Michael Pangburn

Jett Moy

César Chacon

Mathew McHugh

Aaron Rice

Adam Berard

**User Requirements and Functional Requirements**

1. The application shall enable the user to create Do-Its to track their class assignments.
   1. The application shall encode Do-It data as JSON and store it locally in the documents directory of the iOS device. (Functional)
2. The application shall send notifications to the user of upcoming due dates for Do-Its.
   1. The application shall send the user a notification for each Do-It either a day before its due date or at a user-specified time. (Functional)
3. The application shall allow users to share Do-Its with one another.
   1. The application shall enable users to share a Do-It via AirDrop or by email using our custom file format, .doit. (Functional)
4. The application shall allow users to sort Do-Its by class, due date, or completion priority.
   1. The application shall use the sorting algorithm provided by the Swift Standard Library for sorting Do-Its. (Functional)

**Non-functional Requirements**

1. The user shall be compatible with iOS 10 and newer.
2. The user shall be compatible with iPhone 5 and newer.

**User Stories and Acceptance Criteria**

1. As a user, I want to be able to view a list of my Do-Its so that I can see my assignments in order.
   1. I can press a button that leads me to my Do-It section.
2. As a user, I want to edit my Do-Its so that I can update my Do-Its when necessary.
   1. I can select a Do-It to edit.
   2. I can mark a Do-It as complete.
   3. I can hide/delete completed Do-Its.
3. As a user, I want to be able to create a new course so that I can organize my Do-Its by subject.
   1. I can press a button to name a new course.
   2. I can add Do-Its under course sections.
4. As a user, I want to filter my Do-Its by due date, class, or priority so that I can decide which Do-Its are most critical to complete.
   1. I can see which Do-Its are due for a specific day or week.
   2. I can see all my Do-Its for any specific class.
   3. I can sort Do-Its by the priority I have assigned them.

**Risk-Driven Specification**

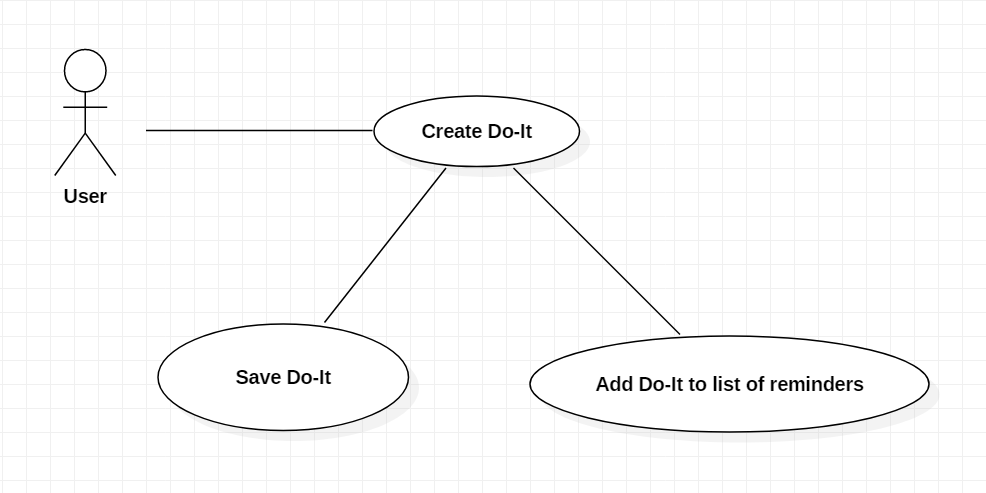
1. Denial of Service Exploit
   1. Identification - Malicious user creates custom .doit file containing extremely large quantity of Do-It data
   2. Analysis - Low risk security issue, possible crashing of application
   3. Decomposition - Our application enables users to import Do-It data via a custom .doit file format
   4. Reduction - Limit size of accepted .doit files; do not attempt parsing if file exceeds size limit
2. Tampering Exploit
   1. Identification - Malicious user gains write/read access to .doit files
   2. Analysis - Medium risk, malicious user may gain information about Do-Its and/or change Do-Its
   3. Decomposition - Our application stores data locally via a custom .doit file format
   4. Reduction - Encrypt .doit files

**Security and Safety Requirements**

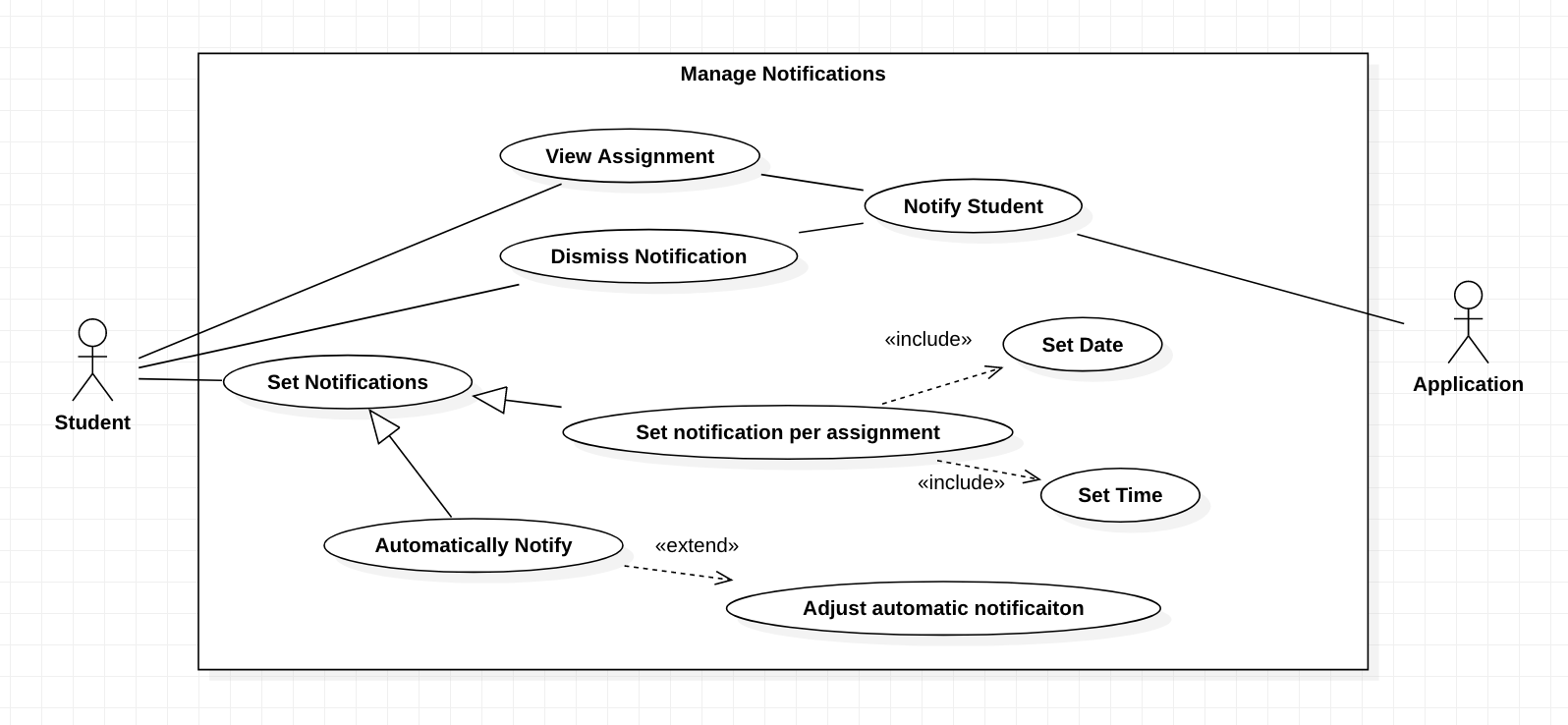
1. The system shall not attempt to parse .doit files over a given size.
2. The system shall encrypt saved .doit files to prevent against tampering.
3. The system shall validate the file format to be a .doit file.
4. The system shall only allow access to the file that was sent.

**Use Case Diagrams**

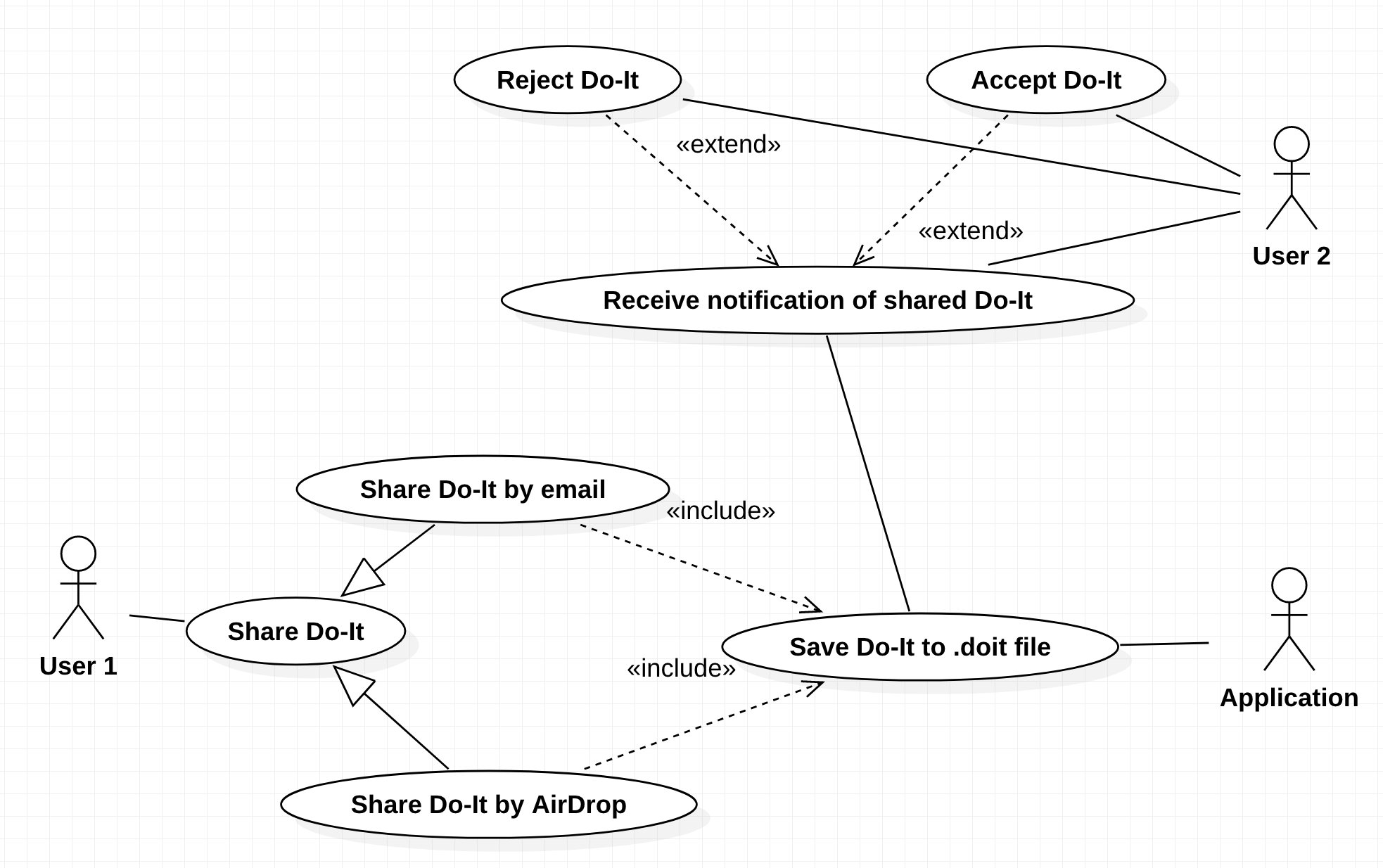
1. Do-It Creation



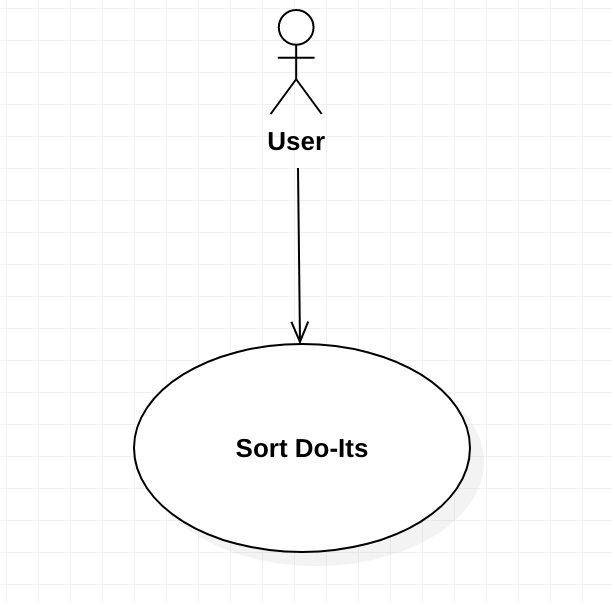
1. Do-It Notifications



1. Do-It Sharing



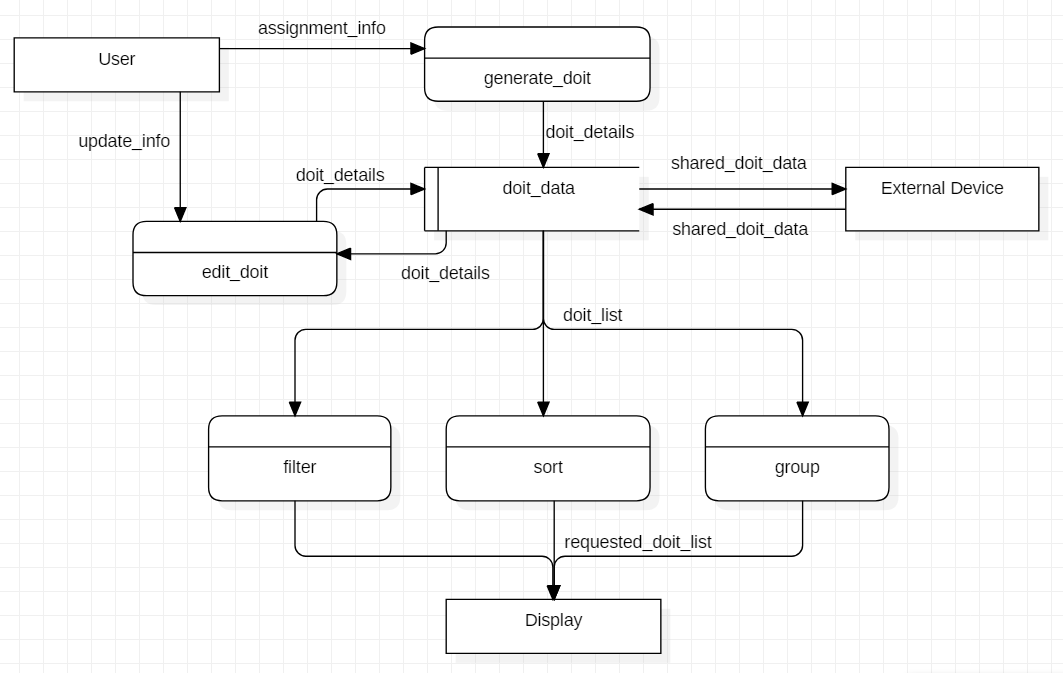
4. Do-It Sorting



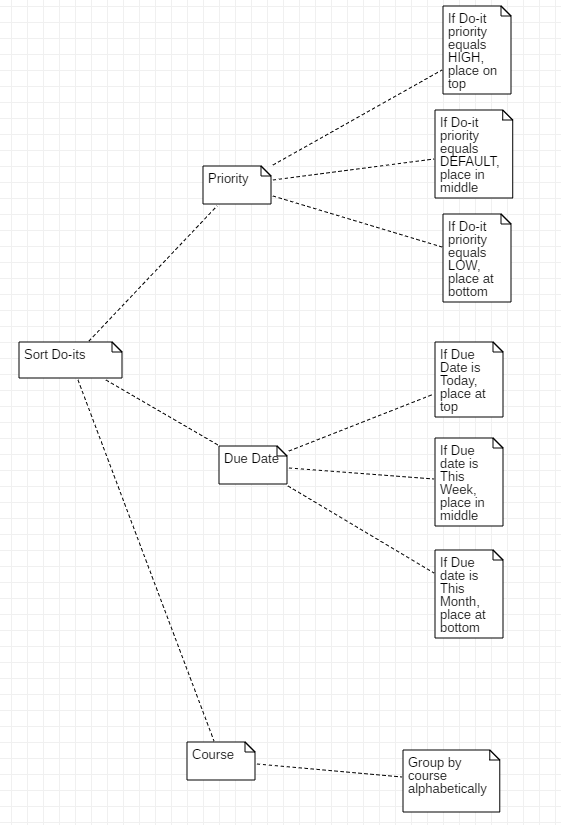
**Glossary**

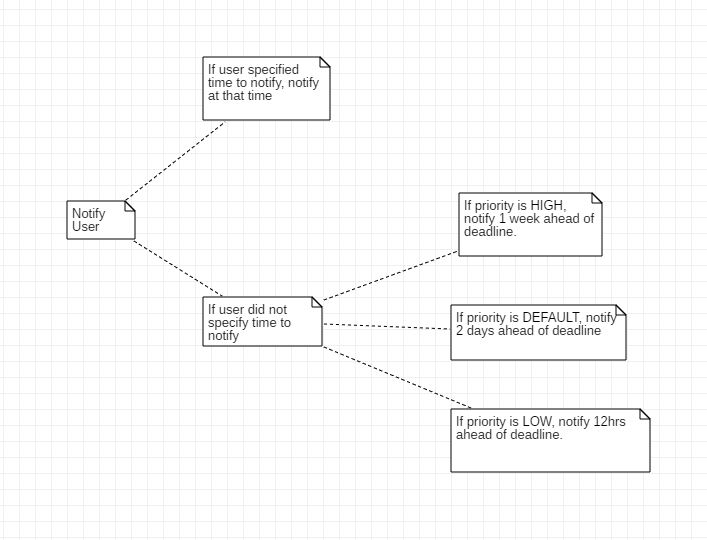
* **AirDrop:** enables the transfer of files among supported Macintosh computers and iOS devices over Wi-Fi and Bluetooth, without using mail or a mass storage device.
* **Do-It:** A record of an assignment, including the class in which it is assigned, a description of the assignment, its completion priority, its due date, and the time before the due date that it will send a notification.
* **iOS:** The operating system that runs on Apple’s iPhone and iPad devices.
* **JSON:** stands for JavaScript Object Notation. It is a lightweight data-interchange format that is easy for humans to read and write and for machines to parse and generate.
* **Notification:** An pushed alert from the app reminding the user of the upcoming Do-It due date.
* **Swift:** A programming language created by Apple for building apps for iOS, Mac, Apple TV, and Apple Watch.
* **Swift Standard Library:** A package defining the base layer of functionality for writing Swift programs.
* .**doit**: A file containing sufficient information to recreate a Do-It, stored as JSON with a custom file extension.

**DFD Diagrams Level 1**

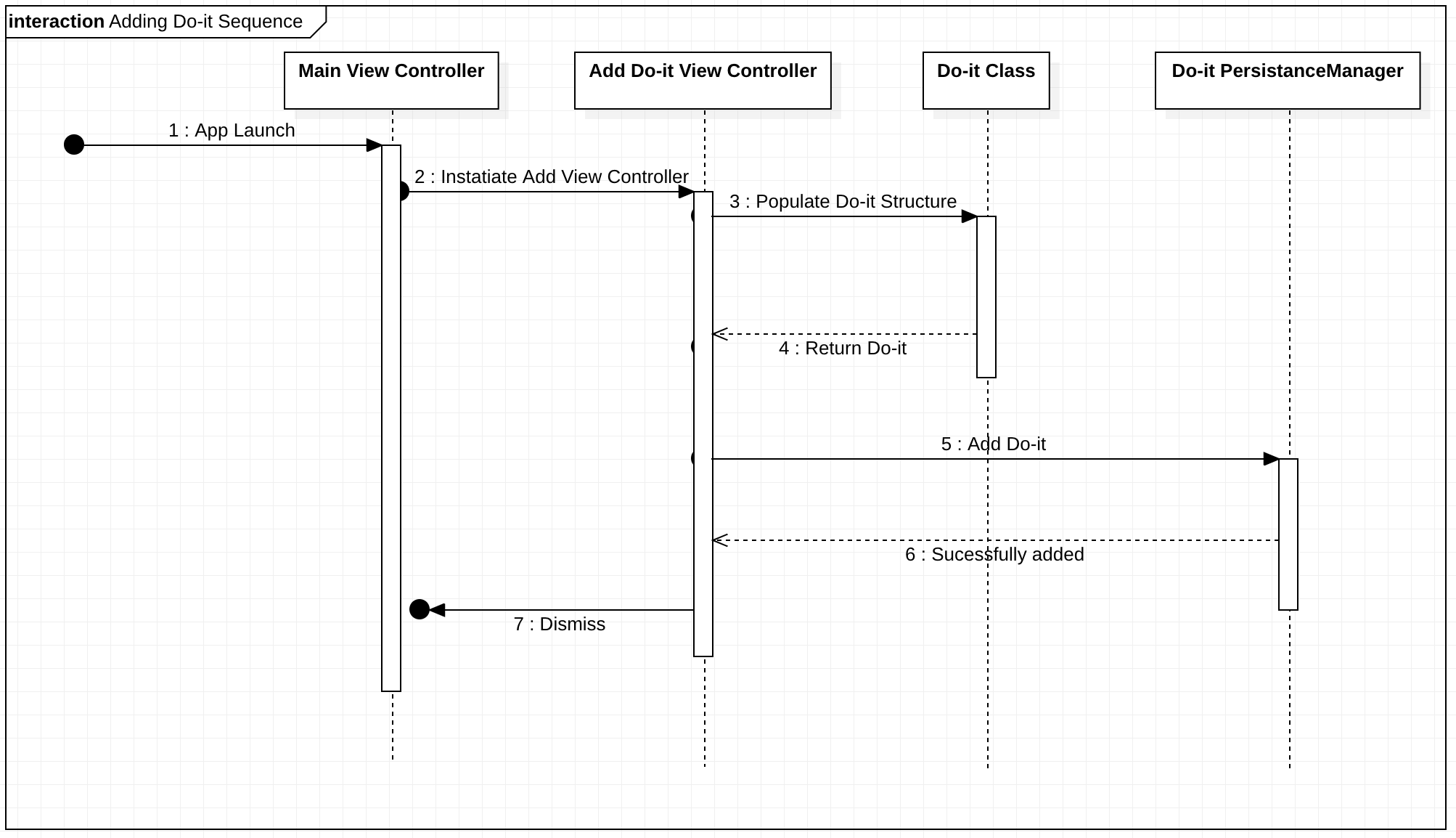


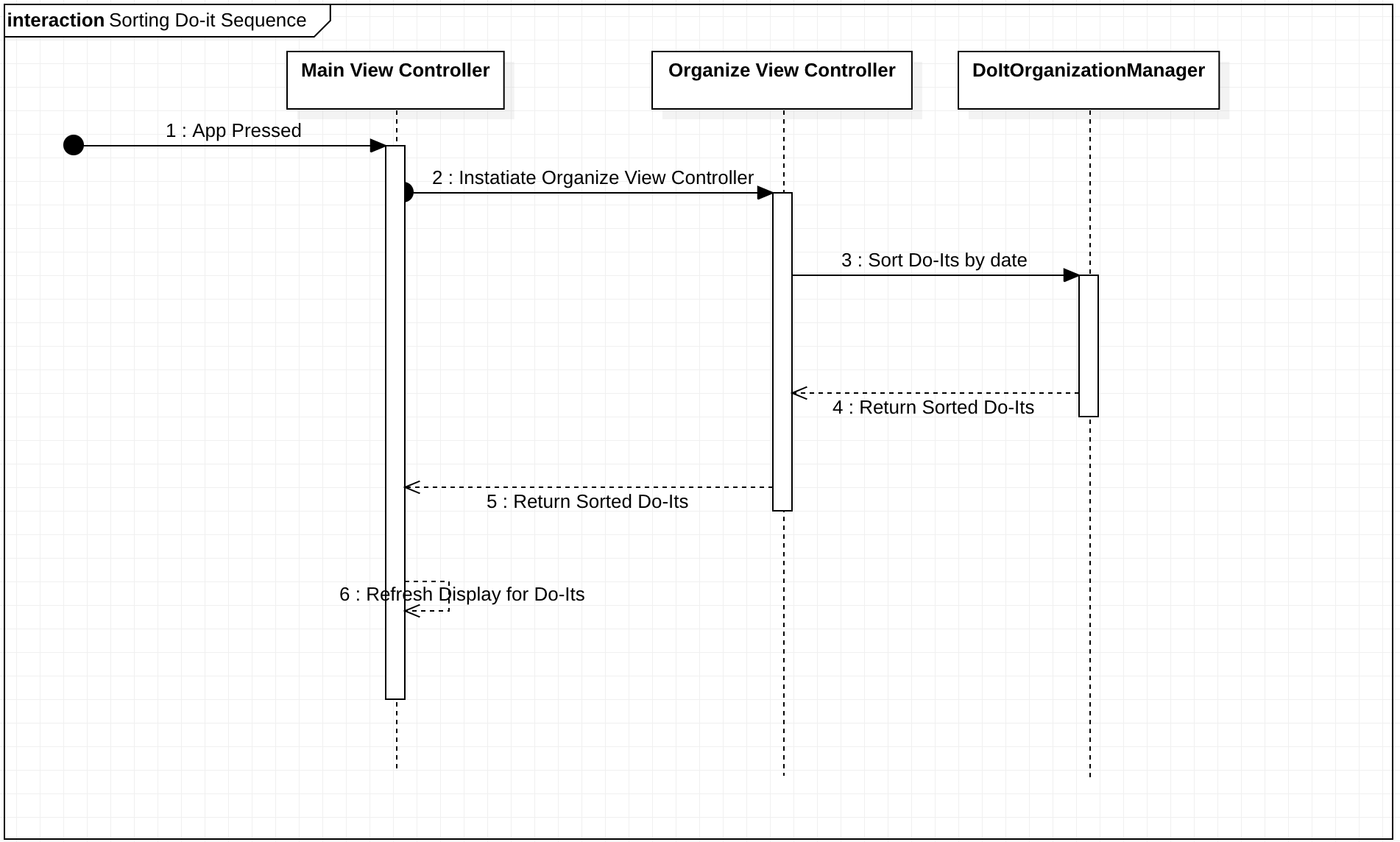
**Decision Trees**

****

****

**Sequence Diagrams**





**User Interface**

4 design criteria

Aesthetic and minimalist design - our app has a minimal color palette and few buttons

Recognition not recall - the user can recognize common iOS buttons like “edit” and “new” to help them edit and create Do-Its.

Control and Freedom - the user has the ability to cancel any action and return to the previous state

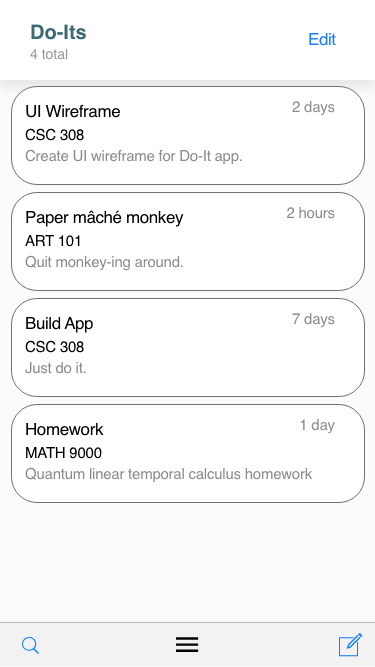
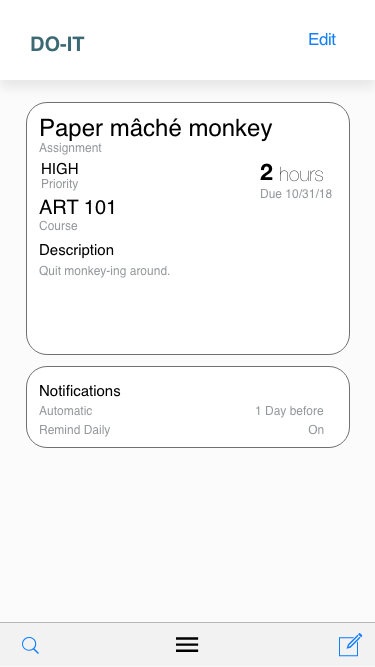
Visibility of System status - each view is clearly labelled so user knows where they are in the app

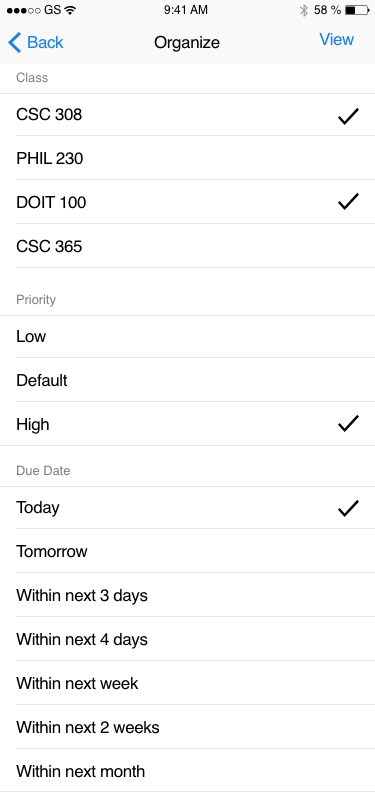
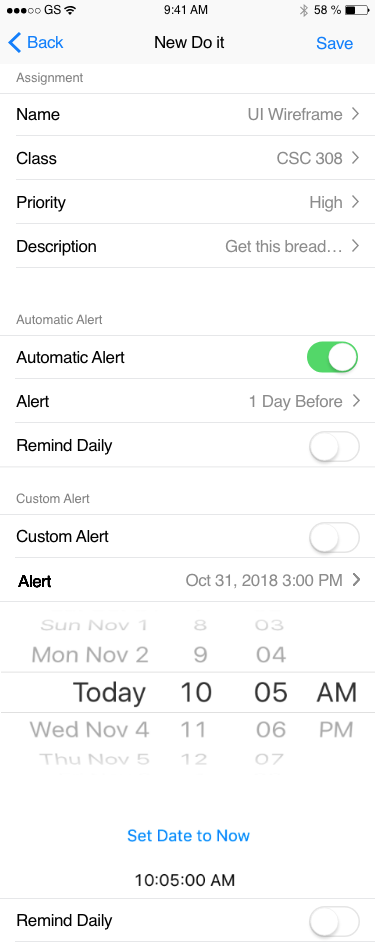
3 design patterns

Continuous scrolling - used to view any number of Do-Its

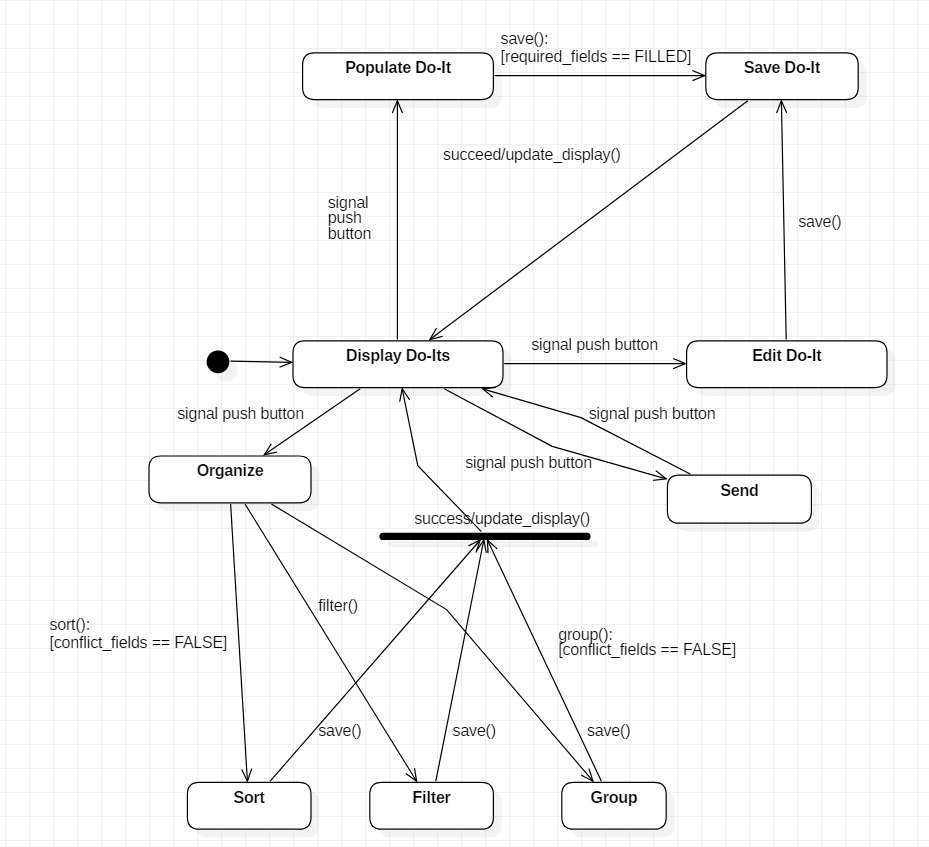
Fill in the Blanks - used for creating a Do-It

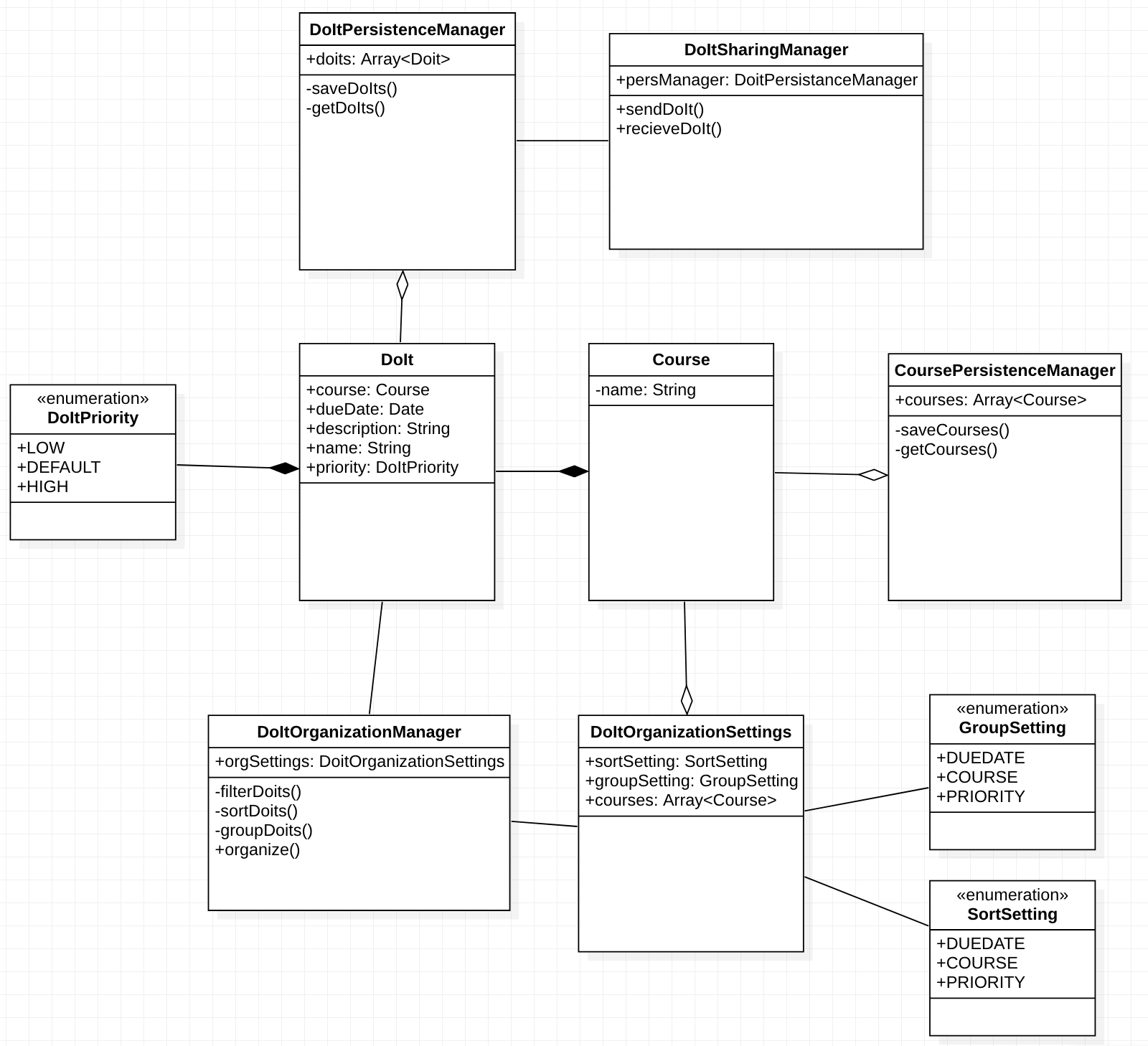
Wizard - to help user understand app the first time the app is launched

****

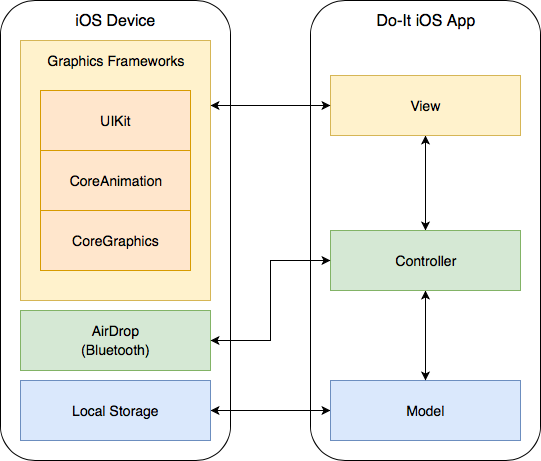
****

**State Chart Diagram**



**Class Diagram**

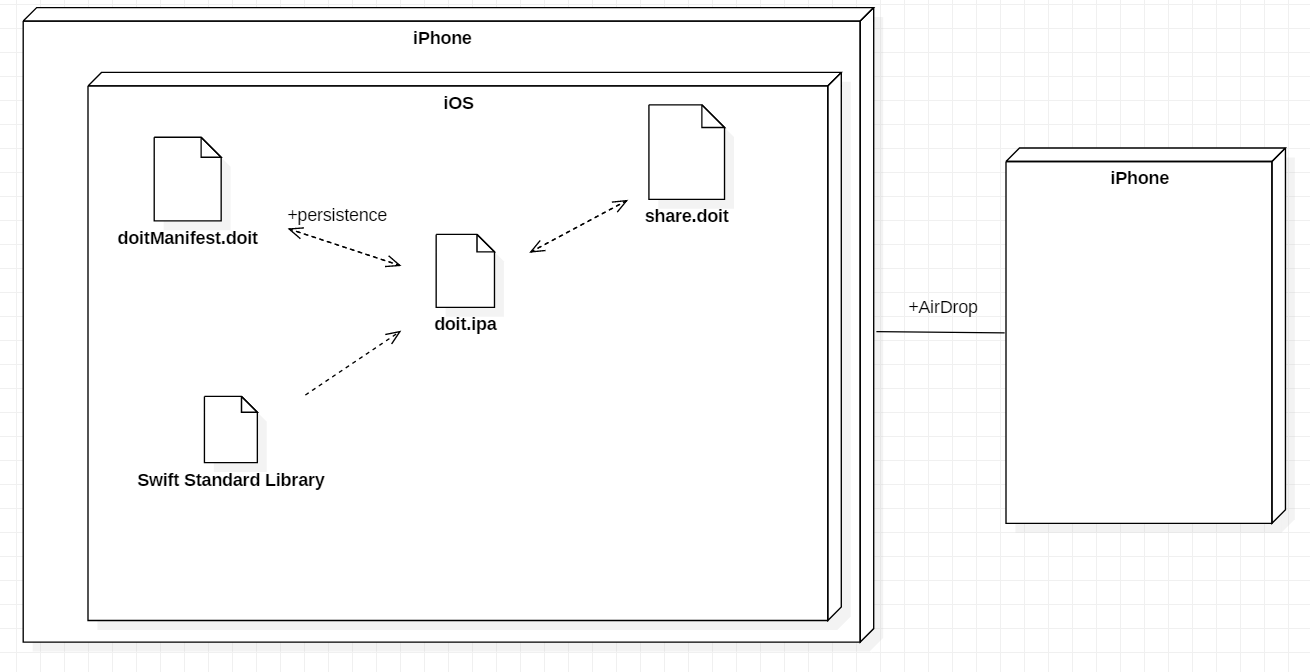
**Software Architecture Diagram**



**Justification**

Apple’s standard graphics framework, UIKit, is designed with MVC in mind. UIKit classes such as UIView and UIViewController serve as bases for the view and controller layers within the application. UIKit has been in development for over a decade and is optimized for performance and security on iOS devices. iOS apps are sandboxed, and files stored locally are encrypted on-device; these provisions ensure the security of our application.

**Deployment Diagram**

****

**Desirable Software Design Characteristics:** 1) **Loose coupling:** We will be following the MVC pattern as required in Apple Development. Classes will be divided into the model (which contains Do-It class and persistence managers), view (which contains customized UI components), and controller (which manages interoperability between the two) layers.

2) **Portability:** Our application will be able to work on many iOS devices with varying screen sizes. Apple provides the frameworks necessary to design dynamic, portable view layouts.

3) **High fan-in (utility class):** The Do-It Persistence Manager will be a commonly used class that reads and writes to the disc. This persistence manager acts as an intermediary for access to Do-It storage.

**Subsystems**

* Persistence Manager (Data Access)
  + The persistence manager will handle all functions for writing and retrieving data from Do-It files
* User interface
  + Using Core Graphics framework to create low-level images in addition to Core Animation framework to animate components. UIKit will create and manage the apps user interface.
* Organizing Do-Its
  + Responsible for filtering, sorting, and grouping Do-Its

**Object-Oriented Design:**

1. Encapsulation: Our Data Manager classes will encapsulate how Do-Its are saved to the the device. Classes using the data managers require no knowledge of how data is actually persisted.
2. Composition: The Organization Manager will use properties of Do-Its to sort and filter them
3. Information Hiding: Each Do-It has a unique identifier modeled by a DoItId class. The DoItId is a wrapper for the underlying data used to uniquely identify each Do-It.

**Class Diagram**