Software Quality Engineering

DEPARTMENT OF SOFTWARE ENGINEERING (SE)

SECTION: BS(SE)-G

SUBMITTED TO:

Mam Saba Kanwal

SUBMITTED BY:

Ushna Nadeem (21i-1225)

Amna Shahid (21i-1148)

Fatima Qurban (21i-1195)

DATE OF SUBMISSION:

December 11th, 2024

Table of Contents

adminRoutes.js

1. User Management

1.1. Use Case A: View All Users

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Fetch all users	Users exist in the database	No users in the database	At least one user	Empty database

1.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC1	Fetch all users from the database	List of users in JSON format	V1
TC2	No users in the database	Empty list or appropriate message displayed	B1

1.2. Use Case B: Disable/Block a User

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Block/unblock a user	Valid user ID and toggle provided	Invalid user ID or missing toggle	User ID exists and toggle provided	User ID does not exist or missing

1.3. Test Case Design Against Use Case B

Test Case ID	Description	Expected Outcome	New Tags Covered
TC3	Block a valid user	"User blocked successfully" message	V2
TC4	Unblock a valid user	"User unblocked successfully" message	V3
TC5	Try to block yourself	"Cannot block yourself" message	B2
TC6	Block with invalid user ID	"User not found" message	B3
TC7	Block with missing toggle value	"Bad request" message	B4

---

2. Blog Management

2.1. Use Case A: View All Blog Posts

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries

Fetch all blog posts	Blog posts exist in the database	No blog posts in the database	At least one blog post	Empty database

2.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC8	Fetch all blog posts	List of blog posts in JSON format	V4
TC9	No blog posts in the database	Empty list or appropriate message displayed	B5

2.2. Use Case B: View a Specific Blog Post

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
View specific blog post	Valid blog post ID provided	Invalid blog post ID provided	Blog post ID exists and valid	Blog post ID does not exist

2.2.1. Test Case Design Against Use Case B

Test Case ID	Description	Expected Outcome	New Tags Covered
TC10	Fetch blog post with valid ID	Blog post details in JSON format	V5
TC11	Fetch blog post with invalid ID	"Blog post not found" message	B6

2.3. Use Case C: Disable a Blog Post

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Disable/enable a blog post	Valid blog post ID and toggle provided	Invalid blog post ID or missing toggle	Blog post ID exists and toggle provided	Blog post ID does not exist or missing

2.3.1. Test Case Design Against Use Case C

Test Case ID	Description	Expected Outcome	New Tags Covered
TC12	Disable a blog post	"Blog post disabled successfully" message	V6
TC13	Enable a blog post	"Blog post enabled successfully" message	V7

TC14	Disable a blog post already in the same state	"Blog post is already disabled/enabled" message	B7
TC15	Disable with invalid blog post ID	"Blog post not found" message	B8
TC16	Disable with missing toggle value	"Bad request" message	B9

---

authmiddleware.js

1. Token-Based Authentication

1.1. Use Case A: Token-Based Authentication

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Verify user token	Token is valid and present	Token is missing or invalid	JWT token matches server secret	No token or malformed token

1.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC1	Authenticate with valid token	User is authenticated successfully	V1
TC2	Authenticate without a token	"No token, authorization denied" message	B1
TC3	Authenticate with invalid token	"Token is not valid" message	B2
TC4	Authenticate with expired token	"Token is not valid" message	B3

---

blogRoutes.js

1. Blog Creation

1.1. Use Case A: Create a New Blog Post

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Create blog post	All required fields (title, content, author) are provided	Missing required fields	Fields meet constraints (e.g., length, format)	Fields exceed constraints (e.g., length, format)

1.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC1	Create a blog post with valid fields	Blog post created successfully	V1
TC2	Create a blog post with missing fields	Error message indicating missing fields	B1
TC3	Create a blog post with fields exceeding constraints	Error message indicating invalid input	B2

---

2. View Blog Posts

2.1. Use Case A: Retrieve All Blog Posts

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Fetch all blogs	Blogs exist in the database	No blogs in the database	Database contains one or more blogs	Database is empty

2.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC4	Retrieve all blogs when blogs exist	List of blogs in JSON format	V2
TC5	Retrieve all blogs when no blogs exist	Empty list or appropriate message displayed	B3

2.2. Use Case B: Retrieve Blogs with Pagination and Sorting

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Retrieve blogs with pagination/sorting	Valid pagination and sorting parameters provided	Invalid or missing pagination/sorting parameters	Pagination and sorting parameters are correct	Pagination and sorting parameters are incorrect

2.2.1. Test Case Design Against Use Case B

Test Case ID	Description	Expected Outcome	New Tags Covered
TC6	Retrieve blogs with valid pagination/sorting	Blogs retrieved with pagination and sorting applied	V3

TC7	Retrieve blogs with invalid pagination/sorting parameters	Error message indicating invalid parameters	B4

3. Blog Updates

3.1. Use Case A: Update a Blog Post

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Update a blog post	Authorized user and valid update fields	Unauthorized user or invalid update fields	User is authorized; update fields valid	User unauthorized or update fields invalid

3.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC8	Update a blog post as authorized user	Blog post updated successfully	V4
TC9	Update a blog post as unauthorized user	Error message indicating unauthorized access	B5
TC10	Update a blog post with invalid fields	Error message indicating invalid fields	B6

4. Blog Deletion

4.1. Use Case A: Delete a Blog Post

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Delete a blog post	Authorized user and valid blog post ID	Unauthorized user or invalid blog post ID	Blog post ID exists in database	Blog post ID does not exist in database

4.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC11	Delete a blog post as authorized user	Blog post deleted successfully	V5
TC12	Delete a blog post as unauthorized user	Error message indicating unauthorized access	B7

5. Comment Management

5.1. Use Case A: Add a Comment to a Blog Post

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Add comment to a blog	Valid blog post ID and comment details provided	Invalid blog post ID or comment details missing	Blog ID and comment details are valid	Blog ID or comment details invalid

5.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC13	Add a comment to a valid blog post	Comment added to blog post	V6
TC14	Add a comment to an invalid blog post	Error message indicating invalid blog post	B8

Index.js

1. User Authentication

1.1. Use Case A: Register a New User

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Register a new user	All fields provided (username, email, password)	Missing one or more required fields	Fields meet constraints (e.g., valid email format, password length)	Fields exceed constraints or invalid formats

1.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC1	Register with all valid fields	User created successfully	V1
TC2	Register with missing fields	Error message indicating missing fields	B1
TC3	Register with invalid email format	Error message indicating invalid email	B2

1.2. Use Case B: Log In a User

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Log in a user	User exists, correct password, not blocked	User does not exist, incorrect password, or user is blocked	Fields match stored credentials	Fields mismatch stored credentials

1.2.1. Test Case Design Against Use Case B

Test Case ID	Description	Expected Outcome	New Tags Covered
TC4	Log in with valid credentials	Logged in successfully, access token generated	V2
TC5	Log in with incorrect password	Error message indicating incorrect password	B3
TC6	Log in with non-existent user	Error message indicating user not found	B4
TC7	Log in with blocked user	Error message indicating user is blocked	B5

2. Middleware and Routing

2.1. Use Case A: Apply Middleware to Protected Routes

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Apply authMiddleware	Token exists and is valid	Token missing or invalid	Token matches server secret	Token is expired or malformed

2.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC8	Access protected route with valid token	Access granted	V3
TC9	Access protected route with missing token	Error message indicating token is required	B6
TC10	Access protected route with invalid token	Error message indicating invalid token	B7

3. Server Setup

3.1. Use Case A: Start Express Server

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Start server	MongoDB is connected successfully	MongoDB connection fails	Environment variables are correct	Environment variables are missing

3.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC11	Start server with successful DB connection	Server starts successfully on port 3000	V4
TC12	Start server with failed DB connection	Error message indicating connection failure	B8
TC13	Start server with missing environment variables	Error message indicating configuration error	B9

---

searchRoutes.js

1. Search Functionality

1.1. Use Case A: Search Blogs

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Search for blogs	Query text provided, valid pagination/sorting params	Missing query text, invalid pagination/sorting params	Query matches title or content of blogs	Query doesn't match any blogs

1.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC1	Search with valid query and parameters	Blogs matching query text are returned	V1
TC2	Search with valid query but no matching blogs	Empty result set	B1
TC3	Search with missing query text	Empty result set	B2
TC4	Search with invalid pagination or sorting params	Error message indicating invalid parameters	B3

---

searchRoutes.js

1. Search Blogs with Filters and Pagination

1.1. Use Case A: Search Blogs

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Search for blogs	Query text, valid pagination, and sorting provided	Missing query text, invalid parameters	Pagination and sorting are within valid ranges	Pagination or sorting exceed constraints

1.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC1	Search with valid query and parameters	Blogs matching query text are returned	V1
TC2	Search with query text but no matching blogs	Empty result set	B1
TC3	Search with missing query text	Empty result set	B2
TC4	Search with invalid pagination/sorting parameters	Error message indicating invalid parameters	B3

---

2. Return Empty Results When No Query is Provided

2.1. Use Case A: Empty Query

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
No query text	Query text is not provided	Query text is present	Query text field is completely absent	Query text is malformed

2.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC5	Search without providing query text	Empty result set	B4

---

3. Handle Errors Gracefully

3.1. Use Case A: Server Errors

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Server-side handling	Server-side operations succeed	Server-side operations fail	Environment and dependencies are correctly set	Missing or misconfigured dependencies

3.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC6	Server error occurs during operation	Error message indicating server failure	B5

---

userRoutes.js

1. View All Users

1.1. Use Case A: Fetch All Users

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Fetch all users	Users exist in the database	No users in the database	At least one user exists	Empty database

1.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC1	Fetch all users when users exist	List of users in JSON format	V1
TC2	Fetch users when no users exist	Empty list or appropriate message displayed	B1

---

2. View a Specific User

2.1. Use Case A: Fetch Specific User by ID

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Fetch a specific user	Valid user ID provided	Invalid or non-existent user ID	User ID exists in the database	User ID does not exist in the database

2.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC3	Fetch user with a valid ID	User details in JSON format	V2
TC4	Fetch user with an invalid ID	"Cannot find user" message displayed	B2

---

3. Update User Information

3.1. Use Case A: Update User Details

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Update user details	Valid user ID and valid fields	Invalid user ID or invalid fields	User ID exists and fields are correct	User ID does not exist or fields invalid

3.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC5	Update user with valid data	User details updated successfully	V3
TC6	Update user with invalid ID	"Cannot find user" message displayed	B3
TC7	Update user with invalid fields	"Bad request" or validation error message	B4

---

4. Delete User

4.1. Use Case A: Delete User by ID

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Delete a user	Valid user ID provided	Invalid or non-existent user ID	User ID exists in the database	User ID does not exist in the database

4.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC8	Delete user with valid ID	"Deleted user" message displayed	V4
TC9	Delete user with invalid ID	"Cannot find user" message displayed	B5

---

5. Follow a User

5.1. Use Case A: Follow a User

Conditions	Valid Partitions	Invalid Partitions	Valid Boundaries	Invalid Boundaries
Follow another user	Valid IDs of both users provided	Invalid or non-existent user IDs	User IDs exist, and they are different	One or both user IDs do not exist

5.1.1. Test Case Design Against Use Case A

Test Case ID	Description	Expected Outcome	New Tags Covered
TC10	Follow a valid user	"Successfully followed the user" message	V5
TC11	Follow with invalid user ID	"User not found" message displayed	B6
TC12	Attempt to follow oneself	"You cannot follow yourself" message displayed	B7
TC13	Attempt to follow an already followed user	"You are already following this user" message displayed	B8