QA Task List: Detox + Jest Testing for React Native Project

---

1. Setup & Environment Preparation

- Verify installation and configuration of required tools:
    - Node.js (LTS)
    - Java JDK 11+
    - Android Studio + Android SDK
    - Git
    - Expo CLI
- Clone the project repository and checkout the correct branch.
- Run `npm ci` (or `npm ci --legacy-peer-deps` if needed) to install all dependencies.
- Confirm `xlsx` is installed as a dev dependency (`npm install --save-dev xlsx`).
- Run `npx expo prebuild --platform android` to generate native Android folders.
- Build the Android APK for Detox testing via Gradle.

---

2. Configuration Validation

- Verify `detox.config.js` is present with Android emulator and Jest runner configured.
- Confirm `package.json` includes `detox:build` and `detox:test` scripts.

- **Auth0 bypass:**
    - Verify that in test mode, Auth0 login is bypassed correctly.
    - Confirm fixed test assets (bio image, profile video, invite video) are loaded during bypass.
    - Check that profile image and videos update correctly when bypassed.
    - Coordinate with in-house developer to obtain exact file names and paths of these media assets to use as test samples.
    - Ensure bypassed data is saved to the database or mocked storage for consistency during tests.

---

3. Mock Data & Test Users

- Confirm `e2e/data/users.json` contains 56 test users with full data (IDs, zip codes, tokens, media paths).
- Validate existence and correct usage of `e2e/data/mock-backend.json` if backend mocking via `json-server` is used.

---

4. Test Suite Structure & Utilities

- Verify `e2e/` folder includes:
    - Jest config files
    - `init.js`
    - Reporting utilities
    - `combinatorial.e2e.js` for match testing

- Review utility functions for login, sliders, match triggering, and verification steps.

---

5. Communication & Coordination

- **Download and install Discord**, the team's main communication platform.
- Join the project's Discord server/channel for:
    - Real-time communication with developers and project managers
    - Getting clarifications, reporting bugs, and feedback
    - Receiving updated media files, configuration details, or test data directly from the in-house developer or team
- Note: All payments for QA services will be processed via Fiverr for now.

---

6. Test Execution & Coverage

- Run combinatorial tests covering all/randomized user pairs with 3 sliders and 3 options each, using up to 9 tokens.
- Ensure full token usage per test session for User A and User B.
- Capture and document per test case:
    - Match result (Yes/No) based on detailed matching criteria.
    - Match type (Friends, Hook-Up, Company, A Date).
    - First ping initiator (User A or User B).
    - Videos and stories viewed by each user, with identification of the **first story/video seen**.

    - Profile image and video updates (verify media changes correctly reflect in the app).
    - Usernames and zip codes of both users for geographic distance validation.
    - Tokens spent per interaction.
- Validate matches against the 15 predefined test cases (see table below).
- Record all results accurately for reporting.

Test Case	User A (Sex, Romance, Friends, LongTerm)	User B (Sex, Romance, Friends, LongTerm)	Match?	First Ping
1	(0, 0, 3, FALSE)	(0, 0, 2, FALSE)	Yes – Friends	User B
2	(3, 0, 0, FALSE)	(2, 0, 0, FALSE)	Yes – Hook-Up	User B
3	(0, 2, 1, FALSE)	(0, 3, 1, FALSE)	Yes – Company	User A
4	(2, 2, 1, FALSE)	(1, 3, 3, FALSE)	Yes – A Date	User B
5	(3, 1, 1, FALSE)	(1, 3, 2, FALSE)	No	N/A
6	(1, 3, 3, TRUE)	(2, 2, 2, TRUE)	Yes – A Date	User B
7	(3, 2, 0, FALSE)	(0, 2, 2, FALSE)	No	N/A
8	(1, 1, 1, FALSE)	(1, 1, 1, FALSE)	Yes – A Date	N/A (Equal Interest)
9	(0, 3, 2, FALSE)	(0, 3, 3, FALSE)	Yes – Company	User A
10	(2, 0, 1, FALSE)	(0, 2, 1, FALSE)	No	N/A
11	(1, 2, 3, TRUE)	(2, 2, 2, TRUE)	Yes – A Date	User B
12	(2, 3, 0, FALSE)	(0, 0, 4, FALSE)	No	N/A

13	(0, 0, 0, FALSE)	(0, 0, 0, FALSE)	No	N/A
14	(0, 1, 1, FALSE)	(0, 1, 1, FALSE)	Yes – Company	N/A (Equal)
15	(3, 0, 1, FALSE)	(3, 0, 1, FALSE)	Yes – Hook-Up	N/A (Equal)

---

7. Reporting & Deliverables

- Confirm Jest produces reports in both `junit.xml` and HTML formats.
- Validate detailed match results are saved in:
    - `./reports/match-results.json`
    - `./reports/match-results.xlsx` (generated with `xlsx` package)
- Reports must include:
    - Summary of matches, non-matches, and tokens used
    - Match type breakdown
    - Videos and stories viewed logs, with first story identification
    - Profile image and video update verification results
    - Geographic distance data based on user zip codes
- Run tests on emulator (and mock backend server if used) using commands:
    - `npm run detox:build`
    - `npm run detox:test`
- Review all reports thoroughly for accuracy and completeness.
- Deliver reports and detailed documentation covering setup, test coverage, and any issues discovered to the client.