Submission Worksheet

CLICK TO GRADE

https://learn.ethereallab.app/assignment/IT202-452-M2024/it202-api-project-milestone-3-2024-m24/grade/vs53

IT202-452-M2024 - [IT202] API Project Milestone 3 2024 m24

Submissions:

Submission Selection

1 Submission [active] 8/5/2024 8:23:47 PM

Instructions

COLLAPSE

Overview Video: https://youtu.be/-4hlb9MXrQE

1. Implement the Milestone 3 features from the project's proposal document: https://docs.google.com/document/d/1XE96a8DQ52Vp49XACBDTNCq0xYDt3kF29cO88EWVwfo/viev
2. Make sure you add your ucid/date as code comments where code changes are done
3. All code changes should reach the Milestone3 branch
4. Create a pull request from Milestone3 to dev and keep it open until you get the output PDF from this assignment.
5. Gather the evidence of feature completion based on the below tasks.
6. Once finished, get the output PDF and copy/move it to your repository folder on your local machine.
7. Run the necessary git add, commit, and push steps to move it to GitHub
8. Complete the pull request that was opened earlier
9. Create and merge a pull request from dev to prod
10. Upload the same output PDF to Canvas

Branch name: Milestone3

Tasks: 21 Points: 10.00

API (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Data Related to Users

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	What's the concept/association?
#2	1	What sort of relationship is it (one to many, many to one, many to many, etc)
#3	1	Note any other considerations

Response:

The concept of the data association is a list of favorite players for each user. Users can mark players as their favorites, creating a personal collection. This relationship is a many-to-many relationship since each user can have multiple favorite players, and each player can be favorited by multiple users. One consideration is how to handle the addition and removal of favorites, which needs to be intuitive and efficient for the user.

COLLAPSE

Task #2 - Points: 1

Text: Updating Entities

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	When an update occurs either manually or from the API how does it affect associated data?
#2	1	Do users see the old data, new data, does data need to be reassociated, etc?

Response:

When an update occurs, either manually or from the API, it affects the associated data by reflecting the changes in the user's list of favorites. Users see the updated data in real-time, as the system fetches the latest information whenever a user accesses their favorites or the player list. Data reassociation isn't necessary since the relationship is maintained through the player IDs and user IDs, which remain consistent even when other data attributes are updated.

COLLAPSE

Handle Data Association (1 pt.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the code

Details:

Option 1: Related pages will have a button to do association (like favorites or similar),
Option 2: a separate page will be used to associate entities to a user by some other user (like assignment of entities)

Include ucid/date comments for each code screenshot

#1) Show the related code

Caption (required) ✓

Describe/highlight what's being shown

3 Codes showing the 'Add to Favorites' button on the player list page.

Explanation (required) ✓

Explain in concise steps how this logically works and mention which option your application handles regarding association

PREVIEW RESPONSE

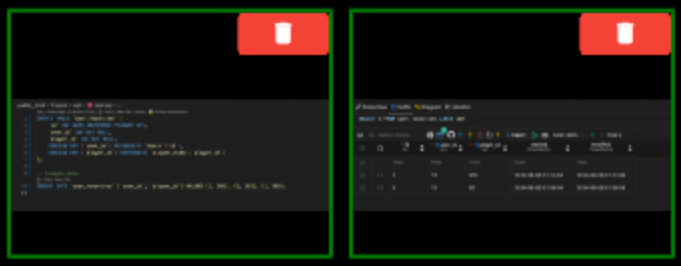
To handle the association, I used the first option, which involves adding a button to the player list page to allow users to add players to their favorites. When the user clicks the "Add to Favorites" button, the player ID is sent to the server via a form submission. The server then processes the request, checks if the player ID is valid, and inserts the association into the database. This creates a link between the user and the player in the user_favorites table. The user can then view their favorite players on a separate page. This setup allows users to manage their associations directly from the player list, making it convenient to add or remove favorites.

COLLAPSE

Task #2 - Points: 1

Text: Screenshot of the association table(s)

#1) Show the table(s) you made to handle the associations (Should have some example data)



Caption (required) ✓

Describe/highlight what's being shown

Showing two screenshots of the tables and example of it

Explanation (required) ✓

Describe each column/association table

PREVIEW RESPONSE

The table I created to handle the associations is called user_favorites. It has three columns: id, user_id, and player_id. The id column is a unique identifier for each entry. The user_id column stores the ID of the user who favorited the player, and the player_id column stores the ID of the favorited player. These columns together create a link between the user and the player, allowing the application to keep track of which players are favorited by which users. This table helps in managing user-specific data, like their favorite players, and supports the functionalities of adding and removing favorites.

COLLAPSE

Task #3 - Points: 1

Text: Add related links

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Include the heroku prod link for the page that creates the association
#2	1	Add the pull request link for the branch related to this feature Note: the link should end with /pull/#. Same pull request shouldn't be used for each feature

URL #1

https://vs53-it202-452-

URL

https://vs53-it202-452-prod-1c77a1c304b2.herol

prod-1c77a1c304b2.herokuapp.com/project/list_players.php

URL #2

https://github.com/VikShah/vs53-it202-452/pull/60

URL

https://github.com/VikShah/vs53-it202-452/pull,

+ ADD ANOTHER URL

Current User's Association Page (2 pts.)

^COLLAPSE^

^COLLAPSE^

Task #1 - Points: 1

Text: Screenshots of this page

Details:

Make sure the heroku dev url is visible in the address bar

Some screenshots may be repeated in subtasks, but ensure they highlight the specific subtask requirement.

#1) Show the summary of the results

Caption (required) ✓

Describe/highlight what's being shown

Showing summay of the results

#2) Show the single view buttons/links

Caption (required) ✓

Describe/highlight what's being shown

Showing the single view buttons and remove favorite buttons

#3) Show variations of the number

Caption (required) ✓

Describe/highlight what's being shown

Showing by searching by nba players

#4) Show variations of the filter/sort

Caption (required) ✓

Describe/highlight what's being shown

Showing what happens when you dont submit any player in the database

^COLLAPSE^

Task #2 - Points: 1

Text: Screenshot the code

Details:

Include ucid/date comments for each code screenshot

#1) Show the code related to fetching

Caption (required) ✓
Describe/highlight what's being shown
Showing the query used to fetch the user's favorite players.

Explanation (required)
✓
Explain in concise steps how this logically works and mention how you determine the result list (include the association logic and filters)
PREVIEW RESPONSE
This code snippet demonstrates the SQL query used to fetch the user's favorite players from the database. It joins the player_stats and user_favorites tables to retrieve the relevant data. The query includes filtering based on the player's first or last name and handles pagination and sorting. This process involves checking the association between the user and players to determine the list of favorite players.

#2) Show the code related to the display

Caption (required) ✓
Describe/highlight what's being shown
Showing the code responsible for displaying the fetched favorite players.

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
This part of the code iterates through the list of favorite players fetched from the database and displays them in a table format on the webpage. Each player's information, including their first name, last name, and position, is shown along with action buttons for viewing details and removing the player from favorites.

#3) Each record should have

Caption (required) ✓
Describe/highlight what's being shown
Showing the view details button/link for each favorite player.

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
The code includes a button/link for each player entry that directs the user to a detailed view page for that specific player. This allows users to click and see more in-depth information about the player from their list of favorites

#4) Each record should have

Caption (required) ✓
Describe/highlight what's being shown
Showing the remove from favorites button/link for each favorite player.

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
This code snippet includes a button next to each player's entry that allows the user to remove the player from their favorites. This deletion only affects the relationship between the user and the player, not the player entity itself in the database.

#5) Show the logic for deleting all

#6) Show the logic related to the count

#7) Show the logic related to the count

#8) Show the logic related to filter/sort

Caption (required) ✓
Describe/highlight what's being shown
Showing the clear all favorites button and its associated logic.

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
The code includes a button to clear all favorite associations for the user. When clicked, it triggers a function that deletes all entries in the user_favorites table for the logged-in user. This functionality is crucial for allowing users or admins to manage associations efficiently.

Caption (required) ✓
Describe/highlight what's being shown
Showing the code to count the total number of favorites associated with the user.

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
This code calculates the total number of favorite players associated with the user, regardless of the current filter or pagination settings. It provides an overall count of favorites, which is displayed in the page heading to give the user a sense of their total interactions.

Caption (required) ✓
Describe/highlight what's being shown
Showing the code to count and display the number of items currently shown based on the applied filter.

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
This part of the code calculates the number of players displayed on the page based on the current filter and pagination settings. It updates dynamically as the user applies different filters or navigates through pages, ensuring that the displayed count accurately reflects the current view.

Caption (required) ✓
Describe/highlight what's being shown
Showing the code to handle filtering, sorting, and limiting the number of displayed items.

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
The code includes logic to handle user inputs for filtering, sorting, and specifying the number of items to display per page. It constrains the limit to be between 1 and 100, defaulting to 10 if no valid input is provided. This allows for flexible yet controlled presentation of the data.

COLLAPSE

Task #3 - Points: 1

Text: Add related links

Checklist

*The checkboxes are for your own tracking

#	Points	Details
#1	1	Include the heroku prod link for the page that creates the association
#2	1	Add the pull request link for the branch related to this feature Note: the link should end with /pull/# Same pull request shouldn't be used for each feature

should end with /pull/#. Same pull request shouldn't be used for each feature

URL #1
https://vs53-it202-452-prod-1c77a1c304b2.herokuapp.com/Project/manage_favorites.php

URL
https://vs53-it202-452-prod-1c77a1c304b2.herol

URL #2
https://github.com/VikShah/vs53-it202-452/pull/61

URL
https://github.com/VikShah/vs53-it202-452/pull

+ ADD ANOTHER URL

All Users Association Page (likely an admin page) (2 pts.)

^COLLAPSE ^

^COLLAPSE ^

Task #1 - Points: 1

Text: Screenshots of this page

Details:
Make sure the heroku dev url is visible in the address bar

Some screenshots may be repeated in subtasks, but ensure they highlight the specific subtask requirement.

#1) Show the summary of the results

Caption (required) ✓
Describe/highlight what's being shown
Showing the All Users assocation page

#2) Show the single view buttons/links

Caption (required) ✓
Describe/highlight what's being shown
TWo screenshots of the profile being clickable and what you see when you click it

#3) Show the username related to the

Caption (required) ✓
Describe/highlight what's being shown
Clicking on the profile

#4) Show variations of the number

Caption (required) ✓
Describe/highlight what's being shown
Searching by the first name

#5) Show variations of the filter/sort

Caption (required) ✓
Describe/highlight what's being shown
Showing what happens when you search with something that is not valid

COLLAPSE

Task #2 - Points: 1

Text: Screenshot the code

Details:
Include ucid/date comments for each code screenshot

#1) Show the code related to fetching

Caption (required) ✓
Describe/highlight what's being shown
Screenshot of the fetching of all associations with the query

Explanation (required)
✓
Explain in concise steps how this logically works and mention how you determine the result list (include the association logic and filters)
PREVIEW RESPONSE
This code fetches

#2) Show the code related to the display

Caption (required) ✓
Describe/highlight what's being shown
Code to display the results

Explanation (required)
✓
Explain in concise steps how this logically works and mention the logic for handling the username requirements
PREVIEW RESPONSE
This code displays the fetched associations in a table format. Each row includes the player's first

#3) Each record should have

Caption (required) ✓
Describe/highlight what's being shown
Showing that each record have link for single view

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
This code snippet shows the action buttons for each player entry, including a "View" link to see the player's details and a form to

#4) Each record should have

Caption (required) ✓
Describe/highlight what's being shown
Showing each record has a username field that is clickable to go to

Explanation (required)
✓
Explain in concise steps how this logically works
PREVIEW RESPONSE
This code makes the username field clickable, redirecting the user to the profile page of the associated user. This allows admins to easily

associations between players and users with filtering, sorting, and pagination. It joins the player_stats, user_favorites, and Users tables. The results are grouped by player_id and username. The query uses placeholders for filtering, sorting, and pagination to determine the result list.

name, last name, username, total users associated, and action buttons for viewing details and removing the association. The table headers include sorting links, and each username is clickable to view the user's profile.

delete the association. The "View" link takes the user to the detailed view page for the specific player.

navigate to the user's profile for further details.

#5) Each record should have

Caption (required) ✓
Describe/highlight what's being shown
Showing each record has a button/link for delete

Explanation (required)
✓
Explain in concise steps how this logically works

PREVIEW RESPONSE

This form allows admins to delete the association between a user and a player. It includes hidden fields to pass the player_id and user_id to the remove_from_favorites.php script, which handles the deletion of the relationship.

#6) Show the logic related to the count

Caption (required) ✓
Describe/highlight what's being shown
Showing the logic related to the count

Explanation (required)
✓
Explain in concise steps how this logically works

PREVIEW RESPONSE

This code calculates the total number of associations, considering all records even if they are not shown in the current filtered results. It performs a count query and uses the result to determine pagination details, ensuring a comprehensive count of associations.

#7) Show the logic related to the count

Caption (required) ✓
Describe/highlight what's being shown
Showing that logic related to the count of the items on the page

Explanation (required)
✓
Explain in concise steps how this logically works

PREVIEW RESPONSE

This line of code displays the total number of associations on the page, dynamically updating based on the applied filters. The value in parentheses reflects the current count of associations, providing users with a clear indication of the total items.

#8) Show the logic related to filter/sort

Caption (required) ✓
Describe/highlight what's being shown
Showing code screenshot of filter/sort

Explanation (required)
✓
Explain in concise steps how this logically works

PREVIEW RESPONSE

This code handles the filtering, sorting, and pagination logic for displaying associations. It constrains the limit between 1 and 100, defaulting to 10 if no valid limit is provided. The query includes a partial match filter for usernames, allowing users to filter the results effectively.

COLLAPSE

Task #3 - Points: 1

Text: Add related links

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Include the heroku prod link for the page that creates the association
#2	1	Add the pull request link for the branch related to this feature Note: the link should end with /pull/#. Same pull request shouldn't be used for each feature

URL #1
https://vs53-it202-452-prod-1c77a1c304b2.herokuapp.com/Project/all_user_associations.php

URL
https://vs53-it202-452-prod-1c77a1c304b2.herol

URL #2
https://github.com/VikShah/vs53-it202-452/pull/62

URL
https://github.com/VikShah/vs53-it202-452/pull,

+ ADD ANOTHER URL

Unassociated Page (2 pts.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of this page

Details:
Make sure the heroku dev url is visible in the address bar

Some screenshots may be repeated in subtasks, but ensure they highlight the specific subtask requirement.

#1) Show the summary of the results

#2) Show the single view buttons/links

#3) Show variations of the number

#4) Show variations of the filter/sort

Caption (required) ✓
Describe/highlight what's being shown
Showing the unassoicated players, players with no favorited nba players

Caption (required) ✓
Describe/highlight what's being shown
Showing what happens when you press on the single view button links

Caption (required) ✓
Describe/highlight what's being shown
Showing searching by first name

Caption (required) ✓
Describe/highlight what's being shown
Showing when you search by nothing

COLLAPSE

Task #2 - Points: 1

Text: Screenshot the code

Details:
Include ucid/date comments for each code screenshot

#1) Show the code related to fetching

Caption (required) ✓
Describe/highlight what's being shown
Code Fetching all unassociated entities from the database

Explanation (required) ✓
Explain in concise steps how this logically works and mention how you determine the result list (include the unassociated logic and filters)

PREVIEW RESPONSE

This code fetches players who are not associated with any user. It uses a subquery

#2) Show the code related to the display

Caption (required) ✓
Describe/highlight what's being shown
Displaying the list of unassociated players

Explanation (required) ✓
Explain in concise steps how this logically works

PREVIEW RESPONSE

This code displays the unassociated players in a table format. Each row shows the player's first name, last name, position, and a link to view more details about the player. If there are no results, it shows a message indicating that

#3) Each record should have

Caption (required) ✓
Describe/highlight what's being shown
Displaying the view button for each player

Explanation (required) ✓
Explain in concise steps how this logically works

PREVIEW RESPONSE

This part of the code provides a "View" link for each player in the list. Clicking the link takes the user to a detailed view of the player, allowing them to see more information about the player.

#4) Show the logic related to the count

Caption (required) ✓
Describe/highlight what's being shown
Counting all unassociated items, including filtered results

Explanation (required) ✓
Explain in concise steps how this logically works

PREVIEW RESPONSE

This code calculates the total number of unassociated players, including those not shown in the current filtered results. It performs a count query and uses the result to determine pagination

to exclude players in the user_favorites table and filters the results based on the player's first or last name. The query is also sorted and paginated to manage the display of results.

no results are available.

details, providing a complete count of unassociated players.

#5) Show the logic related to the count

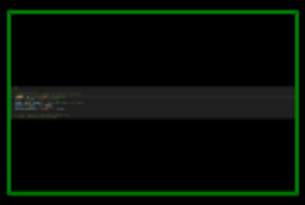
Caption (required) ✓
Describe/highlight what's being shown
Counting the items displayed on the page based on the filter applied

Explanation (required)
✓
Explain in concise steps how this logically works

PREVIEW RESPONSE

This line of code dynamically updates the count of unassociated players displayed on the page based on the applied filter. It shows the total number of unassociated players that match the current filter criteria.

#6) Show the logic related to filter/sort



Caption (required) ✓
Describe/highlight what's being shown
Applying filter and sort logic with constrained limits

Explanation (required)
✓
Explain in concise steps how this logically works

PREVIEW RESPONSE

This code handles the filtering, sorting, and pagination logic for displaying unassociated players. It constrains the limit between 1 and 100, defaulting to 10 if no valid limit is provided. The query includes a partial match filter for usernames, allowing users to filter the results effectively.

COLLAPSE

Task #3 - Points: 1

Text: Add related links

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Include the heroku prod link for the page that creates the association
#2	1	Add the pull request link for the branch related to this feature Note: the link should end with /pull/#. Same pull request shouldn't be used for each feature

URL #1
https://vs53-it202-452-prod-1c77a1c304b2.herokuapp.com/Project/unassociated_players.php

URL
https://vs53-it202-452-prod-1c77a1c304b2.herol

URL #2
https://github.com/VikShah/vs53-it202-452/pull/59

URL
https://github.com/VikShah/vs53-it202-452/pull,

+ ADD ANOTHER URL

Admin Association Management (like UserRoles) (1 pt.)

COLLAPSE

COLLAPSE

Task #1 - Points: 1

Text: Screenshots of the page

Details:
Make sure the heroku dev url is visible in the address bar

Some screenshots may be repeated in subtasks, but ensure they highlight the specific subtask requirement.

#1) Show the search form with valid

Caption (required) ✓
Describe/highlight what's being shown
Shownig the associate entities page, only for

#2) Show the results of the search

Caption (required) ✓
Describe/highlight what's being shown
Showing what happens for the results, all the

#3) Show the result of entities and

Caption (required) ✓
Describe/highlight what's being shown
Showing what happens when you associate

admins

users

them (make them favorite it)

COLLAPSE

Task #2 - Points: 1

Text: Screenshots of the code

Details:

Include ucid/date comments for each code screenshot

#1) Search form field for finding a

Caption (required) ✓

Describe/highlight what's being shown

Search form for finding partial matches of usernames

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

This code creates a search form field for finding partial matches of usernames. The user can enter a partial username, and the form will search for users whose usernames match the input.

#2) Search form field for finding a

Caption (required) ✓

Describe/highlight what's being shown

Search form for finding partial matches of player names

Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

This code creates a search form field for finding partial matches of player names. The user can enter a partial player name, and the form will search for players whose names match the input.

#3) Code related to getting a

Caption (required) ✓

Describe/highlight what's being shown

Limiting the search results to 25 users and 25 players

Explanation (required) ✓

Explain in concise steps how this logically works and describe the steps for the search and how it works for users and entities

PREVIEW RESPONSE

This code limits the search results to a maximum of 25 users and 25 players. It ensures that the query results are not overwhelming by restricting the number of matches returned for each search.

#4) Code that generates

Caption (required) ✓

Describe/highlight what's being shown

Generating checkboxes next to each user and player

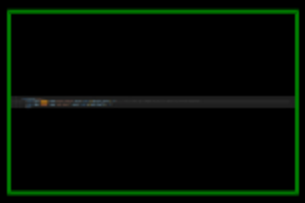
Explanation (required) ✓

Explain in concise steps how this logically works

PREVIEW RESPONSE

This code generates checkboxes next to each user and player in the search results. The checkboxes allow the admin to select multiple users and players for association.

#5) Code related to submitting



Caption (required) ✓
Describe/highlight what's being shown
Form submission for associating users and players

Explanation (required)
✓
Explain in concise steps how this logically works

PREVIEW RESPONSE

This code handles the form submission for associating users and players. It includes hidden fields to retain the search terms and a submit button to process the associations.

#6) Code related to applying the

Caption (required) ✓
Describe/highlight what's being shown
Applying associations between users and players upon form submission

Explanation (required)
✓
Explain in concise steps how this logically works and describe the steps for the associate/unassociate logic for the combination of users and entities

PREVIEW RESPONSE

This code processes the associations upon form submission. It iterates through the selected players and users, updating the user_favorites table to reflect the associations. If the relationship already exists, it doesn't create a duplicate. If it doesn't exist, it adds the new relationship. The process concludes with a success message and redirects back to the association page.

COLLAPSE

Task #3 - Points: 1

Text: Add related links

Checklist *The checkboxes are for your own tracking

#	Points	Details
#1	1	Include the heroku prod link for the page that creates the association
#2	1	Add the pull request link for the branch related to this feature Note: the link should end with /pull/#. Same pull request shouldn't be used for each feature

URL #1
https://vs53-it202-452-prod-1c77a1c304b2.herokuapp.com/project/associate_entities.php

URL
https://vs53-it202-452-prod-1c77a1c304b2.herol

URL #2
https://github.com/VikShah/vs53-it202-452/pull/59

URL
https://github.com/VikShah/vs53-it202-452/pull,

+ ADD ANOTHER URL

COLLAPSE

Misc (1 pt.)

COLLAPSE

Task #1 - Points: 1

Text: Screenshot of your project board from GitHub (tasks should be in the proper column)

Task Screenshots:

Gallery Style: Large View

Small Medium Large

Showing all the milestones on github projects

COLLAPSE

Task #2 - Points: 1

Text: Provide a direct link to the project board on GitHub

URL #1

https://github.com/users/VikShah/projects/1/views/1

URL

https://github.com/users/VikShah/projects/1/vi

+ ADD ANOTHER URL

COLLAPSE

Task #3 - Points: 1

Text: Talk about any issues or learnings during this assignment

Response:

During this assignment, I encountered several challenges and learned a lot. One major issue was dealing with the complexities of user associations and managing relationships between users and entities like favorite players. Understanding how to handle many-to-many relationships in a database and reflecting those associations in the application was tricky. I also had to learn how to properly filter, sort, and paginate the data, which required a good grasp of SQL queries and PHP. Debugging was another significant part of the process, especially when I faced errors related to database constraints and PHP warnings. Despite these challenges, I improved my problem-solving skills and gained a deeper understanding of web development, particularly in handling dynamic data and user interactions. This assignment was a valuable experience in learning how to create and manage complex functionalities in a web application.

COLLAPSE

Task #4 - Points: 1

Text: WakaTime Screenshot

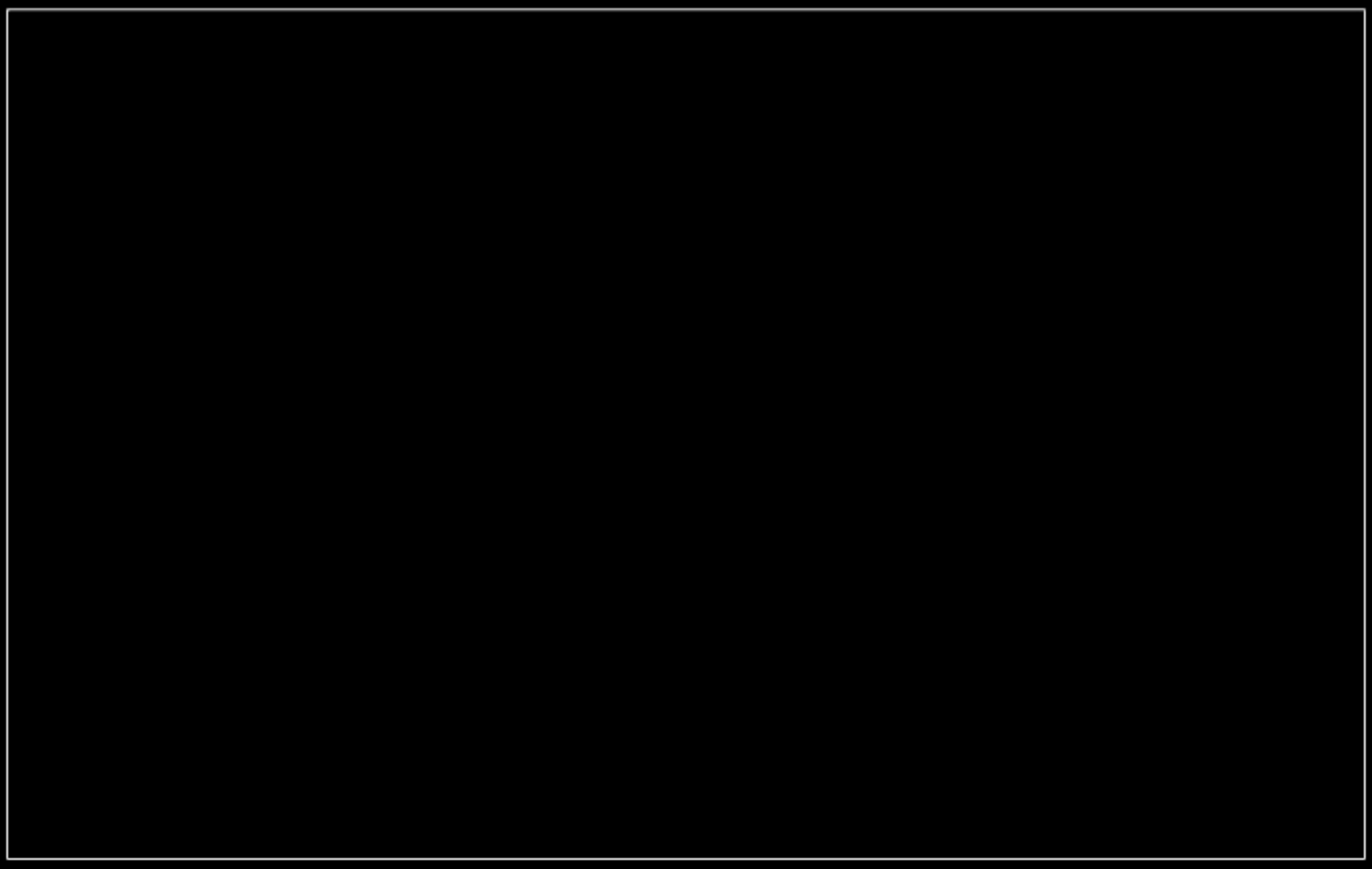
Details:

Grab a snippet showing the approximate time involved that clearly shows your repository. The duration isn't considered for grading, but there should be some time involved

Task Screenshots:

Gallery Style: Large View

Small Medium Large



First picture of wakatime

More detailed view of wakatime

End of Assignment