2022

Memestone

Making and sharing memes with the world

Capstone Project

Red Herrings (Red Group):
Tyler Garman
Alex Look
Brian Salinas
Benjamin Villanueva
Patrick Meyer
Christopher James Navarro
Mohammed Amiwala
Eduardo Gonzalez
Alex Lazarov

Table of Contents

Project Overview

Overview

RedHerring is an environment to connect like minded users together. RedHerring will be the new Hub of the internet where everyone will want to be.

Business Mission

RedHerring is the newest platform to help us search for what we all have always desired. The perfect meme. It allows users to view an always-growing pool of memes. The users that are part of the RedHerring community will always know what's trending or the most popular meme. Do you want to be the host of this community? Do you want to be involved? If so, let's talk about what's needed.

Business Objective

Through marketing, we'll build a network of users. Who we expect to invite their like-minded peers. Our users will then have access to our free services. Our service's costs will be supplemented by ads revenue and partnerships deals.

Benefits

The RedHerring site will be an influence in a cultural movement. This means people who value this culture will want to be involved. Memes control internet culture and have the power to draw tens of millions of followers.

Project Group Structure

Project Manager: Tyler Garman

- Manages the team overall, making sure that everything is and everyone is accounted for, responsible for delegating tasks, checking in with members, resolving issues, and making difficult decisions when necessary.

Video Manger: Brian Salinas

- Responsible for all the video presentations and other video editing responsibilities. Proficient with video editing and leads creative direction with putting professional videos together.

Requirements & Implementation Manager: Alex Lazarov

- Responsible for the requirements document and developer/client contract, and makes sure that the team stays in line with the contract.
- Essentially, the lead programmer. Manages the implementation of the code, making sure each code module is completed and meets the standards for the project.

Planner: Eduardo Gonzalez

- Plans out the entire project, creates deadlines and assigns tasks in ClickUp. Must make sure the deadlines are realistic and modify them if changes must be made. Responsible for creating a Gantt chart and the critical path in ClickUp and putting tasks from the project plan onto GitLab.

Designer: Alex Look

- Responsible for creating the formal design of the project, the blueprint that the programming team uses to build it. Works with the programmers and compiles their needs as well as the needs stated by the requirements document into formal diagrams and documents.

Webmaster & Testing Manager: Mohammed Amiwala

- Manages the web software, the tools we use to manage the site on the server, as well as updating it with new changes.
- Responsible for testing the code, making sure it meets the requirements and reducing the chance for bugs to slip out into release.

Documentation Manager: Christopher James (CJ) Navarro

- Responsible for the project booklet, and making sure all our documentation that goes into it is there and up to standards.

Interface Design & Testing, Remote Collaboration Manager, & Systems Programmer: Patrick Meyer

- Responsible for user testing, and making sure the website is accessible and easy to use.
- Since the Red Herring Team does not have a formal office, we rely on online collaboration tools to get work done, and the Remote Collaboration manager is responsible for setting them up and keeping us moving.
- Responsible for the back-end implementation and making sure the website is up and running so the front-end programmers can get the site working.

Presentation Manager: Benjamin Villanueva

- Responsible for presentations and making sure the team looks professional and the product looks desirable. Works with the video manager, providing audio clips and other relevant media needed for the video presentations.

Requirements

1.0 Functional Requirements

1.1 User Classes

- 1.1.1 Welcome page
    - On arrival at the RedHerring site, the user will be brought to the welcome page. The welcome page will have two components that can be used, User Login or User Registration.

- 1.1.2 User Login
    - If the user has a created account. With correct credentials, the user may log in. On success, this will redirect the user to the Main Page.

- 1.1.3 User Registration
    - If a user is trying to create a new account. They must enter a unique username, unique email address, and a password. On success, this will redirect the user to the Main Page.

- 1.1.4 Main Page
    - On arrival of the Main Page, the user will have access to two components, the Navbar, and the meme-cards.

- 1.1.5 Navbar
    - A component that is available on every page except the Welcome Page. This component lets a User navigate to their desired page.

- 1.1.6 Meme-Cards
    - Users will use this tool "like" or "pass" generated memes. If they made a mistake, they have the option to go back to a meme to correct their decision.

- 1.1.7 Top Ranked page
    - On arrival of the Top Ranked page, the user will see a gallery of memes. The memes will be one-hundred most popular memes on the site.
    - The gallery:
        - This will be the tool for users to view all the memes. Using this tool the user will be able to organize this by meme popularity, by username, by newest to oldest, and by oldest to newest.

- 1.1.8 Liked Memes page
    - On arrival of the Liked Memes page, the user will see a gallery of memes. The memes will be a collection of all the memes the user has "liked."
    - The gallery
        - This will be the tool for users to view all the memes. Using this tool the user will be able to organize this by meme popularity, by username, by newest to oldest, and by oldest to newest.

- 1.1.9 Profile page
    - This page will display the user statistics, gallery, and upload component.
    - The upload component will allow users to upload images from their own devices.

1.2 Security

- 1.2.1 Account Security
    - When creating accounts a user must enter a unique email and username.
        - If a username exists on the database, the user will be prompted to choose a new username. This is done to maintain users' unique identities.
        - If an email exists on the database, the user will be prompted to choose a different email. This is done to prevent duplicate usage of users' emails.

- 1.2.2 Cookies
    - Cookies will be allowed so users can pass the login stage if wanted.

- 1.2.3 Password Encryption
    - When registering the password will be encrypted on submission to the Database.

- 1.2.4 Database
    - To prevent connection vulnerabilities, the front end will not have a direct connection to databases, instead, the front end has access to a function to query the database.

- 1.2.5 Password security
    - User inputs of passwords will not be displayed on the screen. On input, it will be represented by alternative characters to prevent malicious actors from seeing passwords and capturing passwords.

- 1.2.6 JSON Web Token Authentication
    - On login users must enter the correct username and password.
        - If a username exists on the database and password matches it creates a JWT string and sends it to the front-end where it will be saved in the browser's local storage until it expires or user logout.
        - When changing to other pages the front-end app will check local storage if there is a JWT string contained. If not, it will redirect to the Login page.
    - Tokens will be sent in the HTTPS request header and validated on the back end. If a token is not provided or is invalid an error response will be sent.

2.0 Constraints

- **2.1 Time**
    - Although we all have been assigned specific roles and a deadline has been set, it is still possible that time will work against us. We are all students and manage other projects as well.
    - Failure to stay on schedule could result in the failure of the project as a whole. Having a well-defined project plan will be the most crucial strategy for effective time management.

- **2.2 Quality**
    - The quality of the project will be measured by how closely the outcome matches the expectations set in the planning stage. Our quality standard may not be met if our quality standard is poorly defined.
    - Poor quality could lead to customer dissatisfaction. By avoiding potential rework and wasted productivity, we can deliver a product that satisfies the customer.

3.0 Project Roles Specification Requirements

3.1 Project manager

- 3.1.1 Manages the team overall, making sure that everything is and everyone is accounted for, responsible for delegating tasks, checking in with members, resolving issues, and making difficult decisions when necessary.

3.2 Video Manager

- 3.2.1 Responsible for all the video presentations and other video editing responsibilities. Proficient with video editing and leads creative direction with putting professional videos together.

3.3 Requirements manager

- 3.3.1 Responsible for the requirements document and developer/client contract, and makes sure that the team stays in line with the contract.

3.4 Planner

- 3.4.1 Plans out the entire project, creates deadlines and assigns tasks in ClickUp. Must make sure the deadlines are realistic and modify them if changes must be made. Responsible for creating a Gantt chart and the critical path in ClickUp and putting tasks from the project plan onto GitLab.

3.5 Designer

- 3.5.1 Responsible for creating the formal design of the project, the blueprint that the programming team uses to build it. Works with the programmers and compiles their needs as well as the needs stated by the requirements document into formal diagrams and documents.

3.6 Implementation manager

- 3.6.1 Essentially, the lead programmer. Manages the implementation of the code, making sure each code module is completed and meets the standards for the project.

3.7 Webmaster

- 3.7.1 Manages the web software, the tools we use to manage the site on the server, as well as updating it with new changes.

3.8 Documentation manager

- 3.8.1 Responsible for the project booklet, and making sure all our documentation that goes into it is there and up to standards.

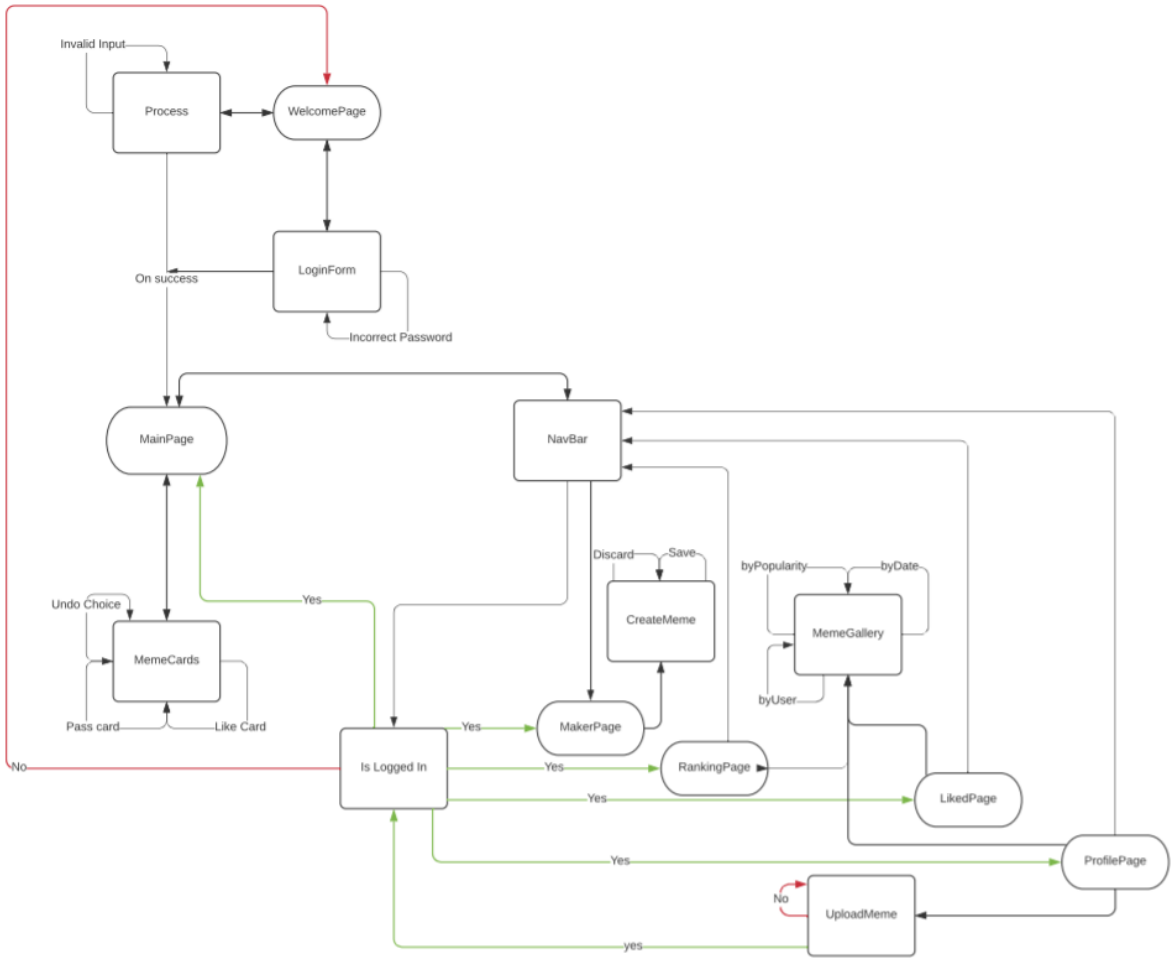
3.9 Interface design and testing

- 3.9.1 Responsible for user testing, and making sure the website is accessible and easy to use.

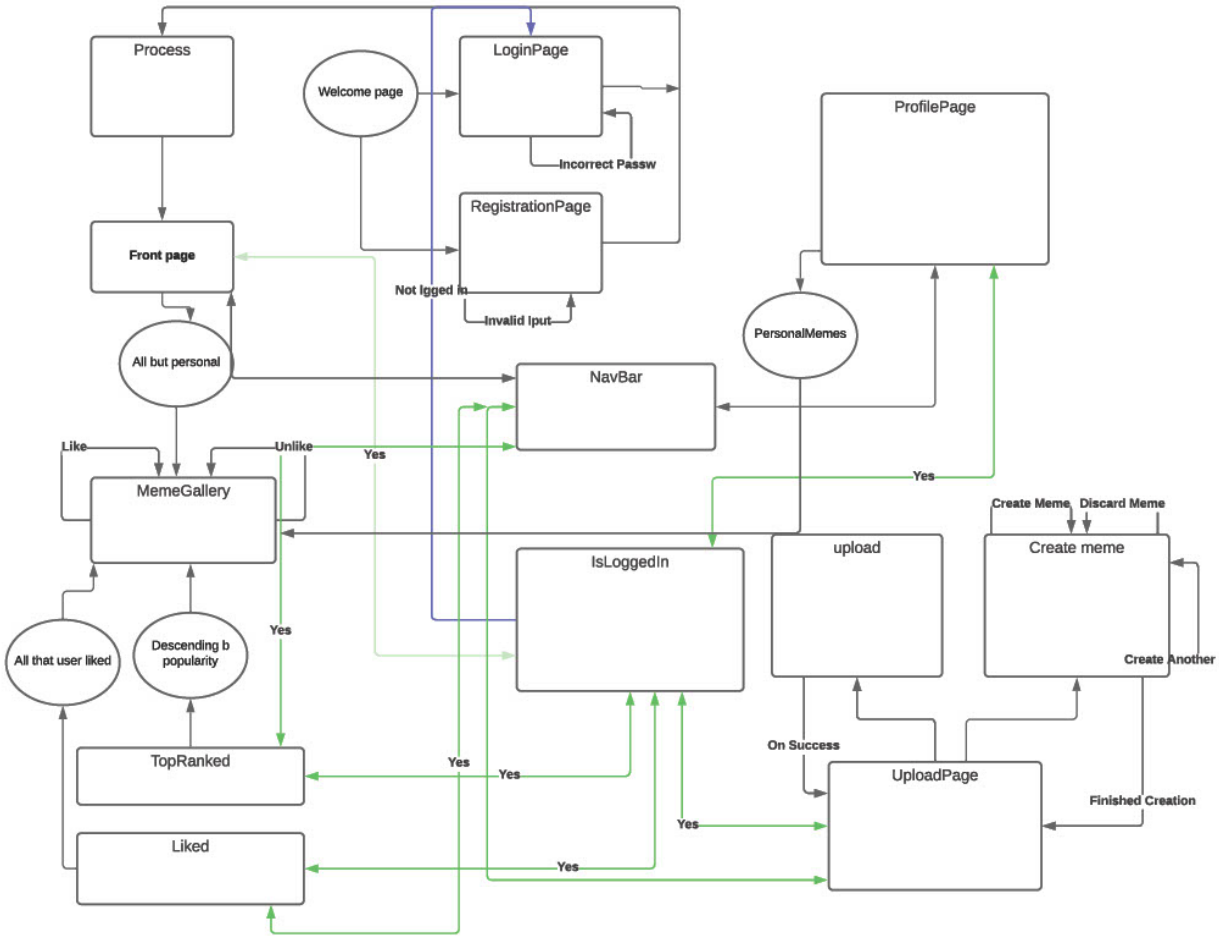
3.10 Testing manager

4.0 Finite State Machine

- 4.1 Version 1



- 4.2 Version 2


Process
Welcome page
LoginPage
Incorrect Passw
ProfilePage
RegistrationPage
Front page
Not lgged In
Invalid Iput
PersonalMemes
All but personal
NavBar
Like
Unlike
Yes
MemeGallery
Yes
Create Meme
Discard Meme
upload
Create meme
IsLoggedIn
All that user liked
Descending b popularity
Yes
Create Another
On Success
TopRanked
Yes
Yes
UploadPage
Finished Creation
Yes
Liked
Yes

5.0 Project Management

- In order to ensure that the product is being developed appropriately to schedule, the project manager will check in regularly with all members of the team, asking for updates on their progress and linking them to internal resources they may need to complete their tasks.
- If members of the development team are unable to meet their requirements, the workload will be redistributed amongst other members.
- **5.1 Replanning**
    - If the development team is unable to realistically meet a deadline, changes will be made to ensure the necessary functionality is completed. In the case that necessary functionality is compromised, a lowest-damage plan will be created and Acme will be presented with said plan.
    - Throughout the course of development there were a few times that the plan had to be re-evaluated in order to meet our deadlines. This issue could often be subjugated by pushing deadlines sooner to create some flexible time that could be used.
    - Some examples of replanning:
        - One member suddenly became unreachable when the deadline for their task came along, and their work had to be redistributed.
        - We had set out in the project plan an additional demo deadline one week before the preview demo, where the goal was to get the product mostly functioning so that it could be tested and revised more comprehensively, but after running into technical difficulties with the code this demo requirement was removed from the plan.
- **5.2 Conflict Resolution**
    - There were not many instances where conflict resolution needed to be used during development, but there were a few times that two or more members had different ideas of how a certain task should be completed, and often those differences could be easily resolved by bringing up the issue to the whole group in a meeting or getting another member's perspective.

6.0 Planning Requirements

- Set up a Discord channel for the Red Herrings team members to be able to communicate effectively and share important information regarding the project.
- Set up and add tasks to ClickUp. ClickUp allows our group to plan, track, and manage any project related work. This software tool is free and easily accessible. It also provides integration with Discord, our main communication channel.

- Open issues on GitLab for programming tasks in the project plan.

7.0 Liaison

- **7.1 Categories for Liaison**
    - In order to ensure that both parties are satisfied with the product, and maintain clear communication throughout the development cycle, Liaisons will be used between both Acme and Red Herring Team. Liaisons must be approved by both Acme and Red Herring Team's Project Manager.

    - 7.1.1 Category A Liaison:
        - A group indicating general interest in the application will be periodically shared demos on the dates dictated by the project plan.
    - 7.1.2 Category B Liaison:
        - A group making contributions to the application will be sent necessary documentation and be invited to relevant meetings.

- 7.1.3 Category C Liaison:
  - A group indicating interest in a component of the application development will be sent reports on its development during the period that the component is scheduled to be developed, as per the project plan.

Functional Requirements Matrix

Section	Number	Revision	Requirement Description	Bad-client attack	Bad-dev attack	Approved?	Sign-off name	Comments
Welcome Page	1.1.1	2	On arrival at the RedHerring site, the user will be brought to the welcome page.	N/A	N/A	Yes	Approved Alex Lazarov 1/24/2022	
	1.1.1	1	When on the welcome page the user will be provided 2 options: login or registration.	Currently Nav-bar is being displayed on login page	Unregistered users have access to other features	Yes	Approved Alex Lazarov 2/19/2022	
Main Page	1.2.1	3	Main page should not be accessible until user is logged in	Users have access to all pages	Users have access to all pages	Yes	Approved Tyler Garman 2/19/2022	
	1.2.2	2	Users should have access to MemeListItem component	yes	Sample component not completed	Yes	Approved Tyler Garman 2/19/2022	
	1.2.3	2	User should have access to NavBar	yes	incomplete component	Yes	Approved Alex Lazarov 2/19/2022	
Meme Maker Page	1.3.1	4	MemeMaker should not be accessible until user is logged in	Users have access to all pages	Users have access to all pages	Yes	Approved Tyler Garman 2/19/2022	
	1.3.2	2	Users should have access to Meme Maker component	yes	incomplete component	Yes	Approved Tyler Garman 2/19/2022	

	1.3.3	2	User should have access to NavBar	yes	incomplete component	Yes	Approved Alex Lazarov 2/19/2022	
Liked Page	1.4.1	2	User should have access to NavBar	yes	incomplete component	Yes	Approved Alex Lazarov 2/19/2022	
	1.4.2	2	User should have access to gallery component	yes	incomplete component	Yes	Approved Tyler Garman 2/19/2022	
	1.4.3	2	LikedPage should not be accessible until user is logged in	Users have access to all pages	Users have access to all pages	Yes	Approved Tyler Garman 2/19/2022	
Top Ranked Page	1.5.1	2	User should have access to NavBar	yes	incomplete component	Yes	Approved Alex Lazarov 2/19/2022	
	1.5.2	2	User should have access to gallery component	yes	incomplete component	Yes	Approved Tyler Garman 2/19/2022	
	1.5.3	2	LikedPage should not be accessible until user is logged in	Users have access to all pages	Users have access to all pages	Yes	Approved Alex Lazarov 2/19/2022	
User Profile Page	1.6.1	2	User should have access to NavBar	yes	incomplete component	Yes	Approved Alex Lazarov 2/19/2022	
	1.6.2	2	User should have access to gallery component	yes	incomplete component	Yes	Approved Tyler Garman 2/19/2022	
	1.6.3	2	UserPage should not be accessible until user is logged in	Users have access to all pages	Users have access to all pages	Yes	Approved Alex Lazarov 2/19/2022	
	1.6.4	2	User has access to UserRanking component	Currently a static Field	User algorithm not	N/A	Not Approved Alex Lazarov 1/26/2022	Feature cut

					implemented			
	1.6.5	3	User has access to UploadCount component	Currently a static Field	Upload api not implemented yet	Yes	Approved Tyler Garman 2/19/2022	
	1.6.6	3	User has access to LikeCount component	Currently a static Field	Upload api not implemented yet	Yes	Approved Tyler Garman 2/19/2022	
Login Component	1.7.1	4	If the user enters existing credentials. i. e Username and password. Users will be redirected to the Main page.	Unregistered users have access to other features	Unregistered users have access to other features	Yes	Approved Alex Lazarov 2/19/2022	
	1.7.2	4	If a user does not have an account associated with RedHerring, they will need a unique username and password. Upon success, the user will be directed to the main Page	Unregistered users have access to other features	Unregistered users have access to other features	Yes	Approved Alex Lazarov 2/19/2022	
	1.7.3	2	If the user enters existing credentials. i. e Username and password. Will make a check to the database and see if the user exists.	N/A	Database api not setup	Yes	Approved Alex Lazarov 2/19/2022	
	1.7.4	3	When logging in database makes secure check encrypting users information	N/A	Database api not setup. Unsecure login	Yes	Approved Alex Lazarov 2/19/2022	
Registration Component	1.8.1	4	If a user does not have an account associated with RedHerring, they will need a unique	N/A	Database api not setup. Unsecure login	Yes	Approved Alex Lazarov 2/19/2022	

			username and password.					
	1.8.2	2	On successful registration the user's information is submitted to database	N/a	Database api not setup. Unsecure login	Yes	Approved Alex Lazarov 2/19/2022	
	1.8.3	2	On successful registration the user can access other parts of the website.	Unregistered users have access to other features	Unregistered users have access to other features	Yes	Approved Alex Lazarov 2/19/2022	
	1.8.4	4	On successful registration user is redirected to main page	Unregistered users have access to other features	Unregistered users have access to other features	No	Not Approved Tyler Garman 2/19/2022	
NavBar Component	1.9.1	2	NavBar Component accessible when user is logged in.	Unregistered users have access to other features	Unregistered users have access to other features	Yes	Approved Alex Lazarov 2/19/2022	
	1.9.2	3	NavBar redirects to other pages	Yes	Nav bar incomplete	Yes	Approved Alex Lazarov 2/19/2022	
	1.9.3	2	On new page click navbar checks if user is logged in, if not it redirects to welcome page	no	Nav bar is incomplete	Yes	Approved Alex Lazarov 2/19/2022	
Swipe Component	1.10.1	3	on arrival Swipe component preloads a pool of images from database	N/A	Swipe does not access database api yet	Yes	Approved Tyler Garman 2/19/2022	
	1.10.2	2	Like swipe submits to database as liked by user	no	swipe api incomplete	Yes	Approved Tyler Garman 2/19/2022	

	1.10.3	2	Like swipe changes to next item in pool and marks as viewed	no	swipe api incomplete	Yes	Approved Tyler Garman 2/19/2022	
	1.10.4	4	dislike swipe passes and goes to next image, disliking and marking as viewed	no	swipe api incomplete	Yes	Approved Tyler Garman 2/19/2022	
	1.10.5	3	Undo goes back one card allowing the user to resubmit a decision	no	swipe api incomplete	Yes	Approved Tyler Garman 2/19/2022	
	1.10.6	2	Swipe properly displays meme creator's information	No it's just hovering randomly	swipe api incomplete	Yes	Approved Tyler Garman 2/19/2022	
MemeMaker Component	1.11.1	3	Pulls templates from database	N/A	No MemeMaker api incomplete	No	Not Approved Alex Lazarov 1/26/2022	Feature cut
	1.11.2	3	On Completion saves new meme to database	N/A	No MemeMaker api incomplete	Yes	Approved Tyler Garman 2/19/2022	
	1.11.3	2	On completion brings user back to start page	No, stalls on a display of creation	No MemeMaker api incomplete	Yes	Approved Tyler Garman 2/19/2022	User is given option
Gallery Component	1.12.1	2	Pulls images from appropriate database	N/A	No gallery api incomplete	Yes	Approved Tyler Garman 2/19/2022	
	1.12.2	2	User allowed to present gallery information by preference	No, gallery is static	No gallery api incomplete	Yes	Approved Tyler Garman 2/19/2022	

Security Requirements Matrix

Number	Revision	Requirement Description	Bad-Client Attack	Bad-Dev attack	Approved?	Sign-off name	
2.1.1	2	Users can create new accounts with individually unique emails not already registered. If the email is tied to an already existing account then the user will be prompted to sign in.	Does not prompt users to sign in.	N/A	Yes	Approved Alex Lazarov 2/19/2022	
2.1.2	3	Users can create new accounts with individually unique usernames not already registered. If the username chosen is not unique to our system then they will be prompted to choose a new username.	N/A	Not connected to database API being applied to built in test	Yes	Approved Alex Lazarov 2/19/2022	
2.1.3	3	If the User attempts to sign in with an email or username with the correct matching password tied to that account then they will successfully log in.	N/A	Not connected to database API being applied to built in test	Yes	Approved Alex Lazarov 2/19/2022	
2.1.4	2	If the User attempts to sign in with an email or username with a password that does not match the password tied to that email or username, then the user will be prompted to reset their password.	N/A	Not connected to database API being applied to built in test	No	Not Approved Alexander lazarov 1/26/2022	Feature cut
2.2.1	4	The user will be allowed to check a box during the sign in process that will allow the user to stay logged in on their device.	N/A	System not implemented yet	No	Not Approved Alexand	Feature cut

		If checked, we will use Cookies that will keep the user logged in on their device and bypass our sign in process.				er lazarov 1/26/2022	
2.3.1	3	On creation of the user account the password will be hashed and salted within our site's database.	N/A	System not implemented yet	Yes	Approved Alex Lazarov 2/19/2022	
2.3.2	2	Whenever the user changes their password using our password change form, the old password will be overwritten and the new password will be encrypted.	N/A	System not implemented yet	No	Not Approved Alexander lazarov 1/26/2022	Feature cut
2.4.1	4	Front-end code will be able to check if image/text content is valid to be added to the database, under a certain file size and character limit.	N/A	System not implemented yet	Yes	Approved Alex Lazarov 2/19/2022	
2.5.1	2	When the user is inputting their password, the password will be censored to hide the user's information.	Yes	Yes	Yes	Approved Alex Lazarov 2/19/2022	

Testing Requirements Matrix

Number	Revision	Test Description	Results	Pass?	Date of test	Sign-off name
T1	2	API calls are tested to make sure we are given back the correct data and parameters.	Success	Yes	2/11/22	Tyler Garman

T2	2	Will use JestJS to test all ReactJS scripts to make sure no fatal errors are prominent in any libraries/scripts	Success	Yes	2/11/22	Mohammed Amiwala
T3	2	Run stress test on back end server to ensure stable traffic handling	Success	Yes	2/11/22	Mohammed Amiwala
T4	3	Check session only cookies to ensure that session does not expire once the browser is closed. Tested on Safari, Chrome, Edge, Firefox on MacOS & Windows 10	Success	Yes	2/11/22	Tyler Garman
T5	3	Test file restriction on image uploader to make sure that nothing other than image type files are able to be uploaded and max file size does not exceed 8MB.	Success	Yes	2/11/22	Mohammed Amiwala
T6	2	3 users are able to create, modify, and delete a meme without issue.	Success	Yes	2/12/22	Mohammed Amiwala
T7	3	Make sure every button works, doing what it is designed for.	Success	Yes	2/12/22	Mohammed Amiwala
T8	2	3 users were able to create 3 memes from the template page without receiving an error.	Success	Yes	2/12/22	Mohammed Amiwala
T9	2	3 users were able to upload a meme at the same time without issue.	Success	Yes	2/12/22	Mohammed Amiwala
T10	2	Verify that the user sign-up page is working and connects to our database.	Success	Yes	2/12/22	Tyler Garman
T11	2	3 users are able to interact with each other through the comments section.	Feature removed		2/12/22	
T12	2	Have 5 users create accounts.	Success	Yes	2/13/22	Mohammed Amiwala
T13	2	Have 5 users log in with	Success	Yes	2/13/22	Mohammed Amiwala

		existing accounts.				
T14	2	Have 5 users open another page and check if they are still logged in.	Success	Yes	2/13/22	Mohammed Amiwala
T15	2	Have 5 users open multiple pages and log out on one. They should be logged out on each page.	Success	Yes	2/13/22	Mohammed Amiwala
T16	2	Ask one user to create a meme on the site, make sure it appears on that user's profile, and another user can find it on their homepage.	Success	Yes	2/13/22	Mohammed Amiwala
T17	2	Have 5 users scroll through their homepage, liking at least one meme, and make sure it can be found in their liked memes page.	Success	Yes	2/13/22	Mohammed Amiwala
T18	2	5 users, testing multiple parts of the website, clicking around, using the different features, should not have to wait more than a second for anything to load.	Success	Yes	2/13/22	Mohammed Amiwala

Plan & Phases

The Planning Manager addressed the development of the plan as well as supporting plans that were developed as needed.The activities and tasks defined in the project plan must be undertaken within the scope of the project. A detailed work task description, work breakdown structure, division of responsibilities and preliminary schedule are included.

To keep track of the tasks that needed to be completed and who was completing them, ClickUp was used. ClickUp is a cloud-based collaboration and management tool that allows our group to plan, track and manage project related work. We talked extensively about having clear objectives and goals throughout the project. They enabled our group to visualize, plan, and document objectives that cleared the path for smart and effective project management. By creating milestones and acknowledging risk factors, the group

is able to devise flexible action steps that effectively respond to any issues that may come up.

Our project was divided into a total of 4 phases, each with a goal in mind that would ultimately lead to our final deliverable.

Phase 1

- Early design concepts
- Documentation guides
- Programming
- Concept Demo

Phase 2

- Continuation of design plans
- Project plan and outline
- Formal design
- Requirements documentation
- RDP video

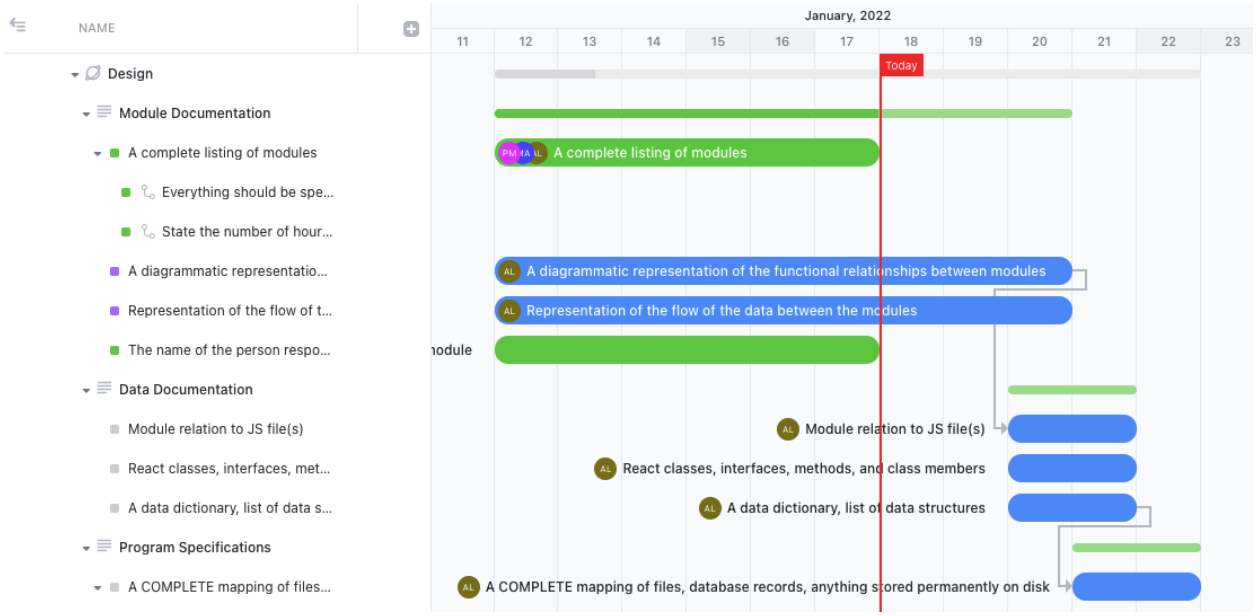
Phase 3

- Building front-end
- Building back-end
- User-testing
- Preview Demo

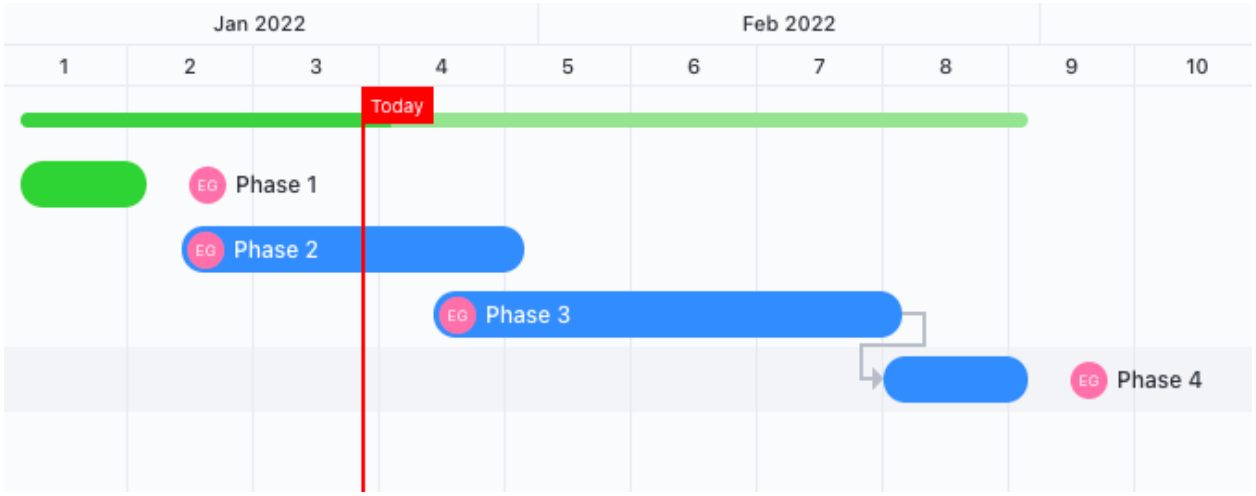
Phase 4

- Revisions/modifications
- Final test
- Final demo

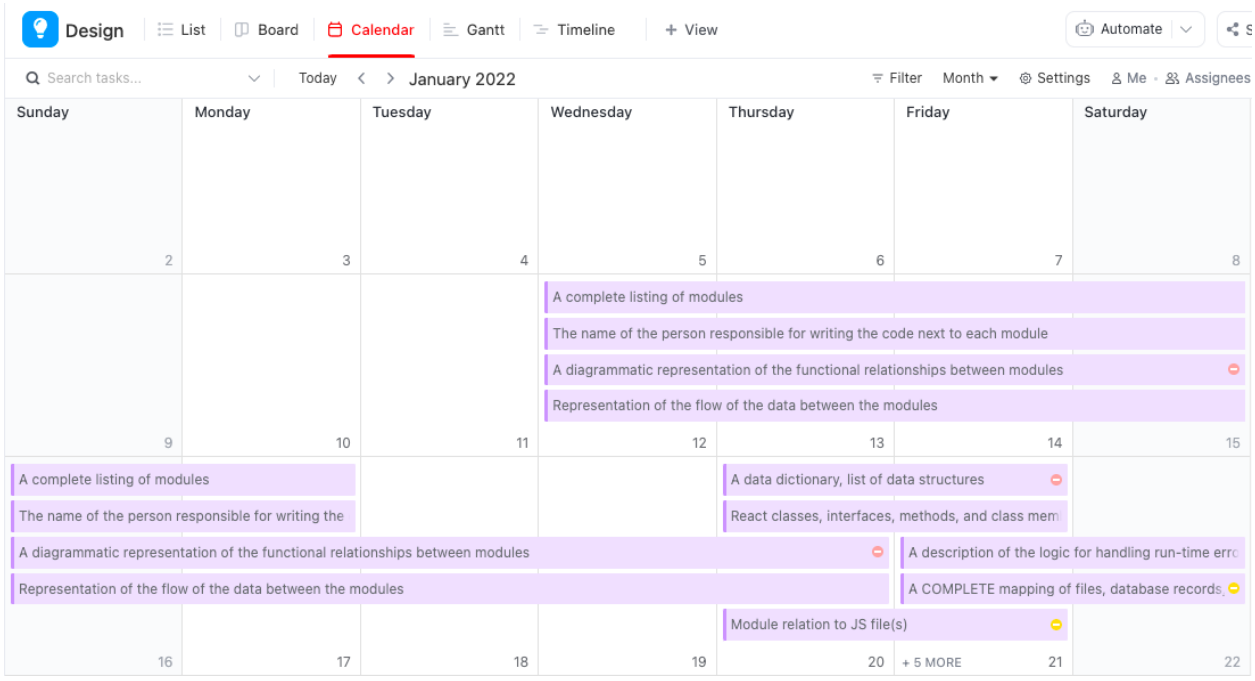
Gantt Chart



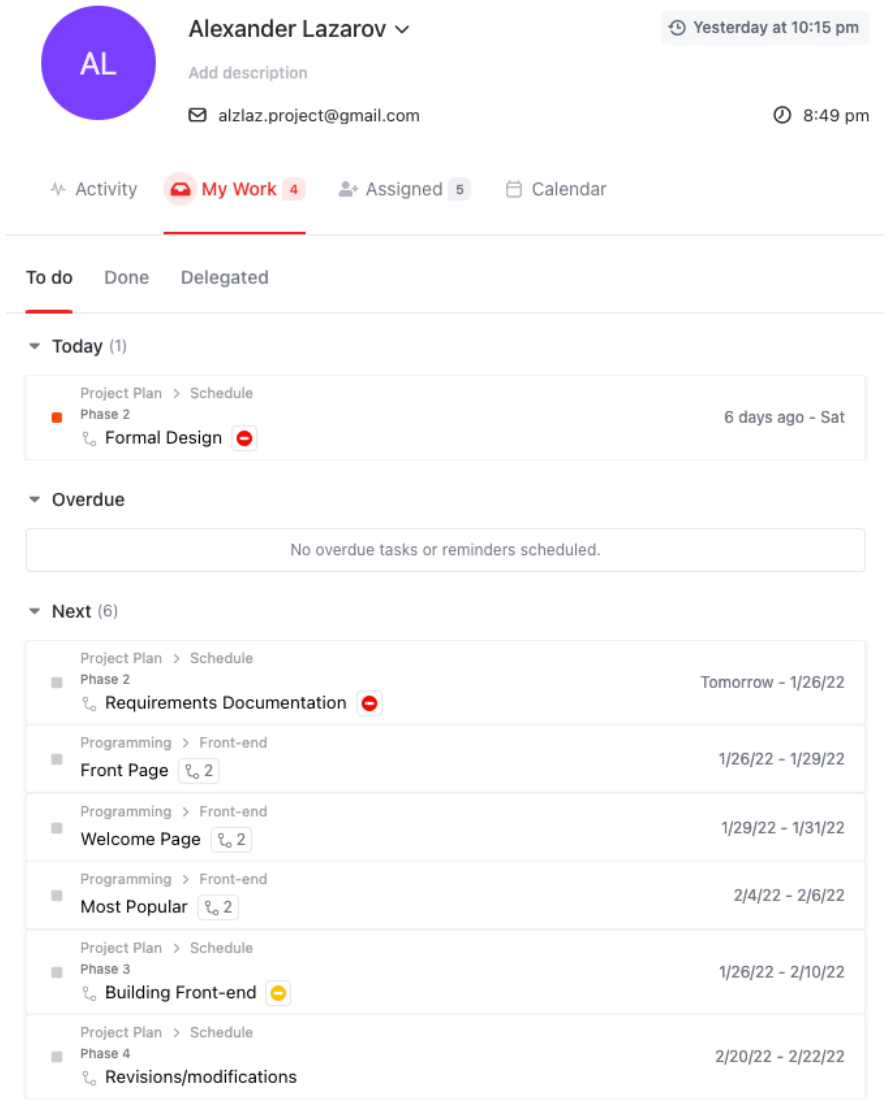
Critical Path

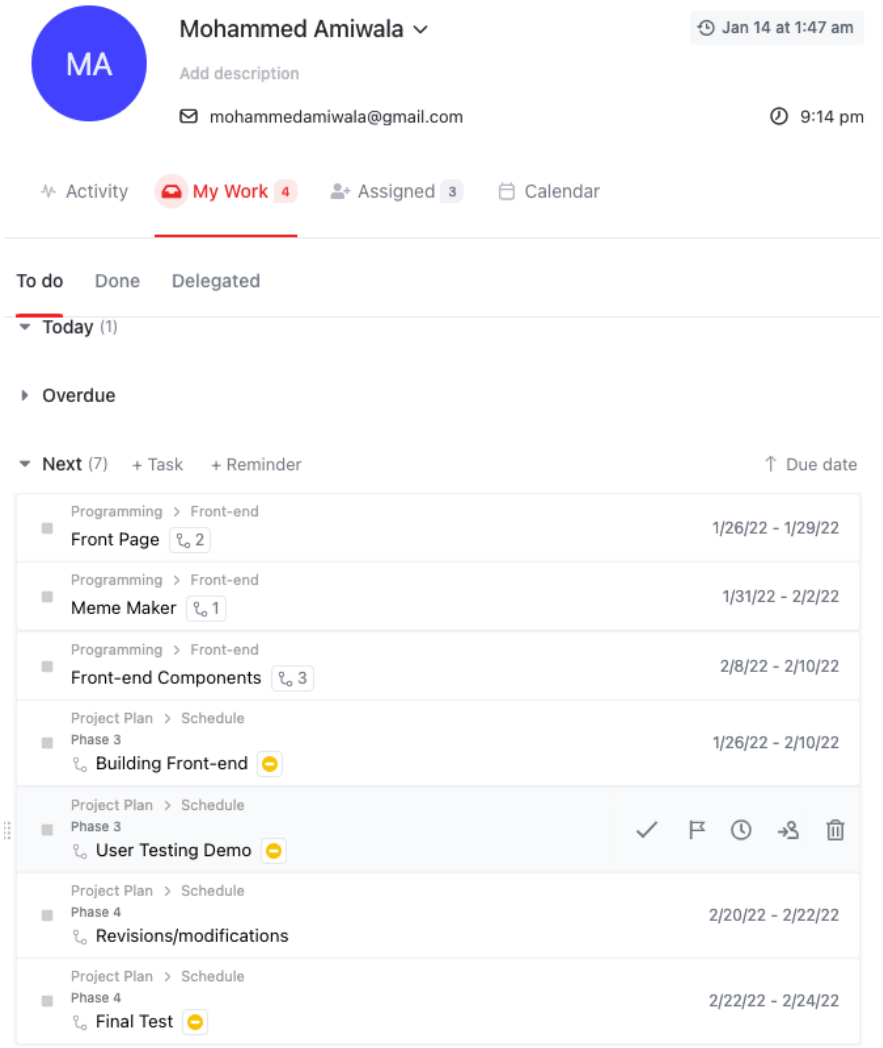


Calendar View



Individual Tasks Examples




MA
Mohammed Amiwala
Jan 14 at 1:47 am
Add description
mohammedamiwala@gmail.com
9:14 pm
Activity
My Work 4
Assigned 3
Calendar
To do
Done
Delegated
Today (1)
Overdue
Next (7)
+ Task
+ Reminder
Due date
Programming > Front-end
Front Page 2
1/26/22 - 1/29/22
Programming > Front-end
Meme Maker 1
1/31/22 - 2/2/22
Programming > Front-end
Front-end Components 3
2/8/22 - 2/10/22
Project Plan > Schedule
Phase 3
Building Front-end
1/26/22 - 2/10/22
Project Plan > Schedule
Phase 3
User Testing Demo
Project Plan > Schedule
Phase 4
Revisions/modifications
2/20/22 - 2/22/22
Project Plan > Schedule
Phase 4
Final Test
2/22/22 - 2/24/22

Overall Project Plan

Phase 1	Assigned Team Members	Dependencies	Timeline/Duration	Planned Hrs	Due Date
Brainstorm ideas/concepts	ALL		1 day	2	1/3/2022
Documentation guides	Eduardo, Tyler, Look, CJ	Need formal concept	2 days	26	1/5/2022
Programming	Mo, Patrick, Tyler, Lazarov	Documentation completion	3 days	32	1/7/2022
Concept Demo	Ben, Brian	Finished programming	1 day	8	1/9/2022
Phase 2	Assigned Team Members	Dependencies	Timeline/Duration	Planned Hrs	Due Date
Brainstorm ideas/features	ALL		2 days	16	1/12/2022
Project Plan	Eduardo, Tyler, CJ		5 days	40	1/17/2022
Formal Design	Look, Tyler, Mo, Patrick, Lazarov		10 days	52	1/22/2022
Requirements Documentation	Lazarov, CJ, Tyler	Need final design	7 days	40	1/26/2022
RDP Video	Ben, Brian	Plan, design & requirements completion	4 days	12	1/30/2022
Phase 3	Assigned Team Members	Dependencies	Timeline/Duration	Planned Hrs	Due Date
Building Front-end	Tyler, Lazarov, Mo	Final design and requirements documentation	16 days	70	2/25/2022
Building Back-end	Patrick	Final design and requirements documentation	16 days	70	2/15/2022
User testing demo	Patrick, Mo	Front-end and back-end completion	3 days	16	2/17/2022
Preview Demo	Ben, Brian	User approval	6 days	12	2/20/2022
Phase 4	Assigned Team Members	Dependencies	Timeline/Duration	Planned Hrs	Due Date
Revisions/modifications	Tyler, Lazarov, Mo, Patrick		2 days	12	2/22/2022
Final test	Mo	User testing	4 days	16	2/24/2022

		finished			
Final Demo	Ben, Brian	Test approval	3 days	12	2/27/2022

Meeting Agendas/Documentation:

For our large group meetings, our project manager led each meeting, often opening with a prewritten agenda to guide the meeting and make sure the team stays on task. For meetings without agendas, some simple notes about what happened during the meeting were recorded.

1/3:

- First Meeting
- Discussed individual strengths and assigned all roles to members

1/5:

- Meeting to discuss what software should be used for programming the concept demo
- Split up basic “modules” such as pages and front end components amongst the programmers

1/8:

- Presentation plan meeting
- Everyone worked on the script for the video
- Decided that our presentation manager would speak for the group for this video, with just spoken introductions from each group member

1/10 Agenda:

- Recognition & Achievement
    - Concept Demo & Presentation was a success!
    - What went right?
- Constructive Feedback
    - What went wrong?
    - How can we improve in the future?
    - Do we all feel comfortable with our roles?
- What is due tonight
    - PM: revised project responsibilities document
- Our next assignment
    - Review assignment due date and all requirements
    - Huge responsibility for **requirement, documentation and planning managers.**

    - 3 weeks to complete
- What we need to work on now
    - Further fleshing out design, brainstorming new ideas
    - Formal Design
        - Non-visual design of all pages, structure for back end components such as the database objects
        - UML Diagrams
    - Formal requirements Document
    - “Big Picture” plan for the rest of the quarter

1/12:

- Brainstorming session
- Brought ideas to the table and brainstormed new ones in
- At the end of the meeting we sorted everything into the categories:
    - Essentials - for things that we must get in there
    - Hard Ideas - for things we really wish to get in there but will definitely take a lot of time and possibly could be moved to the back burner
    - Backburner - ideas that we like but probably will not complete, unless we have sufficient extra time and desire

1/13:

- Programmers Design Meeting
- Attendance: Lead Programmers and Design Manager
- We discussed the basic module breakdown for the formal design of the website.
- Created Module Breakdown and started formal design document

1/17 Agenda:

- Summary of last week’s work:
    - Brainstorming session
    - Finalizing design
    - Design manager and programmer meeting
        - Worked on module structure and details
- This week:
    - Alex Look will make the formal design documents
    - The requirements documentation
    - Planning documentation
- Next week:
    - Presentation/Video
- What is still needed for the formal design
- What is needed for the requirements documentation

1/23 Agenda:

- Review Formal design
    - Review security statements, browsers supported, etc.
- Go over/do what still needs to be done for the requirements document
    - Functional requirements Matrix - All
    - Testing requirements Matrix - Mo, others
    - Technical requirements
    - Constraints - Eduardo
    - Category for Liason (how we communicate with the “clients”) - Tyler
- This week: RDP Video

1/24 Agenda:

- How is everyone doing on the requirements tasks?
- Requirements document due Wednesday, so get on those tasks!
- Compile requirements document by Thursday - CJ
    - Note: At the end please make a final check to make sure no updates were made since you compiled different components!
- Formal Design
    - Modules need to be broken up into submodules <5 hours each. - Alex Look, programmers
- RDP Video
    - Assign people to work on script parts
    - Script Due Thursday!
    - Record presentation - Ben and/or others
    - Produce video - Brian

1/26 Agenda:

- Look over the progress on the functional requirements
    - See new requirements matrix excel sheet Alex Lazarov created
    - Assess/add what is missing
- Formal design modules
- Get your script parts done by tomorrow night!
- Reduce design module time to max 5 hours
- Put all requirements (testing/security) into the excel sheet - CJ
- Add numbering to the excel sheet - CJ
- Reword ambiguous functional requirements - Alex Laz

1/31 Agenda:

- RDP Demo Complete! Good job everybody!

- We're now entering phase 3 of the grand plan!
- What needs to be done now?
- Missing interface design concepts for the Meme Maker
    - Since this is such a big component of the site, we will need to formalize a good interface design incorporating our ideas and requirements.
- Other interface designs/Updates?
- Module Breakdown / Assignment - Starting fresh
- RDP Video Review - Due friday
    - Give feedback for other group's videos

2/7 Agenda:

- Go over how we will attack the programming
    - Patrick's database testing code
- ClickUp tasks
- Assigning Work
    - Css People
    - Graphics Design/Logo

2/9:

- Group check in on programming work
- Troubleshooted issues with back end setup

2/14:

- Additional troubleshooting session for programmers
- Given time constraints and technical difficulties, several features had to be cut in order to keep us on track

2/21 Agenda:

- Professor Elliot's Notes:
    - Not enough content/details
    - Hard to see text
    - Need technical aspect
    - Might want to censor some memes to avoid offending clients
    - Do we need a page structure for a ton of memes
- Fixing issues with the demo for the final demo
- Project Manual
- Minimal user testing/making sure requirements are filled out
- Updating ClickUp (eduardo)
- Formal documents illustrating how code works/showing actual code
- Code updates

2/18 Agenda:

- We finished the final demo! Congrats to everyone!
- Alpha Ranking assignment…
- This week: Project Manual!
- Open discussion: What additional content should we add?
    - Design documents need to actually include authentication details as to how it was implemented - Lazarov
    - Design could be updated overall with more detail about implementation
    - Some level of code needs to be shown - All coders
    - Video/presentation segment - Near requirements - Brian/ben
    - Tools used inventory - Anyone who used tools
    - AWS setup details - Tyler
    - Testing details - Mo
    - Update UML diagrams - Alex Look
    - Update Finite state machine
    - Update requirements matrices - CJ
    - Anything else?
- Meeting wednesday at 7:30

3/2:

- Group work session, working on final project manual
- Went over what still needed to be done
- Looked through examples provided on D2L

Design

The design manager took all ideas presented by the group. Once agreed upon creation of visual diagrams of their modules for the programmers to reference.

Complete Listing of Modules

Front Page - **Alex & Mo (10 hours)**

- Total hours: 10hrs
- Front page (visual): 5 hrs
- FileName: FrontPage.js
- When navigated to it will display a list of images that are held in SwipeAnimation.js
- Includes: SwipeAnimation navbar css

SwipeAnimation Components:

SwipeAnimation.js

- 5 hrs
- Filename: SwipeAnimation.js
- Description: card animation that pulls from the database a group of x new memes can swipe right for likes, swipe right for dislikes, and press the back button to go back to the previous meme. When the end of the group is reached, it will make a request to the database to provide a new group. The specific number in a group will be tested for optimization.
    - User ID
    - Username
    - Meme ID
    - Memes name
    - Memes image
    - List of memes
    - Meme likes
    - User Likes
    - Date posted

Welcome Page - **Alex (15 hours)**

- Total hours: 15hrs
- Welcome page: 5 hrs
    - Integration of components and stylizing welcome page.
- Filename WelcomePage.js
- Description: On arrival if a user is a first time user or a user that has not saved their credentials it will display options to components LoginForm.js and RegistrationForm.js. If they are not a first time user and their login credentials are saved they are redirected to the front page.
- Includes LoginForm, RegistrationForm

Welcome page components:

LoginForm.js

- 5hrs
- Filename LoginForm.js
- Description: Form to fill in user information to submit to db to authenticate if there is an existing user. If successful, it redirects to the front page. On failure, it shows an error.
- Fields
    - Username
    - Password

RegistrationForm.js

- 5hrs
- FileName RegistrationForm.js
- Description: Form to fill in for registration for new account to submit. If successful, it will redirect to the front page. On failure shows error if username or email already exists.
- Fields
    - Username
    - Password
    - Confirmation Password
    - Email

Meme Generator Page - **Mo (25 hours)**

- Visual component - 5 hrs
- Functional component - 5 hrs
- Filename: MemeGenerator.js
- Description: Allows the user to upload an image file (supports PNG, JPEG, and GIF) or use a current meme template and lets them input text to be placed as “top text” or “bottom text”. The meme generator will then edit the image file to display the input text.
- Includes Navbar, ImageUploader, MemeTemplate
- Fields:
    - Template list (fetched from database)
    - Meme image URL (uploaded by user or clicked on from list)
    - Top text field
    - Bottom text field
    - Finished meme URL
- Meme Generator Page components:
    - Template:
        - 5 hrs
        - Filename Template.js
        - Description: Contains information about a given meme template
        - Fields:
            - Template ID (fetched from parent)
            - Template image URL (fetched from template ID)
            - List of textboxes
                - Each textbox contains:

        - x
        - y
        - font
        - font-size
        - angle
    - DemoMeme:
        - 5 hrs
        - Filename: DemoMeme.js
        - Description: Pulls data from a template to generate a demo image for how it could look.
        - Includes Template
    - MemeTemplate:
        - 5 hrs
        - Filename MemeTemplate.js
        - Description: Displays a template and handles its options.
        - Includes Template

Profile Page - **Tyler (10 hours)**

- 5 hrs
- Filename ProfilePage.js
- Description: Profile page, to view other users as well as yourself
- Includes Navbar, Gallery, ProfileEdit
- Fields:
    - User ID (passed by parent)
    - User profile picture URL (fetched with user ID)
    - Number of likes (fetched with user ID)
    - Number of uploads (fetched with user ID)
    - List of memes user uploaded (fetched with user ID)
        - Represented with MemeListItem
- Profile Page components:
    - Profile Edit
        - 5 hours
        - Filename ProfileEdit.js
        - Description: Profile editing form, allows uploading profile photo and changing username
        - Includes ImageUploader

Most Popular - **Alex (5 hours)**

- Hours: 5 hrs
- Filename: MostPopular.js
- Description: Displays a gallery component that will show the top x amount of memes on the server.
- Gallery query:

    - Find memes with the highest liked memes. This can be organized by users, post-date, number of likes, etc.
- Includes Gallery, Navbar

Liked memes - **Tyler (5 hours)**

- Total hours: 5 hours
- Filename LikedMemes.js
- Description: Shows the liked memes for the currently logged in user.
- Gallery query:
    - Find logged in user's liked memes
- Includes gallery, navbar

General Front-End components:

Navbar: Tyler (5 hours)

- 5 hrs
- Filename Navbar.js
- Description: The navbar for the whole site. Placed at the top of each individual page, it allows users to quickly navigate to the main pages.
- Links to:
    - Front page
    - Meme Maker
    - Most Popular
    - Liked Memes
    - Profile page

Gallery Component: Alex (7hours)

- Filename: Gallery.js
- Description: Component that accesses the meme database based on a given search query
- Includes MemeListItem
- Fields:
    - Query (given by parent)
    - User ID (given by parent)
    - Username
    - Meme ID
    - Memes name
    - Memes image
    - List of memes
    - Meme likes
    - User Likes
    - Date posted
- Integrating gallery system to pull data from the database

    - 5hrs.

- Stylizing gallery components to match the theme of the site.

    - 2hrs

MemeListItem (Used in Profile, Most Popular memes, Liked memes) - **Tyler (10 hours)**

- Visual component - 5 hrs
    - Displays meme with given data
- Functional component - 5 hrs
    - Pulls the meme from the database, pulls comments, and submits likes to the database.
- Filename MemeListItem.js
- Description: An individual meme, shown in a list/gallery.
- Includes CommentView component
- Fields:
    - Meme ID (passed by parent)
    - Meme image URL (fetched with meme ID)
    - Associated User ID (fetched with meme ID)
    - Associated User profile picture URL (fetched with user ID)
    - Number of likes (fetched with meme ID)
    - optional: Rank (passed by parent)
    - Comment List ID (fetched with meme ID)
        - Links to CommentView

Image Uploader - Mo (5 hours)

- 5 hrs
- Filename ImageUploader.js
- Description: Handles image uploading, polling the user, and submitting images to the database.
- Fields:
    - Type (Meme or Profile photo)
    - User ID (if Profile photo type)
    - Uploaded image URL

Animations - **Mo (5 hours)**

- 5 hours
- Swipe Direction
- Button animations
    - Like
    - Favorite
    - Next
    - Back

Data Objects and DB Design/Implementation

Assigned to Patrick (15 hours)

Tasks:

- Meme table 2h
- Comments table 2h
- Templates table 2h
- Images table 2h
- User data table 2h
- Test with Node 5h

Table Layout:

- Memes
    - UUID
    - URL
    - Timestamp
    - Caption
    - Tags
    - Template
    - Is Private?
    - Poster ID
    - Users Liked

- Comments
    - Meme UUID
    - UID
    - Text
    - Users
    - ID
    - Email
    - Username
    - Date Joined
    - Seen memes

- Templates
    - Key
    - URL
    - User
- Images
    - Filename
    - Uploader User ID

Physical file organization server-side

Assigned to Patrick (10 hours)

- Stored on bare metal or in a container? 2h

- Create directory structure according to SQL 1h
- Grab initial image stock 2h
- Double check user permissions 1h
- Test with SQL 2h
- Test with Node 2h

Back end NodeJS interface for DB access
Assigned to Patrick (15 hours)

- Agree upon/set API for front end clients 5h
- Sequelize models for all required queries 5h
- Test with SQL 5h

Excluded Modules

These are modules that made it into the design plans and were worked on but were never able to be fully functionally implemented into the final website. Due to reasons such as time constraint for deadlines or ability to functionally work modules within the rest of the website why these were taken out. For future expasnion of the website these ideas would be implemented.

Comments - Tyler (5 hours)

- 5 hrs
- Filename Comment.js
- Description: Individual comment
- Fields:
    - Comment User ID (passed by parent)
    - Comment User Profile Picture (fetched with user ID)
    - Comment text (fetched with comment ID)

Comment View Page - **Alex (10 hours)**

- Visual component - 5 hrs
    - Displays comment components, given list.
- Functional component - 5 hrs
    - Pulls the list from the database, populates the list and relays the comment number to its parent.
- Filename CommentView.js
- Description: Displays a list of comments for a specific meme
- Includes Comment component
- Fields:
    - Meme ID (passed by parent)
    - Meme image URL (fetched with meme ID)
    - Number of likes (fetched with meme ID)
    - Comment List (fetched with Comment List ID)

Brainstorming

For our main brainstorming session, we used a loose voting method where all members would bring ideas to the table and talk out those ideas, assigning votes to ideas they liked. After all the ideas were voted on, we discussed each idea and put it into one of three categories: Essentials, Hard Ideas, and Back-burners. Essentials and hard ideas were those that we were hopefully going to put into the finished product, and back-burners were ones that likely would not make it, only if we had an abundance of extra time. Finally, we also had a category for ideas that we simply rejected.
For voting, each member used an emoji as seen below to mark which ideas they liked.

Alex Look 🦉
Tyler 🐈
Alex Lazarov 🦝
Patrick 🦥
Mo 🦔
Ben 🦚
Brian 🦔
Eduardo 🐸
CJ 🦎

Essential Ideas

- Comments on posts 🐈
- User bios
    - Contact info (other sites) 🐸 🐈
- Private Posts 🐸 🐈 🦚
- Color updating buttons to show an action has been done (like, fave, next/back) 🐸 🐈
- sharing functionality for other social platforms (twitter, fb, etc) 🐈
- For memes we should track likes, views, and shares(?). 🐸 🐈 🦚
- Search functionality? find memes with keywords 🐈 🦎 🦚 🦉
    - Tags/captions for memes?
- Way to exit out of memes in meme maker 🐈 🦚
- Access to sign in screen/ sign out button in profile 🐈
- Advanced meme maker (fonts, placement, editing, etc.) 🐈
- Trending section front page

Harder/Stretch Ideas

- Official type of "template" memes 🐸 🦎 🐈 🦚 🦉

    - These can be created and used by users, allowing meme templates to be designed for new memes, preventing our site from becoming outdated
    - Templates can have text boxes with position, color, and size attributes (though more could be added), and these would inform the meme maker how to create the images
    - Using a template could create a reference in the memes to the template, allowing users to easily find it and make their own
    - Templates could have their own tag or category in liked memes, user profile page
    - Most popular templates (most number of uses)
- PvP zone 🦥🐸🦥🐸🐈🦎
    - Could function like this: Users can request to fight a meme made by someone else, submitting their own meme as a contestant. Both memes must be public. If the user accepts the duel, the two memes will be linked, and will be shown together as fighting in the homepage. Users can decide which of the two they like better (Separate from liking the meme) and that would add to its score. After a certain amount of time/likes, one will triumph over the other, and they will be unlinked, containing a reference to the other to signify who won/lost. 🐈
        - Instead of a request, it could just be that a few most popular memes are chosen each week to duke it out
- "weekly" best meme (based upon user votes/likes) (if likes are being tracked/counted) 🐸🐈
- Security:
    - Check user uploading interval to prevent overuse/abuse of out bandwidth and resources 🐈🦚
    - Enforce maximum per-user upload count for the same reasons as the above 🐈🦚
    - Proper user login/authentication, ala A, B, and C. 🐈🦚
- Ability to delete a published post 🐈🐸🦎

Back-Burner Ideas

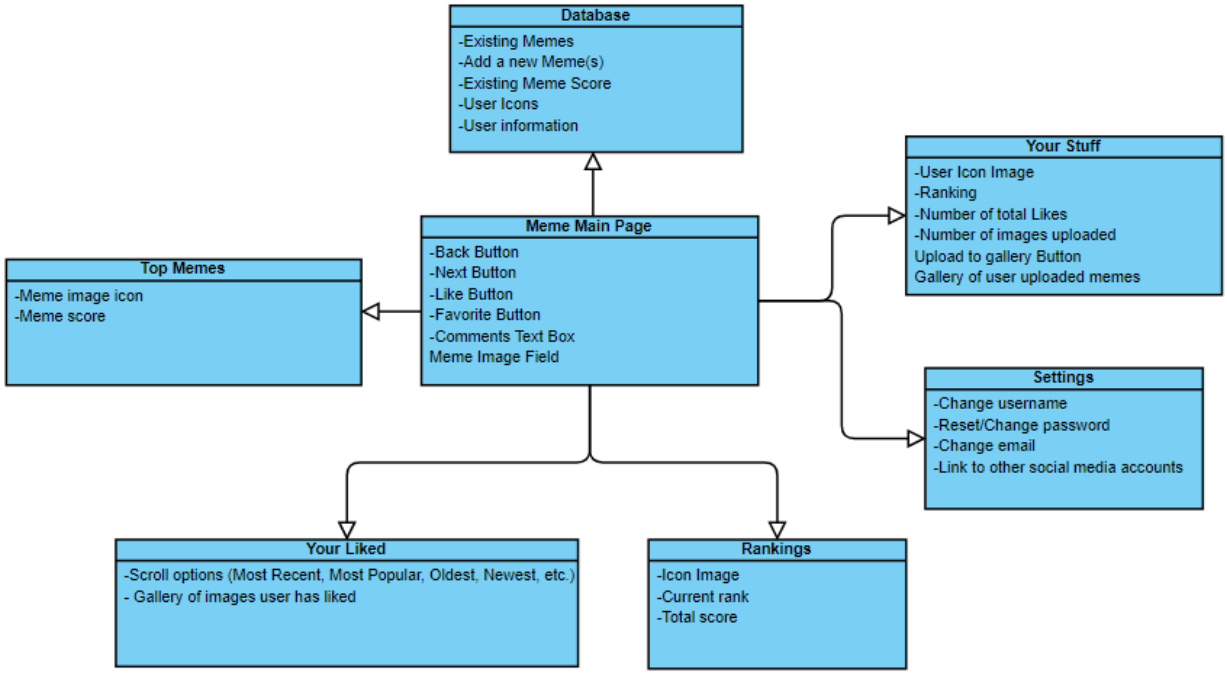
- Votes on comments
- Private accounts - Must be followed to see memes in your home page
    - Home page will sprinkle in private memes from each of YOUR followed user's memes into your home page meme list
- Algorithm for home page based on tags
    - Allow some amount of un-tagged or not liked tags to filter through to avoid users getting too locked out
- Solid color brush in meme maker

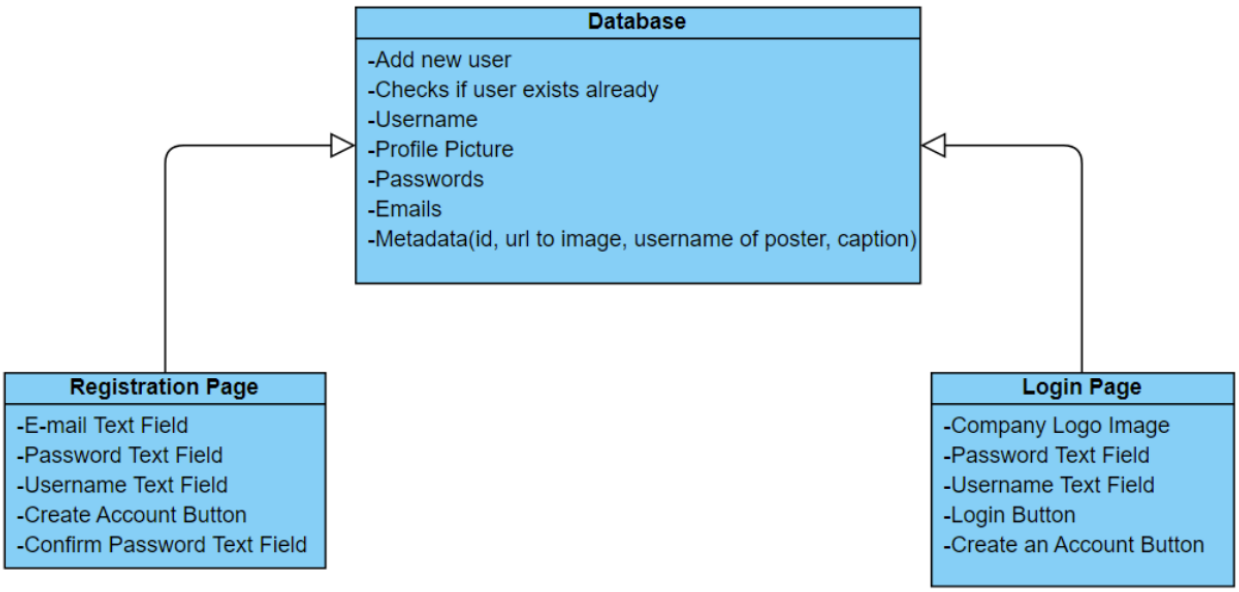
- Removing the dating-app style home-page 🐈
- User follow ability 🐸🦚
    - Would require some distinction, such as a following page to show followed users memes (like TikTok)
- Followed by/following list (similar to instagram)

Rejected Ideas

- Sending messages to other users (Might be a bit much)
- General Meme view page - A larger view of a specific meme, showing everything the front page shows, but can be accessed from other pages, such as Liked Memes and User Profiles.
    - Could be a pop-up
- user page visitor counter
- For the home page, I think we should follow a similar layout to giphy.com

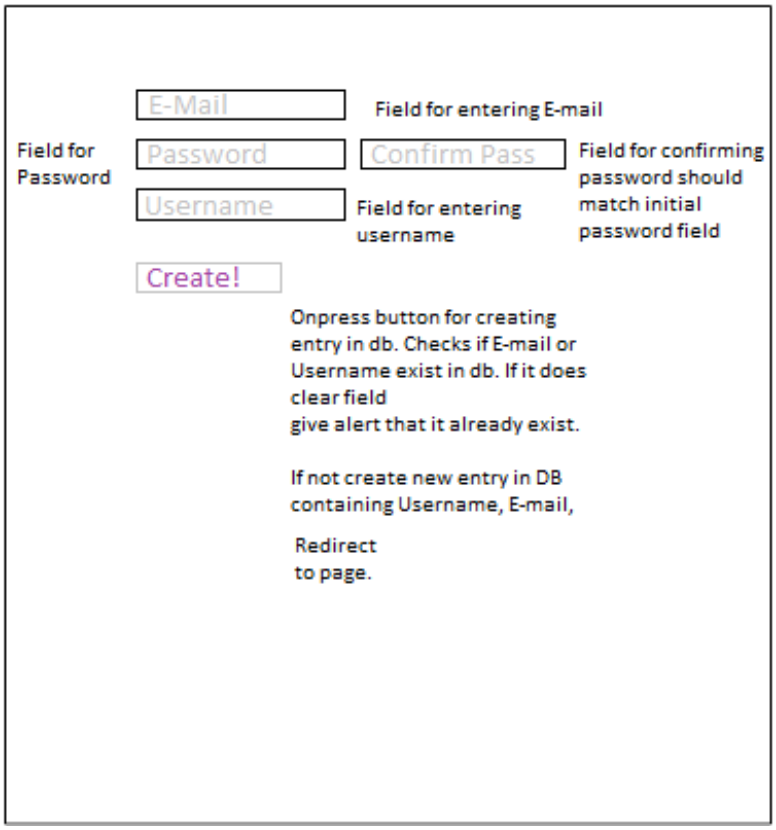
Concept Demo UML Design Front & Back Ends



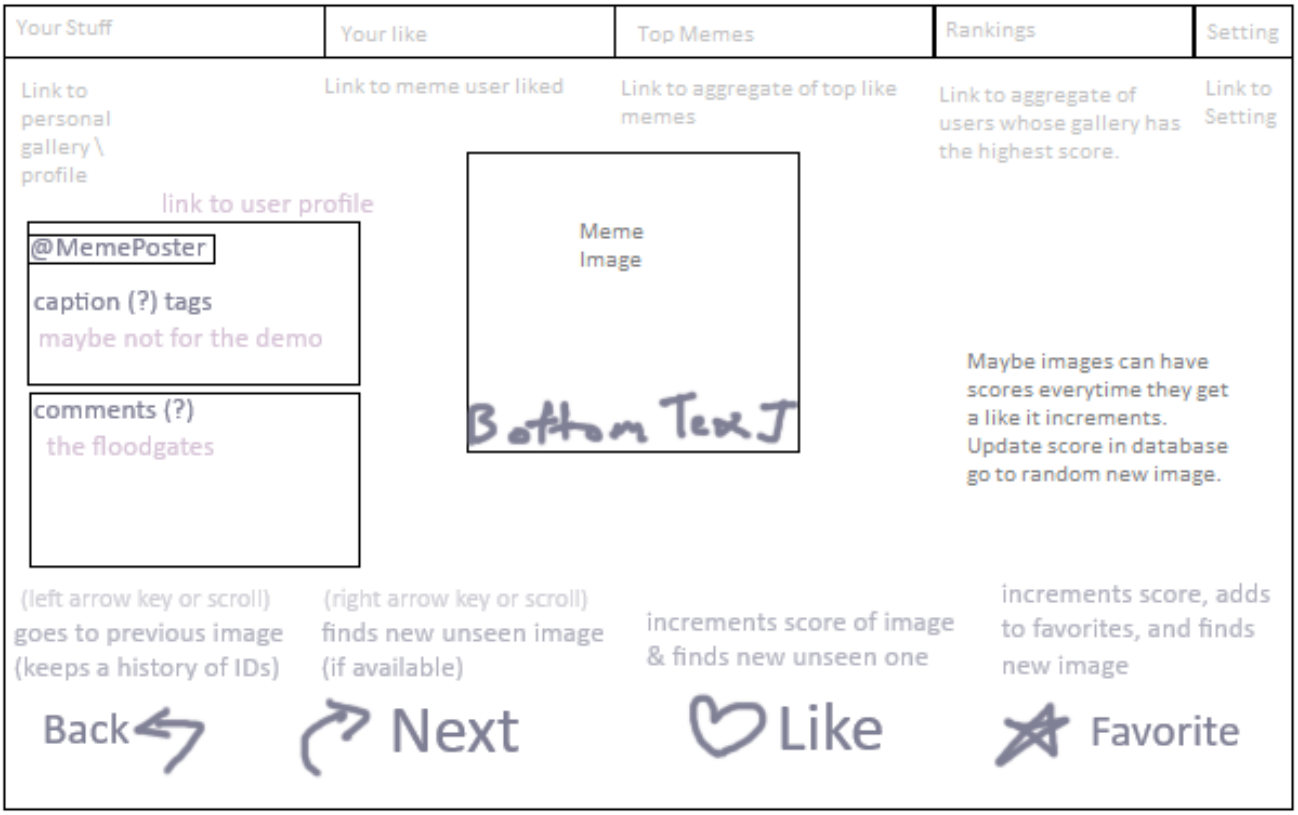


Initial interface design concepts

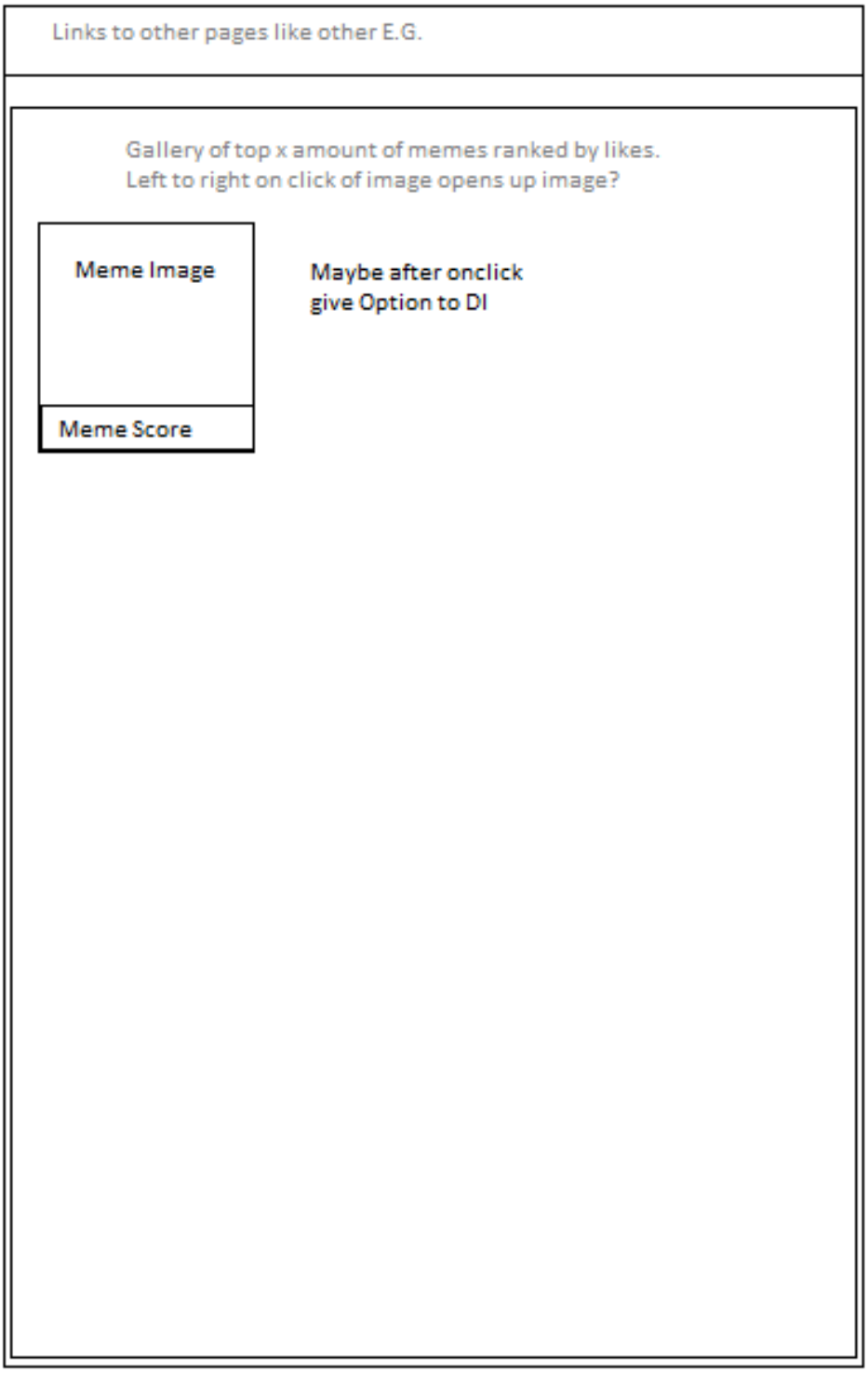
Registration Page



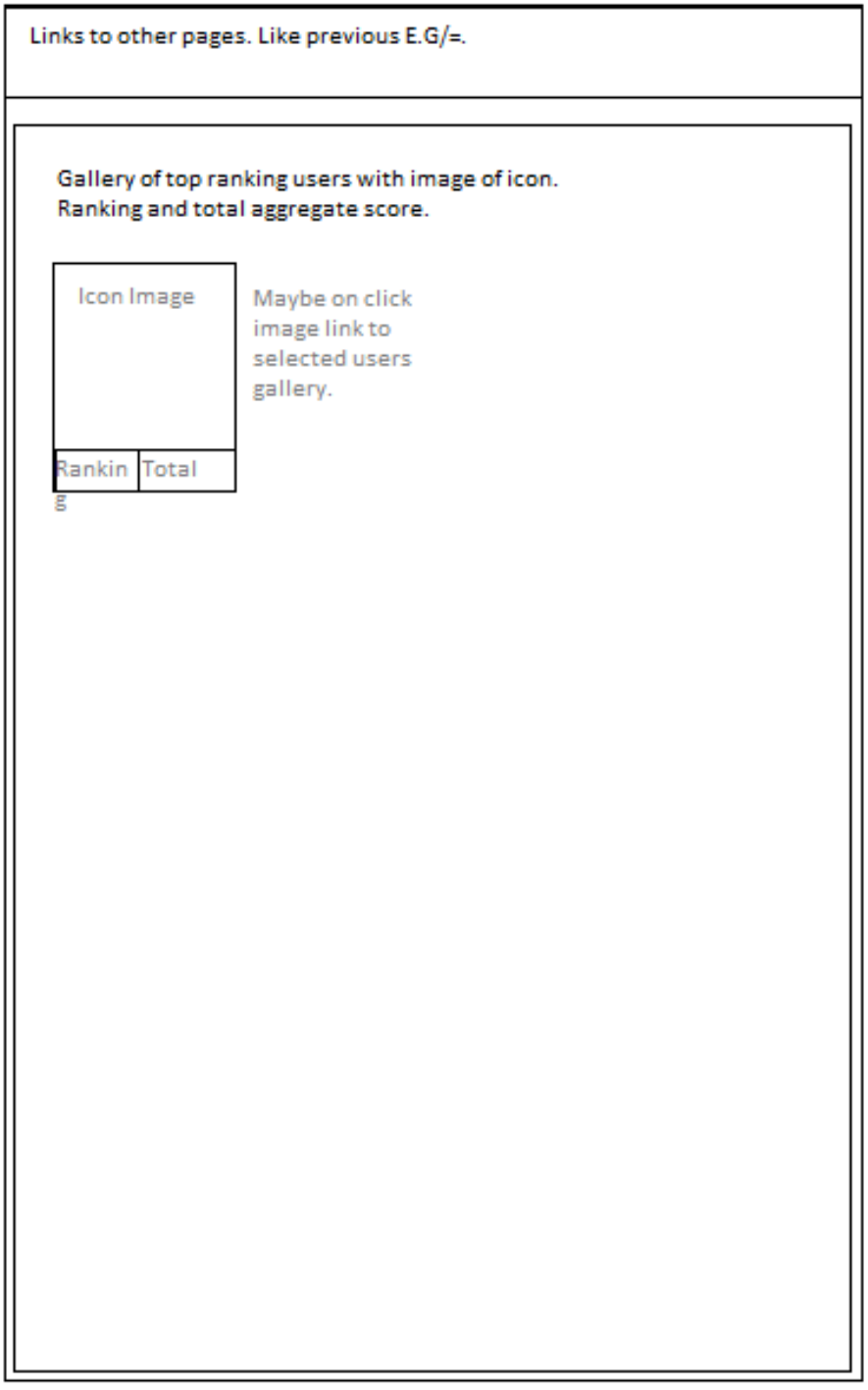
Front Page


Your Stuff
Your like
Top Memes
Rankings
Setting
Link to personal gallery \ profile
Link to meme user liked
Link to aggregate of top like memes
Link to aggregate of users whose gallery has the highest score.
Link to Setting
link to user profile
@MemePoster
caption (?) tags
maybe not for the demo
comments (?)
the floodgates
Meme Image
Bottom Text
Maybe images can have scores everytime they get a like it increments. Update score in database go to random new image.
(left arrow key or scroll)
goes to previous image
(keeps a history of IDs)
(right arrow key or scroll)
finds new unseen image
(if available)
increments score of image & finds new unseen one
increments score, adds to favorites, and finds new image
Back
Next
Like
Favorite

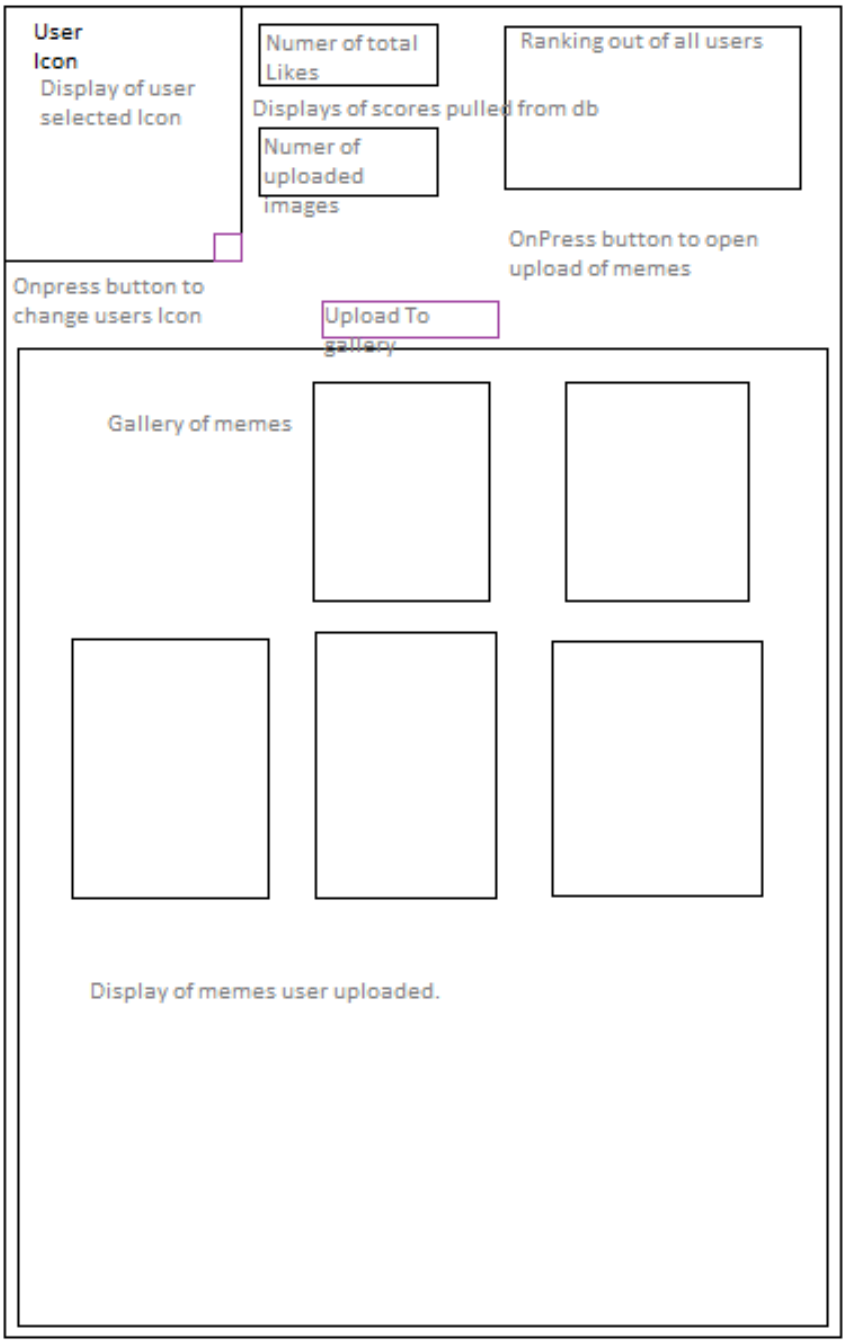
Liked Memes Page Page



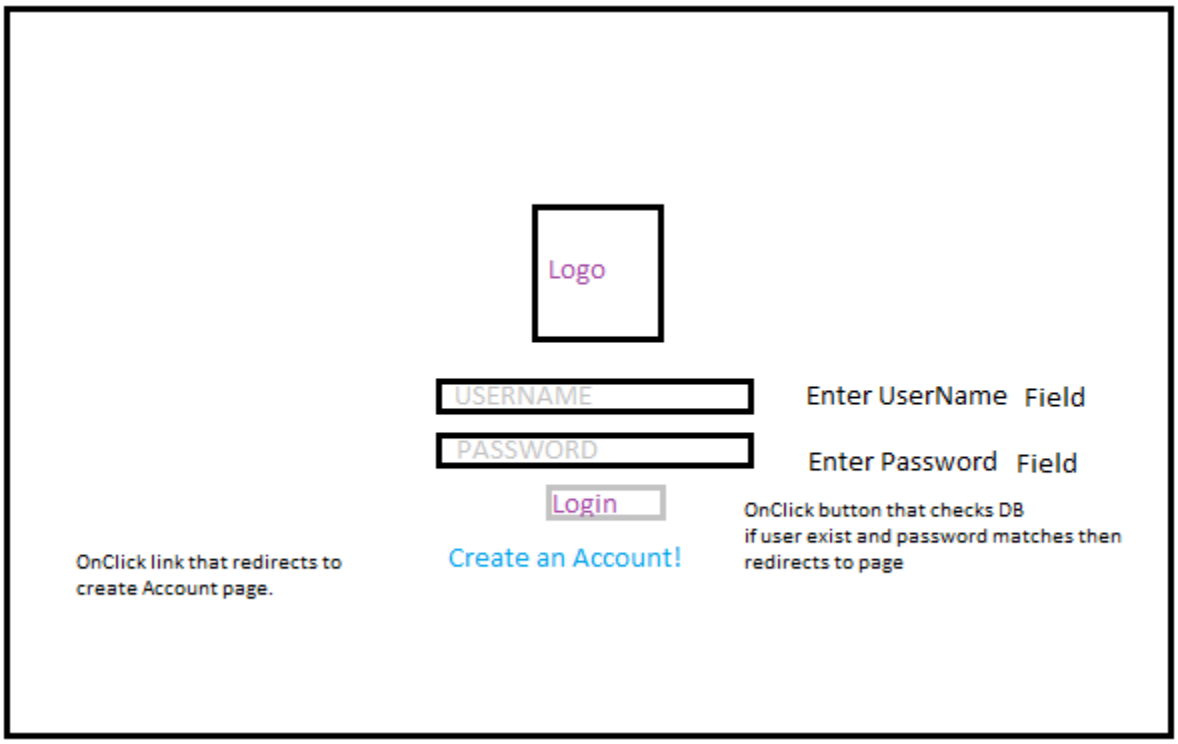
Top Ranked Memes Page



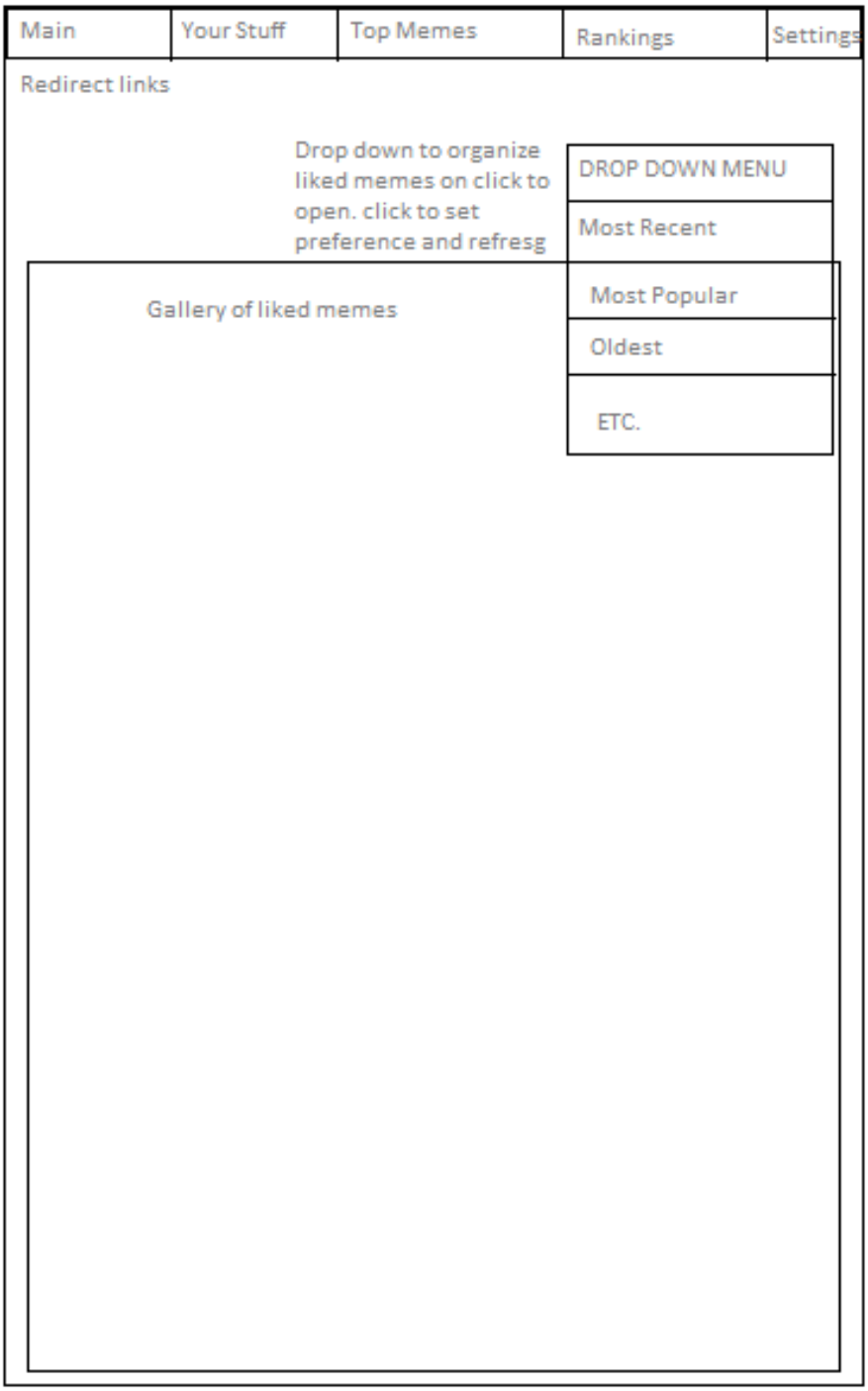
Profile Page



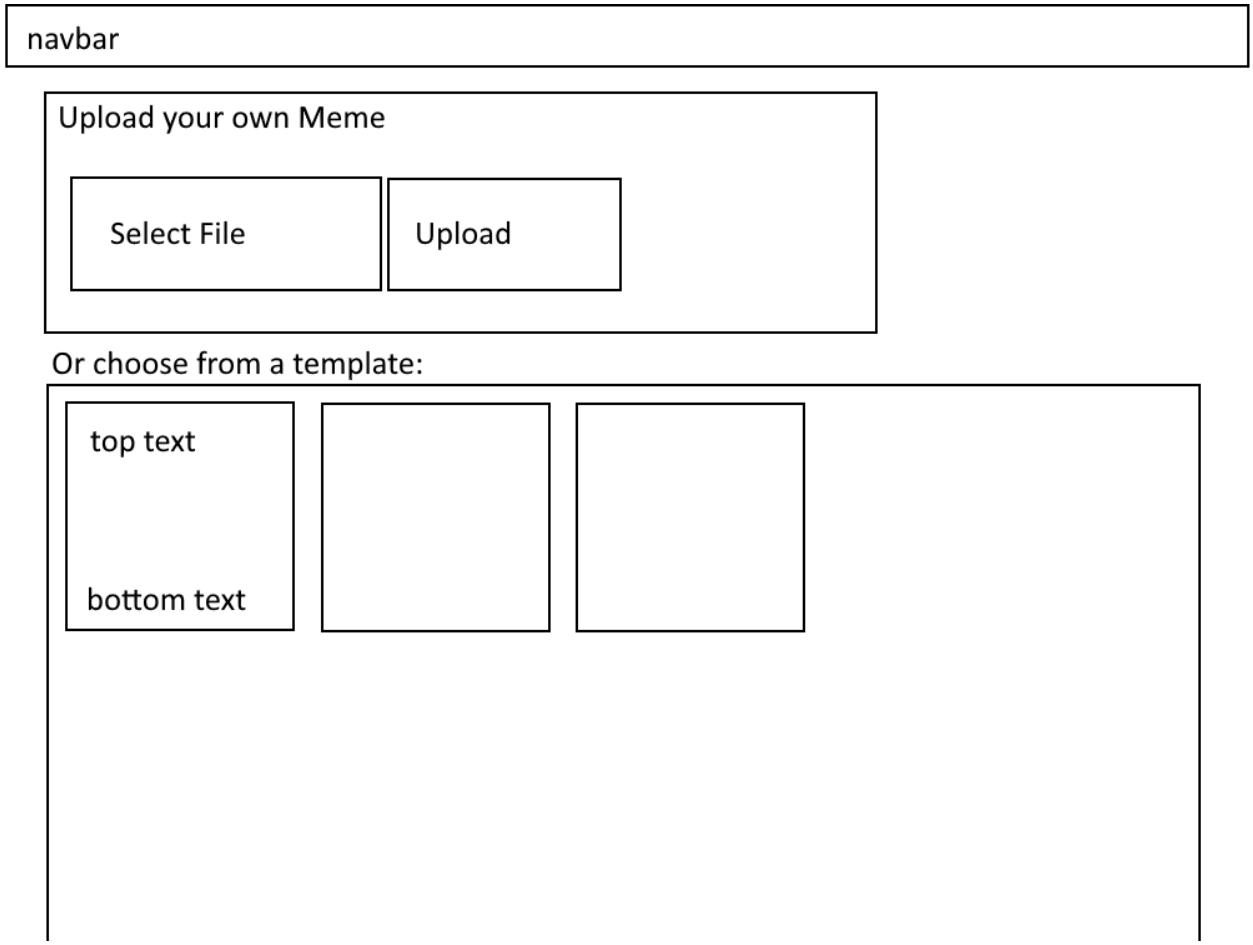
Front Page


Logo
USERNAME
Enter UserName Field
PASSWORD
Enter Password Field
Login
OnClick button that checks DB
if user exist and password matches then
redirects to page
OnClick link that redirects to
create Account page.
Create an Account!

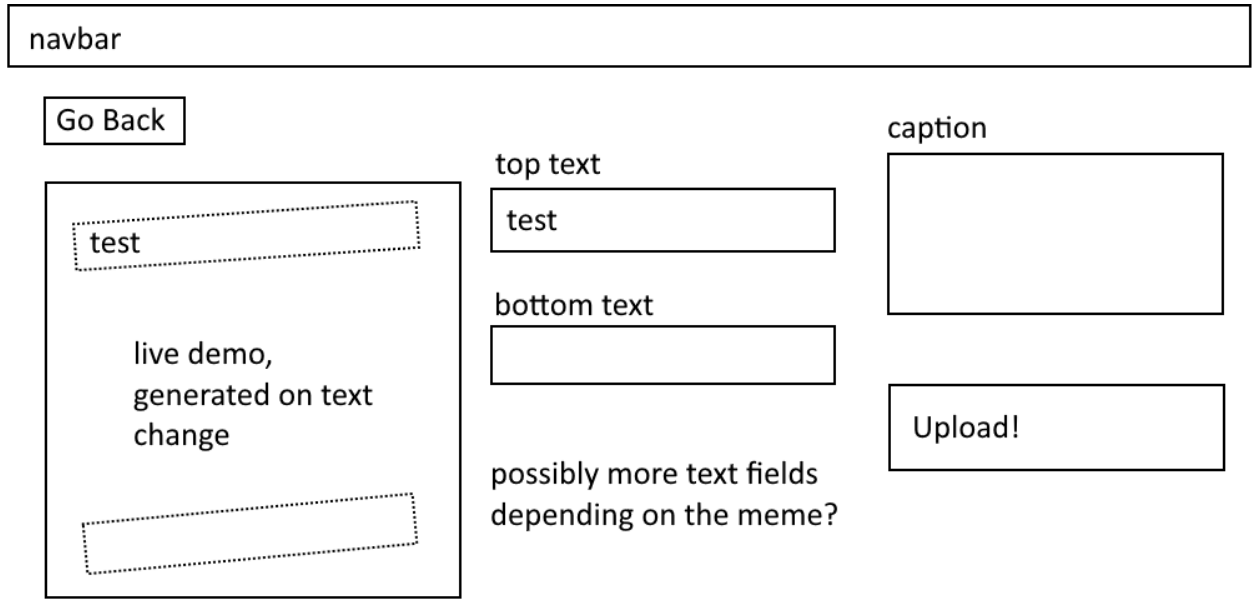
Liked Memes Page



Meme Maker Main Page

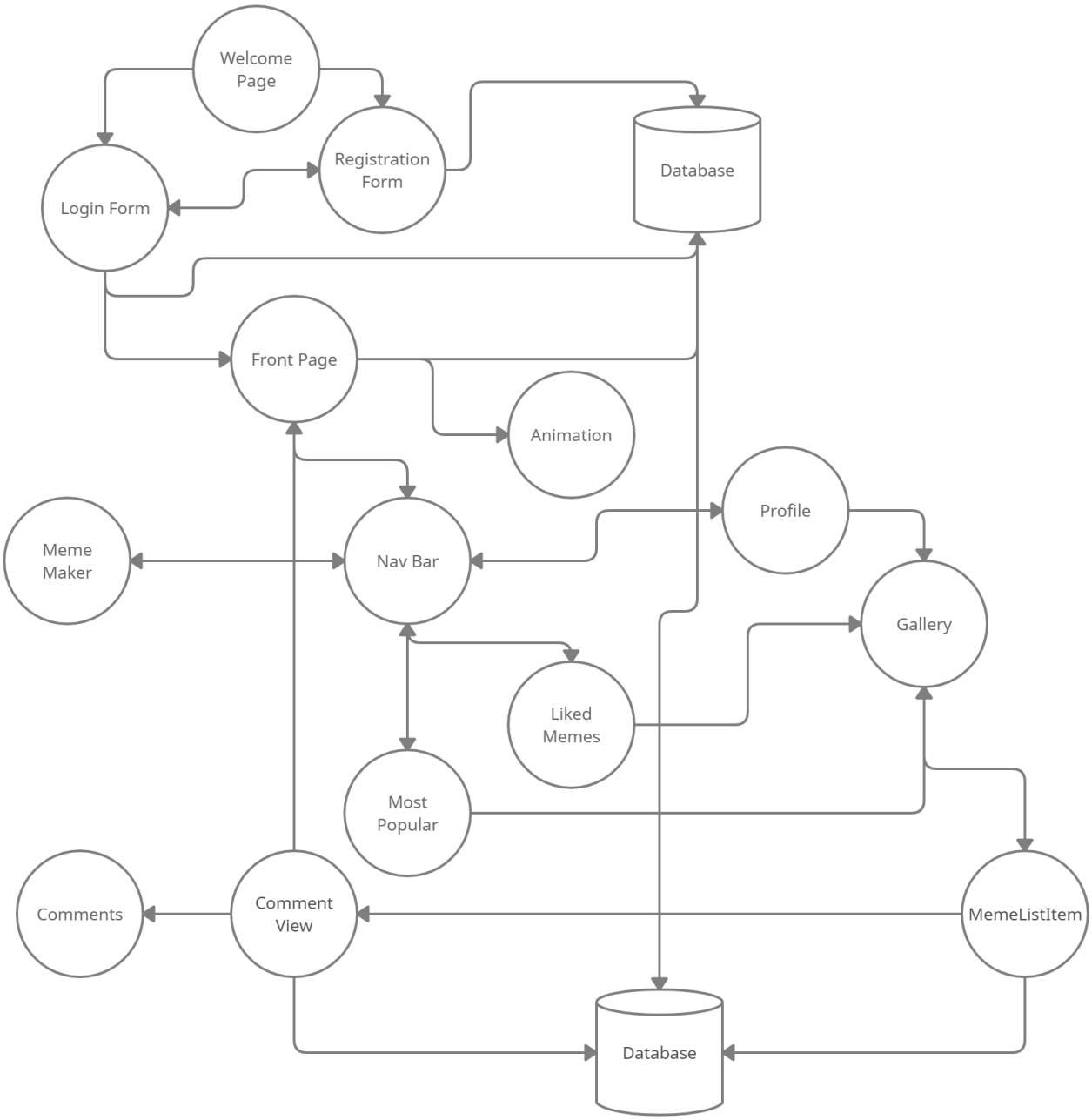


Meme Maker Template Page

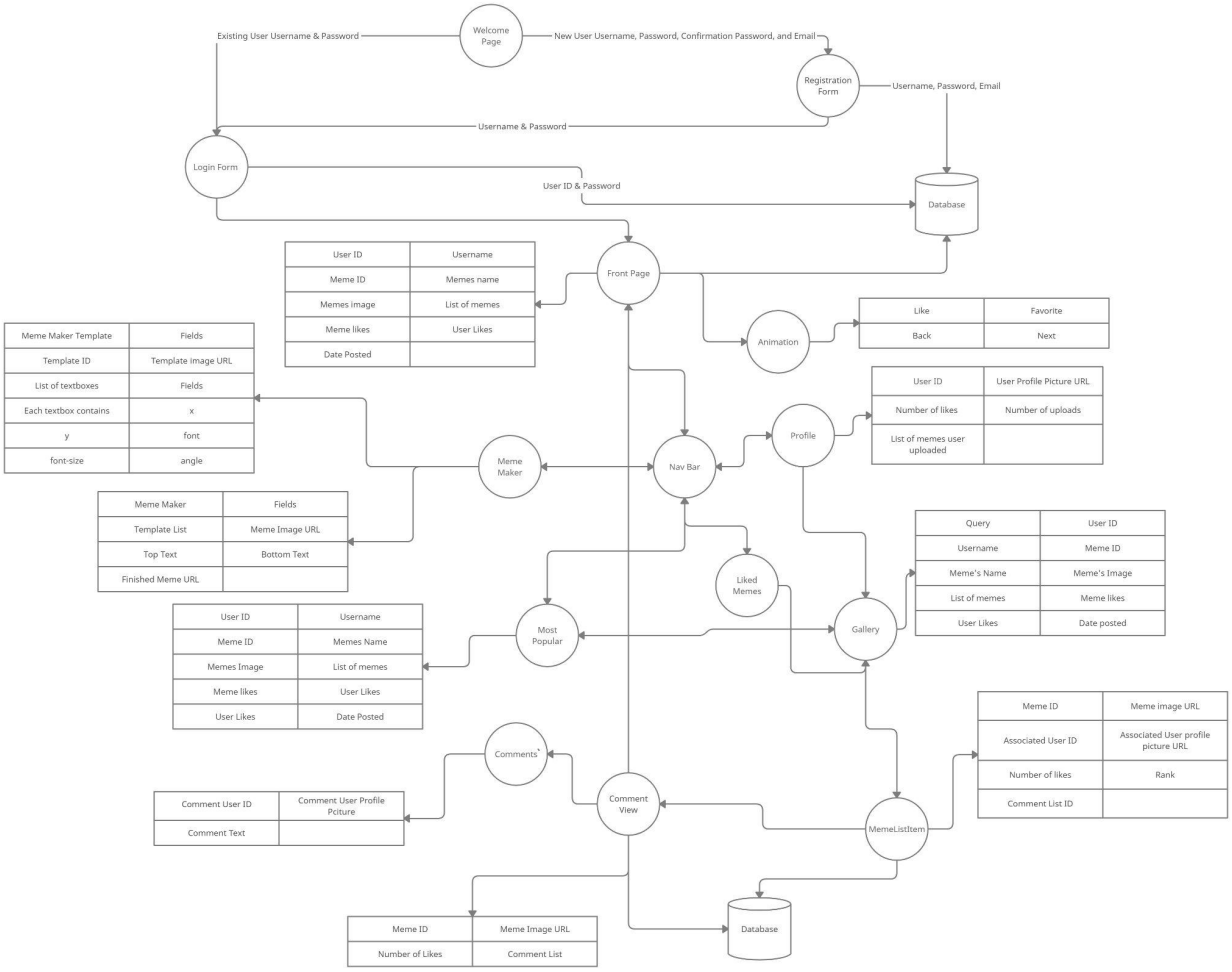


RDP Demo Designs

UML Level 0 Front End



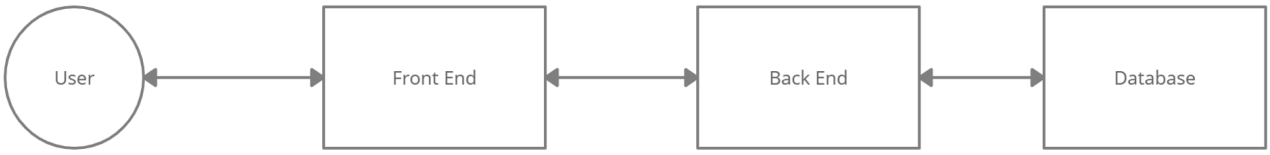
UML Level 1 Front End



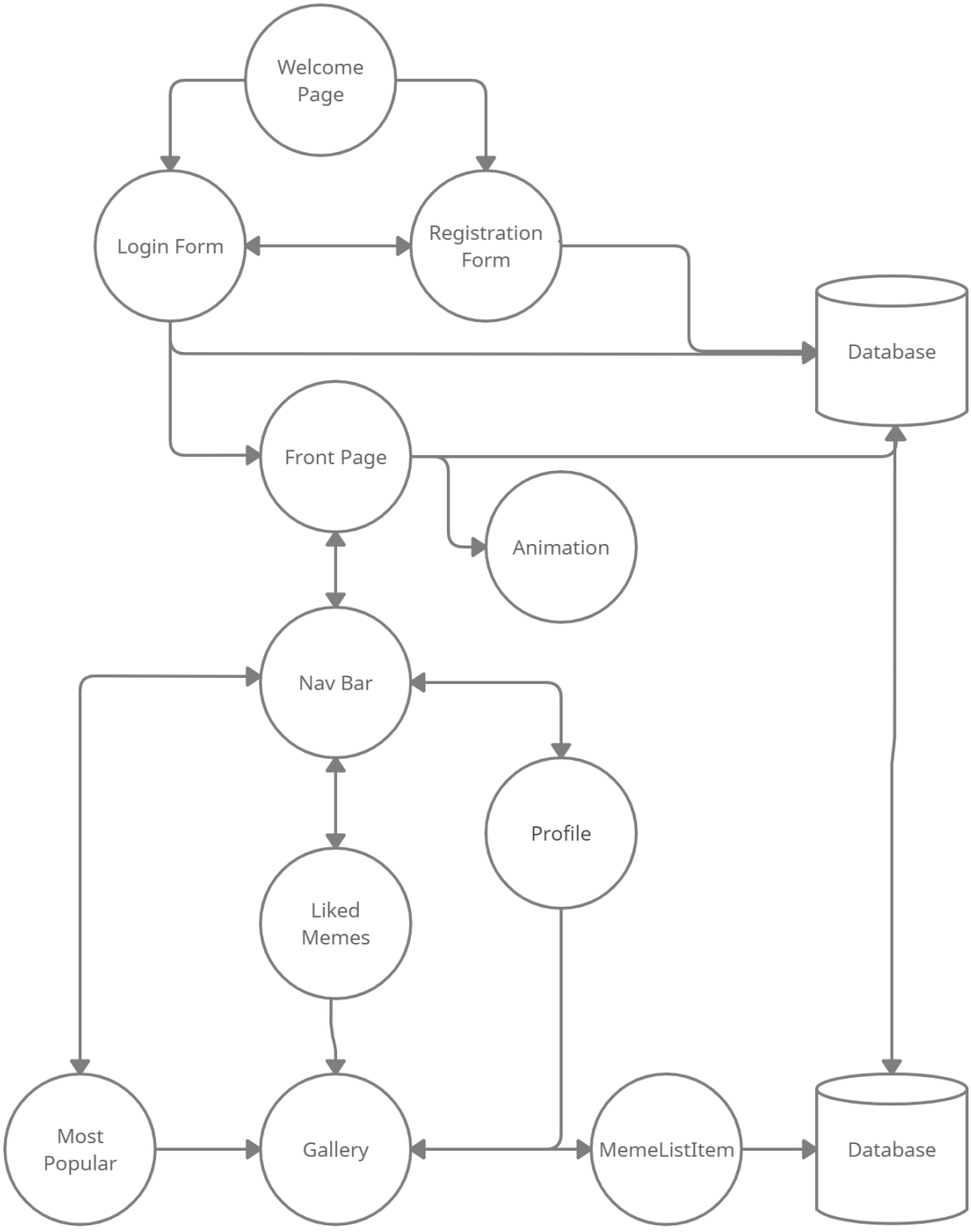
UML Level 0 & 1 Back End remained the same. Reference Pgs. 60 & 61

Final Demo Designs (For interactive UML Presentation)

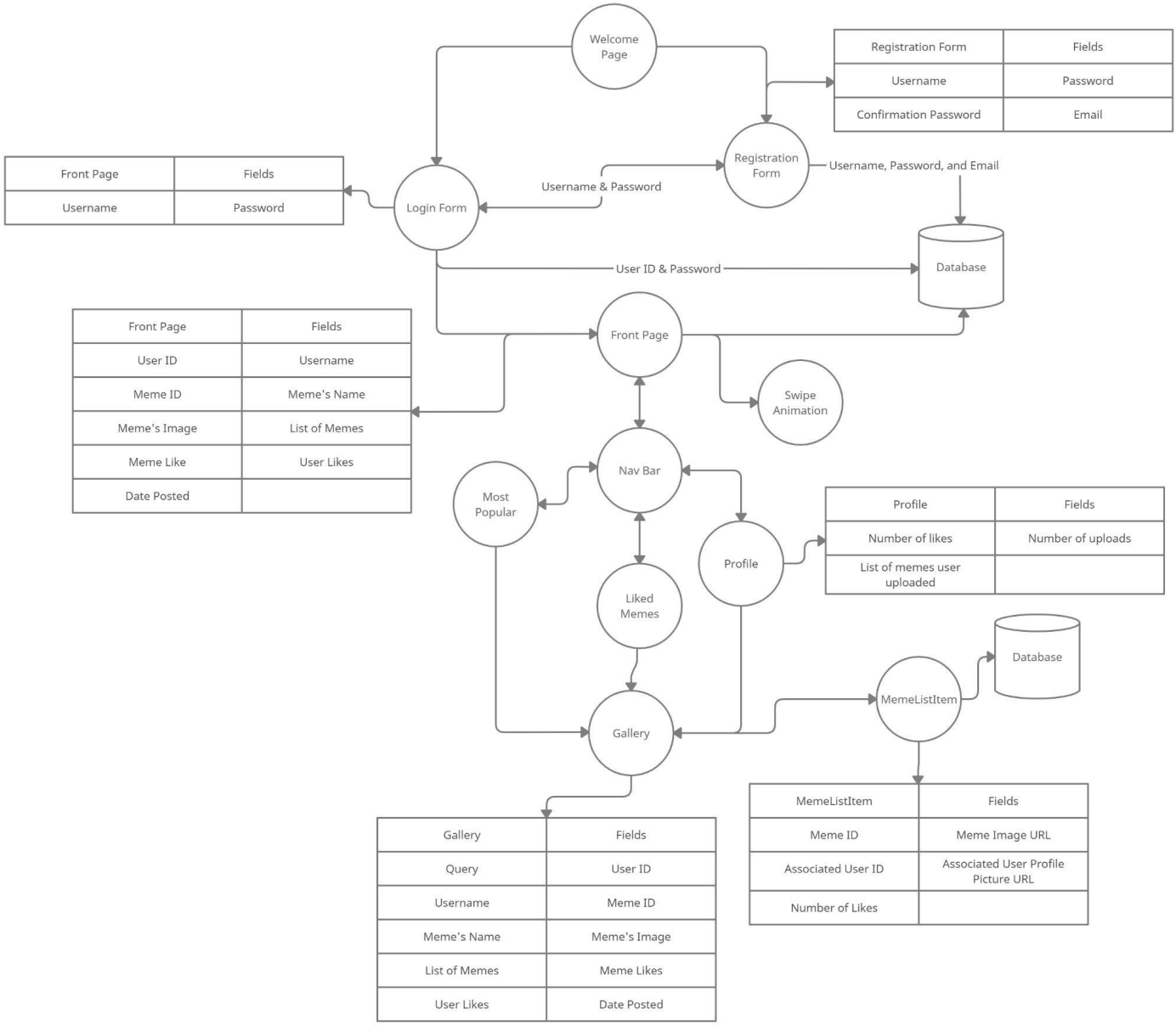
UML Level 0 Overview end to end



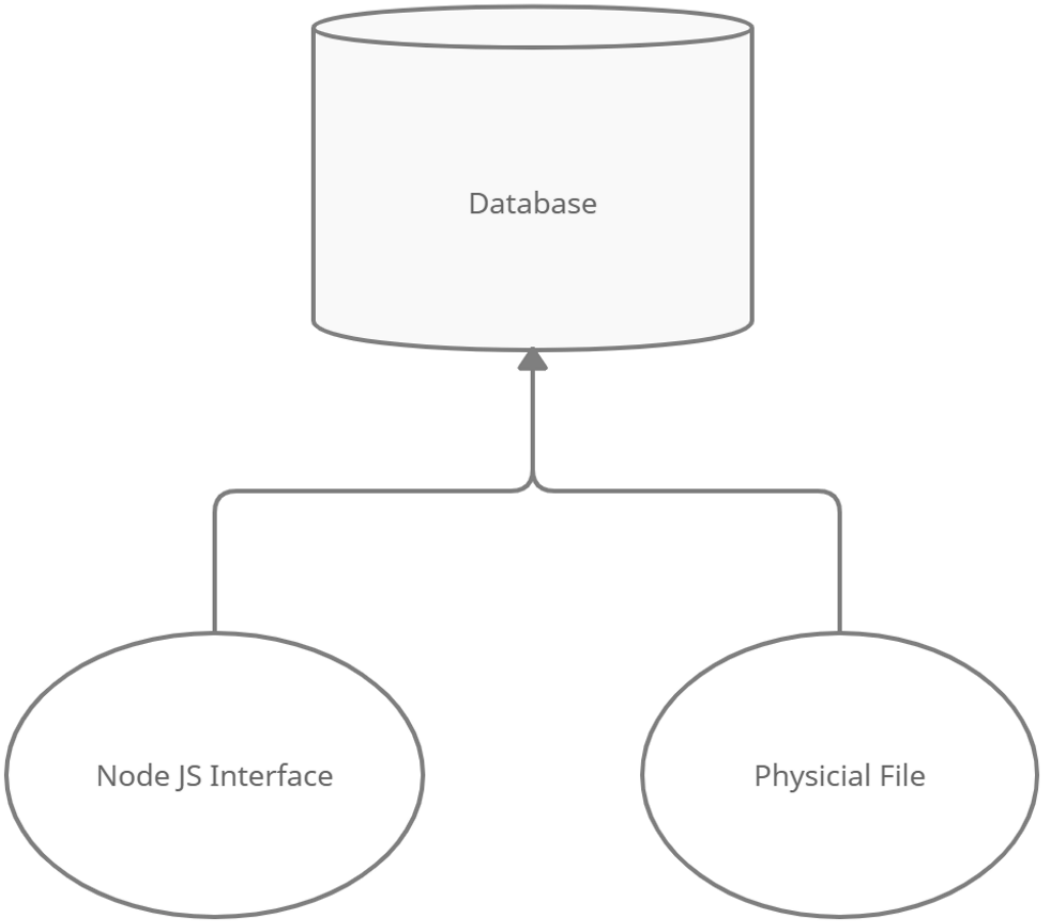
UML Level 0 Front End Overview



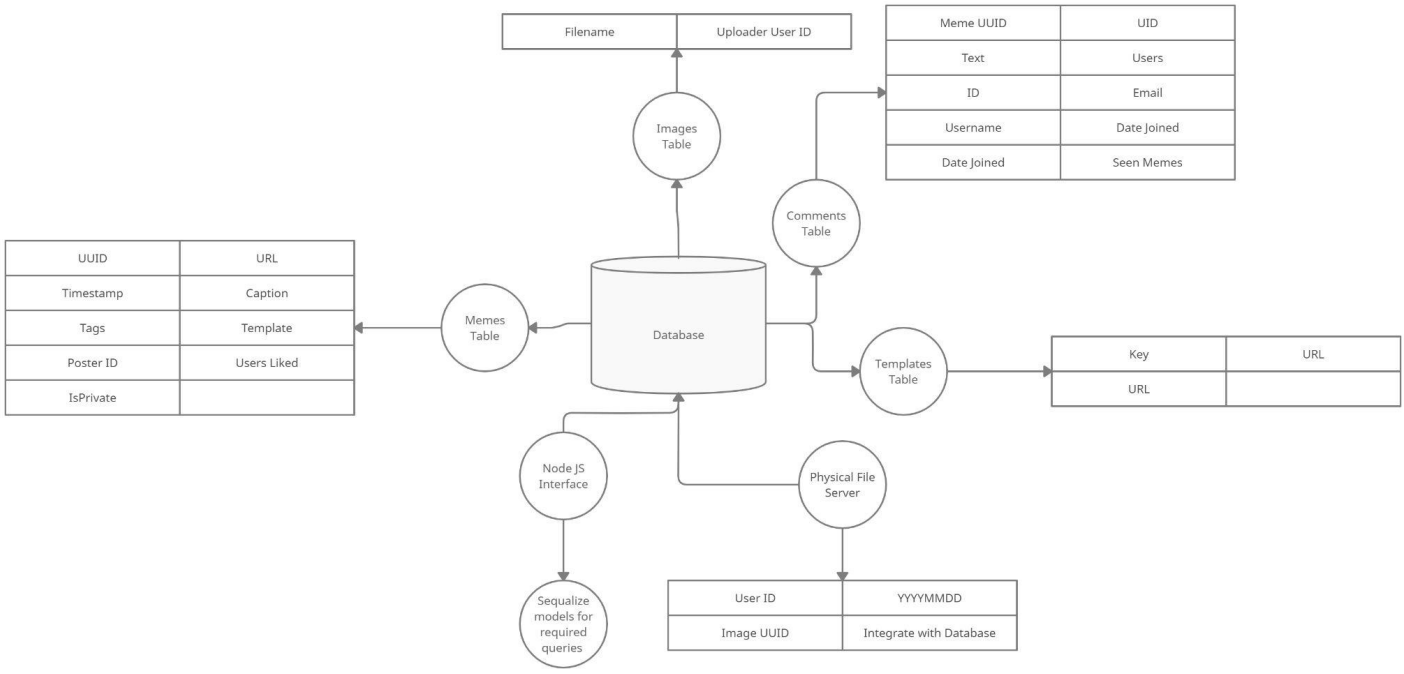
UML Level 1 Front End



UML Level 0 Back-end and Database Overview



UML Level 1 Back End and Database



Data:

- Cookies are stored on the client's browser, containing their user token and session id
- Folder structure
    - User-uploaded memes are uploaded to the "meme" folder.
    - Static builds of the React application are included in the "static" folder, with subfolders:
        - "css" - For compiled CSS files.
        - "js" - For compiled javascript chunks.
        - "media" - For image assets bundled with the application.

Program specifications:

- Programming languages to be used:
    - CSS, HTML, JavaScript, MySQL
- Rocky Linux 8.5; kernel 4.18.0
- Docker CE 20.10.12
    - nginx/1.21.3
    - MariaDB 10.6
    - MemeStone
        - "@testing-library/jest-dom": "^5.16.1"
        - "@testing-library/react": "^12.1.2"
        - "@testing-library/user-event": "^13.5.0"
        - "react": "^17.0.2",

        - "react-dom": "^17.0.2"
        - "react-redux": "^7.2.6"
        - "react-router-dom": "^6.2.1"
        - "react-scripts": "5.0.0"
        - "reactjs-popup": "^2.0.5"
        - "redux": "^4.1.2"
        - "redux-thunk": "^2.4.1"
        - "serve": "^13.0.2"
        - "use-clipboard-copy": "^0.2.0"
        - "web-vitals": "^2.1.3"
    - MemeStone Back end
        - "bcryptjs": "^2.4.3"
        - "cors": "^2.8.5"
        - "express": "^4.17.1"
        - "joi": "^17.2.0"
        - "mysql2": "^2.1.0"
        - "rootpath": "^0.1.2"
        - "sequelize": "^6.3.4"
- Database systems: MariaDB
- Browsers we plan to support
    - ***Chrome ^40, Firefox ^38, Opera ^2015, Safari ^11***
- A complete listing of the hardware to be used
    - Dell R510 rackmount server
    - NetGear ProSAFE JG5524 ethernet switch
    - NetGear ProSAFE JGS516PE POE ethernet switch
    - Intel X540-T2 ethernet adapters
    - Arris bgw210-700 modem
    - Personal development machines
    - Personal smartphones

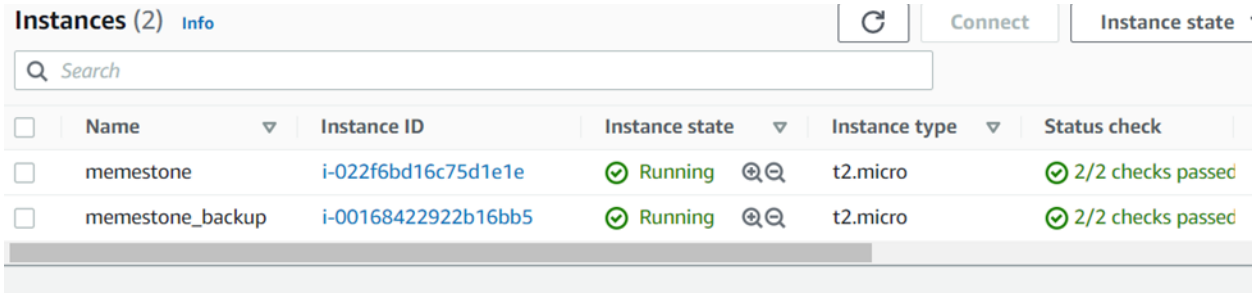
Security/Privacy:

- For our users' safety, The website will only be accessible using HTTPS.
- JSON web tokens will be used to verify user authentication for all API calls, and will expire after a set time.
- All user passwords will be salted and hashed, and we will not store or in any other way have access to users' plaintext passwords.

Server setup specifications:

- AWS Elastic Compute Cloud will be used to host the website.
    - Diverging from our original plans to use a self-hosted setup, AWS allows for simpler maintainability and cuts down costs for the scale of our application.
- The setup used will be the t2.micro ubuntu 64-bit server
- The server will pull from the aws_ec2 branch from our GitLab repository.

- The server will host the react frontend using Nginx.
- Mysql will be installed, using a custom user/password combination for security.
- The back end will be opened to port 8080.
- A certificate for use with HTTPS on redherringteam.xyz will be installed using letsencrypt certbot.
- One live backup server will be available to quickly switch to.



Other specifications:

- Multiple users will be able to use the site simultaneously without issue.
- Public user information and meme pages will be restricted from guests who are not logged-in.
- Errors will be handled with console messages and error pop-ups, directing the user to contact the Red Herring Team.

Final total number of hours for the project: 157 hours

Code

SwipeComponent.js
The swipe component controls the dating-app inspired swiping mechanics and animation. The memes are pulled from the database using the MemeService functions and compiled into a list of cards that can be swiped through.

The swipe component is one of our most robust components since it is the main component we show our website's users. It has to gather the memes we will show and also handle updating various stats that we track for each meme. The first thing we need to do is create an array that will keep track of the memes that we show. Our use of a Tinder-like swiping mechanic means that we only display one intractable meme at a time but we allow the user to either 'dislike' or 'like' a meme in order to view the next meme in our array or use the 'back' button to allow the user to view previously presented memes.

[Code start on next page]

1.

```
import React, { useState, useRef, useEffect } from "react";
import AuthService from "../services/auth.service";
import MemeService from "../services/meme.service";
import styles from "./css/TinderCards.module.css";
import TinderCard from "react-tinder-card";
import UserService from "../services/user.service";
import InteractionService from "../services/interaction.service";

var db = []

/*
* SwipeComponent is the home page when a user is currently logged in.
* Displays a list of recent memes to a user to swipe through in the form of dating app-style cards
*/
function SwipeComponent() {
  const baseUrl = "https://redherringteam.xyz:8080/files/";

  const [currentIndex, setCurrentIndex] = useState(db.length - 1);
  const [usernames] = useState([]);
  var currentUser = AuthService.getCurrentUser();
  // used for outOfFrame closure
  const currentIndexRef = useRef(currentIndex)


  const [childRefs, setChildRefs] = useState(
    () =>
      Array(db.length)
        .fill(0)
        .map((i) => React.createRef()),
    []
  )

  //Get list of memes from database
  useEffect(() => {
    MemeService.getNewMemesFor(currentUser.id).then((response) => {
      db = response.data;

      //Fix meme image urls urls to be full paths
      response.data.forEach(function(part, index) {
        part.url = baseUrl + part.url;

        //Usernames fetched from service can either be of type promise (for database polling) or string (if cached)
        let serviceResponse = UserService.getUserName(part.poster_id);
        if (serviceResponse instanceof Promise) serviceResponse.then((response) => { //Load from database
            usernames[index] = response.data[0].username;
            if (index === db.length - 1) updateCardReferences();
        });
        else { //Load from cache
          usernames[index] = serviceResponse;
          if (index === db.length - 1) updateCardReferences();
        }
      });
    });
  }, []);


```

2.

```
1
2   //Update card references
3   const updateCardReferences = () => {
4     setCurrentIndex(db.length - 1);
5     setChildRefs(() =>
6       Array(db.length)
7       .fill(0)
8       .map(() => React.createRef()),
9     []);
10  }
11
12  //Update current index in meme list
13  const updateCurrentIndex = (val) => {
14    setCurrentIndex(val)
15    currentIndexRef.current = val
16  }
17
18  const canGoBack = currentIndex < db.length - 1
19  const canSwipe = currentIndex >= 0
20
21  //Set last direction and decrease current index
22  const swiped = (direction, nameToDelete, index) => {
23    currentUser = AuthService.getCurrentUser();
24    if (!currentUser) alert("logged out");
25    updateCurrentIndex(index - 1);
26
27    let meme = db[index];
28    if (direction === "right") {
29      //Swipe right likes the current meme
30      InteractionService.submitLike(meme.id);
31    } else {
32      //Swipe left dislikes the current meme
33      InteractionService.submitDislike(meme.id);
34    }
35    //Mark this meme as viewed so it will not be shown again
36    InteractionService.markViewed(meme.id);
37  }
38
39  //Called when a card leaves the frame
40  const outOfFrame = (name, idx) => {
41    currentIndexRef.current >= idx && childRefs[idx].current.restoreCard()
42  }
43
44  //Swipes a card (used by buttons)
45  const swipe = async (dir) => {
46    if (canSwipe && currentIndex < db.length) {
47      await childRefs[currentIndex].current.swipe(dir) // Swipe the card!
48    }
49  }
50
```

The above code handles all of our functions needed to collect, display, and interact with the memes we display. We collect the Username of the user who created the meme so that we are able to display it along with the meme. We also handle the positioning of each meme by keeping track of the current index of the meme we are viewing so that we can handle moving forward and backward in the array of memes we gathered. We update our position forward in the array every time we either dislike or like a meme. To register if a user likes and dislikes a meme we make a call to the interaction.service.js file with the proper function and meme.id. If a meme is 'swiped' we move forward in the array and if a user uses the 'Back' button then we move backward in the array updating our current index to the previous one.

3.

```
1
2   //Increase current index and show card
3   const goBack = async () => {
4     if (!canGoBack) return
5     const newIndex = currentIndex + 1
6     updateCurrentIndex(newIndex)
7     await childRefs[newIndex].current.restoreCard()
8   }
9
10  return (
11    <div className="content">
12      <link
13        href='https://fonts.googleapis.com/css?family=Damion&display=swap'
14        rel='stylesheet'
15      />
16      <link
17        href='https://fonts.googleapis.com/css?family=Alatsi&display=swap'
18        rel='stylesheet'
19      />
20        <div className={styles.cardContainer}>
21          {db.map((meme, index) => (
22            <TinderCard
23              ref={childRefs[index]}
24              className={styles.swipe}
25              key={index}
26              preventSwipe={["up", "down"]}
27              onSwipe={(dir) => swiped(dir, meme.name, index)}
28              onCardLeftScreen={() => outOfFrame(meme.name, index)}
29            >
30              <div className={styles.card}>
31                <img src={meme.url} className={styles.cardImage} alt={usernames[index] + "'s meme"}></img>
32                <h3>@{usernames[index]}</h3>
33                <h3 className={styles.memeLikes}>♥ {meme.likes} like{(meme.likes === 1)?"":"s"}</h3>
34              </div>
35            </TinderCard>
36          ))}
37        </div>
38        <div className={styles.buttons}>
39          <button style={{ backgroundColor: !canSwipe && '#c3c4d3' }} onClick={() => swipe('left')}>Dislike
    Meme</button>
40          <button style={{ backgroundColor: !canGoBack && '#c3c4d3' }} onClick={() => goBack()}>Go Back</button>
41          <button style={{ backgroundColor: !canSwipe && '#c3c4d3' }} onClick={() => swipe('right')}>Like
    Meme</button>
42        </div>
43    </div>
44  )
45 }
46
47 export default SwipeComponent;
```

This code calls our swipe component and gives it it's visual swiping component from an external file called TinderCard which we imported at the start of this file.

MemeGallery.js
The meme gallery is a front-end component used in the Top Ranked, Liked Memes, and Profile pages. It displays a grid of memes split into pages to allow for easy navigating. The page components create a Meme Gallery using a page type parameter, which changes what request will be sent to the memes API.

The Meme Gallery component has three different page types that correspond to the three different pages it is used with: TOP_MEMES, LIKED_MEMES, and PROFILE. By calling this component with either of the page types then we can change which memes we display in a common gallery view to achieve visual consistency but still having modularity.

[Code start on next page]

1.

```
import React, {Component} from "react";
import { Redirect } from "react-router-dom";
import AuthService from "../services/auth.service";
import MemeService from "../services/meme.service";
import ListMeme from './MemeListItem';
import Masonry, {ResponsiveMasonry} from "react-responsive-masonry";
import EventBus from "../common/EventBus";
import styles from "./css/MemeGallery.module.css";

export const PageType = {
    TOP_MEMES: 1,
    LIKED_MEMES: 2,
    PROFILE: 3
}

/*
* The Meme Gallery component displays a list of memes
* Properties:
*   pageType - indicates which query to send to the database, whether it be for the top memes, liked memes, or memes by user
*   byUser - only applies to PageType.PROFILE, which user to fetch data for
*/
export default class MemeGallery extends Component {
    numItemsPerPage = 20;

    constructor(props) {
        super(props);

        this._isMounted = false;

        this.state = {
            redirect: null,
            ready: 0,
            currentUser: {userID: -1},
            pageType: props.pageType,
            byUser: props.byUser,
            pages: [],
            page: 0,
            pageCount: 0
        };
    }

    componentDidMount() {
        this._isMounted = true;
        const baseUrl = "https://redherringteam.xyz:8080/files/";

        //Redirect to home if logged out
        const user = AuthService.getCurrentUser();
        if (!user) {
            this.setState({ redirect: "/" });
            return;
        }
        const byUser = this.state.byUser ? this.state.byUser : user.id;

```

2.

```

        //Request memes from database API
        var data;
        switch(this.state.pageType) {
            case PageType.TOP_MEMES:
                data = MemeService.getTopMemes();
                break;
            case PageType.LIKED_MEMES:
                data = MemeService.getLikedMemes();
                break;
            case PageType.PROFILE:
                data = MemeService.getMemesByUser(byUser);
                break;
            default:
                data = MemeService.getMemes();
        }
        //Handle response from API
        data.then((response) => {
            //Append full path to image URLS
            response.data.forEach(function(part) {
                part.url = baseUrl + part.url;
            });

            //Split list of memes into pages
            let count = Math.ceil(response.data.length / this.numItemsPerPage)
            let array = []
            for (let i = 0; i < count; i++) {
                array.push(response.data.splice(0, this.numItemsPerPage));
            }

            this._isMounted && this.setState({
                pages: array,
                pageCount: count - 1,
                ready: this.state.ready + 1
            });
        },
        error => {
            //Handle errors
            alert((error.response &&
                  error.response.data &&
                  error.response.data.message) ||
                error.message ||
                error.toString());

            if (error.response && error.response.status === 401) {
              EventBus.dispatch("logout");
            }
        });

        this._isMounted && this.setState({currentUser: user, byUser: byUser, ready: this.state.ready + 1});
    }

    //Mark component as unmounted in order to prevent updating state on unmounted component
    componentWillUnmount() {
        this._isMounted = false;
    }

    //Remove a meme from the list, either by deletion or disliking on the liked memes page
    removeMeme = (index) => {
        let memes = this.state.pages[this.state.page];
        memes.splice(index, 1);
        this._isMounted && this.setState({
            pages: this.state.pages
        });
    }

```

The Meme Gallery receives the memes we wish to display by making a call to our meme.service.js file. The meme.service.js file is our Database API file that connects our front end to our back end by asking for specific calls from our database. After receiving the memes that we call for, the Meme Gallery will compile them all into a separate lists to create our various pages (this is done to keep the amount of memes shown at one time to 20). For now we create Meme Gallery pages as needed based on how many memes we have in total in our database.

3.

```
    //Change page
    changePage = (newPage) => {
        newPage = Math.max(0, Math.min(newPage, this.state.pageCount));
        this._isMounted && this.setState({
            page: newPage
        });
        //Scroll back to top
        window.scrollTo(0, 0);
    }

    render() {
        if (this.state.redirect) { //Redirect if not authenticated
            return <Redirect to={this.state.redirect} />
        }

        const { currentUser, pages, ready, page, pageCount} = this.state;
        var memes = pages[page];

        if (ready > 1) //If both user and meme content is loaded
        return (
            <div>
                <ResponsiveMasonry columnsCountBreakPoints={{350: 1, 750: 2, 900: 3, 1400: 4, 2000: 5}}>
                    <Masonry>
                        { memes.map((meme, index) =>
                        <ListMeme
                            meme={meme}
                            pageType={this.state.pageType}
                            index={index}
                            key={meme.id}
                            removeMeme={this.removeMeme}
                            currentUser={currentUser}/>) }
                    </Masonry>
                </ResponsiveMasonry>
                {pageCount > 0 ?
                <div className={styles.pageDiv}>
                    {page > 0 ?
                    <button className={styles.prevButton + " btn btn-primary"} onClick={() =>
this.changePage(page - 1)}>Previous</button>
                    : null}
                    <span className={styles.pageNumber}>{page + 1}</span>
                    {page < pageCount ?
                    <button className={styles.nextButton + " btn btn-primary"} onClick={() =>
this.changePage(page + 1)}>Next</button>
                    : null}
                </div>
                : null}
            </div>
        );
        else return null;
    }
}
```

This code will create the gallery by calling our ListMeme function within our MemeListItem.js file for each meme in our current page's list. This will allow us to display each meme, the meme creator's username, the ability to like a meme (or dislike one), the amount of likes it has, and the ability to delete a meme if you are the owner of the meme. It also dynamically creates the navigation buttons needed to go forward or backward among our Meme Gallery pages.

MemeListItem.js
This component represents an individual meme in the meme gallery. This displays the meme itself, as well as the username of the poster and the amount of likes it has and a button to like/unlike the meme. If the meme was posted by the currently logged in user, it additionally displays a button that allows that user to delete it, which sends a post request to the database to remove the meme as well as a signal to the Meme Gallery to remove it from the list of currently shown memes.

[Code start on next page]

1.

```
import React, {Component} from "react";
import { Link } from "react-router-dom";
import styles from './css/MemeListItem.module.css';
import InteractionService from "../services/interaction.service";
import UserService from "../services/user.service";
import {PageType} from './MemeGallery';

/*
* MemeListItem represents an individual meme shown in the MemeGallery
* Properties:
*   meme
*   currentUser
*   pageType
*/
export default class MemeListItem extends Component {
    removeMeme;
    index;

    constructor(props) {
        super(props);

        this._isMounted = false;
        this.removeMeme = props.removeMeme;
        this.index = props.index;

        this.state = {
            currentUser: this.props.currentUser,
            meme: this.props.meme,
            pageType: this.props.pageType,
            isLiked: 0,
            username: ""
        };
    }

    //Retrieve if the meme is liked by current user and fetch the username from the ID
    componentDidMount() {
        this._isMounted = true;
        InteractionService.isMemeLikedBy(this.state.meme.id).then((response) => {
            this._isMounted && this.setState({isLiked: response.data.length === undefined});
        });

        //Usernames fetched from service can either be of type promise (for database polling) or string (if cached)
        let serviceResponse = UserService.getUserName(this.state.meme.poster_id);
        if (serviceResponse instanceof Promise) serviceResponse.then((response) => {
            this._isMounted && this.setState({username: response.data[0].username});
        });
        else this._isMounted && this.setState({username: serviceResponse});
    }

    //Mark component as unmounted in order to prevent updating state on unmounted component
    componentWillUnmount() {
        this._isMounted = false;
    }

```

2.

```
    //Toggle liked status of the meme for the current user in the database
    LikeMeme = () => {
        let newMeme = Object.assign({}, this.state.meme);
        if (this.state.isLiked) {
            InteractionService.submitDislike(this.state.meme.id);
            newMeme.likes--;
            if (this.state.pageType === PageType.LIKED_MEMES) this.removeMeme(this.index);
        } else {
            InteractionService.submitLike(this.state.meme.id);
            newMeme.likes++;
        }
        this._isMounted && this.setState({isLiked: !this.state.isLiked, meme: newMeme});
    }

    //Delete meme from the database
    DeleteMeme = () => {
        InteractionService.deleteMeme(this.state.meme.id);
        this.removeMeme(this.index);
    }

    render() {
        const {
            meme,
            isLiked,
            username
          } = this.state;

        return (
            <div className={styles.memeDiv}>
                <img src={meme.url} className={styles.memeImage} alt=""></img>
                <div>
                    <Link to={"/profile?user="+username}><span className={styles.memeUser}>@{username}</span>
</Link>
                    <span className={styles.memeLikes}>♥ {meme.likes} like{(meme.likes === 1)?"":"s"}</span>
                    <button className={styles.likeButton} onClick={this.LikeMeme}>{(isLiked) ? "unlike" :
"like"}</button>
                    {this.state.currentUser.username === username ?
                    <button className={styles.deleteButton} onClick={this.DeleteMeme}>delete</button>: null }
                </div>
            </div>
        );
    }
}
```

Interaction.controller.js

On the backend, the interaction controller handles the API post requests for interaction, including liking memes, disliking memes, marking memes as viewed, and deleting memes.

[Code start on next page]

1.

```
const { Sequelize, sequelize } = require("../models");
const db = require("../models");
const Meme = db.meme;
const Likes = db.likes;
const Viewed = db.viewed;

//Check if a meme is liked by the current user
const getLikes = (req, res) => {
  if (req.query.meme) {
      Likes.findOne({
        where: {
          userID: req.userId,
          memeID: req.query.meme
        }
      }).then(memes => res.status(200).send(memes));
  } else res.status(400).send("missing meme parameter");
};

//Submit a like for a specific meme on the server
const submitLike = async (req, res) => {
  const entry = await Likes.findOne({
    where: {
      userID: req.userId,
      memeID: req.body.meme
    }
  });

  if (entry === null) { //Not already liked
    Likes.create({
      userID: req.userId,
      memeID: req.body.meme
    });
    //Increment like count for meme
    Meme.increment('likes', {by: 1, where: { id: req.body.meme }});
  }
}

//Submit a dislike for a specific meme on the server
const submitDislike = async (req, res) => {
  const entry = await Likes.findOne({
    where: {
      userID: req.userId,
      memeID: req.body.meme
    }
  });

  if (entry !== null) { //Already liked
    Likes.destroy({
      where: {
        userID: req.userId,
        memeID: req.body.meme
      }
    });
    //Decrement like count for meme
    Meme.decrement('likes', {by: 1, where: { id: req.body.meme }});
  }
}

```

2.

```
1
2 const viewMeme = async (req, res) => {
3   //Mark meme as viewed
4   Viewed.findOrCreate({
5     where: {
6       userID: req.userId,
7       memeID: req.body.meme
8     },
9     defaults: {
10      userID: req.userID,
11      memeID: req.body.meme
12    }
13  });
14 }
15
16 //Delete a meme from the database
17 const deleteMeme = async (req, res) => {
18   Meme.destroy({
19     where: {
20       poster_id: req.userId,
21       id: req.body.meme
22     }
23   }).then(() => {
24     //Remove references from likes table if the delete succeeded
25     Likes.destroy({
26       where: {
27         memeID: req.body.meme
28       }
29     })
30   });
31 }
32
33 module.exports = {
34   getLikes,
35   submitLike,
36   submitDislike,
37   viewMeme,
38   deleteMeme
39 };
```

Profile.component.js

The Profile.component.js file is a front end page component that displays a given user profile (either the currently logged in user or one provided in the URL query). The page contains statistics about the user such as the amount of likes they have received across all memes as well as the amount of memes they have uploaded, as well as a meme gallery showing the memes they have uploaded.

By making this component we are able to create user profiles for all the accounts in our database dynamically. This allows a user to click on a username on our site which will then take you a page showing all their memes and statistics. We have a dedicated link that will redirect to the currently logged in user's profile page to allow a user to access and delete their own memes easily. Currently we always call this component with the meme gallery component simultaneously which we can change if we add in more functionalities at a later time.

1.

```
import React, { Component } from "react";
import { Redirect } from "react-router-dom";
import AuthService from "../services/auth.service";
import UserService from "../services/user.service";
import MemeGallery, {PageType} from './MemeGallery';
import styles from "./css/UserProfile.module.css";
import profilePicture from '../assets/profile.png';
import statsService from "../services/stats.service";

/*
* Profile Page, showing statistics and memes uploaded by a user
* If there is no username specificied in the query (e.g. /profile?user=username), then it shows the currently logged in user
*/
export default class Profile extends Component {
  constructor(props) {
    super(props);

    this.state = {
      redirect: null,
      userReady: false,
      user: { id: null, username: "" },
      numUploads: 0,
      numLikes: 0
    };
  }

  componentDidMount() {

    const currentUser = AuthService.getCurrentUser();
    if (!currentUser || currentUser===null) {
      this.setState({ redirect: true });
      // alert(currentUser);
    }

    let search = window.location.search;
    let params = new URLSearchParams(search);
    var user = {
      id: null,
      username: params.get("user")
    };

```

2.

```
1
2     //If the username is not provided in the query, use the currently logged in user.
3     if (user.username === null) {
4       user = currentUser;
5       this.getUserStats(user.id);
6       this.setState({ user: user, userReady: true})
7     } else {
8       //Retrieve user ID from database
9       UserService.getUserID(user.username).then((response) => {
10        let id = response.data[0].id;
11        this.setState({user: {
12            id: id,
13            username: params.get("user")
14          },
15          userReady: true
16        });
17        this.getUserStats(id);
18      });
19    }
20  }
21
22  //Get user statistics from the database (number of uploads, total number of likes on uploads)
23  getUserStats(userID) {
24    statsService.getNumLikes(userID).then((response) => {
25      var likes = 0;
26      for (let i=0; i<response.data.length; i++) {
27          likes += response.data[i].likes;
28      }
29      this.setState({
30          numUploads: response.data.length,
31          numLikes: likes
32      });
33    });
34  }
35
36  render() {
37    if (this.state.redirect) {
38      return <Redirect push to={'/login'} />
39    }
40
41    const { user, numLikes, numUploads } = this.state;
42
43    return (
44      <div className="content" key={Date.now()}>
45        {(this.state.userReady) ?
46        <div>
47        {/* Info bar above gallery */}
48        <div className={styles.topBar + " jumbotron"}>
49            <img src= {profilePicture} className={styles.profilePicture} alt="User Icon"></img>
50            <p className={styles.username}>@{user.username}</p>
51
52            <div className={styles.statsDiv}>
53            <p className={styles.rankingLikes}>{numLikes} likes</p>
54            <p className={styles.numUploads}>{numUploads} Uploads</p>
55            </div>
56        </div>
57
58        {/* Meme Gallery */}
59        <MemeGallery pageType={PageType.PROFILE} byUser={user.id}></MemeGallery>
60        </div>
61      : null}
62      </div>
63    );
64  }
65 }
66
```

Meme.controller.js
On the back end, the meme controller handles all requests for memes and lists of memes, sending back to the user a list of memes based on their query parameters. We use Sequelize to send SQL commands to our database in order to receive back these lists. This file can be modified to add in more preset requests that may use in the future but currently we have five that we use throughout our website.

The first request returns a list of memes sorted by the number of likes they have and the time of creation. The second request returns a list containing the memes uploaded by a certain user which we identify by their userID. The third request we have returns a list of memes that a certain user has liked. Currently we only use this functionality to populate the 'Liked Memes' page which is specific to the current logged in user. The fourth request returns the list of memes that we show in our homepage to the currently logged in user. This currently prioritizes memes that the current user has not seen and orders them from newest to oldest. The final request simply returns a list of all the memes in our database and is not used in our site.

[Code start on next page]

1.

```
const { Sequelize, sequelize } = require("../models");
const db = require("../models");
const Meme = db.meme;

//Retrieve a list of memes from the database
const getMemes = (req, res) => {
  //Top Memes (/api/memes?top)
  if (req.query.top != null) {
    Meme.findAll({
      order: [
        ['likes', 'DESC'],
        ['updatedAt', 'DESC']
      ]
    }).then(memes => res.status(200).send(memes));
  }
  //By User (/api/memes?byUser='userID')
  else if (req.query.byUser != null) {
    Meme.findAll({
      where: {
        poster_id: req.query.byUser
      },
      order: [
        ['updatedAt', 'DESC']
      ]
    }).then(memes => res.status(200).send(memes));
  }
  //Liked Memes (/api/memes?liked)
  else if (req.query.liked != null) {
    console.log(req.userId);
    Meme.findAll({
      where: {
        id: {
          [Sequelize.Op.in]: sequelize.literal("(select memeID from likes where userID="+req.userId+")")
        }
      },
      order: [
        ['updatedAt', 'DESC']
      ]
    }).then(memes => res.status(200).send(memes));
  }
  //Get front page memes for a given user (/api/memes?newMemesFor='userID')
  else if (req.query.newMemesFor != null) {
    Meme.findAll({
      where: {
        id: {
          [Sequelize.Op.notIn]: sequelize.literal("(select memeID from viewed where userID="+req.userId+")")
        },
        poster_id: {
          [Sequelize.Op.not]: req.userId
        }
      },
      order: [
        ['createdAt', 'ASC']
      ]
    }).then(memes => res.status(200).send(memes));
  }
  //Get All Memes
  else Meme.findAll().then(memes => res.status(200).send(memes));
};

module.exports = {
  getMemes
};
```

Upload Image

This front end component handles getting a file for uploading from a user and sends it to the upload service which sends the image data to the back end to upload the file and add a new meme.

1.

```
export default class UploadImages extends Component {
  constructor(props) {
    super(props);

    this._isMounted = false;

    this.selectFile = this.selectFile.bind(this);
    this.upload = this.upload.bind(this);

    this.state = {

      currentFile: undefined,
      previewImage: undefined,
      progress: 0,
      message: "",

      uploadName: "", //AL - added as an attempt to alter name

      imageInfos: [],

      redirect: null,
      userReady: false,
      currentUser: { username: "" },

    };

  }

```

2.

```
componentDidMount() {
  this._isMounted = true;
  const currentUser = AuthService.getCurrentUser();

  if (!currentUser) this._isMounted && this.setState({ redirect: "/login" });

  UploadService.getFiles().then((response) => {
    this._isMounted && this.setState({
      imageInfos: response.data,
    });
  });
  this._isMounted && this.setState({ currentUser: currentUser, userReady: true });
}

componentWillUnmount() {
  this._isMounted = false;
}



selectFile(event) {

  //helps us g
  this._isMounted && this.setState({
    currentFile:  event.target.files[0],
    previewImage: URL.createObjectURL(event.target.files[0]),
    progress: 0,
    message: "",
    id: 0,
    username: "",
    uploadName: event.target.files[0].name
  });
}

upload() {
  this._isMounted && this.setState({
    progress: 0,
  });

  UploadService.upload(this.state.currentFile, {
    id: this.state.currentUser.id,
    username: this.state.currentUser.username,
    filename: this.state.currentFile.name
  },(event) => {

```

3.

```
      this._isMounted && this.setState({
        progress: Math.round((100 * event.loaded) / event.total),
      });
    })
      .then((response) => {
        this._isMounted && this.setState({
          message: response.data.message,

        });
        return UploadService.getFiles();
      })
      .then((files) => {
        this._isMounted && this.setState({
          imageInfos: files.data,

        });



      })
      .catch((err) => {
        this._isMounted && this.setState({
          progress: 0,
          message: "Could not upload the image!",
          currentFile: undefined,
        });
      });
  }
```

Upload.js

On the back end side, Upload.js handles the form data sent by the user and prepares the file for upload on the server's physical hard drive.

```
const util = require("util");
const multer = require("multer");
const maxSize = 2 * 1024 * 1024;

let storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, __basedir + "/resources/static/assets/uploads/");
  },
  filename: (req, file, cb) => {
    cb(null, req.query.date + file.originalname);
  },
});

let uploadFile = multer({
  storage: storage,
  limits: { fileSize: maxSize },
}).single("file");

let uploadFileMiddleware = util.promisify(uploadFile);
module.exports = uploadFileMiddleware;
```

File.controller.js

After calling *uploadFile* from Upload.js, the file controller handles the results. If it could not be uploaded, that information is relayed to the user, and if it could, a new meme is added to the database with the URL to the newly uploaded image. The *download* function allows access to the images, as they cannot be directly accessed by users.

[Code start on next page]

1.

```
const uploadFile = require("../middleware/upload");
const fs = require("fs");
const baseUrl = "https://redherringteam.xyz:8080/files/";
const { v4: uuidv4 } = require('uuid');

const db = require("../models");
const Meme = db.meme;

const upload = async (req, res) => {
  try {

    await uploadFile(req, res);

    if (req.file == undefined) {
      return res.status(400).send({ message: "Please upload a file!" });
    }

    Meme.create({
      url: req.query.date + req.file.originalname,
      uuid: uuidv4(),
      poster_id: req.body.id
    })

    res.status(200).send({
      message: "Uploaded the file successfully: " + req.file.originalname,
    });
  } catch (err) {
    console.log(err);

    if (err.code == "LIMIT_FILE_SIZE") {
      return res.status(500).send({
        message: "File size cannot be larger than 2MB!",
      });
    }

    res.status(500).send({
      message: `Could not upload the file: ${req.file.originalname}. ${err}`,
    });
  }
};

```

2.

```
1
2 const getListFiles = (req, res) => {
3   const directoryPath = __basedir + "/resources/static/assets/uploads/";
4
5   //create uploads directory if it does not exist yet!
6   if (!fs.existsSync(directoryPath)){
7     fs.mkdirSync(directoryPath, { recursive: true });
8   }
9
10  fs.readdir(directoryPath, function (err, files) {
11    if (err) {
12      res.status(500).send({
13        message: "Unable to scan files!",
14      });
15    }
16
17    let fileInfos = [];
18
19    files.forEach((file) => {
20      fileInfos.push({
21        name: file,
22        url: baseUrl + file,
23      });
24    });
25
26    res.status(200).send(fileInfos);
27  });
28 };
29
30 const download = (req, res) => {
31   const fileName = req.params.name;
32   const directoryPath = __basedir + "/resources/static/assets/uploads/";
33
34   res.download(directoryPath + fileName, fileName, (err) => {
35     if (err) {
36       res.status(500).send({
37         message: "Could not download the file. " + err,
38       });
39     }
40   });
41 };
42
43 module.exports = {
44   upload,
45   getListFiles,
46   download,
47 };
```

MemeMaker.js

The Meme Maker utilizes the imgflip api to create memes using popular meme templates. It requests the templates from imgflip's server, dynamically displays them as a list, and updates the state when a user clicks on a template to advance to the template text addition sub-page and the final results preview page.

1.

```
import React, { Component } from "react";
import styles from "./css/MemeMaker.module.css";
import AuthService from "../services/auth.service";
import UploadService from "../services/file-upload.service";

const Meme = ({memeTemplate, onClick}) => {
    return(
        <img className={styles.meme} key={memeTemplate.id} src={memeTemplate.url} alt={memeTemplate.name}
onClick={onClick}/>
    )
}

export default class MemeMaker extends Component{

    constructor(props) {
        super(props);

        this.state = {
            allMemes: [],
            memeTemplate: null,
            topText: "",
            bottomText: "",
            meme: null,
            currentUser: null,
            userReady: false
        };
    }

    componentDidMount() {
        fetch("https://api.imgflip.com/get_memes").then(x =>
            x.json().then(response => this.setState({allMemes: response.data.memes}))
        );
        this.setState({currentUser: AuthService.getCurrentUser(), userReady: true});
    }

    objectToQueryParam = obj => {
        const parameters = Object.entries(obj).map(([key, value]) => `${key}=${value}`);
        return "?" + parameters.join("&");
      };

```

2.

```
render() {
  const {
      allMemes,
      memeTemplate,
      topText,
      bottomText,
      meme,
      currentUser,
      userReady
  } = this.state;

  if (!userReady) return (<div></div>);

  if (meme) {
      return (
        <div className="jumbotron">
            <h3>Meme created!</h3>
          <img style={{ width: 200 }} src={meme} alt="custom meme" />
          <button className="btn btn-link" onClick={() => {this.setState({memeTemplate: null, meme: null})}}>Start over</button>
        </div>
      );
    }

  return (
    <div className="jumbotron">
        <h3>Meme Templates</h3>
      {memeTemplate && (
        <form
          onSubmit={async e => {
            e.preventDefault();
            // add parameters to create meme from imgflip api
            const parameters = {
              template_id: memeTemplate.id,
              text0: topText,
              text1: bottomText,
              username: "redherringteam",
              password: "redherring123"
            };

            //Calls imgflip api and uses objectToQueryParam above to create API call
            const response = await fetch(
              `https://api.imgflip.com/caption_image${this.objectToQueryParam(
                parameters
              )}`
            );
            const json = await response.json();
            fetch(json.data.url)
                .then(res => res.blob())
                .then(blob => {
                    blob.name = Math.random()+".jpg";
                UploadService.upload(blob, {
                    id: currentUser.id,
                    username: currentUser.username,
                    filename: "imgflip.jpg"
                  });

                });
            this.setState({meme: json.data.url});
          }}
          //Sets top and bottom text for the meme
        >

```

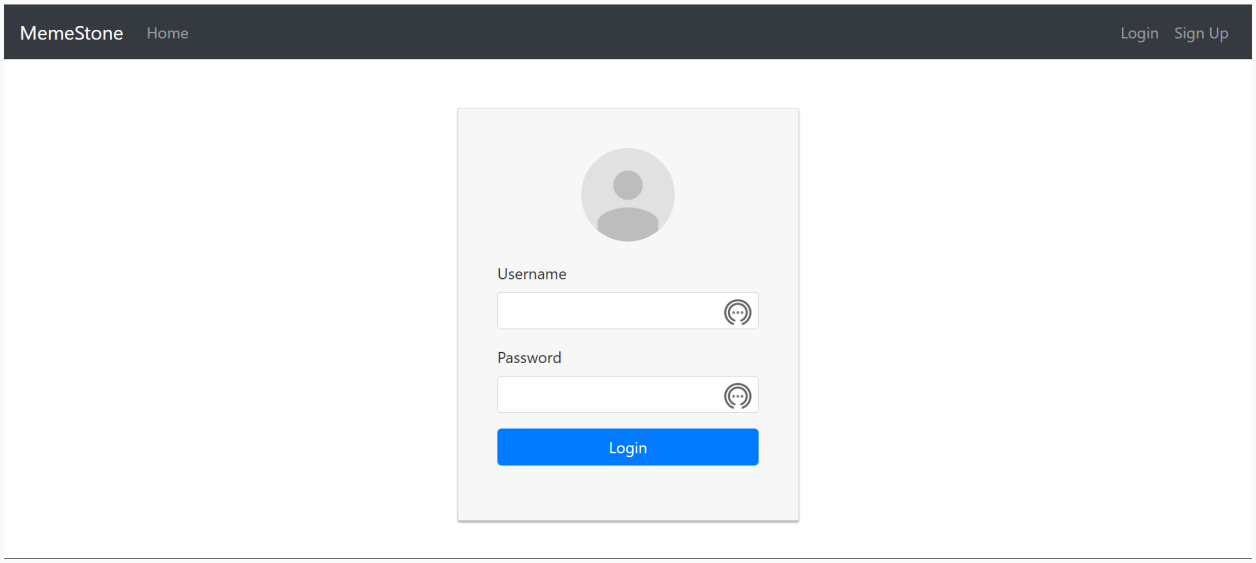
3.

```
1
2         <button className="btn btn-link" onClick={() => {this.setState({memeTemplate:null})}}>Go back</button>
3         <div className="form-group">
4
5           <Meme memeTemplate={memeTemplate} />
6           <input
7             placeholder="top text"
8             value={topText}
9             className="form-control"
10            onChange={e => this.setState({topText: e.target.value})}
11          />
12          <input
13            placeholder="bottom text"
14            value={bottomText}
15            className="form-control"
16            onChange={e => this.setState({bottomText: e.target.value})}
17          />
18          <button type="submit" className="btn btn-primary">Create meme!</button>
19          </div>
20        </form>
21      )}
22
23      {!memeTemplate && (
24        <>
25          {allMemes.slice(0,100).map(memeTemplate => {
26            return (
27              <Meme
28                memeTemplate={memeTemplate}
29                onClick={() => {
30                    this.setState({memeTemplate: memeTemplate});
31                }}
32              key={memeTemplate.id}/>
33            );
34          })}
35        </>
36      )}
37      <p>Meme Templates provided by <a href="https://imgflip.com/" target="blank">imgflip.com</a></p>
38    </div>
39    // Home page which calls all pre-template memes from imgflip meme creator ^
40  );
41        }
42 }
```

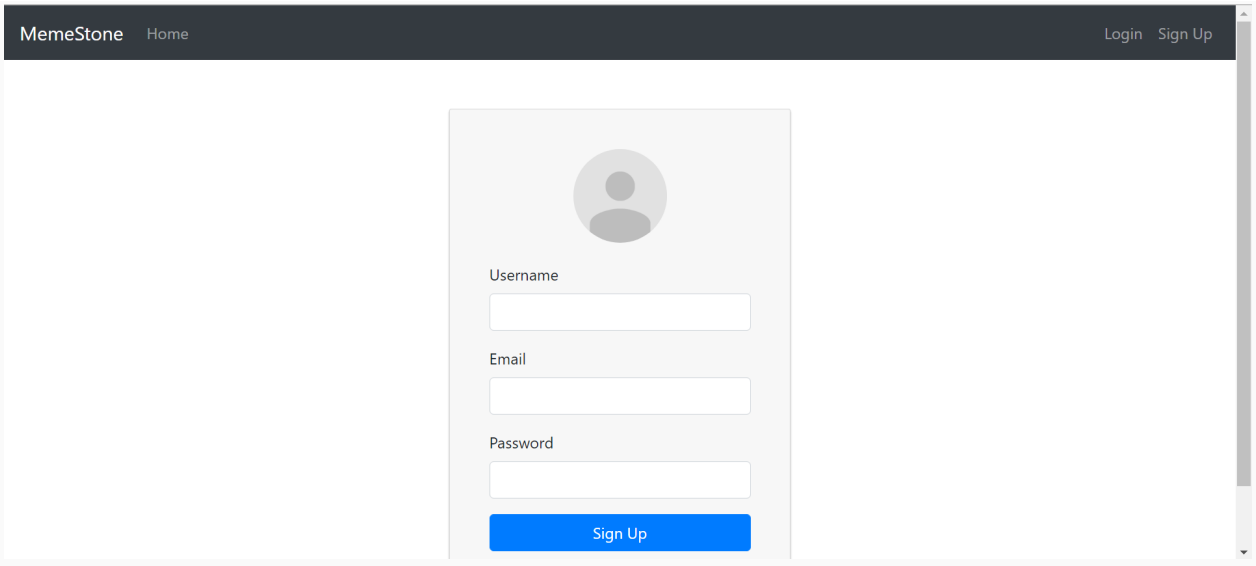
Website Screenshots

These screenshots represent the state of our website at our final demo, which can be accessed at https://redherringteam.xyz/

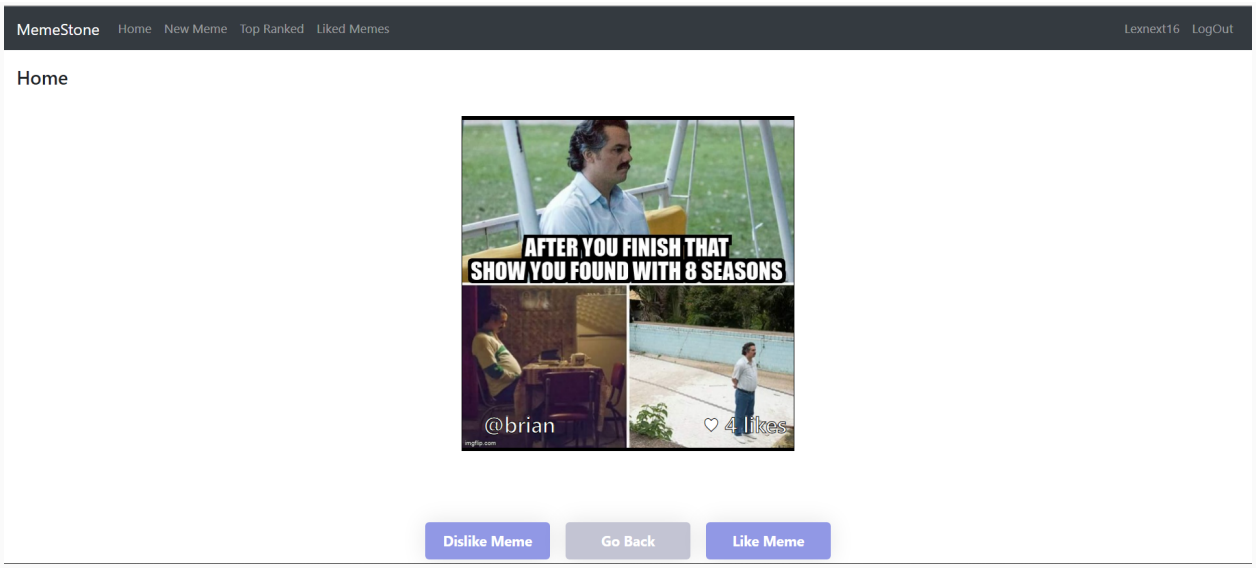
Login Page



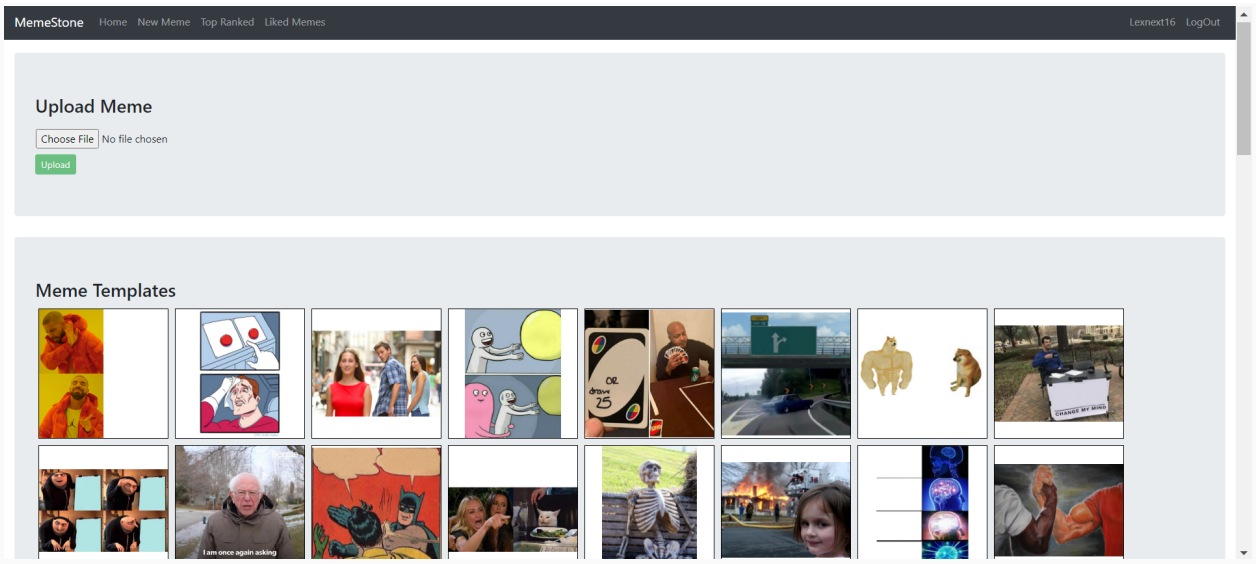
Registration Page



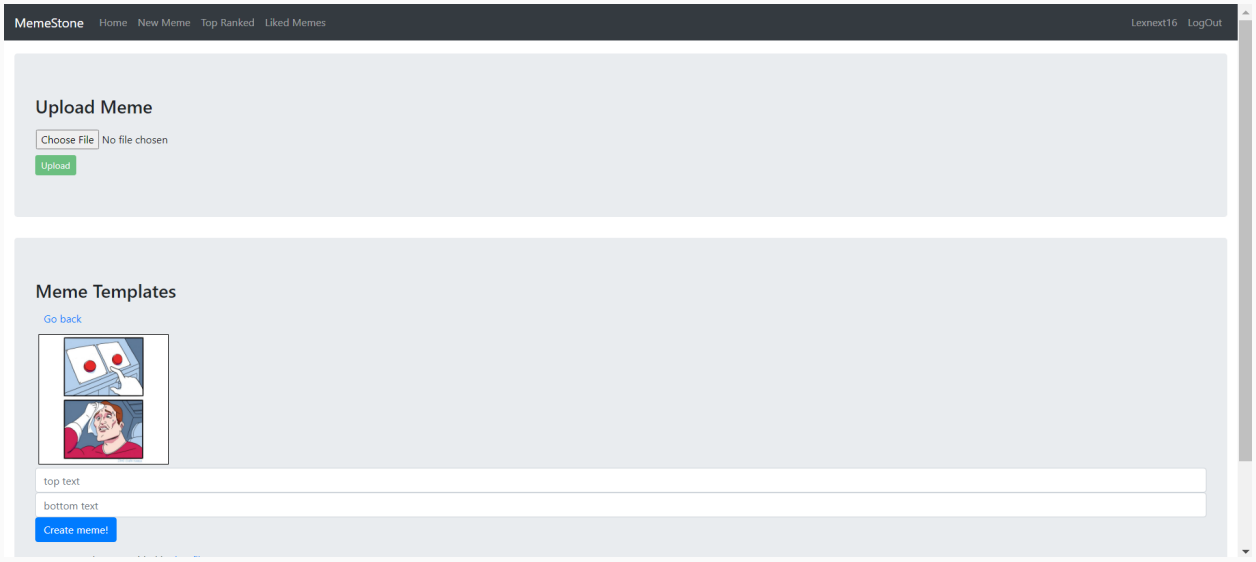
Home Page



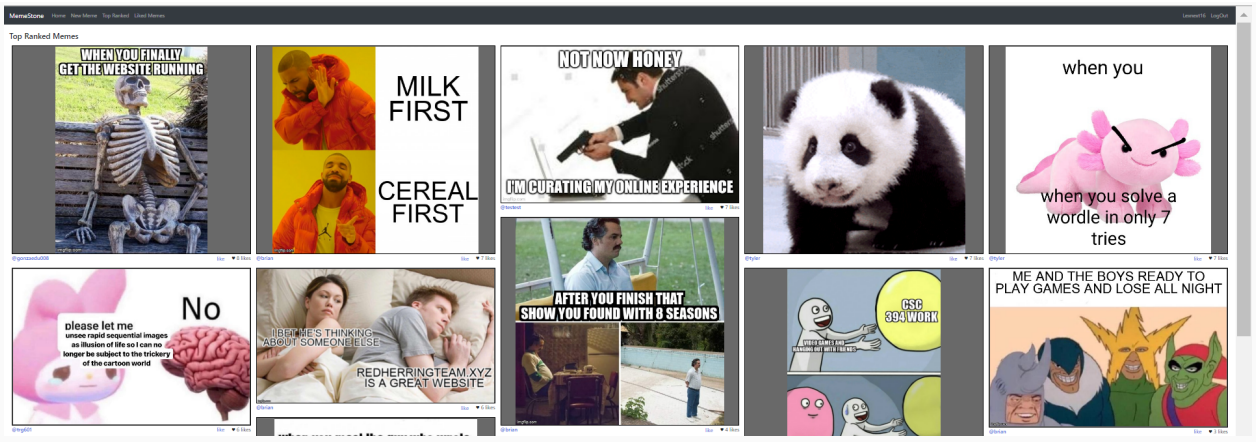
New Meme Page

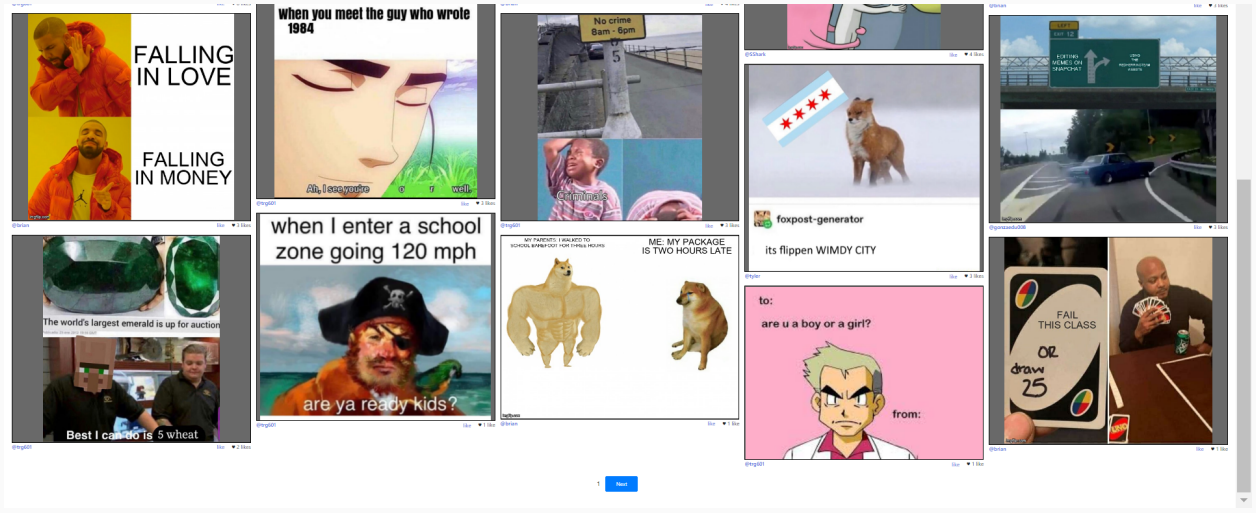


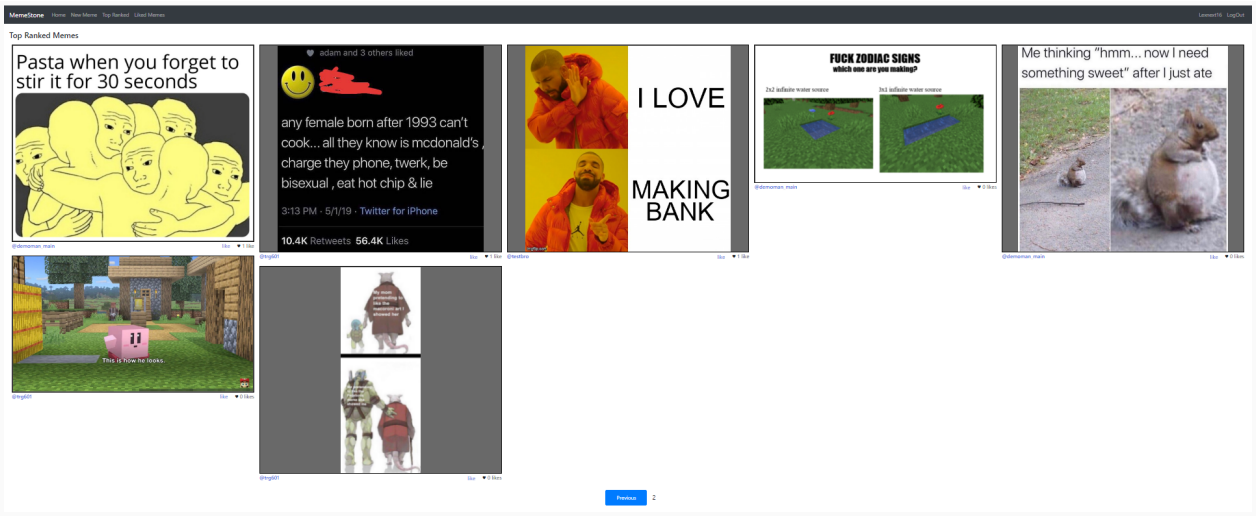
Meme Templates Page



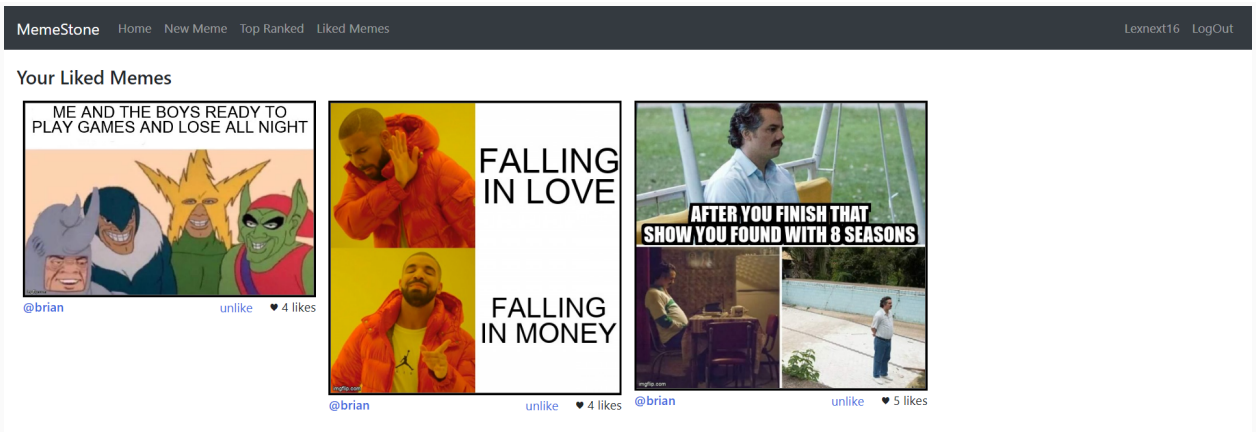
Top Ranked Memes Pages 1 & 2



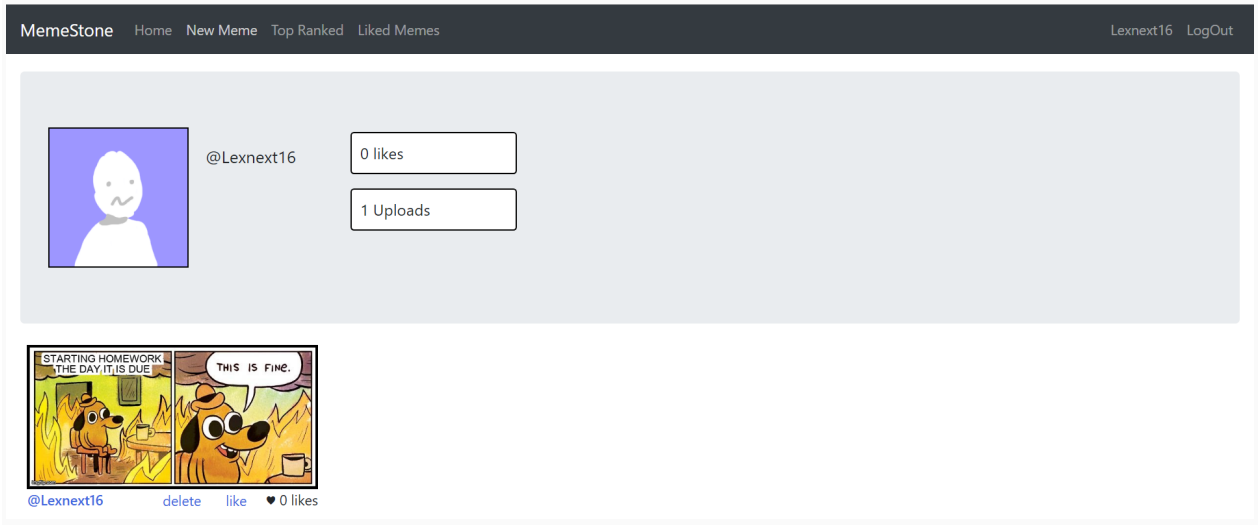




Your Liked Memes Page



Profile Page



Video Creation

For the creation of our videos the final work done is on editing the video and putting it all together. The only work that is done before receiving the resources needed for our video is the choosing of music to be used during the video, the creation of the templates for our slides, and any preliminary slides that can be made looking at the video's script. Most of the work is done during the editing process where our Video Manager goes through all the resources and works chronologically through the script putting together the voice overs, images, and videos.

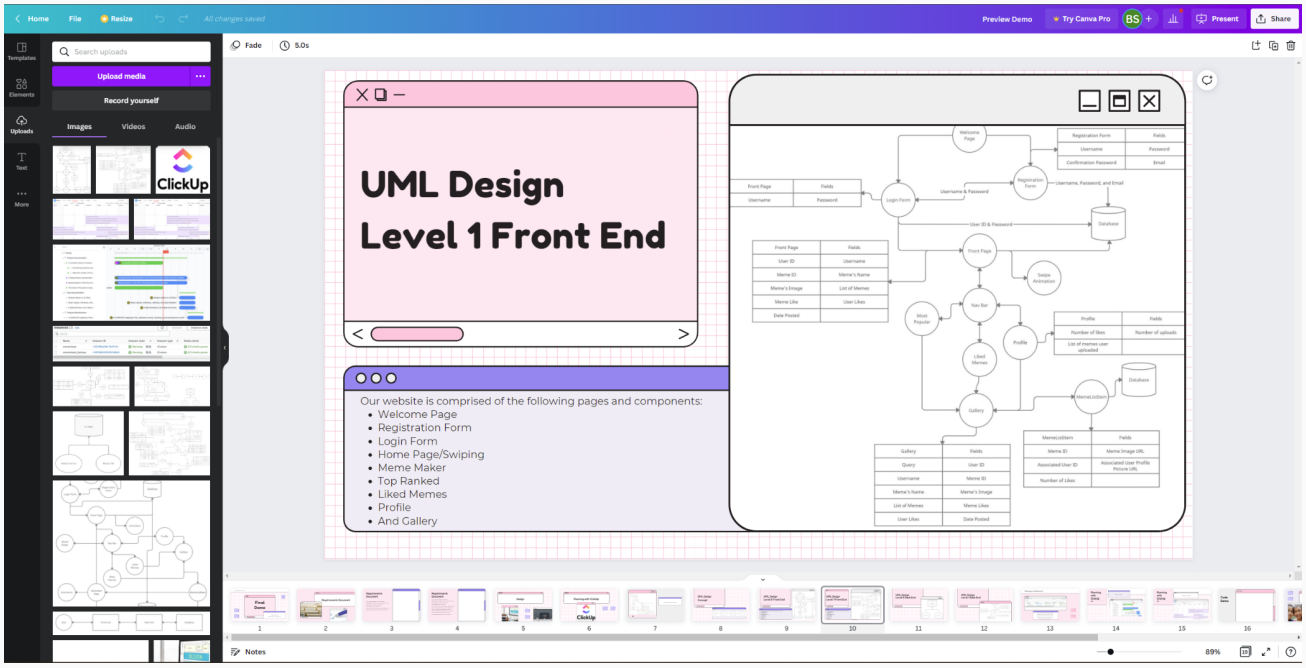
The tools used

- DaVinci Resolve
    - A free application that is used to edit and put the video together.
- Canva
    - A free web tool that allows us to use either pre-made templates or create our own to make the slides used within the video.
- YouTube
    - We use YouTube to find and choose the royalty free music which was used in the background of our videos.
    - To download the music we chose we use the links given in the description where available or youtube.com
- OBS
    - This application was used to record the demos that play during the voice overs in the video.

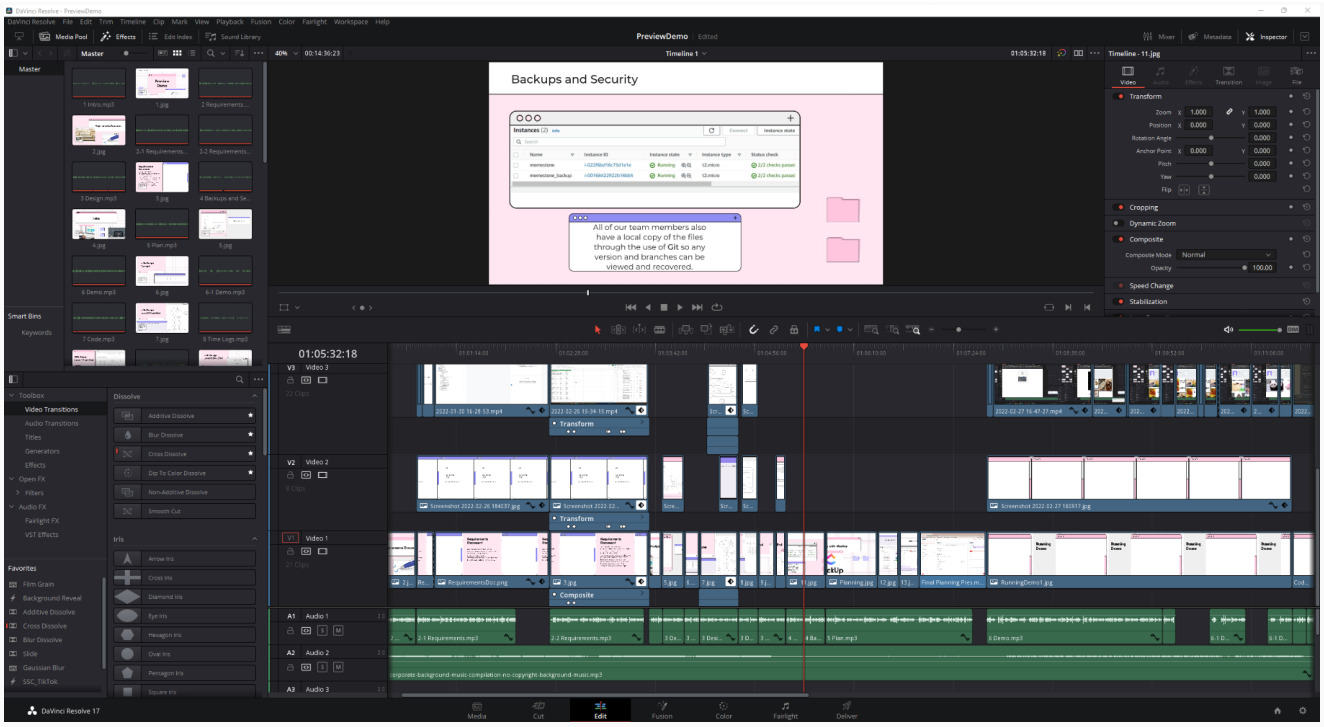
First we go through all the resources given and check to make sure that the voice overs are clear and without issue while also making sure they match our script. If any issues are found then the appropriate team members are contacted for updated resources.

The work done on the videos is done in chronological order as lined up with the scripts. When there needs to be images or demos that need to be screenshot or recorded they are done so when needed. Audio is normalized so that the Presentation Manager's voice can be heard clearly and is louder than the background music that we are using. All edits, cuts, and transitions are made so as to ensure clarity of what we are demonstrating to create a smooth and well paced presentation. Images, text, and videos are highlighted and zoomed in on to draw the viewer's eyes to necessary sections on the screen lining up with what is being talked about in the voice over. Following are images showing two of the tools that we used in action when we were creating our final video:

Canva



DaVince Resolve



Once the video reaches a satisfactory state for the Video Manager it is passed on to the Presentation Manager and the Project Manager for final review and criticism.

Time Logs

Throughout the project, our team maintained our time logs for work done in a project-wide timelog excel sheet, with an individual tab for each member and sum totals for each compiled into one tab.

Group Total

Name:		**Total Hours:**
Alex Lazarov		76.50
Alex Look		40.40
Benjamin Villanueva		25.75
Brian Salinas		44.88
CJ Navarro		25.25
Eduardo Gonzalez		34.00
Mohammed Amiwala		22.15
Patrick Meyer		35.40
Tyler Garman		100.50
Total Group Hours:		404.83

Alex Lazarov

Date	Hours	FULL description of the task in 3 sentences or fewer.
1/3/2022	1.15	First group meeting
1/4/2022	3.00	Familiarizing myself with react framework
1/5/2022	1.00	Mid week meeting
1/6/2022	8.00	Setup the local node.js environment, created basic login page, created basic pop-up registration form.
1/7/2022	2.00	Flushed out basic implementation of mainpage, assisted Tyler basic on react-router-dom rerouting.
1/8/2022	0.50	Researched on possible implementation of gesture based react libraries.

1/8/2022	1.10	Group meeting to discuss the plan for the presentation
1/10/2022	1.25	Class meeting, reassigning roles, talking about future expectations.
1/11-1/12/2022	2.50	Researching express servers for front end to mariadb connectivity (and other facets such as axios library, various forms of authentication forms oauth/jwt)
1/13/2022	1.50	Meeting discussing rdp. discussing expectations.
1/14-1/16/2022	1.25	Pre check design note planning before breaking it down into 5-hr modules.
1/17/2022	1.50	Finished the module design plan and submitted. Participated in Weekly Group meeting, discussing where we are at with the formal design and what must be done this week for the requirements documentation
1/20-1/22/2022	3.25	Worked on requirements documentation . Functional requirements and finite state machine v. 0.
1/24/2022	1.00	Discussed what was over in class, checked up on the requirements tasks, delegated weekly roles for the RDP video.
1/25/2022	1.50	Recompiled requirements matrix. Worked on script for functional matrix and Finite State machine.
1/26/2022	1.50	Group meeting to finalize requirements document and get team approvals, work on script for presentation. Worked on finalizing functional matrix script. Bad client bad dev.
1/31/2022	0.50	Group meeting: Went over plan and assigned tasks for Phase 3 - Coding and user testing the preview demo.
2/7/2022	1.00	Patrick demonstrated how the database/frontend code will work. roles and expectations were decided.
2/9/2022	0.50	Mid week meeting to make sure we are on track
2/10/2022	3.00	Implemented handlebars for navigation between login / registration. Successfully test submission to database.
2/14/2022	1.00	Group meeting, troubleshooting issues with backend/frontend communication, revising plan for the preview demo
2/15/2022	12.00	Implemented frontend login/registration, implemented backend Authentication and authorization system via api.
2/16/2022	12.00	Implemented backend upload/download api and submission to filesystem api, implemented front end submission UI. Meme submission to database
2/17/2022	2.00	Integration of gallery to front end system.
2/18/2022	2.00	Code review session with tyler and working minor issues
2/19/20	2.00	Integrated the meme gallery to the frontpage

22		
2/20/2022	1.00	2nd revision of finite state machine
2/21/2022	1.00	Group meeting, discussing preview demo and planning for final demo
2/22/2022	1.50	Code jam with the project manager, we fixed some errors, researched some solutions and got things working
2/23/2022	1.00	Fixed bugs with page redirects when not logged prevent people from accessing pages they should not have access to.
2/24/2022	2.00	Researching, learning, and implementing how to log out on one page kicks the user out of all tabs so the user can't have access to all other pages.
2/28/2022	1.00	Team meeting for final stretch delegating roles, planning, and understanding expectations.
3/2/2022	0.50	Updating project manual. Authentication
3/4/2022	0.50	Updating project manual. Screenshots

Alex Look

Date	Hours	FULL description of the task in 3 sentences or fewer.
1/3/2022	1.15	First group meeting (First official) assigned roles and went over a concept idea we can go ahead and make
1/5/2022	2.50	Creation of UML designs based on initial drawings done for the project (Thinking time .50)
1/5/2022	1.00	Meeting going over current progress of where the whole group is at and further action regarding programming for demo deadline
1/8/2022	0.75	Meeting going over the demo script and video requirements
1/8/2022	0.25	Writing, recording, and uploading introduction and backup design script for presentation and editor
1/10/2022	1.00	Meeting (Second official) gave feedback about concept demo video, reassigning roles, brainstorm to flesh out project further
1/11/2022	0.50	Thinking time on ideas (quality of life for project) how to flesh the project base out further
1/12/2022	1.50	Meeting as a group to flesh out ideas for further action to build the website
1/13/2022	1.50	Meeting (lead programmers and myself) set responsibility on each module for the website and groundwork for ideas and breakdowns of design
1/16/20	1.00	Starting to put together the design document with module breakdowns and time

22		estimates from programmers thinking time (Thinking time .50)
1/17/2022	0.50	Meeting (Unofficial) went over what is needed for design documents and plan ahead for working on the requirements and planning documents too
1/18/2022	1.50	Compiled coders module breakdowns and put into a formal design document. Made base versions of the overview for design of the project (end to end, front end, and back end). Totaled the planned projected hours
1/20/2022	2.00	Created UML Level 1 document using the coders module break down, and transferred the programmers complete languages listings into design document
1/23/2022	1.00	Meeting went over needs for design documents, made functional and testing matrices and started to fill those out. Deadlines set for RDP documents
1/24/2022	1.00	Meeting (Fourth official) went over plans for the week. Delegated script roles for demo and checked on requirements and rebreak down design modules to fit 5 hour rule
1/24/2022	0.50	Talked with project manager and lead back end on design about filling in browsers supported, cookies, scripting, GUI, security, run time errors, and security privacy in design document
1/25/2022	2.50	Talked with the project manager on making the UML interactive and using Prezi. Updated all the Level 0 and 1 UMLs. Had Eduardo check over design document (Thinking and mental breaks .75)
1/26/2022	1.75	Meeting going over requirements matrices and fixing as a group. Went over the RDP script making sure what is needed by deadline and recording plans too. Went over all UML designs and fixed small errors with the project manager. Went over plans for Prezi
1/27/2022	3.00	Worked on a design script, asked the project manager to revise. Fixed small issues in UML level 0 & 1. Finished Prezi presentation. Helped documentation manager with a section of the requirements document
1/28/2022	1.25	Finished Prezi modules updating the modules per section and importing all the photos needed for the presentation
1/31/2022	0.50	Meeting (Fifth official) went over rdp demo feedback. We talked about steps and tasks for the group into phase 3 with everyone's role based on user and coding demo
2/7/2202	1.00	Meeting (Sixth official) Patrick demonstrated his progress on the backend and how it will work with the front end. Tasks for members floating were assigned and setting group meetings to check in later this week.
2/9/2022	0.50	Meeting went over current progress with the main programmers. Group gave link for CSS help for the design coders to learn basic uses for website
2/14/2022	1.00	Meeting (Seventh official) discussed a plan to get front and back ends to connect. Planned meetings for later this week to set up time to live code and fix errors.
2/17/2022	2.50	Worked with the documentation manager on the project booklet. Transferred overview, requirements and matrices, design, and planing info into final booklet
2/19/2022	1.50	Made changes to UML Prezi condensed subtopics into bigger topics and less transitioning. Made minor changes to the project booklet. Thinking time:.50
2/21/20	1.00	Meeting (Eighth official) Discussed comments to improve for final demo and

22		upcoming week jobs per member
2/22/2022	1.75	Updated UML Front End Level 0 & 1 on modules not implemented in the final website for final demo. Also updated UML images in manual and Prezi presentation too
2/26/2022	0.50	Added in more detail on the design section of the final demo script. Mainly took the information from the RDP demo and moved it in.
3/2/2022	1.00	Meeting group work session going over final manual
3/3/2022	0.50	Talked with project manager about extra aspects to add in and an idea about section for non used design photos based on booklet link and how submission of manual works.emailed professor about it for clarification
3/5/2022	1.50	Took screenshots of the website pages and inserted into manual, titled and did quality of life fixes for the design section images (Thinking time .50)
3/7/2022	1.00	Looked over quality of life for all sections. Resized images for text to easily readable. Also made a section for excluded modules

Benjamin Villanueva

Date	Hours	FULL description of the task in 3 sentences or fewer.
1/3/2022	1.15	First group meeting
1/5/2022	1.00	Second group meeting
1/8/2022	1.10	Third group meeting
1/8/2022		Writing script for concept demo
1/8/2022		Recording
1/8/2022		Review
1/10/2022		Fourth group meeting - discussed the future plan.
1/12/2022	1.50	Fifth group meeting. discussed ideas on what could be implemented and voted on the best and feasible.
1/23/2022	1.00	Eighth group meeting. went over needs for design documents, made functional and testing matrices and started to fill those out.
1/24/2022	1.00	Ninth group meeting. Discussed what mentioned on what needs to be worked on in class by the professor, checked up on the requirements tasks, delegated weekly roles for the RDP video.
1/26/20	1.00	Tenth group meeting.Group meeting to finalize requirements document and get team

22		approvals, work on script for presentation.
1/29/2022	0.50	Gathering group materials and script sections together. Organizing materials in a way I can read and use it to record.
1/29/2022	2.00	Writing script sections as well as editing group members sections for fluency and speakability.
1/29/2022	0.50	Proofread script and edit out mistakes in flow.
1/29/202	2.00	Record script segments and edit out background noise. Combine segments that are together, and edit out individual segments to individual audio files.
1/29/202	0.50	Organize audio files and scripts to be handed off to Video Editor for review and handling.
1/31/2022	0.50	Eleventh group meeting: Went over plan and assigned tasks for Phase 3 - Coding and user testing the preview demo.
2/7/2022	1.00	Twelfth group meeting: Discussed plans for implementing next items into site and who would be taking over the item.
2/12/2022	0.50	Worked on creating logo for site
2/14/2022	1.00	Thirteenth group meeting: Discussed issues with the project thus far, and what will need to get done before the week is done.
2/20/2022	1.50	Reviewed and edited script to flow smoothly and added extra information where needed.
2/20/2022	0.50	Discussed flow and additional details of script with Project Manager and Video Manager, to make sure unneeded segments were cut or needed segments were added.
2/20/2022	2.00	Record Script segments. Combined segments that are together, and edit out individual segments to individual audio files.
2/21/2022	1.00	Fourteenth Group meeting - discussing preview demo and planning for final demo.
2/25/2022	0.50	Edited script to add in more detail and some sections based on project managers discussion on previous meetings.
2/26/2022	2.00	Recording new segments with better audio quality, and recording extra sections that needed to be in the final video
2/28/2022	1.00	Fifteenth group meeting - Discussed project manual and what needs to be added/changed in it
3/2/2022	1.00	Group work session, working on final project manual

Brian Salinas

Date	Hours	FULL description of the task in 3 sentences or fewer.
1/3/2022	1.15	First group meeting
1/3/2022	1.00	Note creation for fleshing out what our project is and how to accomplish it
1/5/2022	1.40	Fleshed out my notes on the project and initial comments on how to create the video
1/7/2022	2.00	Creation of notes and preparing checklist for what is needed for the video
1/8/2022	1.10	Group meeting to discuss the plan for the presentation
1/9/2022	5.00	Video Editing. Chose templates, created 'slides', created voiceover, chose music, normalized audio, and spliced the different clips, videos, pictures, and demos.
1/12/2022	0.50	Wrote down some General Design notes that I shared with the PM since I was unable to attend a brainstorming group meeting. Detailed my overall thoughts and came up with an idea to possibly implement in the website.
1/17/2022	0.50	"Unofficial" meeting with a group to discuss the direction of our RDP/Requirements documentation.
1/23/2022	1.00	Group meeting where we went over the documents that we are working on or going to work on for the RDP portion of our project. Assigned Requirement categories to each group member to be worked on by 01/26
1/24/2022	1.00	Group meeting where we discussed delegating roles for our script and RDP video and talked about our requirements matrix deliverable.
1/25/2022	0.50	Worked and wrote down requirements for the Security portion of our project in the Requirements Matrix document. Wrote down 7.
1/29/2022	1.00	Compiling all the relevant materials and how each will be used and how they belong to each other
1/29/2022		- Went through all the files given by the presentation manager and parsed everything so that I could better line up the Script with the audio, images, and eventually the video.
1/29/2022	1.00	Search and creation of templates to be used as 'slides' within the video
1/29/2022		- Search for royalty free music that will be used for the video on YouTube and the web
1/29/2022	1.00	Start on editing the video by creating a rough compilation of all the materials.
1/29/2022		- Went through the audio given by the Presentation Manager to assure that there are no errors in the script and that there are no errors in the files themselves

1/29/2022		- Communicated with the presentation manager over adding some additional voice-overs
1/30/2022	7:30	Video Editing by continuing to create templates and adding audio
1/30/2022		- Constant creation for templates and slides to be used in all portions of the videos. Includes creation, download, importing, length change, overlay, and audio sync
1/30/2022		- Recorded any supplemental footage that will be used with the voice-overs to show what is being talked about. This is also imported, cut, overlayed, and synced with audio
1/30/2022		- Added overlays and animations to be used when highlighting specific segments in the pictures that are being used within the video. Line up shapes, animate, zoom and smooth out zoom/pans, sync with audio
1/30/2022		- Looked up any tutorials I needed to add certain effects that I was wanting to add.
1/30/2022		- Cut, move, extend, freeze frame, zoom, pan, and added transitions to individual clips as they were being worked on.
1/30/2022		- Worked in a chronological order for most of the video, creating templates and recorded videos, moving audio and clips around, etc.
1/30/2022		- Got in contact with any team members that I needed additional information or supplemental materials from.
1/30/2022		- Added music to the entire video and normalized the audio.
1/30/2022		- Reviewed the video progression as it was being worked on to ensure transitions and animations were smooth, appropriate, and were not missing.
1/30/2022		- Wrote down what I have done for my time log
1/31/2022	0.50	Group meeting to talk about the plan for the next phase in the project and how we will handle coding and module assignments to individuals.
2/7/2022	1.00	Patrick demonstrated how the database and front end works. Design and coding tasks were also assigned.
2/11/2022	1.00	Went through all of the documents and compiled them into a personal document to better address the different things going on in the project to make it easier to help with programming and what needs to be done and what is working. This is for personal use.
2/12/2022	1.00	Went over all the code in our git repository creating my own personal comments on all the files.
2/13/2022	0.50	Set up Mariadb and got it working with the code with the help of our Systems Programmer.
2/14/2022	1.00	Group meeting to troubleshoot issues with backend/frontend communication, revising plan for the preview demo
2/16/20	1.50	Created a personal file with code commentary that was created by Partick.

22		
2/17/2022	3.00	Continued code commentary. Did database testing with connecting our react application to a personal MariaDB database. Still not finished, only set up connections and basic table creation.
2/18/2022	0.50	Searched for a template to be used for our Preview Demo and prepared the video editing environment. Also found some music that may be used.
2/19/2022	3.00	Helped with creating the scripts for our code, demo, and designs. Talked with various members of our group on getting a local version of our most current code working on my machine.
2/20/2022	6.00	Video Editing, video creation, and continued scripting
2/20/2022		- Put the finIshing touches on the script for our video with Benjamin. Condensed some parts to get rid of fluff.
2/20/2022		- Created the slides for the video using the previously chosen templates. Added images, text, and moved around elements to create the slides.
2/20/2022		- Added in slides, audio from Ben, and music that will be in the video.
2/20/2022		- Created the supplemental demo videos that will be accompanying the voice over given by Benjamin.
2/20/2022		- Cut and rearranged all the clips to maintain pacing for the video
2/20/2022		- Added zooms, pans, and transitions where needed.
2/20/2022		- Got in contact with any team members that I needed additional information or supplemental materials from.
2/21/2022	1.00	Group meeting, discussing preview demo and planning for final demo
2/25/2022	1:00	Worked on some scripting and suggestions for the video with Ben and Tyler.
2/26/2022	4:00	Video Editing starts.
2/26/2022		-Created new slides to be used in the video
2/26/2022		-Created new video walkthroughs to be displayed with the voiceovers provided by Ben. (Completed Requirements and Design)
2/26/2022		-Added new effects to better show images and text that appeared a bit small in the video per the Professor's request.
2/27/2022	5:00	Continued Video Editing
2/27/2022		-Created more new slides that were needed

2/27/2022		-Created additional video walkthroughs for the Website demo, Code demo, and Time Log
2/27/2022		-Added in effects, cuts, and transitions to the video to improve visibility of certain parts of our video
2/27/2022		-Audio normalization and cutting
2/27/2022		-Communicated with Tyler about issues with the website that appeared during the creation of the demo.
2/28/2022	1.00	Group meeting where we discussed our project manual and what needs to be added/changed in it
3/5/2022	2.00	Created updated images for our code in our project manual section. Started work on expanding on our code commentary.
3/6/2022	3.00	Finished adding more code commentary to the project manual and fixed the formatting and spacing to be more consistent. Also added in a section explaining the video editing process of our demos.

Christopher James Navarro

Date	Hours	FULL description of the task in 3 sentences or fewer.
1/3/2022	1.15	First group meeting (Mandatory): assigning roles/planning
1/5/2022	1.00	Second group meeting: planning the demo
1/8/2022	1.10	Third group meeting: finishing the demo
1/8/2022	1.00	Started on Group Time Log/Created Task List sheet
1/9/2022	1.10	Updated Group Time Log (20 min break/chores)
1/10/2022	1.00	Fourth Group meeting (Mandatory): reassigned roles/ going over road map (meeting dates, types of meetings, deadlines for work)
1/12/2022	0.45	Working on Brainstorming sheet/researching how to use google sheets
1/12/2022	1.00	Fifth Group Meeting: Brainstorming with team (sharing ideas for website, going through others ideas written down and picking which ones to do)
1/17/2022	0.30	Sixth Group Meeting: Going over formal design of requirements document, assigning dates (individual assignments are due)
1/23/2022	0.30	Seventh Group Meeting: Went over requirements document (assigned people to specific parts of document), worked on Requirements Document (Joined late)
1/24/20	1.00	Eight Group Meeting (Mandatory): Discussed plans for the week (delegated roles,

22		went over requirements tasks todo), worked on task assigned
1/24/2022	0.20	Talked with Project Manager about Requirements document (what I have to do, what parts i have to complete)
1/26/2022	0.15	Worked on the Requirements Matrix (filled in my assigned rows)
1/26/2022	0.30	Ninth Group Meeting: Went over the requirements put down in the matrix, final preparation on requirements document and script
1/27/2022	5.00	Worked on Requirements Document (filling it out), revising the Requirements matrix
1/31/2022	0.30	Tenth Group Meeting: Went over plan for the demos, assigned tasks for coding and user testing for the demo
2/7/2022	1.00	Eleventh Group Meeting (Mandatory): Looked at front end code and database demonstration done by Patrick, assigned tasks and due dates for tasks
2/9/2022	0.30	Twelfth Group Meeting: Checking in on progress (How everyone is doing on their assigned tasks)
2/14/2022	1.00	Thirteenth Group Meeting (Mandatory): Went over what needs to be focused on for final demo, worked on individual tasks
2/17/2022	2.30	Worked on filling in Final Project Manual with Alex Look
2/21/2022	1.00	Fourteenth Group Meeting (Mandatory): Went over the results of RDP Demo, noted what needs to be fixed/added for final demo
2/28/2022	1.00	Fifteenth Group Meeting (Mandatory): Went over whats needed to fill in Project Manual, assigned parts for members to fill out
3/2/2022	1.00	Sixteenth Group Meeting: Checked on individual progress on Project Manual parts, worked on Project Manual
3/3/2022	1.00	Worked on Project Manual
3/6/2022	1.30	Finishing up Project Manual

Eduardo Gonzalez

Date	Hours	FULL description of the task in 3 sentences or fewer.
1/3/2022	1.15	First group meeting
1/4/2022	1.15	Planning course of action with Tyler
1/5/2022	1.00	Created timeline and assigned tasks for the first week
1/5/202	1.00	Team meeting where we discussed important website features and our plan for the

2		rest of the week
1/8/2022	0.30	Worked on presentation script and outline
1/8/2022	1.10	Group meeting to discuss the plan for the presentation
1/9/2022	0.30	Began brainstorming project plans by looking at due dates and assignment requirements
1/9/2022	0.30	Set up ClickUp (Project Management Tool)
1/10/2022	1.00	Took notes on assignment requirements
1/11/2022	0.30	Discussed RDP specifications with Tyler
1/11/2022	2.00	Reviewed assignment notes and projected a timeline for RDP completion
1/12/2022	5.00	Created design documentation plan with Tyler's help
1/12/2022	2.00	Assigned more tasks and due dates using ClickUp
1/12/2022	1.30	Team meeting where we discussed future plans for our project
1/15/2022	0.20	Adjusted project plan as we needed more time for the design document
1/17/2022	0.30	Created Requirements document checklist
1/17/2022	0.30	Team meeting where we discussed design document progress and plans for requirements document
1/18/2022	2.00	Added programming plan and assigned tasks and due dates
1/18/2022	1.00	Modified design schedule and began calculating time estimates
1/19/2022	1.00	Began to break down requirements document into tasks
1/23/2022	0.40	Group discussion on design and requirements document. Began to assign tasks for requirements document
1/24/2022	1.00	Group meeting where we discussed deadlines for the requirements and design documents
1/25/2022	1.00	Created script for Ben that covers planning and the overview of the requirements document
1/31/20	0.30	Group meeting: Went over plan and assigned tasks for Phase 3 - Coding and user

22		testing the preview demo
2/7/2022	1.00	Group meeting: Patrick demonstrated how the database/frontend code will work, tasks were assigned
2/10/2022	2.00	Started working on the profile page of our website. Modified CSS to fit new page
2/14/2022	1.00	Group meeting: Discussed programming issues and mapped out our focus for the next 2 days
2/19/2022	0.30	Attempted to migrate website to AWS
2/19/2022	1.00	Contributed to preview demo script by adding planning information
2/21/2022	1.00	Group meeting: discussed changes for upcoming final demo
2/28/2022	1.00	Discussed project manual and what needs to be added/changed in it
2/2/2022	1.00	Group meeting: group work session where we worked on the project manual
2/2/2022	0.30	Modified planning section of the project manual

Mohammed Amiwala

Date	Hours	FULL description of the task in 3 sentences or fewer.
1/3/2022	1.15	First group meeting
1/5/2022	1.00	Second Group Meeting
1/6/2022	3.00	Frontend React JS Coding
1/8/2022	3.00	CSS Coding, Image modification work, and React Coding
1/10/2022	1.00	3rd Group meeting where we discussed future design plan and roadmap for Meme Generator
1/13/2022	1.50	Programmer meeting discussing future module plans and design flow
1/14/2022	1.00	Meeting with requirements manager and project manager to assess new requirements for project
1/17/2022	1.00	Group meeting & worked on assigned modules workflow sheet
1/23/20	1.00	Group meeting to discuss RDP assessment and assign requirements roles

22		
1/24/2022	1.50	Added modules to testing requirements
1/24/2022	1.00	Group meeting where we discussed delegating roles for our script and RDP video and talked about our requirements matrix deliverable.
1/26/2022	1.00	Group meeting to finalize requirements document and get team approvals, work on script for presentation.
1/31/2022	0.50	Group meeting: Went over plan and assigned tasks for Phase 3 - Coding and user testing the preview demo
2/4/2022	2.00	Went over testing modules and began testing to fill out and approve testing requirements listed in the testing requirements matrix
2/7/2022	1.00	Group meeting: Patrick demonstrated how the database/frontend code will work, tasks were assigned
2/8/2022	1.50	Watched a video tutorial on animations and began working on front-end animations. Image animation module for react completed.

Patrick Meyer

Date	Hours	FULL description of the task in 3 sentences or fewer.
1/3/2022	1.15	First group meeting
1/4-5/2022	12.00	Setting up Docker, Nginx, GitLab, etc etc on my server
1/5/2022	0.50	Configuring/validating VSCode GitLab integration for the project
1/10/2022	1.50	Monday group meeting - Opting not to change roles, distributing a more refined responsibility list.
1/13/2022	1.50	Programmer meeting - Choosing second round features, divvying up documentation and technical tasks.
1/23/2022	1.00	Group meeting, going over the requirements document to-do list, and working on the testing requirements
1/24/2022	1.00	Discussed what was over in class, checked up on the requirements tasks, delegated weekly roles for the RDP video
1/26/2022	1.00	Group meeting to finalize requirements document and get team approvals, work on script for presentation
1/31/2022	0.50	Group meeting: Went over plan and assigned tasks for Phase 3 - Coding and user testing the preview demo
2/6/2022	5.00	Followed along on two CRUD-like tutorials; one was professional and the other was amateur, but both were educational and a good time.
2/7/202	4.00	Completed the professional tutorial from yesterday, minimally adapting it to our

2		purposes. Also got it deployed and running on gordon, resolving properly, etc
2/7/2022	1.00	Wrangled a couple of testing-to-deployment bugs that cropped up. Also, I think I should just convert the Handlebars code to React for the sake of Mo et al.
2/7/2022	1.00	Weekly team meeting. I went over the work I'd done via the CRUD tutorials and showed folks how to check out and run that branch locally.
2/9/2022	4.00	Lent Eduardo a hand troubleshooting some issues he encountered running the patrick/dbplaytime branch locally
2/12/2022	0.25	Likewise offered some guidance to Brian on running the patrick/dbplaytime branch locally

Tyler Garman

Date	Hours	FULL description of the task in 3 sentences or fewer.
1/3/2022	1.15	First group meeting
1/4/2022	1.00	Planning course of action with Eduardo
1/4/2022	1.25	Talking with various members about what they need to do
1/5/2022	1.00	Second group meeting, delving into how to attack programming portion
1/6/2022	4.00	Implemented basic front end for user profile, liked memes, and most popular memes pages
1/7/2022	3.00	Merged Alex Lazarov's changes with mine
1/7/2022	0.50	Nonfunctional front-end for the homepage
1/7/2022	1.00	Status updates and discussions with members
1/8/2022	1.10	Group meeting to discuss the plan for the presentation
1/10/2022	1.00	Worked on agenda for tonight's meeting and big picture plan with Eduardo, making some more concrete goals to add to his existing plan
1/10/2022	1.00	Second post-class group meeting, showing everyone the rough big picture plan, minorly reassigning roles, discussing timelogs, and kicking off what needs to be put in the requirements for programming
1/11/2022	3.00	Started time log rework based on feedback. Talking with Eduardo about the big picture plan timeline and helping with the design plan for this week, writing in more detail what needs to be done
1/11/20	1.00	Soft - Came up with some ideas to bring to the brainstorming/voting

22		
1/12/2022	1.50	Group brainstorming session and voting on ideas for overall design of website
1/13/2022	1.50	Meeting with lead programmers and design manager to talk about module breakdown for design document, assigned each programmer a few modules to write the basic design specifications for
1/15/2022	2.00	Worked on portion of the module design documentation
1/17/2022	0.50	Weekly Group meeting, discussing where we are at with the formal design and what must be done this week for the requirements documentation
1/20/2022	1.50	Checking in with members and writing down a summarized todolist for the requirements
1/22/2022	1.00	Working on project role requirements and to-do list for the requirements document
1/23/2022	1.00	Group meeting, going over the requirements document to-do list, and working on the testing requirements
1/24/2022	1.00	Written project management requirements and category for liaison
1/24/2022	1.00	Discussed what was over in class, checked up on the requirements tasks, delegated weekly roles for the RDP video
1/26/2022	2.00	Review of formal design modules, splitting up too large modules into smaller ones
1/26/2022	1.00	Group meeting to finalize requirements document and get team approvals, work on script for presentation
1/27/2022	2.00	Discussing how to approach formal design presentation with Alex Look and fixing errors in formal design document
1/31/2022	0.50	Group meeting: Went over plan and assigned tasks for Phase 3 - Coding and user testing the preview demo
2/7/2022	1.00	Group meeting: Patrick demonstrated how the database/front end code will work, tasks were assigned
2/9/2022	0.50	Group check-in, discussed issues and shared progress on programming
2/13/2022	5.00	Attempting to get React working with express for development
2/14/2022	3.00	Attempting to get React working with Node using local Docker installation
2/14/2022	1.00	Group meeting, troubleshooting issues with back end/front end communication, revising plan for the preview demo
2/16/2022	2.00	Getting caught up with current code setup Alex Lazarov had been working on and begun work on meme request service

2/16/2022	2.00	Met with Alex Lazarov and Patrick to program and discuss how we will handle certain interactions with the database, such as uploading and liking memes
2/17/2022	5.00	Working on meme request service api, pulling memes from the database for all our needs in react
2/17/2022	3.00	Programmed Meme Gallery/ Meme List Item components and the pages they appear in
2/18/2022	6.00	Implemented profile statistics, meme liking functionality
2/18/2022	2.00	Revised memes/likes tables in the database to use user IDs rather than usernames, requiring a good bit of rework in the code
2/18/2022	2.00	Met with Alex Lazarov to troubleshoot code issues and merge our changes
2/19/2022	4.00	Implemented/updated meme maker code from concept demo
2/19/2022	3.00	Implemented swiping, building off of code from Alex Lazarov
2/19/2022	2.00	Bug Fixing, some css tweaks, and code cleanup
2/20/2022	6.00	Following issues with docker setup on Patrick's home server, I worked on getting our code running on an AWS EC2 Instance and made a backup server
2/20/2022	1.00	Going over video script, talking to video and presentation managers to ensure they have everything they need
2/21/2022	1.00	Group meeting, discussing preview demo and planning for final demo
2/22/2022	1.50	Met with Alex Lazarov to work on code updates and troubleshoot some bugs
2/22/2022	2.00	Security updates, making sure user is properly authenticated and allowed to access resources for all requests
2/23/2022	4.00	Code updates, adding ability to to delete memes, pagination in the Meme Gallery
2/24/2022	1.00	Merged all pending code changes with AWS EC2 branch and pushed the updates to the server
2/24/2022	0.50	Met with Mo to troubleshoot git issues and bring him up to speed on code changes
2/28/2022	1.00	Discussed project manual and what needs to be added/changed in it
3/2/2022	1.00	Group work session, working on final project manual
3/5/2022	2.00	Added meeting agendas and brainstorming notes to project manual, as well as edit some sections to have more cohesive style

3/5/2022	0.50	Added a bit of text to explain what each piece of code shown in the project manual does
3/6/2022	1.00	Fixed some formatting for sections of the manual after noticing some parts were not appearing in the table of contents, other minor style changes