accountManagementServices - [includes] - userServices, and privilegeServices

Req-001: createUser API - userServices

As part of the **userServices**, we need to create an API that will create a new user and store it in db.

➔ Method type: **POST**.

➔ **API:**
- name: **createUser**
- **api/v1/users/createUser**

➔ DB Bucket to be used: **userInfo**.

➔ Path parameter should be empty.
➔ Parameters to be sent as part of the request body: firstName, lastName, phoneNo, emailId, bio, userName, password, and profileImg.
➔ Other parameters: role, blogsCount, createdAt, lastLogin, and isVerified should store default values.
➔ “password” needs to be encoded using bcrypt crypto with saltRounds as 10 before storing it into the db.
➔ A random “verificationCode” needs to be generated and sent to the provided emailId via mail for verification of the user.
➔ Generated “verificationCode” needs to be stored as a value of the parameter “verificationCode” in the user object.
➔ The “verificationLink” which is sent to the user should contain three parameters: userId, time of creation, and verificationCode.
➔ The “verificationLink” should be of below form:
api/v1/users/verify/{userId}/{time}/{verificationCode}

➔ **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.

➔ **Email Template:**
- The Email template should contain the following text in the same order as provided with the font size and color as specified.
  “We got a request for a new account creation with this emailId. Please verify your emailId to activate the account.” (font-size: 20px, type: text)
  “Please click on the following link to proceed.” (font-size: 18px, type: text)
  “Verify” (font-size: 16px, color: blue, type: link)
  “Stamp” (type: text) should be present at the end of the email.

Req-002: verifyAccount API - userServices

As part of the **userServices**, we need to create an API that will activate the user's account by verifying the verificationCode sent in the request.

➔ Method type: **GET**.

➔ **API:**
- name: **verify**
- **api/v1/users/verify**

➔ DB Bucket to be used: **userInfo**.

➔ Parameters to be sent as part of the path: userId, time, and verificationCode.
➔ Request body should be empty.
➔ Verify if the time at which the verification request is received and the "time" parameter, which is received as part of the path, is within six hours or not. If "yes" then update the value of "isValidated" parameter as true for the requested user. And if "no" then send the appropriate message to the user.

➔ **Response messages:**
- Response message should be "201 Created" in case the operation is successful.
- Response message should be "400 Bad Request" in case the verification request time is beyond six hours or verificationCode did not match.
- Response message should be "404 Not Found" in case the data for requested userId does not exist.
- Response message should be "500 Internal Server Error" in case of any error occurred in storing the data in db.

➔ **HTML Templates:**
- The following html template to be displayed on the web in case the verification is successful. The text needs to be in the same order as provided with the font size and color as specified.
  "Congrats! Your account has been verified successfully." (font-size: 28px, type: text, color: wheat)
  "Thanks for joining us, you're officially an Author and part of our Team." (font-size: 22px, type: text, color: wheat)
  "Stamp" (type: text) should be present at the end.

- The following html template to be displayed on the web in case the verification failed. The text needs to be in the same order as provided with the font size and color as specified.
  "Verification Code Expired!" (font-size: 28px, type: text, color: wheat)
  "Please relogin and get a new verification code to activate your account." (font-size: 22px, type: text, color: wheat)

“Note: Your account may have already been verified. Please try to login to the portal. If you're not authorized, you'll get a new verification code to activate your account.” (font-size: 22px, type: text, color: wheat)
“Stamp” (type: text) should be present at the end.

➔ **Email Template:**

- The Email template to be sent if the verification is successful, and the template should contain the following text in the same order as provided, with the font size and color as specified.
  “Welcome to our Family” (type: heading)
  “Congrats! Your account has been created successfully. You're now an Author and part of our Family.” (font-size: 20px, type: text)
  “Following are your registered details:” (font-size: 18px, type: text)
  “First Name:” (type: text)
  “Last Name:” (type: text)
  “Username:” (type: text)
  “Phone Number:” (type: text)
  “Email Id:” (type: text)
  “Role:” (type: text)
  “Bio:” (type: text)
  “If any of your details are wrong, please visit to our portal and update your details.” (font-size: 18px, type: text)
  “Note: You cannot update your Role.” (font-size: 18px, type: text)
  “Stamp” (type: text) should be present at the end.

Req-003: getAllUsersInfo API - userServices

As part of the **userServices**, we need to create an API that will return the list of all the users stored in db.

➔ Method type: **GET**.

➔ **API:**
- name: **getAllUsersInfo**
- **api/v1/users/getAllUsersInfo**

➔ DB Bucket to be used: **userInfo**.

➔ Path parameter should be empty.
➔ Request body should be empty.
➔ Fields to be returned as part of the response: firstName, lastName, phoneNo, emailId, bio, userName, role, profileImg, blogsCount, blogsList, createdAt, lastLogin, and isVerified.

➔ **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-004: getUserDetails API - userServices

As part of the **userServices**, we need to create an API that will return the details of a particular user using its unique id present in db.

➔ Method type: **GET**.

➔ **API:**
- name: **getUserDetails**
- **api/v1/users/getUserDetails/{userId}**

➔ DB Bucket to be used: **userInfo**.

➔ Parameter to be sent as part of the path: “userId”.
➔ Request body should be empty.
➔ Fields to be returned as part of the response: firstName, lastName, phoneNo, emailId, bio, userName, role, profileImg, blogsCount, blogsList, createdAt, lastLogin, and isVerified.

➔ **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 userId does not exist.
- Response message should be “500 Internal Server Error” in case of any error occurred in connection with db.