Image Management Service

System Architecture Specification

Bohorquez, Juan Bohorquez, Tomas Buch, Camila Chen, Davis
Garcia, Kevin Howie, Roy Luu, Steven Smelser, William Vonk, Kelsey

Contents

List of Figures

List of Tables

1 System Analysis

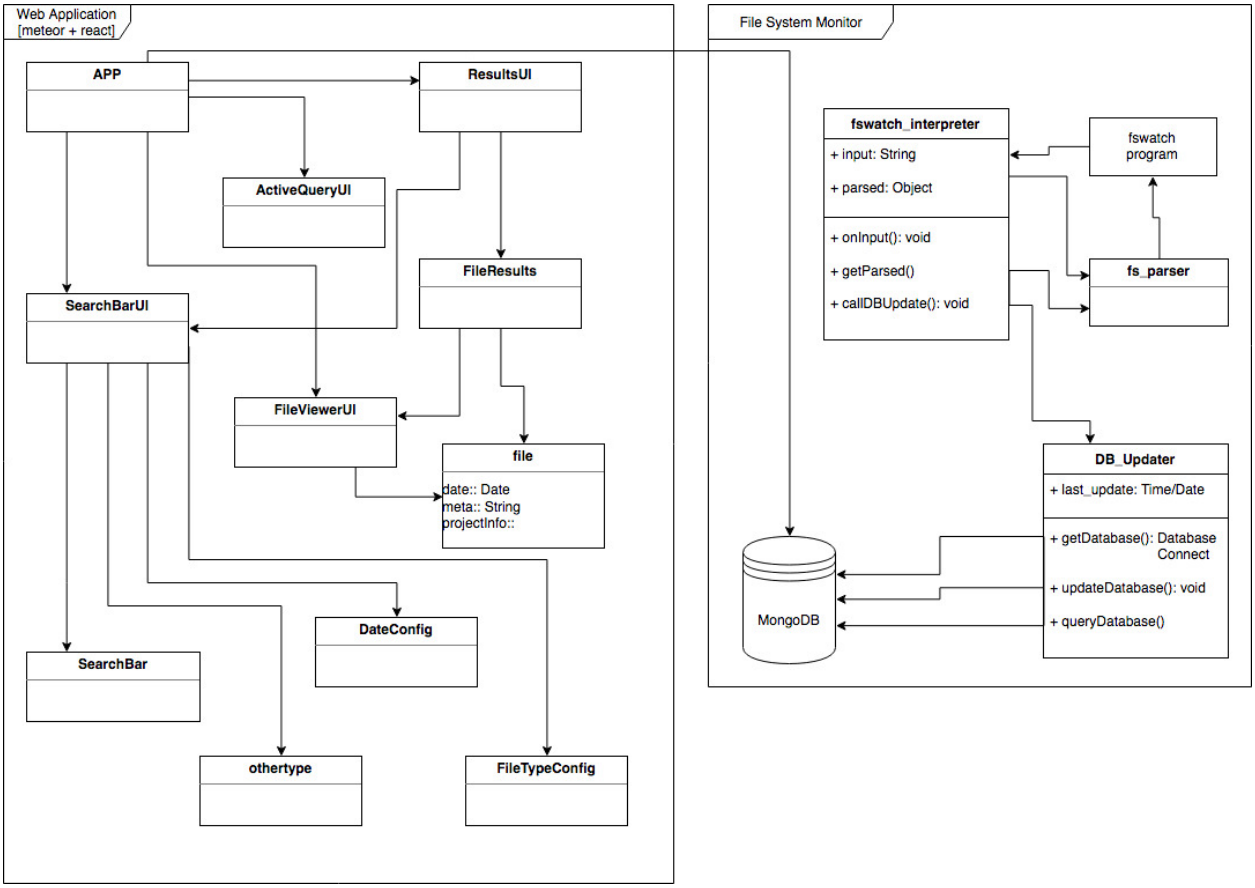
1.1 System Overview

The System will be made up of four main parts: the Database, the user interface, the controller, and a file watcher. The Database will consist of multiple projects (these projects contain pictures, text files and/or videos) that span across various dates and locations. The user interface gives the user the ability to query trough the database by date, project name, and various other paramiters. The controller will take the parameters the user gives and convert them into somthing the database can use to find the requested files, after the database finds the files the controller then tells the UI what the files are so that it can display them. The file watcher will update the database whenever a file is added or removed.

The type of architecture used for the application is a mix between the MVC (model view controller) and MVVM (model view view-model) styles. The models sit in the MongoDB database; Meteor will serve as the controller, and React will be used for the view.

1.2 System Diagram

Figure 1: System Diagram



1.3 Actor Identification

1.3.1 Human Actors

Human actors will come in two forms: general and admin. General users will only be allowed to query and download images. Administrators, in addition to be able to query and download images, will also be able to upload, remove, and reorganize or relabel images.

1.3.2 Non-human Actors

The application will also provide a computer-accessible endpoint, i.e. an API. This API will provide similar levels of access authentication as the human actor endpoints, meaning computers will be able to access the app as general users and as administrators.

1.4 Design Rationale

1.4.1 Architectural Style

For the web application, we will use a mix between the MVC (model view controller) and MVVM (model view view-model) architectural style patterns. This is because

1.4.2 Design Patterns

Meteor and React.js do not emphasize any one particular design pattern. We imagine we will rely mostly on the follow design patterns:

- ⋆ factory method
- ⋆ singleton pattern
- ⋆ façade pattern

1.4.3 Framework

The web application will run via the `Node.js` JavaScript runtime on the `Meteor` framework. Facebook's `React.js` library will be used on the front-end. No framework will be used for the back-end image processing library.

Relevant Links:

- ⋆ `https://nodejs.org/en/`
- ⋆ `https://www.meteor.com`
- ⋆ `https://facebook.github.io/react/`

2 Functional Design

Figure 2: Search and Download Use Case

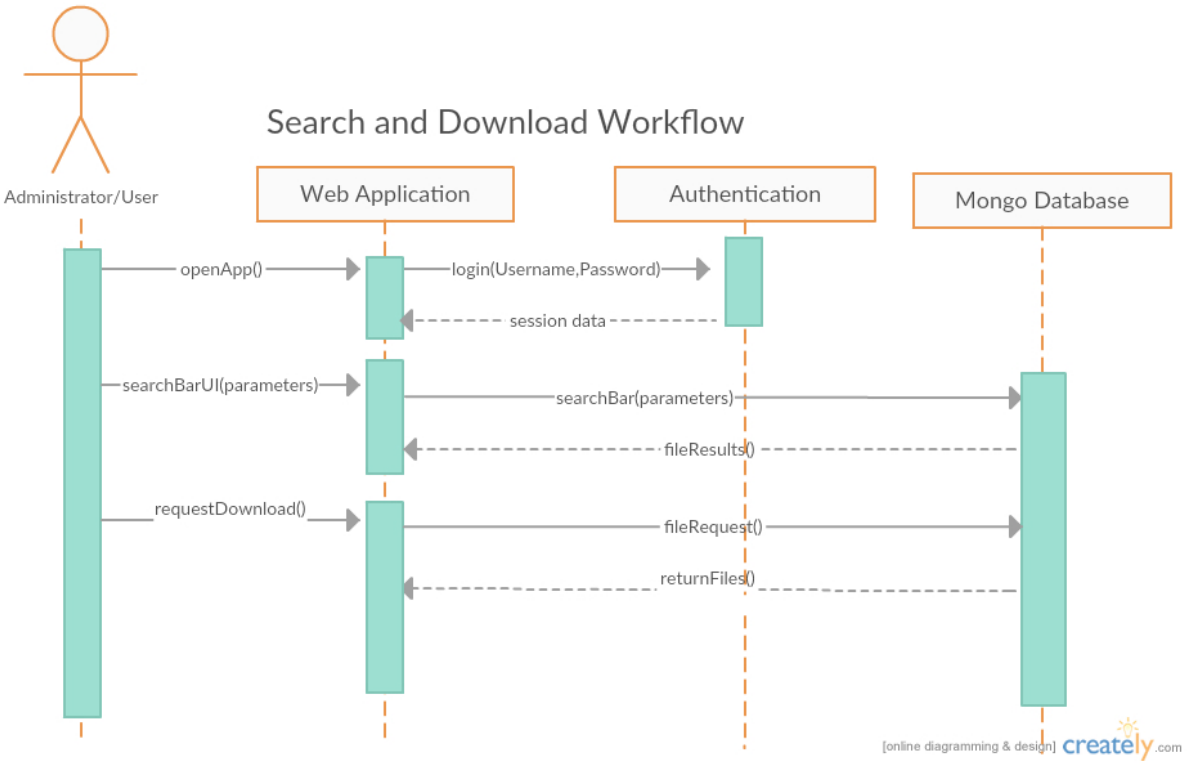


Figure 3: Sequence Diagram

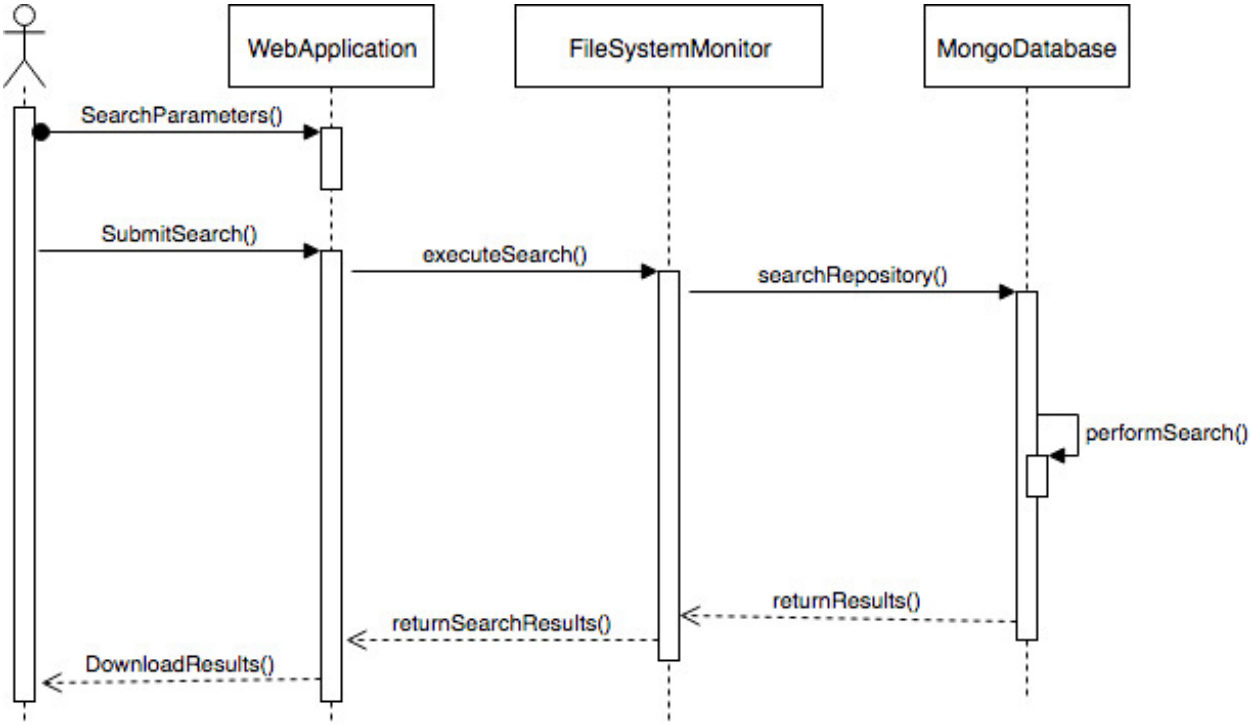


Figure 4: Admin Upload Image Diagram

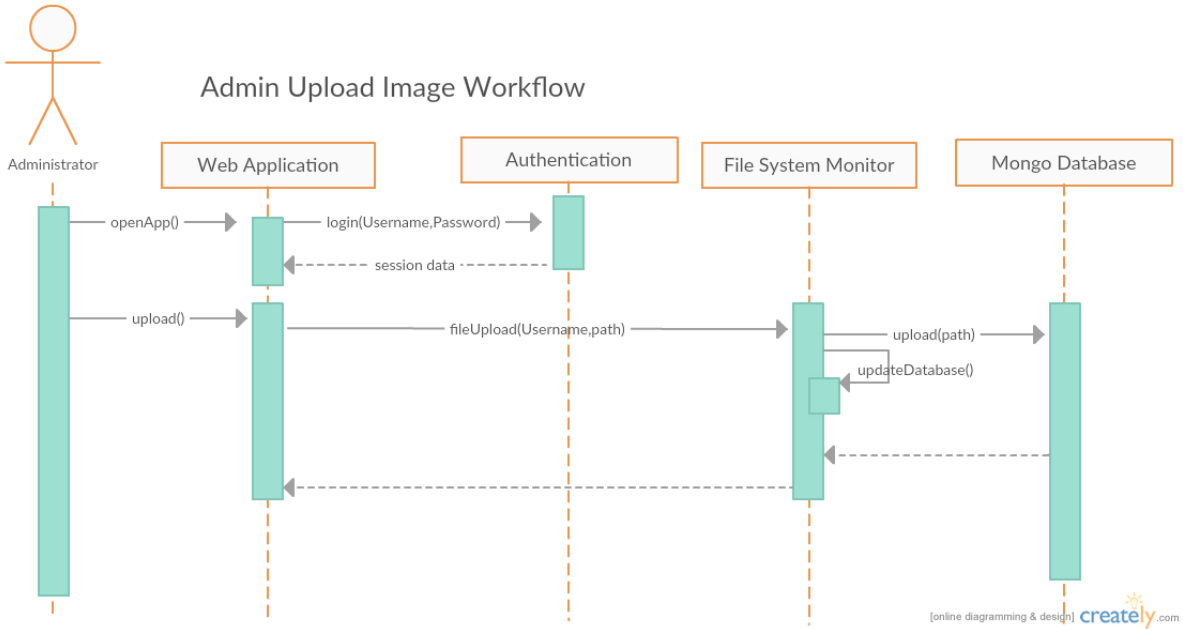


Figure 5: Admin Track Workflow Diagram

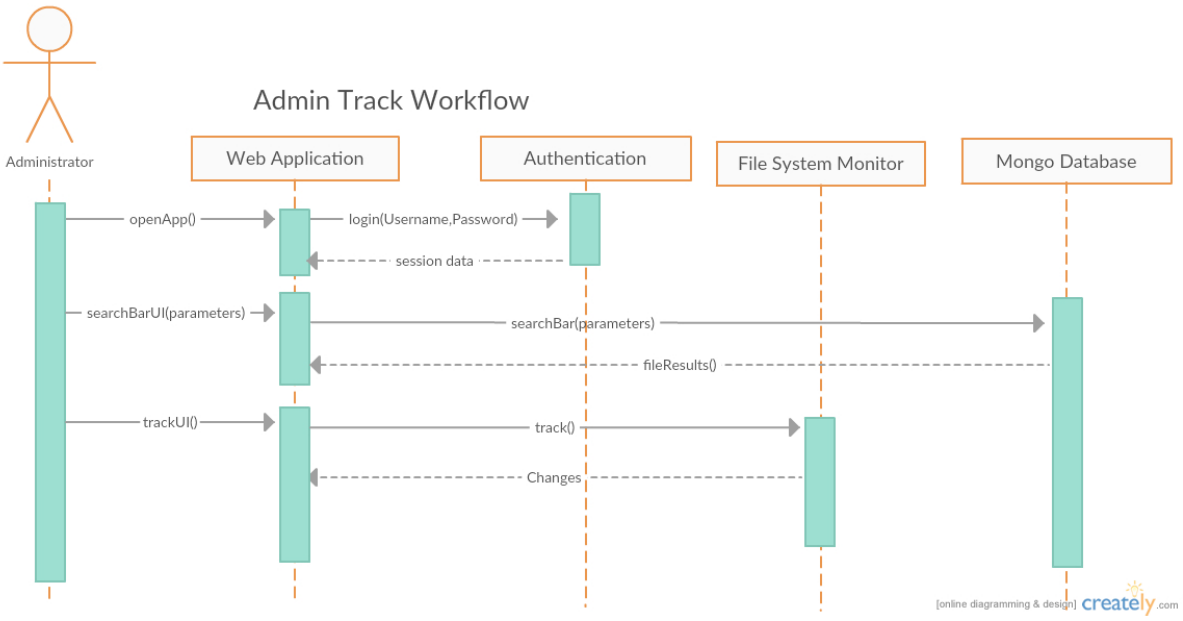
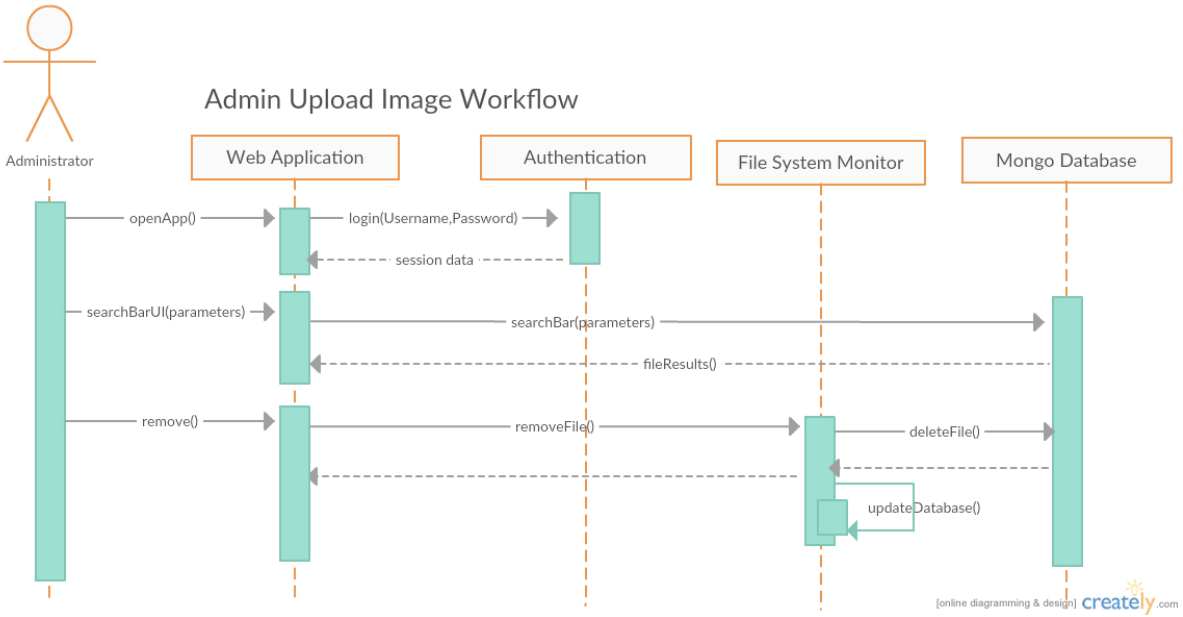


Figure 6: Admin Remove Image Diagram



3 Structural Design

To be added as we move forward with the project.