Software Requirements Specification

Dota Track

Michael Eaton, Hazen Johnson, Sarah Witty

September 18, 2014

Contents

1 Summary

1.1 Client Problem

Modern Dota statistics websites do not provide historical trends in player performance. This requires players interested in analyzing this data to do complicated and unnecessary work. Additionally, it is currently impossible to compare any given player's performance with other players or the Dota community as a whole.

1.2 Goal

Our client, the LeTourneau Moba Club, would like an application that provides the standard Dota statistics while addressing the shortcomings mentioned above. **Dota Track** will enable a user to:

- View recent matches
- Enable users to tag matches with relevant match specific information
- Look at personal trends through various filtering match tags (hero, role, etc.)
- Make comparisons with others in the user base

Ultimately, **Dota Track** will allow them to become better players by highlighting areas that need improvement.

1.3 Proposal

We will create a responsive (mobile-enabled) web application that meets the goal as outlined above. The application will take match history from the Dota 2 WebAPI to construct relevant statistics. These statistics will be stored in a server-side database and displayed in graphical form on the user's device. The application should be built with mobile devices in mind so that players may easily access the provided information while playing the game.

2 Requirements

2.1 Functional

2.1.1 API Interaction

Pulls player and match data from Dota 2 WebAPI.

- Allows player to register to have their data pulled
- Match information pulled from the server includes:
  - Matchmaking level
  - Result
  - Match time
  - Game mode
  - Server region
  - Date
  - Player information
    - Level
    - Hero chosen
    - Kills
    - Deaths
    - Assisted kills
    - Last-hits
    - Denies
    - Experience per minute
    - Gold per minute
    - Hero damage dealt
    - Tower damage dealt
    - Items purchased
    - Skill build
- Stores this information in the database
- When a player logs into the application, all their matches are automatically pulled into the database
- Average statistics should be recalculated at least every hour. These include:

  - Kills
  - Deaths
  - Assisted kills
  - Last-hits
  - Denies
  - Experience per minute
  - Gold per minute

2.1.2 Farm Priority Tags

Allows users to browse and tag matches with farm priority.

- Tags are based on farm priority. (Pre-generated tags useful for categorizing matches. Values: 1, 2, 3, 4, 5)
- Tags will be associated with a hero in a match. A user may enter tags for any hero in a match he/she has access to (includes admins). Modifications should be stored as modifications.

2.1.3 Match Display

Displays all statistics about a specific match and each of the players in the match.

- Data as specified above (2.1.1) should be displayed for each match.
- Matches should be selectable from a central match summary page
- Users should be able to search matches by ID and hero.

2.1.4 Statistics Display

Displays useful statistics to the player based on the data gathered.

- Includes line/historical graphs to show player metrics
  - Match distribution (i.e. matchmaking level)
  - Kills
  - Deaths
  - Assisted kills
  - Last-hits

  - Denies
  - Experience per minute
  - Gold per minute
  - Win rate by match length
- The historical graphs should include historical data (from the past 6 weeks by default. If possible, allow alternative time ranges.)
- Alternative graphs (e.g. pie graphs) should display segmented data
  - Modes played
  - Win rate by faction
  - Position played (i.e. farm priority)
- All graphs may be filtered by hero and/or by farm priority.
- Graphs scale dynamically based on the data content (axis scale).
- Graphs should optionally display the averages from all registered users.
- Users should be able to enlarge graphs to see more detail.
- Graphs should offer helpful popups which help explain the statistics shown.
- Statistics from untagged matches should be included in unfiltered views (ones not requiring tags).

2.2 Non-functional

2.2.1 Performance and Reliability

- Because of its web based nature, the application will load in under 5 seconds.
- The close time will likewise be under 5 seconds.
- Acquiring match data (2.1.1) should take less than 0.03 second per match. A player, having played an range of 200-2000 games, logging in for the first time will experience a load time of no longer than 60 seconds.
- The application will make no more than 100,000 API calls per day.

- Graph generation will take no longer than 5 seconds apiece.
- The application will return correct results with an error tolerance of once every 3 months.
- The application will manage at least 100 users with 500 matches each, totaling 50,000 matches. This will consume no more than 1Gb.

2.2.2 Compatibility

- Should work on Android, iOS, and Windows phone devices (any device with a modern (HTML5) mobile web browser).
- User interface should scale responsively for whatever device it is on with at least two different modes:
  - Mobile (Landscape)
  - Mobile (Portrait)
- User interface will be compatible with both touch screens and traditional mouse and keyboard inputs.
- Interacts directly with the Steam API (2.1.1).

2.2.3 Security

Because users register through the Steam servers, registration and user information storage will be secure. This can be ensured through the use of Steam Guard. The application will meet Steam's security requirements for third-party websites.

2.3 Use Cases

2.3.1 Case 1

- User logs in with Steam credentials.
- Application presents a list of all the user's matches (including the most recent matches).
- User chooses a match from the list.
- Application presents the basic statistics for the match.
- User selects the role he played in this match.

- User selects a graph.
- Application displays a larger, more elaborate? version of the graph.
- Confused, the user clicks the question mark to figure out what Matchmaking-Rank is.
- Application gives a quick explanation of what Matchmaking-Rank is and how the graph displays it.
- With new-found understanding, the user improves his game-play.

2.3.2 Case 2

- User logs in with Steam credentials.
- Application presents a list of all the user's matches.
- User selects the Statistics view to view his statistics over time.
- Application presents a summary of all statistics over the past 6 weeks.
- User selects a greater time range to see his overall statistics over time.
- Application presents a summary of all statistics for all the players matches.
- User selects a filter for the application so that only matches the user played as the hero Puck are considered.
- Application presents a summary of all statistics for all matches played as Puck.
- User selects a role filter to see all matches played as farm rank 1.
- Application presents a summary of all statistics for matches played as Puck with farm rank 1.
- Having looked at these statistics, the user improves his game-play as Puck.

2.4 Prototype

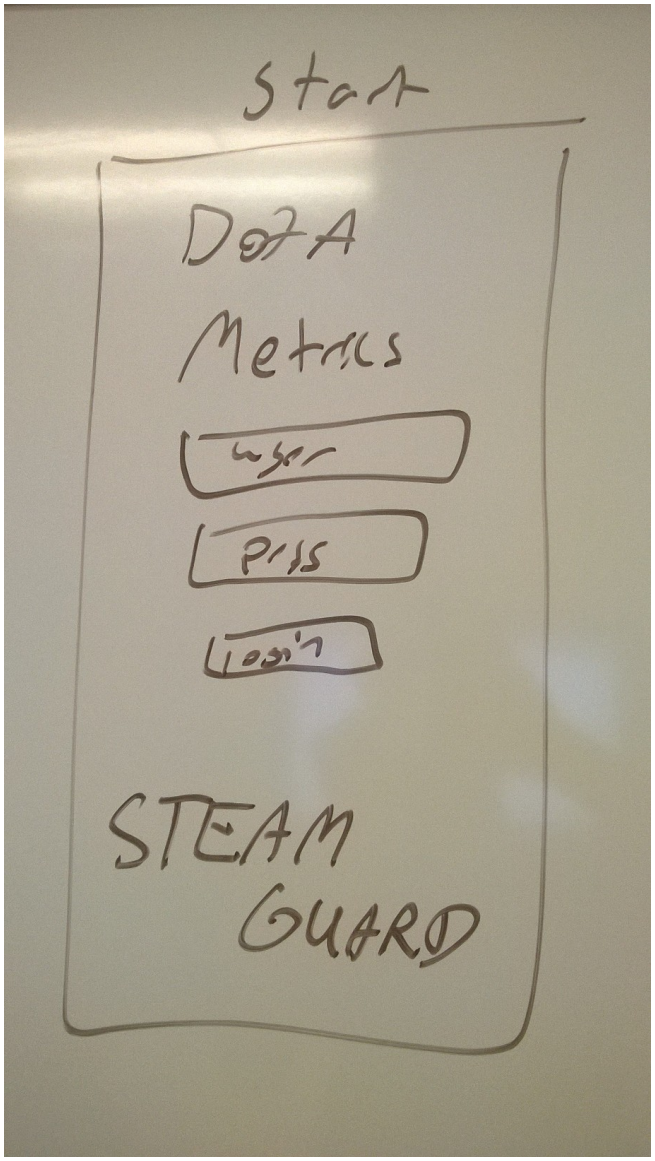

Figure 1: Login page

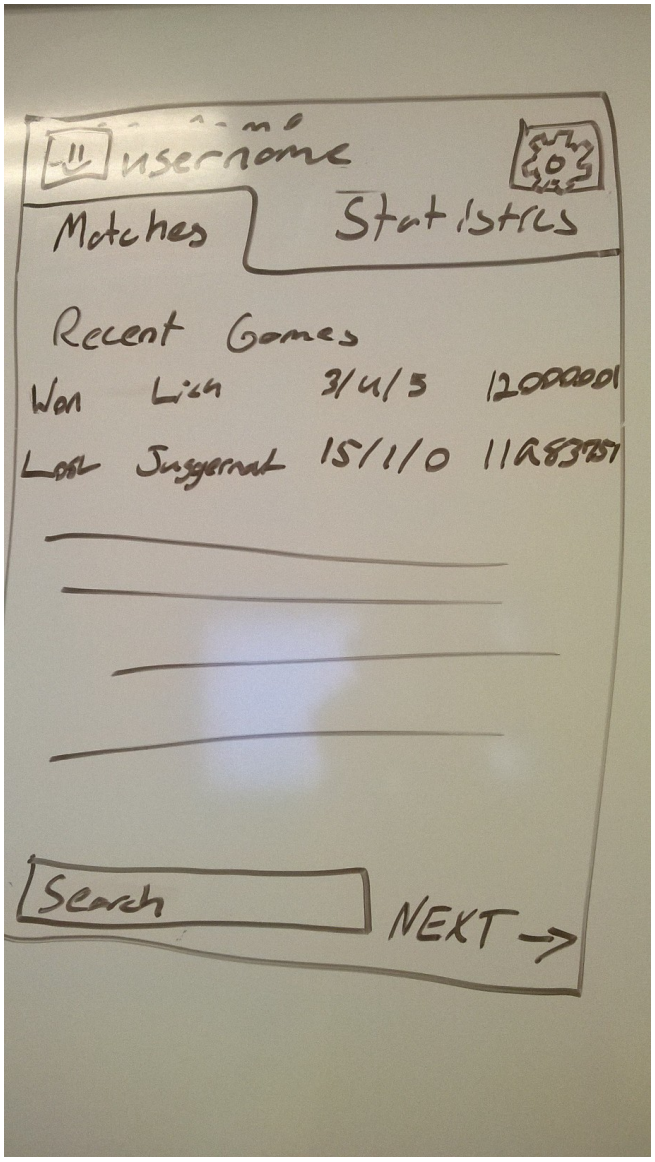

Figure 2: Main page (Matches)

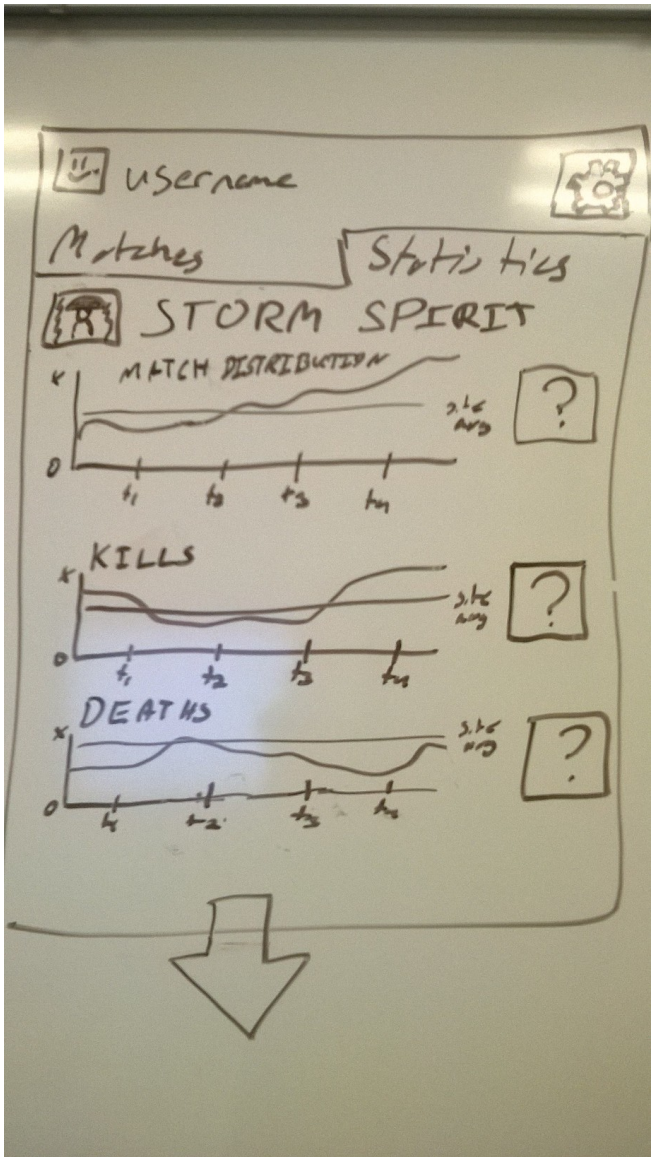

Figure 3: Main page (Statistics)

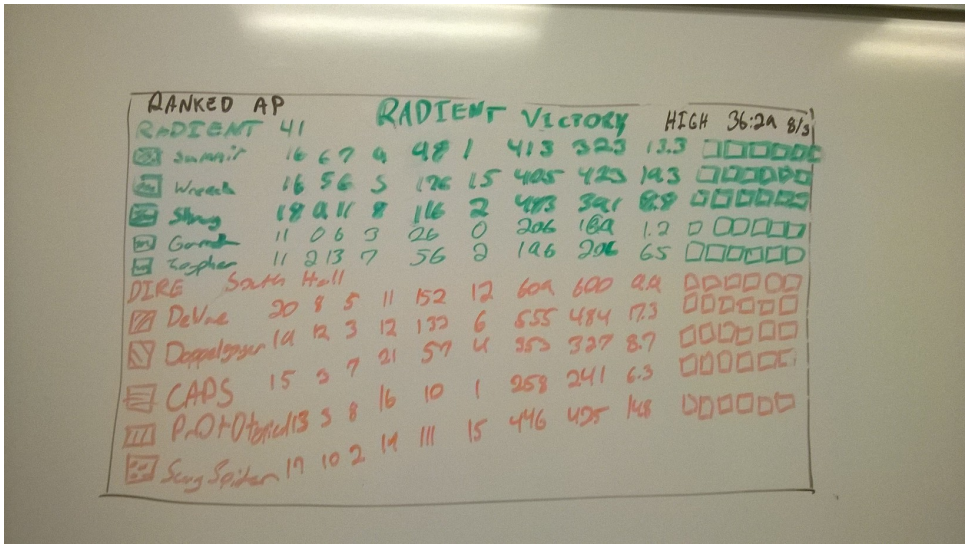

Figure 4: Match page

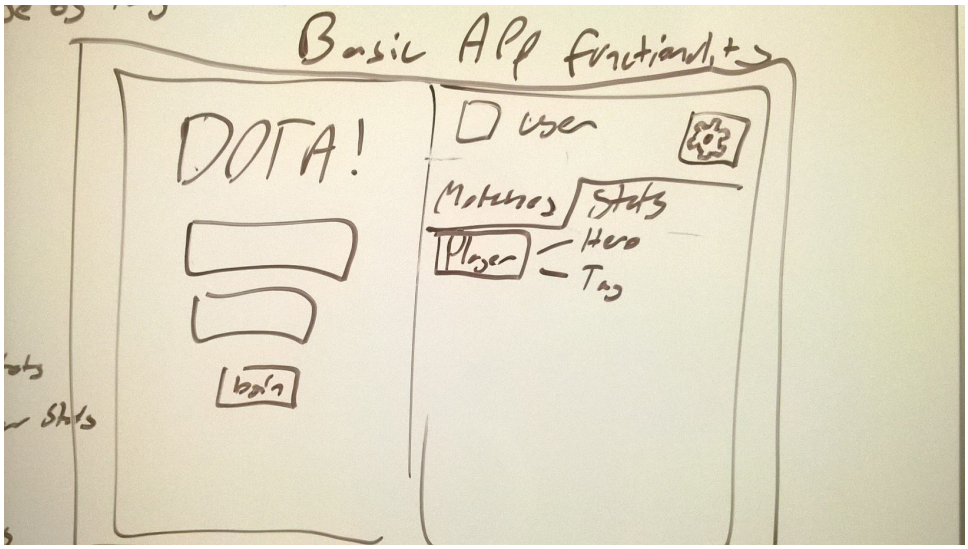

Figure 5: Basic functionality

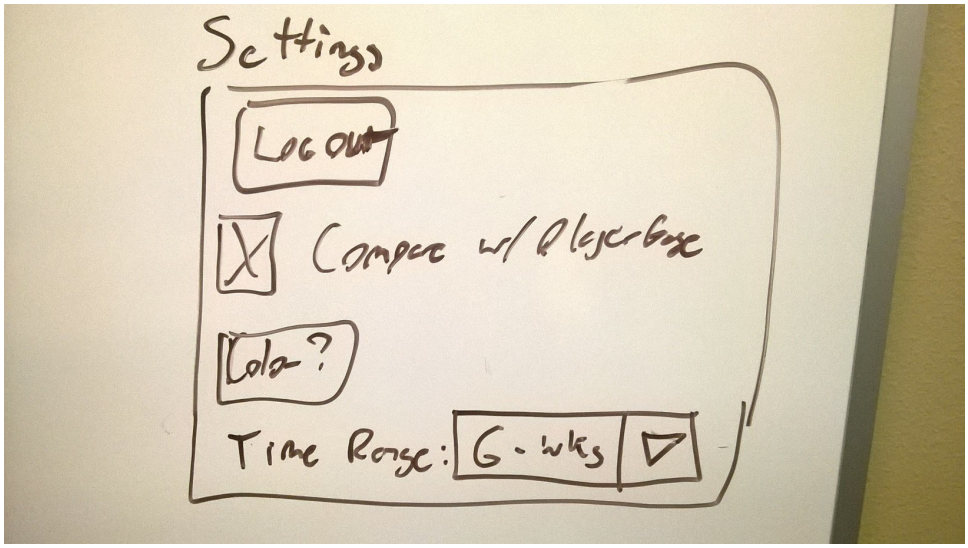

Figure 6: Settings

3 Architecture

3.1 Major Components

- Steam API - This component pulls the data from the Steam servers.
- Database - This component stores the match data and derived data obtained from the Steam API.
- Server Controller - This component functions as the go between for the user, the database, and the Steam API.
- User Views - This component displays the data to the general public in a user friendly format.

3.2 Database Design

This application will have the following entities

- Match
- Player

Player specific match data (2.1.1) will be attached to the relationship between matches and players.

3.3 User Interface Design

- The opening screen should allow users to sign into Steam.
- The main page will allow users to quickly select specific matches or view aggregated statistics.
  - Minimal information about available matches should be displayed on an endless scrolling pane (disregarding Figure 2).
  - Average statistics should be default until a filter is selected.
- When viewing an individual match, users will be presented with a match summary page
  - This page will allow the user to select their farming role (2.1.2).
- Clicking on the button next to a graph zooms in for a more detailed view.

- Default view orientations for each screen should be determined by the previous view orientation.

4 Client Interaction Plan

We will maintain some form of contact with our client at least once per week.

If any unexpected requirements issues are encountered, we will immediately contact our client via email.

We will respond to client requests/interactions no later than 12 hours after receipt.

5 Verification and Validation Plan

5.1 Development Verification and Validation

5.1.1 Requirements

- Each team member will proof-read the requirements for mistakes and/or possible confusions.
- The requirements should be presented to the customer for approval to validate.
- Requirements will be submitted to Dr. Baas for final approval.

5.1.2 Design

- Each team member will proof-read the design, checking that all requirements are adequately met by the proposed design.
- The design document should be presented to the customer for approval to validate.
- Design document will be submitted to Dr. Baas for final approval.

5.1.3 Implementation

This process will be repeated for every iteration until implementation is complete.

- Every iteration, the team will meet to do a basic code review and to update reports (5.2).
- Every iteration's results will be presented to the customer for approval.
- Each iteration, milestone completion will be evaluated and assignment/schedule changes will be made as necessary.

5.1.4 Testing

- Product will be presented to the customer (and other selected test users) to ensure the product meets their needs.
- Product will be validated against the non-functional requirements.

5.2 Reporting

- We will use a Kanban chart to keep track of development progress.
- We will use a variety of a Burndown chart to keep track of the amount of time each task takes in comparison to the estimated time.

5.3 Defect Resolution

Defects will be resolved in a dynamic manner by incorporating the resolution into the next iteration. Defects will be resolved by the developer concerned with that part of the application.

6 Schedule Plan

1. Pull Data from API (API)
    - Make requests to WebAPI
    - Limit requests to fit specifications
    - Format data if needed by database
2. Plan and Build Database (Database)
    - Plan database
    - Build database
    - Populate with test data (possibly from the WebAPI)
    - Interaction between API and database functional
3. Display a Match (Application)
    - Add basic navigation
    - Login to Steam
    - Include matches tab
    - Include match page

- Populate match page using the database
- Settings menu

4. Display Player Stats (Application)
    - Add basic navigation
    - Login to Steam
    - Include stats tab
    - Help buttons
    - Populate graphs using the database
    - Zoom support for graphs
    - Settings menu
5. Parse Stats by Hero (Application)
    - Select hero
    - Database query filters appropriately
    - Stats page responds to new data
6. Tag Matches, Parse by Tag (Application)
    - Can add tags to heroes in matches
    - Filter statistics by tag
    - Database query filters appropriately
    - Stats page responds to new data