Image Management Service

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 Requirements

1.1 Functional Requirements

1.2 Non-Functional Requirements

2 System Constraints

2.1 Tool Constraints

2.2 Language Constraints

2.2.1 Web Application Front and Back End

The front-facing UI will be written using the open-source Meteor platform on the Node.js runtime, as the project must be browser-based. Therefore, JavaScript is an absolute must and cannot be avoided.

Relevant Links:

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

Table 1: Web Application Constraints

Title	Web Application Constraints.
Description	Language in which the front-facing web application will be written.
Priority	Necessary: 0.

2.2.2 Back End Image Processing

Besides the back end associated with Meteor, as mentioned in Section 2.2.1, the application will require a back end to process uploaded images and data so as to make said information available to the web application. The languages in which this may be accomplished have no constraints and may evolve—such as in the form of a modular system—as the project progresses.

Table 2: Back End Image Processing Constraints

Title	Back End Image Processing Constraints.
Description	Language in which back-end image processing programs must be written.
Priority	Low: 5.

2.2.3 Database Queries

MongoDB is effectively by Meteor. It is possible to use another type of database via an object relational mapping (ORM) like facebook's GraphQL or to hook it up with another version of noSQL, but *not* recommended.

Relevant Links:

- ⋆ `https://code.facebook.com/projects/250682645321805/graphql/`

Table 3: Database Query Constraints

Title	Database Query Constraints.
Description	Language in which database queries must be written.
Priority	Necessary: 0.

2.3 Platform Constraints

2.4 Hardware Constraints

2.4.1 Computation Constraints

The University of Miami (UM) Center for Computation Science (CCS) will make available to this project its Pegasus computation platform. Pegasus allows for 220 teraflops of computational power and has over 3 petabytes of available storage.

More information can be ascertained via `http://ccs.miami.edu/resources/compute-systems`.

Table 4: Computational Constraints

Title	Computational Constraints.
Description	Amount of computation and processing power available to the application.
Priority	Low: 5.

2.4.2 Storage Constraints

As mentioned in Section 2.4.1, the Pegasus computation platform has over 3 Petabytes of available storage. Thus, this application will have very limited if not entirely nonexistent storage constraints. Indeed, the bulk of the storage needs will be occupied by the image data the application is to process, which has already been accomodated by the Pegasus system.

See `http://ccs.miami.edu/resources/compute-systems` for more information.

Table 5: Storage Constraints

Title	Storage Constraints.
Description	Amount of memory and storage available to the application.
Priority	Low: 5.

2.5 Network Constraints

2.6 Deployment Constraints

2.7 Transition & Support Constraints

2.8 Budget & Schedule Constraints

2.9 Miscellaneous Constraints

3 Requirements Modeling

4 Evolutionary Requirements

4.1 Functional Requirements

4.2 Non-Functional Requirements