COMP-6905

Nature’s Palette

Assignment 1

Group 2

Shawn Sabraw
Rob Bishop
Asma Javaid
Praveena Pinnika
Samira Saki

Table of Contents

Vision Document

Introduction

For this project, the aim is to design and implement a web application that will allow users to upload, search, and download spectral data that can be used in research and additional projects through useful search tools and data itself.

Problem Statement

Research projects that require quantification of colours usually require thousands of spectral measurements. The issue in this; for researchers, is obtaining all that data. For example, comparative studies in birds have characterized the plumage colours for over 40% of described species of birds. However, less than 5% of that data is publicly available to other researchers. The applications that this data is available through are not designed in a way that makes them easy to use and/or provide the metadata that would allow that information to be used in other studies. Consequently, the possibilities for larger studies are very limited and lots of resources are being wasted in order to obtain (usually duplicating) the data for different research projects.

Stakeholders & Key Interests

Stakeholders	**Key Interests**
Researchers (Professors/Students)	Using the data provided in their projects
Hobbyists/Users	Adding their data to be used in research projects
Establishments	Access to a large database of spectral information for R&D

Users & User-Level Goals

User	Goals
General User (Researchers/Hobbyists)	Login, Upload/edit/search/download spectral data, refine search through meta data/georeferencing/colour range, Upload many files …
Administrator	Login, Upload/edit/search/download/remove spectral data, edit user's access rights, edit parts of the application, adding meta data tags …

Summary of System Features

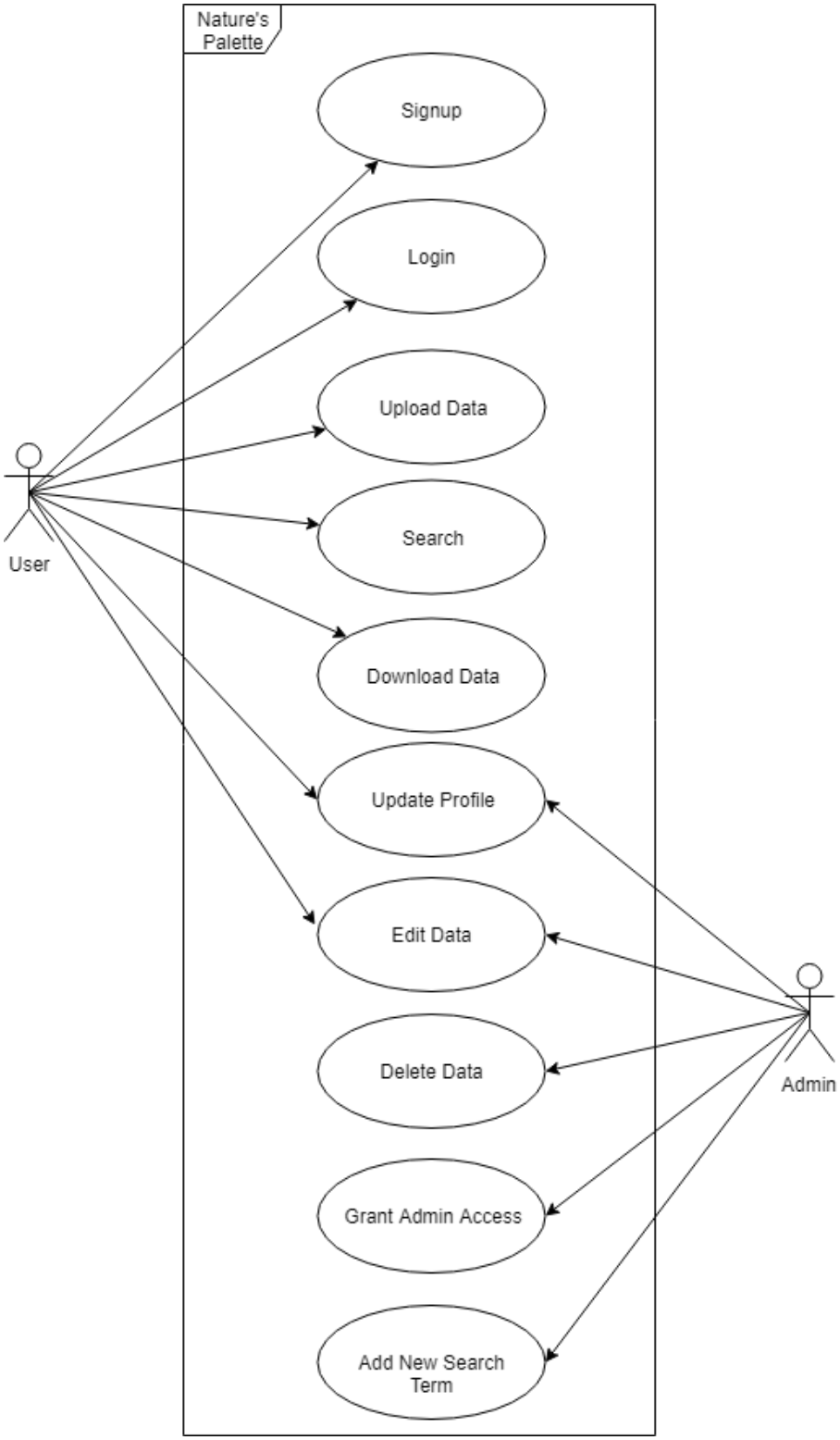
- The application will allow users to upload spectral data with it's meta data tags.
- The application will let users download the data as well as it's meta data.
- The application will let users search through the data using georeferencing, meta data tags, and specifying a colour range.
- The application will allow users to edit their uploads in case of mistakes.

Project Risks

- Utilizing and learning Node.js, MongoDB and other modules as a client -> server model.
- Mass uploading of thousands of files as well as storing them in a way to allow for scaling.
- Using a Python/R package in order to "clean" spectral data that is being uploaded.
- Using a polygon georeferencing tool to search meta data in a database.

Use Cases

System Use Cases



Brief Descriptions

Use case	Description
SignUp	User enters in the required information (username, password, etc.…) to create an account.
Login	User enters in the required information to gain access to their account.
Update Profile	User updates their profile details. Admins are also able to edit their own as well as other user's details.
Upload	User uploads their spectral data accompanied with their meta data.
Search	User searches for spectral data giving metadata as input terms.
Download	User downloads the spectral and metadata provided.
AddNewSearch	Admin adds new search terms for users to search by.
AdvancedSearch	User searches for spectral data by providing multiple filters (regions, colour space, etc.…) given in the advanced search.
ChangeAccess	Admin changes a user's access to ban, unban, or make them an admin.
Edit Data	User (with an account) can edit or delete their own data. Admins can edit and delete any data.

Detailed Descriptions of Use Cases

Use Case: Upload

Use Case Name	**Upload**
Participating Actors	User
Flow of Events	Main 1. User is prompted to upload the specified files. 2. User selects appropriate files to upload and continues. 3. After file upload, the spectral data will be compared to the metadata. 4. The data is confirmed with the metadata and is saved to the repository before displaying a success status message to the user. (**Alt1:** The metadata does not match) **Alt1:** (The metadata does not match) 1. The data and metadata do not match. The data is not saved, and a failure status message is displayed to the user.
Entry Condition	• User clicks upload button
Exit Condition	• User clicks "x" or navigates away from the upload prompt
Quality Requirements	• Upload should be fast for the user; the server should do most of the work.

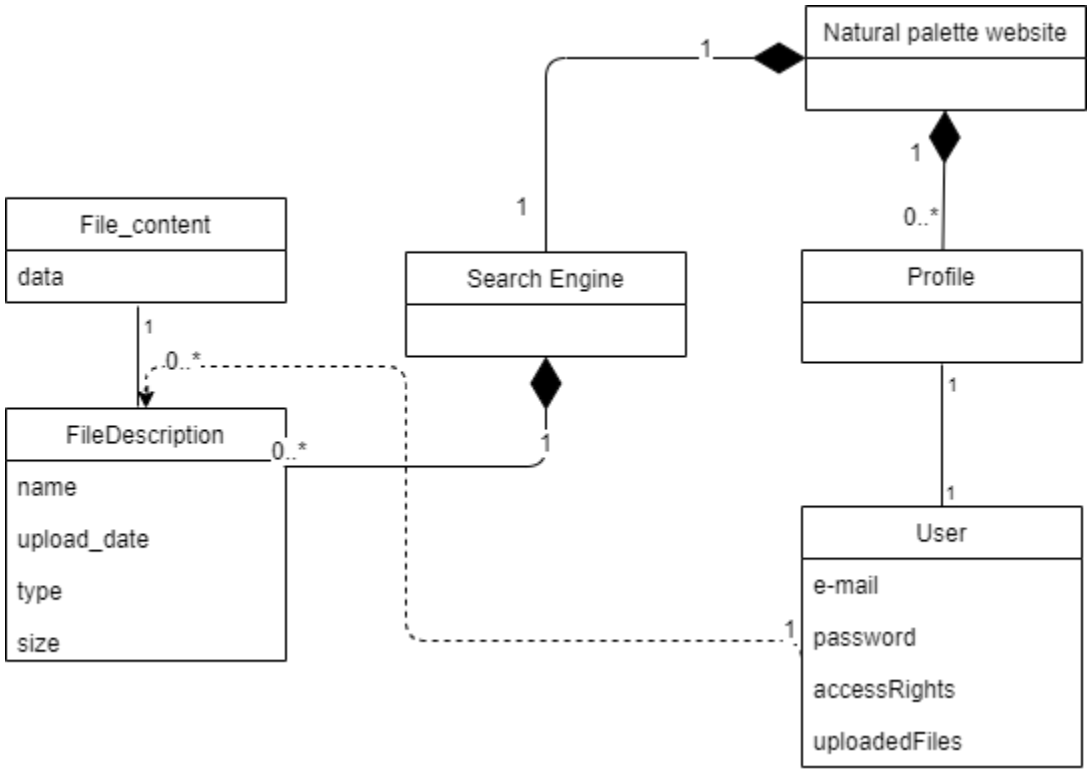
Use Case: Search

Use Case Name	**Search**
Participating Actors	User
Flow of Events	Main 1. The user enters a query. 2. The webpage communicates the query to the server. 3. A series of entries that match the user's query are displayed, each with some information about the entry. (**Alt1:** The query did not match) **Alt1:** (The query did not match) 1. A status message is displayed saying that no data matched the user's query.
Entry Condition	• User clicks on the search field
Exit Condition	• The user navigates away from the page
Quality Requirements	• Search should take place within a reasonable speed (To be determined by client)

Use Case: Download

Use Case Name	**Download**
Participating Actors	User
Flow of Events	Main 1. The user is prompted for the download. 2. The user selects the type of files to download. 3. The server downloads the file data to the user. (**Alt1:** File Is Not Ready Yet) **Alt1:** (File Is Not Ready Yet) 1. The data is not available yet for download.
Entry Condition	• Clicks download button
Exit Condition	• The download finishes or the user navigates away from the page
Quality Requirements	• Download should be available quickly

Domain Model



Entities:

- Users
- FileDescription
- FileData

Boundaries:

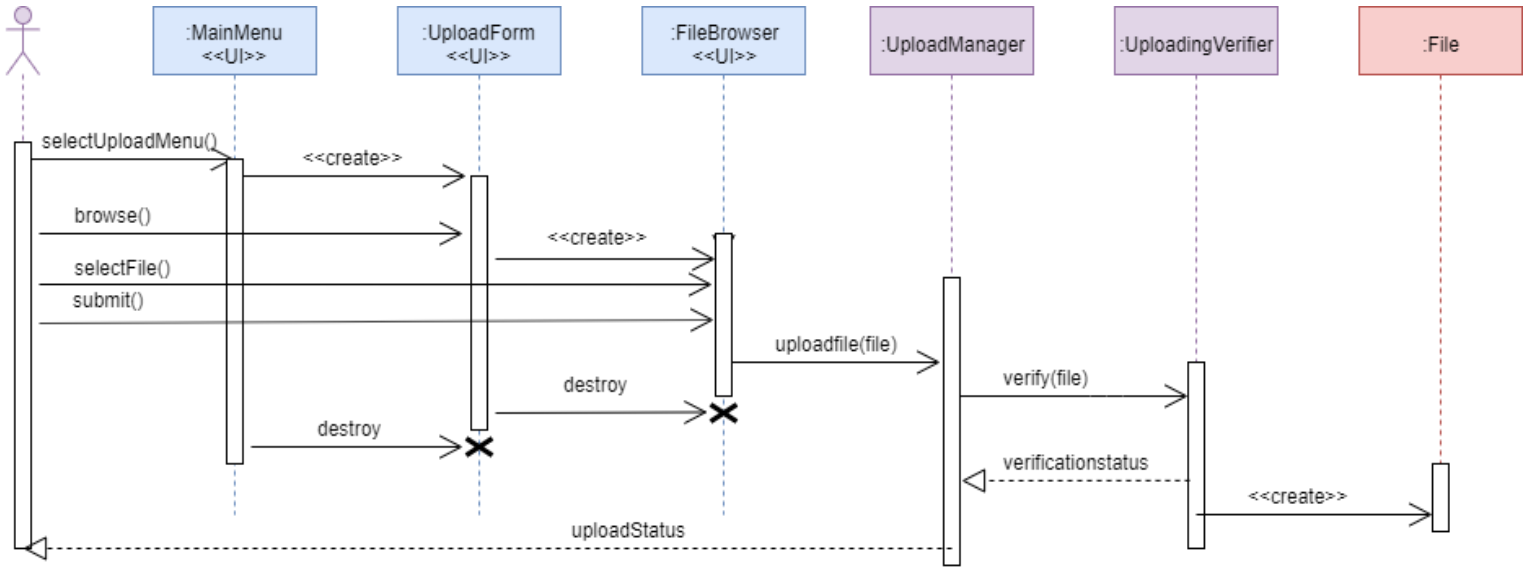
- Nature’s Palette website
- Search Engine
- Profile

Controls:

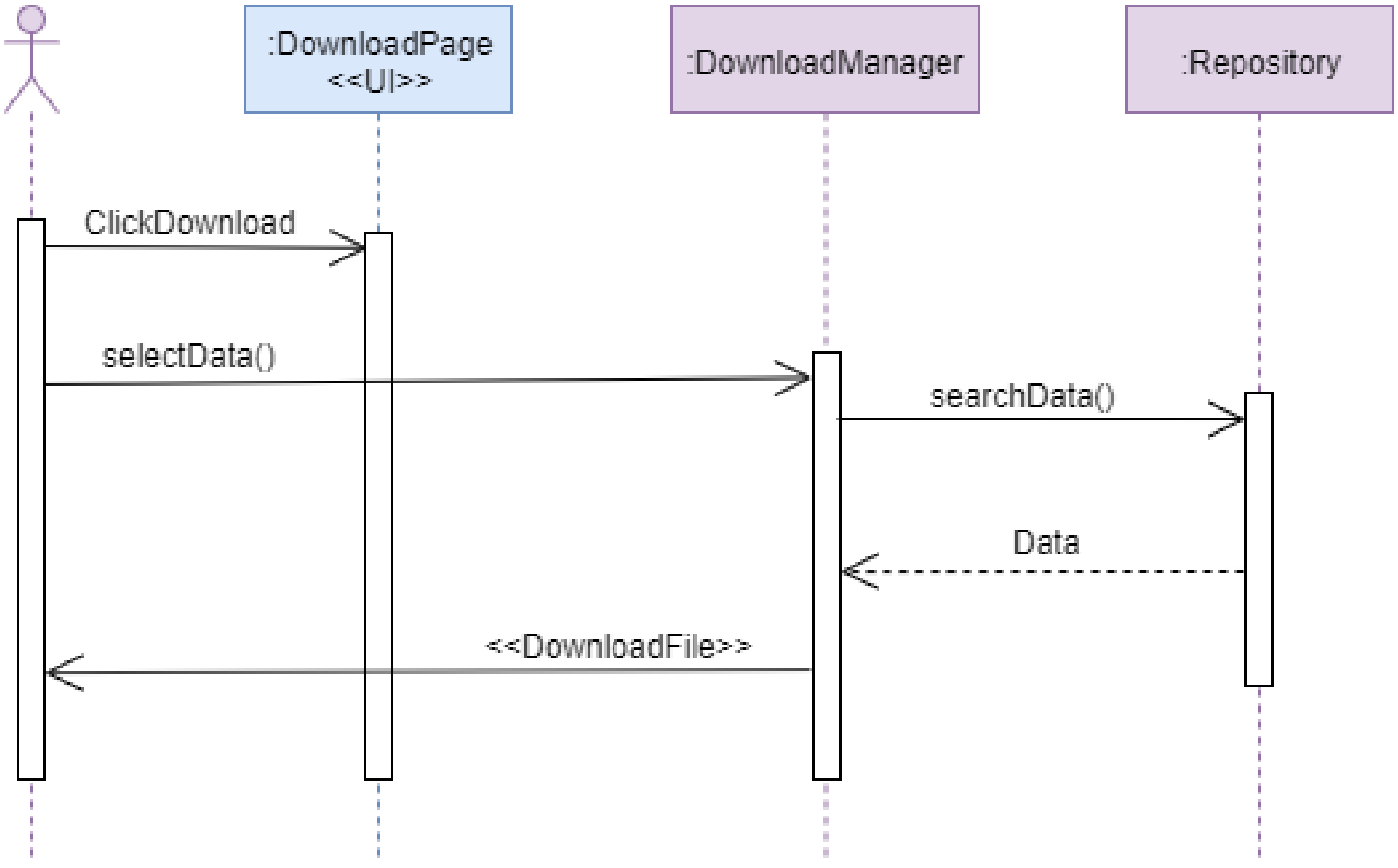
- Upload manager
- Search manager
- Download manager
- Authentication manager
- Profile manager

Sequence Diagrams

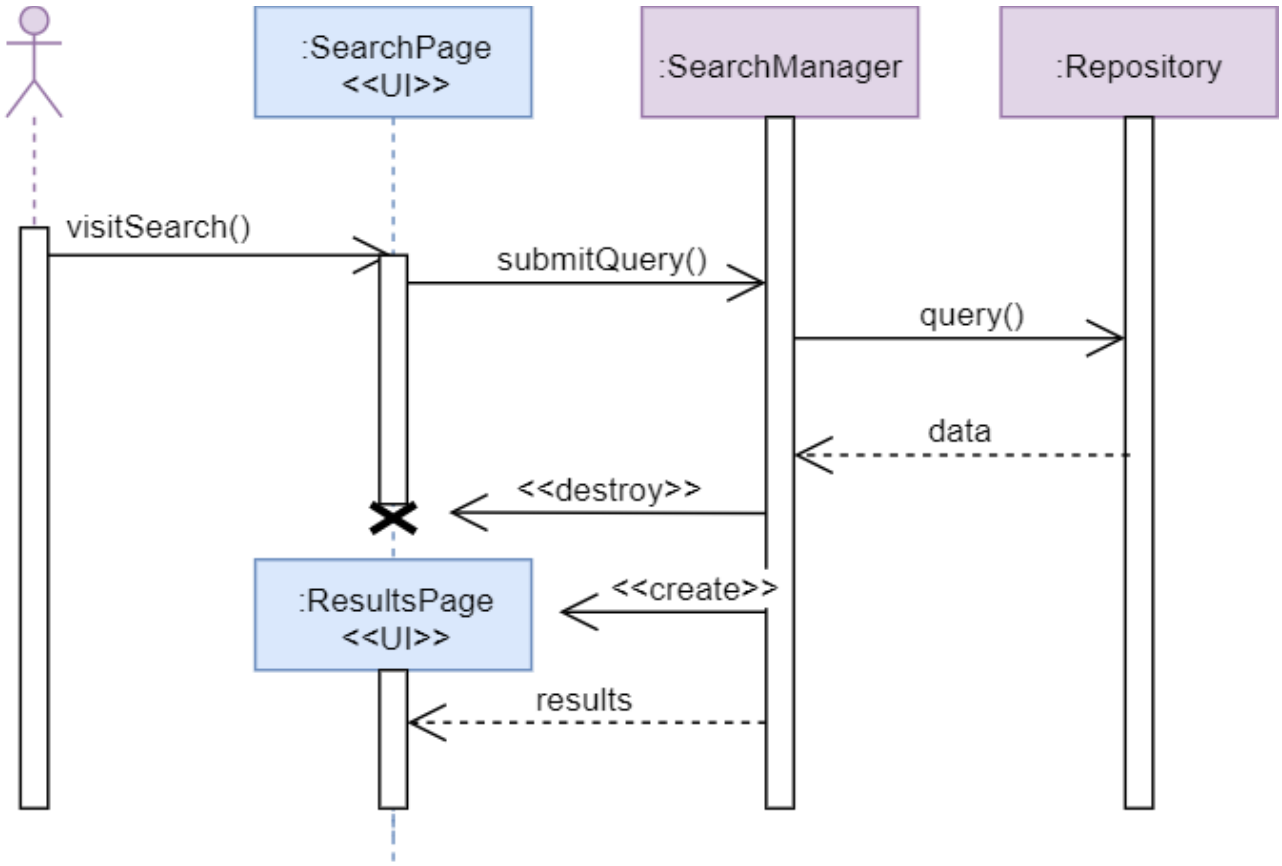
Upload File



Download File



Search


:SearchPage
<<UI>>
:SearchManager
:Repository
visitSearch()
submitQuery()
query()
data
<<destroy>>
:ResultsPage
<<UI>>
<<create>>
results

Design Goals

- **Advanced search functions:** The search engine must allow users to query the database using three main approaches. Through Metadata (subset of Dublin Core and Darwin Core terms) using logical operators. Through georeferencing using polygon limits on a map. As well as regions within the color-spaces of key models.

- **Ease of use:** The interface should be intuitive and very easy to use for users of any skill level.

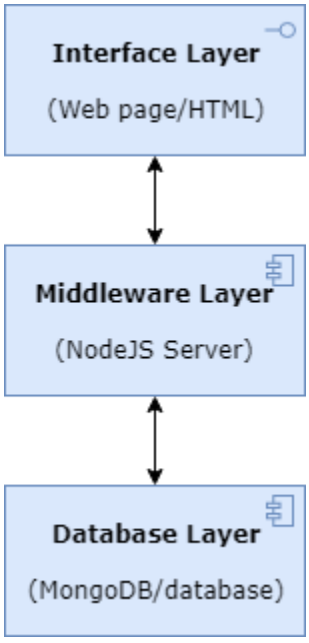
- **Flexibility:** Allow for the addition of search terms and visual model metrics as they may change over time.

- **Scalability:** Approximately 1 million files could be uploaded in the first year and the system should be able to support an additional 1-3 million files a year.

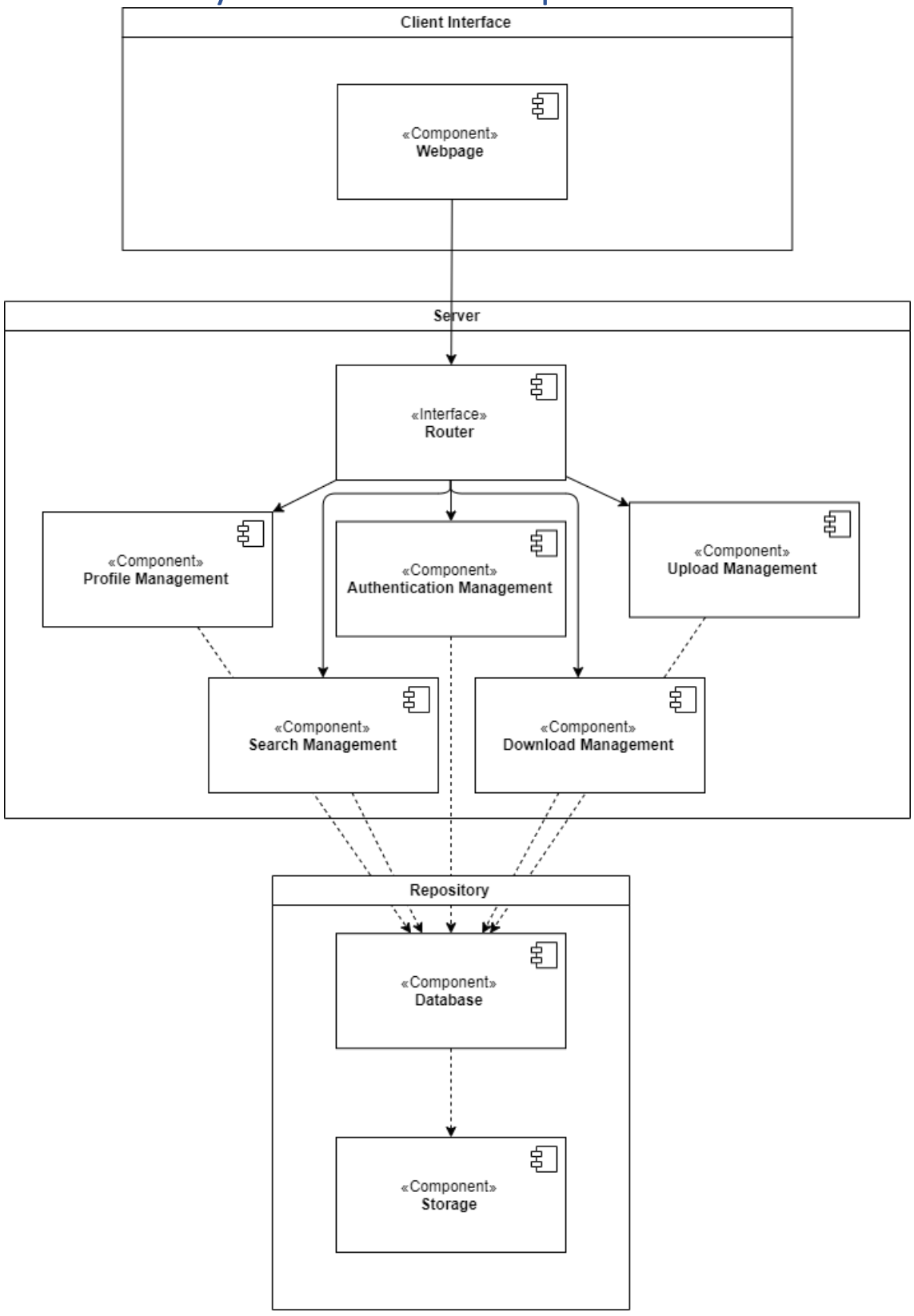
Logical Architecture

The architecture for this application could potentially use the Client/Server architectural style. However, we need to use the repository style as well, so in order to do this we have chosen to use the Three-Layer Architectural Style.

- A user interface layer (Webpage), which will be allow for interaction with the second layer as well as displaying of any information.
- A middleware layer (NodeJS), which is responsible for accepting any requests and of processing of data before it is sent to the first(webpage) or third (MongoDB) layer.
- A database layer (MongoDB), which responds with data queried to/from the second (NodeJS) layer.



System Decomposition



Web Based Resources

As a proof of concept for the viability of the chosen technologies, a sample website has been created. It allows the uploading of files and displays a list of files that have been uploaded. It can be visited here:

http://sc-2.cs.mun.ca

The Github repository for this project is available at:

https://github.com/Zerocrossing/9605SEGroup2