10. New users are given an “unknown” rank, which is calibrated over their first 10 games (a user with an “unknown” rank may play in any league)
11. Redistributes players among leagues once a week with the following constraints:
    - Each league contains at least two users
    - All leagues should have the same number of players ± 1
    - If there are not enough players to fill all leagues, prefer filling the higher leagues first (lower leagues are allowed to be empty if the system has very few users)
12. Store all game information: actions performed by all players, cards dealt at each round, pot, showdown
13. List all active games which either a user can join, or available for spectating, optionally filtered by:
    - Player name
    - Pot size
    - Game preferences as listed in 4
14. Each game has a single message stream (i.e. chat) to which both players and spectators can publish messages, according to the following rules:
    - Players and spectators can see player messages
    - Players cannot see spectator messages
    - Players can send whispers (private messages) to anyone in the chat
    - Spectators can send whispers only to other spectators (not to players)
    - Whispers are private and can only be seen by the sender and the recipient
    - Messages are sent in a “push” manner to connected players only
15. Keep leaderboards (top 20 users) sorted according to:
    - Total gross profit
    - Cash gain
    - Number of games
16. Keep statistics on each user:
    - Average cash gain per game
    - Average gross profit

REMINDER:
Requirements 1-14 prescribe interaction between your desktop client and server.
Requirements 15-16 are initiated by your web client ***only***, and should not be implemented in the desktop client.
The web client should also support requirement 2.

Non-functional Requirements

Security

1. Passwords should be encrypted
2. End-to-end communication should be encrypted

Reliability

1. The system should not crash. Duh.
2. Data should not be lost in case of system failure
3. Multi-stage processes should be done in a transactional manner
4. Errors should be logged

Testability

1. Non-trivial logical units should be thoroughly unit-tested
2. Non-trivial interactions between units should be integration-tested
3. All use-cases should have accompanying acceptance tests that cover all scenarios
4. The system should be load- and stress-tested
5. All tests should be able to run automatically