SUM200 Project Stargazer Documentation

# Introduction

The purpose of developing an application focused on viewing and collecting astronomical phenomena is founded in the strong human desire to explore the vast beyond. The application seeks to tap into the fundamental human traits of curiosity and awe. Stargazer seeks to sate those desires by giving the user a way to experience the advances in space faring and humankind´s exploration of the observable universe through their phone.

# User Stories

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Identification*** | ***User Role*** | ***Description*** | ***Acceptance Criteria*** | ***Priority*** |
| 1 | Explorer | Be able to see cherry picked images of astronomical phenomena | Daily picked images presented to the viewer consisting of astronomical phenomena | High |
| 2 | Explorer | Be able to see a wide variety of images with astronomical phenomena | Large selection of images consisting of astronomical phenomena | High |
| 3 | Explorer | Be able to see videos of astronomical journeys | Videos from spaceflights | Low |
| 4 | Collector | Gather images that interest me to have available later | Save images to a personal collection | High |
| 5 | Collector | Organize images to be able to better track and distribute them | Create lists of images, ability to add images to lists | High |
| 6 | Collector | Remove images that no longer interest me | Delete images from a personal collection | High |
| 7 | Collector | Change the organisational object to better reflect its contents | Ability to edit already created lists | Medium |
| 8 | Collector | Remove organisational objects that are no longer needed | Ability to delete existing lists | High |
| 9 | Collector | Transfer images between organisational objects to better reflect their intended content | Ability to move images between existing lists | Medium |
| 10 | Collector | Preservation of the contents of an organisational object if the object is removed | Images of deleted lists are not deleted | High |
| 11 | Explorer | Read text associated with astronomical images about their content | Ability to see descriptions of images | High |
| 12 | Explorer | Be engrossed in the astronomy atmosphere | Ability to listen to astronomy themed music in the app | High |

# Technical Features

The application has a hybrid data source structure where an internet connection allows the user to partake in new imagery and their contents. While an offline mode still allows the user to explore their existing collection. This is achieved by segregating the data from the different data sources to designated pages within the application.

The “Daily” and “Gallery” pages and their underlying pages consist of data fetched from a publicly available API service provided by the National Aeronautics and Space Administration (NASA). Data is directly fetched from their vast archived storage through the Astronomy Picture of the Day (APOD) API. On the other hand, data in the “Collection” page and underlying pages are fetched from a local SQLite database.

This database has mixed CRUD functionality for the two tables it contains. The collection the user amasses through lists have full traditional CRUD functionality while the images from the API work slightly differently. Since almost all the data related to the images are fetched from the APOD API the user can only add the image to their database in its existing state. Editing the data is only available in a limited fashion through assigning the image to different collection lists. While full traditional functionality for read and delete are available. The database manages the relationship between the images and the lists through a foreign key in the images table that stores the data of their assigned lists id.

The other main feature of the application is related to the in-app music player. To allow the user to be fully engrossed in the atmosphere and the theme of space exploration, the app allows the user to play astronomy themed music throughout the entire app. This music automatically loops without user interaction, can be played and paused, skipped to the next track or the previous one and consists of a number of astronomy themed tracks.

# Navigation Flow Chart

A diagram of a company

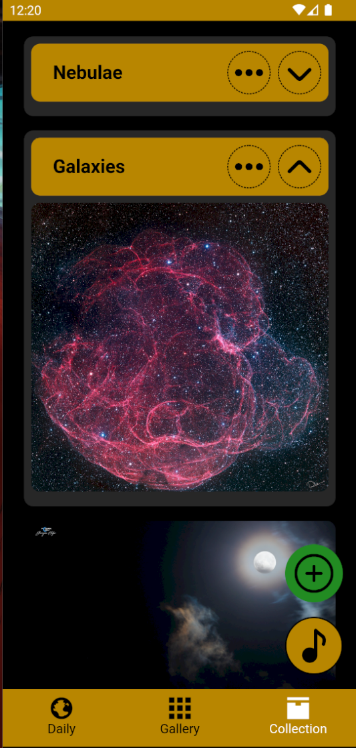
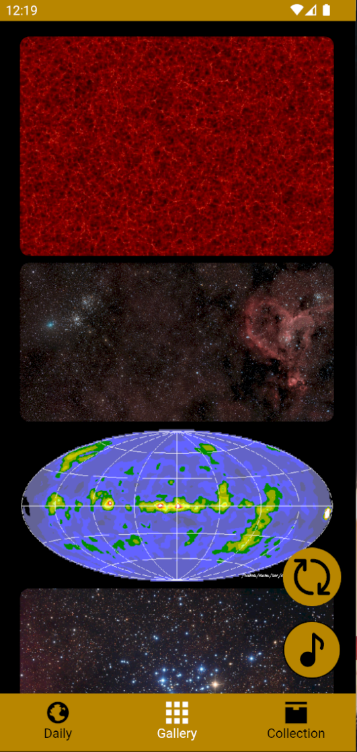
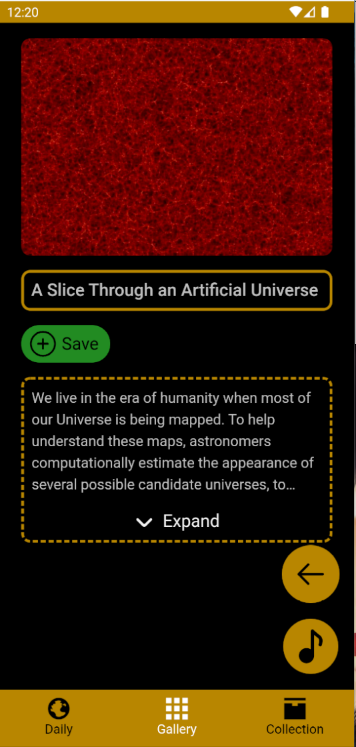
Description automatically generated

When starting the application, the user is always guided to the “Daily” page within the app. This page presents the user with the daily image from NASA´s APOD API, which includes a title, image, and description. The “Daily” as well as the “Gallery” and “Collection” pages are all always available to navigate to through the bottom navigation bar.

The “Galley” page contains 5 random images from the APOD API, these images can be inspected closer by pressing on them which will take the user to the “GalleryDetail” page. The “GalleryDetail” page consists of the same information presented in the “Daily” page but this time specifically for whatever image the user pressed on in the “Gallery” page.

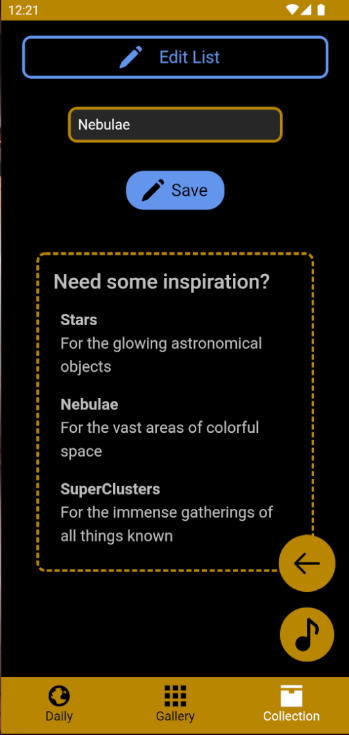
The “Collection” page is made up of images the user has added to their collection as well as lists. These lists contain images the user has assigned to them and the “Collection” page allows the user to navigate to the “CreateList” page where these lists are made. The “CreateList” page allows the user to make new lists for organising their images. Parallel to the “CreateList” page the “EditList” page accessible from the “Collection” page allows the user to edit the attributes of a list they have previously created. Also available from the “Collection” page is the “CollectionDetail” page where the user exactly as in the “GalleryDetail” page can inspect individual images in their collection. The information contained in the “CollectionDetail” page consists of a title, image and a description. Furthermore, this page also allows the user to navigate to the “MoveImage” page related to the image they are currently inspecting. The “MoveImage” page exactly as the “GalleryDetail” and “CollectionDetail” page is individualised based on what image the user inspected previous to navigating there. These pages change their content based on what image they are currently tied to. The “MoveImage” page allows the user to assign an image to a list or reassign an image that already is associated with a list.

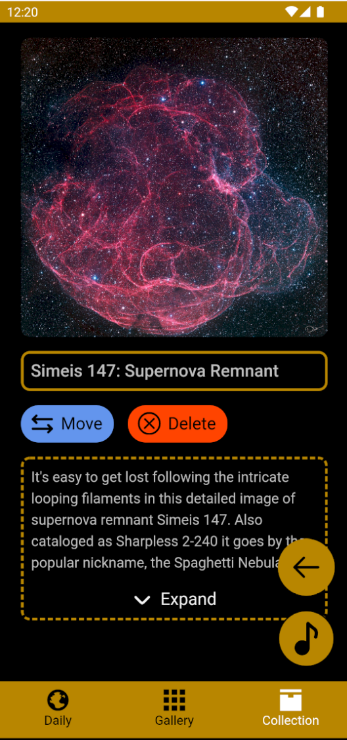
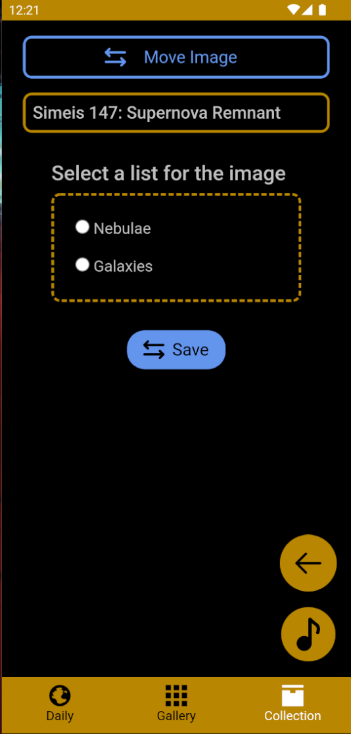
The focus of the chosen navigation pattern is to aggregate related functions in a logical structure. Actions related to images are all accessed from within the detailed view of the image. Actions related to list are accessed from the page where they are presented. A parallel ambition to this was to make pages and actions more easily accessible for the user by widening the app structure instead of deepening it. An example of this is the fact that all fundamental pages (Daily, Gallery, Collection) can be accessed from anywhere within the app and no page is more than two pages away from the fundamental pages. The goal is to make the app easier to navigate and quickly allow new users to understand the layout.

**A screenshot of a cell phone screen

Description automatically generated

*Daily Gallery GalleryDetail Collection*

*A screenshot of a phone

Description automatically generatedCollectionDetail MoveImage CreateList EditList*

# Graphical Design

#B8860B #000000 #bfbfbf #238E23 #42426F #FF4500 #FFFFFF

The application design consists of a set of primary and secondary colours, where the primary colours are found in the fundamental elements of the application, the secondary colours indicate a specific action or element. The primary colours consist of #B8860B, #000000 and can be found all over the application. They are used to mark elements, highlight elements and create a basic foundation for the secondary colours to act upon. As well as serve as a colour for action buttons that don’t add, modify or remove objects.

The less significant of the secondary colours are #bfbfbf and #FFFFFF, they are used for text and bottom navigation icons. The more intense #FFFFFF is selected to create a greater contrast against background of the bottom navigation bar. Which makes the text more easily readable. #B8860B however is used on the pages themselves and is a less intense variant of #FFFFFF. Since #B8860B always contrasts against #000000 there is no need to use a more intense colour which reduces the strain on the eyes.

The mode significant secondary colours are #238E23, #42426F and #FF4500, they always indicate a specific action which alerts the user to the possibility of that action. #238E23 is used when objects can be saved, created or added. #42426F is used when objects can be changed, edited or modified. #FF4500 is used when objects can be removed, erased or deleted. The consistent colours ensure that the user always knows what to expect from a coloured action button.

The primary colours of #B8860B, #000000 were chosen as they symbolize the most significant astronomical phenomena close to earth. That being the Sun itself, looked upon the vast darkness of space the sun stands out as a golden orb of light giving life. The more significant of the secondary colours #238E23, #42426F and #FF4500 where chosen since they form the basis of an RGB colour palette. Which are well established colours that the user can easily identify. Their colour in combination with an icon helps the user identify with a high degree of certainty the rough function of an action button even without any text.

# Code Example

private EventCallback DeleteList(CollectionList collectionListItem)

{

    return EventCallback.Factory.Create(this, async () =>

    {

        bool confirm = await js.InvokeAsync<bool>("confirmDelete", new object[] { "Are you sure you want to delete this list?" });

        if (confirm)

        {

            bool success = await db.DeleteCollectionList(collectionListItem);

            if (success)

            {

                var toast = Toast.Make("List deleted", ToastDuration.Short, 14);

                await toast.Show();

                // Refresh the page

                \_collectionList = await db.GetCollectionLists();

                \_collectionImages = await db.GetImages();

            }

            else

            {

                var toast = Toast.Make("List could not be deleted", ToastDuration.Short, 14);

                await toast.Show();

            }

        }

    });

}

This example is a function inside the “Collection” page pertaining to the deletion of lists. Since the “@onclick” element attribute doesn’t work with normal functions it needs to be structured into an “EventCallback” function. The function takes a “collectionListItem” as a parameter which is supplied by the associated list the button is attached to. The function consists of a return statement involving an EventCallback Factory which contains the page side deletion code. The function first prompts the user to confirm their deletion command by invoking a JavaScript alert that asks the user to confirm their choice. The value returned from this alert tells the function whether it should proceed or not. If the user wishes to proceed the function calls the relevant database function to perform the task. The result of the database function returns and informs the user whether the operation was successful or not.