Android Challenge

Context

A common pattern for a modern, consumer facing native mobile apps has a user account with associated credentials that allow the app to securely interact with a RESTful back-end.

Assume that the user is required to log into the app (and back-end service) and that doing so provides the app with an API token. Assume that to communicate with the back-end's REST services, it requires use of the API token provided as an HTTP header **Bearer** variable.

The app should have a simple login screen that accepts a username and password. Once successfully logged in, the user is presented with a single screen that displays their account information, which consists of their email address (read only), password (read only) and a profile photo (editable). By default, the profile photo should be a blank silhouette avatar. Tapping the photo should allow the user to select an image from their photo library, or to take one with the front-facing camera. Saving the changes should update the avatar locally and with the back-end.

The back-end has the following rest endpoints:

- POST /sessions/new { "email": ":email", "password": ":password"} -> { "userid": ":userid", "token": ":token" }
- GET /users/:userid -> { "email": ":email", "avatar_url": ":avatar_url" }
- POST /users/:userid/avatar { "avatar": ":base64_encoded_data" } -> { "avatar_url": ":avatar_url" }

Assume that the back-end makes of use of standard HTTP success and error codes.

Android Challenge

Challenge

Required

1. Build a simple Android app, following 'Material Design Guidelines', that uses standard native interface controls to provide the login and account profile views.
2. Choose your favourite architecture and focus on separation of concerns and testability.
3. Ensure that on subsequent invocations of the app, that the user is automatically logged in (assuming at least one successful previous login).
4. Provide tests for the critical parts of the app.
5. Add the ability for the user to change their avatar using either by taking a photo with the inbuilt camera, or by selecting a photo from their photo library.

Stretch

1. Implement the app using clean architecture with MVVM.
2. Add the ability for the app to display the user's Gravatar [see: www.gravatar.com] if a) a Gravatar for their email exists, and b) they have not specifically set up their own photo.
3. Display the user's avatar in a circle. Ensure that the photo is correctly positioned within the display area.
4. Assure that the Image uploaded to the backend does not exceed 1 MB.