LIVE | Educational Video Game Database
Final Report

Team Members: Zachary McDowell, Xiuyu Tang, Yeon Chae, Nilo Lisboa, Komo Zhang

Project Summary:

Our project started from the client (Dr. Rugh)'s request to create a cloud-based database of Educational Video Games (EVGs) that serves as a resource for teachers, faculty, administration, and students. The main goal is to provide a platform where users can easily search for EVGs relevant to their respective classes or industries. The main items that the client required us to make this platform should be interactive, simple, and user-friendly, allowing users to search for EVGs using terms, keywords, or subjects. Additionally, users should be able to access information about the EVGs, such as descriptions, reviews, and links to websites or other resources. The applications meeting these requirements can be listed in the following: (1) a search engine that users can search games with related information, (2) the creation of filters that users can filter based on their interests (publication year, costs, etc.), (3) functionalities with "save games" that users can save their games and can review those saved games in their profile page. Additionally, on the game index page, users can review all the games listed in the current database.

In terms of functionality, based on the client's requirements for adding, deleting, or modifying information about the EVGs, ensuring that the database remains comprehensive and up-to-date. During the project, we classify three different roles (user, moderator, admin) to differ in their authority for the tasks including adding, deleting, or modifying information in the database. For instance, users can only create their account, and search the games, while the admin can delete the game in the database, and allocate roles for each user. Additionally, Our team deployed the client's database into Heroku, and the accessibility to Heroku is shared with the client, enabling them to simply update the latest modification based on their needs. With an aim to support educators and industry instructors in finding and incorporating valuable EVGs into their curriculum, our project meets all requirements that the client requested and we believe this created web page can work successfully both in terms of functionality and user-friendliness.

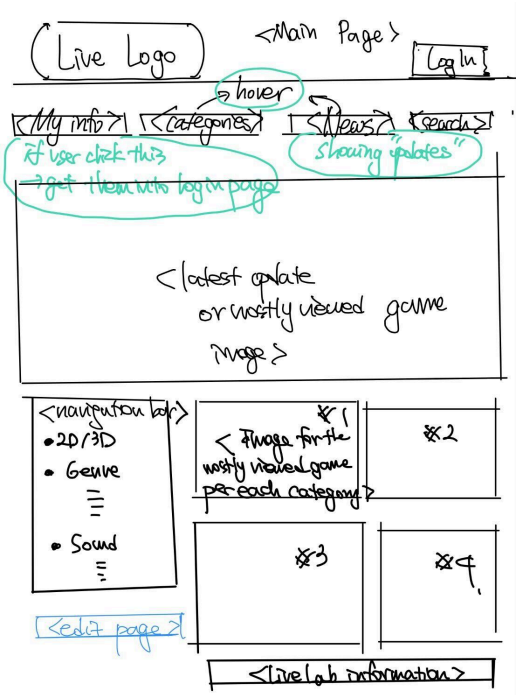
User Stories:

User Stories and UI Mockups for Sprint 1

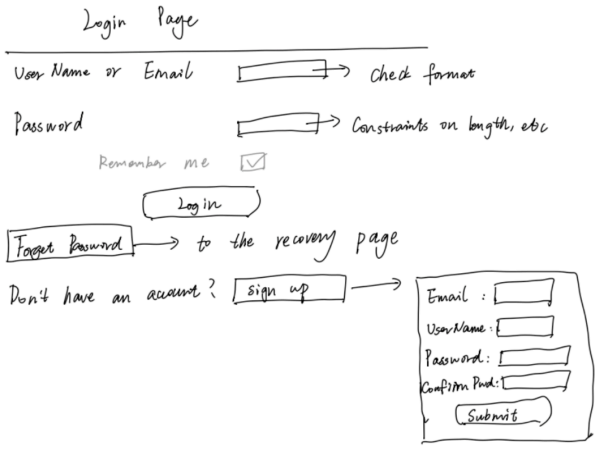
- Chore: Create Heroku deployment for the application
  Given points: 2. Status: completed.
- Feature: Populate the database with CSV data given by LIVE Lab
  Given points: 2. Status: completed.

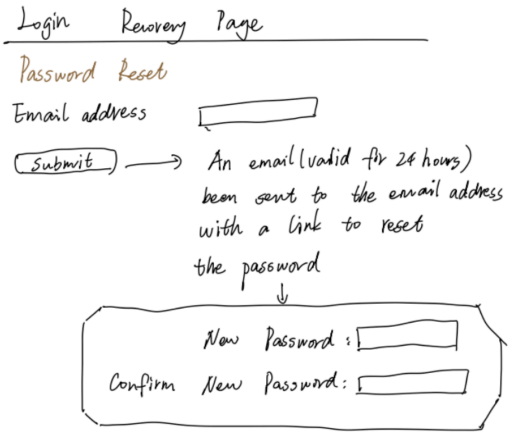
- Feature: Create a database using MongoDB or PostgreSQL
  Given points: 2. Status: completed.
- Feature: Implement a basic main page that introduces the application
  Given points: 2. Status: implemented.
- + Chore: Change deployment DB from Mongo to PostgreSQL
  Given points: 2. Status: completed.
- * Feature: Allow any user to create new game entries in the application
  Given points: 1. Status: changed and implemented.
    - Originally, it was meant to permit only privileged users, but focusing on the base functionality of any user was a better target for the story in the first sprint.
- * Feature: Allow any user to delete games from the application
  Given points: 1. Status: changed and implemented.
    - Originally, it was meant to permit only privileged users, but focusing on the base functionality of any user was a better target for the story in the first sprint.
- * Feature: Allow any user to edit information on an education game's details screen
  Given points: 1. Status: changed and implemented.
    - Originally, it was meant to permit only privileged users, but focusing on the base functionality of any user was a better target for the story in the first sprint.
- + Feature: Implement links to other functions on the main page
  Given points: 3. Status: added after the sprint started, implemented.
- + Feature: Allow users to create a new account
  Given points: 2. Status: added after the sprint started, implemented.
- + Feature: Create a login page for users
  Given points: 1. Status: added after the sprint started, implemented.
- - Feature: Create a search bar that is capable of retrieving data from the DB
  Given points: 3. Status: removed from this sprint.
    - A search bar was visually created as a part of developing the front page, but it is not yet capable of retrieving DB data.

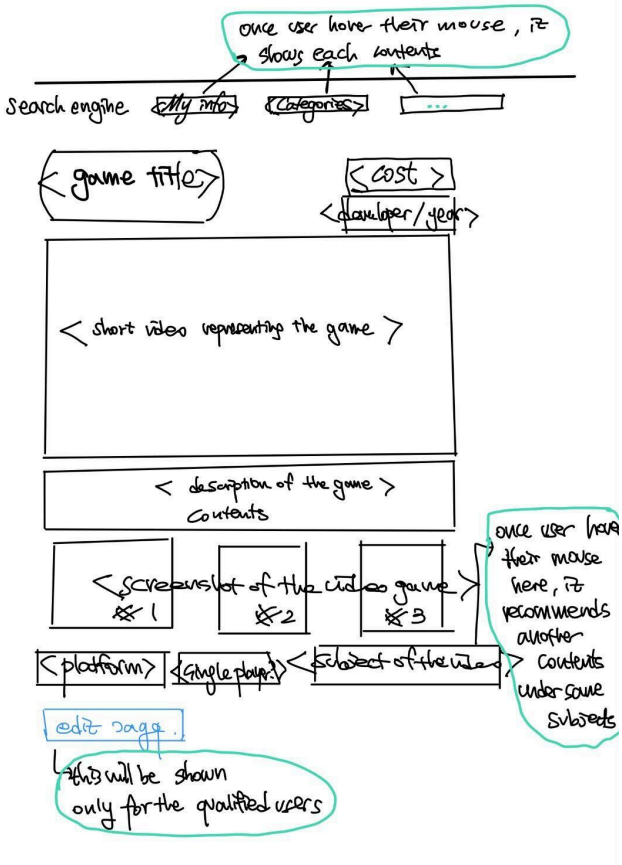
The corresponding UI Mockups for Sprint 1 are as follows:


Live Logo
<Main Page>
Log In
hover
My info
Categories
News
Search
If user click this
→ get them into login page
showing updates
<latest update
or mostly viewed game
image>
<navigation bar>
• 2D/3D
• Genre
• Sound
※1
<Image for the
mostly viewed game
per each category>
※2
※3
※4
<edit page>
<Live Lab information>


LIVE Database
ALL Video Game List.
Title
Game Type
Game Time
Platform
More
Add New Video Game
LIVE Database.
Create New Video Game
Title
Game Type
Game Time
Platform
More
Save changes.
LIVE Database
2D/3D
Genre
Sound
Save changes.
Scroll bar
LIVE Database
Title. . . .
Game Type . . .
scroll bar.
Back to List.


Login Page
User Name or Email
Check format
Password
Constraints on length, etc
Remember me
Log in
Forget Password
to the recovery page
Don't have an account?
sign up
Email :
User Name :
Password :
Confirm Pwd :
Submit

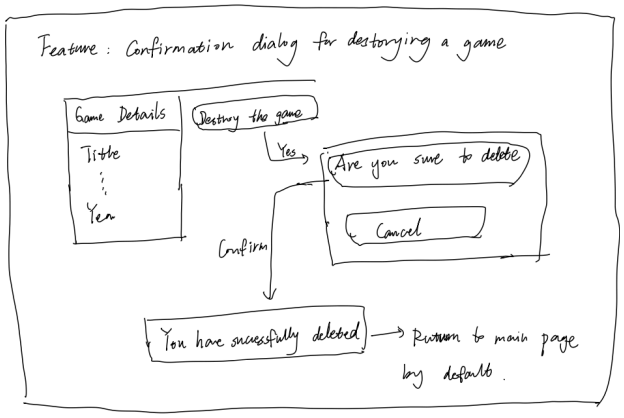

Login Recovery Page
Password Reset
Email address
Submit
An email (valid for 24 hours)
been sent to the email address
with a link to reset
the password
New Password :
Confirm New Password :

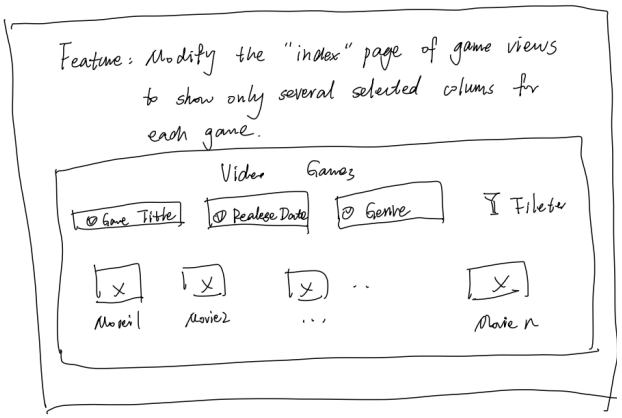

once user hove their mouse, it
shows each contents
search engine
My info
Categories
...
<game title>
<cost>
<developer/year>
<short video representing the game>
<description of the game>
Contents
<screenshot of the video game>
※1
※2
※3
once user hove
their mouse
here, it
recommends
another
contents
under some
subjects
<platform>
<Single play>
<subject of the video>
edit page
this will be shown
only for the qualified users

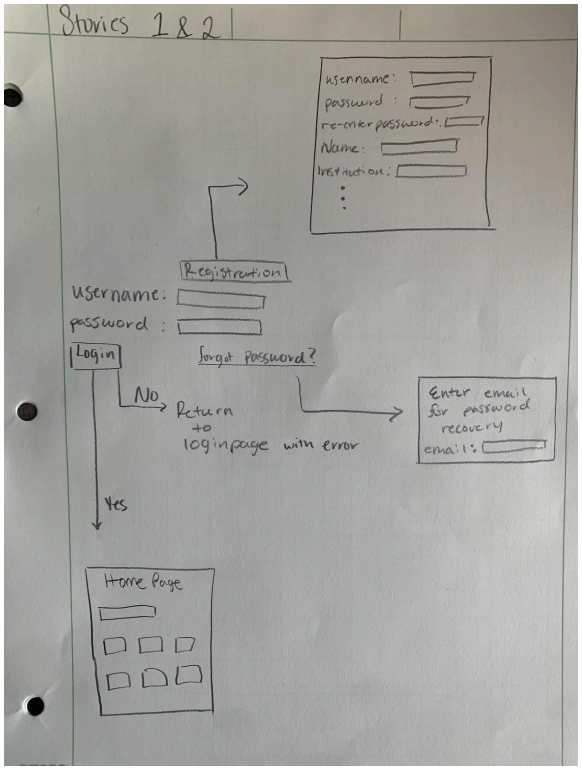
User Stories and UI Mockups for Sprint 2

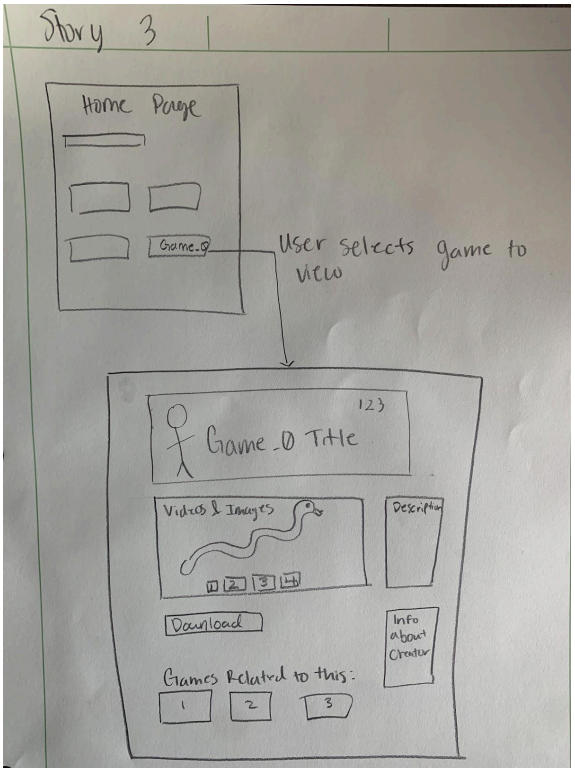
- Feature: Implement Sign Up and Login
  Given points: 2. Status: implemented.
- Feature: Post-Login Functionality
  Given points: 3. Status: implemented.
- - Feature: Visualization of the Game.
  Given points: 2. Status: removed from this sprint.
    - Since we don't have the images for visualization of the games at this stage, we only changed the six images on the main page. More detailed visualization will be implemented in the next sprint.
- + Feature: Change the example games on the main page
  Given points: 1. Status: added after this sprint started, implemented.
- Feature: Confirmation dialog for destroying a game
  Given points: 2. Status: implemented.
- Feature: Modify the "index" page of game views to show only several selected columns for each game
  Given points: 2. Status: implemented.
- Feature: Modify the "new" page of game views, to add text input boxes for the new columns
  Given points: 2. Status: implemented.
- Feature: Modify the "show" page of game views, to show the new columns
  Given points: 2. Status: implemented.
- Feature: Modify the "edit" page of game views, to show the new columns and make them editable.
  Given points: 2. Status: implemented.
- Feature: Navigation back to the main page from the game list page
  Given points: 2. Status: implemented.

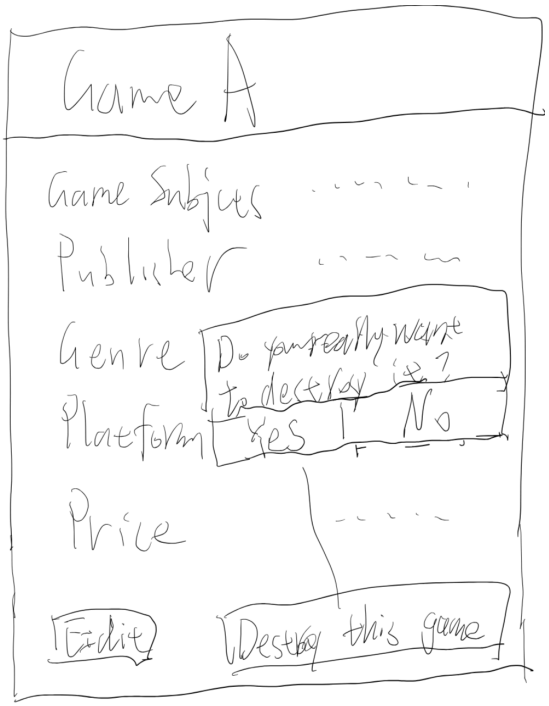
The corresponding UI Mockups for Sprint 2 are as follows:

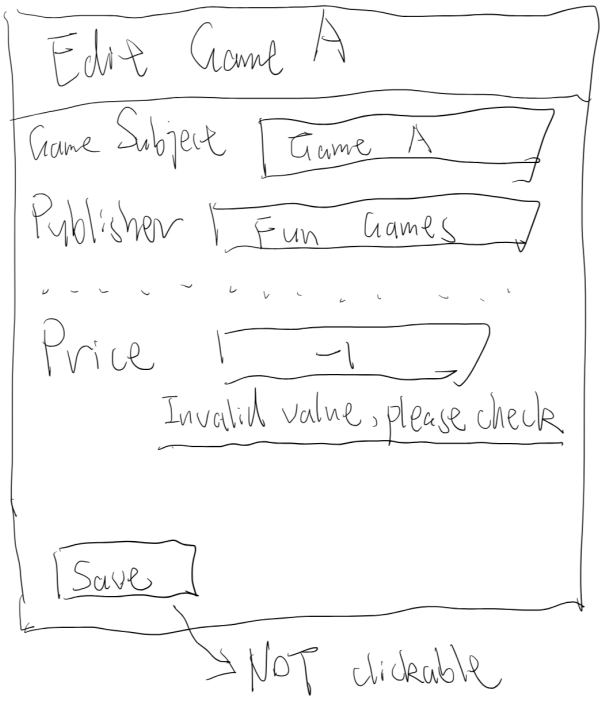


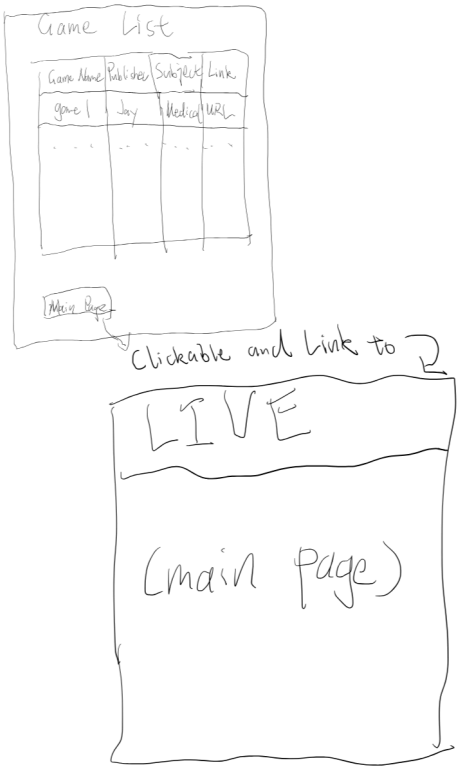


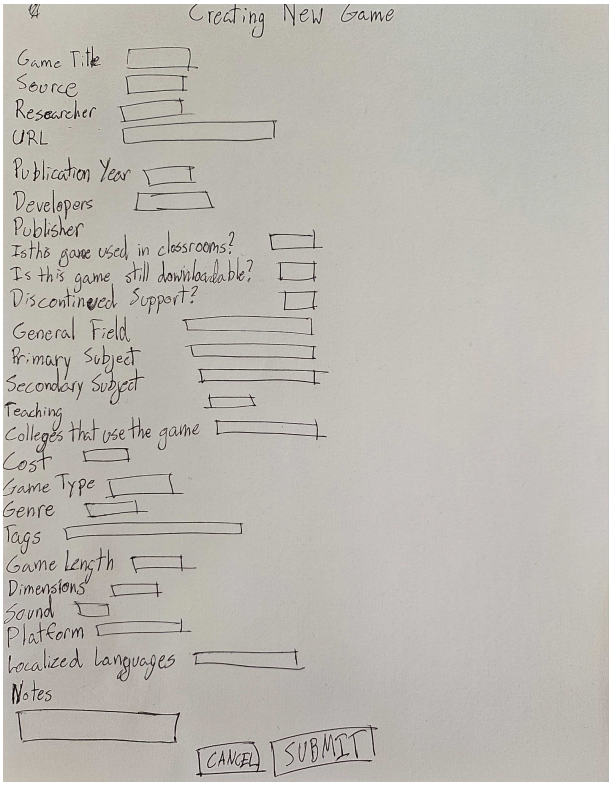

Stories 1 & 2
username:
password:
re-enter password:
Name:
Institution:
Registration
username:
password:
Login
forgot password?
No
Return to login page with error
Enter email for password recovery
email:
Yes
Home Page


Story 3
Home Page
Game_0
User selects game to view
123
Game_0 Title
Videos & Images
Description
Download
Info about Creator
Games Related to this:
1
2
3


Game A
Game Subjects
Publisher
Genre
Platform
Price
Do you really want to destroy it?
Yes
No
Edit
Destroy this game


Edit Game A
Game Subject
Game A
Publisher
Fun Games
Price
-1
Invalid value, please check
Save
NOT clickable





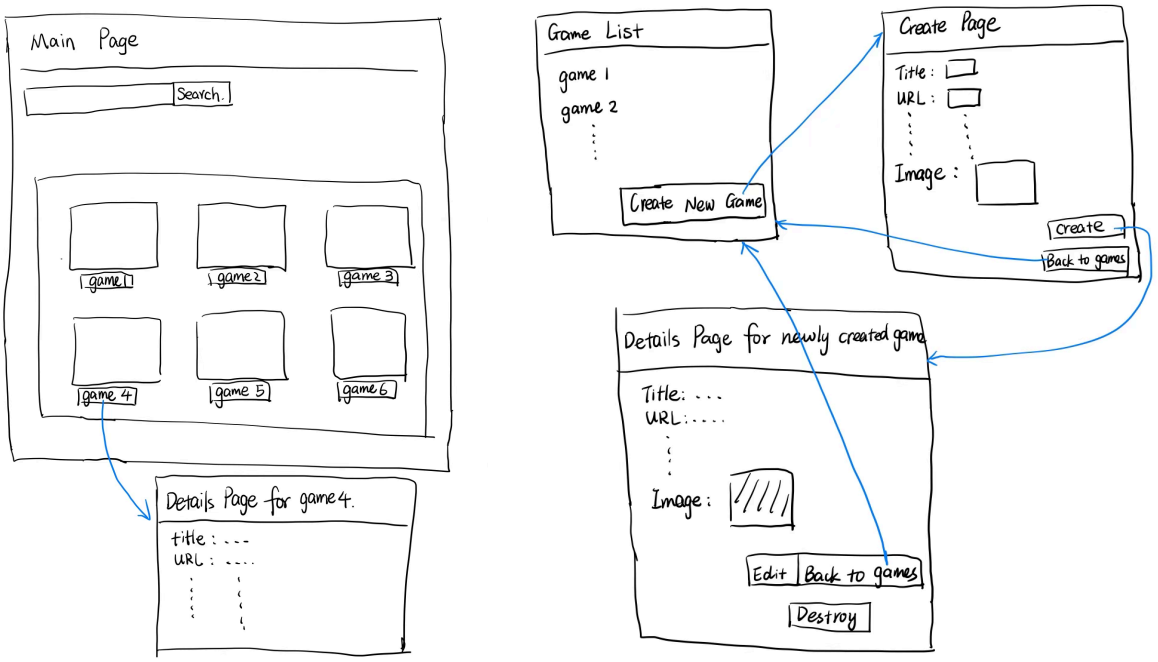
User Stories and UI Mockups for Sprint 3

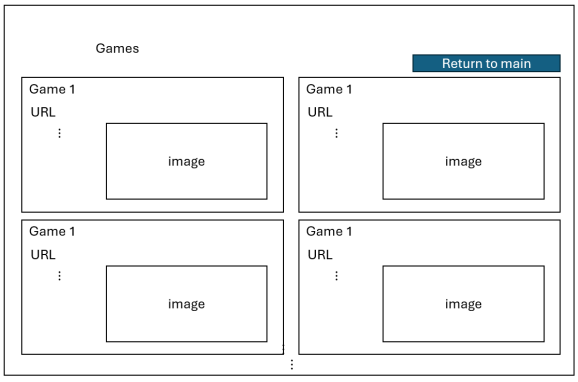
- Feature: Images on the main page redirect to the details page of the game.
  Given points: 1. Status: implemented.
- Chore: Import new image URLs from the Image Link column in the CSV.
  Given points: 2. Status: completed.
- Feature: Add Image Link field to New Game page.
  Given points: 2. Status: implemented.
- Feature: Display Images for games on the details page.
  Given points: 2. Status: implemented.
- Feature: Display Images for games on the index page.
  Given points: 1. Status: implemented.
- Feature: Allow the user to edit the image link on a game's details page.
  Given points: 2. Status: implemented.
- - Feature: Allow users to create accounts of moderator and administrator levels.
  Given points: 2. Status: removed from this sprint.
    - Instead of making users directly create accounts of moderator and administrator levels, we now set all new accounts to be "user" as a default role. If some of these new users want to be moderators or administrators, they need to ask an administrator to change the role for them.
- Feature: Create an administration page that allows administrator users to alter permissions on other existing users.

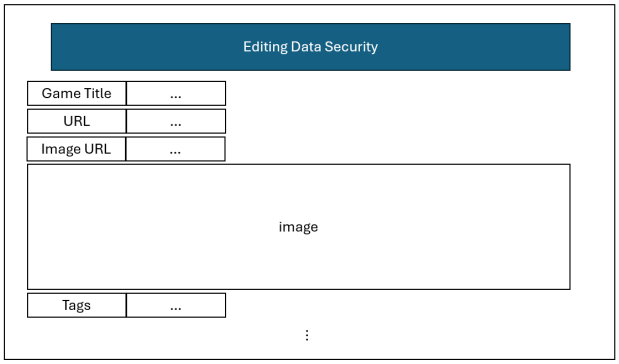
Given points: 2. Status: implemented.

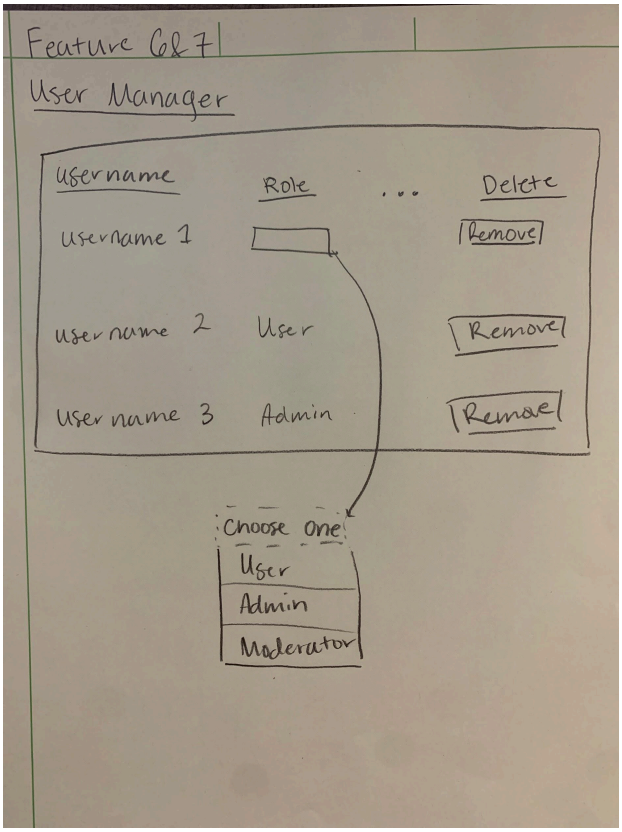
- - Feature: Display option to ‘destroy’ game or link to ‘admin’ page only to administrator users.
  Given points: 2. Status: removed from this sprint.
  - We don’t have enough time to implement this feature, so this feature is postponed to the next sprint.
- - Feature: Display option to ‘edit’ game only to administrator and moderator users.
  Given points: 2. Status: removed from this sprint.
  - We don’t have enough time to implement this feature, so this feature is postponed to the next sprint.
- + Feature: Split the game index page into multiple pages.
  Given points: 1. Status: added after this sprint has started, and implemented.

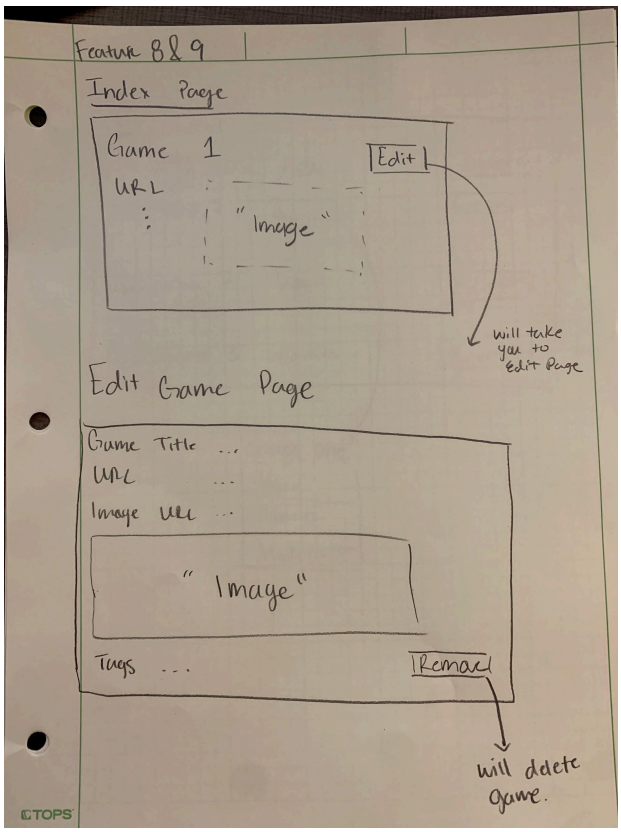
The corresponding UI Mockups for Sprint 3 are as follows:









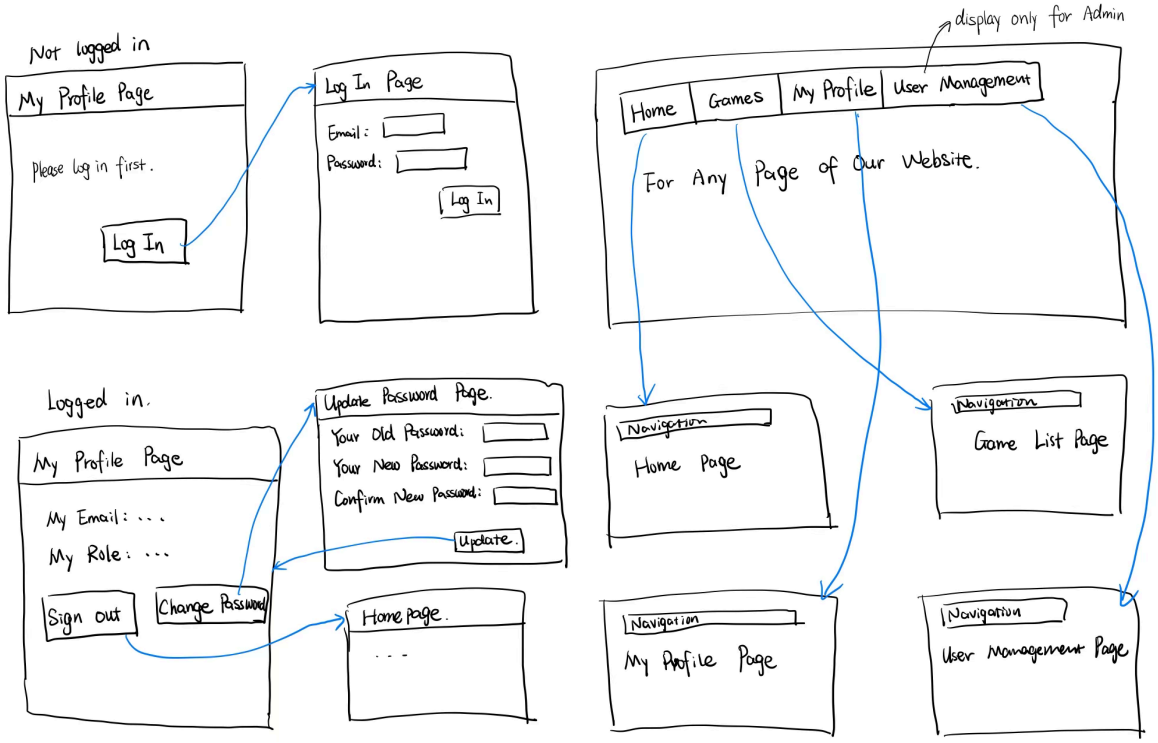


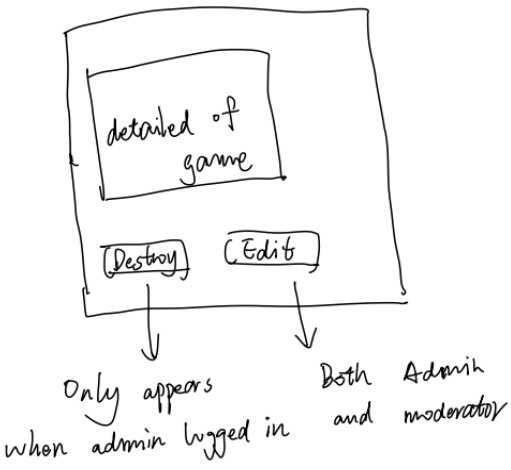
User Stories and UI Mockups for Sprint 4

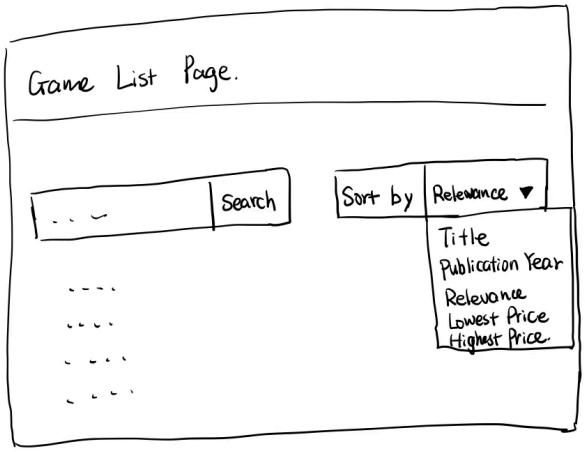
- Feature: Automatic logout after time
  Given points: 1. Status: implemented.
- Feature: Navigation bar
  Given points: 3. Status: implemented.
- Feature: ‘My Profile’ page creation
  Given points: 2. Status: implemented.
- Feature: Visit the game via the link
  Given points: 2. Status: implemented.
- Feature: Search games in interested categories
  Given points: 3. Status: implemented.
- Feature: Visualization update for game page
  Given points: 1. Status: implemented.
- Feature: Conditional display of user management in the navigation bar
  Given points: 1. Status: implemented.
- Feature: Sort search by various criteria
  Given points: 2. Status: implemented.
- Feature: Visualization of ‘My Profile’ page contents update with saved games
  Given points: 1. Status: implemented.

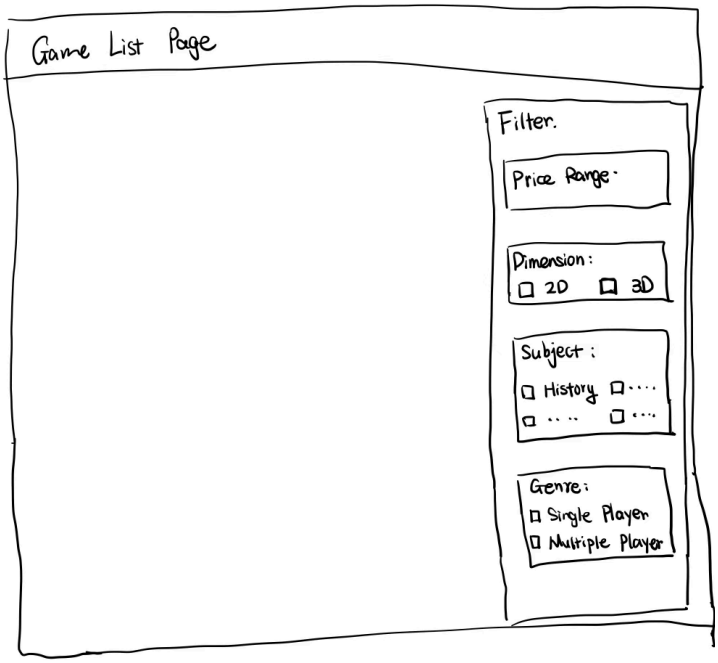
- Feature: Create an ‘About Us’ page that links to other social media sites
  Given points: 3. Status: implemented.
- Feature: Filter games by various criteria
  Given points: 3. Status: implemented.
- Feature: Conditional ‘Login’ or ‘Log out’ on the user authentication status
  Given points: 2. Status: implemented.
- Feature: Display the ‘edit’ option to admins
  Given points: 1. Status: implemented.
- Feature: Display ‘destroy’ option to admins
  Given points: 1. Status: implemented.
- +Feature: Save games if the user wants to save info in game index or game details page
  Given points: 3. Status: added after the sprint started, implemented.
- +Feature: Update the game details page by eliminating bad columns and reordering elements
  Given points: 2. Status: added after the sprint started, implemented.
- -Feature: Account password reset
  Given points: 2. Status: not implemented because we didn’t have enough time.
- -Feature: Vague search game title
  Given points: 2. Status: not implemented because we didn’t have enough time.
- Chore: ReadMe updated for project verification
  Given points: 3. Status: completed.

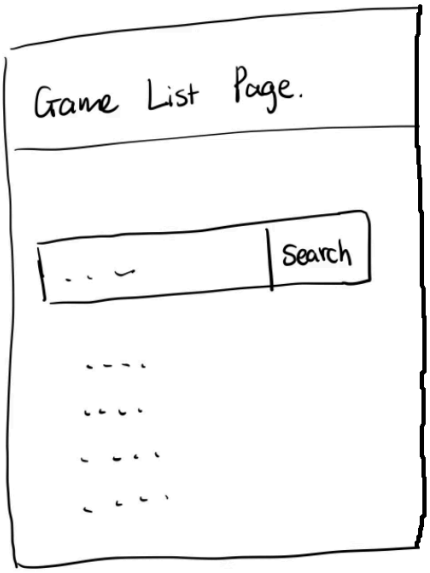
The corresponding UI Mockups for Sprint 4 are as follows:

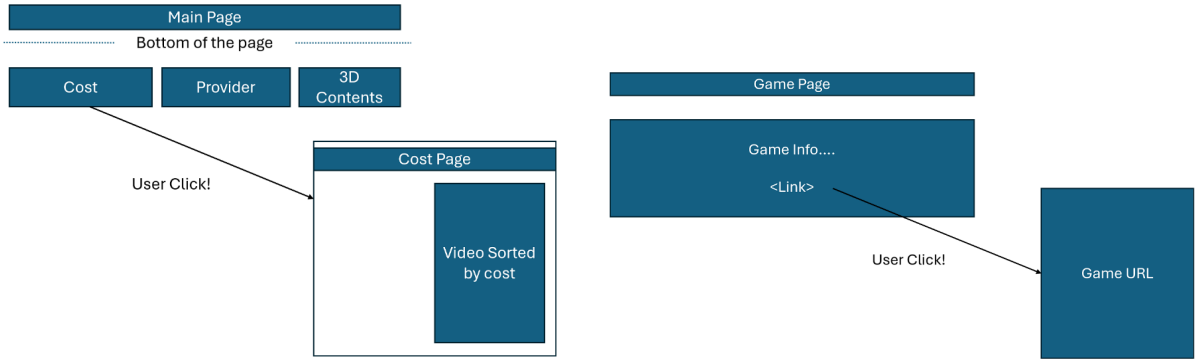



detailed of game
Destroy
Edit
Only appears when admin logged in
Both Admin and moderator


Game List Page.
Search
Sort by
Relevance
Title
Publication Year
Relevance
Lowest Price
Highest Price


Game List Page
Filter.
Price Range
Dimension:
2D
3D
Subject:
History
Genre:
Single Player
Multiple Player


Game List Page.
Search


Main Page
Bottom of the page
Cost
Provider
3D Contents
User Click!
Cost Page
Video Sorted by cost
Game Page
Game Info....
<Link>
User Click!
Game URL

Team Roles:

This project is divided into 4 sprints. At each sprint, the role is assigned as follows:

Role	Product Owner	Scrum Master	Developer
Sprint1	Zachary McDowell	Nilo Lisboa	Komo Zhang, Yeon Chae, Xiuyu Tang
Sprint2	Yeon Chae	Xiuyu Tang	Nilo Lisboa, Komo Zhang, Zachary McDowell
Sprint3	Nilo Lisboa	Komo Zhang	Zachary McDowell, Yeon Chae, Xiuyu Tang
Sprint4	Zachary McDowell	Komo Zhang	Yeon Chae, Xiuyu Tang, Nilo Barachisio Lisboa

Achievements in Scrum Iterations:

Throughout the sprint, our team's focus was based on clients' requests. Sprint 1 focused on initiating the project. Creating the main page of the website, implementing basic navigational features, establishing the database, and deploying the project on Heroku for customer accessibility were done **(23 pts completed).**

In Sprint 2, the primary focus is on implementing user account functionalities such as sign-up and login as per customer request. Additionally, displaying multiple games from the database, enhancing existing features with improvements like a return button to the main page, confirmation dialogs for game deletion, and expanding game information by adding more columns has been accomplished **(18 pts completed).**

In sprint 3, our team enhanced our login system to offer varying levels of access based on user roles. Following implementation, we customized page elements to be visible only to users with specific roles; for instance, administrators had sole authority to delete database entries. Additionally, we aim to enhance site aesthetics by embedding image links on web pages, ensuring a strong visual representation of games on the main, index, and details pages **(15 pts completed).**

In our last sprint, we implemented a range of new features and enhancements to existing functionalities. Additions include automatic logout after a period of inactivity, a navigation bar for easier site navigation, and the creation of a 'My Profile' page allowing users to manage their information. Users can now directly visit games via links and search for games within interested categories, while the visualization of game pages has been improved for better presentation. Conditional display of user management options based on roles, sorting search results by criteria, and visualizing 'My Profile' contents with saved games were also implemented. Additionally, an 'About Us' page linking to social media sites was created, and users can now filter games by various criteria. The conditional display of 'Login' or 'Log out', along with the

ability for admins to edit and destroy game entries, were added, alongside the option for users to save games with video. Furthermore, the game details page was optimized by removing unnecessary columns and reordering elements. Finally, we removed features related to account password reset and vague search game titles and updated the ReadMe for project verification **(31 pts completed).**

Stories and Points Completed by Each Member:

Role	Stories	Points	Total	Each Member
Sprint1	Chore: Create Heroku deployment for the application	3	6 User Stories, 9 pts	Xiuyu Tang
	Chore: Change deployment DB from Mongo to PostgreSQL	2		
	Feature: Allow any user to create new game entries in the application	1		
	Feature: Allow any user to delete games from the application	1		
	Feature: Allow any user to edit information on an education game's details screen	1		
	Chore: Fix Style Offenses with Rubocop so there is only one offense per file max	1		
	Feature: Implement a basic main page that introduces the application	3	4 User Stories, 9 pts	Yeon Chae
	Feature: Implement links to other functions on the main page	3		
	Feature: Allow users to create a new account	2		
	Feature: Create a login page for users	1		
	Feature: Populate database with CSV data given by LIVE Lab	2	2User Stories, 5 pts	Komo Zhang
	Feature: Create a database using MongoDB or PostgreSQL	3		

Role	Stories	Points	Total	Each Member
Sprint2	Modify the “show” page of game views, to show the new columns	2	4 User Stories, 8 pts	Nilo Lisboa
	Modify the “edit” page of game views, to show the new columns and make them editable.	2		
	Modify the “new” page of game views, to add text input boxes for the new columns	2		
	Modify the “index” page of game views to show only several selected columns for each game	2		
	change the example games on the main page	1	1 User Stories, 1 pt	Yeon Chae
	Confirmation dialog for destroying a game	2	2 User Stories, 4 pts	Komo Zhang
	Navigation back to the main page from the game list page	2		
	Post-Login Functionality	3	2 User Stories, 5 pts	Zachary McDowell
	Implement Sign Up and Login	2		

Role	Stories	Points	Total	Each Member
Sprint3	Feature: Split the game index page into multiple pages	1	4 User Stories, 7 pts	Xiuyu Tang
	Feature: Images on the main page redirect to the details page of the game.	1		
	Feature: Search games with keywords	3		
	Chore: Import new image URLs from the Image Link column in the CSV	2		
	Feature: Display Images for games on the details page	2	3 User Stories, 5 pts	Yeon Chae
	Feature: Display Images for games on the index page	1		
	Feature: Display Images for games on the edit page and allow the user to edit the image link	2		
	Feature: Create an administration page that allows administrator users to alter permissions on other existing users.	3	1 User Stories, 3 pts	Zachary McDowell

Role	Stories	Points	Total	Each Member
Sprint4	Automatic log out after timer	1	6 User Stories, 12 pts	Xiuyu Tang
	Navigation bar	3		
	'My Profile' page creation	2		
	Conditional display of user management in navigation bar	1		
	Sort search by various criteria	2		
	Filter games by various criteria	3		
	Search game in interested categories	3	5 User Stories, 10 pts	Yeon Chae
	Visit game via link	2		
	Visualization Update for Game Page (Image)	1		
	Save games if user want to save video in the game page	3		
	Visualization of 'My Profile' page contents update with saved games	1		
	Create 'About Us' Page that links to other social media sites	3	2 User Stories, 5 pts	Nilo Lisboa
	Update Game Details Page to match stakeholder request, eliminating bad columns and reordering elements.	2		
	Conditional 'LogIn' or 'Log Out' on user authentication status	2	3 User Stories, 4 pts	Komo Zhang
	Display 'edit' option to admin and mods	1		
	Display 'destroy' game option to admin users	1		

Customer Meetings:

- Friday 2/2/2024 3:00 pm
  First meeting with our customers, Dr. Michael Rugh, and his team. Two team members, Zach and Nilo, attended the meeting. Dr. Rugh described their main needs, including information about the intended users, main use cases, and the overall impact the project aims to achieve. The regular weekly meeting time with Dr. Rugh is set to 6:00 pm every Monday.
- Monday 2/5/2024 6:00 pm
  We reviewed our first Sprint Plan and discussed some functions with Dr. Rugh. To be specific, we confirmed with Dr. Rugh what features they wanted to show on the main page. We also demonstrated our initial deployment page, which includes the blank main page, game index page, and show/new/edit/delete game page. The client and our team agreed on using MongoDB as the production database.
- Tuesday 2/13/2024 6:00 pm
  This is the second weekly meeting. Regularly it is on Monday but rescheduled to Tuesday per our customer's request. Before the meeting, Prof. Ritchey suggested we use Postgres as the production database instead of MongoDB. We discussed this with our customer and decided to move to Postgres because it was more convenient to use on Heroku. We also demonstrated several features that have been finished up to now. These contain the following user stories: 1. Populate the database with CSV data given

by LIVE Lab; 2. Allow any user to create new game entries in the application; 3. Show links to other functions on the main page

- Monday 2/19/2024 6:00 pm
  Meeting for Sprint 1 Demo. Zach presented our deployment to Dr. Rugh and his team, showcasing an initial web page that consisted of a simplistic game browsing page, an example featured games, a section for the user to log in, and a search bar. Clicking the login button redirects the user to a page where they can enter a username and password. For those not registered, a corresponding “Registration” button is displayed to direct them through a sign-up process.
- Monday 2/26/2024 6:00 pm
  We reviewed the plan for the second Sprint with Dr. Rugh and his team. They provided suggestions for the game details page and visualization of the game.
- Monday 3/4/2024 6:00 pm
  We demonstrated some completed features to our customers: 1. Modify the “index” page of game views to show only several selected columns for each game; 2. Confirmation dialog for destroying a game; 3. Visualization of the Game.
- Monday 3/11/2024 6:00 pm
  Meeting for Sprint 2 Demo. As the scrum master, Yeon presented our deployment to Dr. Rugh and his team, showcasing the visualization of games, especially the game details page, visualizing the contents for each game. The contents are based on the database that our customer has shared. Additionally, we requested our customer to add an image URL for the visualization of each game, which will be added up in our following sprint. Regarding the login functionality, we share our current system with the customer.
- Monday 3/18/2024 6:00 pm
  We reviewed the plan for the third Sprint with Dr. Rugh and his team.
- Monday 3/25/2024 6:00 pm
  We demonstrated some completed features to our customers, including 1. Images on the main page redirect to the details page of the game; 2. Add Image Link field to New Game page; 3. Display Images for games on the details page and the index page; 4. Allow the user to edit the image link on a game’s details page; 5. Split the game index page into multiple pages.
- Monday 4/1/2024 6:00 pm
  Meeting for Sprint 3 Demo. As the scrum master, Nilo led the presentation of our sprint results to Dr. Rugh and his team. In particular, we demonstrated the newly implemented search function, the new formatting across the edit pages, game images that had been imported from the CSV, and the new user management screen that would allow website admins to monitor and alter user account roles.
- Monday 4/8/2024 6:00 pm
  We reviewed the plan for the last Sprint with Dr. Rugh and his team. Dr. Rugh mentioned that they wanted a website ready to present to their boss, so our team needed to replace some unuseful text and add their logos, header, footer, etc. to the proper positions of the website.
- Monday 4/15/2024 6:00 pm
  We demonstrated the new layout with their header and footer and our newly

implemented features including the navigation bar, user authentication for the role management page, and conditionally displaying the Login/Logout button based on the user's account status.

- Wednesday 4/24/2024 6:00 pm
  Last meeting with the client. We went over our last deliveries, final notes, and the customer survey. As the scrum master, Zachary led the presentation of our sprint results to Dr. Rugh and his team. The meeting consisted of the app's final added features, scrapped features, and the customer feedback survey that will be shared after our final report. The highlighted feature categories were a functional navigation system, a search functionality, sorting mechanics, and profile management. Additionally, we added Dr. Rugh's team logos and design to the HTML.

Analysis of BDD/TDD Practices:

During the development of this LIVE project, we implemented both Behavior-Driven Development (BDD) and Test-Driven Development (TDD) strategies, which helped us build high-quality and robust code. For the BDD process, we created detailed scenarios for each feature at the beginning of each sprint by using user-friendly language. We used Cucumber to write these scenarios and also documented them in the sprint plan. When our developers implemented a feature, they could look through the scenarios for the specific feature to figure out what to do to meet the customers' expectations. For the TDD process, we wrote tests, both Cucumber tests and RSpect tests, before adding code for the new features. RSpec was applied in our project to do unit testing. It helped us verify the functionality of each individual part. Cucumber was used to test whether the features worked as expected., we also used SimpleCov to get the test coverage. In the beginning, the tests are pending or failed. Then we started to write code to implement the new features under the guidance of these tests and tried to produce high-quality code and ensure a high code coverage.

As mentioned above, BDD and TDD processes help us build better and more robust code. However, they also brought some problems to our team. The major problem was that none of our team members were familiar with BDD or TDD. All of us had to learn these processes from the very beginning, which slowed down the project. Another problem that made us feel stressed was about modifying some previous features or modules. Since there were existing tests for the previous modules, any modification had the chance to cause failures in those tests. Fixing these failures was very time-consuming especially when the tests were originally written by a different person.

Configuration Management:

(Discuss your configuration management approach. Did you need to do any spikes? How many branches and releases did you have?)

Version Control:

We utilized Git as our version control system, hosted on GitHub. This setup allowed us to manage changes effectively, with distinct branches for different stages of development:

- *Master Branch:* Served as our main branch, where the production-ready code was maintained.
- *Feature Branches:* Temporary branches were created off the development branch for new features or significant changes, ensuring that the development branch remained stable. We have 42 branches.
- *Releases*: Heroku integrates seamlessly with GitHub, enabling automated deployments directly from the repository. We set up the workflow of deploying from the GitHub main branch. It was manually triggered since it could remind every developer to validate their changes after the manual deployment.

Spikes:
During the initial stages, we conducted a spike to determine the most suitable database for our application needs and deployment environment. This involved a trial of MongoDB and ultimately transitioning to PostgreSQL, which was better supported on Heroku and offered the necessary features we required, such as reliability and scalability for handling complex queries.

Database Configuration:
Our project started with MongoDB as the primary database; however, due to compatibility issues with Heroku, we transitioned to PostgreSQL. This decision was influenced by PostgreSQL's robust support of Heroku and its capability to handle our data requirements more efficiently.

- *Data Migrations*: We handled database changes through Rails migrations, which allowed us to modify the database schema in a controlled and reversible manner.

Challenges in the Heroku Production Release Process:

In the Heroku production release process, the first challenge we met was what to use as the production database. We tried MongoDB at first but found it was not easy to use on Heroku. There were several extra steps needed to set up the MongoDB on Heroku, which complicated our deployment process. After discussing with Prof. Ritchey and Dr. Rugh and because our data structure was very simple, our team decided to switch to Postgres, which could be easily used on Heroku as an add-on to our app.

Another challenge was how to maintain the production database during each release. We had several tables in the database. Initially, these tables were created by seed data. After we first deployed our app, our customers and users could dynamically modify the table contents but these modifications couldn't be stored in our seed file. During the second release, we simply reset the database to apply the new changes to a certain table. Then we found that the information added to other tables through the website since the first release was lost and couldn't be recovered. This would be a serious error if we really released the app to real users because the users would lose all the information they stored in the app. We realized that we should try to avoid resetting the production database and make a backup if we have to reset it.

In the following release, when we got new data about the video games from our client and needed to update the table for games, we only emptied the game table and ran `heroku run rails db:seed` to update the production database. We also noticed that destroying only the game table and then running `heroku run rails db:migration` would cause errors. The best way to update the content of only one table was to empty that table and seed again.

Besides these problems with the database, we also encountered issues with implementing the forget password functionality. We utilized Devise, which is a gem in Ruby on Rails for account management within our app and includes a built-in feature for password resetting. While setting up this operation, we configured an action mailer for a simple mail transfer protocol (SMTP). The reset password link was successfully delivered to the intended email address. However, this worked only when using personal Gmail accounts. When attempting to send emails from a newly created Gmail account, specifically for the LIVE program, we encountered some issues. It appears that this problem may be attributed to the age of the newly created account. We have identified this issue and left it for the consideration of future development teams.

Development Tools and Gems

(Describe the tools/Gems you used, such as GitHub, CodeClimate, SimpleCov, and their benefits and problems)
The development of the LIVE project leveraged several tools and Ruby gems that enhanced our productivity and ensured high-quality code. Here's a detailed look at some of these tools:

Github:

- *Benefits:* Served as a central repository for our code, facilitating collaboration, code review, and version control. Integrated well with other services like Heroku and Code Climate, simplifying our CI/CD pipelines.
- Problems: We haven't set up the auto-test workflow for Rspec and cucumber, which brings us a little repeating manual work

Code Climate:

- *Benefits:* Provided automated code reviews for maintainability and potential security vulnerabilities. It helped us improve code quality by offering insights and suggestions for refactoring.
- Problems: Sometimes the metrics provided by Code Climate were too rigid or didn't align perfectly with our project goals, requiring us to adjust our coding practices or configurations to meet its standards.

Devise:

- *Benefits:* Devise is highly modular, which makes it can be easily integrated with our project. It provides various straightforward authentications, and we implemented our basic username and password auth with little effort. Gem installation and rails compatibility is also a very good aspect of this dependency.
- *Problems:* While it's so powerful it introduces too many unnecessary dependencies and configurations we don't need. Which makes it a bit overkill.

Rubocop:

- *Benefits:* It helps maintain high quality and consistent code style, especially in a group project every developer has their own flavor of code style. It can auto-correct the obvious defenses which makes life much easier.
- *Problems:* Configuring and maintaining perfect style guidelines could be time-consuming. The default configuration does not fit for every project, so we have to customize the configurations but the documentation is not so obvious.

Cucumber-Rails:

- *Benefits:* Cucumber allows you to write tests in almost plain English, which makes the tests more intuitive, readable, and understandable. The tests themselves are the comments, there is no need to explain each test.
- *Problems:* Cucumber separates the features and the steps, and the steps can be located in any file. It makes it a little hard to manage the steps, sometimes one step collides with the other one, and sometimes steps repeatedly violate the "DRY" principle.

SimpleCov:

- *Benefits:* This gem was crucial for monitoring code coverage across our test suite, ensuring that we maintained high test coverage throughout development.
- Problems: On occasion, SimpleCov's coverage reports could be misleading if not configured correctly, especially when integrating with parallel test processes or dynamic code execution paths.

Heroku CLI:

- *Benefits:* Essential for managing our Heroku deployments, allowing us to run commands to deploy applications, scale resources, and manage database migrations directly from the command line.
- Problems: Required a learning curve for team members unfamiliar with command-line interfaces, and troubleshooting deployment issues sometimes required in-depth logs analysis.

By utilizing these tools and gems, we were able to maintain a high standard of code quality, manage our application configuration efficiently, and ensure smooth and consistent deployments. Each tool, while presenting its own set of challenges, significantly contributed to the success of our project development cycle.

Repository Overview:

(Make a separate section discussing your repo contents and the process, scripts, etc., you use to deploy your code. Make very sure that everything you need to deploy your code is in the repo. We have had problems with legacy projects missing libraries. We will verify that everything is in the repo.)

Contents:

- app - the web app. controllers, models, views, etc...
  | - assets - images and CSS files
  | - controllers - controllers accept requests and return responses.
  | - models - models provide algorithms corresponding to the actual models.
  | - views - views actually display on the portal.
- bin. - rails executables.

- config - dependency and environment configurations.
- db - database-related files.
  | - migrate - rails migrate auto-generated files. Each file contains 1 step of database change, adding columns, adding index, etc...
  | - schema - database schema for all the tables in the app.
  | - seeds - defines how the data is imported into the database
- documentation - Group discussion minutes, including plans and retros.
- features - cucumber tests
  | - steps_definitions - contains all the cucumber steps
  | - *.feature - the actual cucumber tests
- spec - rspec tests
  | - controllers - controller tests like responses to the requests
  | - features - feature tests, like role-based control
  | - models - model tests like sorting
  | - requests - requests' response test without knowing the controllers
  | - view - view tests like what to display

Links:

Github Repo: https://github.com/yeonchae62/LIVE
Heroku Deployment: https://evg-library-8a920fa9c3cb.herokuapp.com/
Code Climate Report: https://codeclimate.com/github/yeonchae62/LIVE
Presentation and Demo: https://www.youtube.com/watch?v=m_TyqeothPk