Written by Jacob Levy, Trevor Koehler, Nathan Batten

Part A

Naming Written together

We decided on the website name of *GGTracker*, with a simple, to-the-point team name of *GGTracker Team*.

Intro and Motivation Written by Jacob

The goal of our website is to provide a concise, easy-to-use method of tracking video games and finding recommendations for new video games. Games can be sorted by genre, and users can track their personal collection or playing history, along with playtime, whether or not they enjoyed the game, and the start and end dates of their time playing the games. The website will, ideally, be able to provide recommendations based on playtime, commonly played genres, and average ratings given within a genre.

We're aiming to provide a "one-stop shop" for people both trying to find new games that they would enjoy and for those who just want to keep track of what games they've played. It also allows people to easily discuss games they've played, by showing others their list, and to ensure that users don't forget what games they've played or how long they played them - the "tracking" side of the website is essentially aiming to work the way that a myanimelist.net or imdb.com watchlist would.

Business Logic Requirements

Key Data Needs Written by Trevor

There are five tables that need to be stored in our database for our website to function properly. We need tables to store Games, Genres, and Users, as well as M-M tables linking Games-Genres and Games-Users. Each of these have at least two columns, as described below:

- Games
    - Name
    - ID [Primary Key/Identifier]
    - Description
    - Image URL
- Genres
    - ID [Primary Key/Identifier]
    - Name/Type
- Users
    - ID [Primary Key/Identifier]

        - Username
        - Password
        - Personalization options
            - Description
        - Admin state
    - GamesGenres Linking Table
        - Game ID [Foreign Key - Games.ID]
        - Genre ID [Foreign Key - Genres.ID]
    - UsersGames Linking Table
        - User ID [Foreign Key - Users.ID]
        - Game ID [Foreign Key - Games.ID]
        - Playtime
        - Rating (positive or negative)
        - Optional bonus columns
            - Start time
            - End time

Rest API Table written by Nathan

Purpose	HTTP Method	Path	Controller	Controller Method	Model Method	Database Method
Home screen	GET	/ or /home	home	renderHome	NA	NA
Handle invalid url	*	*	error	renderError	NA	NA
Create game	POST	/games	game	addGame	addgame	INSERT
List games	GET	/games	game	listGames	getGameList	SELECT
Remove game	POST	/games/delete	game	removeGame	deleteGame	DELETE
Edit game	POST	/games/edit	game	editGame	editGame	UPDATE
Genre add	POST	/genres	genre	addGenre	addGenre	INSERT
Genre list	GET	/genres	genre	listGenres	getGenresList	SELECT
Genre remove	POST	/genres/delete	genre	removeGenre	deleteGenre	DELETE
Genre edit	POST	/genres/edit	genre	editGenre	editGenre	UPDATE
User add	POST	/users	user	addUser	addUser	INSERT
User list	GET	/users	user	listUsers	getUsersList	SELECT
User remove	POST	/users/delete	user	deleteUser	deleteUser	DELETE

User edit	POST	/users/edit	user	editUsers	editUsers	UPDATE

Data Rules Written by Jacob

Every table has slightly different data rules, but certain aspects are shared. IDs must all be non-null, unique integers, and foreign key integers must always point to existing entries in their respective tables. Name and type entries must be non-null strings. Certain columns, however have specific requirements:

- Games.Description - nullable string
- Games.Image - nullable string, but must point to an image if non-null
- Users.Password - must be a non-null, hashed string
- Users.Description - nullable string
- Users.Admin - non-null boolean (1 or 0) value
- UsersGames.Playtime - non-null, positive integer (0 inclusive), tracked in hours
- UsersGames.Rating - nullable boolean value representing either positive or negative
- UsersGames.Start and UsersGames.End - nullable datetime

UI Requirements Written by Jacob

There needs to be UI representation of much of the data available in the database, but not all of it, since it's not all available to users. There needs to be:

- A list of games, where:
    - They can be added to a user's collection - including options to set start and end dates, playtime, and rating.
- A user creation screen, with:
    - Input options for username, password, and description.
    - Verification to avoid duplicate usernames.
- A way for users to see their own games, where:
    - They can edit any personalized information on a game they own (such as playtime)
- Admin users should have access to:
    - A list of users
    - The option to modify or delete users (but NOT user passwords)
    - To add or remove games and genres.

These UI elements need to be presented using various HTML and BootStrap elements. Anything that the common user will see (such as games lists) should be created using BootStrap cards, and should use the color scheme entirely, as well as being as visually appealing and easy-to-use as possible. However, admin pages (such as users lists), should be more "raw" pages, providing simply the information requested and basic functionality - no "fluff" is required, and basic HTML tables can be used. The purpose of those pages is simply to provide information and capabilities to the site admins, not to be used by common users, and normal users shouldn't even be able to load them without getting errors.

Part B

Interface Design Written/designed by Trevor

COLORS:

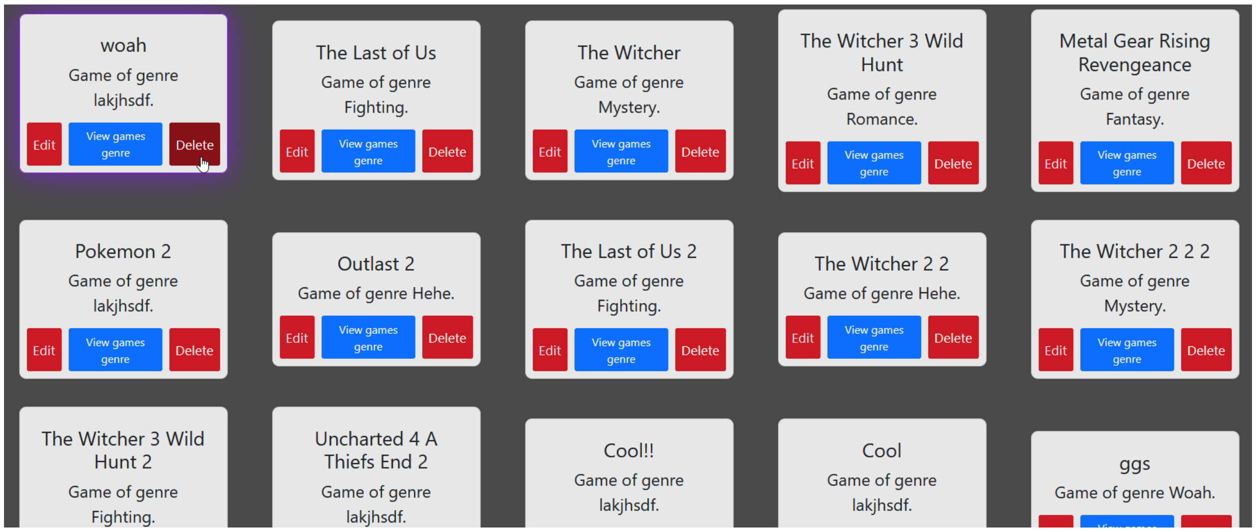
The core color scheme uses black, white, and red. For various actions like hover or confirming, or popups, different shades of the colors can be used. You can also use colors like purple for "glow" effects. But it should not be used in the core UI. Blue can be used on cards, but should otherwise be avoided.
SCHEME: https://coolors.co/7a28cb-cb1a23-000000-494949-f7f7ff
Header and logo text will use the tokyo drifter font: https://www.dafont.com/tokyo-drifter.font

Design goals+justification

The general feel should be very "cool." The goal is to fit to a bit of a gamer aesthetic, while keeping a touch of the feel from modern UI. Keeping dark crisp colors with just flashy reds for the UI, but still having rounded edges and card/modal based UX. This is aimed to *appeal* to those who more closely bring themselves to "gamer" aesthetics, while remaining accessible to more casual users, or those who may not be a fan of that kind of style over substance. The design should look cool, but remain very approachable.

Visual design concepts

For the basic games list, it is akin to the following design.



woah
Game of genre lakjhsdf.

The Last of Us
Game of genre Fighting.

The Witcher
Game of genre Mystery.

The Witcher 3 Wild Hunt
Game of genre Romance.

Metal Gear Rising Revengeance
Game of genre Fantasy.

Pokemon 2
Game of genre lakjhsdf.

Outlast 2
Game of genre Hehe.

The Last of Us 2
Game of genre Fighting.

The Witcher 2 2
Game of genre Hehe.

The Witcher 2 2 2
Game of genre Mystery.

The Witcher 3 Wild Hunt 2
Game of genre Fighting.

Uncharted 4 A Thiefs End 2
Game of genre lakjhsdf.

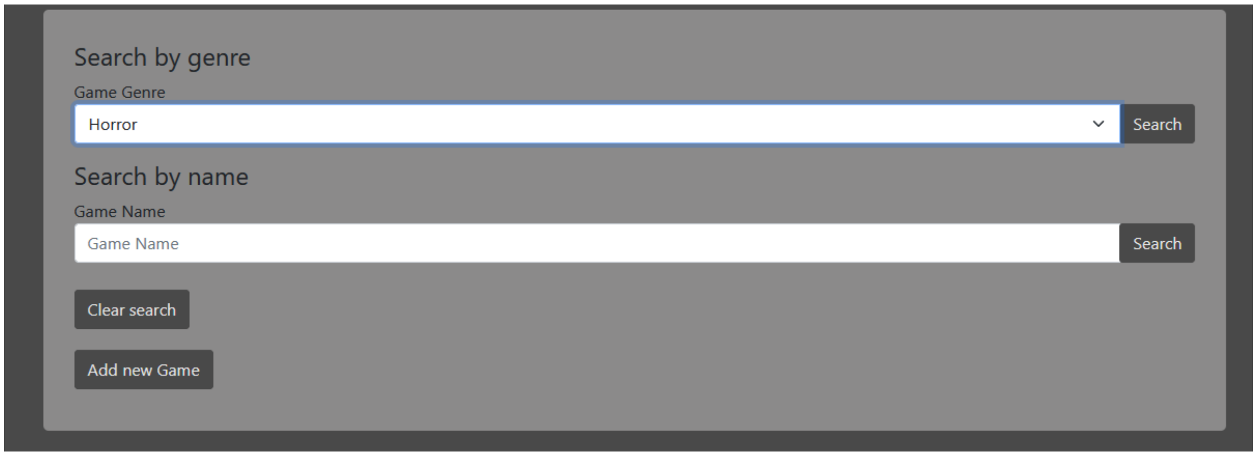
Cool!!
Game of genre lakjhsdf.

Cool
Game of genre lakjhsdf.

ggs
Game of genre Woah.

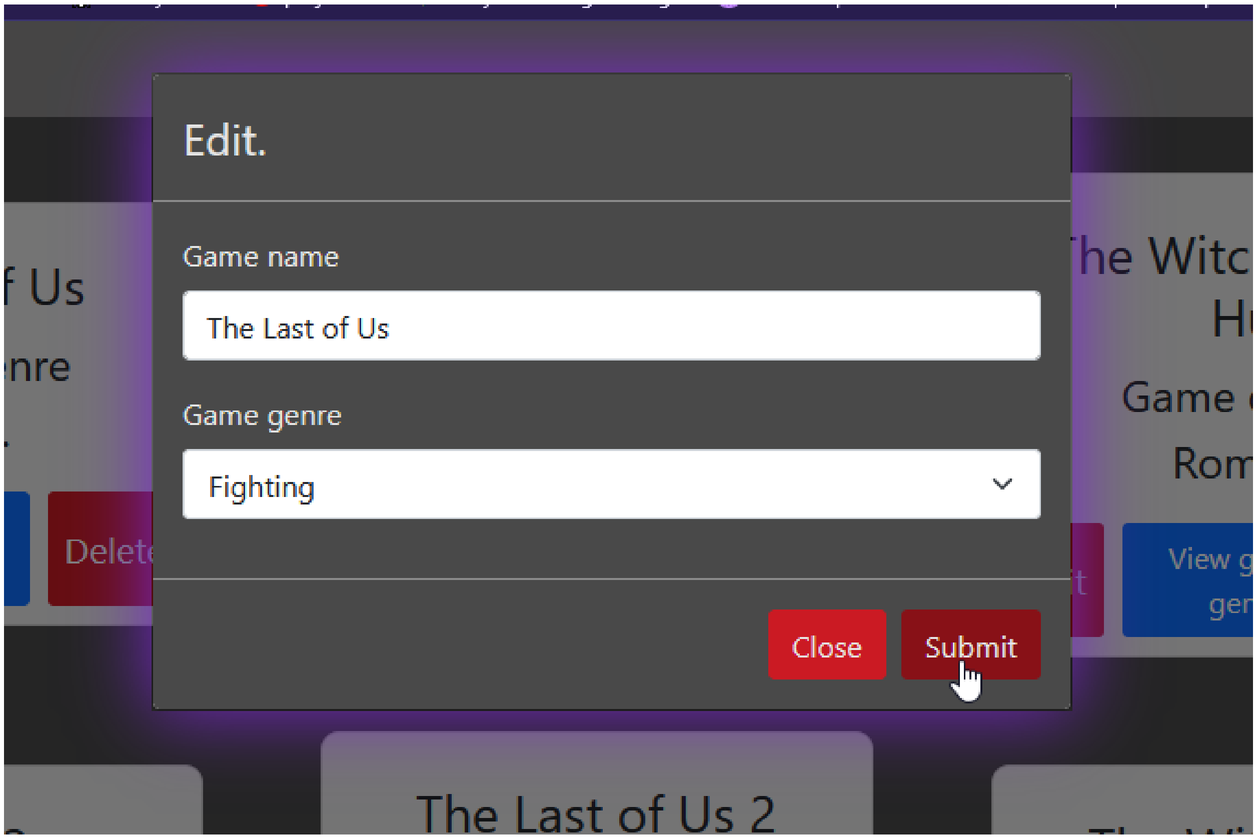
Cards push themselves up a bit and glow when hovered over. This is a tiny bit of added polish to try and make the app feel more “fun” to use. Each card will have basic info on the game, and in the future, a small header with an image for the game.
The same view will be used for genres, and viewing a users games.

There are several situations where a search UI is fitting, for this, put a slightly lighter grey box around a form, in which one line is one potential search property.



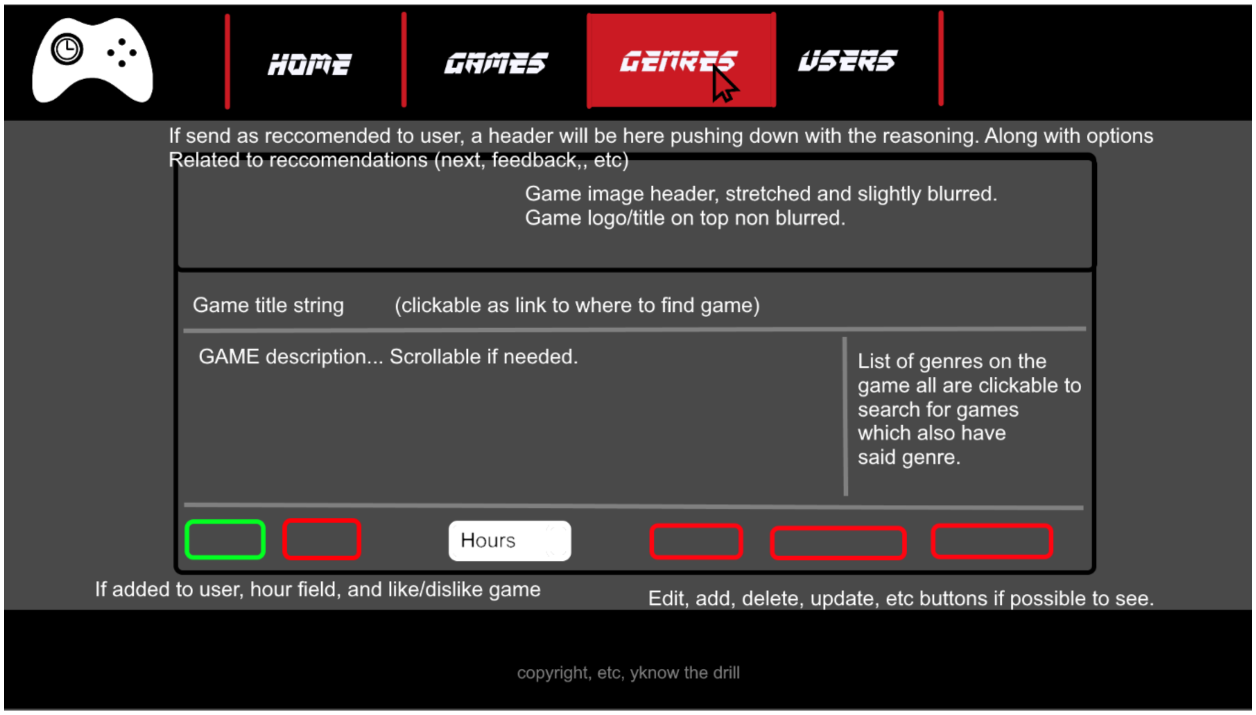
If the search is just from that input, the search bar should be to the right. If there is more, it should be clearly illustrated in a bordered section, with a search bar on the bottom. Admin elements for list views should be added here as well, as it is easily accessible, and is not important enough to justify a larger interface, since only administrators will be using it.

Adding, deleting, modifying, etc on any element should use a popup modal. I.e.



Popup modals should have a glow relevant to the form of action. So modifying may have a purple glow, but deleting something will have a red glow. This communicates for more attention if the user is performing a more severe/undoable action.

Games can be viewed in individual popups, which take form in a view themselves. We go to this view if a user just added the game to their profile, got recommended the game, or decided to view the game from a normal game search.



If send as reccomended to user, a header will be here pushing down with the reasoning. Along with options Related to reccomendations (next, feedback,, etc)

Game image header, stretched and slightly blurred. Game logo/title on top non blurred.

Game title string (clickable as link to where to find game)

GAME description... Scrollable if needed.

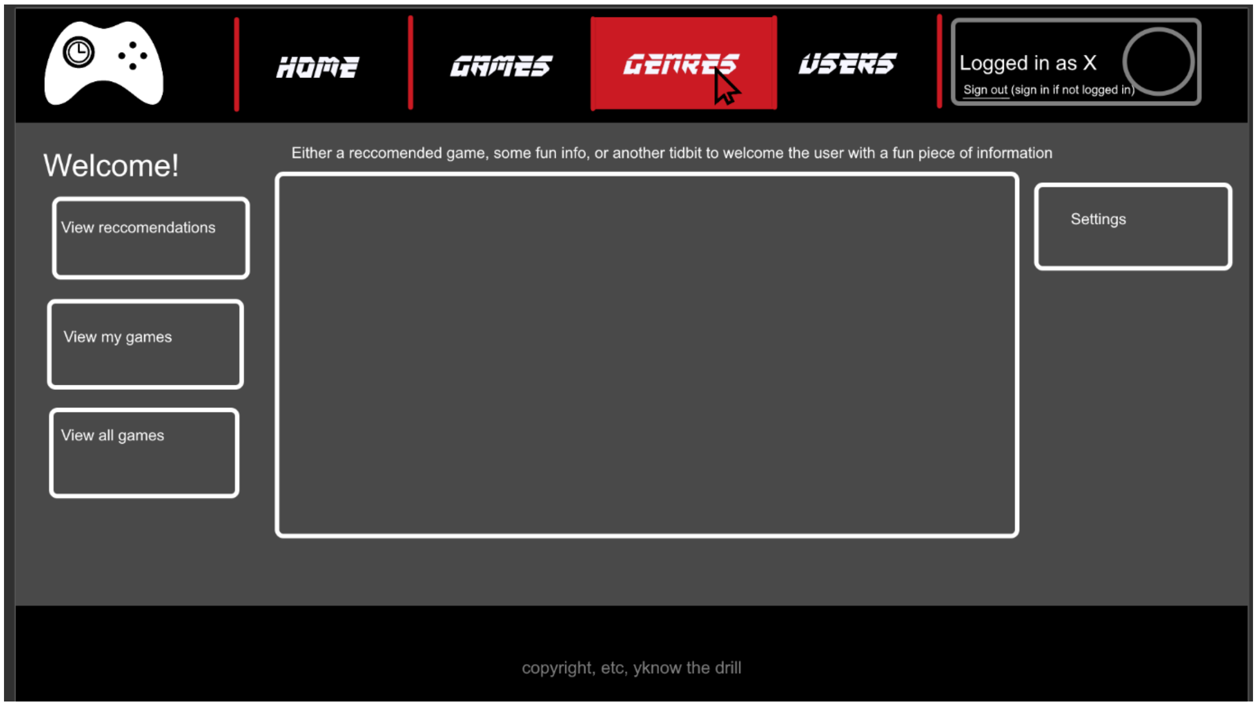
List of genres on the game all are clickable to search for games which also have said genre.

If added to user, hour field, and like/dislike game

Edit, add, delete, update, etc buttons if possible to see.

copyright, etc, yknow the drill

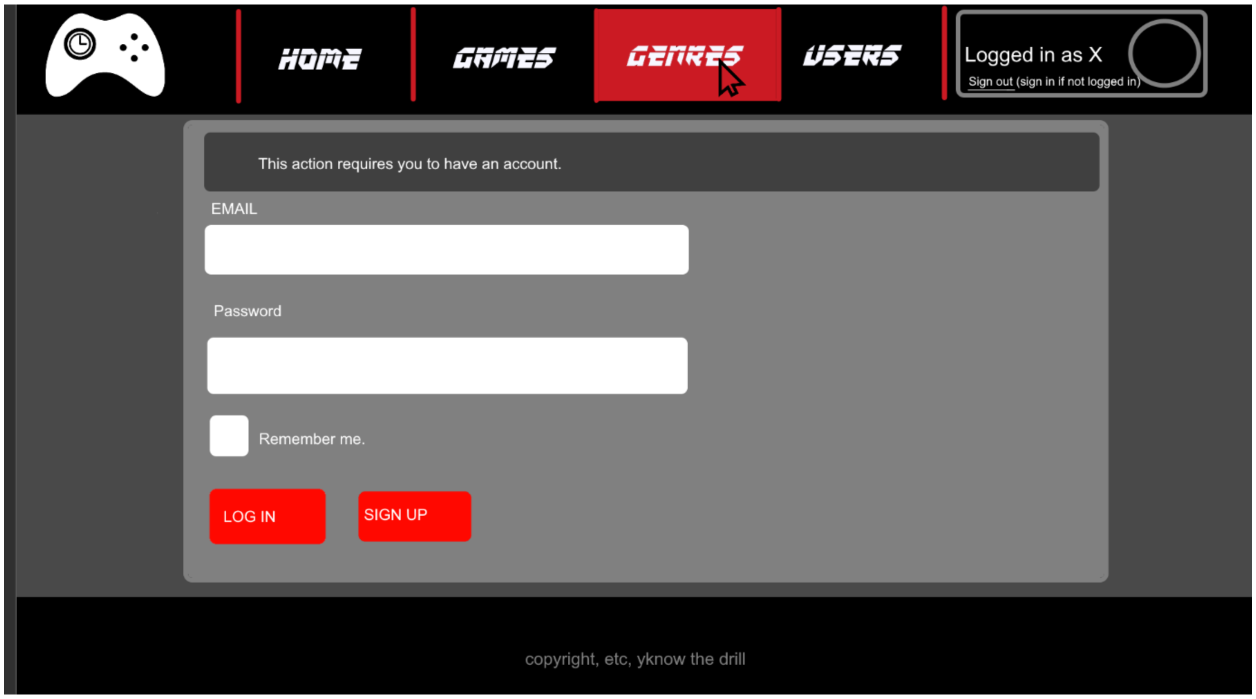
Finally, when a user logs in, they are directed to their home page. The home page will contain easy access to pages relevant to them, and a fun piece of information chosen by the server for the user. Like a recommended game a list of stats, etc.



Either a reccomended game, some fun info, or another tidbit to welcome the user with a fun piece of information

copyright, etc, yknow the drill

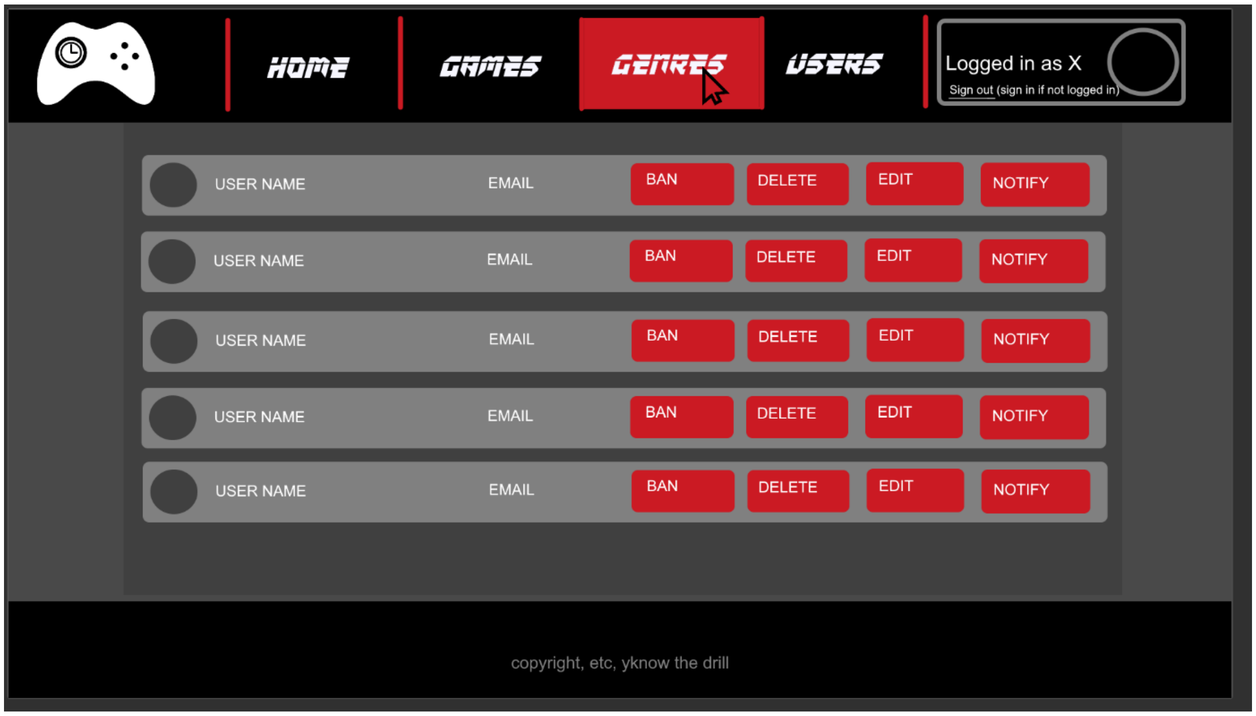
There is also a little user indicator in the corner for when one is not logged in or is logged in. When not logged in, a user can still browse games and genres, but as soon as they attempt to perform an action, they will be redirected to the login page with a message explaining. The login page is straight forward. It also doubles as a sign up page.



This action requires you to have an account.

copyright, etc, yknow the drill

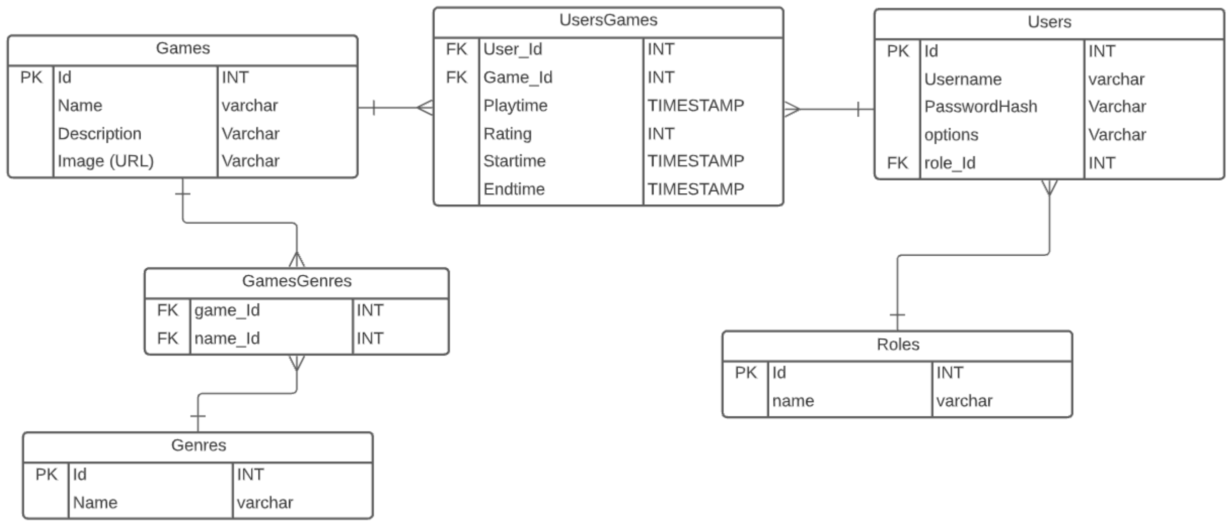
For administrator views of users, it is a straightforward list.



copyright, etc, yknow the drill

This side of the site does not need to be pretty, just functional. It will use the modal system regardless, but glow is not required, animations aren't required, polish can be minimal. This page’s purpose is just for the admin to perform actions on users if needed.

Database Design (UML) draw by Nathan



Application Features Written by Jacob

Required Features

Below is a list of the "required features" for our site to be considered finished, or what we would essentially call the minimum for a functioning site. These features must be implemented before May 20th, and should be fully implemented by May 10th, assuming the Milestones are balanced fairly. Most of the features listed involve Model functions, Controller functions and endpoints, and View templates and designs, but the exact requirements for each feature will be listed in the **UI, User, and Back-End Requirements** section. For the purposes of the following list, "visitor" means any site visitor (regardless of login status), "site user" means logged-in user, and "administrator" means site administrator

- Games List
    - Will allow the user to view all games on the site, sorted either:
        - Not at all
        - By Genre
        - By Search
    - Accessible to any visitor, ignoring login status
- Game View
    - Will allow the user to view all information relevant to a specific, selected game
    - Accessible to any visitor, ignoring login status
- Genres List
    - Will allow the user to view all genres on the site, optionally sorted by Search

    - Accessible to any visitor, ignoring login status
- Genre View
    - Will allow the user to view all information relevant to a specific, selected genre
    - Accessible to any visitor, ignoring login status
- Users List
    - Will allow administrators to view all site users
    - Accessible to administrators only
- Users Add/Sign-up and Log-in
    - Will allow the user to sign in to an existing account or create a new account
    - Accessible to visitors only - the user must not be signed in
    - Will include password hashing
    - Will ensure that there are no duplicate usernames
- Games Add
    - Will allow administrators to add games to the database
    - Accessible to administrators only
- Genres Add
    - Will allow administrators to add genres to the database
    - Accessible to administrators only
    - Will ensure that there are no duplicate genres
- Games Remove
    - Includes View, Model, and Controller
    - Accessible to administrators only
    - Must also remove the relevant Games-Genres entries
- Genres Remove
    - Will allow administrators to remove games from the database
    - Accessible to administrators only
    - Will also remove the relevant GamesGenres database entries
- Users Remove
    - Will allow administrators to remove any user
    - Accessible to administrators only
    - Will also remove the relevant GamesUsers entries
- Users Edit
    - Will allow users to modify their own information (username, description, etc)
    - Will be accessible to logged in users/administrators only, on themselves
- Games Edit
    - Will allow administrators to modify existing games in the database
    - Accessible to administrators only
- Genres Edit
    - Will allow administrators to modify existing genres in the database
    - Accessible to administrators only
- Collection Add
    - Will allow users to add games to their collection
    - Will be accessible to logged in users, on themselves (adding to their collection)
- Collection Edit

    - Will allow users to modify their collection
    - Will be accessible to logged in users, on themselves (modifying their collection)
- Collection List
    - Will allow users to view someone’s collection
    - Will be accessible to any visitor
- Recommendations
    - Will provide recommendations for new games that users should play based on their play history, play time, etc
    - Will act based on personalized user collection
    - Accessible only to logged in users/administrators, based on their collection
- 404 Error Handling
    - Functioning 404 error pages for invalid endpoints
    - Accessible to anybody
- 500 Error Handling
    - Functioning 500 error pages for server-side errors
    - Accessible to anybody
- 400 Error Handling
    - Functioning 400 error pages for user-caused errors (such as input errors)
    - Accessible to anybody
- Upgrade Users to Admin
    - Allows administrators to promote other users to administrator
    - Accessible only to administrators

Stretch Goals

- User Self-Delete
    - Will allow a user to delete their own account
    - Accessible to any logged in user/administrator
- User Privacy
    - Will allow users to set their own privacy settings, making their pages unavailable for viewing from other users (in case of private)
    - Accessible to logged in users/administrators, to modify themselves
- More User Information
    - Will allow users to add more information to their account, such as descriptions, profile pictures, etc
    - Accessible to logged in users/administrators, to modify themselves
    - Administrators can modify other users
- Notifications
    - Will allow users to get notifications about games in their collection (for example, if a game has been removed from the database)
    - Accessible to logged in users/administrators, for their own notifications

UI, User, and Back-End Requirements

Feature	**Minimum User Status**	**Controller**	**Model**	**View**	**Database Table(s)**
Games List	Visitor+	🟩	🟩	🟩	Games GamesGenres
Game View	Visitor+	🟩	🟩	🟩	Games GamesGenres
Genres List	Visitor+	🟩	🟩	🟩	Genres GamesGenres
Genre View	Visitor+	🟩	🟩	🟩	Genres GamesGenres
Users List	Administrator	🟩	🟩	🟩	Users
Users Add / Sign-up / Login	Visitor	🟩	🟩	🟩	Users
Games Add	Administrator	🟩	🟩	🟩	Games GamesGenres
Genres Add	Administrator	🟩	🟩	🟩	Genres GamesGenres
Games Remove	Administrator	🟩	🟩	🟩	Games GamesGenres
Genres Remove	Administrator	🟩	🟩	🟩	Games GamesGenres
Users Remove	Administrator (Any) User (Self)	🟩	🟩	🟩	Users GamesUsers
Users Edit	User (Self)	🟩	🟩	🟩	Users
Games Edit	Administrator	🟩	🟩	🟩	Games GamesGenres
Genres Edit	Administrator	🟩	🟩	🟩	Genres GamesGenres
Collection Add	User+ (Self)	🟩	🟩	🟩	GamesUsers
Collection Edit	User+ (Self)	🟩	🟩	🟩	GamesUsers
Collection List	User+ (Self)	🟩	🟩	🟩	GamesUsers

Recommendations	User+ (Self)	🟩	🟩	🟩	GamesUsers
404 Error Handling	Visitor+	🟩	🟥	🟩	N/A
500 Error Handling	Visitor+	🟩	🟩	🟩	N/A
400 Error Handling	Visitor+	🟥	🟩	🟥	N/A
Users -> Admin	Administrator	🟩	🟩	🟩	Users
User Self-Delete	User+ (Self)	🟩	🟩	🟩	Users GamesUsers
User Privacy Settings	User+ (Self)	🟩	🟩	🟩	Users
Detailed User Information	User+ (Self)	🟩	🟩	🟩	Users
Notifications	User+ (Self	🟩	🟩	🟩	Users Games Genres GamesUsers GenresUsers

Coding Techniques written by Nathan

Technologies Used

We had made the decision to use various technologies in our website, which includes some essential technologies such as NodeJS and Express as well as some frameworks to improve the user experience on our website, such as Bootstrap. The technologies we will be using are as follows:

NodeJS Frameworks	**Languages and Others**
- Express - Handlebars - Jest - Bootstrap - Supertest - Puppeteer (jest-puppeteer) - Pino - JSDoc - Validator	- JavaScript - NodeJS - Docker - MySQL - CSS - HTML

Coding and Documentation Practices Written by Nathan

Documentation:

- We have decided to use JSDoc as our general documentation practice so we can generate a useful and informative API. This includes parameters, return values, description of methods, etc.

Project Coding practice:

- Project practices
    - We will be using MVC as a project structure practice,this allows us to very easily separate the assignment into 3 distinct parts, the model, the controller and the view. Which can also be very easily divided into tasks when coding within a group.

- Error practices
    - We will use the Pino.js web framework for enhanced logging, this will allow us to have multiple logging levels. We will be using structured logging to log JSON objects, also will also use our own logging module to use throughout our program

- Code Style Practices
    - Start curly braces on the same line as the method statement
    - lowerCamelCase for constants variables and functions
    - UpperCamelCase for classes
    - Use const when you can but default to let, var should be avoided
    - Use strict equals (===) over abstract equals (==) when possible

Part C

Timeline Written by Jacob

Milestone 1

Milestone 1 is planned to include mostly a combination of our respective Assignment 3 and Assignment 5 code. When completed, it should include at least a basic user authentication function, the fully-functioning models for the Games, Genres, and Users tables, the linking GamesGenres and GamesUsers tables, and fully functioning front-end and back-end. The front-end will be very simplistic, and mostly just a holdover from the Assignment 3 work.

We plan to have this milestone completed, or at worst almost completed, by May 3rd.

Milestone 2

Milestone 2 is planned to include the recommendation system, a functioning user creation and (potentially) account recovery feature, and an overhaul on the front-end UI according to the updated design templates. The back-end Games and Genres tables will also be filled with a lot of example games, ideally automatically, and more information will be added to games and genres, such as descriptions and poster screenshots.

We plan to have this milestone completed, or at worst almost completed, by May 10th.

Milestone 3

Milestone 3 is planned to be a stretch goal milestone, encompassing extra goals and touch-ups more than completely new features. It will also be used as a buffer period between Milestones 1 and 2 before the final submission date, allowing for unexpected errors and issues that may slow them down. Ideally, we will be able to polish the code, use a custom JSDoc module, add smaller features (such as more detailed recommendations and Steam account linking), and make the website more easily deployable.

This milestone will be completed by May 20th, the final submission date, but will be aimed to be completely a few days before.

Tasking Plan

Written by Trevor

Tasks

Link to spreadsheet:
https://docs.google.com/spreadsheets/d/1i7bHHwBd6B4FN5h1JdhMnLSp0b_Yqy9p363MzrlIFHk/edit?usp=sharing

Summary	Assignee	Due date	Sprint	Epic
Create a way to delete user accounts as admin	Jacob Levy	May 3, 2022	1	UI/Users
Create a view for admins to create user accounts	Jacob Levy	May 3, 2022	1	UI/Users
Implement endpoints for deleting genres, ensure error handling.	Nathan Batten	May 3, 2022	1	Controller/Genres
Implement endpoints for changing a genre name	Nathan Batten	May 3, 2022	1	Controller/Genres
Implement endpoints for adding new genres.	Nathan Batten	May 3, 2022	1	Controller/Genres
Implement endpoints to get genres from a game.	Nathan Batten	May 3, 2022	1	Controller/Genres
Implement endpoints to get genres from genre name	Nathan Batten	May 3, 2022	1	Controller/Genres
Implement endpoints to get games from its genres.	Nathan Batten	May 3, 2022	1	Controller/Games
Implement endpoints to search for a game from part of its name	Trevor Koehler	May 3, 2022	1	Controller/Games
Implement endpoint for deleting games	Trevor Koehler	May 3, 2022	1	Controller/Games
Implement endpoint for removing genres from games	Nathan Batten	May 3, 2022	1	Controller/Games

Implement endpoint for adding new genres to games.	Nathan Batten	May 3, 2022	1	Controller/Games
Implement endpoint for adding new games	Trevor Koehler	May 3, 2022	1	Controller/Games
Basic routing and test files, file structure	Trevor Koehler	May 3, 2022	1	Core
Implement basic express handling	Trevor Koehler	May 3, 2022	1	Core
Create git repo	Trevor Koehler	May 3, 2022	1	Core
Implement a way to delete user accounts and all relevant methods	Jacob Levy	May 3, 2022	1	Model/Users
Implement a way to set a user as an admin	Jacob Levy	May 3, 2022	1	Model/Users
Implement methods for adding a new user account	Jacob Levy	May 3, 2022	1	Model/Users
Implement methods for deleting a genre and handling/options if games rely on the genre	Nathan Batten	May 3, 2022	1	Model/Genres
Implement methods for changing a genres name.	Nathan Batten	May 3, 2022	1	Model/Genres
Implement methods for adding a new genre (NO DUPLICATE)	Nathan Batten	May 3, 2022	1	Model/Genres
Implement methods for getting genres from a game.	Nathan Batten	May 3, 2022	1	Model/Genres
Implement methods for getting genres by part of its name	Nathan Batten	May 3, 2022	1	Model/Genres
Implement methods for getting genre by name.	Nathan Batten	May 3, 2022	1	Model/Genres
Implement methods for getting genres by id.	Nathan Batten	May 3, 2022	1	Model/Genres
Implement methods for deleting games and ensuring all connections are severed, likely implement a way to notify users using the games.	Trevor Koehler	May 3, 2022	1	Model/Games
Implement methods for changing game genres.	Trevor Koehler	May 3, 2022	1	Model/Games
Implement methods for updating game names	Trevor Koehler	May 3, 2022	1	Model/Games
Implement methods for adding new games	Trevor Koehler	May 3, 2022	1	Model/Games
Implement methods for searching for a game by part of its name	Trevor Koehler	May 3, 2022	1	Model/Games
Implement methods for getting games by name	Trevor Koehler	May 3, 2022	1	Model/Games
Implement methods for getting games by genre.	Trevor Koehler	May 3, 2022	1	Model/Games
Implement methods for getting games by id.	Trevor Koehler	May 3, 2022	1	Model/Games
Make database connection accessible	Jacob Levy	May 3, 2022	1	Model/Base
Make commands for making database tables	Jacob Levy	May 3, 2022	1	Model/Base
Implement endpoints to make new accounts.	Jacob Levy	May 3, 2022	1	Controller/Users
Implement endpoints to search for a genre from part of its name.	Nathan Batten	May 3, 2022	1	Controller/Genres

Implement endpoint for modifying games.	Trevor Koehler	May 3, 2022	1	Controller/Games
Implement endpoints to search for a game from its name.	Trevor Koehler	May 3, 2022	1	Controller/Games
Implement endpoints to get games from id.	Trevor Koehler	May 3, 2022	1	Controller/Games
Implement middleware and overall error handling	Nathan Batten	May 3, 2022	1	Controller/General
Create views for admins to add or edit genres.	Nathan Batten	May 3, 2022	1	UI/Genres
Create a search element to search for genres.	Nathan Batten	May 3, 2022	1	UI/Genres
Create a view to view an individual genre.	Nathan Batten	May 3, 2022	1	UI/Genres
Create views to view list of genres	Nathan Batten	May 3, 2022	1	UI/Genres
Create views for admins to add or edit games.	Trevor Koehler	May 3, 2022	1	UI/Games
Create a search element to search for games.	Trevor Koehler	May 3, 2022	1	UI/Games
Create a view to view an individual game	Trevor Koehler	May 3, 2022	1	UI/Games
Create views to view lists of games.	Trevor Koehler	May 3, 2022	1	UI/Games
Create a method for elegant error handling based off response for all endpoints.	Trevor Koehler	May 3, 2022	1	UI/General
Create consistent layout, headers, and footers for the UI.	Trevor Koehler	May 3, 2022	1	UI/General
Ensure only the correct user or admin can modify details from a game on that user.	Jacob Levy	May 10, 2022	2	Controller/Users
Ensure only the correct user or admin can remove games from the user	Jacob Levy	May 10, 2022	2	Controller/Users
Ensure only the correct user or admin can add games to the user.	Jacob Levy	May 10, 2022	2	Controller/Users
Ensure only admin accounts can delete genres	Jacob Levy	May 10, 2022	2	Controller/Genres
Ensure only admin acconuts can modify genres	Jacob Levy	May 10, 2022	2	Controller/Genres
Ensure only admin accounts can add new genres	Jacob Levy	May 10, 2022	2	Controller/Genres
Ensure game deleting endpoints can only be used by admin users.	Jacob Levy	May 10, 2022	2	Controller/Games
Ensure game adding endpoints can only be used by admin users	Jacob Levy	May 10, 2022	2	Controller/Games
Implement endpoints for a user to add hours and information to a game	Jacob Levy	May 10, 2022	2	Controller/Users
Implement endpoints for a user to remove games from themselves	Jacob Levy	May 10, 2022	2	Controller/Users
Implement endpoints for a user to add games to themselves	Jacob Levy	May 10, 2022	2	Controller/Users
Implement endpoints to get games from a user	Jacob Levy	May 10, 2022	2	Controller/Users

Implement endpoints to get user information, checking if they have permission	Jacob Levy	May 10, 2022	2	Controller/Users
Ensure game modifying endpoints can only be used by admin users.	Jacob Levy	May 10, 2022	2	Controller/Games
Supply justification/reasoning with recomendations.	Trevor Koehler	May 10, 2022	2	Model/Recommendations
Create a method to generate reccomendations based off user connections	Trevor Koehler	May 10, 2022	2	Model/Recommendations
Implement a way to get game reccomendations from a users details.	Jacob Levy	May 10, 2022	2	Model/Users
Implement methods to get games from a user account	Jacob Levy	May 10, 2022	2	Model/Users
Implement methods for adding stats in relation to the games on a user account.	Jacob Levy	May 10, 2022	2	Model/Users
Implement methods for adding a game to a user account	Jacob Levy	May 10, 2022	2	Model/Users
Implement methods for handling auth sessions.	Jacob Levy	May 10, 2022	2	Model/Users
Implement methods for password hashing	Jacob Levy	May 10, 2022	2	Model/Users
Implement ways to ensure only an admin can change things in the genres table.	Jacob Levy	May 10, 2022	2	Model/Genres
Implement a way to ensure only an admin can change things in the games tables.	Jacob Levy	May 10, 2022	2	Model/Games
Implement endpoints for users to get game recommendations.	Trevor Koehler	May 10, 2022	2	Controller/Users
Implement different endpoint responses if user is admin, or not logged in.	Jacob Levy	May 10, 2022	2	Controller/Auth
Implement methods for users to log in and keep their sessions.	Jacob Levy	May 10, 2022	2	Controller/Auth
Create a way to like/dislike recommendations	Trevor Koehler	May 10, 2022	2	UI/Users
Create a view to receive recommendations	Trevor Koehler	May 10, 2022	2	UI/Users
Create a view to edit hours, and personal info on a game.	Jacob Levy	May 10, 2022	2	UI/Users
Create a view to see all games from the user's profile.	Jacob Levy	May 10, 2022	2	UI/Users
Create a view to add/remove games from the users profile.	Jacob Levy	May 10, 2022	2	UI/Users

Create a user-specific element across all pages (i.e. username, avatar)	Jacob Levy	May 10, 2022	2	UI/Users
Change what's visible on a genres options based off auth.	Nathan Batten	May 10, 2022	2	UI/Genres
Change what's visible in games list and individual based off authentication.	Trevor Koehler	May 10, 2022	2	UI/Games
Handle being authenticated/unathenticated in header	Trevor Koehler	May 10, 2022	2	UI/Auth
Create view for account settings	Trevor Koehler	May 10, 2022	2	UI/Auth
Create elegant sign up and login pages	Trevor Koehler	May 10, 2022	2	UI/Auth
Send notification on relevant actions not erformed by the user to the user	Jacob Levy	May 20, 2022	3	Controller/Users
Notification view if something relevant happened to the user	Trevor Koehler	May 20, 2022	3	UI/Users
Implement a way to get a user's profile as any user if set to public.	Jacob Levy	May 20, 2022	3	Model/Users
Implement a way for users to get games from their game profiles and sync it	Jacob Levy	May 20, 2022	3	Model/Users
Implement ways for a user to pick view privacy	Jacob Levy	May 20, 2022	3	Model/Users
Implement methods for adding user description, pfp, etc	Jacob Levy	May 20, 2022	3	Model/Users
Implement an expandable way for extra info/tags on a game.	Trevor Koehler	May 20, 2022	3	Model/Games
Create a way to delete your own user account.	Jacob Levy	May 20, 2022	3	UI/Users

Coordination Plan Written Together

The following lists what websites and programs we plan to use during our website creation

- Code collaboration versioning: Git (specifically GitHub)
- Design information coordination: Google Drive, with discussions over Discord
- Deadline tracking and management: Jira
- Meeting schedule:Meetings will be in person or over Discord voice call. We will have a brief standup meeting to discuss progress and whatever is needed on
    - Monday at 7pm
    - Wednesday at 7pm
    - Friday after webdev (when not in class, around 2 or 3ish)

WRITTEN AGREEMENT

By placing my name here, it is considered a digital signature in agreement to work on the product listed and described in this document, following the plan and assigned tasks also described above.

Trevor Koehler.
Jacob Levy.
Nathan Batten.