

Rebecca Popper

Assignment 5- OSS Analysis & Design

Project: Routine Visualizer

Github: <https://github.com/popperr2/RoutineVisualizer>

October 27, 2017

Dr. Pulimood

Project Analysis & Design

Use Case Descriptions

Use Case: Login with Account Username and Password

Primary Actor:	Mobile Application User
Goal in Context:	Enter username and password, get validation, and continue into the mobile application.
Preconditions:	The user has a registered username and password with the system
Trigger:	The user opens the application for the first time or a time following a logout command.
Scenario:	<ol style="list-style-type: none">1. The user opens the mobile application on their mobile device.2. The user enters their username into the username input area3. The user enters their password into the password input area4. The system checks to see if the entered information is valid5. The information is validated and the mobile application continues to the next screen
Exceptions:	<ol style="list-style-type: none">1. The username or password are incorrect - the system notifies the user that their input is incorrect and does not continue into the application.
Priority:	Moderate priority, will be implemented after basic app functionality.
Frequency of Use:	Infrequent, the user will only do this the first time they use the app or the first time they open the app after a logout.
Channel to Actor:	Via mobile device and internet connection
Open Issues:	<ol style="list-style-type: none">1. Will a “remember me” login feature be added to ease use?2. What mechanisms protect unauthorized use of the login System?

Use Case: Create New Routine

Primary Actor:	Mobile Application User
Goal in Context:	To create a routine, or a series of steps, that can be used over and

	over again in the daily schedule.
Preconditions:	User must have validated at login.
Trigger:	User wishes to add a new routine to their list of routines.
Scenario:	<ol style="list-style-type: none"> 1. The user taps on the “new routine” UI element. 2. The user types in a descriptive name or emoji for the new routine. 3. The user adds a pre-loaded or a personal action to the routine. 4. The user will add an image, multiple images, and / or text to the action. The user will set the action’s time to complete property and priority property. 5. The continues adding new actions until their routine is complete. 6. The user will give the routine a tag, such as “morning”, “night”, “hygiene”, or “pet.” 7. The user taps on the “confirm” UI element. 8. The system adds the new routine into the database of routines.
Exceptions:	<ol style="list-style-type: none"> 1. The user does not name the routine with text, an image, or an emoji, or gives the routine a pre-existing name - the routine is invalid. 2. The user does not add 1 or more steps to the routine - the routine is invalid.
Priority:	High priority, an essential function of the program.
Frequency of Use:	Moderately frequent, especially when first using the application.
Channel to Actor:	Via mobile device and internet connection
Open Issues:	<ol style="list-style-type: none"> 1. Will the user be able to give a description of what the routine is, rather than just a title? 2. Will the system save this information locally?

Use Case: Create Daily Schedule (with Algorithm)

Primary Actor:	Mobile Application User
Goal in Context:	To create a daily schedule for the day, and be assigned routines that
	match the schedule.
Preconditions:	User must have validated at login.
Trigger:	User wants to create a daily schedule for the day or for a future day.

Scenario:	<ol style="list-style-type: none"> 1. User taps on “Create daily schedule” UI element. 2. User has the option to add things to the day that must be completed at specific times, such as “Doctor’s appointment at 10:00 AM.” 3. System will generate a daily schedule for the user based on the routines they have under certain tags, creating a daily routine around the user’s responsibilities. 4. The user will have the option to manually change the generated schedule’s routines. 5. The system will upload this schedule to be able to be completed.
Exceptions:	<ol style="list-style-type: none"> 1. The user manually removes all routines and appointments from the daily schedule - the schedule is invalid.
Priority:	High priority, an essential function of the program.
Frequency of Use:	Very frequent, a daily action almost every day
Channel to Actor:	Via mobile device and internet connection
Open Issues:	<ol style="list-style-type: none"> 1. Will an option be created to save or repeat a daily schedule to use another day?

Use Case: View and Complete Daily Schedule

Primary Actor:	Mobile Application User
Goal in Context:	To view the daily schedule and be able to complete routines and actions as the day progresses.
Preconditions:	User must have validated at login and created a daily schedule.
Trigger:	The user wants to view and complete their daily schedule to help them get things done and use the application’s main functionality.
Scenario:	<ol style="list-style-type: none"> 1. The user navigates the to main page of the mobile application 2. The mobile application will display the current action of the current routine the user should be doing. If the user should not currently be doing a routine or action, the application will display what they will be doing next and when it takes effect. 3. The user can complete actions by tapping UI elements notifying the system they are complete. 4. The system keeps track of the current time and what the user should currently be doing.

5. The system will skip actions if their time window disappears (ex: the system won't tell you to brush your teeth if it's already after lunch).

Exceptions:

1. The user has not created a daily schedule - the daily schedule viewer will not display a schedule unless a new one is created.
2. The user has completed their entire daily schedule for the day - the daily schedule viewer will display "Done" UI element.

Priority:

High priority, an essential function of the program.

Frequency of Use:

Very frequent, a daily action every day

Channel to Actor:

Via mobile application and internet connection

Open Issues:

1. Will other elements other than just routines and appointments be able to be added to the schedule? (breaks, sensory stimuli)
2. Will the schedule be customizable in the daily schedule viewer Module?

! UPDATED

Modularity and Encapsulation

Encapsulation and association will be used for the classes of Routine, VisualizerRoutine, and Action. Routine contains Actions, and VisualizerRoutine is built off of Routine.

VisualizerRoutine is a class that is used specifically for the DailySchedule view of the application. This view implements two new variables, the IsDone and IsInProgress Booleans. This is why there needs to be a visualizerRoutine to extend off of Routine.

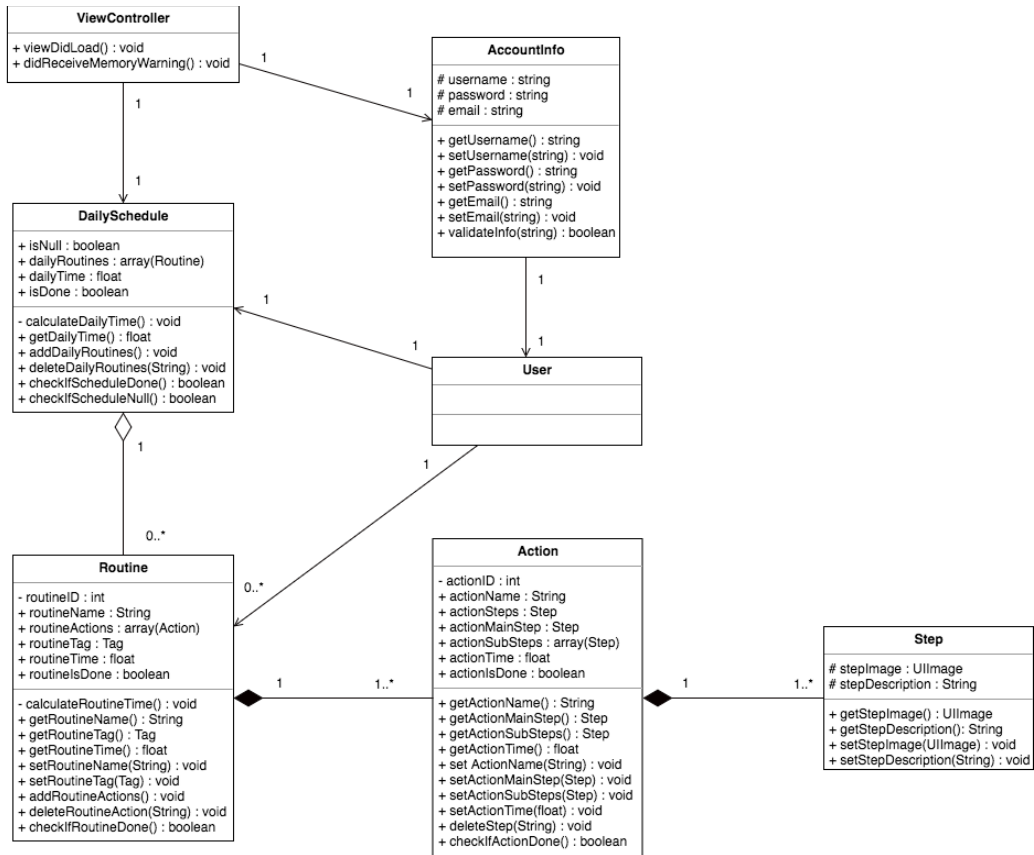
The program is modular because there are very few functions that are dependant on other functions. There are a few functions that are dependant on other functions, but only where it is necessary as per the Swift 4 language structure such as implementing UITableViews.

! UPDATED

Data Structure

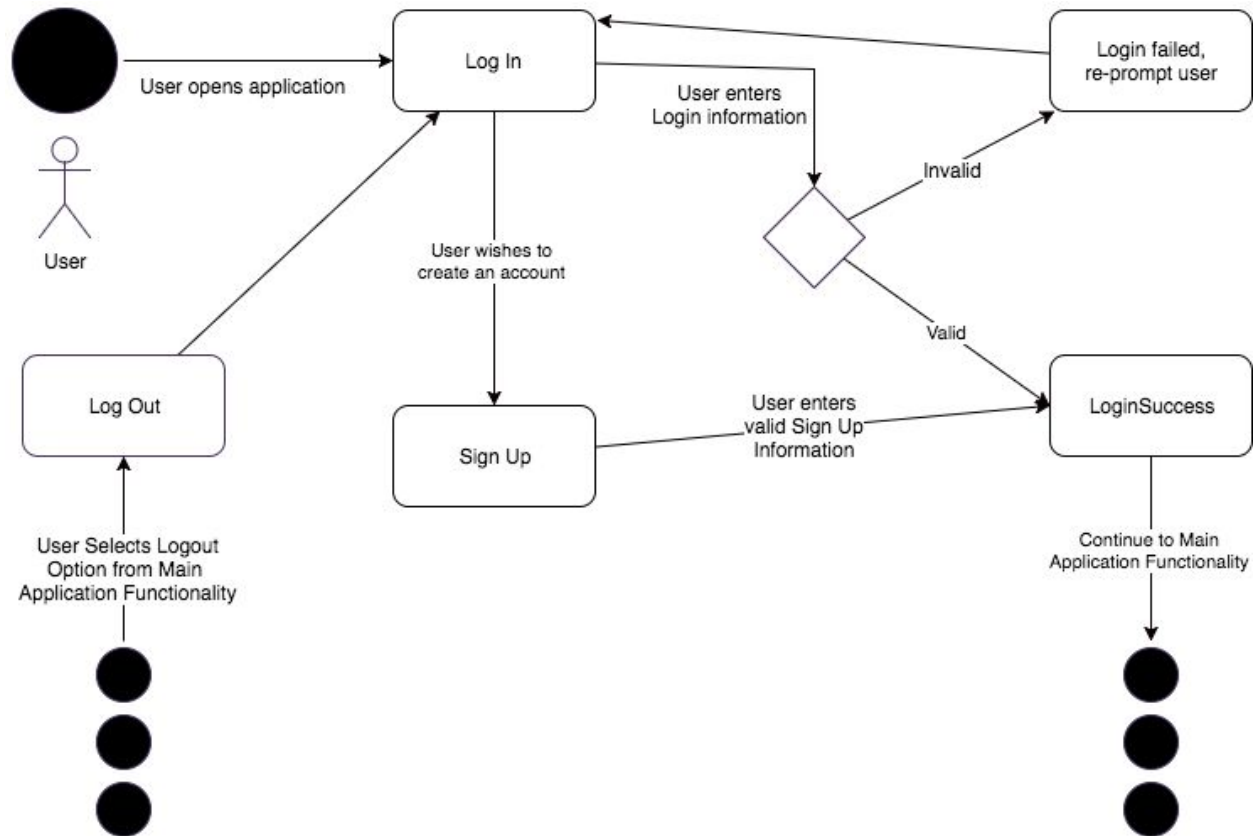
The data structure that this project is going to use a lot of is an Array. I will also be using Maps and SQLite tables to create the app. All of these will work together and interact with each other in the algorithm. The program will take information from the SQLite table data structure and import it into an array when loading. Then when saving, the program will take the array's contents and insert it back into the SQLite table data structure. Maps are used to navigate these structures at times.

Detailed Design Class Diagram

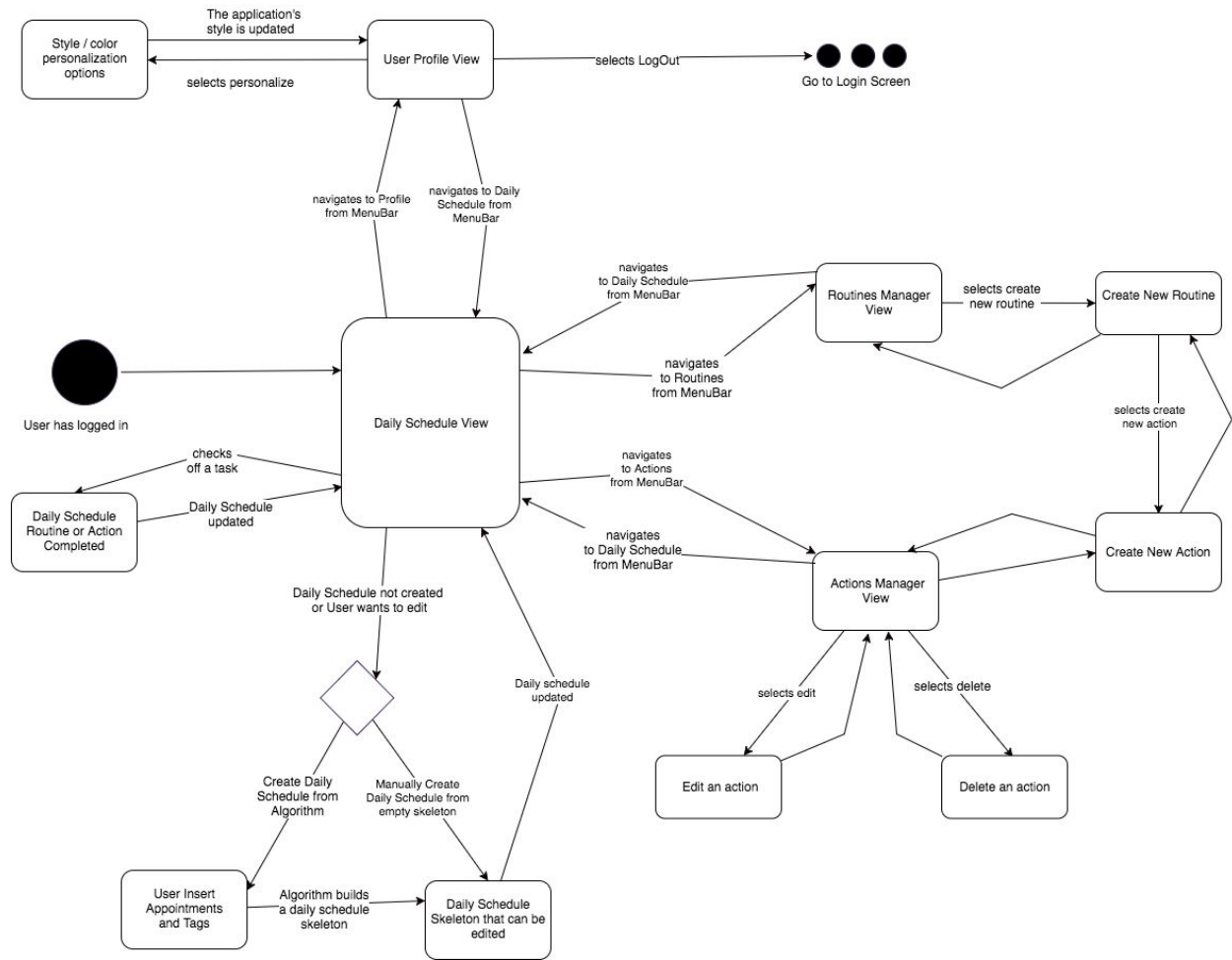


Statecharts

Login / Account Creation System - Statechart

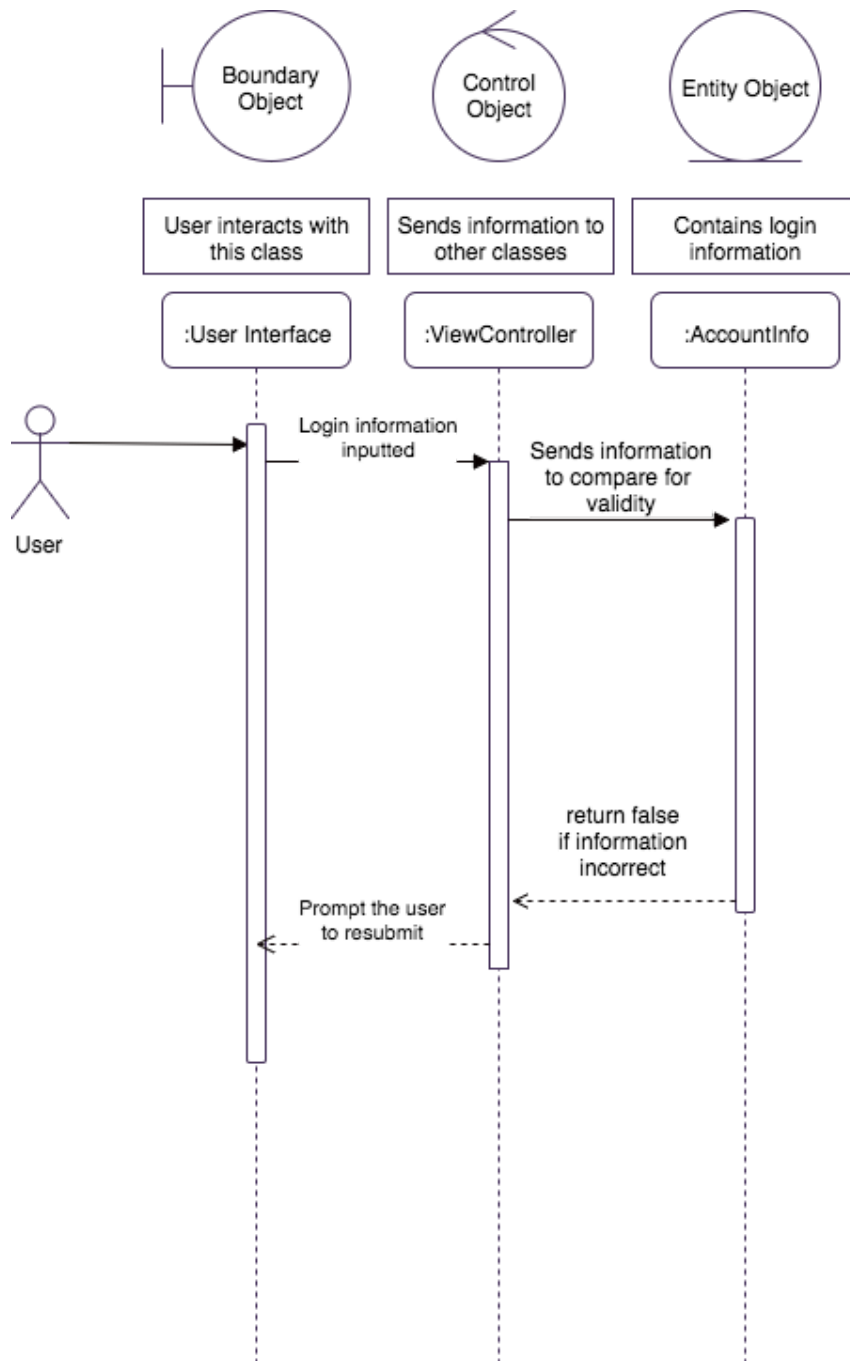


Main Application Functionality - Statechart

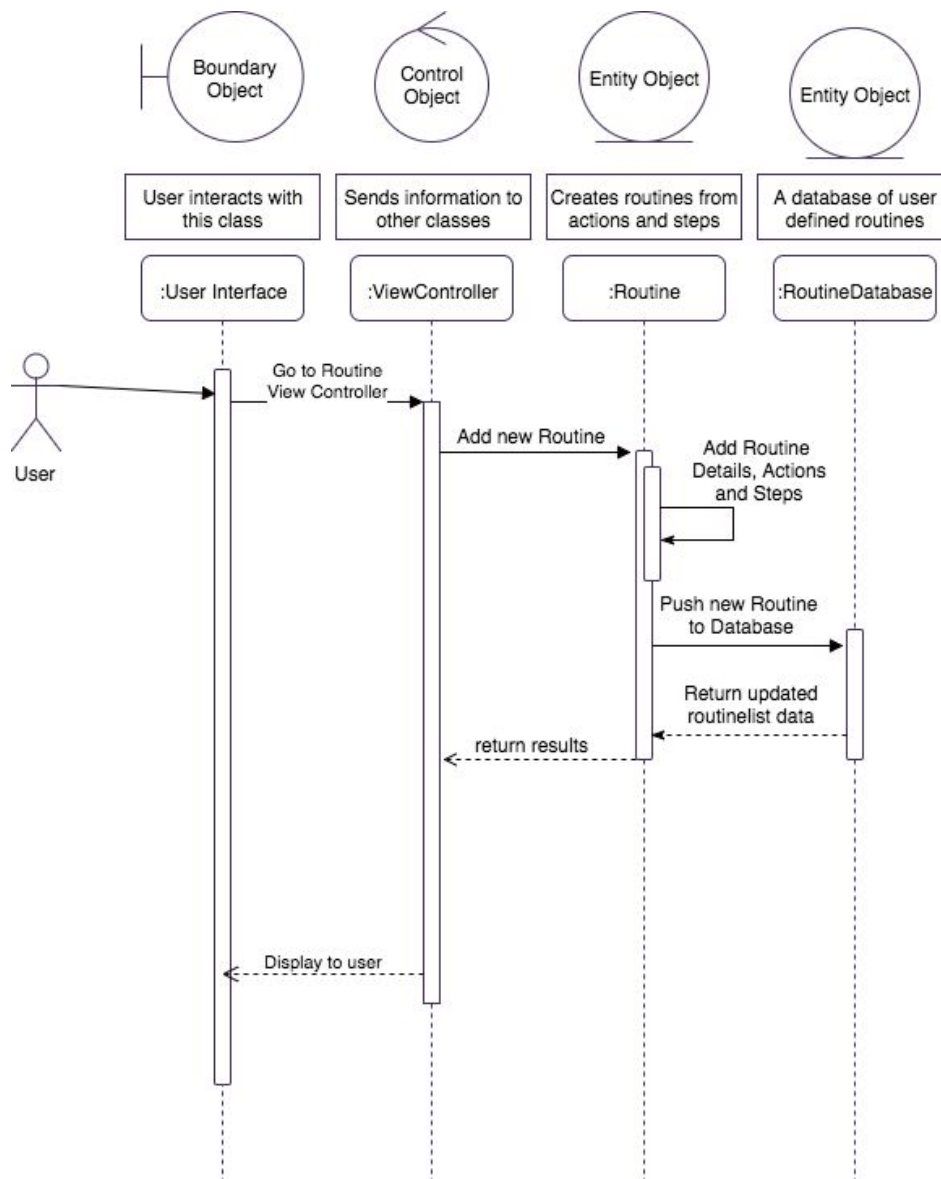


Detailed System Sequence Diagram(s)

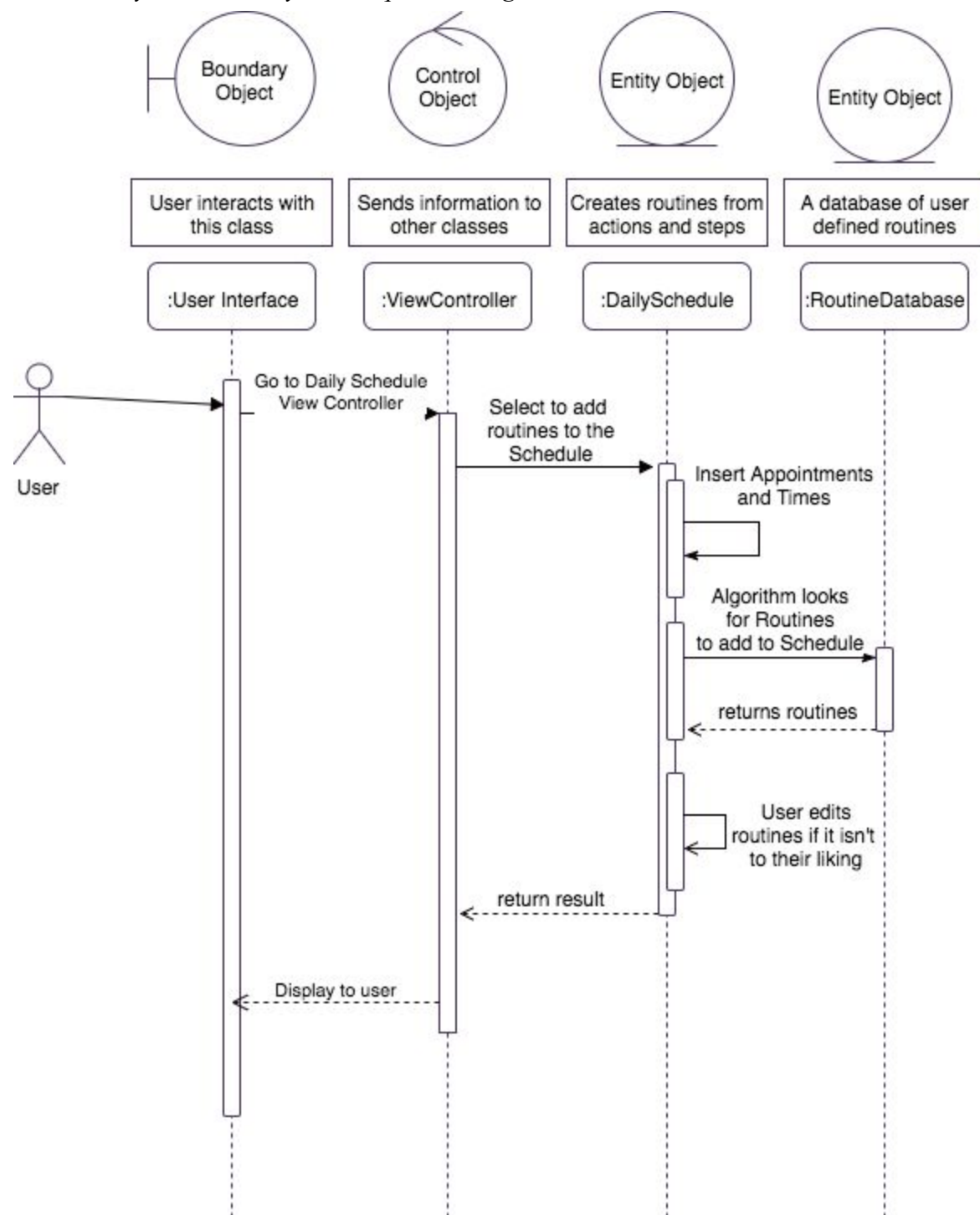
Login System - System Sequence Diagram



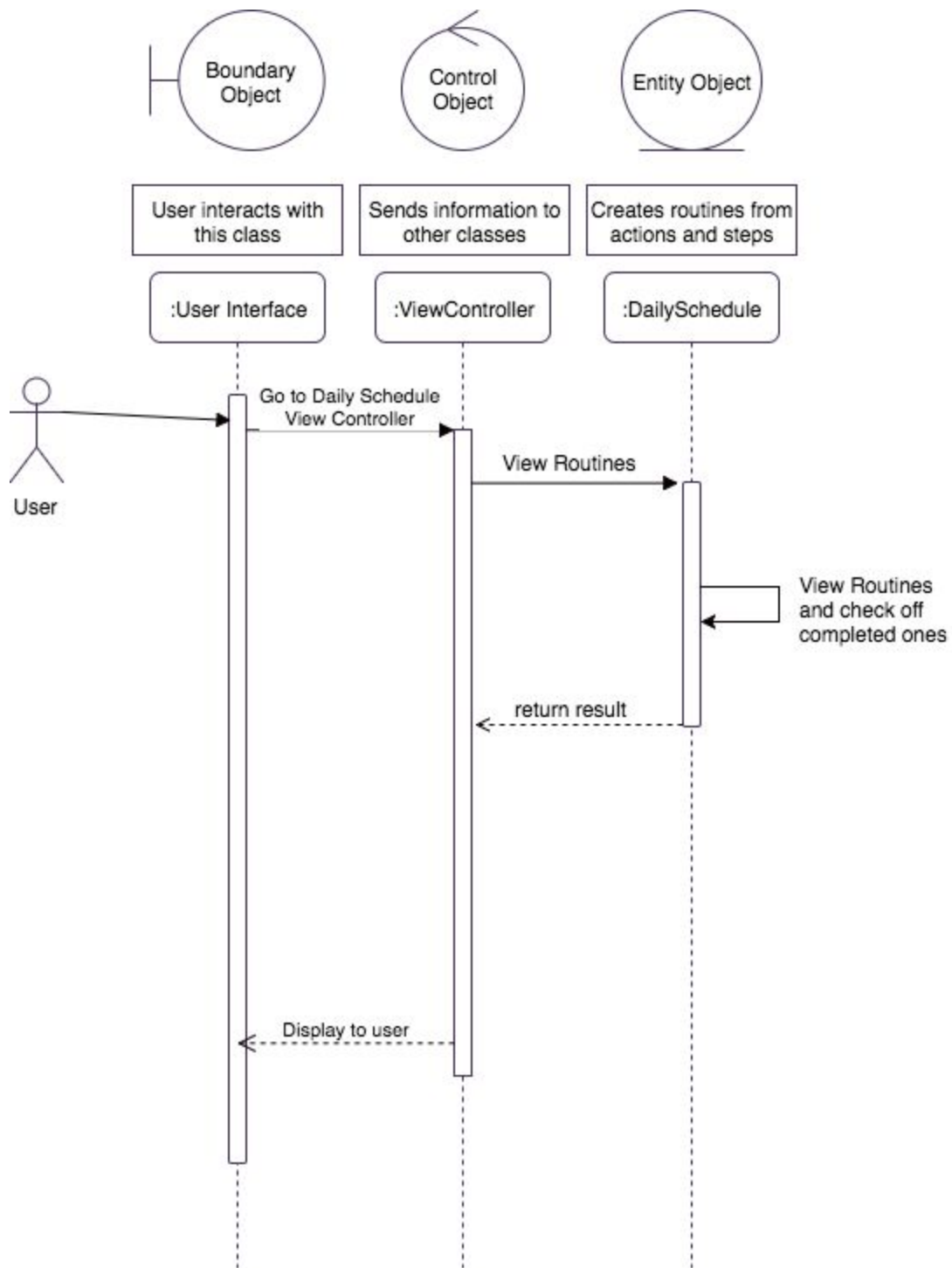
Create New Routine - System Sequence Diagram



Create Daily Schedule - System Sequence Diagram



View / Complete Daily Schedule - System Sequence Diagram




User Interfaces (UI) Mockups

Mockup Login / Register Views:

ROUTINE
VISUALIZER
LOGO

USERNAME

PASSWORD 


LOG IN

Don't have an account? Sign Up

ROUTINE
VISUALIZER
LOGO

EMAIL

USERNAME

PASSWORD 

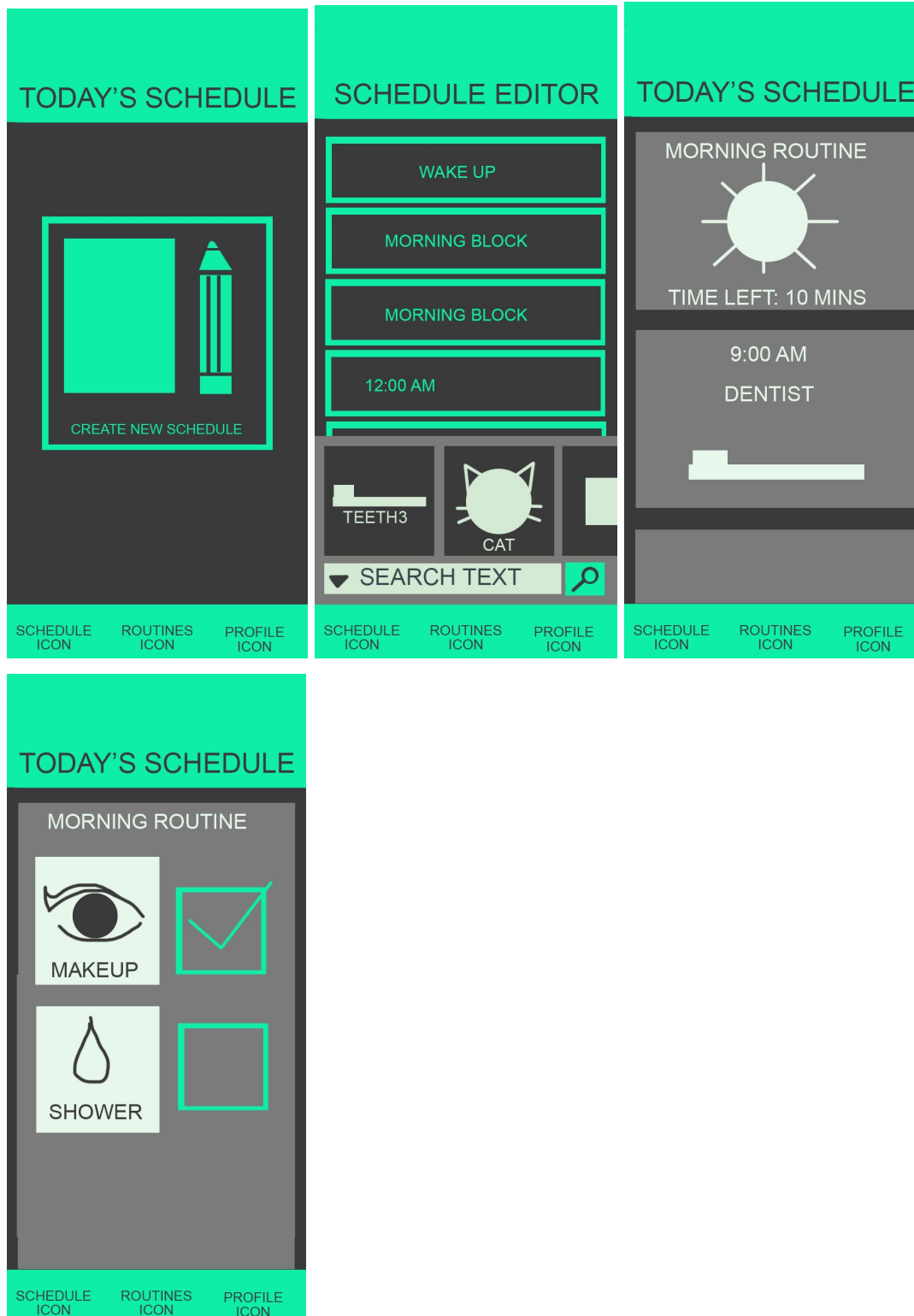
SIGN UP

Already have an account? Sign In

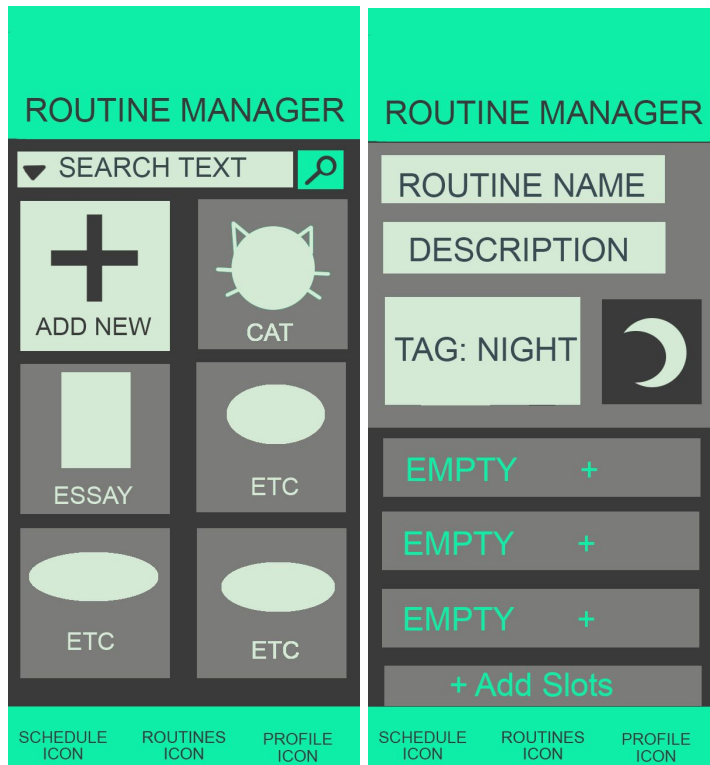
Log in

Register

Mockup Daily Schedule View:



Mockup Routine Manager View:



View Routines

Add New Routine

The Eight Golden Rules of UI Design

1. Strive for Consistency

The application is consistent in many ways. Familiar icons, colors, and menus adorn the application. Icons always mean the same thing, such as magnifying glasses always meaning “search” and plus symbols always meaning “add.” Knowledge the user learns in the one part of the application, such as the Routine Manager, can be applied to the Daily Schedule Editor. They both create different schedules, but are laid out very similarly in structure. The color scheme is simple and consistent across all interface screens. The user flow of the application is very consistent; the way that daily schedules and routines are created will stay the same no matter what length of schedule or type of routine it is.

2. Enable Frequent Users to Use Shortcuts

Frequent users have many shortcuts available at their disposal. When creating the daily schedule, frequent users will be able to reach into their pool of user-created routines to add to the schedule from the routines database instead of having to create their routines every time they open up the application. Frequent users will also be able to use the daily schedule they used yesterday, or another day.

3. Offer Informative Feedback

The application will always display the title of which page the user is on. It will also display the title in icon form at the bottom of the user interface on the tab bar. The user will have a clear sense of where they are in the application due to these properties. The user will know what is going on at all times due to the apps image-based and text-based clear and concise feedback. There will be icon feedback on pages such as the login page to show the user if their login information is validated or not.

4. Design Dialogue to Yield Closure

The application will tell the user what they have done where it is needed. If they have created a new daily schedule, the user interface will alert them with “daily schedule created” and take them to view the schedule. If they have created a new routine, the interface will say “routine created” and add it to the top of the current routine list. The user will be able to see the results of their actions very clearly.

5. Offer Simple Error Handling

The application will notify users with concise error messages if they are not meeting the requirements, along with images to convey the idea (“X” symbolizing that something went wrong). Most text fields do not need to be filled out in the application and are optional. Only selecting images will be required for most commands, so that the application could be used with minimal use of words.

6. Permit Easy Reversal of Actions

There will be a cancel option for all commands that can be inputted. There will also be options to delete routines and parts of the schedule if the user changes their mind. If the user does the wrong actions and panics, hitting any of the icons on the bottom tab bar will return the Views to their default states (E.g. the routine view will return to routine manager; the daily schedule will return the default daily schedule view). The user will never be lost in the menus due to these precautions.

7. Support Internal Locus of Control

The user can create daily schedules manually or by the algorithm, and they can use pre-programmed routines or create their own. The system itself will feel under their full control especially if they upload their own images. If they don’t upload their own images, they will still have options to choose the images they want, so they will feel like the application is personalized and their own. The colors of the app will be able to be chosen by the user from the profile view tab.

8. Reduce Short-Term Memory Load

The application will be easy to use for first-timers or for long-timers. Recognition will be high, as everything will have image labels and word labels. The use of affordance will aid in reducing short-term memory load, as people associate checkmarks with done and plus symbols with add, and magnifying glasses with search. The application will feel very familiar and easy to use upon first touch, and yet bring comfort due to its simplistic styling.

Test Case Design

Unit Testing, Integration Testing, and System Testing Test Case Approaches

My unit testing approach will be to test core functionality, UI workflows, and boundary conditions. For example, unit testing will be very helpful in making sure that the user can only input approved usernames and passwords into the login area, which will help avoid attacks to the software or database. I will be using Unit tests that are automated and return a pass or fail value. My integration testing and system testing approaches will be similar to the unit testing approach in the fact that it should be checking that core functionality works. I will try to make both of these automatic as well.

Debugging / Testing Tools

Unit Testing is a built-in feature of XCode Development Tools for Swift native application development. I will be using this feature to create Unit Tests. This is done by creating a swift file that extends the class XCTestCase. XCode will allow this Unit Test to be run from the program and has its own testing UI for it.

Integration testing and system testing are not pre-built into XCode, so I will use a third party application to help me write tests in these areas. There is a test automation framework called Appium that works on iOS native applications. I will implement something like this in order to write these tests.

Test Cases

Functionality Tested	Inputs	Expected Output	Actual Output
Login System	String username String password	Password is checked and if it is validated with the username, then the application continues to the next screen.	
Signup System	String email	All inputs are	

	String username String password	checked and if they meet requirements, an account is made for the user and they are logged into the application.	
Create Routine	Action[] actions	The user inserts 1 or more actions into a routine object that is saved into their routines database.	
Create Daily Schedule Algorithm	Routine[] routines	The algorithm inserts 1 or more actions into the daily schedule object that is displayed for the user to confirm or cancel.	
View & Complete Daily Schedule	Bool isDone	The daily schedule updates as the user checks off if certain actions are done or not, and it displays the current schedule. If there is no schedule to display, it tells this to the user.	

Prototype

<https://github.com/popperr2/RoutineVisualizer>

A prototype containing “dummy” functions and functionalities that prototypes the user flow of the application through different application pages and viewcontrollers.