Requirement Analysis Document -
Group 16

The goal of this project is to develop a computer-based version of the classic card game UNO by Mattel. The primary focus is to preserve the core gameplay mechanics while enabling a multiplayer experience. Additionally, the design will allow for adaptability, making it possible to support various versions of UNO by utilizing object-oriented principles.

GUI sketch/mockups

The GUI is designed to be visually appealing and easy for players to navigate and interact with. We want it to be simple to understand the state of the game and make moves. It consists of a start view, a lobby view and a game view. We have a figma prototype to display our vision.

The start view, our main menu, serves as the starting point. In the middle of the screen there's a large play button, and a question mark button for instructions and rules. Pressing play switches the player over to the lobby.

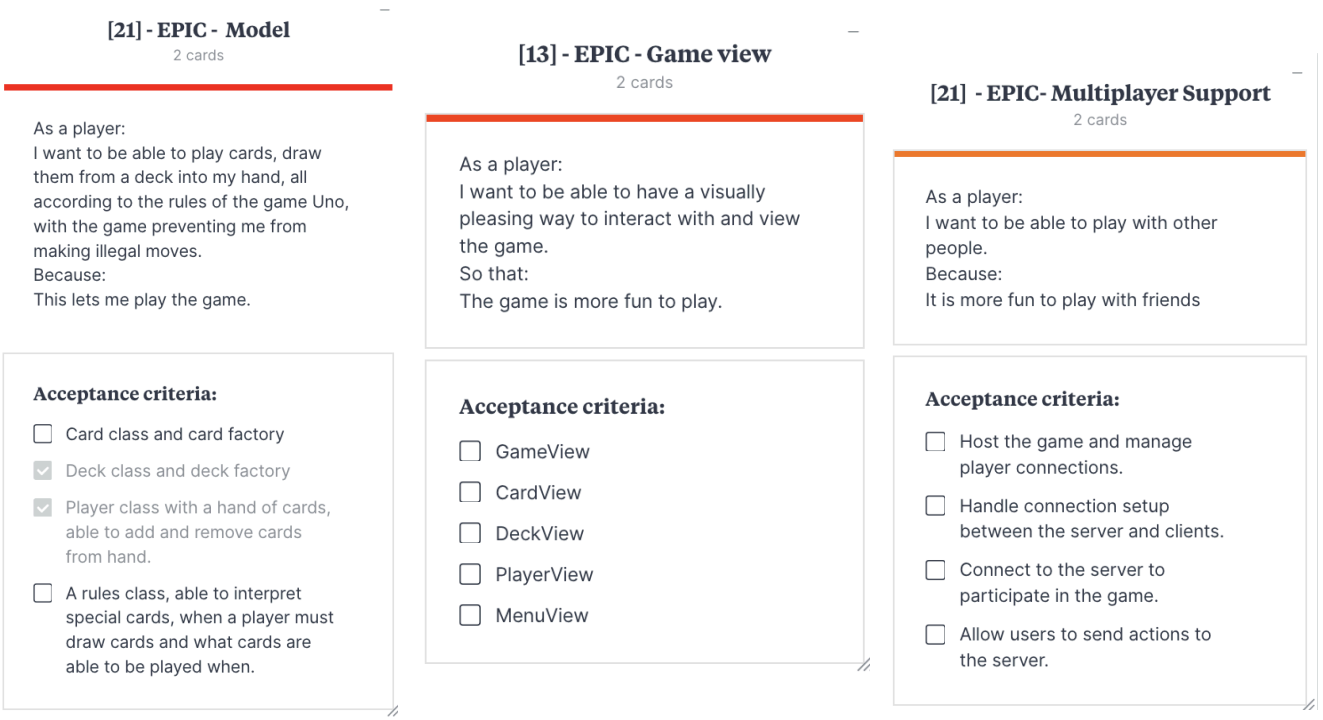
The Lobby View handles multiplayer setup, allowing players to join a room and see who else is in the game. When all players have joined, pressing the play button starts the game.

Once the game begins, the Game View becomes the main interface, showing the current game state, player hands, and actions. In the central area a discard pile and draw pile is displayed. Only the player's cards are visible. There is also going to be a turn indicator and animations for example a hover animation when choosing cards to play.

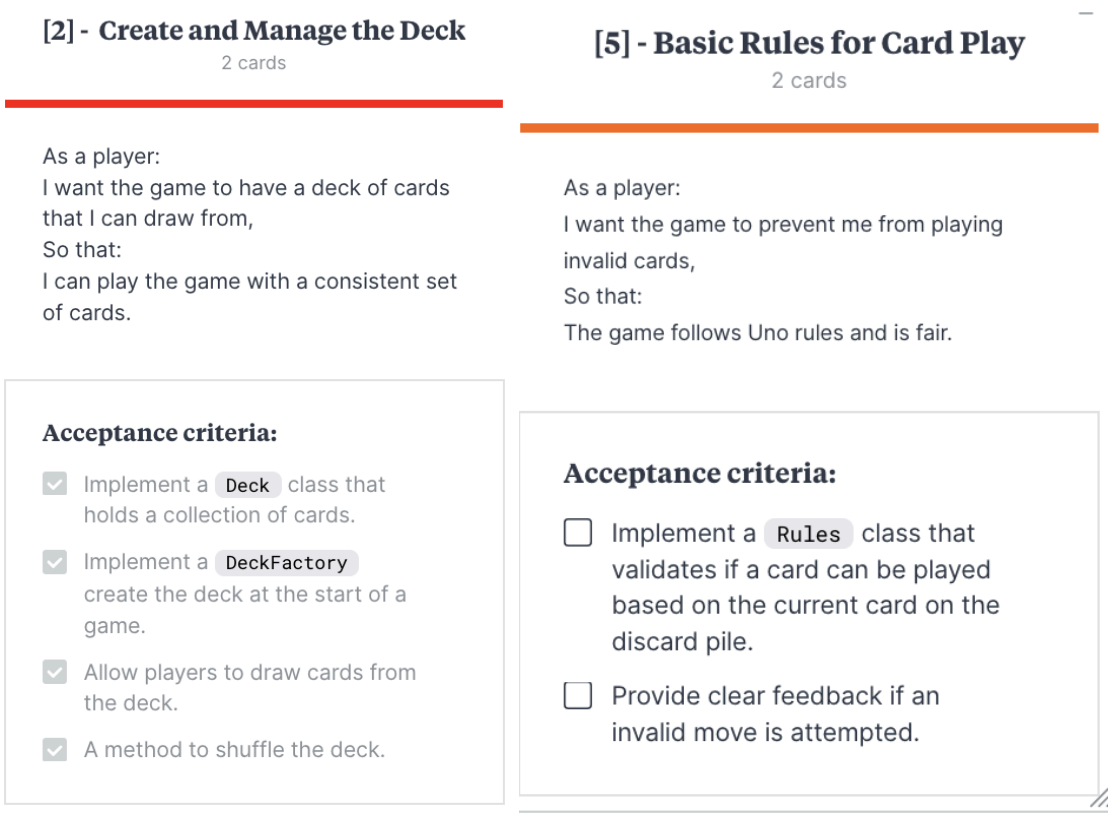
User stories

We have three epics that address core aspects of the game. One for the model, view and multiplayer support.

The goal of the epic centered around the model is to ensure the game mechanics function according to the rules of UNO. The view epic focuses on creating a visually engaging and intuitive interface for interacting with the game. Lastly, the multiplayer epic aims to enable players to connect and enjoy UNO with friends in a multiplayer environment.



The epics are divided into user stories that focus on specific features or functionality that needs to be implemented. For example, the model epic is divided into 5 stories. There is one for managing the deck and one for rules when playing cards, shown below.



Stages/DoD

Our team follows a structured approach to complete user stories, divided into five stages:

1. Stories: Everything we want to implement
2. Backlog: higher priority stories we plan to start implementing
3. In progress: working on the implementation.
4. In review: reviewing and testing the finished implementation.
5. Completed: Stories are marked as complete once acceptance criteria are met, and the implementation has been tested either by users or by passing all automated/manual tests

We consider a user story done when all of the acceptance criteria has been met and the implementation passes all tests, either by unit tests or by a user testing by playing the game.

To keep track of our process we use a kanban board, we estimate required time and complexity using the Fibonacci sequence and colour the cards to show urgency/importance with red being the highest priority.

Domain model

The domain model consists of three entities: Deck, Player and Game. The Deck contains a collection of cards, the Player holds a hand of cards, and the Game manages the overall gameplay, including player interactions and the state of the game. There is a value object called Card, which represents the individual playing cards in the game. Three service objects , Rules, GameLogic and Model, are responsible for handling the operational logic of the game. The Model is also responsible for initializing the game. Two factories, CardFactory and DeckFactory, are responsible for creating instances of cards and the deck.