BLOG APPLICATION

EPIC: Simple Online Blog Application

- [US] Login page

Type: Story

Owner: Frontend Developer

Description:

Implement user login functionality that allows registered users to authenticate using email and password and access the blog system.

Design: [Attach design link]

Acceptance Criteria:

- Scenario 1: Login Page Display

  Given the user navigates to /login

  Then email and password fields are displayed

  And a login button is visible

- Scenario 2: Successful Login

  Given a registered user exists

  When valid credentials are submitted

  Then the message "Login Successful" is displayed

  And the user is redirected to /posts

- Scenario 3: Invalid Credentials

  Given invalid email or password

  When login is attempted

  Then an error message "Invalid email or password" is displayed

-- [FE][US] Login page

Type: Sub-task

Owner: Frontend Developer

Acceptance Criteria:

- Scenario 1: UI Rendering

  Given user navigates to login page

  Then email and password fields are displayed

  And login button is visible

- Scenario 2: Successful Login

  Given valid credentials

  When login button is clicked

  Then the message "Login Successful" is displayed

  And the user is redirected to /posts

- Scenario 3: Error Handling

  Given invalid credentials

  Then error message "Invalid email or password" is displayed

-- [BE][US] Login page

Type: Sub-task

Owner: Backend Developer

API Endpoint:

  POST /api/auth/login

Acceptance Criteria:

- Scenario 1: Successful Authentication

  Given a registered user exists

  When valid email and password are sent to /api/auth/login

Then HTTP 200 is returned

And a JWT token is included in the response

And user basic information is returned

- Scenario 2: Invalid Credentials

Given invalid email or password

When login request is made

Then HTTP 401 is returned

And error message is provided

- [US] View Posts List

Type: Story

Owner: Frontend Developer

Description:

Users can view a paginated list of blog posts.

Users can search blog posts by keyword (title/summary/author).

Design: [Attach design link]

Acceptance Criteria:

- Scenario 1: Display Posts

Given posts exist

When user navigates to /posts

Then a paginated list of posts is displayed

And each post shows title, summary, author, created date and updated date

- Scenario 2: Display empty Posts

Given posts are not exist or not match the search condition

When user navigates to /posts

Then a empty list is displayed

- Scenario 3: Pagination

  Given multiple pages of posts exist

  When user changes page

  Then corresponding posts are displayed

- Scenario 4: Search Field Visible

  Given user is on posts page

  Then a search input field is available

- Scenario 5: Perform Search (case insensitive)

  Given user enters a keyword

  When search is triggered

  Then posts matching title or summary are displayed

- Scenario 6: Navigation to Detail

  When user clicks a post

  Then user is redirected to /posts/{id}

-- [FE][US] View Posts List

Type: Sub-task

Owner: Frontend Developer

Acceptance Criteria:

- Scenario 1: Display Posts

  Given user navigates to /posts

  When API returns posts

  Then posts are displayed with title, summary, author, created date and updated date

- Scenario 2: Pagination UI

  Given multiple pages exist

  When user clicks next page

Then next set of posts is fetched

- Scenario 3: Search Input

  Given user is on posts page

  Then a search input field is visible with maxlenght (50)

- Scenario 4: Search Execution

  Given user enters a keyword

  When user stops typing (debounced 300–500ms)

  Then list updates dynamically

- Scenario 5: Navigation

  When user clicks a post

  Then user is redirected to /posts/{id}

-- [BE][US] View Posts List

Type: Sub-task

Owner: Backend Developer

API Endpoint:

  GET /api/posts?search={keyword}&page={page}&limit={limit}

Acceptance Criteria:

- Scenario 1: Fetch Posts

  Given posts exist in DB

  When /api/posts is requested

  Then HTTP 200 is returned

  And paginated posts list is returned

  And response includes total count

- Scenario 2: Pagination

  Given page and limit parameters

Then only relevant posts are returned

- Scenario 3: Keyword Search (Case Insensitive)

  Given posts exist

  When search parameter is provided

  Then posts matching title or content are returned

- [US] View Blog Post Details

Type: Story

Owner: Frontend Developer

Description:

Users can view full details of a selected blog post.

Design: [Attach design link]

Acceptance Criteria:

- Scenario 1: View Post

  Given a post exists

  When user navigates to /posts/{id}

  Then full title and content are displayed

  And author and timestamps are shown

- Scenario 2: Post Not Found

  Given invalid post ID

  When user attempts to access it

  Then system shows 404 or Not Found message

- Scenario 3: Edit Visibility

  Given authenticated author

  Then Edit button is visible

  Otherwise it is hidden

-- [FE][US] View Blog Post Details

Type: Sub-task

Owner: Frontend Developer

Acceptance Criteria:

- Scenario 1: Display Content

  Given user navigates to /posts/{id}

  When API returns data

  Then full title and content are displayed

  And author and dates are shown

- Scenario 2: Edit Visibility

  Given user is author

  Then Save button is visible

  And Cancel button is visible

- Scenario 3: Edit Visibility

  Given user navigates to /posts/{id}

  When user click <- button

  Then user is redirect to Posts List page

-- [BE][US] View Blog Post Details

Type: Sub-task

Owner: Backend Developer

API Endpoint:

  GET /api/posts/{id}

Acceptance Criteria:

- Scenario 1: Retrieve Post

Given post exists

When /api/posts/{id} is called

Then HTTP 200 is returned

And full post details are included

- Scenario 2: Not Found

  Given post does not exist

  Then HTTP 404 is returned

- [US] Edit Blog Post

Type: Story

Owner: Frontend Developer

Description:

Authenticated authors can edit their blog posts.

Design: [Attach design link]

Acceptance Criteria:

- Scenario 1: Access Edit Page

  Given authenticated author

  When navigating to /posts/{id}

  Then form is prefilled with existing data

- Scenario 2: Successful Update

  Given valid updated data

  When user submits changes

  Then post is updated

  And updatedAt field is modified

  And the message "Update successfully" is displayed

  And user is redirected to post detail page

- Scenario 3: Cancel changes

  Given user is on Detail page

  When user cancel changes

  Then post data is reverted

- Scenario 4: Unauthorized Access

  Given unauthenticated user

  When update is attempted

  Then HTTP 401 is returned

  And user is redirected to login page

-- [FE][US] Edit Blog Post

Type: Sub-task

Owner: Frontend Developer

Acceptance Criteria:

- Scenario 1: Prefill Data

  Given user opens /posts/{id}

  Then existing data is prefilled

- Scenario 2: Submit Update

  When user update the post detail with valid data

  Then Save button is clicked

  And the message "Update successfully" is displayed

- Scenario 3: Cancel changes

  When user update the post detail

  Then Cancel button is clicked

  And the changes is reverted

Validation:

Title: String, maxlenght: 50

Content: String, maxlenght: 1000

-- [BE][US] Edit Blog Post

Type: Sub-task

Owner: Backend Developer

API Endpoint:

PUT /api/posts/{id}

Acceptance Criteria:

- Scenario 1: Successful Update

  Given authenticated author

  When valid data is submitted

  Then HTTP 200 is returned

  And post is updated

  And updatedAt is modified

- Scenario 2: Invalid data Update

  Given authenticated author

  When invalid data is submitted

  Then HTTP 400 is returned

  And the error message is returned

- Scenario 3: Unauthorized

  Given unauthenticated user

  Then HTTP 401 is returned

- Scenario 4: Forbidden

  Given authenticated but not author

  Then HTTP 403 is returned