Phase 2
CS 4376.0U1
Team 1
Team URL: https://cs-4376-cyberminer.herokuapp.com/

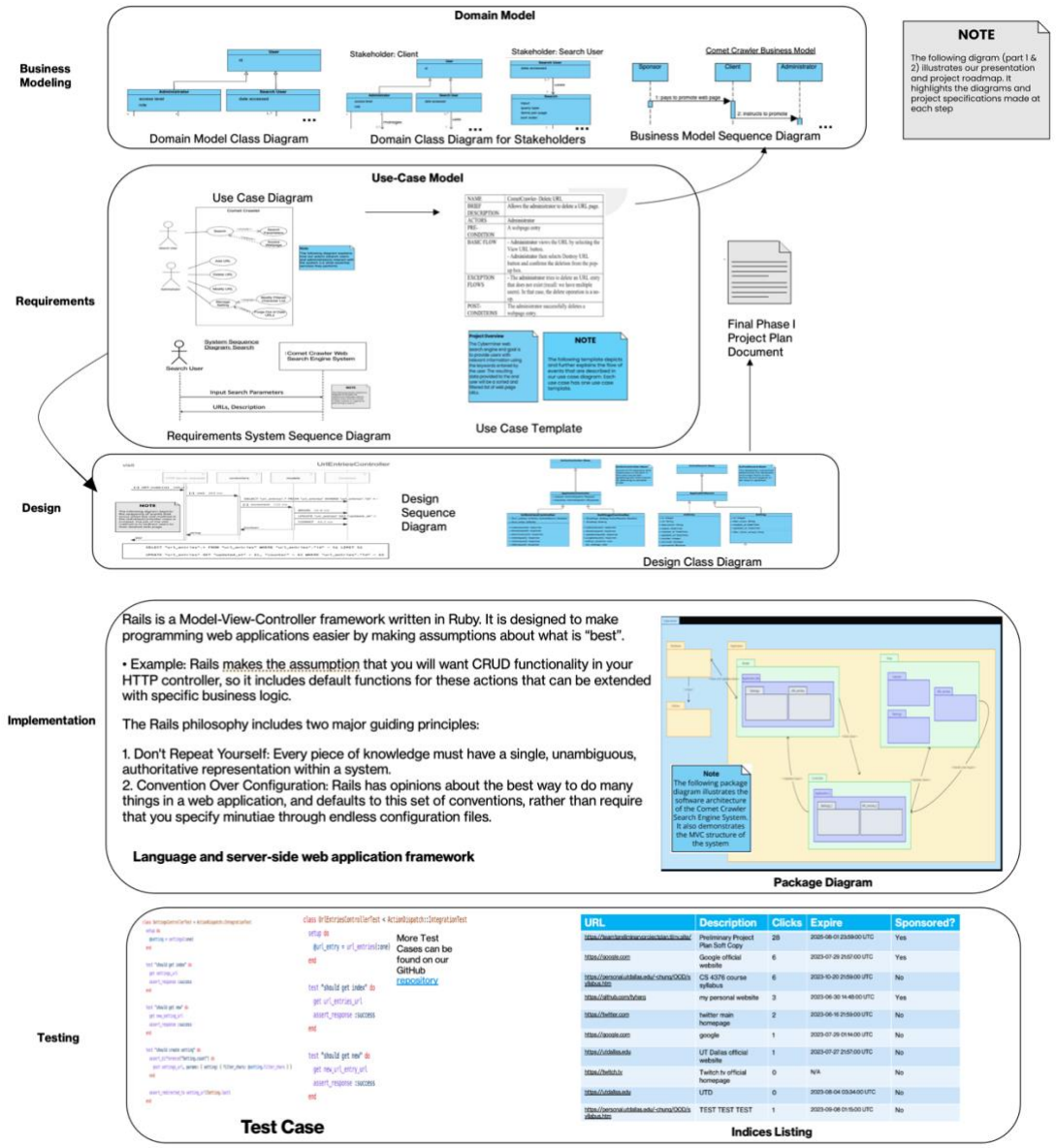
Project Outline

By: Areebah Fatima, Ilhaam Syed, Mathew Bedford, Nathan Heindl, Tyler Hargreaves, and Darrien Kramer

CONTENTS

Section 1: Outline

UP Road Map



Project overview

The project's end goal is to create a system that will provide users with relevant information using the keywords entered by the user. The resulting data provided to the end user will be a sorted and filtered list of web page URLs. The system also allows sponsors to pay to promote their websites to the top of the search result list.

The major components of the search engine will include a search interface, indexing system, result filtering, query processing, etc. The search interface will allow the end user to interact with the system, enter search queries, and retrieve information. The indexing system will organize, store, and rank information to allow for a reasonably fast search. Result filtering will ensure that our system provides users with relevant, up-to-date data in sorted order. Finally, the query processing component of the project will be responsible for interpreting user inputs, identifying keywords in said input, performing index lookup, and retrieving results.

Section 2: Requirements Specification

2.1 Functional Requirements

1F. Case-sensitive Search: The system shall store and retrieve the input exactly as provided, preserving case sensitivity.

2F. Hyperlink enforcement: When a user clicks on a URL obtained from a query result, the system shall redirect the user to the corresponding website.

3F. Specifying OR/AND/NOT Search: The system shall allow users to specify the search mode (OR, AND, or NOT) for keyword-based searches.

4F. Multiple search engines: The system shall support the concurrent execution of multiple search engines.

5F. Purging of out-of-date URL and descriptions: The system shall provide functionality to delete outdated URLs and their corresponding descriptions from the database.

6F. Query result listing options: The system shall offer the ability to list query results in ascending alphabetical order, most frequently accessed order, or based on payment preferences.

7F. Customizable result display and navigation: The system shall allow users to set the number of results displayed per page and provide navigation functionality between pages.

8F. Autofill with error correction: The system shall automatically provide autofill suggestions while correcting typographical errors.

9F. Symbol filtering based on user configuration: The system shall filter out symbols that are not considered meaningful, according to user-configured settings.

10F. Sponsored Web Entries with Priority Display: The system shall allow web entries to be designated as sponsored if an advertiser has made a payment for promotional purposes. Sponsored URLs shall be given priority in the display of query results, ensuring they are shown prominently before non-sponsored entries.

11F. Search Query Result List: The system shall provide users with the ability to input a search query, and upon submission, generate and display a list of URL entries as the output, based on the search query.

12F. Parsing and Filtering of User Input Queries: The system shall parse user input queries to extract relevant keywords while ignoring specified filter characters. The extracted keywords shall be utilized to generate the resulting output list of URL entries.

13F. URL Entry Management: The system shall provide specific users with the ability to add, modify, delete, and promote URL entries. These privileged users shall have the necessary access and permissions to perform these actions on the URL entries within the system. This privileged user functionality must be implemented during the second iteration of the project.

2.2 URL Syntax Context Free Grammar (BNF)

URL ::= {['http://'|'https://']} {www.} Identifier '.' ['edu' | 'com' | 'org' | 'net' | 'gov' | 'mil']

Identifier ::= [letter | digit]$^{+}$

letter ::= ['a' | 'b' | 'c' | 'd' |… | 'y' | 'z' | 'A' | 'B' | 'C' | 'D' | ... | 'Y' | 'Z']

digit ::= ['1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9' | '0']

2.3 Non-Functional Requirements

1N. **Performance**: The Comet Crawler system shall provide fast and efficient search results, ensuring minimal response times even when handling many concurrent user requests.

2N. **Compatibility and Portability**: The Comet Crawler system shall be compatible with commonly used web browsers, ensuring proper functionality and consistent user experience across different browser versions. It should adhere to web standards and compatibility guidelines.

3N. **Usability**: The Comet Crawler system shall have a user-friendly interface, providing a seamless and intuitive search experience for users. It should be accessible across different devices and platforms.

4N. **Error Handling**: The Comet Crawler system shall implement comprehensive error handling mechanisms to handle exceptions and provide informative error messages to users.

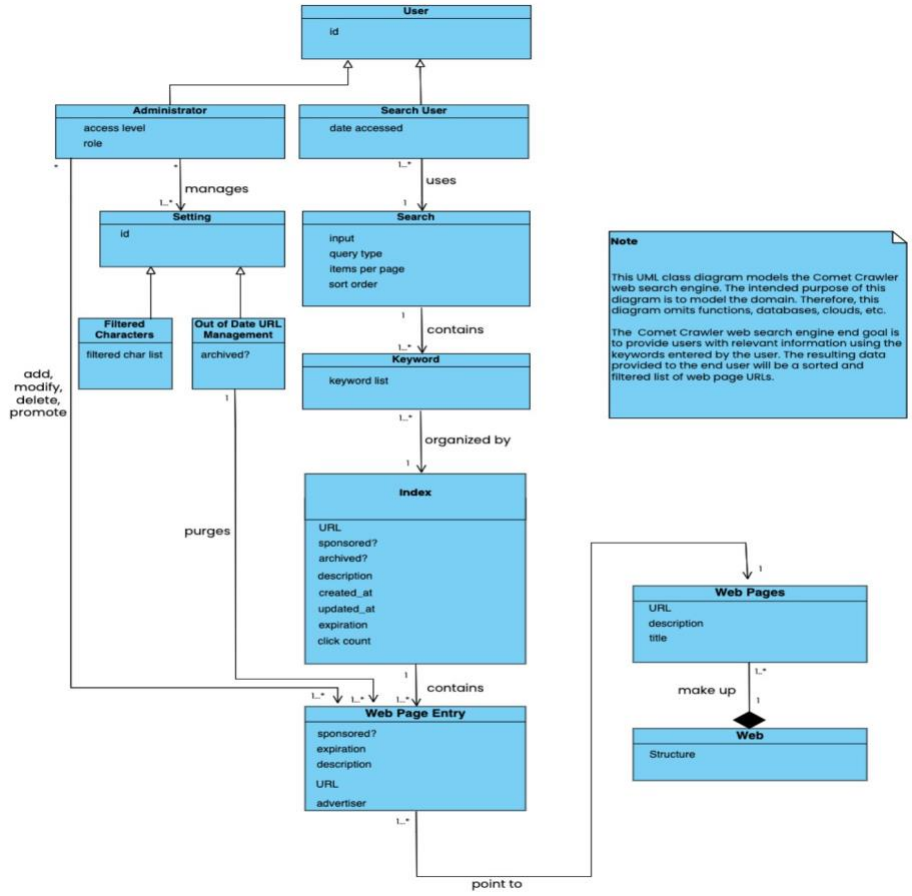
5N. **Reliability**: The Comet Crawler shall be highly reliable, providing consistent and uninterrupted search functionality to users. It should minimize system failures and errors, ensuring reliable access to search capabilities and delivering accurate results.

Section 3: Identifying Stakeholders

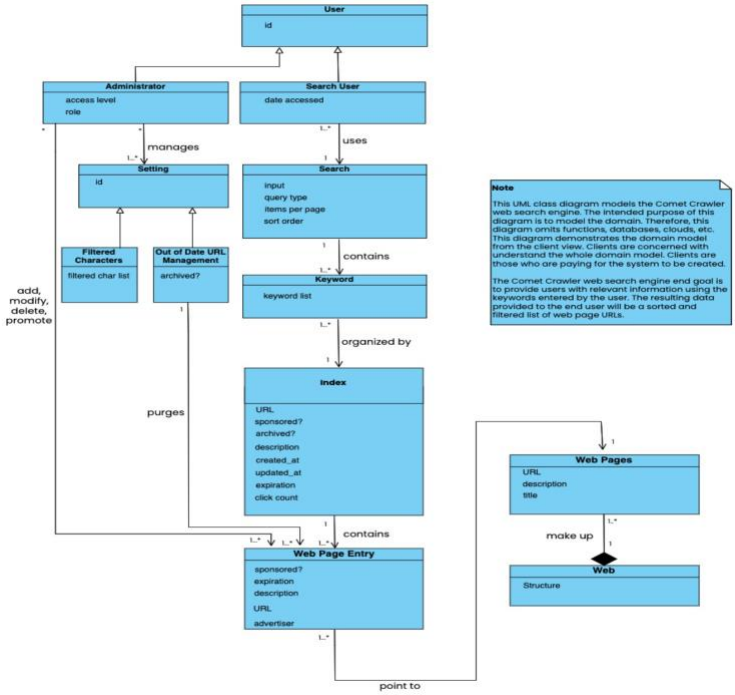
- **Administrator:** Are privilege users with the ability to add, delete, and modify URL entries. In addition to this, they also manage the system filtering and purging settings.
- **Client:** Are the ultimate owner of the project. This ownership makes them the recipients of project deliverables and the resulting product.
- **Sponsor:** Are the entities that pay money to promote their websites to the top of the search result list. This stakeholder is the central piece of the business model.
- **Search Users:** Are the individuals who interact with the Comet Crawler System. They will provide the search parameters and view the associated results. Sponsors will target these users.

Section 4: Diagrams

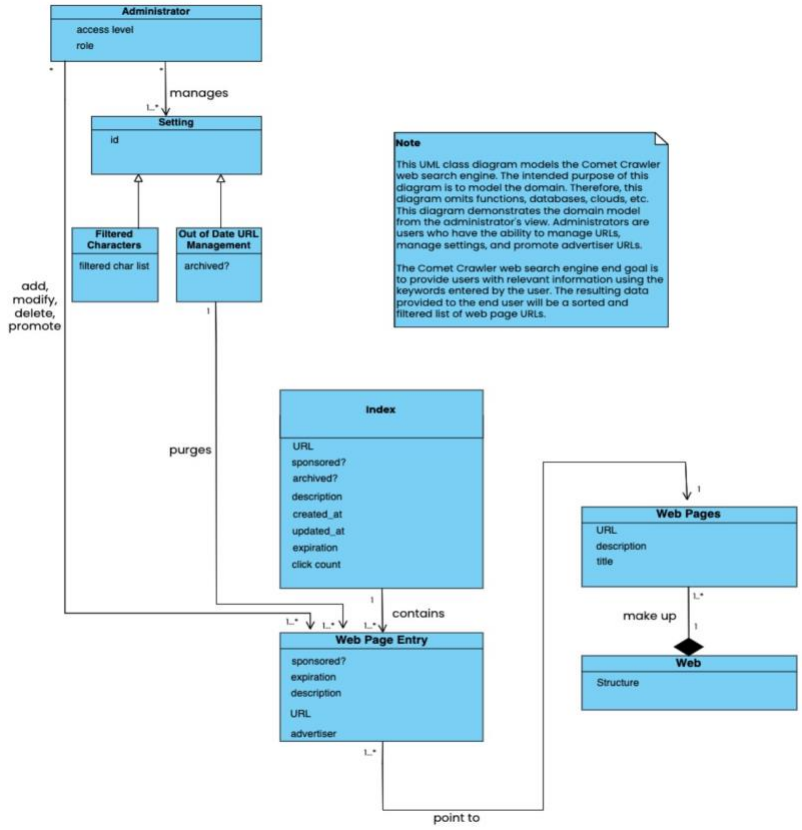
Domain Model Class Diagram: (Part of business model)



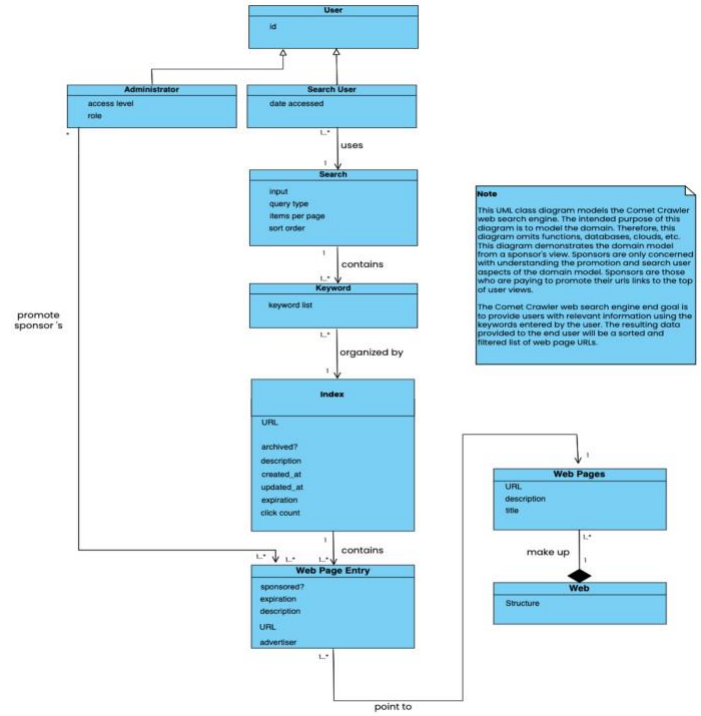
Stakeholder Class Diagram: Client (Part of business model)



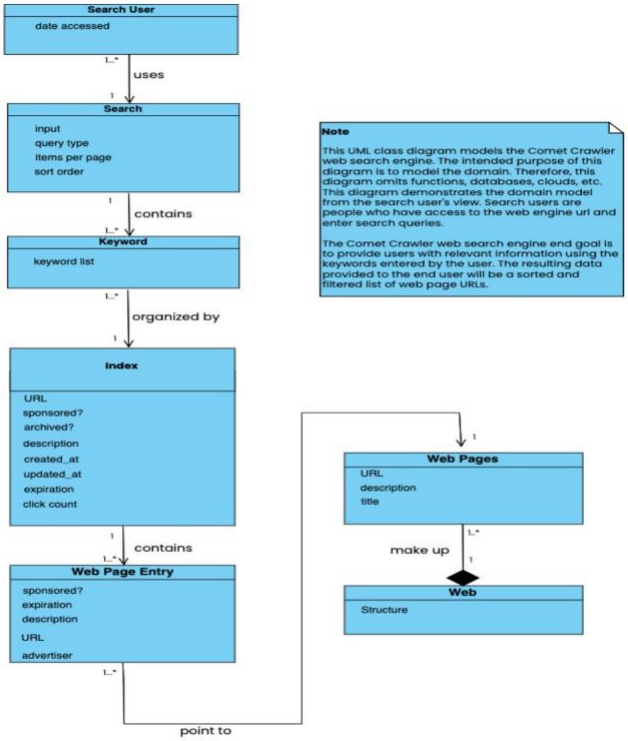
Stakeholder Class Diagram: Administrator (Part of business model)



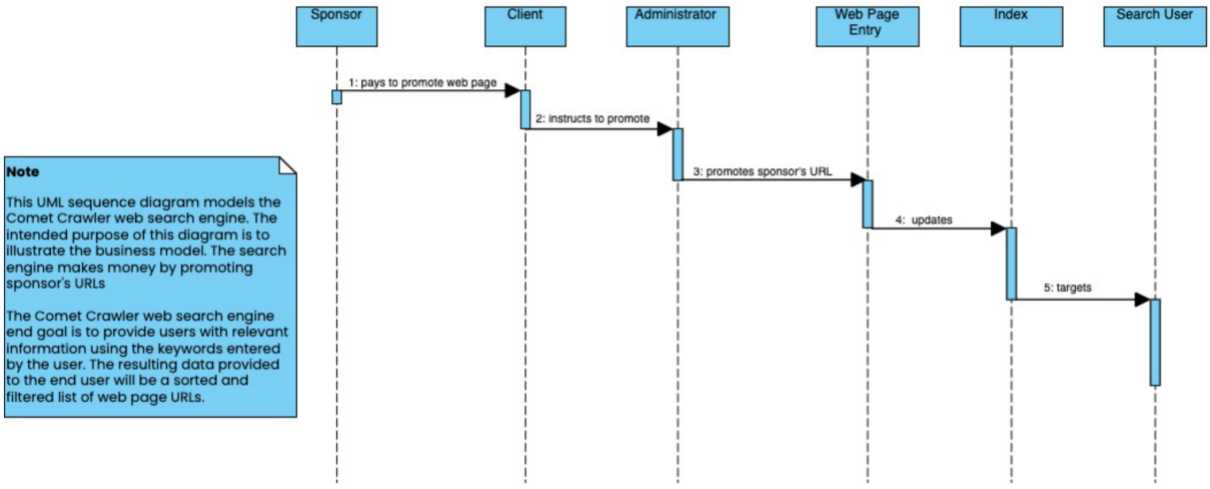
Stakeholder Class Diagram: Sponsor (Part of business model)



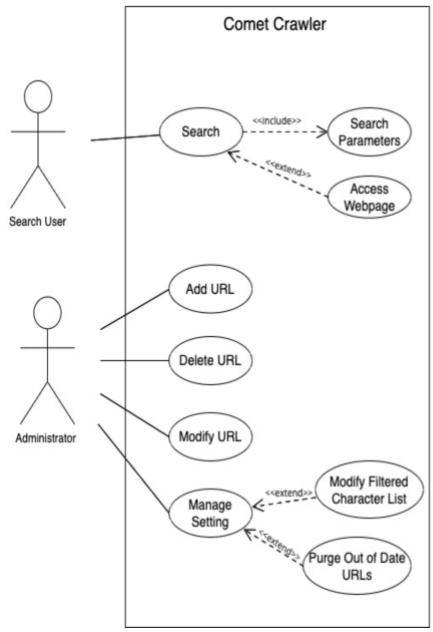
Stakeholder Class Diagram: Search Users (Part of business model)



Business Model Sequence Diagram: (Part of business model)



Use Case Diagram: (Part of requirements)



Note

The following diagram explains how our actors (search users and administrators) interact with the system (i.e what essential services they perform)

Use Case Templates: (Part of requirements)

NAME	CometCrawler- search
BRIEF DESCRIPTION	Allows search users to search URL pages based on a single or a set of keywords. It also extends the search by allowing search users to view the return pages.
ACTORS	search user
PRE-CONDITION	Query type, Sort Order, Items per page. Default is provided for all of them.
BASIC FLOW	Search: - User enters keywords in the search bar. - Keywords are interpreted and matched with entries. - The results are posted according to the preconditions provided. - The user accesses the webpage(s)
EXCEPTION FLOWS	Search: - The user enters keywords that are not in the database. - The search will lead to 0 results.
POST-CONDITIONS	Users can do a successful search. Users can view/delete/update URL entries successfully

NAME	CometCrawler- Add URL
BRIEF DESCRIPTION	Allows the administrator to add a URL page.
ACTORS	Administrator
PRE-CONDITION	URL entry form
BASIC FLOW	- Administrator must select a new webpage entry from the home page. - Administrator provides the URL to be added, a description of the URL, and an expiration date. - The new webpage entry is created when the Administrator clicks Create URL Entry.
EXCEPTION FLOWS	- The administrator tries to add an URL entry that is already in the database. In that case, the create falls back to modify operation.
POST-CONDITIONS	The administrator successfully adds a new webpage entry or modifies an existing webpage entry.

NAME	CometCrawler- Delete URL
BRIEF DESCRIPTION	Allows the administrator to delete a URL page.
ACTORS	Administrator
PRE-CONDITION	A webpage entry
BASIC FLOW	- Administrator views the URL by selecting the View URL button. - Administrator then selects Destroy URL button and confirms the deletion from the pop-up box.
EXCEPTION FLOWS	- The administrator tries to delete an URL entry that does not exist (recall: we have multiple users). In that case, the delete operation is a no-op.
POST-CONDITIONS	The administrator successfully deletes a webpage entry.

NAME	CometCrawler- Modify URL
BRIEF DESCRIPTION	Allows the administrator to modify a URL page.
ACTORS	Administrator
PRE-CONDITION	A webpage entry
BASIC FLOW	- Administrator views the URL by selecting the View URL button. - The Administrator then selects the Edit URL button. - The Administrator then updates the information and confirms the edit.
EXCEPTION FLOWS	- The administrator tries to modify an URL entry that is no longer in the database (recall: we have multiple users). In that case, the modify is a no-op operation.
POST-CONDITIONS	The administrator successfully modifies a webpage entry.

NAME	CometCrawler- Manage Setting
BRIEF DESCRIPTION	Allows the administrator to manage settings.
ACTORS	Administrator
PRE-CONDITION	Modify filtered character list or purge out-of-date URLs request.
BASIC FLOW	- Administrator views the settings page. - The Administrator updates the settings or requests to purge out-of-date URLs. - The administrator receives a confirmation page.
EXCEPTION FLOWS	- The administrator tries to update a setting that is not allowed. In that case, the return page informs the user of no-op.
POST-CONDITIONS	The administrator successfully updates settings or purges expired pages.

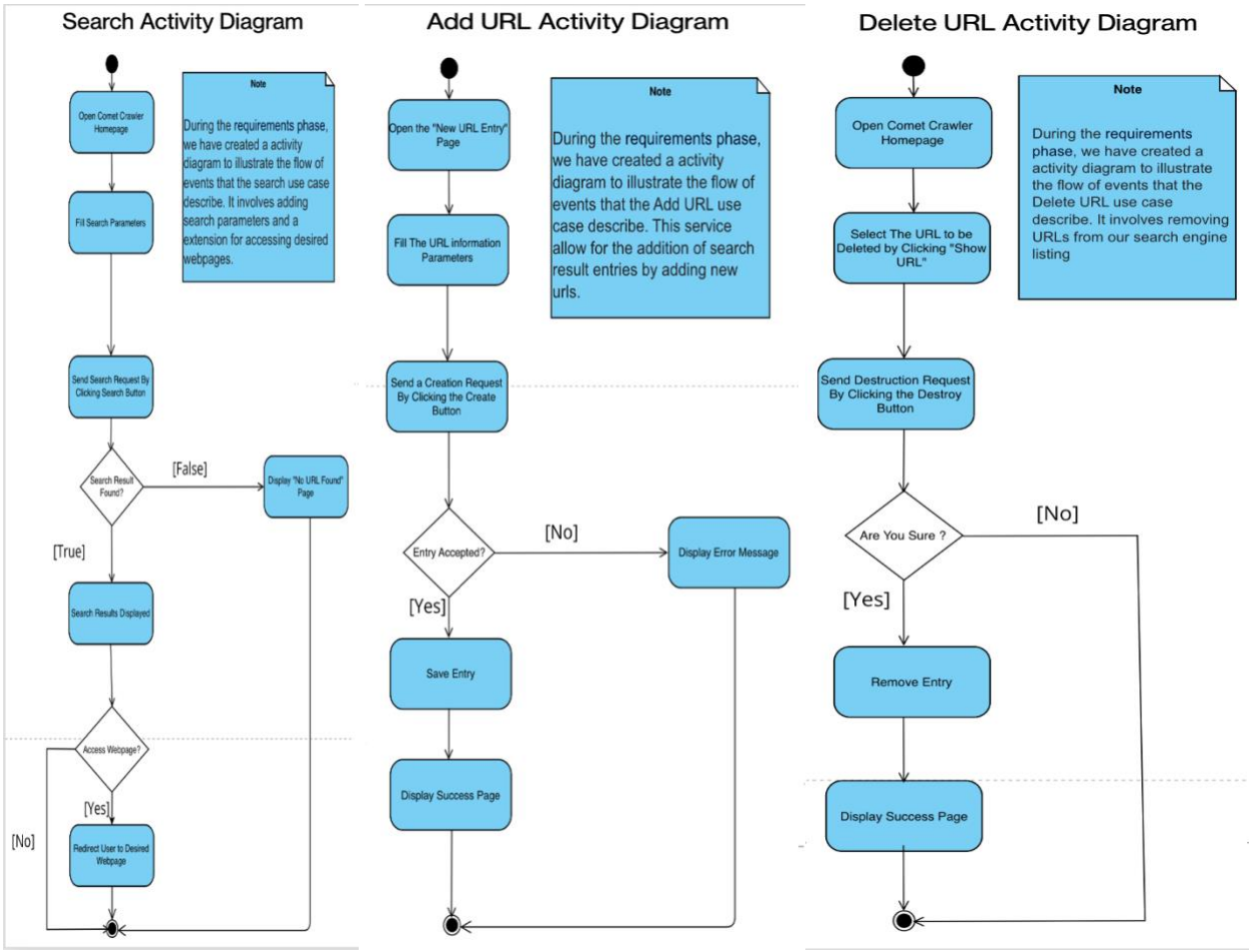
Project Overview

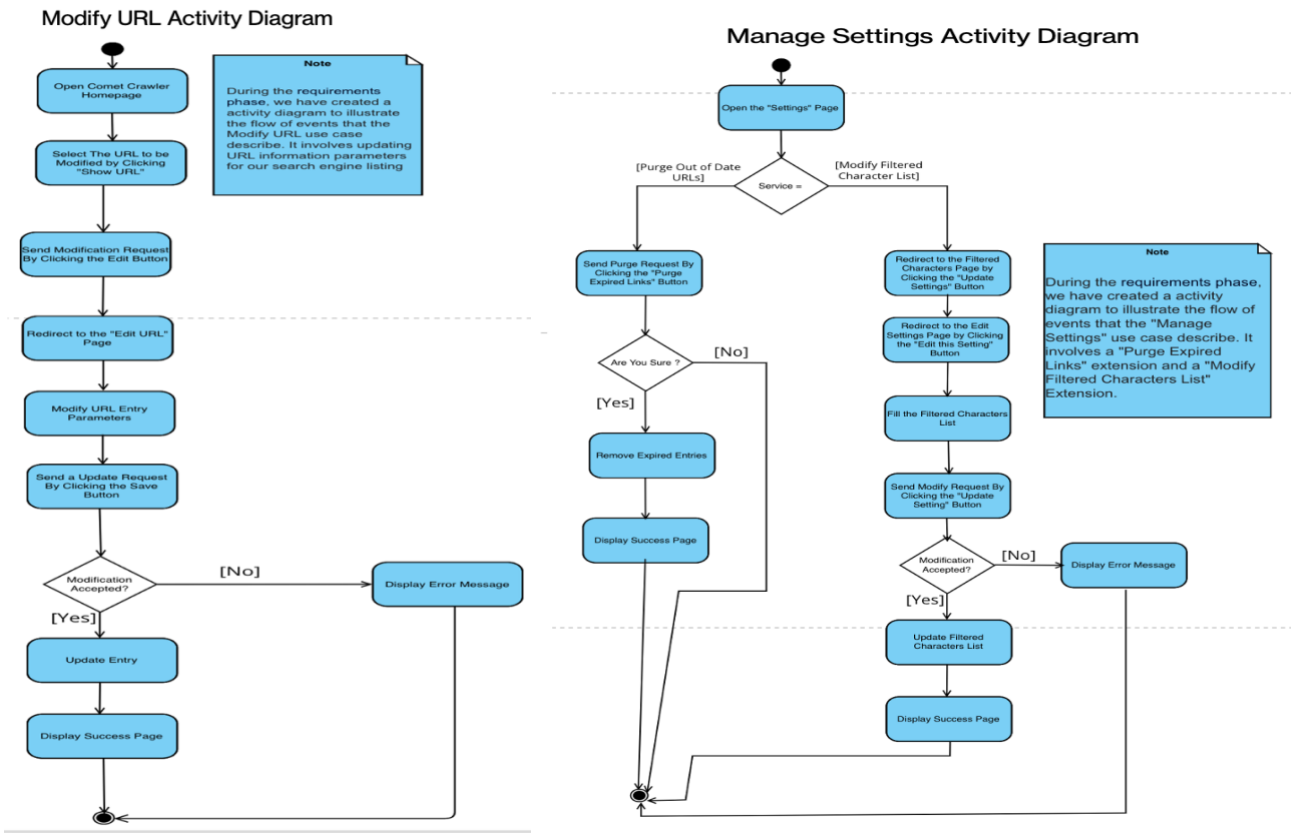
The Cyberminer web search engine end goal is to provide users with relevant information using the keywords entered by the user. The resulting data provided to the end user will be a sorted and filtered list of web page URLs.

NOTE

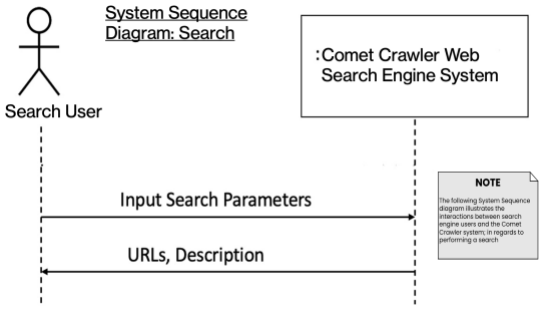
The following template depicts and further explains the flow of events that are described in our use case diagram. Each use case has one use case template.

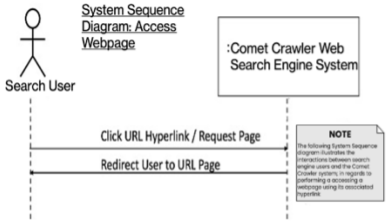
Activity Diagrams:

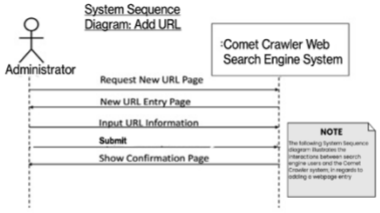


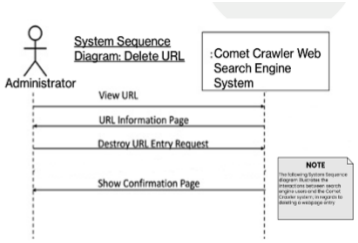


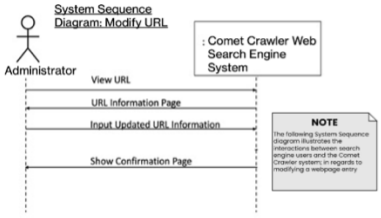
System Sequence Diagrams: (Part of requirements)

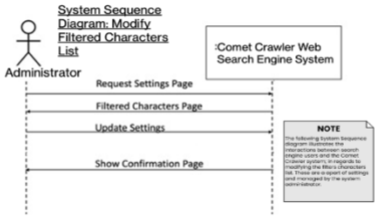


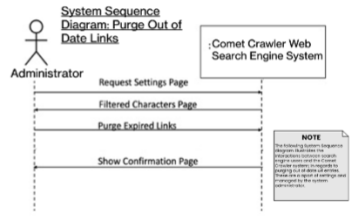






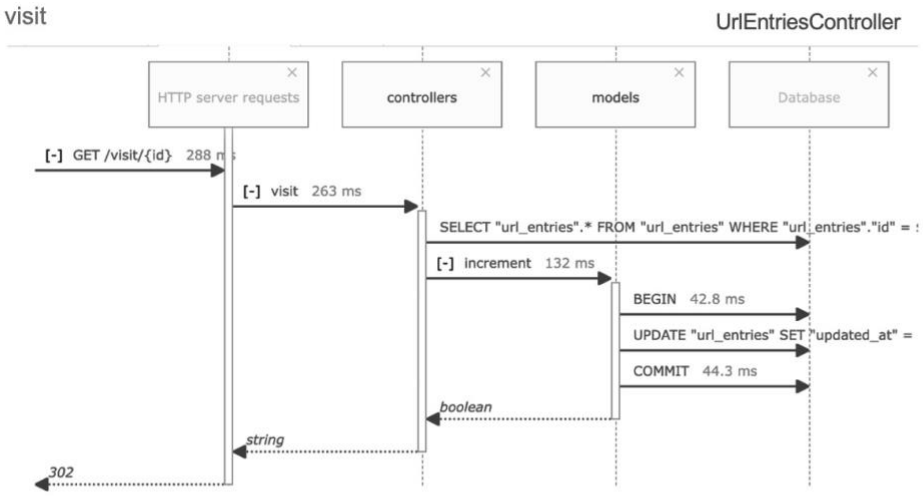






Sequence Diagrams: (Part of Design)

URLEntriesController Diagrams

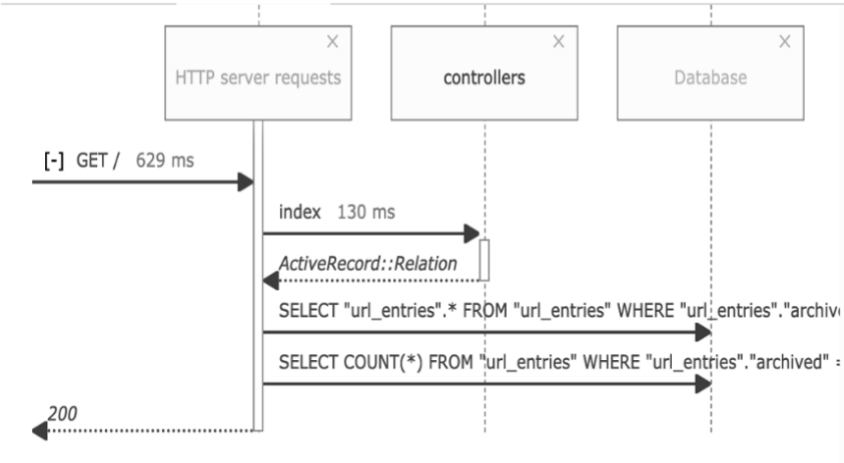



```
SELECT "url_entries".* FROM "url_entries" WHERE "url_entries"."id" = $1 LIMIT $2

UPDATE "url_entries" SET "updated_at" = $1, "counter" = $2 WHERE "url_entries"."id" = $3
```

NOTE

The following digram depicts the sequence of events that occur when the visit method in the UrlEntriesController class is invoked. The job of the visit method is to redirect users to their desired web page

Index

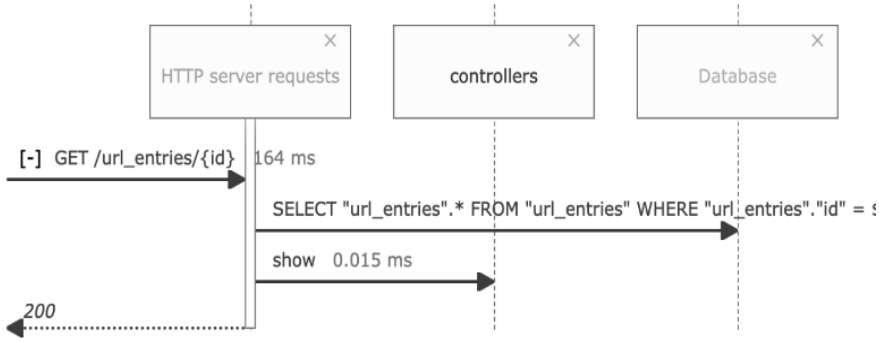



```
SELECT "url_entries".* FROM "url_entries" WHERE "url_entries"."archived" = $1
AND ("url_entries"."sponsored" = $2 OR "url_entries"."sponsored" = $3) ORDER
BY "url_entries"."sponsored" DESC LIMIT $4 OFFSET $5

SELECT COUNT(*) FROM "url_entries" WHERE "url_entries"."archived" = $1 AND
("url_entries"."sponsored" = $2 OR "url_entries"."sponsored" = $3)
```

NOTE

The following digram depicts the sequence of events that occur when the Index method in the UrlEntriesController class is invoked.

show

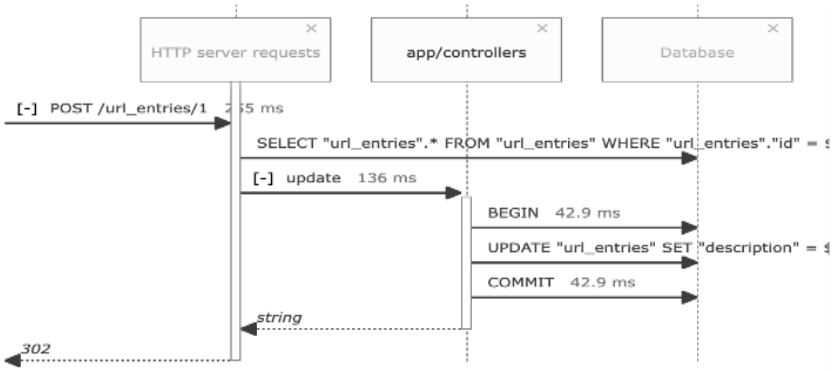



```
SELECT "url_entries".* FROM "url_entries" WHERE "url_entries"."id" = $1 LIMIT $2
```

NOTE

The following digram depicts the sequence of events that occur when the Show method in the UrlEntriesController class is invoked.

update

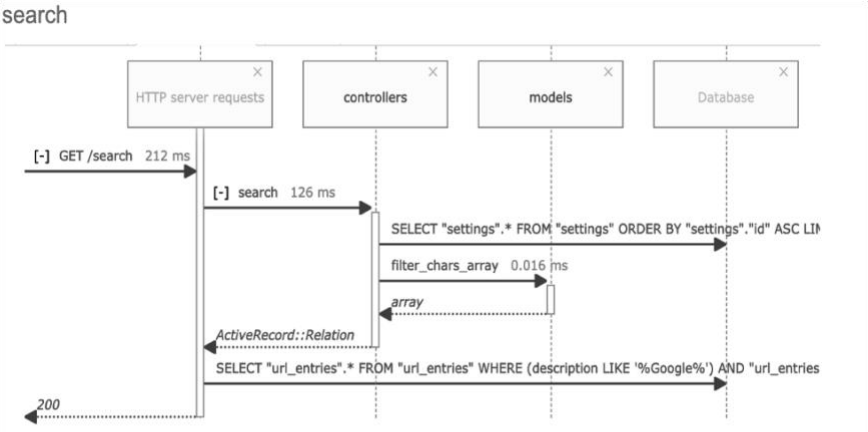



```
SELECT "url_entries".* FROM "url_entries" WHERE "url_entries"."id" = $1 LIMIT $2

UPDATE "url_entries" SET "description" = $1, "updated_at" = $2 WHERE "url_entries"."id"
= $3
```

NOTE

The following digram depicts the sequence of events that occur when the Update method in
the UrlEntriesController class is invoked.

search

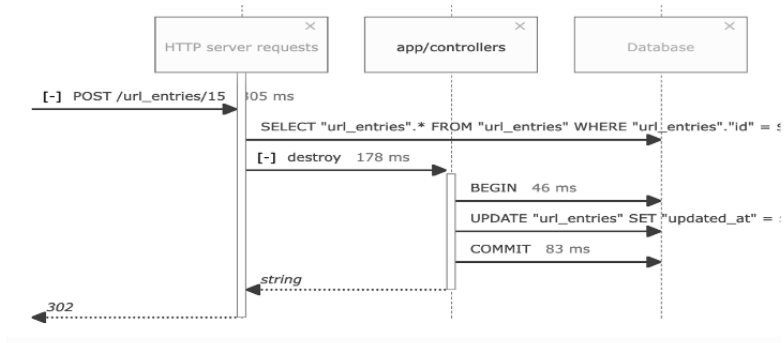



```
SELECT "settings".* FROM "settings" ORDER BY "settings"."id" ASC LIMIT $1

SELECT "url_entries".* FROM "url_entries" WHERE (description LIKE '%%') AND
"url_entries"."archived" = $1 AND ("url_entries"."sponsored" = $2 OR
"url_entries"."sponsored" = $3) ORDER BY "url_entries"."counter" DESC,
"url_entries"."sponsored" DESC LIMIT $4 OFFSET $5
```

NOTE

The following digram depicts the sequence of events that occur when the search method in the UrlEntriesController class is invoked.

destroy

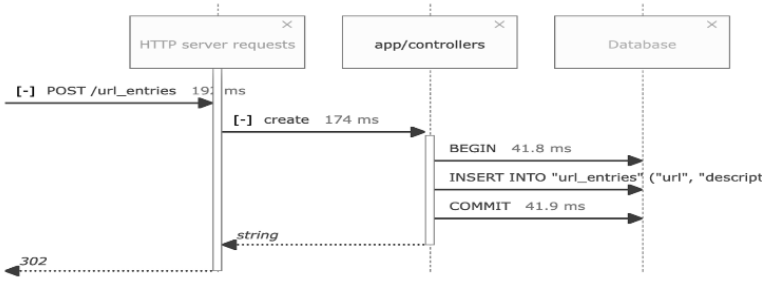



```
SELECT "url_entries".* FROM "url_entries" WHERE "url_entries"."id" = $1 LIMIT $2
```

NOTE

The following digram depicts the sequence of events that occur when the destroy method in the UrlEntriesController class is invoked.

create



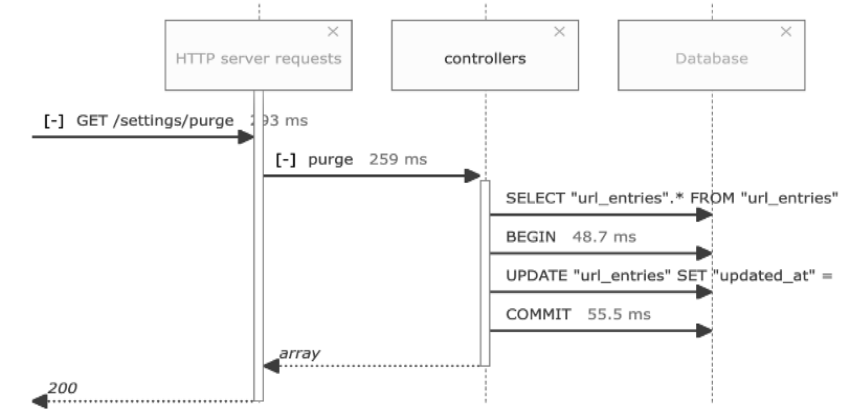

```
INSERT INTO "url_entries" ("url", "description", "expire", "created_at",
"updated_at", "counter", "archived", "sponsored") VALUES ($1, $2, $3, $4, $5, $6,
$7, $8) RETURNING "id"
```

NOTE

The following digram depicts the sequence of events that occur when the create method in the UrlEntriesController class is invoked.

SettingController Diagrams

purge

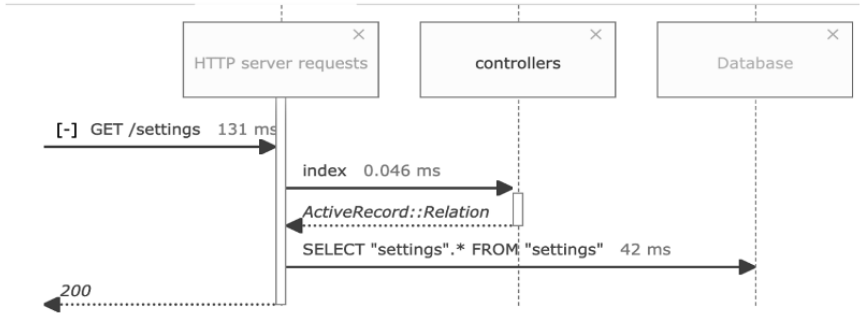



```
SELECT "url_entries".* FROM "url_entries" WHERE "url_entries"."archived" = $1 AND (expire <
'2023-06-07 18:46:55.172212')

UPDATE "url_entries" SET "updated_at" = $1, "archived" = $2 WHERE "url_entries"."id" = $3
```

NOTE

The following digram depicts the sequence of events that occur when the purge method in the SettingController class is invoked.

index

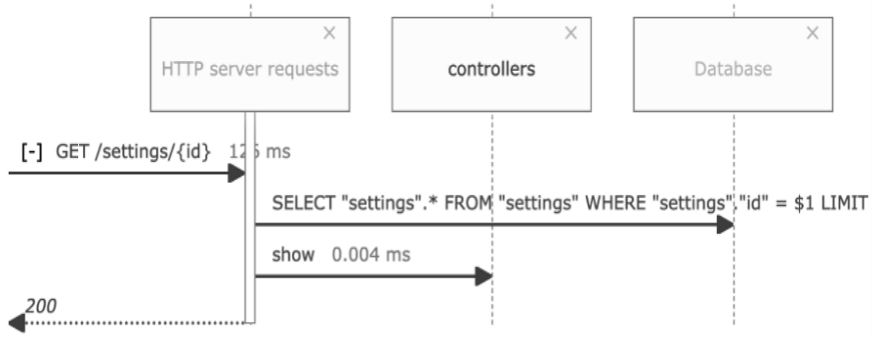



```
SELECT "settings".* FROM "settings"
```

NOTE

The following digram depicts the sequence of events that occur when the index method in the SettingController class is invoked.

show

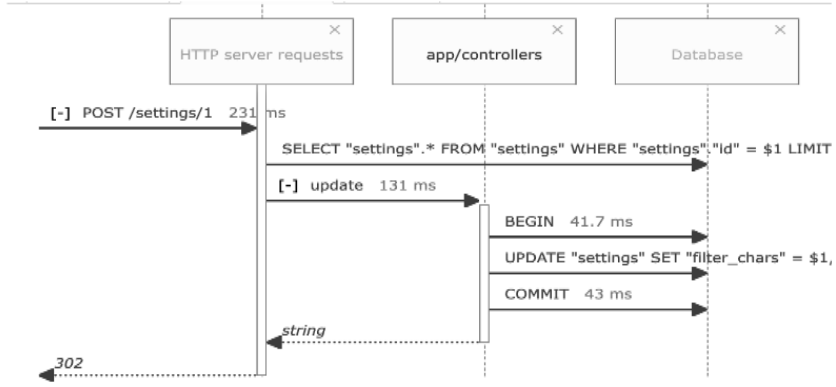



```
SELECT "settings".* FROM "settings" WHERE "settings"."id" = $1 LIMIT $2
```

NOTE

The following digram depicts the sequence of events that occur when the show method in the SettingController class is invoked.

update



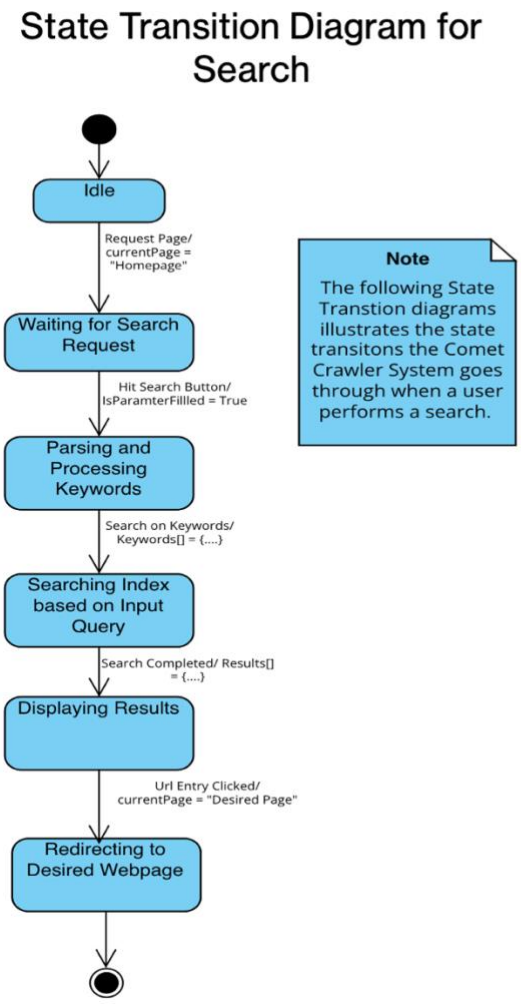

```
SELECT "settings".* FROM "settings" WHERE "settings"."id" = $1 LIMIT $2

UPDATE "settings" SET "filter_chars" = $1, "updated_at" = $2 WHERE
"settings"."id" = $3
```

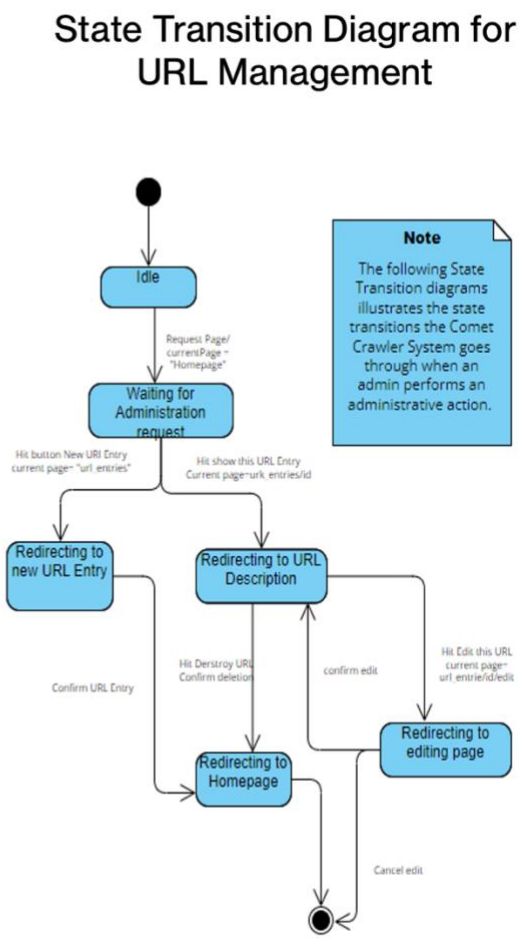
NOTE

The following digram depicts the sequence of events that occur when the update method in the SettingController class is invoked.

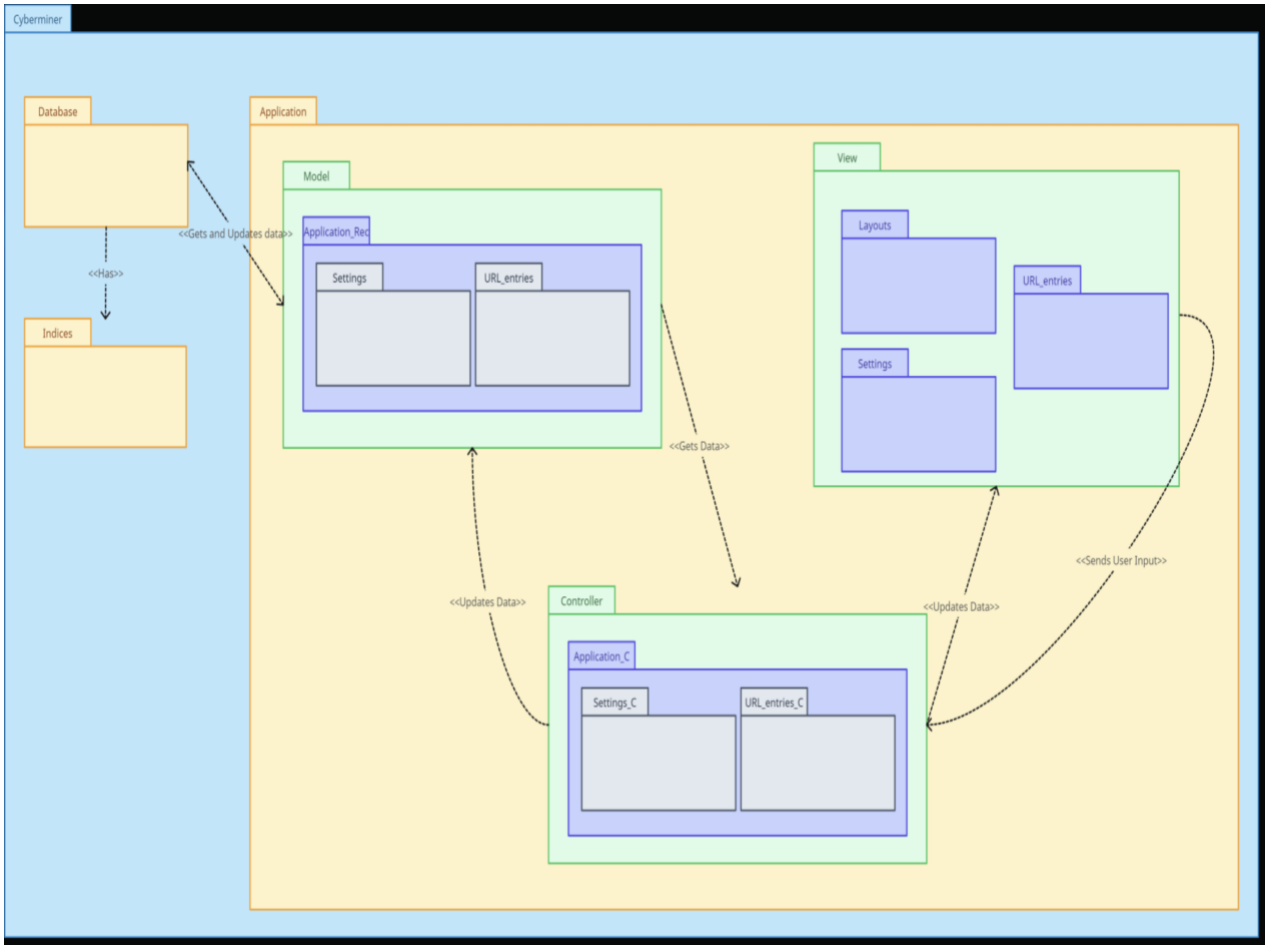
Design Class Diagram (This is a part of Design):

State Transition Diagrams:





Package Diagram:



Section 5: Test Cases

Test Cases can be found on our GitHub repository

Section 6: User Manual

Our user manual can be found here -> Click Here to View