System Design Document

Overview

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, processing logic, and external interfaces.

# *system requirements*

The project 1 of CPSC449 build, secure, and test an effective RESTful API using Python and Flask, the Python microframework. Find out how to use Flask with Python to approach database access, authentication, and other common tasks.

## operating environment

* Using PyCharm to create a Flask project, and requires the below packages been installed:

Flask-JWT-Extended 3.24.1 3.24.1

Flask-Mail 0.9.1 0.9.1

Flask-SQLAlchemy 2.4.1 2.4.1

flask-marshmallow 0.11.0 0.11.0

* Testing with Postman
* Communicating JSON
* Using with SQLITE databases and requires the below tools been installed:

DB Browser for SQLite

## Functionality Summary

### User Account Microservice

**Each user who registers should have the following data associated with them:**

* Username
* Email
* Karma

**The following operations will be exposed:**

* Create user
* Update email
* Increment Karma
* Decrement Karma
* Deactivate account

The data for the user can be in the same database or different database as the other services.

### Posting microservice

Each post should have a title, text, a community (subreddit), an optional URL linking to a resource (e.g. a news article or picture), a username, and a date the post was made.

The following operations should be exposed:

* Create a new post
* Delete an existing post
* Retrieve an existing post
* List the n most recent posts to a particular community
* List the n most recent posts to any community

When retrieving lists of posts, do not include the text or resource URL for the post.

### Voting microservice

Each post maintained by the posting microservice can be voted up or down. This service should maintain the number of upvotes and downvotes for each post. A post’s score can be computed by subtracting the number of downvotes from the number of upvotes.

The following operations should be exposed:

* Upvote a post
* Downvote a post
* Report the number of upvotes and downvotes for a post
* List the *n* top-scoring posts to any community
* Given a list of post identifiers, return the list sorted by score.

Each upvote or downvote should include a unique identifier (e.g., a URL or database key) for the post that can be used to match votes with the posts maintained by the posting microservice.

If this service is implemented with a database separate from the posting microservice, it is not responsible for verifying the existence of a post before recording or reporting votes.

### User messaging Microservice

Users can send and receive messages to each other. Messages will consist of the following data associated with them:

* Message ID
* User from
* User to
* Message timestamp
* Message contents
* Message flag

The following operations will be exposed:

* Send message
* Delete message
* Favorite message

Messaging data can be in the same database as other services or a separate one.

# SYSTEM ARCHITECTURE

## System Software Architecture

# Interfaces

## Create user

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/register |
| Usage | Register a new user |
| Request Method | post |
| Parameters | **'user\_name'**, **'first\_name'**, **'last\_name'**, **'email'**, **'password'**, **'karma'**, |
| Return | Fail (message=**'That email already exists.'**), 409  Success (message=**'User created successfully!'**), 201 |

## Update email

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/update\_email |
| Usage | Update the user email |
| Request Method | put |
| Parameters | **'user\_name'**, **'email'**, |
| Return | Fail (message=**'The user does not exist!'**), 404  Success (message=**'You updated the email!'**), 202 |

## Increment karma

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/increment\_karma |
| Usage | Increment Karma |
| Request Method | put |
| Parameters | **'user\_name'**, **'karma'**, |
| Return | Fail (**'Fail to Increment karma!'**), 404  Success (message=**'Increment karma successfully!'**), 202 |

## Decrement karma

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/decrement\_karma |
| Usage | Increment Karma |
| Request Method | put |
| Parameters | **'user\_name'**, **'karma'**, |
| Return | Fail (**'Fail to Decrement karma!'**), 404  Success (message=**'Decrement karma successfully!'**), 202 |

## Deactivate account

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/deactivate\_account/<string:user\_name> |
| Usage | Deactivate account |
| Request Method | delete |
| Parameters | **'user\_name'** |
| Return | Fail (**'Fail to Deactivate account!'**), 404  Success (message=**'You deleted a user!'**), 202 |

## Create a new post

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/create\_post |
| Usage | Create a new post |
| Request Method | post |
| Parameters | **'user\_name'**, **'title'**, **'text'**, **'community'**, **'resource\_url'** |
| Return | Fail (**'That user\_name no exists.'**), 409  Success (message=**'Post created successfully!'**), 201 |

## Delete a post

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/delete\_post/<int:id> |
| Usage | Delete a post |
| Request Method | delete |
| Parameters | **'id'** |
| Return | Fail (**'That post does not exist'**), 404  Success (message=**'You deleted a post'**), 202 |

## Retrieve an existing post

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/retrieve\_post/<int:id> |
| Usage | Retrieve a post |
| Request Method | Get |
| Parameters | **'id'** |
| Return | Fail (**'That post does not exist'**), 404  Success (message= post) |

## List the n most recent posts to a particular community

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/list\_post\_comm/<string:community> |
| Usage | List n post from a particular community |
| Request Method | Get |
| Parameters | **'community'** |
| Return | Fail (**'That post does not exist'**), 404  Success (message= post) |

## List the n most recent posts to any community

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/list\_posts/ |
| Usage | List n post recently |
| Request Method | Get |
| Parameters |  |
| Return | Fail (**'That post does not exist'**), 404  Success (message= posts) |

## Upvote a post

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/up\_vote\_post/ |
| Usage | Up vote a post |
| Request Method | put |
| Parameters | resource\_title |
| Return | Fail (**'Fail to Up vote!'**), 404  Success (message=**'Up vote successfully!'**), 202 |

## Downvote a post

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/down\_vote\_post/ |
| Usage | Down vote a post |
| Request Method | put |
| Parameters | resource\_title |
| Return | Fail (**'Fail to Down vote!'**), 404  Success (message=**'Down vote successfully!'**), 202 |

## Report the number of upvotes and downvotes for a post

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/list\_post\_votes/<string:title> |
| Usage | Report the number of upvotes and downvotes for a post |
| Request Method | Get |
| Parameters | **'title'** |
| Return | Fail (**'That post does not exist'**), 404  Success (message= post) |

## List the n top-scoring posts to any community

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/list\_n\_post\_votes/<int:n> |
| Usage | List the n top-scoring posts to any community |
| Request Method | Get |
| Parameters | **Top Number** |
| Return | Fail (**'That post does not exist'**), 404  Success (message= post) |

## List Given a list of post identifiers, return the list sorted by score.

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/list\_post\_votes\_in\_list/<string:title1,title2,titleN> |
| Usage | Given a list of post identifiers, return the list sorted by score. |
| Request Method | Get |
| Parameters | title1,title2,titleN  Example: http://localhost:5000/list\_post\_votes\_in\_list/Carlton from 'Love Is Blind' still hoping for love,Supreme Court will take up challenge to Obamacare's individual mandate |
| Return | Fail (**'That post does not exist'**), 404  Success (message= post) |

## Send message

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/send\_message |
| Usage | Send a new message |
| Request Method | post |
| Parameters | **'user\_from'**, **'user\_to'**, **'contents'**, **'flag'** |
| Return | Fail (**'fail to send Message.'**), 409  Success (message=**'Message sent successfully!'**), 201 |

## Delete a message

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/delete\_message/<int:id> |
| Usage | Delete message |
| Request Method | delete |
| Parameters | **'id'** |
| Return | Fail (**'That message does not exist'**), 404  Success (message=**'You deleted a message'**), 202 |

## List the Favorite messages

|  |  |
| --- | --- |
| Request URL | http://localhost:5000/list\_favorite\_messages |
| Usage | List Favorite messages |
| Request Method | Get |
| Parameters |  |
| Return | Fail (**'**Favorite messages **does not exist'**), 404  Success (message= post) |

# DATABASE DESIGN

Add new table TB\_USERS, TB\_POSTS, TB\_VOTES, TB\_MESSAGES.

## TB\_USERS Data Description

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TABLE NAME** | **FIELD** | **TYPE** | **NULL** | **DEFAULT** |
| TB\_USERS | ID | INT(11) | NOT NULL | AUTO |
|  | Username | VARCHAR | NOT NULL | NOT |
|  | First\_Name | VARCHAR | NULL | NOT |
|  | Last\_Name | VARCHAR | NULL | NOT |
|  | Email | VARCHAR | NOT NULL | NOT |
|  | Password | VARCHAR | NOT NULL | NOT |
|  | Karma | INT(11) | NULL | 0 |
|  | CREATE\_TIME | DATETIME | NOT NULL | AUTO\_NOW |
|  | MODIFY\_TIME | DATETIME | NOT NULL | NOT |
|  |  |  |  |  |
| TB\_POSTS | ID | INT(11) | NOT NULL | AUTO |
|  | USER\_ID | INT(11) | NOT NULL | TB\_USERS(ID) |
|  | Username | VARCHAR | NOT NULL | TB\_USERS(Username) |
|  | Title | VARCHAR | NOT NULL | NOT |
|  | Text | VARCHAR | NOT NULL | NOT |
|  | Community | VARCHAR | NOT NULL | NOT |
|  | ResourceURL | VARCHAR(255) | NOT NULL | *ForeignKey("tb\_vote.ResourceURL")* |
|  | CREATE\_TIME | DATETIME | NOT NULL | AUTO\_NOW |
|  | MODIFY\_TIME | DATETIME | NOT NULL | NOT |
| TB\_VOTES | ID | INT(11) | NOT NULL | AUTO |
|  | ResourceURL | VARCHAR(255) | NOT NULL |  |
|  | Votes | INT(11) | NULL | 0 |
|  | UpVotes | INT(11) | NULL | 0 |
|  | DownVotes | INT(11) | NULL | 0 |
|  | CREATE\_TIME | DATETIME | NOT NULL | AUTO\_NOW |
|  | MODIFY\_TIME | DATETIME | NOT NULL | NOT |
|  |  |  |  |  |
| TB\_MESSAGES | ID | INT(11) | NOT NULL | AUTO |
|  | User\_from | VARCHAR | NOT NULL |  |
|  | User\_to | VARCHAR | NOT NULL |  |
|  | Contents | VARCHAR(255) | NOT NULL |  |
|  | Flag | VARCHAR | NULL |  |
|  | CREATE\_TIME | DATETIME | NOT NULL | AUTO\_NOW |
|  | MODIFY\_TIME | DATETIME | NOT NULL | NOT |

# EXTERNAL INTERFACES