Teamwork Plan

1. **Stakeholders:**
    a. People who like word games
    b. **(Added in revision)** Potential competitors: game web sites who would share our traffic, and would possibly try to imitate our product if they saw we were successful
    c. **(Added in revision)** People interested in surveying public interest in word games: e.g., the makers of Bananagrams may decide to rebrand or release new variants of their game if they see that our online version of Anagrams has become popular
2. **Resources:**
    a. Computational: We'll need a server to run the game. We'll use Heroku.
    b. Cost: none
    c. Time: 6.170 has due dates, and team members have other classes and things to do. The week before Thanksgiving break is a big opportunity since our availability is higher than usual that week.
3. **Tasks:**
    a. Design (during the preliminary/MVP phase; during the revised/final phase):
        i. Purpose and goals
            1. Prelim: Damien
            2. Revised: Leon, Bethany
        ii. Context diagram
            1. Prelim: Leon
            2. Revised: no changes needed
        iii. Key concepts
            1. Prelim: Leon
            2. Revised: Leon, Damien
        iv. Data model
            1. Prelim: Ben
            2. Revised: Ben (with much discussion by all)
        v. Feature descriptions
            1. Prelim: Damien/Leon
            2. Revised: Leon/Damien
        vi. Security:
            1. Prelim: Damien
            2. Revised: Leon
        vii. User interface:
            1. Prelim: Bethany
            2. Revised: Bethany
        viii. Design challenges:
            1. Prelim: All
            2. Revised: All

b. Implementation (main; secondary)

i. Database schema (Ben)

1. Not very much effort. Consists of creating a skeleton of each Rails model class, so that data fields and associations are laid out.

ii. UI (Bethany; Leon)

1. Very involved, on both the planning/design and coding fronts. Includes making images of game and lobby components, building views in HTML and JS, and styling with CSS.

iii. Game model (Ben; All)

1. One of the most involved parts. Includes the logic of the game rules, and all computations for creating and modifying game state.

iv. Signup and accounts (Bethany; Ben)

1. Not much effort, mostly built into Rails, just need to make views for it.

v. Networking (Damien; Ben)

1. Could be a significant effort. Will need to either get websockets working, or determine some other websocket-like medium (e.g. long pull), or change the rules of the game to eliminate time-dependence.

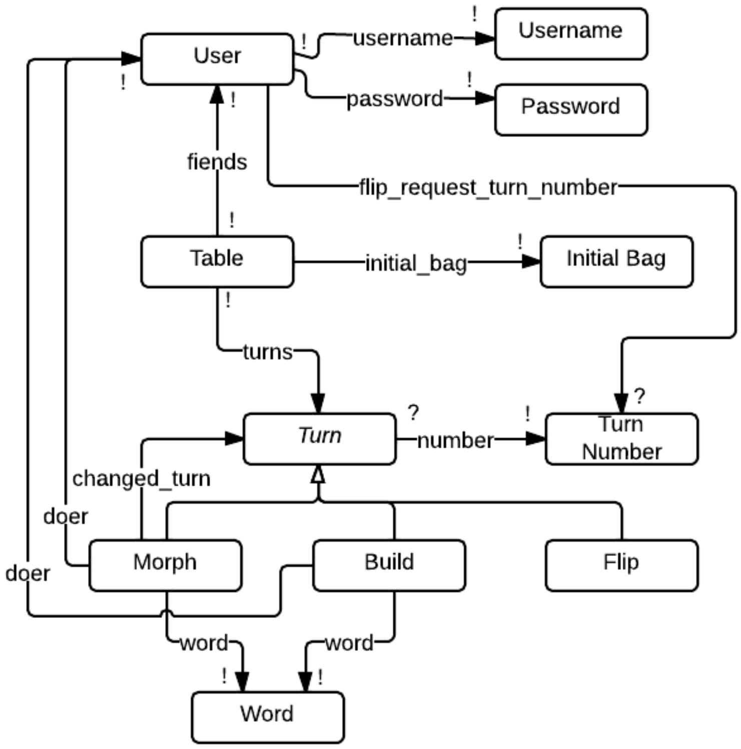
vi. Team management (Leon)

1. Having someone explicitly think about how to allocate effort and assign tasks for optimal productivity is quite helpful. Should be done continuously, though doing another task at the same time as this one is (usually) possible.

Minimum Viable Product

One-player games only. Just the gameplay component (no lobby).

1. **Feature descriptions**
    a. **Play Anagrams by yourself.** Just sign up, and you can play Anagrams in single-player mode!
    b. **(Revised: cut out match history feature.)**
2. **Issues postponed:**
    a. Multi-player games.
    b. Lobby, including "online status" and the process of challenging another user / accepting challenges within the lobby.
    c. Ranking system.
    d. Match history.
    e. Viewing and sharing replays.
3. **Reduced object model diagram (revised version)**



Extra constraints:
In the relation changed_turn: Morph -> Turn, the target Turn cannot be a Flip.

Notes:
Same as the notes for the full object model.

Revisions from original reduced object model:

- User now has a flip_request_turn_number field, which is the same as the one explained in our revised design doc. (The functionality is trivial in the MVP, though, as in single-player mode all flip requests are granted immediately.)
- The high_score field is eliminated, since we did not implement high score lists in our MVP (though we did have a score display within the game).