**EPL Centennial Website**
**CMPUT 401 – Winter 2013**

# Project Future Outlook

## Table of Contents



# Deferred Features and Known Defects

- **Support for IE8:** Our application does not currently work with IE8. During development we tested regularly with Chrome, Firefox, and IE10, as these are the browsers we had installed on our development machines (unfortunately, IE8 cannot be installed on recent versions of Windows). When we finally obtained a computer with IE8 to test our application on, we found that some errors specific to IE8 prevented the website from working. We were unable to fix these errors by our submission deadline, but it should be possible to address these errors in the future.
  Priority: High

- **Add Terms and Conditions for account creation:** When a user creates a new Centennial account, they must agree to the Terms and Conditions. There is a link to the Terms and Conditions on the Create Account form, but this document only contains placeholder text. EPL will need to determine what the appropriate Terms and Conditions are, and add this content to the Terms and Conditions document in the application. This file can be found in the source code under /static/views/termsAndConditions.html.
  Priority: High

- **"Launch" screen for the TimeMap:** The design mockups contained an image of a "Launch" screen for the TimeMap that would appear on the EPL website. The screen would contain a button that the user would click to open the TimeMap website. We did not develop this screen ourselves because we did not know exactly how EPL would link to the TimeMap - there will probably be multiple different links on separate pages on the EPL website. We have left this screen out to allow EPL the decision for how they would like to open the TimeMap. Priority: High

- **Quests based on Google+ social media:** Currently there are social media quests based on Twitter and Facebook social media components. For example, the user can

complete a quest that requires them to share a TimeMap story on Twitter, or share a story on Facebook. While we have a Google+ Share button on the TimeMap, there is currently no functionality that connects Google+ sharing to a quest. Unfortunately, it was not as easy to "hook" our quests into the Google+ social widget as it was for Twitter and Facebook widgets, so quests based on Google+ sharing were deferred for now.
Priority: Medium

- **Advanced search based on branches:** The search functionality on the TimeMap (in the "Search" tab on the right of the screen) has options such as Year, Keywords, and Content Type. Currently this search is global across all branches. It may be useful to add another search parameter to select a specific branch to search under. This would become more useful as the number of stories associated wtih each branch increases.
  Priority: Medium

- **No built-in contest or prize support:** The game component of the website keeps track of the user's points and levels. As the user completes quests and gains more points, their level in the game will increase. But, as it stands, there is no "point" to the points or levels in the game. EPL talked about having prizes based on the points, but this is not currently referenced anywhere on the website. Having a prize or contest would further engage the public into playing the game.
  Priority: Medium

- **Create a "leaderboard" of high scores:** In the game component of the website, the user is shown their points and their level, but they do not see the points about any other user. Users cannot compare their own level against other users, and as a result, they would not be as motivated to complete more quests. Having a "leaderboard" of users with the most points would increase the level of engagement in the game. This was not in our requirements, but we feel it is important for the game's success in the future.
  Priority: Medium

- **Social interaction with the game:** We have added social media components to the TimeMap, but we did not have time to add social media components to the game. The user should be able to share the game on Facebook or Twitter, to increase the visibility of the game to other users.
  Priority: Medium

- **Reset password:** A user has the ability to change their password once they are logged in, but the user cannot request a password reset if they forget it. In this case, the user would need to create a new account. This is a low priority because it only applies to Centennial accounts (passwords for EPL accounts are managed through the EPL site).
  Priority: Low

- **Facebook Logins:** Users can either login with their EPL account or with their Centennial account. An initial requirement was to also support Facebook logins, but this was a lower priority because the vast majority of users will use their EPL account logins. We also wanted to encourage the user to use their EPL login (rather than Facebook

logins) so that they could participate in the catalog-based quests in the game component.
Priority: Low

- **Add additional supported video types:** As per our requirements, the TimeMap only supports playing MP4 video files. This may be a challenge if EPL has a video in another format that they want to display on the site.
  Priority: Low (unless needed)

- **Pinterest social media** We have social media sharing with Twitter, Facebook, and Google+. Another possibility for a social media component is Pinterest (particularly for image-based stories on the TimeMap). We did not add Pinterest yet because we felt that the other social media components were much more popular - but Pinterest is something that could be added in the future to expand the social media interaction.
  Priority: Low

- **Duplicate emails allowed in accounts:** Right now the Centennial accounts enforce the uniqueness of usernames when accounts are created. It would be additionally beneficial to validate that the user's email is unique across accounts as well. Adding this check would prevent a user from reusing emails in multiple accounts (and reduce the possibility of spam account creation). This is a low priority because most users will not be using Centennial accounts.
  Priority: Low

# Architecture

## Strengths

- **Strong division between the backend and the frontend components of the application:** The backend is essentially a black box that exposes data through API methods. The front end can use these API calls to obtain the needed data without knowing anything about the internal representation of the backend code.

- **JSON Data:** There is a clear and consistent method of communication between the backend and the frontend, through JSON data. JSON data is a simple and easy method of data exchange for both the client and server components of the application.

- **Administration functionality:** Using the administration functionality of Django provides a very robust interface for EPL staff to use to manage the site's data. Using this framework saved us a lot of time because we didn't have to manually create all of the administration screens for creating quests or TimeMap data ourselves. In addition, if the data models change in the future (for example, a new quest type is added), it will be relatively simple to update these interface screens with the new changes.

- **Knockout JavaScript library for the front-end screens:** Knockout is a great library

for managing the data that is displayed on the HTML pages. If another development team takes over our project, we recommend that new developers complete the Knockout JS tutorial (http://learn.knockoutjs.com/) as it is an excellent resource that explains the basics of all of our front-end UI.

- **Single-page applications:** Using a single dynamic page reduces loading times significantly. The user will load the page logic once, and simply execute it for remaining interactions. It also creates fluid navigation that facilitates users' expectations of instant feedback. In addition, using a single page maintains the user's state on the site. For example, the users' timeline interactions are maintained throughout timemap navigation.

- **Using LESS for CSS styling:** LESS is a pre-compiler for CSS that allows nesting, variables, and functions. This is advantageous because, for example, we could define the background color in a variable that could be reused in many locations in the CSS.

- **Mobile device integration:** Our application works great on mobile devices, including interaction with the timeline.

## Weaknesses

- **Extremely complex framework for the front-end:** There are many dependencies between Javascript libraries that result in a complicated setup for the front-end. Unfortunately, this was difficult to avoid with web development.

- **Timing issues with the front-end components:** Due to the nature of our JavaScript-heavy front end, there are often components that need to be loaded in a particular order. Because these components are often loaded from servers with variable response times, there is the possibility for an incorrect loading order. This was visible for our social media components - the Facebook script had to load last after the rest of the page had loaded, but often it would be loaded first. While we fixed most of these problems with the current code, there is a strong possibility that this problem will arise again if there is new code added in the future.

- **Tight coupling between the TimeMap and the game components:** It would not be straightforward to easily take out the game from the TimeMap, if EPL decided to only use the TimeMap by itself. A better architectural approach would have been to contain the quest-based code within a module in the TimeMap code.

- **Duplicate code between the TimeMap and the HYQ components:** For example, the common menu at top (containing the login and account management links) is duplicated in both files.

## Future

Aside the from the features listed above, there are a few general goals that future developers should keep in mind:

- Maintain a strong division between the back end and the front end. Essentially, each component should not need to know the inner working of other components. They should communicate only through an API using a consistent method of data exchange (JSON). APIs should be well documented so that the exposed functionality is clearly visible to other developers who may use the API.

- Maintain a single-page dynamic application where possible. We feel that the fluid approach to this website results in a much better user-experience when transitioning between components of the website. Just make sure that the website navigation continues to meet standard browsing expectations (using the back button goes back, being able to bookmark a specific page, etc).

- Styling could be greatly improved. We did not have a great depth of CSS experience on our team, so our current styling on the website is mediocre. We feel like the website would benefit from a significant styling overhaul.

# Lessons Learned

- **Do Unit Testing:** We often view Unit Tests as tedious and time consuming. However, Oscar did find a bug or two by writing Unit Tests for his back end TimeMap component - proving that Unit Testing is an important testing step that should not be overlooked.

- **Do cross-browser testing continuously and early:** We tested our application regularly with modern browsers throughout our development process (Chrome, Firefox, IE10). Quite often we found inconsistencies between browsers that we would need to address. Unfortunately, we were not able to test our application on IE8 until quite late in the development process (only once we had set up our site on an external server). We found that there were some errors with IE8 that we were not able to fix by the submission deadline. We learned that, in the future, it will be essential to test on all required browsers continuously throughout the development process.

- **Create more realistic milestone goals:** With such a large project and only a short amount of time to complete it, it was difficult to plan realistic milestones. We did not follow our milestones very closely and we quickly fell behind. It was quite difficult to plan accurate milestones because many people in our group were new to web development and there was a big learning curve. Everyone in the group probably learned a great deal about planning and time management. If we were to do another project together, we would be able to more accurately plan our milestones.

- **Split up tasks based on area of expertise (when possible):** Evidently, it makes sense to assign tasks to the person who is most experienced with that technology.

However, our group also learned that sometimes the tasks cannot be split up so easily. Our application is a very interactive and visual application, but only three out of seven members of the group had any experience with front end web application development. The last couple days of development on the project were challenging because there was mostly styling or polishing tasks left to be done, but only a few members of the group were capable of completing this work. It would have been easier if another member of two learned some of the front-end technologies earlier in the development process.

- **Create documentation continuously:** As mentioned in the "Strengths" section above, one of the greatest advantages of our application was that we had a very clear separation between the front end and back end components of the website (the two components communicated through an API). We clearly documented these APIs on our wiki, so that a developer who was working on a front end components could easily look up the API documentation and quickly obtain information about the API interface.

- **Automating tasks is useful:** Whenever possible, we automated tasks in our development process. For example, we created a script for resetting our database to a consistent state. We also had a script that would upload our IRC chat history to the Git wiki. Using these processes saved a lot of time and facilitation rapid development.

- **Having focused, goal-driven meetings is helpful:** At the beginning of our project, our group would have long meetings where we did not accomplish a lot. We learned that it is much more productive to have focused Scrum meetings where everyone talks about what they accomplished the previous week, and what they're going to accomplish in the following week. This ensures that everyone has work to do and is on task. It also provides a mechanism for individuals to ask for help if they're struggling on a problem. Having a time limit (15 minutes) ensures that the meeting does not drag on unproductively.

- **Working together increases productivity:** Our group met to work on the project regularly (usually three times a week - Tuesday, Friday, and Saturday). These meetings were usually very productive, as we could directly help each other with problems. It also enhanced the level of communication across the group, because many members were present when we had a discussion about an issue.