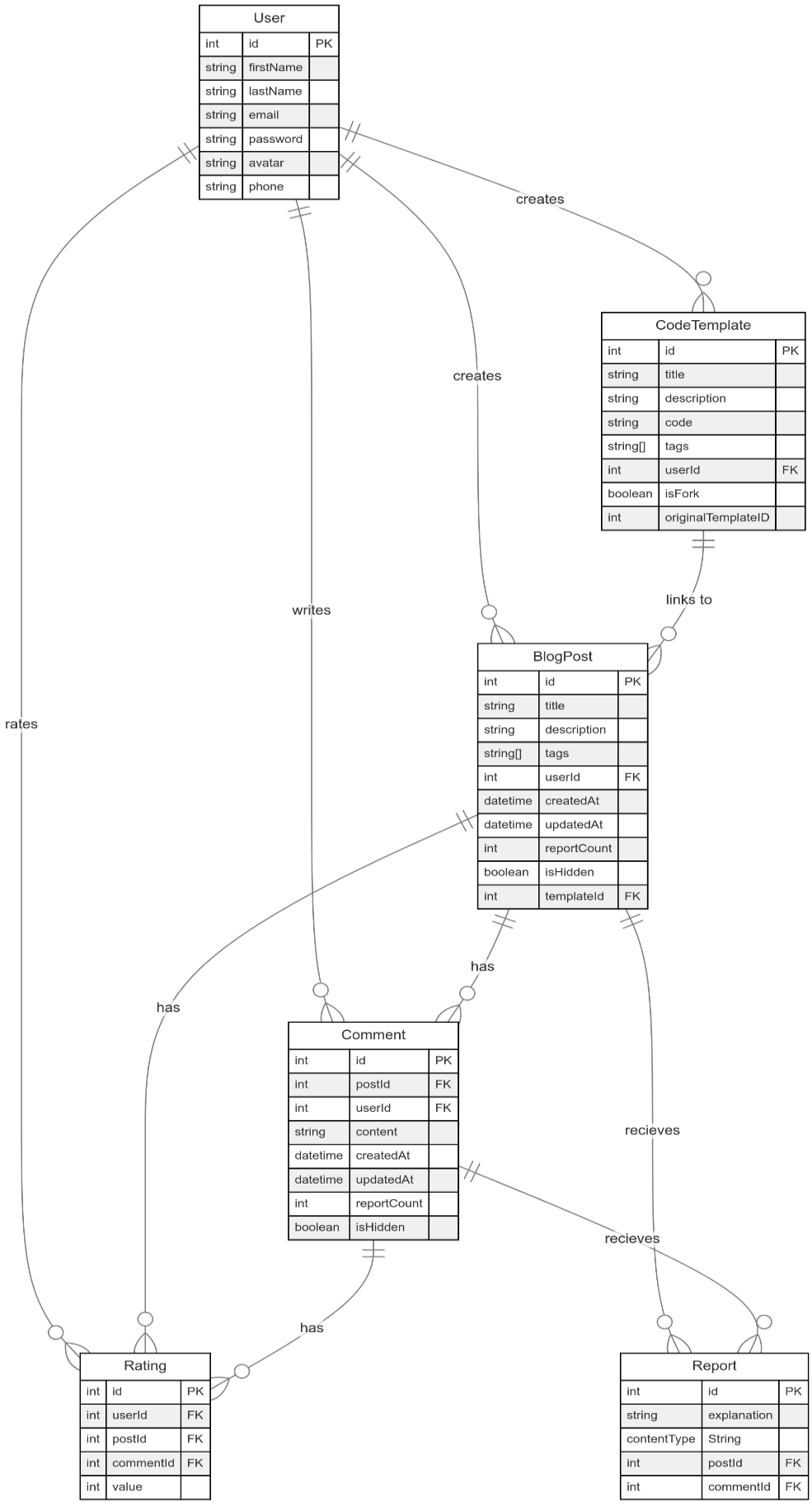
CSC309 - API DOCUMENTATION

ER DIAGRAM:


User
int id PK
string firstName
string lastName
string email
string password
string avatar
string phone
creates
CodeTemplate
int id PK
string title
string description
string code
string[] tags
int userId FK
boolean isFork
int originalTemplateID
creates
writes
links to
rates
BlogPost
int id PK
string title
string description
string[] tags
int userId FK
datetime createdAt
datetime updatedAt
int reportCount
boolean isHidden
int templateId FK
has
has
Comment
int id PK
int postId FK
int userId FK
string content
datetime createdAt
datetime updatedAt
int reportCount
boolean isHidden
recieves
recieves
has
Rating
int id PK
int userId FK
int postId FK
int commentId FK
int value
Report
int id PK
string explanation
contentType String
int postId FK
int commentId FK

Code Execution

execute.js: API endpoint used for executing code with optional inputs via stdin, with either output or error message being shown to the user. Only POST method is allowed. The required fields in the payload are language and code, with input being optional. eg.

request: {
  "language": "Python",
  "code": "print('hello')"
}

response: {
  "output": "hello\n"
}

User Authentication

1. **Sign Up**
    - **Endpoint**: POST /api/auth/signup
    - **Description**: Create a new user account.
    - **Payload**:

```
{
  "firstName": "John",
  "lastName": "Doe",
  "email": "john@example.com",
  "password": "securepassword"
}
```

    - **Response**:

```
{
  "message": "User created successfully."
}
```

2. **Log In**
    - **Endpoint**: POST /api/auth/login
    - **Description**: Log in a user and return a JWT token.
    - **Payload**:

```
{
  "email": "john@example.com",
  "password": "securepassword"
}
```

    - **Response**:

```
{
  "token": "JWT_TOKEN"
}
```

3. **Log Out**

- **Endpoint**: POST /api/auth/logout
- **Description**: Log out a user.
- **Response**:

```
{
  "message": "Logged out successfully."
}
```

Blog Posts

4. **Create Blog Post**
   - **Endpoint**: POST /api/posts
   - **Description**: Create a new blog post.
   - **Payload**:

```
{
  "title": "My First Post",
  "content": "This is the content of my first post.",
  "tags": ["introduction", "first"]
}
```

   - **Response**:

```
{
  "message": "Post created successfully."
}
```

5. **Edit Blog Post**
   - **Endpoint**: PUT /api/posts/:id
   - **Description**: Update an existing blog post.
   - **Payload**:

```
{
  "title": "Updated Title",
  "content": "Updated content.",
  "tags": ["updated"]
}
```

   - **Response**:

```
{
  "message": "Post updated successfully."
}
```

6. **Delete Blog Post**
   - **Endpoint**: DELETE /api/posts/:id
   - **Description**: Delete a blog post.
   - **Response**:

```
{
  "message": "Post deleted successfully."
```

}

7. **Get All Blog Posts**
    - **Endpoint**: GET /api/posts
    - **Description**: Retrieve a paginated list of all blog posts.
    - **Query Parameters**: page, limit
    - **Response**:

```
{
  "totalPosts": 50,
  "totalPages": 10,
  "currentPage": 1,
  "posts": [ /* Array of posts */ ]
}
```

8. **Get Single Blog Post**
    - **Endpoint**: GET /api/posts/:id
    - **Description**: Retrieve a specific blog post by ID.
    - **Response**:

```
 {
  "id": 1,
  "title": "My First Post",
  "content": "This is the content of my first post.",
  "tags": ["introduction", "first"]
}
```

Comments

9. **Create Comment**
    - **Endpoint**: POST /api/posts/:postId/comments
    - **Description**: Add a comment to a blog post.
    - **Payload**:

```
{
  "content": "This is a comment."
}
```

    - **Response**:

```
{
  "message": "Comment created successfully."
}
```

10. **Reply to Comment**
    - **Endpoint**: POST /api/comments/:commentId/replies
    - **Description**: Reply to a specific comment.
    - **Payload**:

```
{
  "content": "This is a reply."
}
```

- **Response**:

```
{
  "message": "Reply created successfully."
}
```

11. **Edit Comment**
    - **Endpoint**: PUT /api/comments/:id
    - **Description**: Edit an existing comment.
    - **Payload**:

```
{
  "content": "Updated comment content."
}
```

    - **Response**:

```
{
  "message": "Comment updated successfully."
}
```

12. **Delete Comment**
    - **Endpoint**: DELETE /api/comments/:id
    - **Description**: Delete a comment.
    - **Response**:

```
{
  "message": "Comment deleted successfully."
}
```

13. **Get Comments for a Post**
    - **Endpoint**: GET /api/posts/:postId/comments
    - **Description**: Retrieve a paginated list of comments for a specific post.
    - **Query Parameters**: page, limit
    - **Response**:

```
{
  "totalComments": 20,
  "totalPages": 4,
  "currentPage": 1,
  "comments": [ /* Array of comments */ ]
}
```

Ratings

14. **Rate Blog Post**
    - **Endpoint**: POST /api/posts/:postId/rate
    - **Description**: Rate a blog post.
    - **Payload**:

```
{
  "rating": 1 // or -1 for downvote
}
```

- **Response**:

```
{
  "message": "Post rated successfully."
}
```

15. **Rate Comment**
    - **Endpoint**: POST /api/comments/:commentId/rate
    - **Description**: Rate a comment.
    - **Payload**:

```
{
  "rating": 1 // or -1 for downvote
}
```

- **Response**:

```
{
  "message": "Comment rated successfully."
}
```

Templates and Reports

- Postman contains examples of request.

1. Create Template

Path: /api/templates/create.js
Method: POST
Description: Allows authenticated users to create a new template.

- **Request Headers**:
  - Authorization: Bearer token (JWT for user authentication).
- **Request Body**:
  - title (String, required): The title of the template.
  - explanation (String, required): Explanation of the template.
  - code (String, required): The code content of the template.
  - tags (String, required): Tags associated with the template (comma-separated).
- **Responses**:
  - 201 Created: Returns the created template.
  - 400 Bad Request: Missing required fields.
  - 401 Unauthorized: Missing or invalid token.
  - 405 Method Not Allowed: Incorrect HTTP method.

2. Search Templates

Path: /api/templates/search.js
Method: GET
Description: Allows anyone to search templates by title, explanation, code, or tags. Returns all templates if no search term is provided.

- **Query Parameters**:
  - search (String, optional): Term to search within templates.
  - page (Number, optional): Page number for pagination (default is 1).
  - limit (Number, optional): Number of templates per page (default is 10).
- **Responses**:
  - 200 OK: Returns matching templates with pagination data.
  - 500 Internal Server Error: Unexpected error during database query.
  - 405 Method Not Allowed: Incorrect HTTP method.

3. Get User's Saved Templates

Path: /api/templates/userSavedTemplates.js
Method: GET
Description: Retrieves all templates created or forked by the authenticated user. If search parameters are given, returns only templates matching those criteria.

- **Request Headers**:
  - Authorization: Bearer token (JWT for user authentication).
- **Query Parameters**:
  - search (String, optional): Term to search within user's templates.
  - page (Number, optional): Page number for pagination (default is 1).
  - limit (Number, optional): Number of templates per page (default is 10).
- **Responses**:
  - 200 OK: Returns user's templates with pagination data.
  - 401 Unauthorized: Missing or invalid token.
  - 405 Method Not Allowed: Incorrect HTTP method.

4. Get Blog Posts Containing a Template

Path: /api/templates/[tid]/blogPosts.js
Method: GET
Description: Retrieves all blog posts that reference a specific template.

- **Path Parameters**:
  - tid (Number, required): The template ID to find associated blog posts.
- **Query Parameters**:
  - page (Number, optional): Page number for pagination (default is 1).
  - limit (Number, optional): Number of blog posts per page (default is 10).

- **Responses**:
    - 200 OK: Returns blog posts containing the specified template with pagination data.
    - 400 Bad Request: Missing or invalid template ID.
    - 404 Not Found: No blog posts found.
    - 405 Method Not Allowed: Incorrect HTTP method.

5. Fork Template

- **Path:** /api/templates/[tid]/fork
- **Method:** POST
- **Description:** Allows authenticated users to create a new template by forking an existing one.
- **Request Headers:**
    - Authorization: Bearer <token>
- **Query Parameters:**
    - tid: Template ID (required)
- **Responses:**
    - 201 Created: New template created from the forked template.
    - 400 Bad Request: Missing ID.
    - 401 Unauthorized: Token may have expired.
    - 404 Not Found: Template not found.
    - 405 Method Not Allowed: Invalid request method.

6. Get Template by ID

- **Path:** /api/templates/[tid]/getTemplate
- **Method:** GET
- **Description:** Retrieves a template by its ID for use cases such as displaying a selected template in blog posts.
- **Query Parameters:**
    - tid: Template ID (required)
- **Responses:**
    - 200 OK: Returns the requested template.
        - Response Body: { template: <template object> }
    - 400 Bad Request: Missing ID.
    - 404 Not Found: Template not found.
    - 405 Method Not Allowed: Invalid request method.

7. Update or Delete Template

- **Path:** /api/templates/[tid]/
- **Method:** PUT / DELETE
- **Description:** Updates or deletes a template by its ID, ensuring the user ID matches that of the template owner.
- **Query Parameters:**
    - tid: Template ID (required)
- **Request Body (for PUT):**
    - title: Updated title of the template (required)
    - explanation: Updated explanation of the template (required)
    - code: Updated code of the template (required)

- tags: Updated tags of the template (required)

Body:

```
{

   "title": "apple",

   "explanation": "sauce",

   "code": "is",

   "tags": "good"

}
```

- **Responses:**
  - 200 OK:
    - For PUT: Returns the updated template.
      - Response Body: <updated template object>
    - For DELETE: Returns a message indicating the template was deleted.
      - Response Body: { message: "Template deleted" }
  - 400 Bad Request: Missing required fields or missing ID.
  - 403 Forbidden: User is not authorized to modify this template.
  - 404 Not Found: Template not found.
  - 401 Unauthorized: Invalid or expired token.
  - 405 Method Not Allowed: Invalid request method.

8. Get Template Code

- **Path:** /api/templates/[tid]/use
- **Method:** GET
- **Description:** Returns only the code portion of the template specified by its ID. This is used to display the code in the editor.
- **Query Parameters:**
  - tid: Template ID (required)
- **Responses:**
  - 200 OK: Returns the code from the template.
    - Response Body: { code: "<template code>" }
  - 400 Bad Request: Missing ID.
  - 404 Not Found: Template not found.
  - 405 Method Not Allowed: Invalid request method.

9. Submit Report

- **Path:** /api/reports/submitReport/[contentType]/[id]
- **Method:** POST
- **Description:** Allows anyone to report a BlogPost or Comment. An explanation for the report is required.
- **Path Parameters:**
    - contentType: The type of content being reported (blogpost or comment) (required).
    - id: The ID of the BlogPost or Comment being reported (required).
- **Request Body:**
    - explanation: The reason for the report (required).
- **Responses:**
    - 201 Created: Returns the created report.
        - Response Body: { /* report details */ }
    - 400 Bad Request: Missing ID, type, or explanation.
    - 404 Not Found: Blog Post or Comment not found.
    - 405 Method Not Allowed: Invalid request method.

Body:

```
{

   "explanation": "I don't like it lol"

}
```

10. Get Specific Report

- **Path:** /api/reports/admin/getReports/[id]
- **Method:** GET
- **Description:** Allows an admin to retrieve a specific report by its ID.
- **Path Parameters:**
    - id: The ID of the report being retrieved (required).
- **Headers:**
    - Authorization: Bearer token required for authentication.
- **Responses:**
    - 200 OK: Returns the report details.
        - Response Body: { /* report details */ }
    - 400 Bad Request: Missing ID.
    - 401 Unauthorized: Missing or invalid token.
    - 403 Forbidden: User is not an admin.
    - 404 Not Found: Report not found.
    - 405 Method Not Allowed: Invalid request method.

11. Get Reported Blogs

- **Path:** /api/reports/admin/getReports/blogReports
- **Method:** GET
- **Description:** Allows an admin to retrieve blogs with the highest number of reports, sorted in descending order.
- **Headers:**
    - Authorization: Bearer token required for authentication.
- **Query Parameters:**
    - page: Page number for pagination (optional, defaults to 1).
    - limit: Number of records per page (optional, defaults to 10).
- **Responses:**
    - 200 OK: Returns a list of reported blogs with pagination details.
    - 401 Unauthorized: Missing or invalid token.
    - 403 Forbidden: User is not an admin.
    - 405 Method Not Allowed: Invalid request method.

12. Get Reported Comments

- **Path:** /api/reports/admin/getReports/commentReports
- **Method:** GET
- **Description:** Allows an admin to retrieve comments with the highest number of reports, sorted in descending order.
- **Headers:**
    - Authorization: Bearer token required for authentication.
- **Query Parameters:**
    - page: Page number for pagination (optional, defaults to 1).
    - limit: Number of records per page (optional, defaults to 10).
- **Responses:**
    - 200 OK: Returns a list of reported comments with pagination details.
    - 401 Unauthorized: Missing or invalid token.
    - 403 Forbidden: User is not an admin.
    - 405 Method Not Allowed: Invalid request method.

13. Get Reports by Content Type and ID

- **Path:** /api/reports/admin/reportsByContent/[contentType]/[id]
- **Method:** GET
- **Description:** Allows an admin to retrieve all reports associated with a specific content type (blog post or comment) and its ID.
- **Headers:**
    - Authorization: Bearer token required for authentication.
- **Path Parameters:**
    - contentType: The type of content for which reports are requested (blogpost or comment).
    - id: The ID of the content (blog post or comment).
- **Responses:**
    - 200 OK: Returns a list of reports for the specified content type and ID.
    - 400 Bad Request: Missing ID or content type, or invalid content type.
    - 401 Unauthorized: Missing or invalid token.
    - 403 Forbidden: User is not an admin.
    - 404 Not Found: No reports found for the specified content type and ID.

    - 405 Method Not Allowed: Invalid request method.

14. Admin Actions on Content

- **Path:** /api/reports/admin/actions/[contentType]/[id]
- **Method:** PATCH
- **Description:** Allows an admin to flag inappropriate content or mark it as okay (pass).
- **Headers:**
    - Authorization: token required for authentication.
- **Path Parameters:**
    - contentType: The type of content to act upon (blogpost or comment).
    - id: The ID of the content (blog post or comment).
- **Query Parameters:**
    - action: The action to perform (flag or pass).
- **Responses:**
    - 200 OK: Returns the updated content after the action is performed.
    - 400 Bad Request: Missing ID, content type, or action; invalid content type; or invalid action.
    - 401 Unauthorized: Missing or invalid token.
    - 403 Forbidden: User is not an admin.
    - 404 Not Found: The specified content (blog post or comment) was not found.
    - 405 Method Not Allowed: Invalid request method.