Image Management Service

Software Requirements 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 Requirements

1.1 Functional Requirements

1.1.1 Search Requirements

Table 1: Search Requirements

Title	Search Functional Requirements
Description	Database must be able to search through a large data set consisting of various file types.
Source Scenario	FR1.
Priority	Necessary: 0.
Preconditions	Database must be properly sorted in accordance to the pre-existing data structure. Files must be properly tagged to match their specified collection and search information.
Postconditions	Application will search through database using array of parameters and will return all relevant files.
Use Case Diagram	

1.1.2 File Requirements

Table 2: File Requirements

Title	File Processing Functional Requirements
Description	Ability to accept unknown file types.
Source Scenario	FR2.
Priority	Necessary: 0.
Preconditions	None.
Postconditions	Application will be able to search for and display information on a file regardless of the filetype.
Use Case Diagram	

1.1.3 UI Requirements

Table 3: Graphical UI Requirement

Title	Graphical UI Functional Requirement
Description	Application must use a graphical UI for searching and displaying files.
Source Scenario	FR3.
Priority	Necessary: 0.
Preconditions	Working back-end search functionality.
Postconditions	Application will be able to navigate through the database displaying search results, file information, etc.
Use Case Diagram	

1.1.4 File Interaction Requirements

Table 4: File Interaction Requirement

Title	Graphical UI Interaction Requirement
Description	Application must be able to view details about and download collections and files.
Source Scenario	FR4.
Priority	Necessary: 0.
Preconditions	Working back-end search functionality, Graphical UI.
Postconditions	Application will be able to navigate through the database displaying search results, file information, and be able to interact with and download the displayed files and collections.
Use Case Diagram	

1.1.5 API Requirements

Table 5: API Requirement

Title	API Requirement
Description	Application must have an API.
Source Scenario	FR5.
Priority	Necessary: 0.
Preconditions	Application functions.
Postconditions	Seperate applciation functions will be able to work together elegantly and cleanly.
Use Case Diagram	

1.1.6 Store Data Requirements

Table 6: Store Data Requirement

Title	Store Data Requirement
Description	Save unique data that will be used to query images.
Source Scenario	n/a
Priority	Necessary: 0.
Preconditions	Have image that needs to be saved.
Postconditions	Process data to receive information.
Use Case Diagram	

1.1.7 Show Images in Gallery Requirements

Table 7: Show Images in Gallery Requirement

Title	Show Images in Gallery Requirement
Description	You should be able to look through photos inside of a specific gallery.
Source Scenario	n/a
Priority	Necessary: 0.
Preconditions	See all galleries and click on a gallery.
Postconditions	Show all images in gallery.
Use Case Diagram	

1.1.8 Download Requirements

Table 8: Download Requirement

Title	Download Requirement
Description	Users should be able to select a photo or file and then download it.
Source Scenario	n/a
Priority	Necessary: 0.
Preconditions	Select a photo/file to download.
Postconditions	Copy of photo/file is saved to desktop.
Use Case Diagram	

1.2 Non-Functional Requirements

1.2.1 User-friendly Interface

The front-facing web application should have a user-friendly interface. It should be simple, easy to use, ergonomic, and so on. Users should be able to focus on working—not fiddling with the UI—while searching and querying the database.

Table 9: User-friendly Interface

Title	User-friendly User Interface.
Source Scenario	None.
Description	User interface should be easy to use.
Priority	Medium: 3.
Applicable FRs	None.

UI should ideally be *optimistic* and *reactive*. Optimistic refers to performing changes locally—that is, assuming a permissible action was performed—before notifying the server. If the action should not have happened, it should be rolled back. Reactive refers to how the interface is built, i.e. in the style of React.js.

This should not be difficult to accomplish, as Meteor and React.js easily allow for an optimistic and a reactive UI, respectively.

Relevant Links:

⋆ `https://www.meteor.com`

⋆ `https://facebook.github.io/react/`

Table 10: Optimistic User Interface

Title	Optimistic User Interface.
Description	User interface should be optimistic.
Priority	Necessary: 0.

Table 11: Reactive User Interface

Title	Reactive User Interface.
Description	User interface should be reactive.
Priority	Necessary: 0.

1.2.2 Scaleable database

Database must be scaleable. As mentioned in Section 2.2.3, MongoDB will be the database of choice. Fortunately, MongoDB is built to scale, so no work needs to be done.

Table 12: Scaleable database

Title	Scaleable database.
Description	Database must scale.
Priority	Low: 5.

1.2.3 Fast Search Time

Queries performed via the API and the front-facing web application need to be blazing fast. This non-functional requirement pairs well with the one mentioned in Section 1.2.1, as an optimistic and reactive UI is also one which is fast.

Table 13: Fast search

Title	Fast search.
Description	Queries and searches should be fast.
Priority	Medium: 3.

1.2.4 System Downtime

System Downtime should only happen during major updates and should take no more than 1 day to complete.

Table 14: System Downtime

Title	System Downtime.
Description	The only downtime that should be experienced would be due to updating the software.
Priority	Medium: 3.

1.2.5 Real Time Updates

Anytime the database is changed, the application must automatically account for the change in real time.

Table 15: Real Time Updates

Title	Real Time Updates.
Description	Any uploaded images should be automatically added to the database and can be queried in real time by any user.
Priority	High: 4.

2 System Constraints

2.1 Tool Constraints

2.1.1 Access Pegasus

Title	Usable on Pegasus Supercomputer
Description	The database makes use of Pegasus thus our program should be able to use it as well.
Priority	High: 1.

2.1.2 Access Data

Title	Access Cloud with database
Description	The database is stored on a cloud thus our program should be able to access the cloud
Priority	High: 1.

2.1.3 Interactivity/Reactivity

Title	Interactivity/Reactivity
Description	Since we are using React, users will be able to automatically get search results in real time.
Priority	High: 1.

2.2 Language Constraints

2.2.1 Web Application Front and Back Ends

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 16: 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 17: 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 required 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 such is *not* recommended.

Relevant Links:

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

Table 18: Database Query Constraints

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

2.3 Platform Constraints

2.3.1 Platform Constraints

Although there are no specific platform constraints the client has specified, it is expected that the application might be compatible with the most common operating systems.

Table 19: Platform Constraints

Title	Platform Constraints.
Description	The operating system that the program/application must be compatible with.
Priority	Low: 5.

2.4 Hardware Constraints

2.4.1 Computational 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 20: 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 21: Storage Constraints

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

2.5 Network Constraints

2.5.1 Data Access

Title	Access Cloud with database
Description	The database is stored on a cloud thus our program should be able to access the cloud.
Priority	High: 1.

2.5.2 Offline Availability

Title	Offline Availability
Description	The program should be able to run both offline and online.
Priority	High: 1

2.6 Deployment Constraints

2.6.1 Upgrading

Title	Upgrades to the System
Description	Periodically, there may be upgrades to the software in order to add more functional and non-functional requirements.
Priority	High: 1.

2.7 Transition & Support Constraints

2.7.1 After the Semester

Title	After the Semester Transition
Description	We must have a working prototype of the Image Management Service by the final exam date for the class. After the semester, we will no longer provide support for the system and another development team must manage any updates.
Priority	High: 1

2.8 Budget & Schedule Constraints

2.8.1 Budget Constraints

This project has no budget, as no available funds have been provided by the client. The completion of this project is a requirement of the software engineering course (`CSC431`).

Table 22: Budget Constraints

Title	Budget Constraints.
Description	The amount of money and funds which are available for the application.
Priority	Low: 5.

2.8.2 Schedule Constraints

The application or a suitable working prototype must be completed before the end of the semester or grading period. Throughout the semester, we will follow the software development life cycle by completing all necessary phases of the SCRUM methodology.

See `https://www.scrumalliance.org/why-scrum` for more information.

Table 23: Schedule Constraints

Title	Schedule Constraints.
Description	Timeline and schedule that must be followed for the application.
Priority	High: 1.

2.9 Miscellaneous Constraints

2.9.1 Font Usage

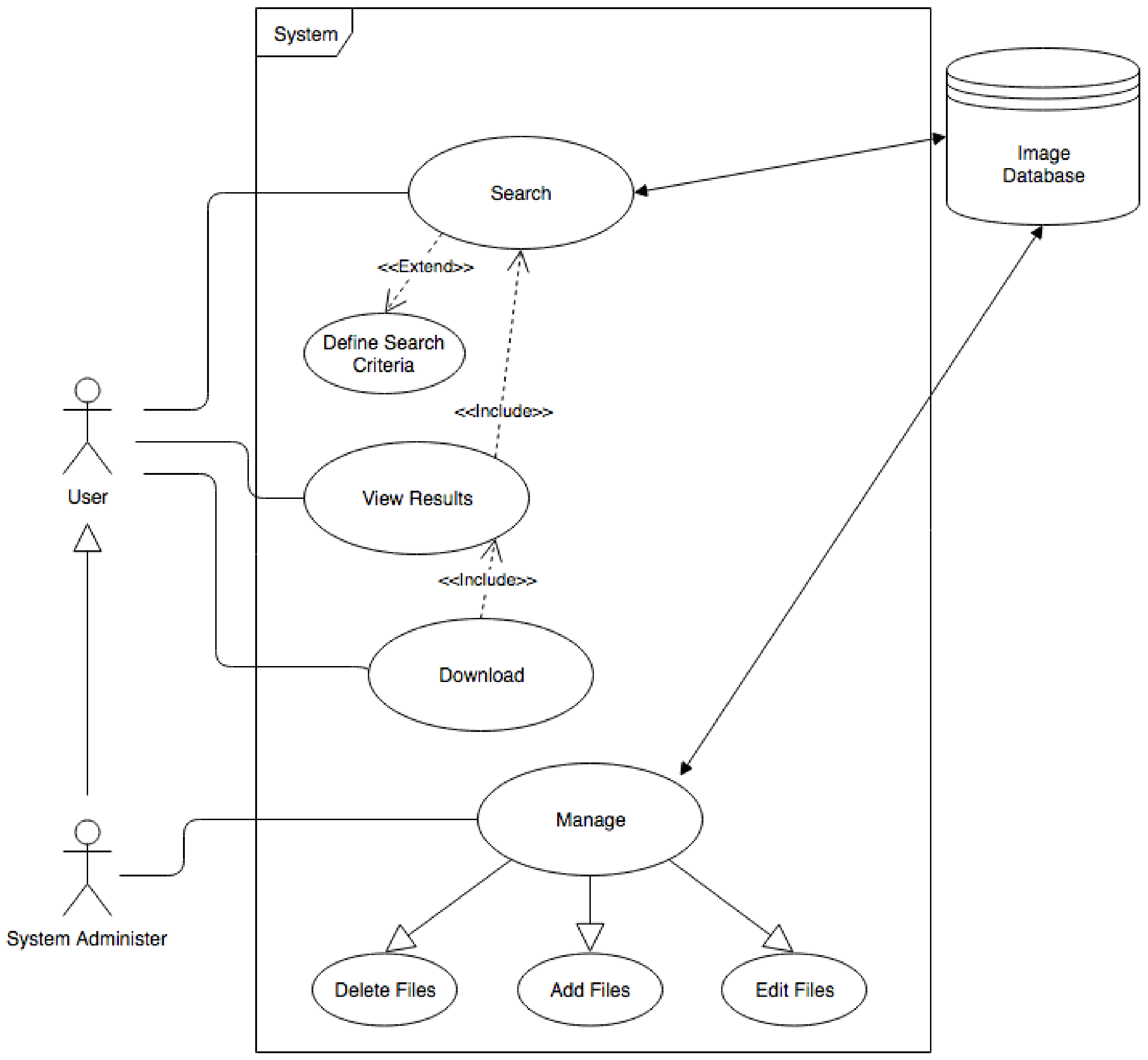
A legible and easily read font should be incorporated into the front-facing web application at all times, so that the application retains a professional look. For examples, `Times New Roman` would be appropriate, but `WingDings` would not.

Table 24: Font Usage Constraints

Title	Font Usage Constraints.
Description	Font that the GUI must use.
Priority	Low: 5.

3 Requirements Modeling

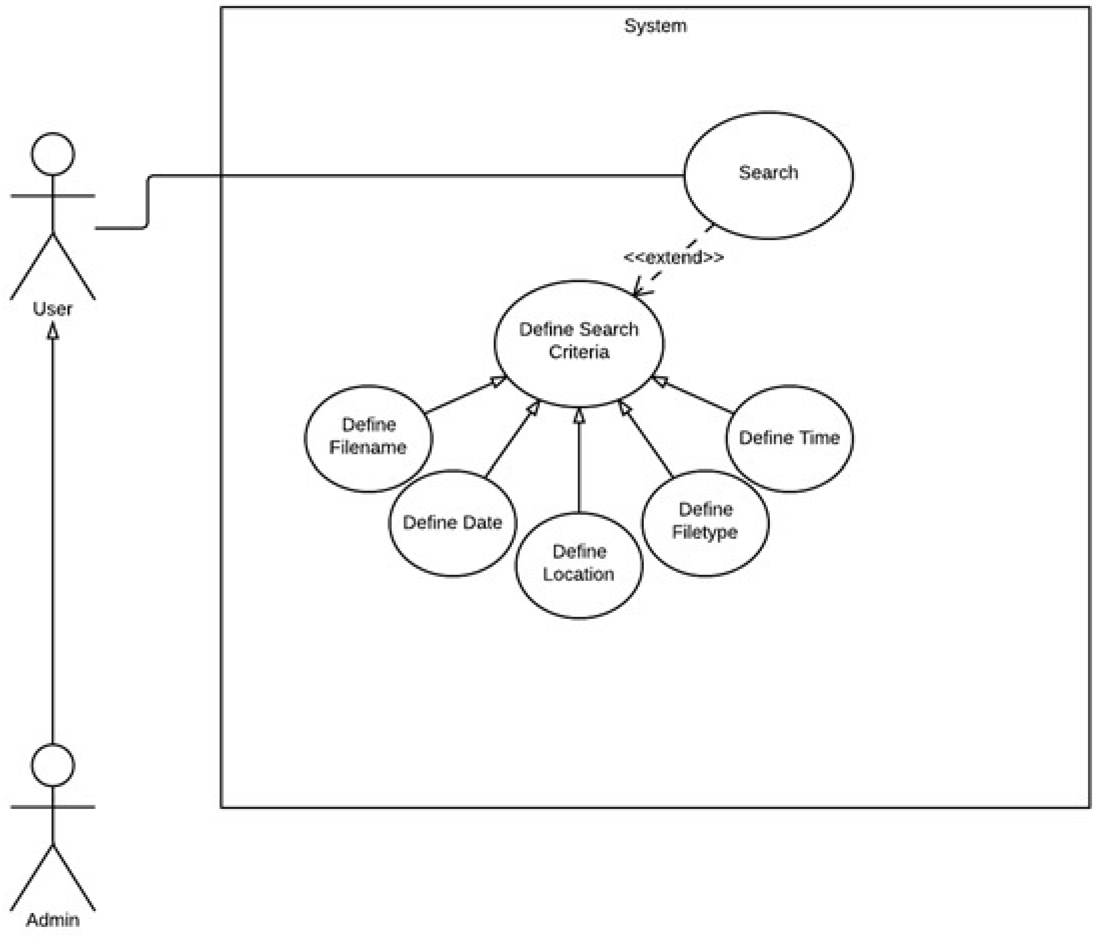
Figure 1: Sample Use Case



3.1 Search Photos

Title	Search Photos in the System
Description	Query server for photos matching certain criteria
Actors	The user and the image server
Trigger	Server is searched and images matching criteria are returned
Pre Condition	Query fields are filled in
Post Condition	Images matching criteria are displayed
Basic Flow	User fills in query fields, query criteria are sent to the server, server is searched using criteria, images matching criteria are returned, returned images are displayed
Exceptions	N/A.

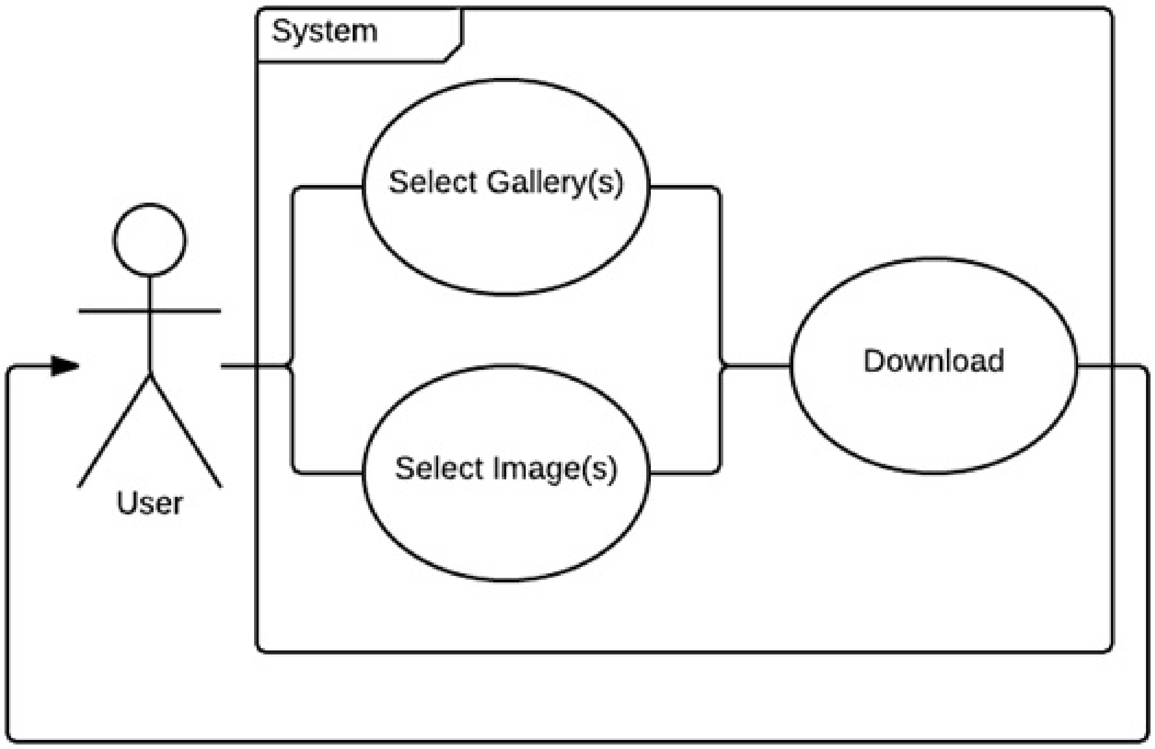
Figure 2: Search Photos Use Case



3.2 Download Files

Title	Download Files in the System
Description	Download selected galleries or images
Actors	The user and the image server
Trigger	Download button is pressed and selection is downloaded
Pre Condition	Pictures or Galleries are found and selected
Post Condition	A file is downloaded into the users computer
Basic Flow	Users select Galleries or pictures -¿ Users click download -¿ file is downloaded.
Exceptions	Files don't exist.

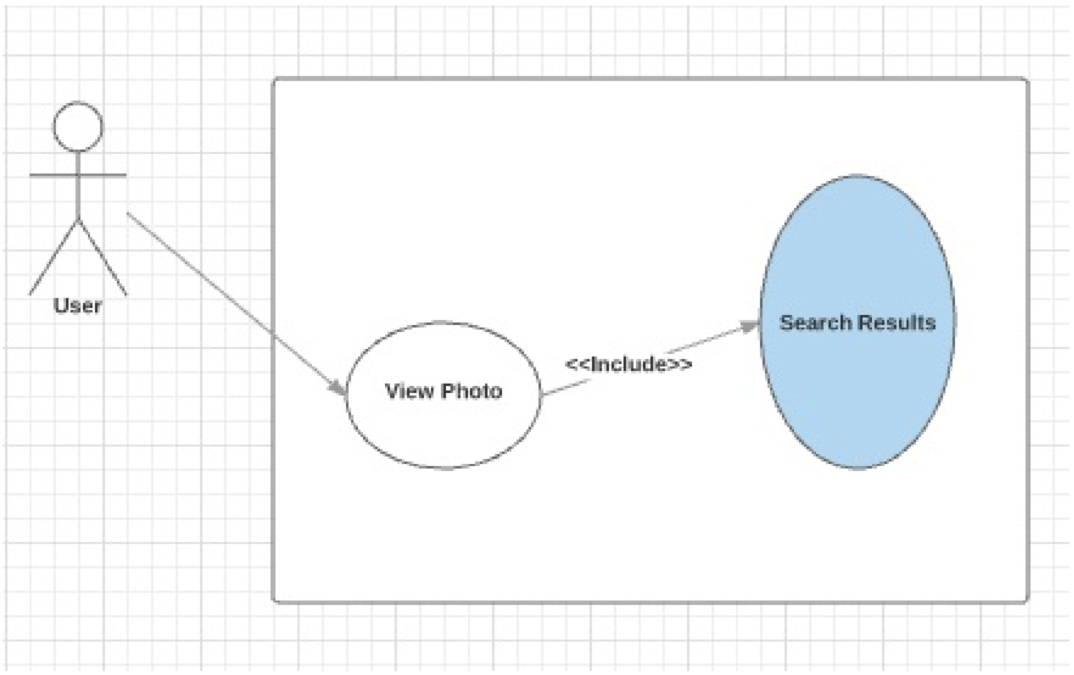
Figure 3: Download Files Use Case



3.3 View Photos

Title	View Photos in the System
Description	View Photo returned from query
Actors	The user
Trigger	Displays in real-time as user types into the search fields.
Pre Condition	Search fields contain input.
Post Condition	Opens path to photo.
Basic Flow	User clicks, returns path to photo to open it.
Exceptions	No matching photos are returned from search.

Figure 4: View Photos Use Case



4 Evolutionary Requirements

4.1 Functional Requirements

4.1.1 Placeholder

Currently, there has been no evolution of functional requirements.

4.2 Non-Functional Requirements

4.2.1 Placeholder

Currently, there has been no evolution of non-functional requirements.