Revised Service Layer Design

Bryan Goggin

Login

POST /login

Sends login information for the user

Payload:

```
{
    "username":"abcd",
    "pass":"1234"
}
```

Response Status:

```
200: OK
400: Bad Request. Username and pass must be str.
401: Unauthorized: invalid credentials.
404: Username not found
5XX: Unexpected Error (usually server-side)
```

POST /register

Adds a user to the Users table

Payload:

```
{
    "username":"abcd",
    "pass":"1234"
}
```

Response Status:

```
200: OK
400: Bad Request. Username and pass must be str.
403: Forbidden. User already registered
5XX: Unexpected Error (usually server-side)
```

Search Recipes

GET /recipe/{recipeId}

Retrieves a particular recipeId

Response:

```
{
     “recipeId”:”abcd”,
     “recipeName”: “WXYZ”,
     “Ingredients” : [
          “flours”: “1lb”
     ]
}
```

Response Status:

```
200: OK
400: Bad request. {recipeId} must be str
404: recipeId not found
5XX: Unexpected Error (usually server-side)
```

GET /recipe/search/{search_string}

Conducts a broad search and returns results as array

Response:

```
[
    {
        "recipeId":"abcd",
        "recipeName": "WXYZ",
        "Ingredients" : [
            "flours": "1lb"
        ]
    }
]
```

Response Status:

```
200: OK (may return no results)
400: Bad request. Incorrect format.
5XX: Unexpected Error (usually server-side).
```

POST /recipe/search

Conducts a specific search and returns results as array

Payload:

```
{
    "ingredients: "cheese",
    "meal": "lunch"
}
```

Response:

Same as GET /recipe/search/

Upload/ Edit / Delete Recipes

POST /upload

Uploads a recipe

Payload:

```
{
     “recipeName”: “WXYZ”,
     “Ingredients” : [
          “flours”: “1lb”
     ],
     “uploadDate”: “27Mar2021”,
     “uploadedBy”: “User5000”
}
```

Response Status:

200: OK

400: Bad request. Incorrect format.

401 Unauthorized: Invalid credentials

5XX: Unexpected Error (usually server-side).

PUT /recipe/{recipeId}

Updates an existing recipe

Payload:

```
{
     “recipeId”:”abcd”,
     “recipeName”: “WXYZ”,
     “Ingredients” : [
          “flours”: “1lb”
```

```
            ]
        }
```

Response Status:

```
200: OK
400: Bad request. Incorrect format.
401: Unauthorized: Invalid credentials
403: Forbidden. Operation not allowed.
404: {recipeId} not found
5XX: Unexpected Error (usually server-side).
```

DELETE /recipe/{recipeId}

Deletes a recipe from the Recipes table

Response Status:

```
200: OK
400: Bad request. Incorrect format.
401: Unauthorized: Invalid credentials
403: Forbidden. Operation not allowed.
404: {recipeId} not found
5XX: Unexpected Error (usually server-side).
```

Reviews

GET /reviews/{recipeId}

Get all the reviews for a given recipeId

Response Status:

```
200: OK
400: Bad request. Incorrect format.
403: Forbidden. Operation not allowed.
404: {recipeId} not found
5XX: Unexpected Error (usually server-side).
```

POST /reviews/{recipeId}

Upload a new review for a given recipeId

Payload:

```
{
    “recipeId”:”abcd”,
    “reviewText”: “this is a great food!”,
    “reviewScore”:4.0,
    “reviewedBy: “foodcritic10”,
    “reviewDate”: “12Feb2021”
}
```

Response Status:

```
200: OK
400: Bad request. Incorrect format.
404: {recipeId} not found
5XX: Unexpected Error (usually server-side).
```

GET /reviewsByUsers/{userId}

Get all the reviews submitted by a given user.

Response:

```
{
    [
        "reviewId": 121,
        "recipeId": "recipe123",
        "recipeName" : "chicken dumpings",
        "reveiwText": "so good",
        "reviewScore": 4.9
    ],
    [
        …
    ]
}
```

Response Status:

200: OK (may return no results)

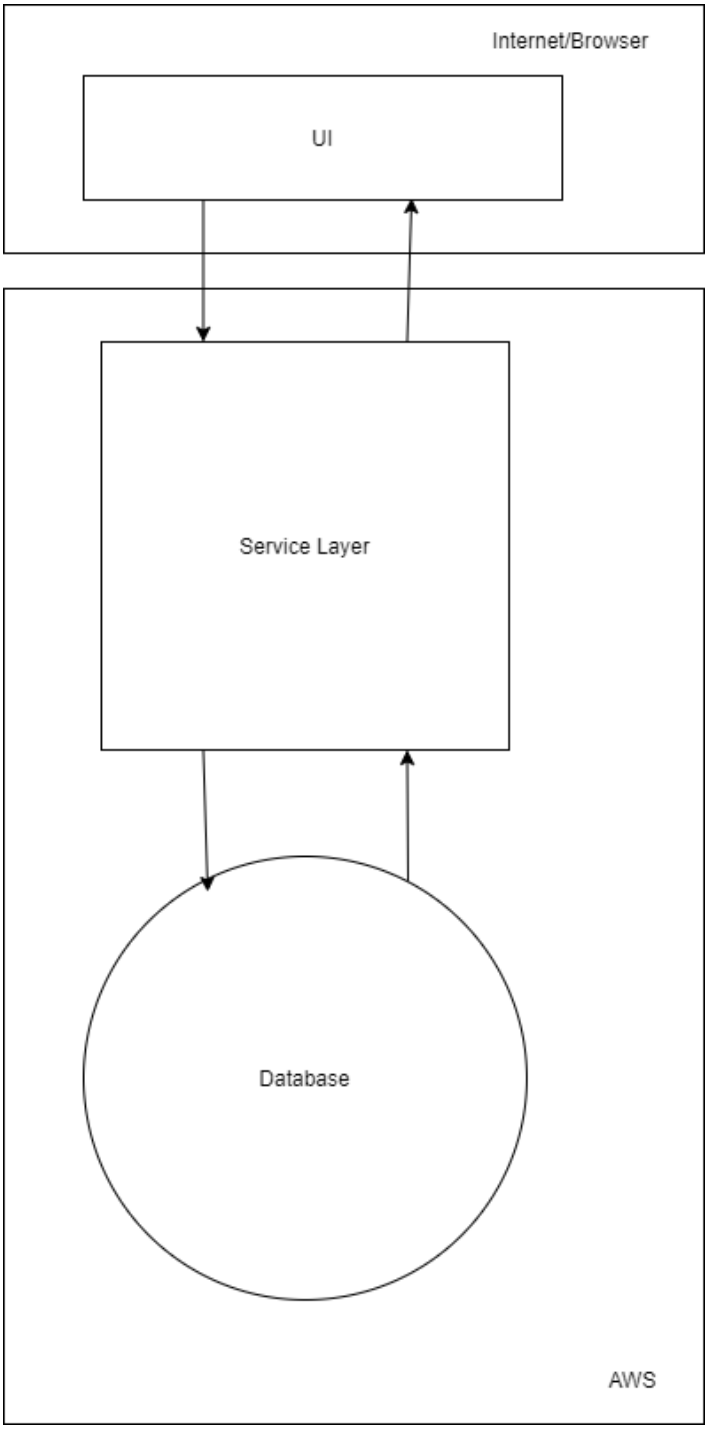
400: Bad request. Incorrect format.

404: {UserId} not found

5XX: Unexpected Error (usually server-side).

Diagram

The basic diagram will look like this



The Users will connect to the UI on their browser, which will send requests to the service layer within the AWS infrastructure. The service layer will process the requests to query the database to pull the information the user is wanting. The database will return the data to the service layer, where it will be processed/formatted and passed back to the UI for the user to see or to be rendered into a web page. Searches will be conducted using a combination of DB built ins and processing using python or javascript.

Stretch Features

As a stretch feature, users may want to create or join a user group. Below are requests and responses to facilitate that functionality.

POST /usergroup/create

Creates a usergroup

Payload:

```
{
     "groupName": "Sushi Supremos",
     "Members" : [],
     "createDate": "27Mar2021",
     "CreatedBy": "User5000",
     "Settings": {
          "Privacy": "public",
          "Notifications" : "Off"
     }
}
```

Response Status:

```
200: OK
400: Bad request. Incorrect format.
401 Unauthorized: Invalid credentials
5XX: Unexpected Error (usually server-side).
```

GET /usergroup/join/{usergroupId}

Adds logged-in user to a usergroup

Response Status:

```
200: OK
400: Bad request. Incorrect format.
401 Unauthorized: Invalid credentials
403: Forbidden. Operation not allowed
404: {usergroupid} not found
5XX: Unexpected Error (usually server-side).
```

GET /usergroup/{usergroupId}

Retrieves information for given usergroup

Response Status:

```
200: OK
400: Bad request. Incorrect format.
404: {usergroupid} not found
5XX: Unexpected Error (usually server-side).
```

GET /usergroupRecipe/{usergroupId}/{recipeId}

Share a recipe to a usergroup

Response Status:

```
200: OK
400: Bad request. Incorrect format.
401: Not Authorized
403: Forbidden. Operation not allowed.
404: {usergroupid} or {recipeId) not found
5XX: Unexpected Error (usually server-side).
```