Req-001: createUser API - Account Management Services

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

- ➔ Method type: **POST.**

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

- ➔ DB Bucket to be used: **userInfo**.

- ➔ Path parameters should be empty.
- ➔ Mandatory parameters to be sent as part of the request body: firstName, lastName, userName, emailId, and password.
- ➔ Payload must be verified if all the mandatory parameters are available or not before proceeding with the create user functionality.
- ➔ "password" needs to be encoded using bcrypt crypto with saltRounds 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.
- ➔ On successful creation of a new customer the verification email should be sent to the provided emailId for the verification of the customer.

- ➔ **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 "400 Bad Request" in case of any error occurring while saving the user in db.
  - Response message should be "500 Internal Server Error" in case any other error occurs in the entire operation.

- ➔ **Email Functionality:**
  - The "verificationLink" which is sent to the user as part of the verification email should contain three parameters: userId, time of creation, and verificationCode.
  - The "verificationLink" should be of below form:
    api/users/verify/{userId}/{time}/{verificationCode}
  - The Email template should contain the following text in the same order as provided.
    "An account request has been received for this email address. To activate your account on financeTracker, please verify your email."
    "To continue, kindly click the link below."

“Verify”
“Regards, Team financeTracker”

Req-002: verifyUser API - Account Management Services

As part of the **accountManagementServices**, 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/users/verify/{verificationLink}**

➔ DB Bucket to be used: **userInfo**.

➔ Parameters to be sent as part of the path: userId, time of creation, and verificationCode.
➔ Request body should be empty.
➔ Payload must be verified if user id is present or not before proceeding with verifying user functionality.
➔ 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 the above validation is successful then match the verification code provided as part of the payload with the verification code that is stored in db against the user for which the id is provided as part of the path of an api.
➔ If the above validation is also successful then update the value of "isValidated" parameter as true for the requested user, and send the successful validation mail to the user. And if the above validation fails then send the new verification mail to the user.

➔ **Response messages:**
- Response message should be "200 Ok" in case user verification 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.

➔ **Verification successful/failed message html page:**
- 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.
  "Congrats! You're Officially a Member of financeTracker."
  "Thanks for joining us."
  "Regards, Team financeTracker"

- 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.
  "Verification Code Expired!"
  "Please relogin and get a new verification code to activate your account."

“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.”
“Regards, Team financeTracker”

➔ **Email Functionality:**

- The Email to be sent if the verification is successful, and the template should contain the following text in the same order as provided:
  “Welcome to Your Finance Companion”
  “Congratulations! You're Officially a Member of financeTracker.”
  “Following are your registered details:”
  “First Name:”
  “Last Name:”
  “Username:”
  “Phone Number:”
  “Email Id:”
  “If you find any discrepancies in your details, please visit our portal to make updates.”
  “Regards, Team financeTracker”

Req-003: userLogin API - Account Management Services

As part of the **accountManagementServices**, we need to create an API that will validate the credentials entered by the user and return the token if credentials are correct.

- ➔ Method type: **POST**.

- ➔ **API:**
  - name: **userLogin**
  - **api/users/userLogin**

- ➔ DB Bucket to be used: **userInfo**.

- ➔ Path parameters should be empty.
- ➔ Mandatory parameters to be sent as part of the request body: userName, and password.
- ➔ Payload must be verified if all the mandatory parameters are available or not before proceeding with the login user functionality.
- ➔ Find the correct user record in db based on entered userName, then match the user entered password with the stored password to check if the credentials are correct or not. If the credentials are correct then check if the user is verified or not.
  - If the user is verified then generate the token and return the token as a response along with userId and userName.
  - If the user is not verified then generate the verification code and send it to the provided emailId via mail for verification of the user.
  - Also if the user is a deactivated user then reactivate the user and send a verification code to reverify the user.
- ➔ Use the secret key to generate the token with an expiration time of 1 hour.
- ➔ Update the login count and last login time of the user.

- ➔ **Response messages:**
  - Response message should be “200 Ok” in case the verification is successful and return the token as a response.
  - Response message should be “201 Created” in case the verification is required and email is sent.
  - Response message should be “400 Bad Request” in case the required parameters are missing.
  - Response message should be “401 Unauthorized” in case the users entered credentials are incorrect.
  - Response message should be “404 Not Found” in case the required user is not found.
  - Response message should be “500 Internal Server Error” in case any other error occurred.

Req-004: validateToken API - Account Management Services

As part of the **accountManagementServices**, we need to create an API that will validate the provided token and user id, and return the appropriate response based on verification results if it is successful or not.

The validateToken is an internal API used by other APIs to verify the validity of the token before delivering the API response to the user.

➔ Method type: **POST**.

➔ **API:**
- name: **validateToken**
- **api/users/validateToken**

➔ DB Bucket to be used: **userInfo**.

➔ Path parameters should be empty.

➔ Mandatory parameters to be sent as part of the request body: userId and token.

➔ Payload must be verified if all the mandatory parameters are available or not before proceeding with the verify token functionality.

➔ Verify the token if the token is correct and not expired.

➔ Once the token has been verified then check if the user with the provided user id exists or not.

➔ **Response messages:**
- Response message should be “200 Ok” in case the token verification is successful.
- Response message should be “400 Bad Request” in case the required parameters are missing.
- Response message should be “401 Unauthorized” in case the user id or token is not valid.
- Response message should be “500 Internal Server Error” in case any other error occurred.

Req-005: getUserInfo API - Account Management Services

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

➔ Method type: **GET**.

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

➔ DB Bucket to be used: **userInfo**.

➔ Parameter to be sent as part of the path: userId.
➔ Request body should be empty.
➔ Token must be sent as part of authorization bearer token.
➔ Call a validateToken API internally with the userId and token.
➔ If the validateToken API returns 200 response, then proceed to get the records from db for the required user.
➔ Fields to be returned as part of the response: userId, firstName, lastName, userName, bio, gender, dob, occupation, emailId, contactNumber, createdOn, lastLogin, and loginCount.

➔ **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 other error occurred.
- Responses from validateToken API will be returned.

Req-006: updateUserDetails API - Account Management Services

As part of the **accountManagementServices**, we need to create an API that will update the information of users using their unique id stored in db.

- ➔ Method type: **PUT**.

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

- ➔ DB Bucket to be used: **userInfo**.
- ➔ Parameter to be sent as part of the path: userId.
- ➔ Token must be sent as part of authorization bearer token.
- ➔ Parameters that can be sent as part of the request body: firstName, lastName, contactNumber, emailId, bio, userName, gender, and occupation.
- ➔ Call a validateToken API internally with the userId and token.
- ➔ If the validateToken API returns 200 response, then proceed with updating the details of the user in db.
- ➔ If the operation is successful then send the mail to the user with the updated details of 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 required parameters are missing.
  - 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 any other error occurred.
  - Responses from validateToken API will be returned.

- ➔ **Email Functionality:**
  - The Email to be sent if the user update details operation is successful, and the template should contain the following text in the same order as provided:
    “Your Information Successfully Updated!”
    “Congrats! Account Info Updated.”
    “Following are your updated details.”
    “First Name:”
    “Last Name:”
    “Username:”
    “Email Id:”
    “Contact Number:”
    “Gender:”
    “DOB:”
    “Bio:”

“Occupation:”
“Account creation date:”
“Last login time:”
“If any of your details are wrong, please visit our website and update your details.”
“Regards, Team financeTracker”

Req-007: updateUserPassword API - Account Management Services

As part of the **accountManagementServices**, we need to create an API that will update the password of users using their unique id stored in db.

- ➔ Method type: **PUT**.

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

- ➔ DB Bucket to be used: **userInfo**.

- ➔ Parameter to be sent as part of the path: userId.
- ➔ Token must be sent as part of authorization bearer token.
- ➔ Parameters that can be sent as part of the request body: oldPassword, and newPassword.
- ➔ Call a validateToken API internally with the userId, and token.
- ➔ If the validateToken API returns 200 response, then proceed with updating the password of the user in db.
- ➔ Validate if the user's oldPassword is the same as the stored password of the required user or not. If the password is the same then check if the newPassword and oldPassword are not the same. If both the conditions satisfy then update the password.
- ➔ Before updating the password, encode it using bcrypt crypto with saltRounds.
- ➔ Send the user a mail informing them that their password has been successfully updated if the process is successful.

- ➔ **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 parameters are missing.
  - Response message should be "401 Unauthorized" in case the oldPassword mismatched with the stored password for the requested user.
  - 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 any other error occurred.
  - Responses from validateToken API will be returned.

- ➔ **Email Functionality:**
  - The Email to be sent if the user password update operation is successful, and the template should contain the following text in the same order as provided:
    "Password updated successfully!"
    "Congrats! Your password has been updated successfully."
    "Regards, Team financeTracker"

Req-008: deactivateUser API - Account Management Services

As part of the **accountManagementServices**, we need to create an API that will update the parameter “isDeleted” of the users to deactivate their account, using their unique user id stored in db.

➔ Method type: **PUT**.

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

➔ DB Bucket to be used: **userInfo**.

➔ Parameter to be sent as part of the path: userId.
➔ Token to be sent as part of authorization bearer token.
➔ Parameter to be sent as part of the request body: userName, and password.
➔ Call a validateToken API internally with the userId and token.
➔ If the validateToken API returns 200 response, then proceed with deactivating the user.
➔ Verify the payload to check if the userName, and password is present or not.
➔ Verify if the entered userName, and password are correct or not for the user which is stored in db with the entered userId.
➔ If both the verifications succeed then update the parameter “isDeleted” of the requested user to deactivate it’s account.
➔ Send the user a mail informing them that their account has been deactivated, and after 1 month their account will automatically be deleted, and to reactivate they’ve to login back within 1 month.

➔ **Response messages:**
- Response message should be “200 Ok” in case the operation is successful.
- Response message should be “400 Bad Request” in case the required parameters are missing.
- Response message should be “401 Unauthorized” in case the userName or password mismatched with the stored details of the requested user.
- 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 any other error occurred.
- Responses from validateToken API will be returned.

➔ **Email Functionality:**
- The email to be sent if the user deactivation operation is successful, and the template should contain the following text in the same order as provided:
  “Account Deactivated!!!”
  “We would like to inform you that your account has been deactivated successfully.”

“To re-activate your account, kindly login to the portal, you'll get the mail with the verification code to reactivate your account.”
“Regards, Team financeTracker”

Req-009: requestPasswordReset API - Account Management Services

As part of the **accountManagementServices**, we need to create an API that will check for the userName or emailId, and if the record found in db with the provided userName or emailId then send the mail to the user to reset their password.

- ➔ Method type: **GET**.

- ➔ **API:**
  - name: **requestPasswordReset**
  - **api/users/requestPasswordReset**

- ➔ DB Bucket to be used: **userInfo**.

- ➔ Path parameter should be empty.
- ➔ Mandatory parameter to be sent as part of the request body: userName or emailId.
- ➔ Payload must be verified if the mandatory parameter is present or not.
- ➔ Validate if there’s any user present in db with the entered emailId or userName or not. If the user is found then a random “verificationCode” needs to be generated and sent to the provided emailId via mail for resetting the password of the user.
- ➔ Generated “verificationCode” needs to be stored as a value of the parameter “verificationCode” in the user object.

- ➔ **Response messages:**
  - Response message should be “200 Ok” in case the operation is successful.
  - Response message should be “400 Bad Request” in case the required parameter is missing.
  - Response message should be “404 Not Found” in case the requested user does not exist.
  - Response message should be “500 Internal Server Error” in case any other error occurred.

- ➔ **Email Functionality:**
  - The email with the reset password link to be sent to the designated user in case the requested user is found, and the template should contain the following text in the same order as provided.
  - The “Password reset link” should be of below form:
    api/users/requestPasswordReset/{userId}/{verificationCode}
  - The Email template should contain the following text in the same order as provided:
    “Reset your password”
    “For your account, a request to reset your password has been received. If you need to reset your password, visit the link below.”
    passwordResetLink
    “Regards, Team financeTracker”

Req-010: resetPassword API - Account Management Services

As part of the **accountManagementServices**, we need to create an API that will update the password of users using their unique id stored in db.

- ➔ Method type: **PUT**.

- ➔ **API:**
  - name: **resetPassword**
  - **api/users/resetPassword/{userId}/{verificationCode}**

- ➔ DB Bucket to be used: **userInfo**.

- ➔ Parameter to be sent as part of the path: userId, and verificationCode.
- ➔ Mandatory parameter to be sent as part of the request body: password.
- ➔ Payload must be verified if the mandatory parameter is present or not.
- ➔ Validate if the user with the entered userId exists in DB or not. If the user exists then update the password with the new password.
- ➔ Before updating the password, encode it using bcrypt crypto with saltRounds.
- ➔ Send the user a mail informing them that their password has been successfully updated if the process is successful.

- ➔ **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 parameters are missing.
  - 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 any other error occurred.

- ➔ **Email Functionality:**
  - The Email to be sent if the user password update operation is successful, and the template should contain the following text in the same order as provided:
    “Password updated successfully!”
    “Congrats! Your password has been updated successfully.”
    “Regards, Team financeTracker”

Req-011: userVerificationCheck Scheduled Job - Account Management Services

As part of the **accountManagementServices**, we need to create a job scheduler that runs everyday at 12 in the afternoon and sends the mail to the registered users, who have not yet verified their account and their account creation time beyond 6 hours.

- ➔ DB Bucket to be used: **userInfo**, and **userLogs**.

- ➔ Run a query in the database to check for all the users whose email is not yet verified and their account is not yet activated.
- ➔ Then further check for all those users for whom their account creation time is beyond 6 hours.
- ➔ Then send out a mail to all these users that they've to verify their account within 10 days, or after 10 days their account will be self deactivated.
- ➔ Additionally, remove the users' existing verificationCode because doing so will help to ensure that the mail is sent just once.
- ➔ Generate the logs and store the records of such users in DB, also in case of an error generate the appropriate log and store in DB.

- ➔ **Email Functionality:**
  - The Email to be sent to the required users, and the template should contain the following text in the same order as provided:
    "Account Activation Required!"
    "We can see that you have not yet activated your account by verifying your email. Please use the new verification link to verify your account after logging in to our website."
    "Notification: Our system will terminate your account if you don't authenticate and activate it within the next 10 days."
    "Regards, Team financeTracker"

Req-012: Auto Deactivate User Scheduled Job - Account Management Services

As part of the **accountManagementServices**, we need to create a job scheduler that runs at every midnight and mark those registered users as soft deleted, who have not yet verified their account and their account creation time is beyond 10 days.

➔ DB Bucket to be used: **userInfo**, and **userLogs**.

➔ Run a query in the database to check for all the users whose email is not yet verified and their account is not yet activated.

➔ Then further check for all those users for whom their account creation time is beyond 10 days.

➔ Then send out a mail to all these users saying that, to reactivate the account, the user has to login to the website and get the new verification code and verify their account to activate within the next 30 days, or after 30 days, their account will be permanently deleted.

➔ Mark the user as soft deleted, and last login time as current time. This will help in further proceedings.

➔ Generate the logs and store the records of such users in DB, also in case of an error generate the appropriate log and store in DB.

➔ **Email Functionality:**

- The Email to be sent to the required users, and the template should contain the following text in the same order as provided:
  “Account Deactivated!”
  “We can see that you have not activated your account by verifying your email in the past 10 days, which caused your account to be self-deactivated. To reactivate your account, use the new verification link to verify your account after logging in to our website.”
  “Notification: Our system will delete your account if you don’t reactivate it in the next 30 days.”
  “Regards, Team financeTracker”

Req-013: Auto Delete User Scheduled Job - Account Management Services

As part of the **accountManagementServices**, we need to create a job scheduler that runs at midnight and delete those users who were marked as soft deleted and whose last login time is before 30 days.

- ➔ DB Bucket to be used: **userInfo**, and **userLogs**.

- ➔ Run a query in the database to check for all the users who are marked as soft deleted and their last login time is before 30 days.
- ➔ Then send out a mail to all these users saying that their account has been deleted permanently.
- ➔ Delete all such users from the database that were marked as soft deleted and whose last login time is beyond 30 days.
- ➔ Generate the logs and store the records of such users in DB, also in case of an error generate the appropriate log and store in DB.

- ➔ **Email Functionality:**
  - The Email to be sent to the required users, and the template should contain the following text in the same order as provided:
    “Account Deleted!”
    “With a heavy heart, we would like to inform you that your account has been deleted. We’ll miss you and hope to have you back soon, but unfortunately, you've got to start from scratch.”
    “Regards, Team financeTracker”

Req-014: Auto Delete Logs Scheduled Job - Account Management Services

As part of the **accountManagementServices**, we need to create a job scheduler that runs at every midnight and delete the logs which are generated before 30 days.

- ➔ DB Bucket to be used: **userLogs**.

- ➔ Run a query in the database to check for all those logs which are generated before 30 days and delete all those logs.
- ➔ Before deleting the logs, generate an excel file with all the logs getting deleted and send it via mail to the official mail.
- ➔ In case of an error generate the appropriate log and store it in DB.

- ➔ **Email Functionality:**
  - The Email to be sent to our official mail, and the template should contain the following text in the same order as provided:
    “Deleted Logs Details”
    “Dear Admin, please find the details of the deleted logs in the attached file.”
    “Number of records:”
    “Log date interval:”
    “Regards, Team financeTracker”