

CS307 - Database - Project2

Group members: 刘达洲12311004, 魏宇晴12311043

Lab session: Monday class 9-10 Wangzhongqiu

Contributions: 刘达洲 Task2.3 and Task2.4 (50%) +

魏宇晴 Task2.1 and Task2.2 and Task2.4 (50%)

Project2.rar

- DatabaseProject2(codes_containing_backend_connected_to_frontend)
- my-vue-app (frontend_code)
- pubmed-main
- Datagrip_diagram.png
- E-R图 v1.drawio
- E-R图 v1.png
- Report_12311004_12311043

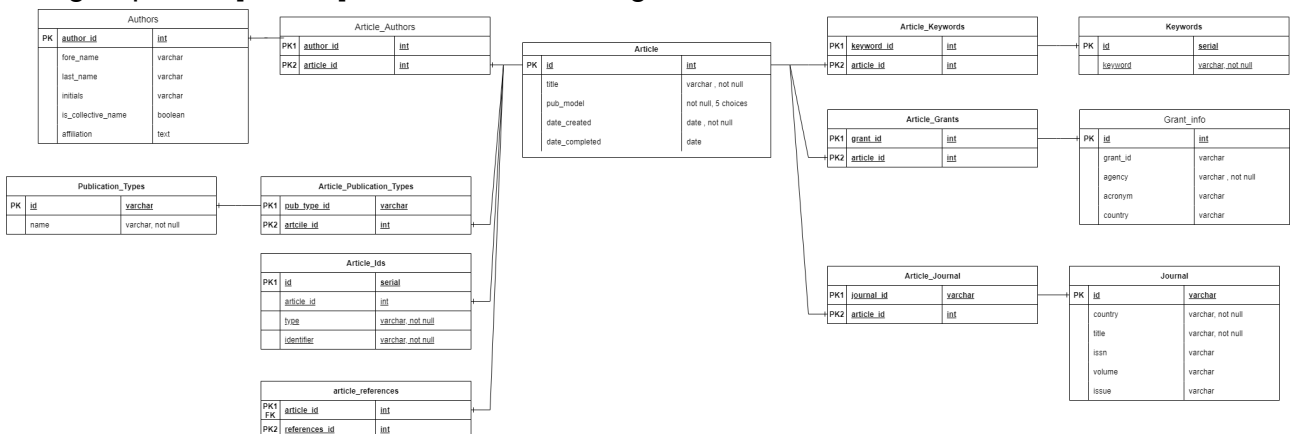
submit.rar

- sustc-api.jar
- GoodLoader.java

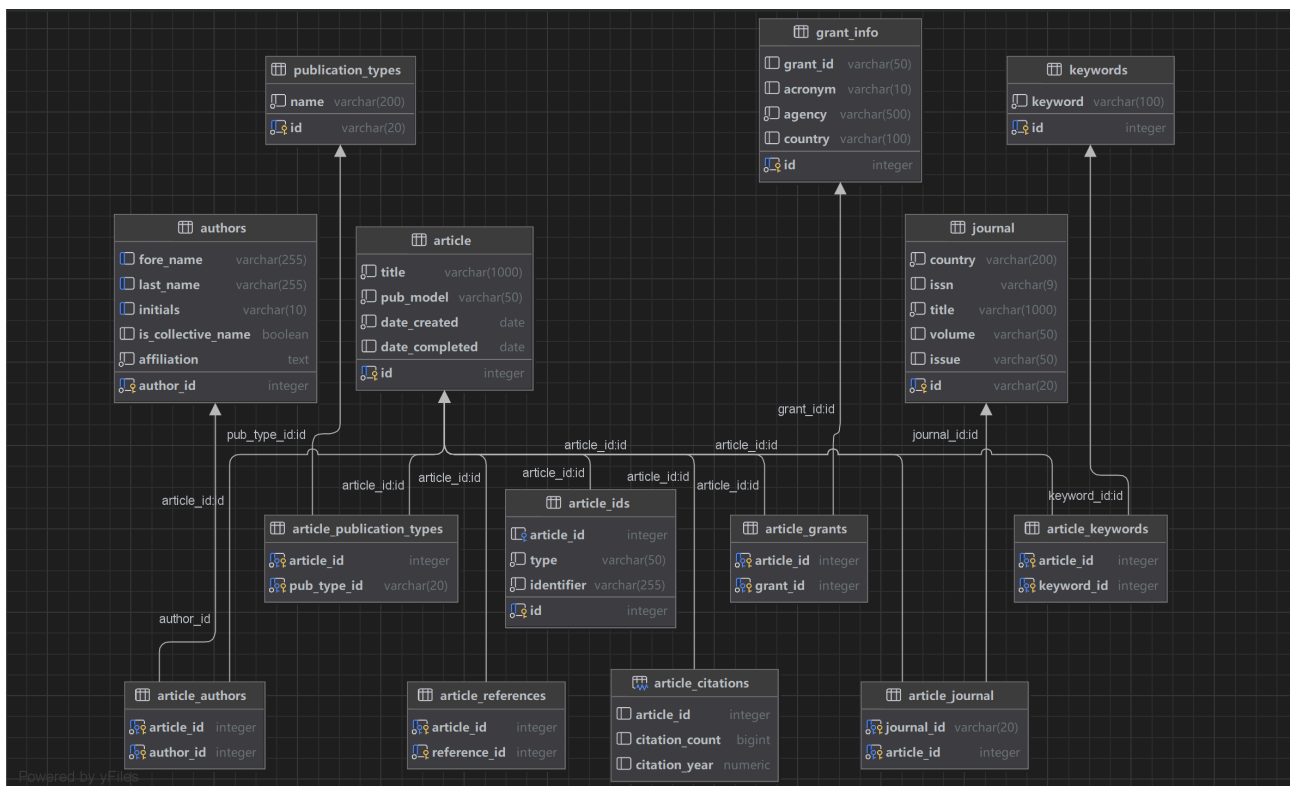
Task2.2

1. E-R diagram

Our group uses [drawio] to draw the E-R diagram, the screenshot is as follow :



2. The following graph is generated from the database diagram via the tool in Datagrip "Show Visualization".



3. Tables and Columns

- **Article Table:**
 id (Primary Key)
 title
 pub_model
 date_created
 date_completed
- **Article_Authors Table :**
 author_id (Primary Key)
 article_id (Primary Key)
- **Authors Table :**
 author_id (Primary Key)
 fore_name
 last_name
 initials
 is_collective_name
 affiliation
- **Article_Publication_Types Table :**
 pub_type_id (Primary Key)
 article_id (Primary Key)
- **Publication_Types Table :**
 id (Primary Key)
 name

- `Article_Ids` Table :
id (Primary Key, Serial)
article_id
type
identifier
- `article_references` Table :
article_id (Primary Key)
references_id (Primary Key)
- `Article_Keywords` Table :
keyword_id (Primary Key)
article_id (Primary Key)
- `Keywords` Table :
id (Primary Key, Serial)
keyword
- `Article_Grants` Table:
grant_id (Primary Key)
article_id (Primary Key)
- `Grant_info` Table :
id (Primary Key)
grant_id
agency
acronym
country
- `Article_Journal` Table :
journal_id (Primary Key)
article_id (Primary Key)
- `Journal` Table :
id (Primary Key)
country
title
issn
volume
issue

4. The database user creation and privilege descriptions

- Methods:
 - If the user has *admin* permissions, they can access all methods.
 - If the user has *journal_admin* permissions, they cannot access the `truncate` method, but can access all other methods.

- If the user has *user* permissions, they cannot access the `truncate` and `updateJournalName` methods, but can access all other methods.
- Table Permissions:
 - If the user has *admin* permissions, they have all permissions on all tables.
 - If the user has *journal_admin* permissions, they have select, insert, and update permissions on the `journal` table and the `article_journal` table, and select permissions on all tables.
 - If the user has *user* permissions, they only have select permissions on all tables.

Task2.3

Detailing the special design our group made.

1. We change the original "CREATE OR REPLACE VIEW" to "CREATE MATERIALIZED VIEW" in `GoodLoader` .

This way, it can avoid updating every time an operation is performed, and instead, fix the view, reducing the time needed for back-and-forth changes.

```
CREATE MATERIALIZED VIEW Article_Citations AS
SELECT
    ar.reference_id AS article_id, -- 被引用的文章 ID
    COUNT(ar.article_id) AS citation_count, -- 被引用的次数
    EXTRACT(YEAR FROM a.date_created) AS citation_year -- 引用发生的年份
FROM article_references ar
JOIN Article a ON ar.article_id = a.id -- 引用文章的创建日期
GROUP BY ar.reference_id, EXTRACT(YEAR FROM a.date_created)
```

2. We use **BFS** while solving the Bonus Task in `AuthorServiceImpl` .

The method `getMinArticlesToLinkAuthors` is designed to calculate the minimum number of articles needed to connect two authors, A and E, through their shared publications.

Essentially, this can be viewed as finding the shortest path between two nodes (authors) in an undirected graph, where articles serve as the edges connecting authors. The goal is to determine how many articles (or shared collaborations) are required to establish a connection between the two authors.

--Main Steps:

1. **Querying Articles:** First, the method queries all articles that authors A and E have contributed to, using SQL to retrieve their respective article IDs and author IDs from the `Article_Authors` table.

2. **Boundary Check:** If there are no articles in common between the authors, the method

immediately returns `-1`, indicating no connection through articles.

3. **Breadth-First Search (BFS):** The method uses BFS to explore authors layer by layer, trying to find the shortest path from A to E:

- **Queue:** A queue is maintained to process authors in order of traversal.

- **Visited Authors:** A `visitedAuthors` set ensures that no author is processed more than once, preventing redundant checks.

- **Distance Map:** A `distance` map stores the shortest number of articles connecting A to other authors.

4. **Fetching Authors for Articles:** As the BFS progresses, for each author, the method retrieves all articles they have written using `getArticlesByAuthor` and then fetches all authors associated with each article using `getAuthorsByArticle`. This helps in exploring potential paths through shared articles.

5. **Returning Results:** If a connection is found between authors A and E, the method returns the shortest path (number of articles). If no path exists, it returns `-1`.

-- Advantages:

1. **BFS Ensures Shortest Path:** BFS is an optimal algorithm for finding the shortest path in an unweighted graph, making it perfect for this task of calculating the minimum number of shared articles between authors.

2. **Prevents Redundant Computation:** The `visitedAuthors` set ensures that each author is processed only once, avoiding unnecessary recalculations and infinite loops.

3. **High Flexibility:** While the method is tailored to calculate the connection between two authors, it can easily be adapted for finding connections between multiple authors, offering great flexibility.

4. **Efficient Article and Author Queries:** The `getArticlesByAuthor` and `getAuthorsByArticle` methods efficiently fetch articles and authors, minimizing the number of database queries and improving performance.

5. **Clear and Modular Design:** The method follows a clear and modular structure, broken down into steps such as article retrieval, BFS traversal, distance calculation, and result return, making it easy to understand and extend.

```
public int getMinArticlesToLinkAuthors(Author A, Author E) {
    // 假设作者 A 和 E 已经通过某种方法获取到他们的 ID 或其他唯一标识
    String query =
        "SELECT aa.article_id, aa.author_id " +
        "FROM Article_Authors aa " +
        "WHERE aa.author_id IN (?, ?)";

    // 获取作者 A 和 E 的文章
    List<Integer> articlesByA = jdbcTemplate.queryForList(query, Integer.class,
        A.getAuthorId(), E.getAuthorId());
```

```

// 如果作者 A 和 E 没有文章, 返回 -1    if (articlesByA.isEmpty()) {
    return -1;
}

// 创建一个队列用于 BFS    Set<Integer> visitedAuthors = new HashSet<>();
Queue<Integer> queue = new LinkedList<>();
queue.offer(A.getAuthorId());
visitedAuthors.add(A.getAuthorId());

Map<Integer, Integer> distance = new HashMap<>();
distance.put(A.getAuthorId(), 0);

// BFS 遍历
while (!queue.isEmpty()) {
    int current = queue.poll();

    // 如果找到了目标作者, 返回到达目标的步骤数
    if (current == E.getAuthorId()) {
        return distance.get(current);
    }

    // 获取当前作者所参与的所有文章
    List<Integer> connectedArticles = getArticlesByAuthor(current);
    for (int articleId : connectedArticles) {
        // 对每个共同文章, 获取文章的所有作者
        List<Integer> authorsInArticle = getAuthorsByArticle(articleId);

        for (int authorId : authorsInArticle) {
            // 如果这个作者还没有访问过, 则加入队列
            if (!visitedAuthors.contains(authorId)) {
                visitedAuthors.add(authorId);
                distance.put(authorId, distance.get(current) + 1);
                queue.offer(authorId);
            }
        }
    }
}
}

```

```
// 如果无法连接两个作者，返回 -1    return -1;
}
```

The followings are detailed descriptions of every methods:

- `getArticleCitationsByYear` : The method uses a `PreparedStatement` to prevent SQL injection and sets the article ID (`id`) and year (`year`) as parameters for the query. The citation count is retrieved from the `ResultSet` , and if the citation count for the given year is found, it is returned. If no data is found, the method returns 0. The method also includes exception handling, throwing a `RuntimeException` in case of any SQL-related errors during the database connection or query execution.
- `addArticleAndUpdateIF` : First, it retrieves the creation year of the article from the provided `Article` object, then attempts to insert the article and its associated journal information into the database. After the insertion, the method executes two SQL queries: one to count the total number of articles in the journal for the given year, and another to sum the total citations for that journal in the following year (the year for impact factor calculation). The impact factor is computed as the total citations divided by the total number of articles in the journal. Finally, the method deletes the inserted article and journal association records to keep the database clean.
- `getArticlesByAuthorSortedByCitations` : The method retrieves all articles by a given author, sorted by their citation count in descending order. The method uses a `LEFT JOIN` to connect the `Article_Authors` table with the `all_Article_Citations` table, ensuring that articles without citation data (i.e., where `citation_count` is `null`) are still included, with citation counts treated as 0. The sorting is handled by the `COALESCE` function to use 0 if the citation count is `null` . Finally, the method returns the article IDs as an integer array.
- `getJournalWithMostArticlesByAuthor` : The method is designed to find the journal with the most articles published by a given author (identified by their first and last names). The method joins the `Journal` , `Article_Journal` , `Article` , `Article_Authors` , and `Authors` tables to retrieve all the articles of the author. It then groups the results by journal title and orders them by the article count in descending order, limiting the result to the journal with the most articles. If no results are found (i.e., the author has no articles or no qualifying journals), the method returns an empty string.
- `getCountryFundPapers` : The method `getCountryFundPapers` is designed to retrieve the IDs of all articles funded by a specific country. The method first performs an SQL query that joins the `grant_info` and `article_grants` tables, filtering by the country specified in the input. To ensure case-insensitive matching of the country name, the method uses the `LOWER()` function to convert both the input and the database values to lowercase. The result of the query is a list of article IDs, which is then converted into an integer array and returned.

- `getImpactFactor` : The method first executes a SQL query to retrieve the total number of citations for the specified journal in the given year. The query also filters articles based on their creation year, considering only articles published in the two years prior. Next, another SQL query is executed to retrieve the total number of articles published by the journal during the same time period (the two preceding years). The impact factor is calculated as the ratio of total citations to total articles. If the number of articles is zero, the impact factor is returned as 0. The method uses `Objects.requireNonNullElse` to handle potential `null` values in the query results, ensuring that the calculations proceed smoothly without exceptions.
- `updateJournalName` : The method starts by disabling auto-commit (`conn.setAutoCommit(false)`), which begins a transaction. This ensures that if any error occurs during the update process, all changes can be rolled back, maintaining data consistency. To avoid foreign key constraint violations during the update, the method first modifies the foreign key constraint (`article_journal_journal_id_fkey`) to be deferrable. This allows foreign key checks to be deferred until the transaction is committed, thus preventing conflicts during the update process. The method then updates the `Article_Journal` table, setting the journal ID for articles created after the specified year. Only relevant article records are updated based on the article creation year. Next, the method updates the `Journal` table, changing the journal's name and ID. If both the article and journal updates are successful, the transaction is committed (`conn.commit()`). The method returns `true` only if both the article and journal records are successfully updated.
- `getArticleCountByKeywordInPastYears` : The SQL query counts the articles associated with the given keyword, grouping by the article creation year and ordering by year in descending order. The query is executed using `jdbcTemplate.queryForList`, and the results are converted into an integer array. The method returns an array of integers, where each value represents the number of articles associated with the keyword for a given year.

Task2.4

This task is mainly coded in `Visual Studio Code` and `IntelliJ IDEA`.

We mainly use programming languages, such as `java`, `javascript`, `css`, `html`.

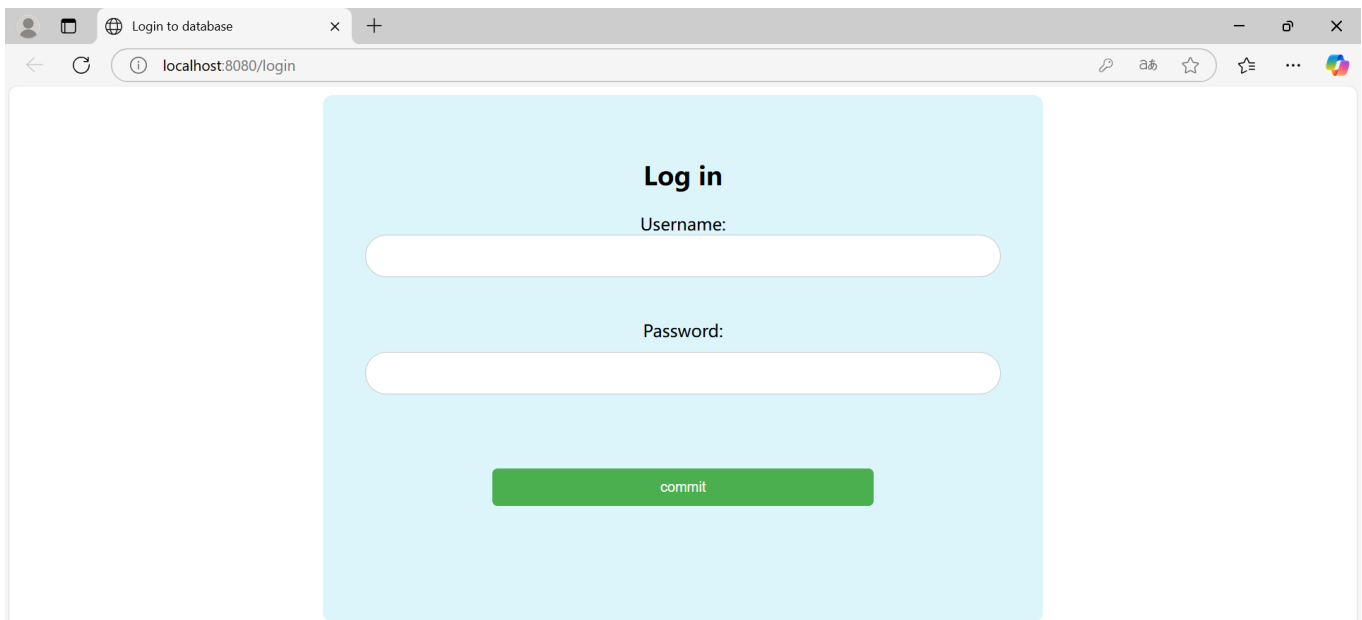
The submitted code files are parts of a `Vue.js` application with a Spring Boot backend. The application includes user authentication, role-based access control, and a dashboard with various services.

Here is a brief description of each frontend file:

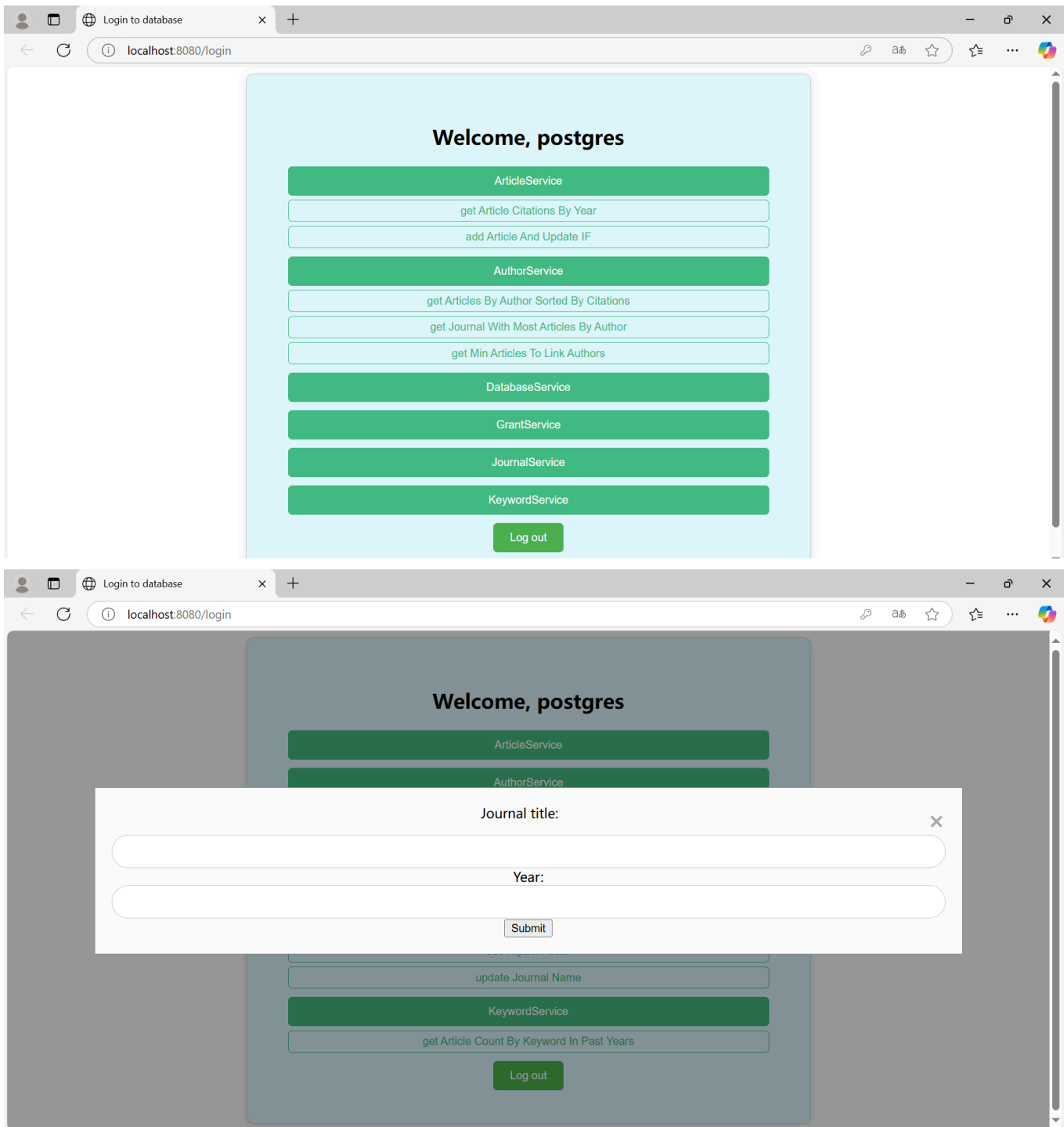
1. **PermissionControl.vue**: This component checks the user's role and displays content based on their permissions. If the user has access, it shows a welcome message; otherwise, it shows a permission denied message.

2. **Login.vue**: This component handles user login. It includes a form for entering the username and password. Upon successful login, it displays a welcome message and a list of service buttons. Each button toggles the display of service details. Each service corresponds to an interface in the Java file, and the service details represent the methods within that interface. When you click the sub-button below, a modal popup will appear prompting you to enter the required input for running the method. After entering the input, the backend `Spring Boot application` will respond and invoke the corresponding method.
3. **Logout.vue**: This component handles user logout. It includes a button that logs the user out and redirects them to the login page.
4. **index.js (router)**: This file configures the Vue Router. It defines routes for the dashboard, login, and logout pages. It also includes navigation guards to check for authentication and role-based access.
5. **index.js (store)**: This file configures the Vuex store. It includes state, mutations, actions, and getters for user authentication and role management.
6. **index.js (server)**: This file sets up an Express server for handling API requests. It includes a login endpoint that checks user credentials and returns user information.
7. **main.js**: This file is the entry point of the Vue.js application. It creates the Vue instance, uses the router and store, and mounts the app to the DOM.

All in all, based on the strict constraints of each key in the database, we have chosen to prompt the user for the required input data via a modal popup when the sub-button is clicked. This data will be passed to the corresponding methods in the `MyController` file of the backend Spring Boot application, thereby invoking the methods associated with each interface and obtaining the return values.



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/login'. The page content is a light blue rectangular box containing a login form. The form has the title 'Log in' in bold black text. Below the title, there are two input fields: the first is labeled 'Username:' and the second is labeled 'Password:'. Both fields are white with rounded ends and a thin grey border. At the bottom of the form is a green rectangular button with the text 'commit' in white. The browser's address bar and tabs are visible at the top of the window.



Here is a brief description of each backend file:

1. The **MyController** class is a Spring Boot controller class designed to handle HTTP requests and invoke corresponding service methods to perform various operations. This class interacts with the client through various RESTful API endpoints, passing client request parameters to the backend services for processing and eventually returning the results.
 - The class is annotated with `@RestController` and `@RequestMapping`, marking it as a RESTful controller and mapping all requests under the `/api` path.

- Various service classes are injected using the `@Autowired` annotation, which contain the actual business logic.
- Each method is annotated with `@GetMapping`, specifying the corresponding HTTP GET request path.
- Methods use the `@RequestParam` annotation to retrieve parameters passed by the client and call the respective service methods for processing, ultimately returning the results to the client.
- Logging: Each method logs the receipt and processing of requests to trace the request handling process.

2. This **MySpringBootApplication** class is the entry point for a Spring Boot application. Its main purpose is to bootstrap the entire Spring Boot application. Key functionalities include:

- `@SpringBootApplication` annotation: This is a composite annotation that includes `@Configuration`, `@EnableAutoConfiguration`, and `@ComponentScan`, used to auto-configure the spring application infrastructure.
- `@ComponentScan` annotation: Specifies the base packages `{"com.example.my_spring_boot_app", "io.pubmed.service.impl"}` for Spring to scan, ensuring that all components, services, and configurations in these packages are detected and registered.

In general, we have added numerous Gradle and Maven dependencies and merged the Spring Boot files with the files of project2. This facilitates the calling of methods and the returning of values.

Objectively speaking, we currently do not have a complete implementation of creating users in the frontend and corresponding access permissions in the backend database.

In the `DatabaseProject2` in the compressed file, there are `DatabaseLogin.java` and `DatabaseUserCreation.java`.

`DatabaseLogin.java` contains `validatePassword` method checks if the password meets certain criteria: at least 8 characters long, contains at least one uppercase letter, one lowercase letter, one digit, and only contains valid characters (letters, digits, underscores, asterisks, and dollar signs).

`DatabaseUserCreation.java` calls `validatePassword` to ensure the password is valid.

It determines the role based on the user type input and constructs a JDBC URL for connecting to a PostgreSQL database using admin credentials. It attempts to establish a connection to the database and create a new user with the specified username and password. It assigns roles and permissions to the new user based on the determined role.

```

✓ set role a3;
-- USER:
-- Should succeed (SELECT permission)
✓ SELECT * FROM article LIMIT 5;
✓ SELECT * FROM journal LIMIT 5;
✓ SELECT * FROM authors LIMIT 5;

-- Should fail (no INSERT permission)
❗ INSERT INTO article (id, title, pub_model, date_created)
VALUES ( id 99999999, title 'Test Article', pub_model 'Print', date_created '2024-03-21');

-- Should fail (no UPDATE permission)
UPDATE journal SET country = 'USA' WHERE id = '1';

```

] 错误: 对表 article 权限不够

```

✓ set role a2;
-- JOURNAL ADMIN:
-- Should succeed (SELECT, INSERT, UPDATE on journal tables)
✓ SELECT * FROM journal LIMIT 5;
✓ INSERT INTO journal (id, country, issn, title)
VALUES ( id 'TEST123', country 'USA', issn '1234-5678', title 'Test Journal');
✓ UPDATE journal SET country = 'Canada' WHERE id = 'TEST123';

-- Should succeed (SELECT, INSERT, UPDATE on article_journal)
✓ SELECT * FROM article_journal LIMIT 5;
✓ INSERT INTO article_journal (journal_id, article_id)
VALUES ( journal_id 'TEST123', article_id 1);

-- Should fail (no permission on other tables)
❗ INSERT INTO article (id, title, pub_model, date_created)
VALUES ( id 99999999, title 'Test Article', pub_model 'Print', date_created '2024-03-21');
DELETE FROM authors WHERE author_id = 1;

```

错误: 对表 article 权限不够

From these two files, we achieve the permission control as the followings:

- Methods:
 - If the user has *admin* permissions, they can access all methods.
 - If the user has *journal_admin* permissions, they cannot access the `truncate` method, but can access all other methods.
 - If the user has *user* permissions, they cannot access the `truncate` and `updateJournalName` methods, but can access all other methods.
- Table Permissions:
 - If the user has *admin* permissions, they have all permissions on all tables.

- If the user has *journal_admin* permissions, they have select, insert, and update permissions on the `journal` table and the `article_journal` table, and select permissions on all tables.
- If the user has *user* permissions, they only have select permissions on all tables.

How to use these Frontend and Backend

1. Run the Backend File :

Firstly, use IntelliJ IDEA to open the backend interface (DatabaseProject2) .

Open the `MySpringBootApplication.java` under this path `sustc-`

`spring_boot/src/main/java/com/example/my_spring_boot_app/MySpringBootApplication.java` .

Run this file and wait for a few seconds.

2. Run the Frontend File :

Firstly, use VSCode to open the frontend interface (my-vue-app) .

Then, click Ctrl+ J, and enter `npm run dev` in the command line.

3. Log in :

Click the generated `http://localhost:8080/login` link.

Enter `postgres` in the Username box and `huarui66` in the password.

If you do not enter these two, you will not be able to enter the next interface and will receive a warning that you have entered an invalid username and password.

4. Choose a method :

If you successfully log in, you will enter the welcome page, where you will first see many service buttons corresponding to the backend interfaces.

Click on a service button, and below it, there will be some sub buttons corresponding to the methods in each interface.

Clicking a sub button will pop up a pop-up window, prompting you to enter the data required for the corresponding interface.

After entering these data, click submit, and the backend terminal will display your input to this interface and the query results it returns

```
2024-12-22T22:55:24.736+08:00 INFO 16120 --- [my-spring-boot-app] [nio-8082-exec-2] o.s.web.servlet.DispatcherServlet : Completed
initialization in 4 ms
2024-12-22T22:55:24.955+08:00 INFO 16120 --- [my-spring-boot-app] [nio-8082-exec-2] c.e.my_spring_boot_app.MyController : Received
request for sum
a 54 b 66
result 120
2024-12-22T22:56:37.988+08:00 INFO 16120 --- [my-spring-boot-app] [nio-8082-exec-6] c.e.my_spring_boot_app.MyController : Received
request for getArticleCountByKeywordInPastYears
keyword Ovulation
result [4, 3, 1, 82, 13, 6, 2, 1, 1, 26, 2, 1, 3, 4, 4, 2]
2024-12-22T22:57:24.729+08:00 INFO 16120 --- [my-spring-boot-app] [io-8082-exec-10] c.e.my_spring_boot_app.MyController : Received
request for getCountryFundPapers
country Canada
result [95614, 456622, 487272, 758223, 1313743, 1351773, 1356796, 1359919, 1375757, 1376872, 1377041, 1393302, 1436497, 1493225, 1515936,
1599446, 1615118, 1636151, 1645677, 1654254, 1665781, 1685349, 1697219, 1717888, 1719454, 1722327, 1724068, 1868420, 1967903, 1972033,
1981173, 2107559, 2162234, 2282633, 2285889, 2302582, 2326096, 2408908, 2410814, 2428434, 2437174, 2438667, 2448708, 2449638, 2452992,
2456135, 2458544, 2460194, 2462189, 2478930, 2481740, 2482950, 2484181, 2551531, 2581174, 2591972, 2635220, 2745558, 2765879, 2779757,
2790461, 2804666, 2895320, 2942088, 2999643]
```

5. Log out :

You can log out when clicking the log out button at the bottom of the welcoming page. Then, you need to put the correct username and password again to reaccess into the welcome page.