blogsManagementServices - [includes] - blogsServices
collectionServices
tagServices

Req-020: createBlogs API - blogsServices

As part of the **blogsServices**, we need to create an API that will create and store the blogs in the db.

➔ Method type: **POST**

➔ **API:**
- name: **createBlogs**
- **api/v1/blogs/createBlogs**

➔ DB Bucket to be used: **blogs**.

➔ Path parameter should be empty.
➔ Parameters to be sent as part of the request body: title, img, overview, content, tags, and authorId.
➔ Other parameters: createdAt, updatedAt, version, and views should store default values when created.
➔ “content” needs to be encoded as Base64 before storing it into the db.

➔ **Response messages:**
- Response message should be “201 Created” in case the operation is successful.
- Response message should be “400 Bad Request” in case any required parameter is missing.
- Response message should be “500 Internal Server Error” in case of any error occurred in storing the data in db.

Req-021: getAllBlogs API - blogsServices

As part of the **blogsServices**, we need to create an API that will return the list of all the blogs present in the db.

➔ Method type: **GET**

➔ **API:**
- name: **getAllBlogs**
- **api/v1/blogs/getAllBlogs**

➔ DB Bucket to be used: **blogs**.

➔ Path parameter should be empty.
➔ Request body should be empty.
➔ Fields to be returned as part of the response: title, img, overview, version, views, tags, createdAt, updatedAt, and authorId.

➔ **Response messages:**
- Response message should be “200 Ok” in case the operation is successful.
- Response message should be “500 Internal Server Error” in case of any error occurred in connection with db.

Req-022: getBlogsById API - blogsServices

As part of the **blogsServices**, we need to create an API that will return all the details of a particular blog using its id from the db

➔ Method type: **GET**

➔ **API:**
- name: **getBlogsById**
- **api/v1/blogs/getBlogsById/{blogsId}**

➔ DB Bucket to be used: **blogs**.

➔ Parameter to be sent as part of the path: “blogsId”.
➔ Request body should be empty.
➔ Fields to be returned as part of the response: title, img, overview, content, version, views, tags, createdAt, updatedAt, and authorId.
➔ “content” needs to be decoded from Base64 format to normal text before storing it into the db.

➔ **Response messages:**
- Response message should be “200 Ok” in case the operation is successful.
- Response message should be “404 Not Found” in case the data for requested blogsId does not exist.
- Response message should be “500 Internal Server Error” in case of any error occurred in connection with db.

Req-023: updateBlogs API - blogsServices

As part of the **blogsServices**, we need to create an API that will update the information of blogs using their unique id present in the db.

➔ Method type: **PUT**

➔ **API:**
- name: **updateBlogs**
- **api/v1/blogs/updateBlogs/{blogsId}**

➔ DB Bucket to be used: **blogs**.

➔ Parameters to be sent as part of the path: “blogsId”, and “typeOfUpdate”.
➔ Parameters that can be sent as part of the request body: title, img, overview, content, and tags.
➔ “content” needs to be encoded as Base64 before storing it into the db.
➔ “updatedAt” needs to get updated and store the time of updating the note.
➔ “version” needs to get incremented based on the “typeOfUpdate” parameter as either “minor” or “major” if:
- “minor” then update the version as v1.1, v1.2,... so on.
- “major” then update the version as v2, v3,... so on.

➔ **Response messages:**
- Response message should be “201 Created” in case the operation is successful.
- Response message should be “400 Bad Request” in case any required parameter is missing.
- Response message should be “404 Not Found” in case the data for requested blogsId does not exist.
- Response message should be “500 Internal Server Error” in case of any error occurred in connection with db.

Req-024: deleteBlogs API - blogsServices

As part of the **blogsServices**, we need to create an API that will delete the blog using its unique id present in the db.

➔ Method type: **DELETE**

➔ **API:**
- name: **deleteBlogs**
- **api/v1/blogs/deleteBlogs/{blogsId}**

➔ DB Bucket to be used: **blogs**.

➔ Parameter to be sent as part of the path: “blogsId”.

➔ Request body should be empty.

➔ **Response messages:**
- Response message should be “200 Ok” in case the operation is successful.
- Response message should be “404 Not Found” in case the data for requested blogsId does not exist.
- Response message should be “500 Internal Server Error” in case of any error occurred in connection with db.

Req-025: updateBlogViews API - blogsServices

As part of the **blogsServices**, we need to create an API that will update the views count of the blog using its unique id present in the db.

➔ Method type: **PUT**.

➔ **API:**
- name: **updateBlogsViews**
- **api/v1/blogs/updateBlogsViews/{blogsId}**

➔ DB Bucket to be used: **blogs**.

➔ Parameter to be sent as part of the path: “blogsId”.
➔ Request body should be empty.
➔ Everytime the api is called, update the view count by 1.

➔ **Response messages:**
- Response message should be “202 Accepted” in case the operation is successful.
- Response message should be “404 Not Found” in case the data for requested blogsId does not exist.
- Response message should be “500 Internal Server Error” in case of any error occurred in connection with db.

Req-026: createTags API - tagServices

As part of the **tagServices**, we need to create an API that will create and store the tags in the db.

➔ Method type: **POST**.

➔ **API:**
- name: **createTags**
- **api/v1/tags/createTags**

➔ DB Bucket to be used: **tags**.

➔ Path parameter should be empty.
➔ Parameters to be sent as part of the request body: tagName.
➔ Other parameters: isValidated should store default value when created.

➔ **Response messages:**
- Response message should be “201 Created” in case the operation is successful.
- Response message should be “400 Bad request” in case the required parameter is missing.
- Response message should be “500 Internal Server Error” in case of any error occurred in connection with db.

Req-027: getAllTags API - tagServices

As part of the **tagServices**, we need to create an API that will return the list of all the tags present in the db.

➔ Method type: **GET**.

➔ **API:**
- name: **getAllTags**
- **api/v1/tags/getAllTags**

➔ DB Bucket to be used: **tags**.

➔ Path parameter should be empty.
➔ Request body should be empty.
➔ Fields to be returned as part of the response: tagName.

➔ **Response messages:**
- Response message should be “200 Ok” in case the operation is successful.
- Response message should be “500 Internal Server Error” in case of any error occurred in connection with db.

Req-028: getActiveTags API - tagServices

As part of the **tagServices**, we need to create an API that will return the list of all the active tags present in the db.

➔ Method type: **GET**.

➔ **API:**
- name: **getActiveTags**
- **api/v1/tags/getActiveTags**

➔ DB Bucket to be used: **tags**.

➔ Path parameter should be empty.
➔ Request body should be empty.
➔ Fields to be returned as part of the response: tagName.

➔ **Response messages:**
- Response message should be “200 Ok” in case the operation is successful.
- Response message should be “500 Internal Server Error” in case of any error occurred in connection with db.

Req-029: deleteTags API - tagServices

As part of the **tagServices**, we need to create an API that will delete the tag using its id present in the db.

➔ Method type: **DELETE**.

➔ **API:**
- name: **deleteTags**
- **api/v1/tags/deleteTags/{tagId}**

➔ DB Bucket to be used: **tags**.

➔ Parameter to be sent as part of the path: “tagId”.

➔ Request body should be empty.

➔ **Response messages:**
- Response message should be “200 Ok” in case the operation is successful.
- Response message should be “404 Not Found” in case the data for requested notesId does not exist.
- Response message should be “500 Internal Server Error” in case of any error occurred in connection with db.

Req-030: activateTags API - tagServices

As part of the **tagServices**, we need to create an API that will activate the tag to be available for the users to be used with their notes.

➔ Method type: **PUT**.

➔ **API:**
- name: **activateTags**
- **api/v1/tags/activateTags/{tagId}**

➔ DB Bucket to be used: **tags.**

➔ Parameter to be sent as part of the path: “tagId”.
➔ Request body should be empty.
➔ “isValidated” parameter value should be stored as true for the respective tagId.

➔ **Response messages:**
- Response message should be “202 Accepted” in case the operation is successful.
- Response message should be “404 Not Found” in case the data for requested notesId does not exist.
- Response message should be “500 Internal Server Error” in case of any error occurred in connection with db.

Req-031: deactivateTags API - tagService

As part of the **tagServices**, we need to create an API that will deactivate the tags, using its id present in the db.

➔ Method type: **PUT**.

➔ **API:**
- name: **deactivateTags**
- **api/v1/tags/deactivateTags/{tagId}**

➔ DB Bucket to be used: **tags.**

➔ Parameter to be sent as part of the path: “tagId”.
➔ Request body should be empty.
➔ “isValidated” parameter value should be stored as false for the respective tagId.

➔ **Response messages:**
- Response message should be “202 Accepted” in case the operation is successful.
- Response message should be “404 Not Found” in case the data for requested notesId does not exist.
- Response message should be “500 Internal Server Error” in case of any error occurred in connection with db.

Req-032: createCollections API - collectionServices

As part of the **collectionServices**, we need to create an API that will create and store the collections in the db.

➔ Method type: **POST**.

➔ **API:**
- name: **createCollections**
- **api/v1/collections/createCollections**

➔ DB Bucket to be used: **collections**.

➔ Path parameter should be empty.
➔ Parameters to be sent as part of the request body: title, overview, and tags.
➔ Other parameters: collectionList, createdAt, and views should store default value when created.

➔ **Response messages:**
- Response message should be “201 Created” in case the operation is successful.
- Response message should be “400 Bad Request” in case any required parameter is missing.
- Response message should be “500 Internal Server Error” in case of any error occurred in storing the data in db.