

**Figure 7-1 – Possible System Entities**

Product: ART

Team: 63

Date: 02/25/2019

Type	Name	Description or Notes
Code	Request handler	Handle requests to the API
Code	Database Interface	Interface for programmers to interact with the database
Code	Authentication Flow	Login into the application
Code	API Interface	Logic for the client to interact with the API.
Code	Client View	Logic for manipulating the client view
Screen	Upload page	Allows users to upload new pieces of art and returns a QR code for distribution
Screen	Browse page	Displays all publically-available artworks and their corresponding QR codes
Screen	iOS scanner	The default view of the iOS app- displays a live view of the camera, and waits for a QR code
Screen	iOS viewer	Renders a requested artwork in 3D space using ARKit. A swipe up menu gives access to comments and report areas

For 4 entities:

### Figure 7-2 – Template for Detailed Design for a Screen

Name: Upload page

Type: Screen

Purpose: This screen is needed to meet requirement 3.1 *Will have Image uploading interface*

Description: Figure 1 shows the layout for this screen. This screen allows users to upload new artworks and returns a QR code for distribution.

The screen contains the following elements: Logo, Nav bar, upload form, upload requirements text

Layout:

The wireframe shows a rectangular screen layout. At the top left is a box labeled 'ARt'. At the top right is a navigation bar containing the text 'Browse', 'Upload' (which is bolded), and 'My Account'. In the center of the screen is a light blue rectangular button labeled 'Upload File'. At the bottom of the screen is a line of text: 'Requirements: Must be less than X megabytes, less than X by Y in size, etc...'.

Figure 1 - Upload Screen

Name: Browse Page

Type: Screen

Purpose: This screen is needed to meet requirement 3.12 *The system will allow users to browse art uploaded to the website.*

Description: Figure 2 shows the layout for this screen. This screen displays artworks sorted by newest uploaded.

The screen contains the following elements: Logo, Nav bar, art thumbnails

Layout:

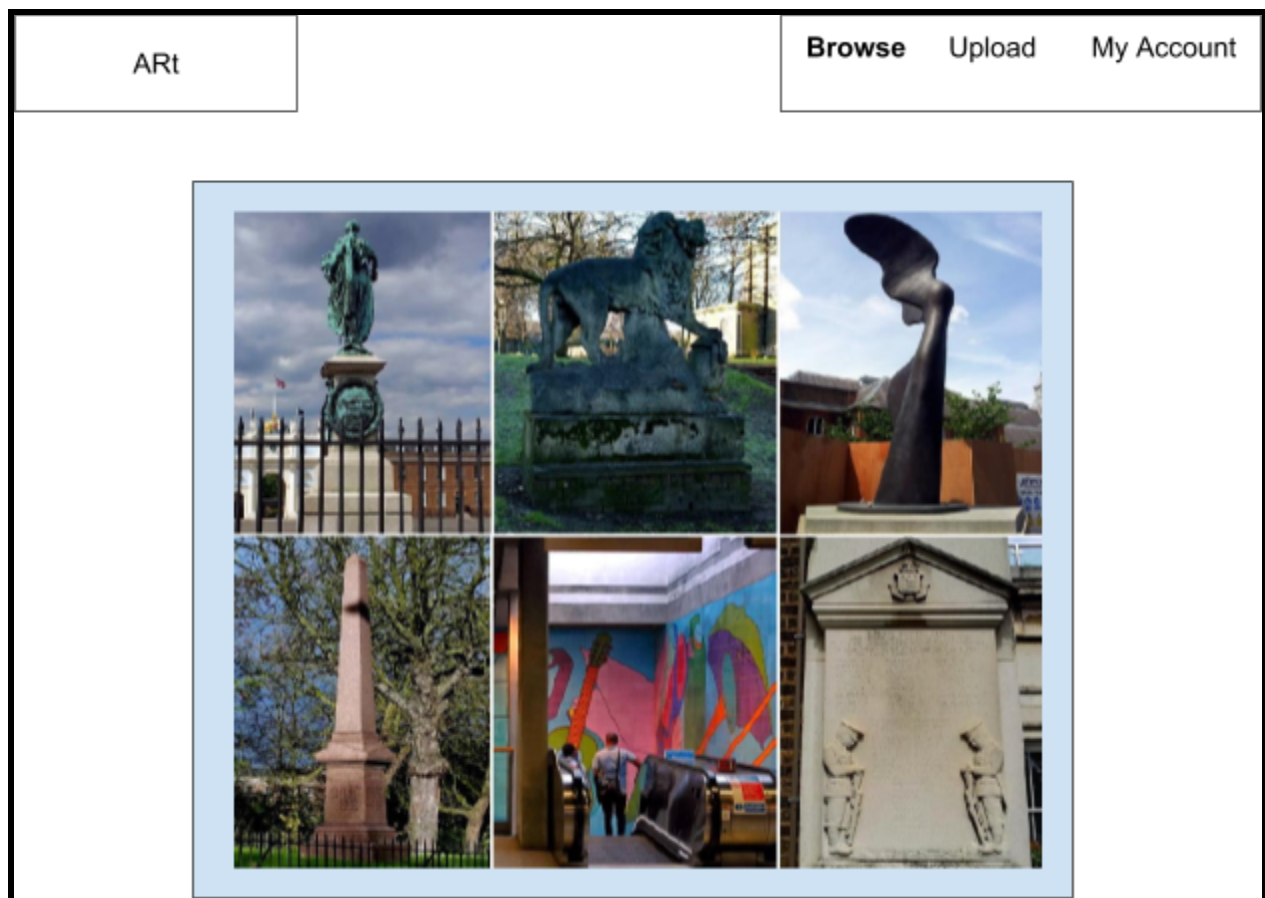


Figure 2 - Browse Screen

Name: iOS Scanner

Type: Screen

Purpose: This screen is needed to meet requirement 3.2 *Will be accessible to mobile users.*

Description: Figure 3 shows the layout for this screen. This screen will allow users to scan QR codes to get the corresponding artwork.

The screen contains the following elements: Settings, QR code scanner, swipe up menu for comments.

Layout:

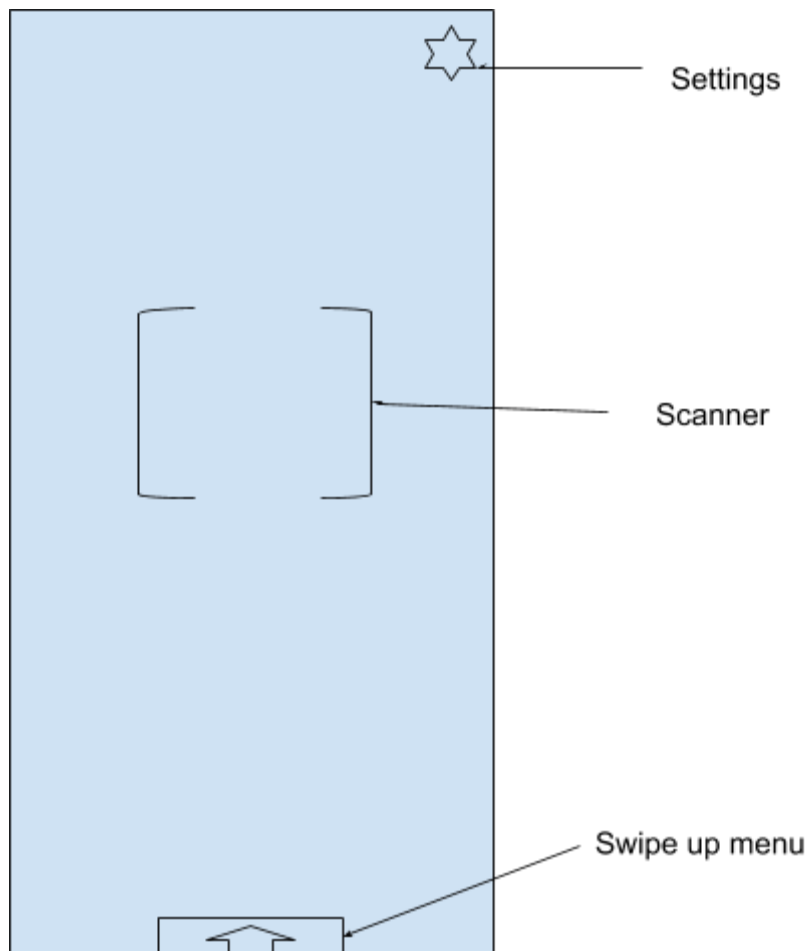


Figure 3- iOS Scanner Screen

Name: iOS Viewer

Type: Screen

Purpose: This screen is needed to meet requirement 3.3 *Will communicate with the API.*

Description: Figure 4 shows the layout for this screen. This screen renders the requested artwork using ARKit.

The screen contains the following elements: Settings icon, 3D artwork, swipe up menu for comments.

Layout:

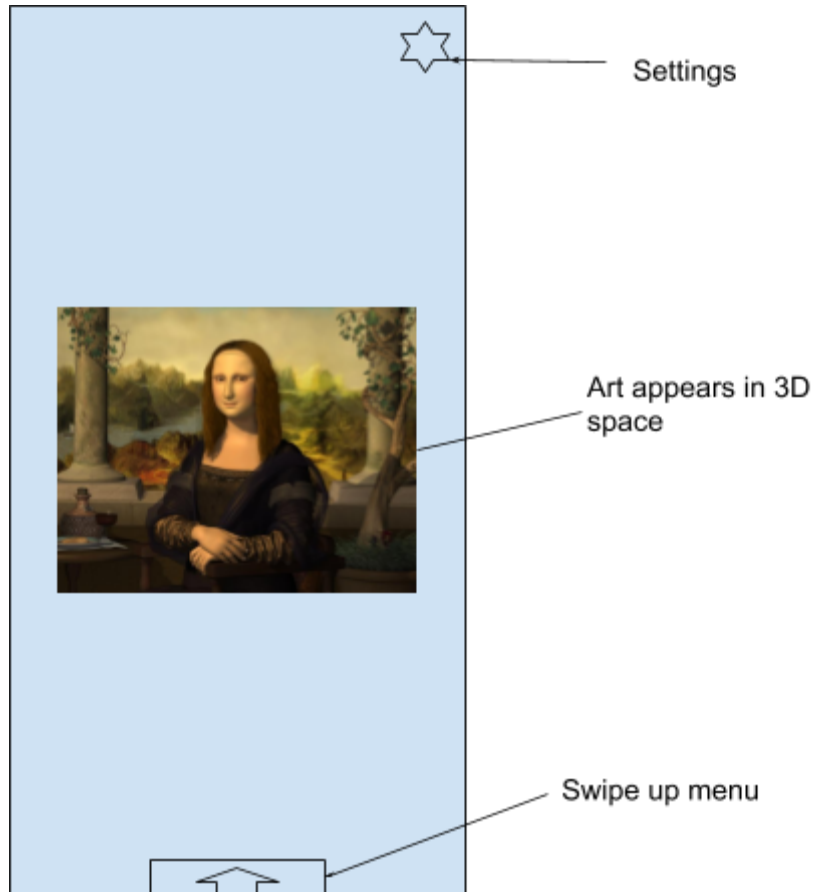


Figure 4 - iOS Viewer Screen

### Figure 7-3 – Template for Detailed Design for a Database Table

Name: ARt Stop

Type: Database Table

Purpose: This table is needed to meet requirement **3.1.2**

Description: Figure <N> shows the contents for this table. This table <describe the contents>. One row of this table represents <describe what one row represents>

Table Contents:

Data Element Name	Data Type	Key	Notes
image	string		URL to the image of the art to display
tags	List of strings		List of tags for the image.
genre	string		Genre of the art.
dateCreated	Integer		The date that the art was uploaded to the database
artistName	string		Real name of the artist.
artistUsername	string		Username of the artist.
id	string	AID	ID of the <b>ARt Stop</b> . Every <b>ARt Stop</b> has a unique ID, so that the client is guaranteed to get what it asked for.
vanityKey	string		The key of the decoration that a client could use to determine what to display.
comments	List of strings formatted as comment IDs.		IDs of all the comments on the ART work.

Figure 5 - Art Entry Database Table

**Figure 7-4 – Template for Detailed Design for a Code Function**

Name: getArt

Type: Function

Purpose: This function is needed to meet requirement **3.1.1**.

Parameters: The following parameters are used to call this function:

Name	Data Type	Notes
id	string	Gets an image from the database based on an ID.

Return Type: Promise<ArtObject>

Processing:

```
function queryArtDatabase() {  
  //...  
}  
  
function getArt(id) {  
  return new Promise((resolve, reject) => {  
    queryArtDatabase(id)  
      .then((art) => {  
        resolve(art);  
      })  
      .catch((error) => {  
        reject(error);  
      });  
  });  
}
```

**Figure 7-5 – Team Capability Assessment**

**	Matthew Kleiner	Phoenix Kline-Sanfosso	Anca Scarlat	Gurleen Singh
HTML, CSS	3	3	3	2
Frontend JavaScript	3	3	3	3
Backend Node.JS	3	2	2	3
MongoDB/Database Architecture	3	2	2	2
Cryptography/Security Practices	2	2	2	3
API Development	3	3	2	3
iOS Development	1	1	1	3
Graphic Design	2	3999999999	3	2
User Interface Design	3	3	3	3

\*\* The table values represent an assessment of team member capabilities. The values are:

- 1 – No knowledge or not much relative to the needs of this project
- 2 – Enough knowledge to accomplish part but not all this project
- 3 – Knowledge probably sufficient for this project