Final Draft: Fitz

Creators: Wesley Vance Blair, Katarina Marsteller

1. Detailed Plan Documentation

Below are the specific components and functionalities that will be included in the Fitz application.

List of React components:

Below is a list of React components we plan to implement into Fitz, including their functionalities and interactions.

addUserToDB

- Add new user to the database
- Include name, username, email address
- Utilizes UserSchema model

addPostToDB

- Add new post to the database
- Include username, caption, image, comments, status
- Utilizes PostSchema model

addContestToDB

- Add new contest to the database
- Include contest name, description, start date, end date, entries
- Utilizes ContestSchema model

addVoteToContestPost

- Adds a new vote from a user to the contest in the database
- Utilizes ContestSchema model

addContestToDB

- User sets the contest status to concluded
- Users can no longer enter or vote for posts
- Results and winners are determined
- Utilizes ContestSchema model

isUserAuthenticated

- Checks for cookie and verifies token validity with validateToken

isUser

- Checks if the user has a “user” role through the decoded token information

isAdmin

- Checks if the user has a “admin” role through the decoded token information

PostSchema

- Model for displaying the post data on the front-end
- Include username, caption, image, comments

ContestSchema

- Model for displaying the contest data on the front-end
- Include contest name, description, start date, end date, entries

UserSchema

- Model for user
- Include name, username, email address

Functions and Logic:

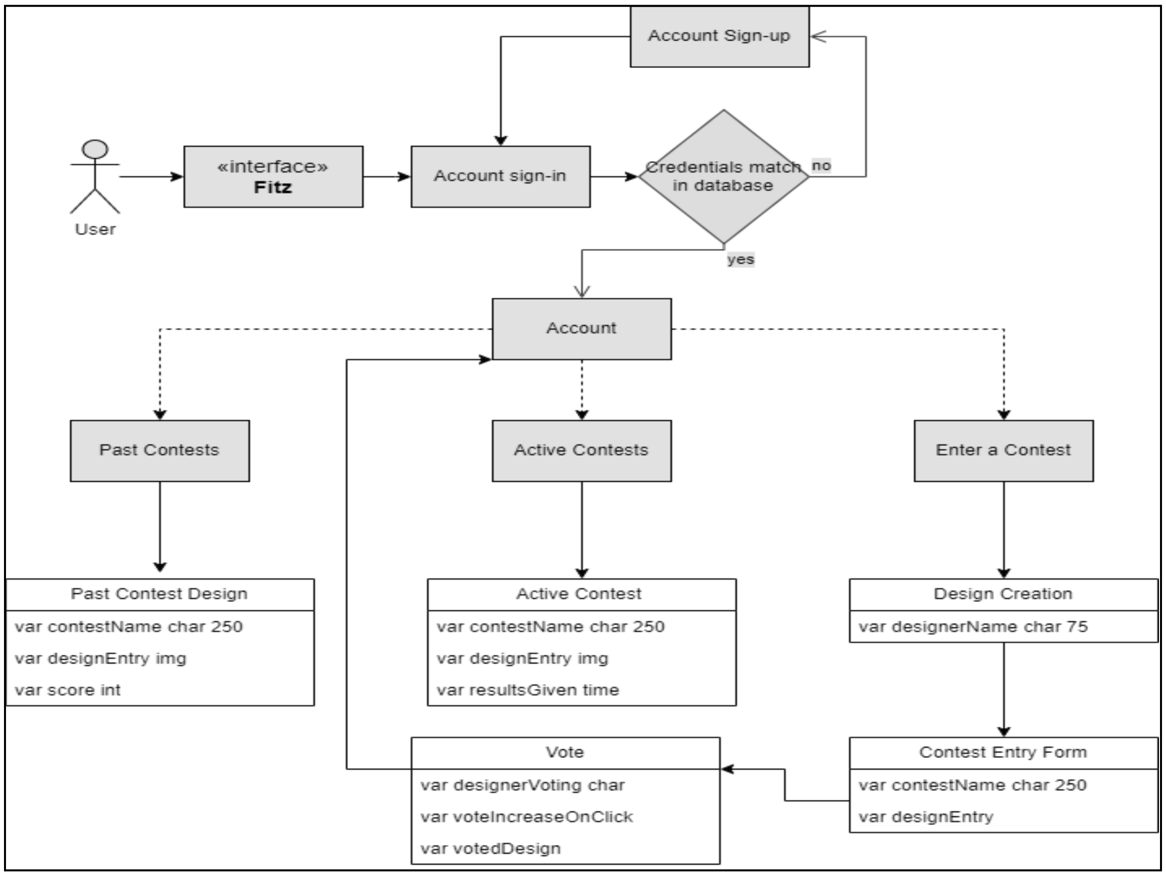

Fig 1.1 : Diagram of intended features for Fitz Application

API Request-Response Formats:

Specify the formats of API requests and responses that your frontend will expect and handle.

generateToken

○ Creates JWT from the json web token library

validateToken

○ Verifies the validity of the JWT

Endpoint Routes:

Define the routes and endpoints that will be utilized for the API integration, including the purpose and expected data for each endpoint.

/createPost

○ Handling post creation. Each post will be created with a unique ID. The username of the user posting will be taken in, along with a caption, image, and comment section, which will initially be empty.
○ POST
○ Request: body - {“uniquePostID”: “id”, “user”: “username”, “caption”: “caption text”, “image”: “URL”, “commentSection”: []}
○ Response: status – 201, message: “Post successfully created”

· **/createContest**

○ Creation of contests which will feature a unique ID as well. The contest name, start date, end date, and description will need to be entered. There will be a list of entries that will initially be blank.
○ POST
○ Request body - {“uniqueContestID”: “id”, “contestName”: “contest name”, “startDate”: date, “endDate”: date, “description”: “contest description”, “entries”: []}
○ Response: status – 201, message: “Contest successfully created”

/addComment

○ Purpose involves handling the addition of comments to the empty list of a post (with the ID for that post) with the contents of the text entered by the user.
○ POST
○ Request: body – {“uniquePostID”: “id”, “user”: “username”, “comment”: “comment text”}
○ Response: status – 201, message: “Comment successfully added”

· **/enterContest**

○ The users will be able to enter a contest where the API will simply add the post ID to the contest entries list.
○ POST
○ Request: body – {uniqueContestID, entries, uniquePostID}
○ Response: status – 200, message: “Contest successfully entered”

/removeEntry

○ Removal of an entry from a contest will involve updating the entries list of that contest to not include that post ID anymore.
○ POST
○ Request: body – {uniqueContestID, entries, uniquePostID}

/getContestInfo

○ Retrieves contest information such as name, description, start date, end date, and posts associated with it.
○ GET
○ Response: status – 200, data

/getPostInfo

○ Retrieves post information such as username, image, caption, and comments.
○ GET
○ Response: status – 200, data

/addUser

○ Adds a user to the database.
○ POST
○ Request: body – {“name”: “name”, “user”: “username”, “email”: “email”}
○ Response: status – 200, message: “success”

/addAdmin

○ Adds adminto the database.
○ POST
○ Request: body – {“name”: “name”, “user”: “username”, “email”: “email”}
○ Response: status – 200, message: “success”

Design References:

We were primarily inspired by Instagram. Its modern design and colorful palette will make for a very unique space. Moreover, the idea of a social media platform for fashion design is in-line with the original idea.

In addition, we were also inspired by voting and competition based games such as Home Design. This game utilizes an internal voting system to determine winners for contests for in-game rewards.

Fig 1.2 : Instagram mobile application

Fig 1.3 : Home Design: Lifestyle Game competition results

2. Phase-1 and Phase-2 Deliverables:

Plan

Phase 1:
- Wesley: Account creation and user basic functionality i.e. login and post creation
- Katarina: Voting system and rewards

Phase 2:
- Wesley: Authorization for users to edit posts/contests allow interactions via comments
- Katarina: Incorporate post entry into contests and display

3. GitHub Repository: