Final Project Report
PhoneArt - MathArt iOS Application

Aditya Prakash, Noah Huang, Emmett De Bruin, Zachary Letcher
Paul Phillips
Dr. Saturino Garcia

1: Project Summary

The MathArt App is an iOS mobile application inspired by the MathArt Playground website (https://mathart.us/ng-playground/). This app seeks to simplify the process of creating visually stimulating images by enabling users to manipulate shapes, colors, and transformations through mathematical functions. Beyond merely providing tools for artistic expression, the app aims to fill an educational gap by making the intersection of art and math more accessible and engaging. By fostering creativity and learning, the MathArt App addresses the challenge of introducing mathematical principles in a visually intuitive and user-friendly manner.

2: Problem Statement

The MathArt App addresses the challenge of creating accessible and engaging tools that merge art and mathematics. Many existing creative tools have steep learning curves, making them less accessible to educators, students, and hobbyists. The app seeks to solve this by offering an intuitive platform that allows users to generate complex artistic designs with minimal effort. By simplifying interactions with math-based art creation, the app lowers barriers for those with limited technical or mathematical expertise.

The broader goal is to provide a platform for artistic expression and learning that benefits educators, students, and creative hobbyists alike. This project will have a significant impact by making math concepts more tangible and engaging, fostering creativity in educational settings, and providing a unique tool for exploring the interplay of math and art. Ultimately, the app serves as a bridge between two traditionally separate domains, offering practical and educational value.

3: Stakeholders

Client/customer stakeholders:

Paul Phillips
Description: Creator of the MathArt Playground website and conceptual initiator of the MathArt App. While Paul is not directly involved in the app's technical development, his vision drives its goals.
Goals:

- Ensure the app aligns with the website's core functionality and vision.
- Expand the MathArt platform to mobile users.

Pain Points:

- Limited technical expertise in iOS development may challenge collaboration.
- Ensuring the mobile app maintains the quality and scope of the original web platform.

End user (user roles) stakeholders:

Hobbyists/Mathematicians
Description: Individuals with an interest in art and math who wish to explore mathematical transformations creatively.
Goals:
- Create visually appealing artwork using mathematical transformations.

Pain Points:
- Limited understanding of technical jargon or tools might hinder usage.
- Ensuring real-time responsiveness of sliders and transformations.

STEAM Academy Instructors
Description: Educators aiming to introduce math concepts through creative tools in their classrooms.
Goals:
- Demonstrate the app's features effectively to enhance student engagement.
- Use the app to connect abstract mathematical concepts with practical application.

Pain Points:
- Students may struggle with the app's features without adequate training materials.
- Difficulty incorporating the app into existing curricula seamlessly.

STEAM Academy Students
Description: Students learning math concepts such as geometric transformations, who will use the app to explore and apply these concepts creatively.
Goals:
- Develop a deeper understanding of math concepts through creative exploration.
- Enjoy the process of creating art while learning.

Pain Points:
- Initial learning curve with the app interface.
- Potential frustration with understanding complex mathematical transformations.

4: Requirements Overview

4.1: Functional Requirements

ID	User Story	Status
1	As a client, I want to customize artwork parameters with intuitive controls so that I can adjust elements like size, position, and rotation to create unique MathArt..	✅
2	As a client, I want access to various geometric shapes so I can create different types of MathArt compositions.	✅
3	As a Client, I want to save my artwork and be able to view or edit it later, so I can	✅

	continue working on it, refine my designs, or showcase my completed creations anytime.	
4	As a Client, I want my artwork codes to be safely stored and easily retrievable, so I can access them from different devices or share them with others, without requiring user accounts.	✅
5	As an instructor, I want students to be able to upload their artwork to a shared class folder so that I can easily access, organize, and review my students' work for assessments and class discussions.	❌
6	As an instructor, I want to be able to comment on students' artwork directly within the app so that I can provide constructive feedback and encourage learning.	❌
7	As a student, I want to be able to share my artwork in multiple ways (e.g., social media, print, email) so that I can show it to my friends and family.	✅
8	As a student, I want to access a tutorial page that explains the different art translation functions (like Skew, Zoom, etc.) so that I can better understand how to use these features independently.	✅
9	As a student, I want to easily remove artwork I no longer need and share my creations with others so I can manage my gallery efficiently and collaborate with peers.	✅
10	As a student, I want clear permissions controls when sharing my artwork, so I can protect my work and privacy.	✅

Table Key:
✅: Complete
❌: Incomplete

4.2: Non-Functional Requirements

ID	Type	Description	Status
1	Performance	Maintains 60 FPS during transformations	✅
2	Performance	Smooth real-time updates	✅
3	Performance	Efficient memory management	✅
4	Reliability	Robust error handling for invalid inputs	✅
5	Reliability	State preservation during view updates	✅
6	Reliability	Crash-free operation during testing	✅

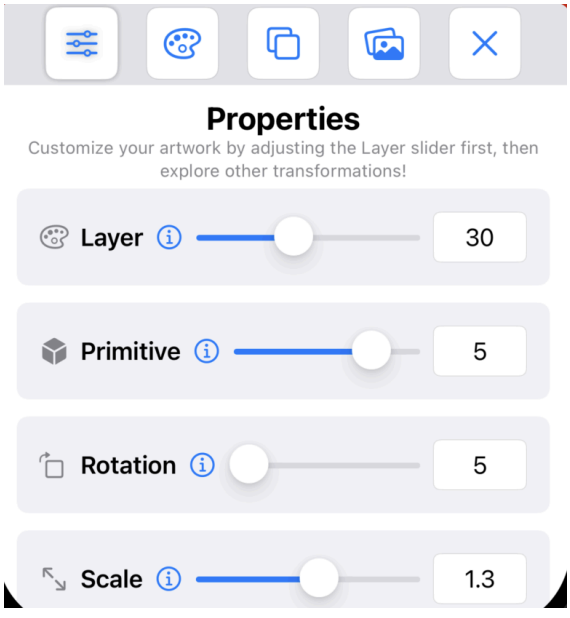
7	Usability	Intuitive dual-input controls	✅
8	Usability	Responsive touch interface	✅
9	Usability	Clear visual feedback	✅

5: Implementation Overview

1. As a client, I want to customize artwork parameters with intuitive controls so that I can adjust elements like size, position, and rotation to create unique MathArt.

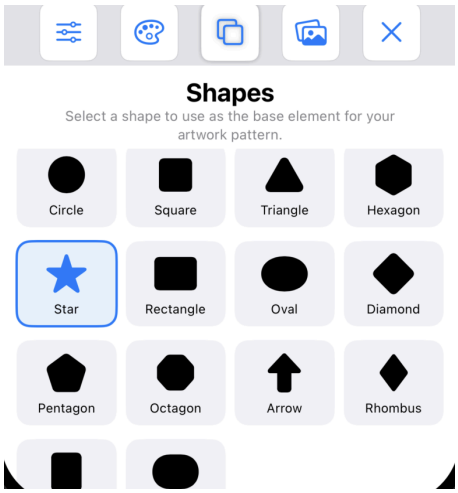
   Discussion: A user is able to edit their artwork canvas with nine sliders which affect every shape on the canvas and they are able to choose the colors for the artwork, canvas, and shape stroke. The sliders are
   - Layer - affects the number of shapes on the canvas
   - Primitive - affects the number of arrays of shapes
   - Rotation - determines how much each layer is rotated in relation to the previous layer
   - Scale - determines the size of the next shape bases on the last layer
   - Spread - determines the distance between two layers
   - SkewX - stretches each layer in the x direction
   - SkewY - stretches each layer in the y direction
   - Horizontal - shifts the artwork left / right from the origin point of the canvas
   - Vertical - shifts the artwork up / down from the origin point of the canvas

   Design:



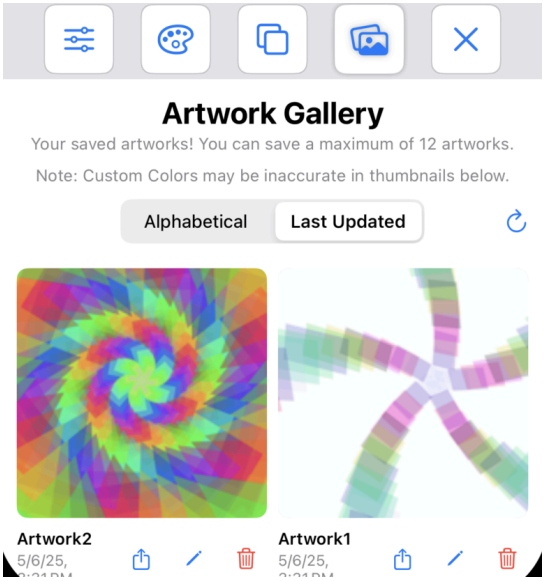
2. As a client, I want access to various geometric shapes so I can create different types of MathArt compositions.

   Discussion: We implemented 14 different geometric shapes which can be used on the artworks. We did this by either using the pre-existing shapes in SwiftUI (circle, square, triangle) or by hardcoding the canvas to draw each of the other unique shapes (star, arrow, trapezoid, etc.). The current limitation on shape selection is a user is only able to choose our 14 shapes. There is no custom shape importation like there is for MathArt.us , this is a feature left to the future PhoneArt developers.

   Design: 

3. As a Client, I want to save my artwork and be able to view or edit it later, so I can continue working on it, refine my designs, or showcase my completed creations anytime.

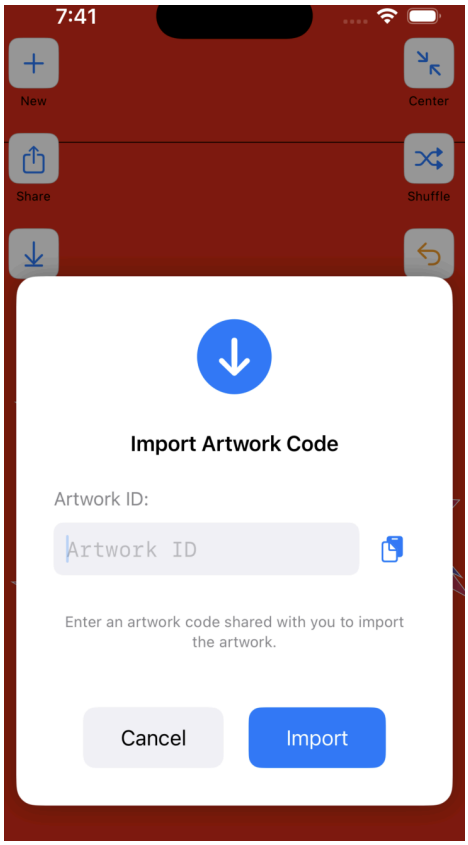
   Discussion: We implemented a gallery which can save up to 12 artworks at a time. A user can load each of these artworks by pressing on them from the GalleryPanel. Before an artwork is loaded, a pop-up will appear prompting the user to confirm if they want to load the artwork if they are on an unsaved canvas. Once a user is done editing the current artwork, they can either save over the previous file or save as a new artwork, if there is space in the gallery.

   Design: 

4. As a Client, I want my artwork codes to be safely stored and easily retrievable, so I can access them from different devices or share them with others, without requiring user accounts.

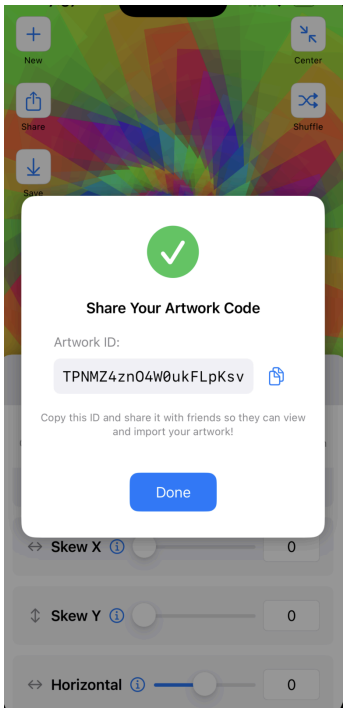
   Discussion: The artwork codes are stored in our database, which will save a user's gallery and populate the gallery upon rebooting the app. In order to load artworks, a user will need to paste their codes by selecting the 'New' button, then choosing the 'Import Artwork Code…' option. They will then be prompted to input the artwork code, which will then load the artwork onto the canvas.

   Design:



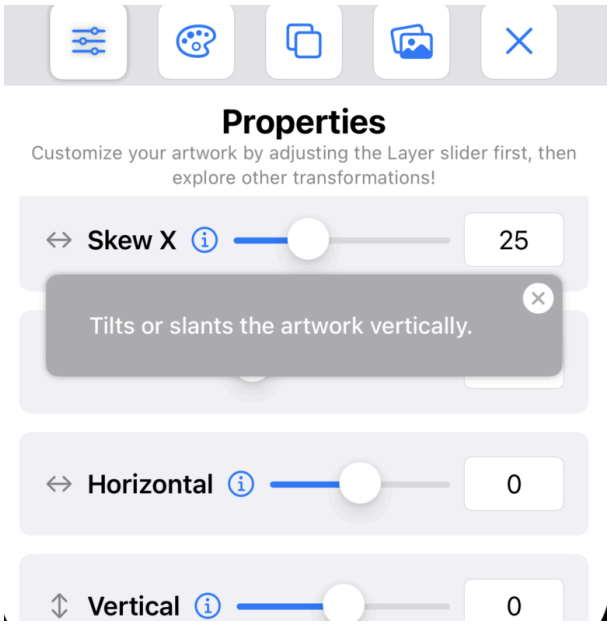
5. As a student, I want to be able to share my artwork in multiple ways (e.g., social media, print, email) so that I can show it to my friends and family.

   Discussion: In order to share the artwork codes, the user will need to select the 'Share' button on the top left, then a window pops up with an option to copy the artwork code to their device's clipboard. Once copied, a user can paste the artwork code in whichever text editor they desire. (notes app, iMessage, email, social media, etc.) The app does not directly share to the apps by design as we do not want access to any of the user's device / info, other than their photo gallery for saving artworks.

Design: 

6. As a student, I want to access a tutorial page that explains the different art translation functions (like Skew, Zoom, etc.) so that I can better understand how to use these features independently.

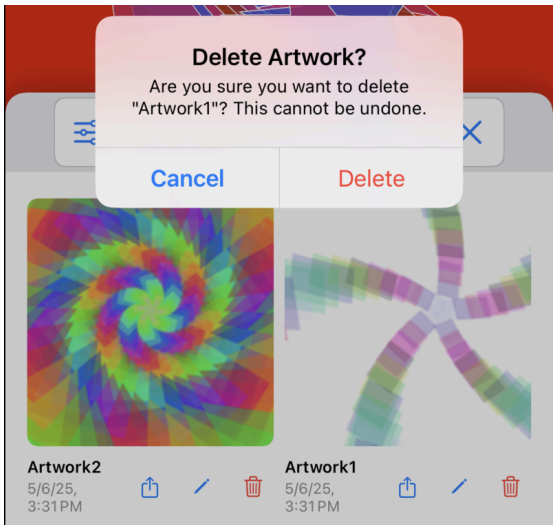
   Discussion: Next to the name of each slider, there is an i icon which will display a textbox which will explain in short detail. The textbox will have an x on the top right to close it but a user can also tap anywhere else outside of the textbox to close it. The descriptions are not meant to be fully detailed explanations but rather short reminders for the user. This is meant to encourage the students to inquire about these transformations with the instructor and lead to a lesson in mathematics / geometry.

Design: 

7. As a student, I want to easily remove artwork I no longer need and share my creations with others so I can manage my gallery efficiently and collaborate with peers.

   Discussion: On the gallery panel, beneath each artwork, are two buttons. One is a blue pencil and the other is a red trash bin. If the user wishes to remove an artwork from their gallery, they will press the red trash bin button. Once pressed, a pop up will appear asking a user to confirm they want to delete the selected artwork. Once they confirm, the artwork is removed from theri gallery.

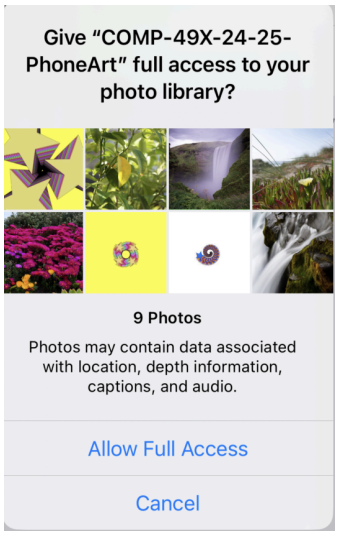
   Design:



8. As a student, I want clear permissions controls when sharing my artwork, so I can protect my work and privacy.

   Discussion: Upon first selecting the share button, a user will be prompted to give the app consent to access their camera roll. This is the only time the app will ask the user for any permissions or information. All other information of the user saved on our database is a randomly generated UserID, which is only linked to the device they are currently using.

   Design:



6: Ethical Usage

Ethical Considerations

For our MathArt App, our primary ethical concern revolves around inclusivity and equal access for all users, regardless of disabilities or unique challenges. We acknowledge that each user, whether considered an “edge case” or not, deserves full engagement and enjoyment from our app. To meet this commitment, we will implement user-centered design practices, such as engaging diverse user groups during regular testing sessions and incorporating feedback loops to continually refine the app's features. By prioritizing inclusivity in every stage of development, we ensure that our app’s design embraces the entire spectrum of users.

Consent and data use are equally important in our ethical framework. While our app does not heavily rely on personal data collection, any data collected will be handled transparently, with clear consent processes in place. Users will be informed when and how their data is collected, including an option to opt-out at any time. We will also provide a simple method for users to request data deletion. For instance, if a user’s artwork is featured in promotional materials, explicit consent will be obtained beforehand, and they will retain the right to revoke permission. These measures emphasize our commitment to respecting user privacy and using data solely to enhance their experience.

Accessibility Considerations

Our MathArt app prioritizes accessibility in several key ways, ensuring adaptability for users with varying abilities. One primary focus is visual accessibility, particularly for users who are colorblind or have other visual impairments. To address this, we will integrate features such as customizable color contrast options, alternative color schemes, and compatibility with Apple's built-in accessibility settings, like VoiceOver and Dynamic Type. Additionally, we will use algorithms to detect and suggest optimal color modes for users based on their preferences.

For users with cognitive impairments, the app will feature a simplified and intuitive interface with large buttons, visual cues, and customizable display options to reduce cognitive load. The onboarding process will include step-by-step tutorials designed to assist first-time users, ensuring they feel confident navigating the app.

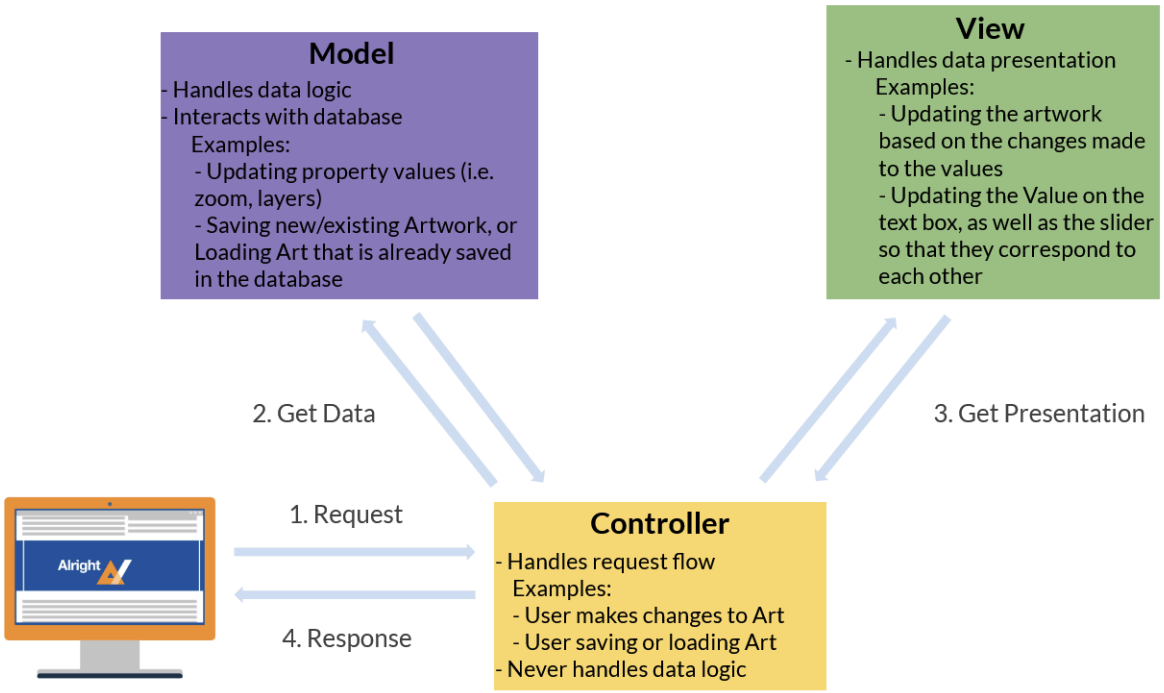
To support users with limited vision, we will incorporate text-to-speech functionalities and enlargeable interface elements. Comprehensive user guides will be accessible in multiple formats—text, audio, and video—with walkthroughs to explain app features and functionality. These guides will be regularly updated to ensure alignment with any new app updates, allowing all users to fully utilize the app's potential. By integrating these considerations, MathArt will serve as a universally accessible tool, fostering an inclusive environment for users to explore and enjoy the intersections of math and art.

7: System Design

7.1: Design Overview

7.1.1: Software Architecture

Our group has chosen to adopt a MVC architecture as its independent component structure – Model, View, and Controller – is a good fit for developing our mobile application. This separation will allow us to better maintain and test each aspect of our application, which also promotes scalability as the components work independently from each other. The MVC model also facilitates code reuse as we may have multiple viewing formats for the same model. A good example of this is how the user may visualize the properties for the artwork in the application. The same numerical data will be displayed in the form of a slider, as well as a box, that may be manually updated via the on screen keyboard. This change would then be reflected on the slider accordingly



Here is a brief example of how our application will interact with the different components of the MVC:
Controller - The End User changes the values of the art properties, for this example let's say that the user changes the number of layers that the art has.
The Controller will then fetch this request to Model, which will handle the data logic, as well as interact with the Database, when needed. Then it will
Model - The model will take the request and update the values using the mathematical logic that the backend runs under. Once all of the backend values have been updated by the model, it will then send the greenlight to the Controller to now communicate with the View to visually represent the changes made to the data. This performs the updates to reflect user input, such as calculations for artwork rotations, layers and other sliders.
View - The View will now gather the updated values, and update the UI to represent the changes requested by

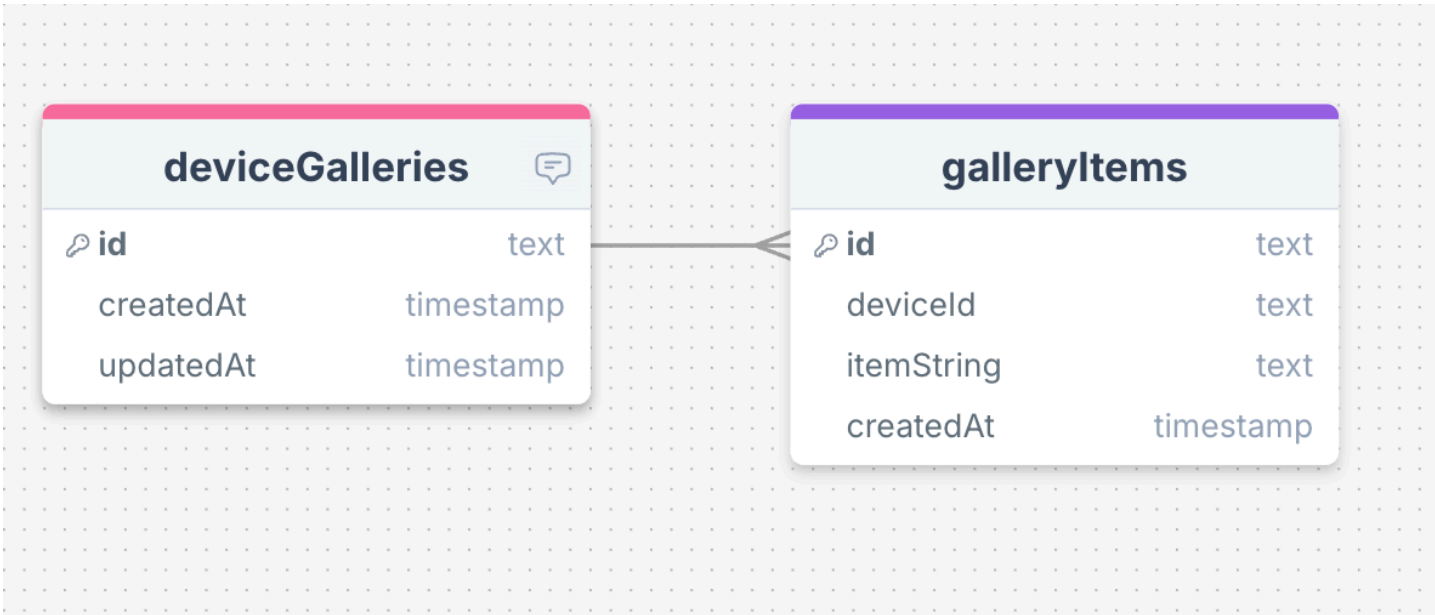
the user. The View will then fetch the presentation to the Controller. Finally the Controller will respond to the End User's request with the updated artwork in real-time, ensuring a responsive and an engaging experience.

7.1.2: Frameworks and Software Dependencies

Name	Version	Purpose / Justification
Framework: SwiftUI	5.10	● Declarative syntax allows for streamlined code and enhances usability and readability for newer teams ● Easier state management, creating more efficient prototyping and testing ● Integrates simply with swift, reducing the boilerplate and includes faster UI updates with real time code to display updates ● an easier learning curve for beginner users, and although it lacks the more advanced features, it still includes all features we would need for our project. The other con being "limited compatibility" is true if any user has an IOS version between 9-13, but as IOS is now onto IOS 18, we won't see anything prior to IOS 13 often.
API: Firebase	Blaze Plan	● Real-time database gives instant updates and cross-device synchronization ● Very scalable, including user authentication, giving a more enhanced security and user experience ● Gives availability with a free tier, suitable for smaller-scale app development that has a potential for growth ● We want to give users the option to save and bring back their artwork from a gallery. This can be fixed with using Firebase as an API, this would be used to authenticate users and give storage of their data and artwork. We prefer Firebase over iCloud for the real-time updates available and Firebase over SQLite for the cross-device compatibility for users with iPhones and iPad.

7.2: System Models

7.2.1: Class Diagrams



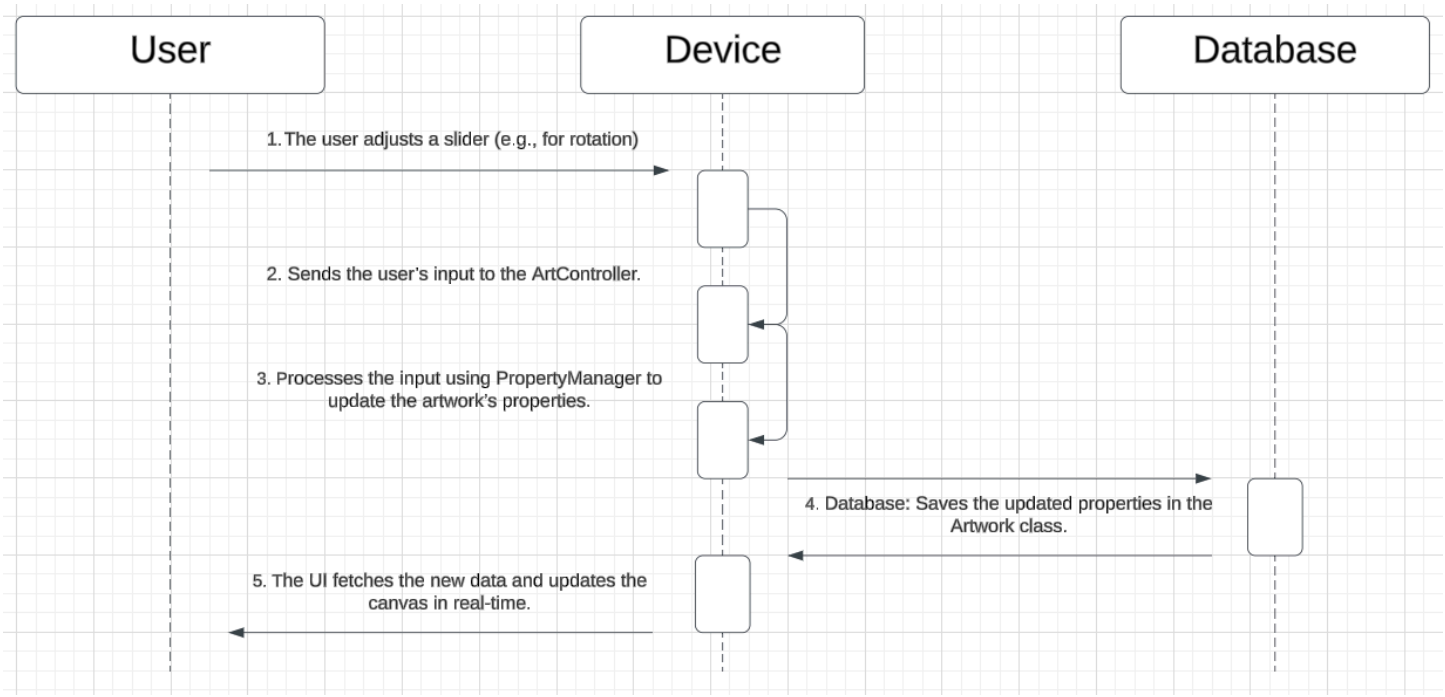
Note: This diagram includes all of the necessary attributes. Please reference the applicable methods below

The Class Diagram outlines the key classes in the MathArt App and their relationships:

1. DeviceGallery Class
    - **Attributes:** id (text), createdAt (timestamp), updatedAt (timestamp)
    - **Methods:** createGallery(), updateGallery()

2. GalleryItem Class
    - **Attributes:** id (text), deviceId (text), itemString (text), createdAt (timestamp)
    - **Relationships:** Many-to-One with DeviceGallery (through deviceId)
    - **Methods:** addItem(), removeItem(), getItems()

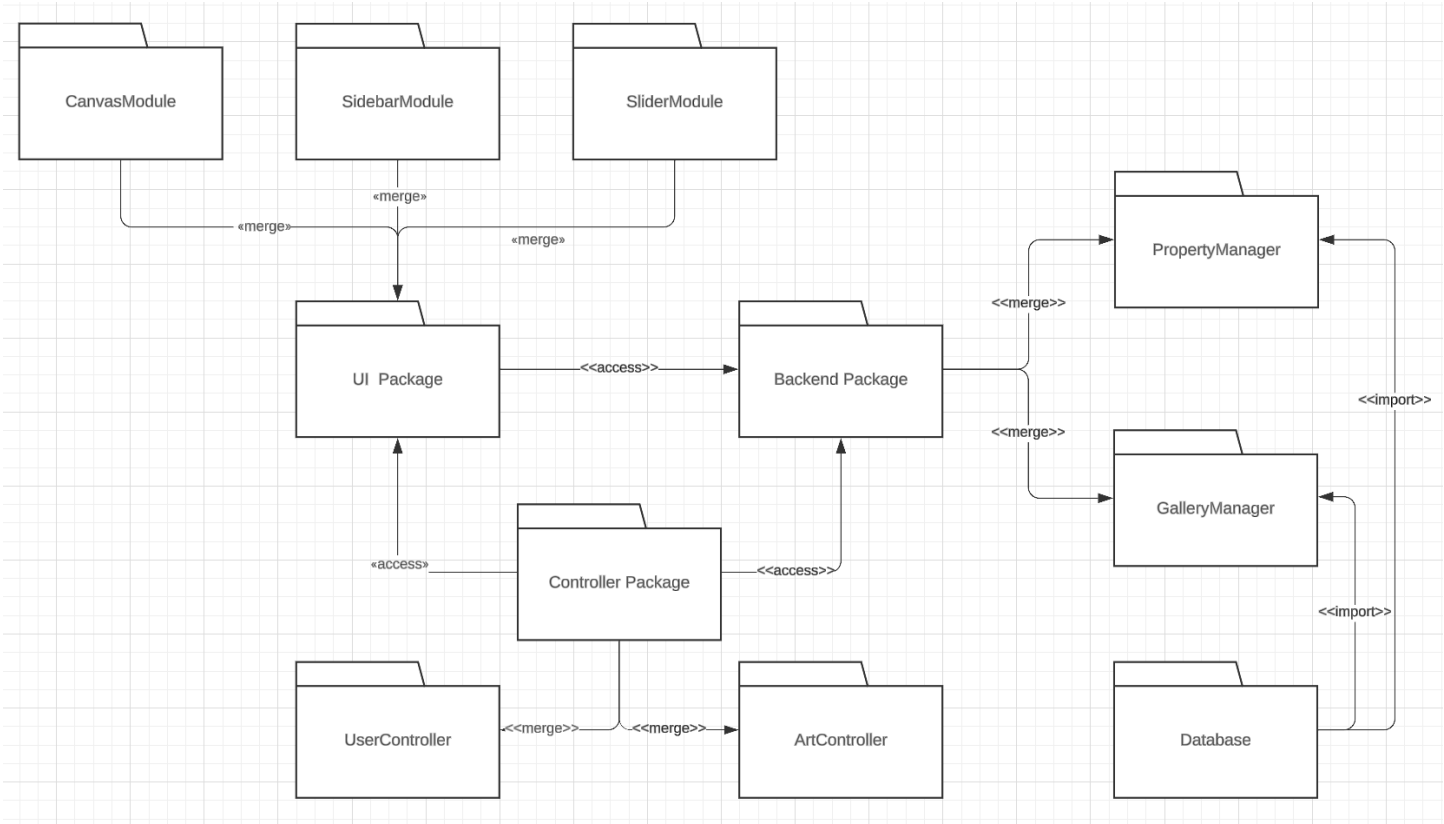
7.2.2: Sequence Diagrams



The **Sequence Diagram** demonstrates the interaction flow when a user modifies artwork:

1. **User Action:** The user adjusts a slider (e.g., for rotation).
2. **Controller:** Sends the user's input to the `ArtController`.
3. **Model:** Processes the input using `PropertyManager` to update the artwork's properties.
4. **Database:** Saves the updated properties in the `Artwork` class.
5. **View Update:** The `UI` fetches the new data and updates the canvas in real-time.

7.2.3: Package Diagrams



*Files have been renamed and properly formatted Package Diagram will be worked on
The Package Diagram shows how the app's components are organized:

1. UI Package
    - Includes components like `CanvasView`, previously CanvasModule, `PropertiesPanel`, previously SidebarModule and SliderModule.
2. Backend Package
    - Manages the database and logic for artwork generation and storage.
    - Example: GalleryManager, PropertyManager.

7.4: Data Architecture

7.4.1: Data Sources

External / Pre-existing Data Sources

Data Source 1: mathart.us
Description: The mathart.us page includes the web-version of PhoneArt, which we will be referencing as we begin to design and implement the project. The page also includes other projects that our client has worked on, which may be referenced for functional inspiration . Finally, mathart.us will be used to test our implementation

to ensure that the mathematics being coded is adjusting the artwork as intended (i.e. rotating a shape by 90 degrees clockwise).
Type: Structured. All the data is well presented and set up for use.

8: Future Work

These features are the ‘nice to have’ features which we were not able to work on during this semester.

- ☐ Shape importation
  - ☐ Separate drawing app (?)
  - ☐ Converting .png to drawable swift shape or SVG

- ☐ Features for instructors / classroom
  - ☐ Classroom folder
  - ☐ Comments on artworks

Shape Importation:

An unimplemented feature from MathArt.us, this feature would allow a user to import a .svg (Scalable Vector) file to the website and use it on the canvas. The software would treat this .svg file as any other shape and would color, scale, rotate, and skew each layer individually. This feature was desired so a user can use more uncommon shapes from different cultures. (ex. Native American and African shapes)

Since PhoneArt uses SwiftUI to draw the shapes rather than using .svg files, either the implementation of drawing shapes would need to be reworked or find a way to have Swift trace / draw out these custom shapes. One potential solution would be to have a separate shape drawing app which will connect to PhoneArt.

Instructor / classroom features:

For the instructors, we wanted to have a shared classroom folder so students could upload their artworks for grading and/or commenting. This feature was proposed but not required and would need to be focus tested by the instructors who use PhoneArt to see if they would like this feature or if they would like another way to interact with the student’s artwork.