SnapBox

Software Requirements Specification

by

Weston Odom - Mason Volk - Rainer Powell

# Change History

Initial draft - 24 February 2014

Design elements added – 20 March 2014

* Detailed class diagram
* Sequence diagrams
* Communication diagrams
* Added DB\_Interface and FileSaver control classes as a sort of service layer betweenthe UI and the database/filesystem

Table of Contents

Change History 2

Motivation, Scope, and Purpose 2

Goals 3

Key Definitions 3

Features 3

Functional Requirements 3

Non-functional Requirements 4

Hardware Requirements 4

Use Case Scenarios 4

Use Case Diagram 7

Activity Diagrams 8

Class Diagram 10

Sequence Diagrams 11

Communication Diagram 14

# Motivation, Scope, and Purpose

SnapBox automatically names and sorts pictures you take according to your own specifications and guidelines. Users create schedules and optionally define naming schemes and SnapBox handles the rest. The end result is that organizing and retrieving images related to specific events requires very little effort on the part of the user.

# Goals

The primary goal of the project is to create an app that allows users to define a schedule, take pictures, and easily and efficiently browse those images. A stretch goal is to add functionality allowing users to join groups, which would have photos taken during group events synced across all members’ phones.

# Key Definitions

**Schedule**: A Schedule is a collection of Events associated with a particular User or Group.

**Event**: An Event is an entity associated with a User or Group consisting of a timespan, name, and File-naming Scheme.

**File-naming Scheme**: A File-naming Scheme is a format according to which the filename of an image will be created.

**User**: A User is someone who uses the app.

**Group**: A Group is a collection of Users sharing a Schedule and Group Name.

**Group Name**: The name associated with a particular Group.

# Features

The app will contain:

* a calendar-based scheduling system
* basic camera functionality
* a social aspect in the form of Groups
* a file browser for viewing captured images

# Functional Requirements

The app must provide functionality enabling users to:

* define custom File-naming Schemes.
* perform CRUD operations on Schedules and Events.
* search for and manage membership in Groups.
* take pictures with their phone’s camera that will be stored conveniently.
* browse and view pictures that have been saved locally.
* basic image-saving must function with or without an Internet connection.

# Non-functional Requirements

* In-app navigation must be intuitive to users.
* Common tasks must require minimal effort on the part of the user.

# Hardware Requirements

* The user’s phone must have a working camera to take pictures.

# Use Case Scenarios

|  |  |
| --- | --- |
| Use Case: | Taking Picture. |
| Primary Actor: | Photographer. |
| Secondary Actors: | Camera, Clock, Database, File System, Internal. |
| Goal in Context: | To take a picture of a weekly event. |
| Preconditions: | Event timeframe and at least one folder location have been associated in SQLite database. |
| Postconditions: | Picture is stored in folders associated with the current time frame. |
| Trigger: | Photographer decides to take a picture. |
| Scenario: | 1. Interface: Presents Camera screen. 2. Photographer: Presses camera icon. 3. Interface: Presents checkbox list of folder names. 4. Photographer: Selects wanted folder names. 5. Photographer: Presses confirm. |
| Exceptions: | 1. Database has no folder location associated with given time frame: request will be made to create one. 2. Database has only one folder associated with given time frame: Photographer will not be prompted to choose which folder location to store picture in. 3. Folder in database does not exist: Toast is generated with warning indicating picture was not saved. |

SnapBox Use Case Scenario 1. A use case scenario depicting a photographer taking a picture with SnapBox. It should be noted that the presentation of the checkbox list is expected to be a special case. It is reasonable to assume instead that scheduled events will occur at disjoint times.

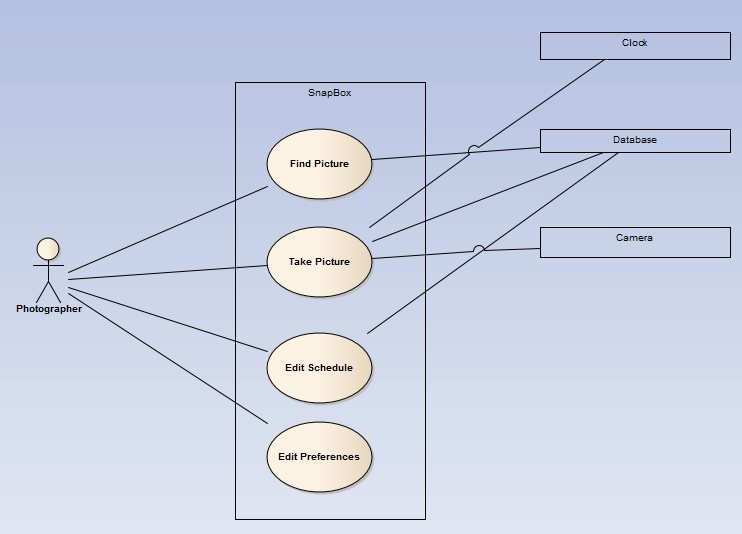
|  |  |
| --- | --- |
| Use Case: | Editing Schedule. |
| Primary Actor: | Photographer. |
| Secondary Actors: | Interface, Internal. |
| Goal in Context: | To alter the photo schedule. |
| Preconditions: | An initial schedule is in existence or has just been created. |
| Postconditions: | Schedule is altered and includes new timeframe and folder location. |
| Trigger: | The photographer decides to make a new folder for a class. |
| Scenario: | 1. Interface: Presents Camera screen. 2. Photographer: Selects “Navigation” icon. 3. Interface: Presents Navigation screen. 4. Photographer: Selects “Schedule” in navigation screen. 5. Interface: Presents schedule customization option screen. 6. Photographer: Selects “Add timeframe” 7. Interface: Presents timeframe frequency selection screen. 8. Photographer: Selects “Weekly”. 9. Interface: Provides timeframe customization screen. 10. Photographer: Highlights all time-spans to include in folder. 11. Photographer: Selects folder location browser. 12. Interface: Presents file browser. 13. Photographer: Chooses folder location. 14. Photographer: Confirms. 15. Photographer: Confirms timeframe addition. |
| Exceptions: | 1. Photographer enters a timeframe that does not make sense: Photographer will be prompted for a new timeframe. 2. Cannot communicate with File System: Toast appears with this indication. 3. Photographer does not select a folder name before confirming a timeframe addition: The photographer will be prompted to enter a valid folder name. |

SnapBox Use Case Scenario 2. A use case scenario depicting a photographer editing the SnapBox schedule. This is a specific instance in which the photographer adds a folder representing a repeating weekly event. Other options might include monthly, or yearly.

|  |  |
| --- | --- |
| Use Case: | Editing Preferences. |
| Primary Actor: | Photographer. |
| Secondary Actors: | Interface, Internal. |
| Goal in Context: | To alter the preferred mode of picture selection. |
| Preconditions: | Initial preferences are in existence or have just been created. |
| Postconditions: | Preferences are altered and old picture selection mode is replaced with new picture selection mode. |
| Trigger: | The photographer decides searching for pictures by name would be faster than using the browser. |
| Scenario: | 1. Interface: Presents Camera screen. 2. Photographer: Selects “Navigation” icon. 3. Interface: Presents Navigation screen. 4. Photographer: Selects “Preferences” in navigation screen. 5. Interface: Presents preference option screen. 6. Photographer: Selects “Picture selection mode”. 7. Interface: Presents picture selection options menu. 8. Photographer: Selects “Search by name”. 9. Photographer: Confirms change. |
| Exceptions: | 1. Photographer attempts to navigate back to the camera screen without confirming changes: The photographer will be asked to choose whether or not to confirm changes. 2. Preferences not successfully altered: Provide toast with warning. |

SnapBox Use Case Scenario 3. A use case scenario depicting a photographer editing the SnapBox preferences. The photographer changes the default picture selection mode, which is the default mode presented for finding files.

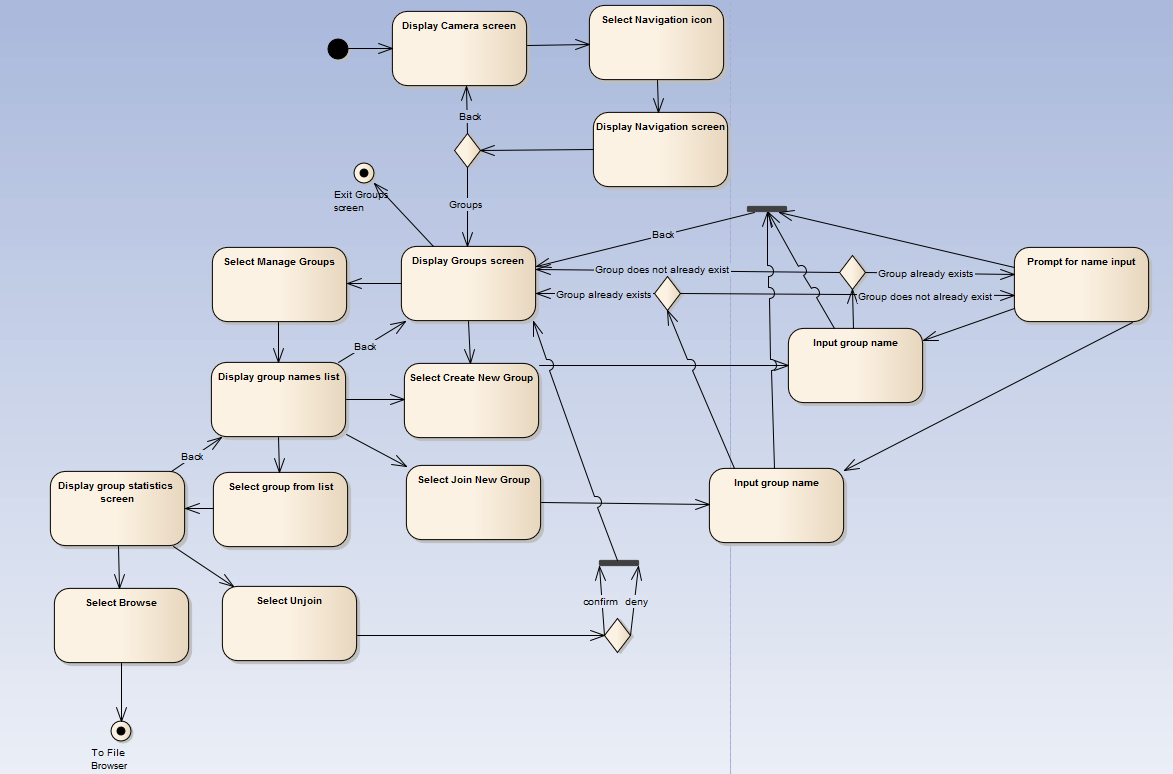
# Use Case Diagram



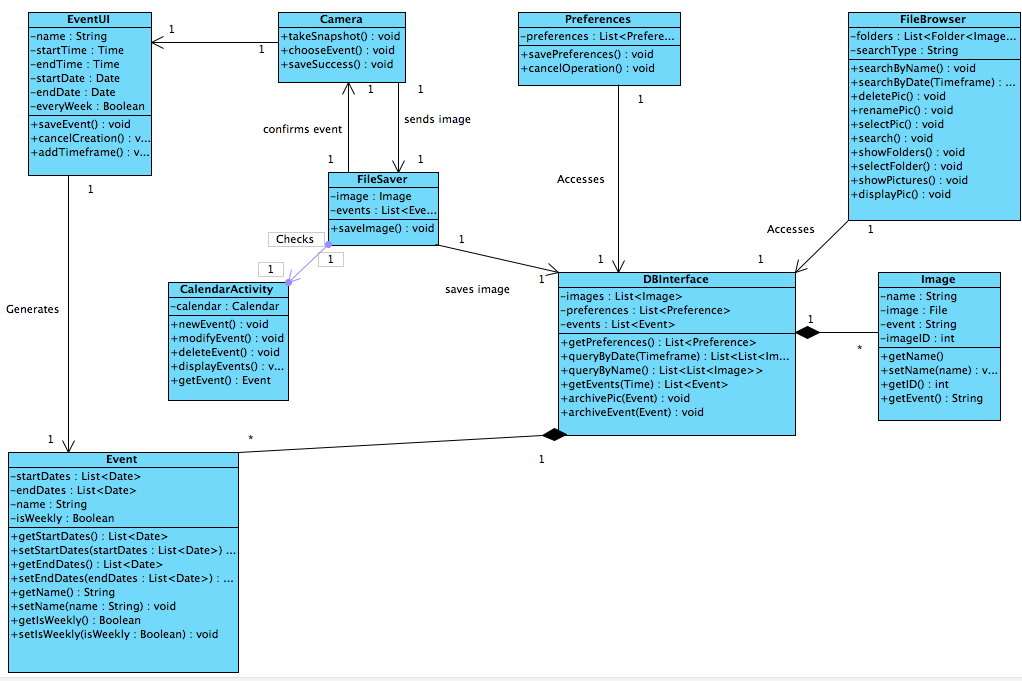
Use Case Diagram 1. Currently, the ability to create and join groups using Google+ has been omitted, as the viability of this feature is still in question until a clear assessment of the Google+ API can be made.

# Activity Diagrams

Activity Diagram 1. This diagram depicts a walkthrough of the picture finding features included in SnapBox. The ability to quickly and effectively locate organized pictures is one of the main goals of SnapBox, so the smoothness of the depicted activities is high priority. It should be noted that this diagram depicts a case in which the folder browser is selected as the default file search method. The calendar or name searches might also be used.

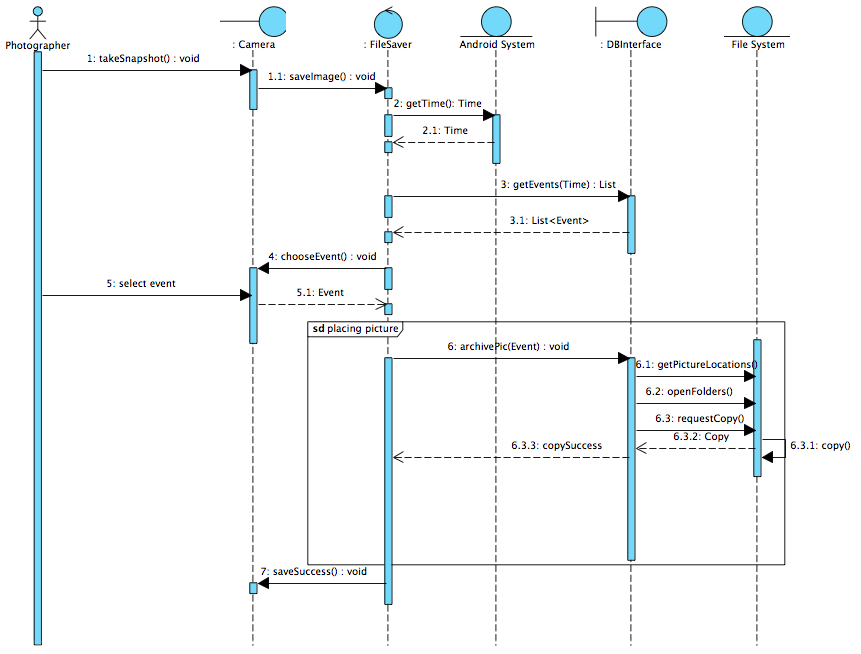
Activity Diagram 2. This diagram depicts a walkthrough of the possible group mechanics of SnapBox. Although the viability of these features, especially taking into account time restrictions, is questionable, the group mechanic is nonetheless important to take into consideration as a possible next step for the application.

# Class Diagram

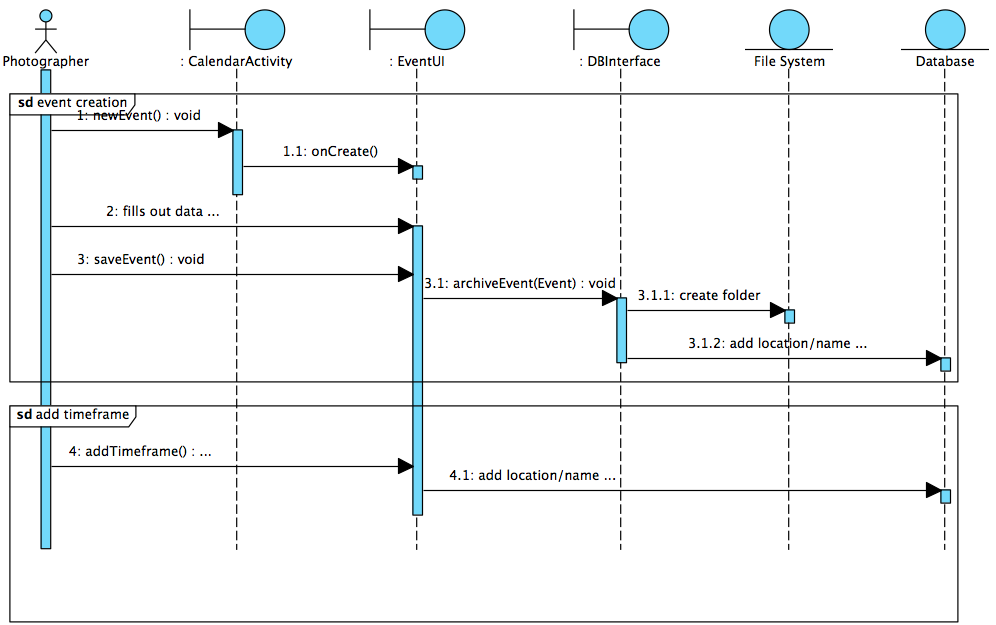


This class diagram shows the classes of the SnapBox mobile application and their relationships with other classes in the system. Interactions with the database are funneled through the DBInterface class, while the information processing of integrating the metadata for the photos is handled in the FileSaver class. Events are generated through the EventUI class, which is generally accessed from CalendarActivity. Images are accessed through the FileBrowser, which in turn accesses the database through the DBInterface class.

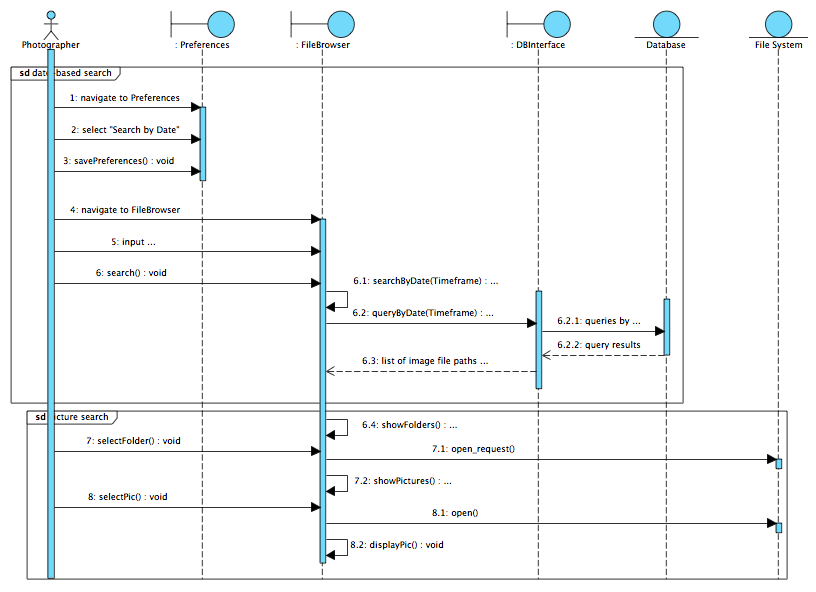
# Sequence Diagrams



A sequence diagram for taking a picture. Internal represents the go-between classes that access the clock and database. The query performed is assumed to retrieve folder locations corresponding to a given time. The placing picture sequence covers the interaction between Internal and the File System, when moving the picture to the new location specified in the database.

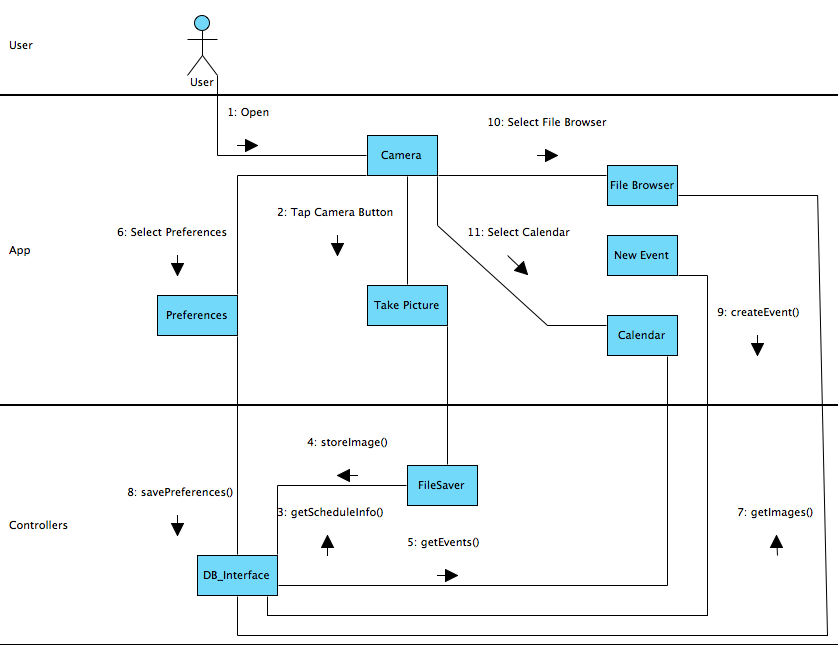


A sequence diagram for creating a new event. The diagram assumes that the Photographer wishes to create a new timeframe for the event immediately after creating the event. The event creation and the addition of corresponding timeframes have been separated so as to make the way in which timeframes are added uniform and situation independent.



A sequence diagram for searching pictures. The diagram assumes that the Photographer changes to the date-based search option before searching, allowing for interaction with the database to be shown. The entire search consists of two separate subsequences; the first involves specifying a search type and locating the proper folder, and the second involves a more granular search in which a specific picture is retrieved from the folder to display.

# Communication Diagram



This shows the basic flow of information through the app. The app always begins in the camera activity. From there users can go to the preferences menu and edit them, or to the file browser or calendar. Each of these activities communicates with the database interface class to retrieve whatever information it needs. The FileSaver class also communicates bi-directionally with the DB Interface during photo saving.