Writers' Bloc Developer Documentation
Team 6 - CS 4911 - Spring 2014

1. Architecture

Much of the architecture was dictated by Meteor, since it is intended to serve as a full-stack framework. We use a client-server model, but it follows a model-view-viewmodel design pattern, as opposed to the traditional model-view-controller pattern. The model on the server updates the database and determines which subset of the database a client has access to, but from there, the viewmodel on the client holds its own version of the database, based on what the server publishes, and the logic used to display information in the view is handled here, rather than on the server. User interactions with the view cause events to fire, which are received by the viewmodel to trigger a server call.

2. Detailed Design

The user sees an html file that simultaneously contains all of the screens. These screens are stored in templates, which are javascript objects that have corresponding html blocks. Every template that represents a screen has a function that determines whether or not it is visible, based on the user's current session. These templates also contain events, which are usually triggered by a button click. Information regarding a user's current screen is saved on the server, so that the client knows that it is always in the proper state. Any template that displays data dynamically will also have helper functions, which retrieve information from either the session or from the Minimongo database and return a value that depends on that information. The html will contain Handlebars.js tags that display the returned value in the html.

3. Data Storage

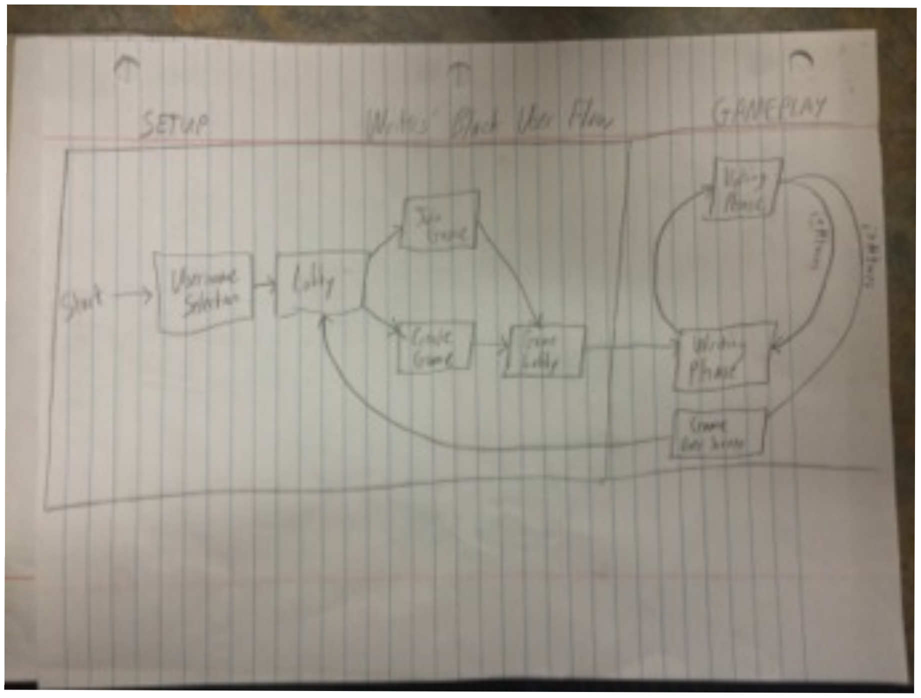
We have three MongoDB collections used to store games, players, sentences, and messages. These collections are stored on the server, but each client has a Minimongo version of each collection, which contains only the elements that the server publishes to them. Documents in the game collection contain an id, a game name, a current number of players, a max number of players, the current round number, the total number of rounds, and the first, last, and user-submitted sentences.

Players have an id, a name, a score, a game state (telling which screen they are currently on), a game id (for which game they are currently playing), a boolean marking them as the game host or not, a boolean marking whether or not they have played their appropriate action yet, and a boolean that marks whether or not they voted (as non-voters do not receive points). Sentences have an id, a game id (which game they belong to), a player id (which player submitted them), a round number (which round they were submitted on), the text that was submitted, the number of votes, and a random number used to display them in a random order on the ballot. Finally, messages simply contain an id, the id of the game they belong to, the name of the player who submitted them, the message text, and the date on which they were submitted.

4. User Interface

User flow, as seen in the figure below, is kept simple by design. There are only eight screens accessible to users, five during the setup phase (username selection, main menu, join game screen, create game screen, and game lobby) and three during the gameplay phase (writing phase, voting phase, and game over screen). Setup phase screens tend to utilize linear flow, while the gameplay phase tends to be cyclical. This makes sense as joining or creating a game should not require backtracking through pages, while the gameplay phase is designed to be cyclic, swapping between two different gameplay states for a set number of turns, and then providing users to a means of returning to the beginning of the setup phase upon game completion.

User flow in *Writers' Bloc* is not complex because it doesn't have to be. There are only two classes of user: hosts (those who create a game) and guests (those who join a game). There is only one branch in user flow, choosing to host a game or join a game, and these branches converge once the game has begun, as there is no functional gameplay difference between hosts and guests. The only difference between hosts and guests is that hosts are given the opportunity to configure a game and start their own game early, while guests have the ability to choose from multiple games from the game list.



As far as individual screen design goes, our prototype, implemented as a Facebook app using Google App Engine, served as a staging ground for UI design and experimentation. The *Writers' Bloc* prototype a heavy art-asset heavy approach, including a main menu that features warm colors and a background of an opened book, encouraging a linear book metaphor of flow.

Like a book's cover, the main menu features relatively few words, with wordier pages (and the word-based game itself, of course) appearing later in the user flow. Our final version, implemented as a standalone app using Meteor, largely abandons this notion due to early play tester concerns about loading times (as the asset-heavy approach is certainly more data intensive). Testers also noted that they were not sure that a button did based on its picture alone, leading to the insertion of words to the side of each button that, frankly, removes the point of associating images with buttons. Given the recent trend towards minimalist web designs, the Meteor version uses a more succinct text-based approach. Easier for users to understand (text explanations make the buttons in the main menu less ambiguous, ensuring players don't choose the wrong button by accident) and less data-intensive, the final version abandons the prototype's book metaphor in exchange for succinctness and a less-cluttered interface, a decision that we believe to be a favorable trade-off.

The title screen is straightforward, allowing both hosts and guests to begin their respective pre-game goals. Hosts encounter a game setting screen that allows them to change the game titles, the starting/ending sentences, number of turns, and number of players. Default values are already inserted in each category, giving players a good idea of what an average game's settings might look like, allowing them to make more informed setup decisions. Meanwhile, guests are given access to a game list, where they can see relevant game information (the starting sentence, number of players, and number of turns), allowing guests to also make an informed decision and only join games that suit their tastes or their schedule.

The gameplay screen itself is largely unchanged from the prototype's implementation, keeping a window-based approach that helps segregate different aspects of the game. Each window is associated with a different action that user may do. For instance, the user may *read the story so far*, *check his score*, *or chat with other players* using the three base windows. At the center of it all is the core gameplay mechanic, sentence submission. Overall, this modular approach keeps information organized and allows for the user to quickly establish a cognitive map of where the information he's looking for should be. In fact, the only time where a window's function changes is when the voting phase's interface replaces the chat window, an intentional design decision that seeks to reduce the ability for players to sway other players' voting decisions. The look and feel of the voting panel is vastly different from the chat interface, ensuring that users will not confuse one for another and will understand fairly quickly that chat is unavailable while voting.

Finally, notification strings are printed at the top of the screen near the game's logo. All game timers (including time left for an individual phase as well as number of rounds remaining), appear at the top and changes color when the game is almost over, providing a visual cue for players to be mindful of remaining time. Other game information (such as a notification of the bonus round beginning) also appear at the top of the screen in red, increasing the chance of users noticing it. Although we briefly explored using audio, we decided against it, opting to be as non-invasive as possible, especially in light of playtesting that suggested that most players noticed these cues and did not need additional prompting. Once the game is over, the sentence submission interface is replaced with a button allowing users to return to the main menu. The substance submission interface was chosen to be replaced as it is the only panel that no longer serves a function (as players will still want to use the other panels to chat, read the story they made, or view their final score). The button returns the user to the main menu, allowing them to begin anew. In summary, the interface, while certainly more simplistic than the original prototype, loses no features and streamlines the experience. The actual gameplay interface has received a minor visual overhaul, but is functionally equal and is easier on the eyes, a core requirement for a text-based game.