Progress Update

Project: *Scrape*

Prepared by: Jack Cummings
Due Date: 11/15/2019

Scrape
Milestone 5

Current Progress

Functional

- Upload capabilities have been fulfilled using the Dropzone.js library.
    - Drag-and-drop, along with a file picker dialog, is supported through this library.
    - Additional form data can be attached to the file send request, which can then be parsed and inserted into the database by ASP.
    - Smaller resolution, height limited pictures (thumbnails) are generated automatically on upload and saved into the same folder with '*_thumb*' appended to the filename.
    - The use of AJAX (instead of relying on ASP.NET Update Panels) will be used for any further seamless web application interaction. The horror stories I could share from even my short time of working with them…
        - This is most definitely user-error on my part, the use of Triggers throughout a page, along with conditionally updating the Update Panels, would likely solve any issues I've had.
    - The *UploadPicture.aspx* has been optimized for ease-of-access.
        - The dropzone.js drag-and-drop area and the dropzone.js file picker dialog allow for multiple pictures at once to be selected.
        - The selection for which Gallery to upload the picture to allows for creation of a new gallery without leaving the page. This is provided for through a popup modal where a user can choose a title, description, and whether the gallery is public or private.
        - Selecting a picture that is queued for upload will allow a user to enter a name & description for the picture that will be saved into the database.
- A *very* basic user gallery viewing feature has been designed.
    - The user gallery shows a grid of pictures belonging in the gallery (using thumbnails).
    - Clicking on a picture will bring up a lightbox viewer provided through Photoswipe, a JavaScript library for viewing & navigating images in a desktop/mobile friendly way.
    - Pictures are loaded dynamically through ASP onto the page using basic HTML. Although ASP.NET's GridView component would've worked as well, it was proving difficult to style in a way I found acceptable.
- User pictures, galleries, and pictures have all been implemented, along with the required relationships and *many-to-many* foreign relationship tables. Though the implementation is not quite to the planned extent, it is operational.
    - Further work to add *tags*, and *comments* features will provide for a decently functioning gallery site.

Non-Functional

- Optimization in both performance & user flow has been heavily considered.
  - JavaScript loading & scripts have been deferred to the end of the *.aspx* files.
  - Thumbnails are generated for quicker loading in gallery views.
  - User-flow for uploading new pictures has been designed for efficiency & ease-of-use.
- Database procedures have been streamlined to not only process the updating of multiple tables at once, but to be optimized by returning only the necessary columns.

Future Optimization

- Plans for further optimization are also been planned, as time allows.
  - Making sure compression is enabled using gzip or perhaps even Brotli.
  - Switching from jQuery to Zeptojs/something similar that would be more lightweight and/or more modern.
  - Enabling caching of pages client-side.
  - Avoiding client-side redirects during gallery switching, user-not-logged-in redirects, and between any confirmation screens.
  - Possibly using tooling such as webpack or gulp.
    - Minification would be able to be easily accomplished.
    - Using tooling is a good practice as most real-world project have a tool chain one needs to adapt to.
- Measuring of SQL stored procedures, page load times, and clicks required to get through each page will be the key items that will be tested, timed, and counted.
  - A table of time measurements will ideally be included with the final document.
  - Thoughts concerning improving these aspects of the program include usage of the SQL OUTPUT clause during query execution to prevent additional INSERT queries from having to execute for the many-to-many relationships, an external program that counts clicks & time spent on each page, and in-code timer measurements to properly measure code execution.

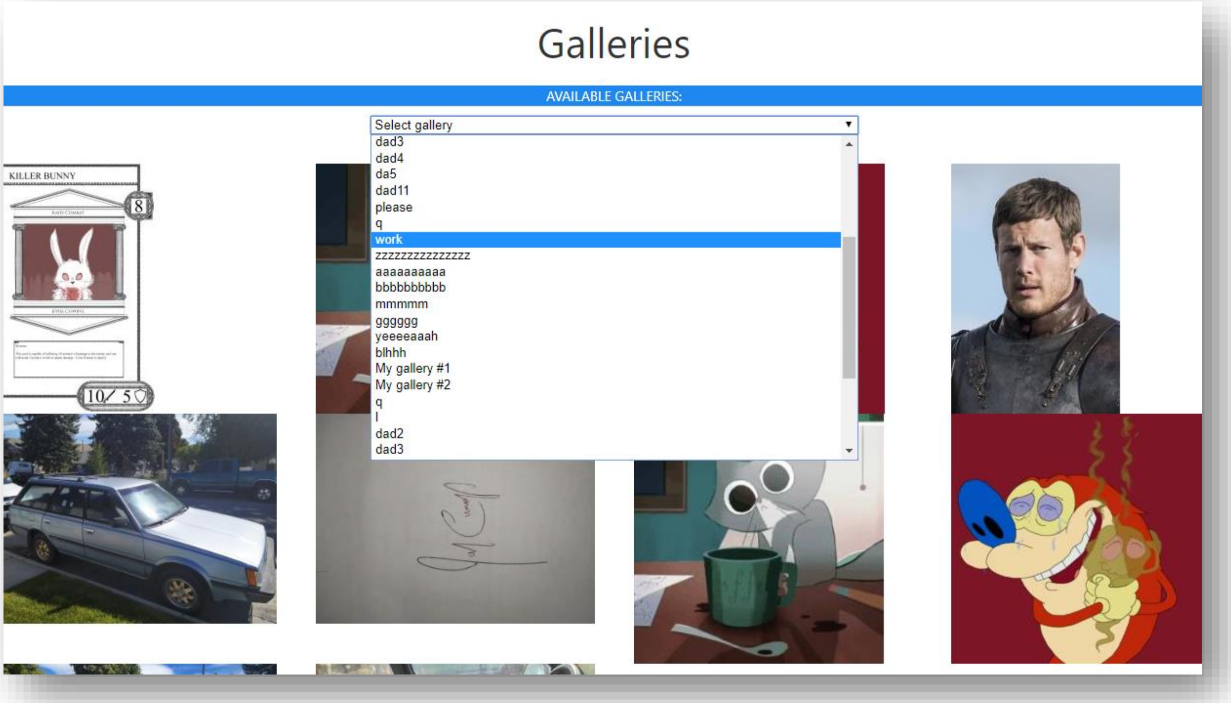


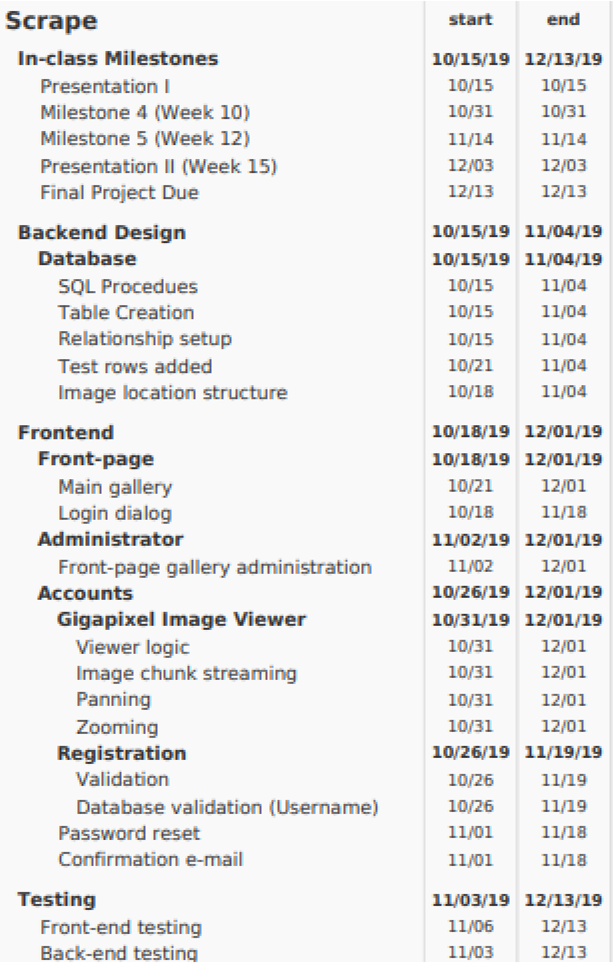
Figure 1 - My assortment of ridiculous test galleries…

Timeline Comparison

Comparisons with the timeline versus with the actual, current progress are not promising. While much of the backend tasks have been accomplished, many others have not been started. Password resets, confirmation e-mails, anything to do with the chunking/streaming of the gigapixel viewing feature, picture comments, picture tags, and the bulk of the testing (along with proper feedback…) have not been started. The frontend is in similar condition, with the main gallery page, administrator pages, gigapixel viewer UI, and testing all having not been worked on.

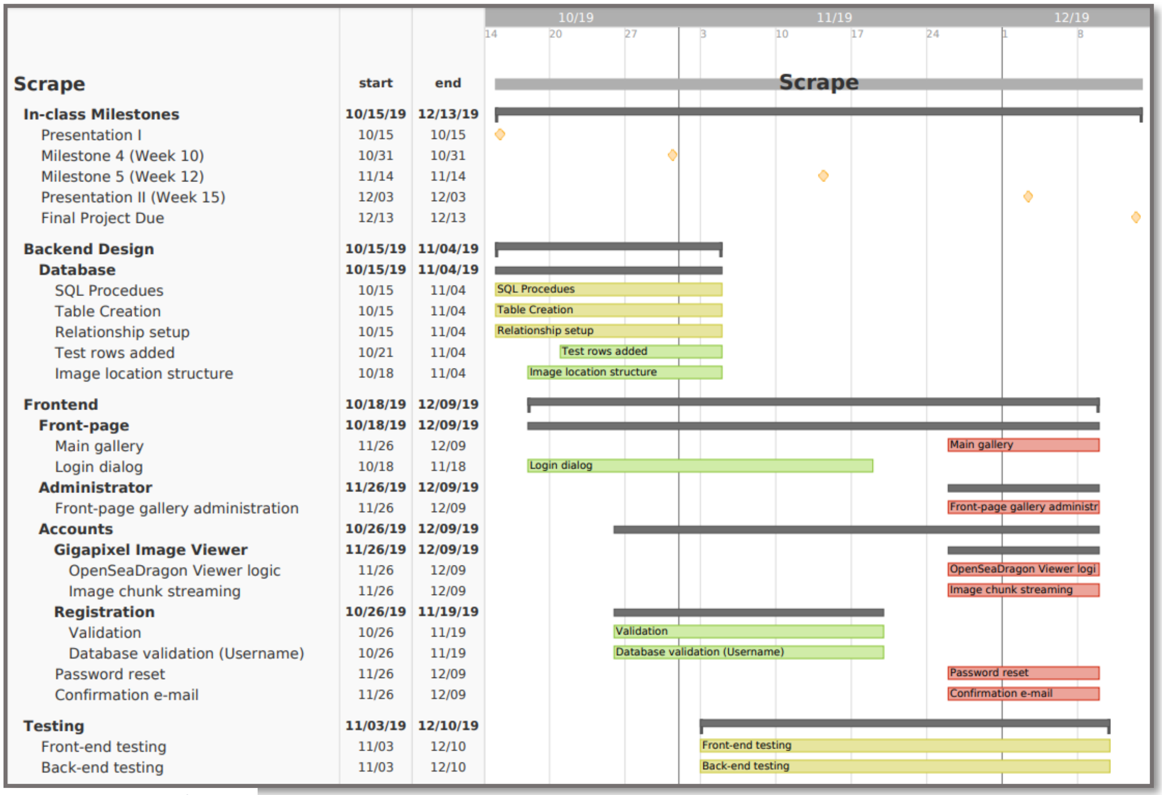
The current plan is to attempt to upload the work I've done to the web server Professor Cassens has provided. I feel this is an important part of the implementation, as I will need to rename my tables in order to properly coexist with other student's tables. This will also require modifying stored procedures. Progress on other aspects of the project will also continue; implementation of OpenSeaDragon for gigapixel pictures and continued refinement of existing pages is also currently on the docket.

Hope is still running high of finishing most of this project (along with the secondary CS project requirement). With the upcoming days off I hope to make good use of extra-time to pull off the daunting tasks ahead.



Scrape	start	end
In-class Milestones	**10/15/19**	**12/13/19**
Presentation I	10/15	10/15
Milestone 4 (Week 10)	10/31	10/31
Milestone 5 (Week 12)	11/14	11/14
Presentation II (Week 15)	12/03	12/03
Final Project Due	12/13	12/13
Backend Design	**10/15/19**	**11/04/19**
Database	**10/15/19**	**11/04/19**
SQL Procedues	10/15	11/04
Table Creation	10/15	11/04
Relationship setup	10/15	11/04
Test rows added	10/21	11/04
Image location structure	10/18	11/04
Frontend	**10/18/19**	**12/01/19**
Front-page	**10/18/19**	**12/01/19**
Main gallery	10/21	12/01
Login dialog	10/18	11/18
Administrator	**11/02/19**	**12/01/19**
Front-page gallery administration	11/02	12/01
Accounts	**10/26/19**	**12/01/19**
Gigapixel Image Viewer	**10/31/19**	**12/01/19**
Viewer logic	10/31	12/01
Image chunk streaming	10/31	12/01
Panning	10/31	12/01
Zooming	10/31	12/01
Registration	**10/26/19**	**11/19/19**
Validation	10/26	11/19
Database validation (Username)	10/26	11/19
Password reset	11/01	11/18
Confirmation e-mail	11/01	11/18
Testing	**11/03/19**	**12/13/19**
Front-end testing	11/06	12/13
Back-end testing	11/03	12/13

Figure 2 – Old timeline



Scrape	start	end
In-class Milestones	**10/15/19**	**12/13/19**
Presentation I	10/15	10/15
Milestone 4 (Week 10)	10/31	10/31
Milestone 5 (Week 12)	11/14	11/14
Presentation II (Week 15)	12/03	12/03
Final Project Due	12/13	12/13
Backend Design	**10/15/19**	**11/04/19**
Database	**10/15/19**	**11/04/19**
SQL Procedues	10/15	11/04
Table Creation	10/15	11/04
Relationship setup	10/15	11/04
Test rows added	10/21	11/04
Image location structure	10/18	11/04
Frontend	**10/18/19**	**12/09/19**
Front-page	**10/18/19**	**12/09/19**
Main gallery	11/26	12/09
Login dialog	10/18	11/18
Administrator	**11/26/19**	**12/09/19**
Front-page gallery administration	11/26	12/09
Accounts	**10/26/19**	**12/09/19**
Gigapixel Image Viewer	**11/26/19**	**12/09/19**
OpenSeaDragon Viewer logic	11/26	12/09
Image chunk streaming	11/26	12/09
Registration	**10/26/19**	**11/19/19**
Validation	10/26	11/19
Database validation (Username)	10/26	11/19
Password reset	11/26	12/09
Confirmation e-mail	11/26	12/09
Testing	**11/03/19**	**12/10/19**
Front-end testing	11/03	12/10
Back-end testing	11/03	12/10

Figure 3 – New timeline