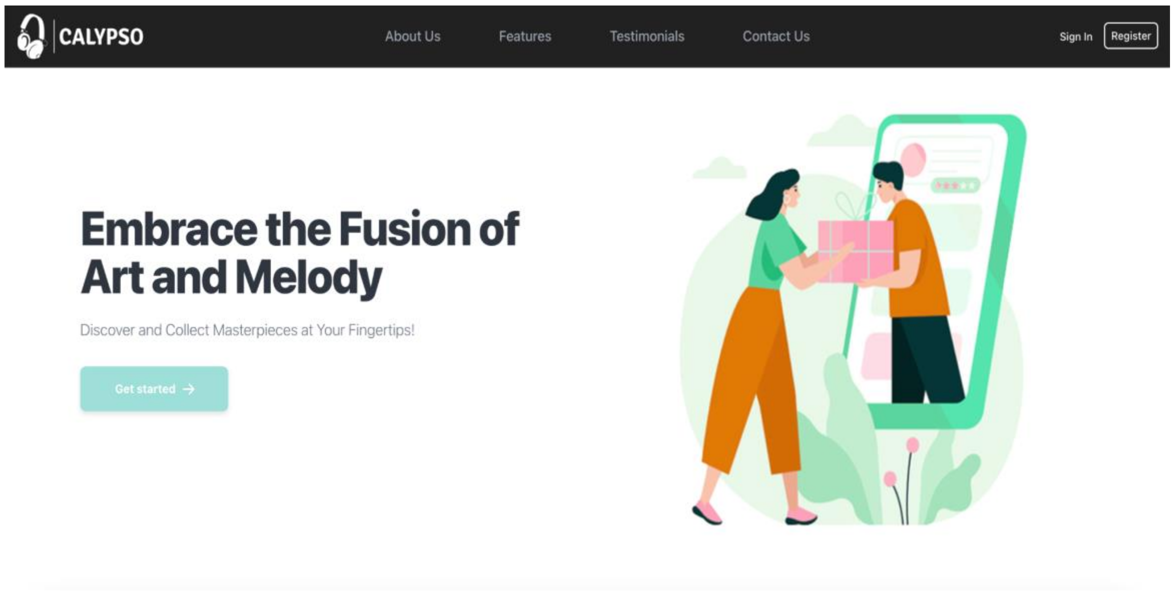
Calypso Project Presentation (Planning Doc) - August 9, 2023

Demo

- Problem:
  - We realized there were many artists, those that paint, draw, make music, that struggle to gain recognition and to reach a wide audience
  - We also noticed on the other hand, there were art buyers who searched through seas of options before finding art that they like, and would have to pay excessive amts for unique art
- Solution:
  - Built Calypso, which is a platform dedicated to encouraging artists of all levels to showcase their creativity and get recognition
  - While also proving a place for art lovers to explore new art and artist
- Target audience:
  - That being said, calypso targets artists and art enthusiasts with two main features.
- Our messaging system facilitates communication between users.
  - One on one chats can be created and used so that buyers can communicate with sellers to request art pieces to be made specifically for them or can also request for modifications to existing art pieces.
  - One on one chats can be used for art buyers to get to know artists and find out more about the creator of their favourite art pieces.
  - Users can also build a community and simply share their love of art by creating group chats to communicate with other artists or art lovers and share their passions.

- Our Recommendation feature
  - Our recommendation feature was designed so that users can find other products they may like based on their purchases. This way they don't have to browse through all kinds of products trying to find something else they may like.


CALYPSO
About Us
Features
Testimonials
Contact Us
Sign In
Register
Embrace the Fusion of Art and Melody
Discover and Collect Masterpieces at Your Fingertips!
Get started →

Process Discussion

- **Workflow**
    - Used the scrum format.
        - Had a sprint planning meeting for the beginning of each sprint
        - Had a couple standup meetings (almost daily) to update everyone about what was done
        - Had a sprint review meeting at the end of every sprint
        - Also incorporated pair programming, where two people worked on one feature or at least worked closely to help each other out.
    - Used JIRA to organize tasks and attempted to finish each task before each sprint.
- **Communication**:
    - Used mainly discord for calls, standup meetings, reviews, and cooperative programming.
    - We also used other messaging apps to remind team members of meetings and what not
- **Github**:
    - Branching:
        - We would create branches corresponding to each task on JIRA
        - Named the branch and commit messages as: CAL followed by the task number.
        - Once acceptance criteria were met, the individual pushes up their branch and any changes.
    - PR's and REVIEWS:

        - At least and normally one person would be requested to review new code and functionality.
        - Normally this person is the person who an individual worked closely with.
        - Once the review is completed, changes are made if the reviewer asks, and afterwards the branch is merged into main.
- **Task Division**
    - During the sprint planning meeting, we would first decide what user stories to include, what tasks to include for each story, and had people speak out about what tasks they would like to take on for each sprint.
    - If two or more people wanted the same task/story, we would pick randomly.
    - In our case, this never happened and everyone got their preferred user story and tasks each sprint
    - After this, we assigned story points and if user stories were a bit too much for one person, we adjusted and distributed the number of tasks to go to other team members as well to even the workload.

- **Our Good & Bad Decisions**
    - Some of the good decisions that we made were:
        - Working on individual features at the start, which allowed for us to get a lot of features done early on. Each person was assigned a separate feature and we just kept building on that.
        - Another good decision we made was reducing the amount of sprint meetings we had. This allowed for more meaningful contributions during meetings
        - During sprint 4 we paired up to work on some features and that allowed for the harder features to have more people working on them. The team members that already finished their features would lend their help to work on these harder features.
    - A Bad Decision that we made was
        - Not breaking down some tasks into subtasks at the start. That resulted in us overestimating how much work we could get done during the given time. This in turn resulted in user stories being left in the backlog

Technical Discussion
- Interesting bugs:
    - Proxy
        - Used to make api requests to the backend.
        - Does not work on UofT wifi

        - Required for cookies
- Technologies:
    - Cookies
        - Used to store the JWT token of the logged in user. This allowed different pages in our project to access the current user. This was especially important as a lot of our functions such as recommendation, messages, transactions, etc. require the currently logged in user.
        - Aside from cookies, we could have also used localstorage to store the currently logged in user. However, we decided to use cookies as they are more secure for storing authorization tokens.

Software Architecture

- Our application was built with a React frontend with Tailwind styling and TypeScript
- React used for real time updating of elements, Tailwind for easily styling components, and TypeScript was used for having structure and more strict types for JavaScript
- The backend was made using Express and NodeJS
- Our databases are MongoDB and Neo4j
- MongoDB for messages and user credentials
- Neo4j for storing Artworks, Musics, Users, and Bids
- We make use of graph relationships which show which users own which artworks and which users purchased which products
- The flow of our app is the user interacts with the React frontend, and if an certain action is performed such as purchasing a product, a request is sent to the backend using an API request through Express
- NodeJS will run queries to the Database if needed
- The challenge we faced was decided whether to use MongoDB, Neo4j or both
- There was the problem of keeping items in sync but we were able to avoid this by storing everything in Neo4j and only using MongoDB for separate info like user credentials (this was easily synced by storing the username in both DBs)
- The reason why we chose to use both is because the ability to use graph relationships in Neo4j was important to not have since it is vital for our product recommendation service and for showing ownership of products

Individual Contribution
William:

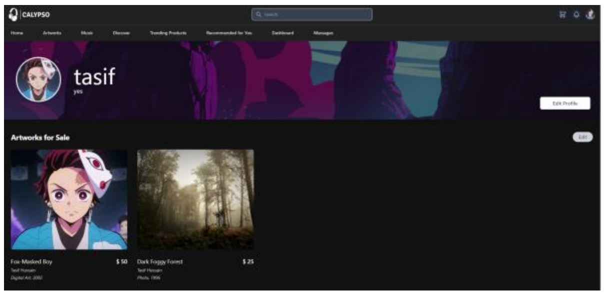
- Scrum master
- Home page
- Artworks page
- Seller Dashboard
- Was going to learn React and NodeJs on my own time, but working on this project helped me learn React, NodeJs, Express, Neo4j, and MongoDB all at the same time
- Also learned a lot more about how frontend/backend interact with each other through APIs

Arielle:

- Submissions person
- Landing page
- Registration page
- Login
- UI updates - making pages more consistent
- Learnt a lot of new information on front end, tailwind, amination of components as well as
- Strengthened my backend skills as i previously only worked with firebase which i realized was not highly recognized by hiring managers,
- I got to work with MonogoDB for the first time and a little of Neo4j - both nosql and graph db experience

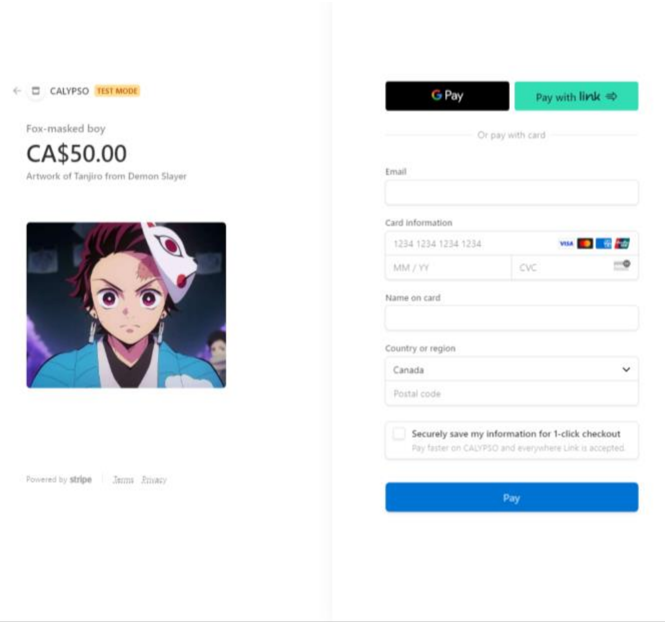
Tasif:

- Worked on Profile Page
    - Users can customize their profiles such as username and images.
    - Users can publish new products or remove existing ones.
- Worked on the UI for message page
- Helped establish real-time communication between users by using sockets.
- Gained useful insights and experiences
    - Learned how to use React and TypeScript to make a front-end.
    - Learned how to implement and use APIs to communicate from front-end to back-end.
    - Learned MERN stack as well as Neo4j database



Shadman:

- Product Page
  - Learned React, Typescript, JavaScript and Tailwind while working on this feature
- Stripe API implementation
  - Learned about how to use API and integrate them by reading documentation and adjusting the code to work with our codebase
- Transactions
  - Worked on creating a secure transaction flow using Stripe Secure Checkout. This transaction flow supports credit card payment as well as Google Pay and other payment methods.



Austin:

- Bidding: Took care of the bidding page UI and backend, including placing bids, making a product a biddable product and not for purchase, and retrieving the highest bid in the database.

- React Library Setup: Set up initial react libraries like router.js and tailwind.js.
- Also helped out UI design for the product page, and worked closely with Shadman for the product page backend.
- Also took care of the shopping cart and recognizing how implement libraries and
- In summary, learned react, Typescript, Java script and tailwind. Also learned Cypher and how to use Neo4J. Finally, learned how to use axios and to link the backend to the UI, using javascript code, tsx files, and JS files to do so.

Cassy

- Worked on the create chat page which allows users to create one on one chats and group chats
- Worked on the Messaging page in which users can see all of the chats that they are part of and can communicate with other users in the various chats.
- Learned React, how to use APIs and how to use sockets.

Maaneth

- Worked on the recommendation engine that suggests new artworks and songs.
- Worked on the admin dashboard that shows the most clicked art and pages.
- Redesigned the UI for the music, discover and trending page.
- Learned React, typescript, Neo4j, recommendation engine algorithms, building an API.