NeuroLock (User Guide)

Contributors: Riya Jain, Sanskriti Gupta, Satatya De, Shubhi Jain, Yug Brahmbhatt

TEAM VirUsS

This guide will help you understand what the project is, how to run it locally, explore its detailed features, identify assumptions made during development, and outline the currently observed vulnerabilities (with remediation plans) that we have identified.

1) What is this project about?

NeuroLock is a hospital- and staff-focused secure storage system that provides authentication and streamlined workflows for patient management and assessments in a psychiatric setting.

To view the entire working of the app, watch the demo video:
https://drive.google.com/file/d/1BCetUVzjlIAK9cayMIvx_gUpNRolrvqc/view?usp=sharing

The repository contains:

- A mobile-first frontend built with Expo / React Native (TypeScript).
- A companion backend service (TypeScript / Node / Express) that implements staff authentication, database migrations, and APIs for staff actions.
- SQL migration scripts and helper utilities for database management.

Primary goals:

- Provide secure login, multi-factor authentication (MFA) and device registration for staff.
- Offer role-specific dashboards (nurse, psychiatrist, psychologist, admin) and patient record access.
- Make offline-capable assessment screens and secure data-sync paths.
- This repository is intended for hospital/internal deployments and development environments. Production deployment requires secure infrastructure, secrets management, and compliance checks (see the vulnerabilities section below).

2) How to run it WITH APK

- Download the Apk
- You can login with the following mails:
    - psychiatrist@neurolock.com

    - psych@neurolock.com
    - nurse@neurolock.com
    - admin@neurolock.com

Password is **demo@123**
You can also use biometrics and run different dashboards for a demo.
Same passwords to run locally.

Furthermore, we acknowledge that the app is not perfect, so please refer to points 5 and 6 to understand the potential errors you may encounter.

3) How to run it LOCALLY

This section explains how to run both the frontend and backend locally on Windows (using PowerShell) and notes alternative options.

Prerequisites

- Node.js >= 14 (backend engines asks for >=14). We recommend Node 16+ or 18+.
- npm (or yarn) and Git
- Expo CLI (for the frontend mobile app); install globally: npm install -g expo-cli (or use npx).
- A local PostgreSQL or MySQL instance for the backend (the repo contains SQL/migration helpers). Ensure DB user + database are reachable.
- Optional: Docker / Docker Compose (a docker-compose.yml exists in neurolock-staff-backend/ to help containerize services).

Frontend (Expo React Native)

- Open PowerShell and change to the frontend folder:
- cd "frontend" folder
- Install dependencies:
- npm install
- Start the development server (Expo):
- npm start
- Launch on platform of choice from Expo DevTools or directly:
    - Android emulator: npm run android (this runs expo start --android)
    - iOS simulator (macOS only): npm run ios
    - Web: npm run web

Notes:

- The frontend uses Expo SDK (package.json shows expo and react-native versions). Use the Expo client or simulator to preview the app.
- For building release binaries (APK/IPA) use eas or Expo build processes — see Expo docs.

Backend (neurolock-staff-backend)

- Open a second PowerShell and change to the backend folder:
- cd "backend" folder
- Install dependencies:
- npm install
- Prepare environment variables:
- Create a .env file (not committed to git) with DB connection strings, JWT secret, and other runtime configs. The app uses dotenv.
- Example keys to set: DATABASE_URL (or separate DB_HOST/DB_USER/etc.), JWT_SECRET, NODE_ENV=development.
- Run migrations (if applicable):
- npm run migrate (This runs the repo's migrate script located at src/infra/db/migrate.js; inspect and ensure credentials are correct.)
- Start in development mode:
- npm run dev
- To run tests and linting:
- npm run test npm run lint

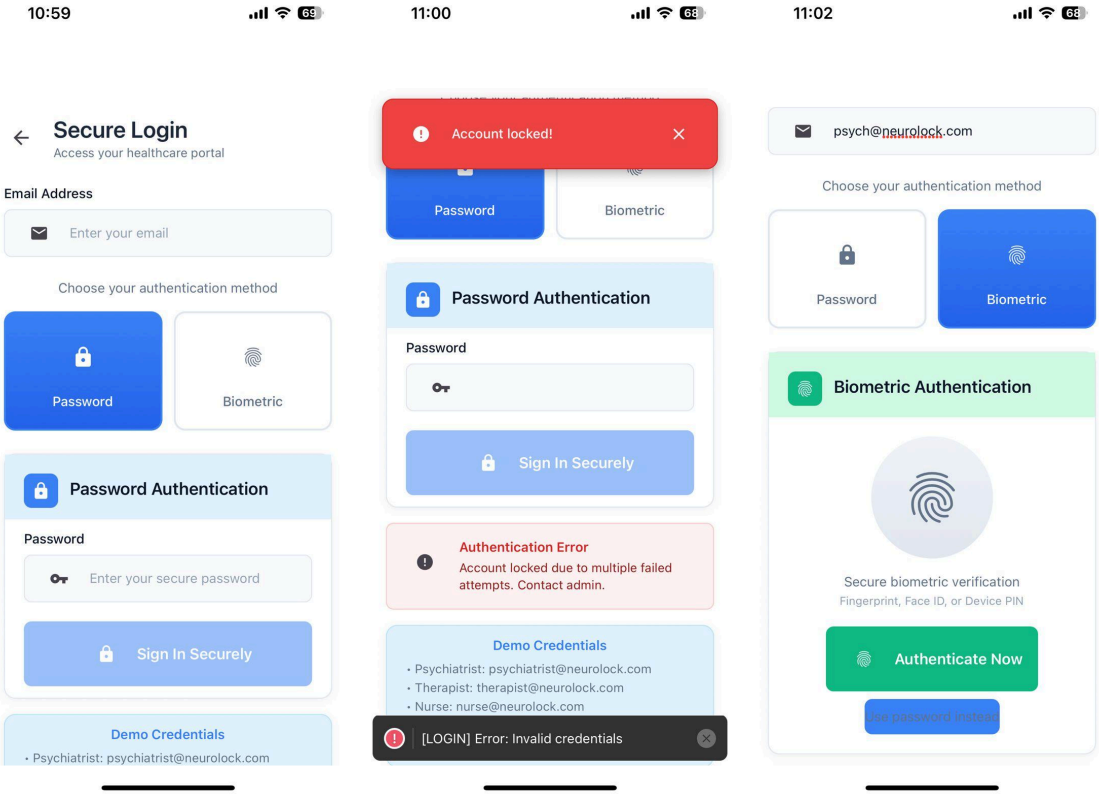
Notes:

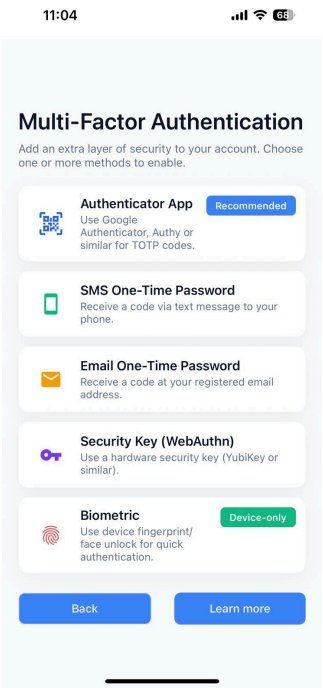
- The backend package.json exposes dev, start, migrate, and rollback scripts. dev runs ts-node-dev for fast TypeScript reload during development.
- The backend depends on pg and mysql2 packages — the expected DB engine will be evident from your environment and migration scripts.
- If you prefer containers, use the included docker-compose.yml inside neurolock-staff-backend/ (review contents before running). Integration and quick ways to run both
- You can run the backend and then start the Expo frontend; configure the frontend's API base URL to point to the backend host (localhost or host IP)
- There are convenience scripts in the repo root such as start.bat, start.sh, and several helper PowerShell scripts (check_status.ps1, test_login.ps1); inspect them to see automation the repo already provides.
- The emails are supposed to be verified at the backend to create accounts, so we will be giving dummy accounts for testing.

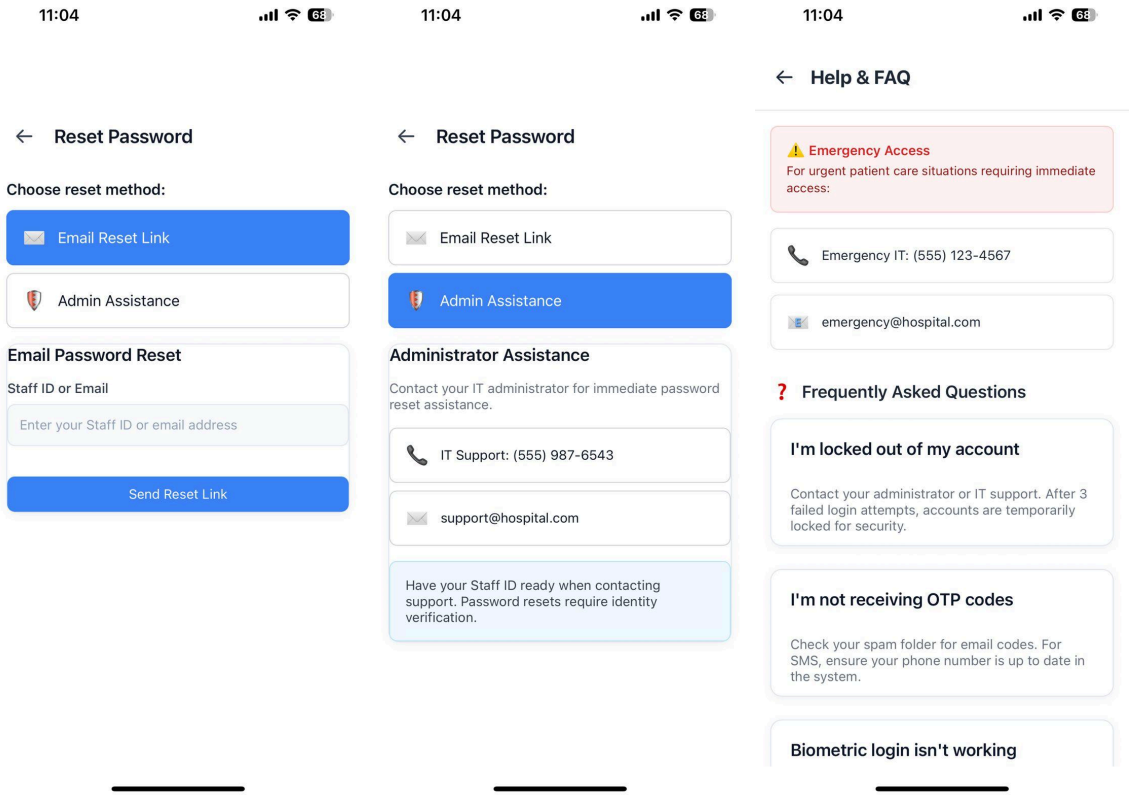
3) What are the various features you can explore?

Authentication & Account Management

- Login flow
  - Standard username/password login.
  - Account locked screen for too many failed attempts.
  - Given 3 tries, the app gets locked for 2 minutes
  - On the first page, you can check out FAqs and MFA methods
  - A way to recover passwords
  - Biometeric signing(Currently coded for demo purpose)

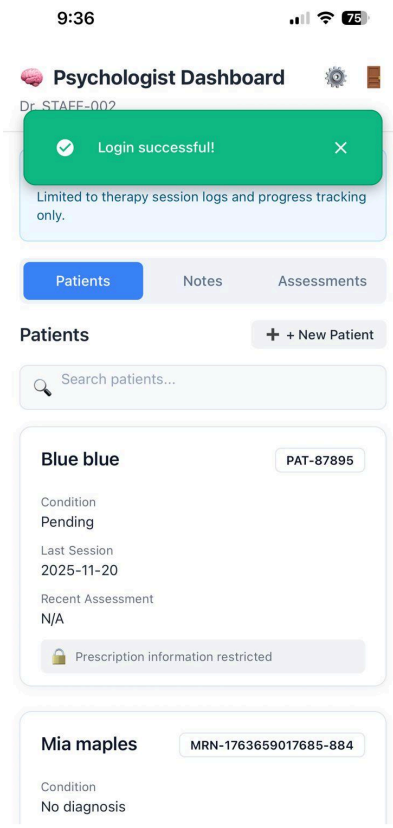


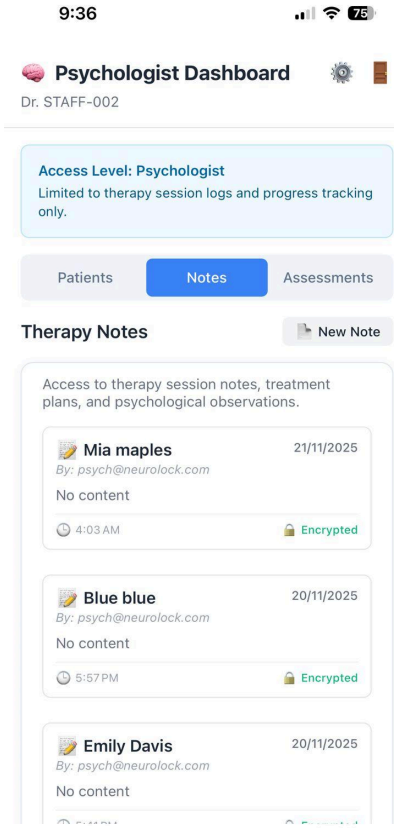


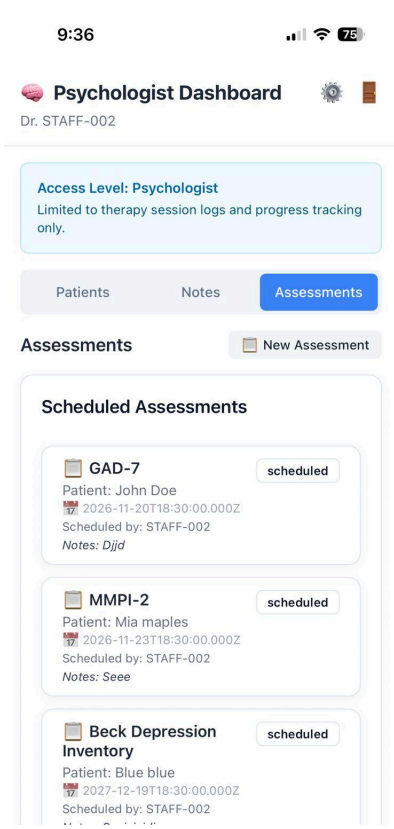


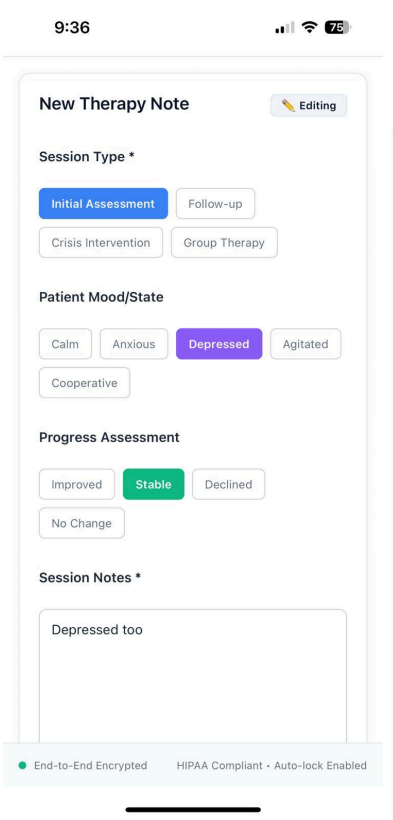
Dashboards & Role-based UIs

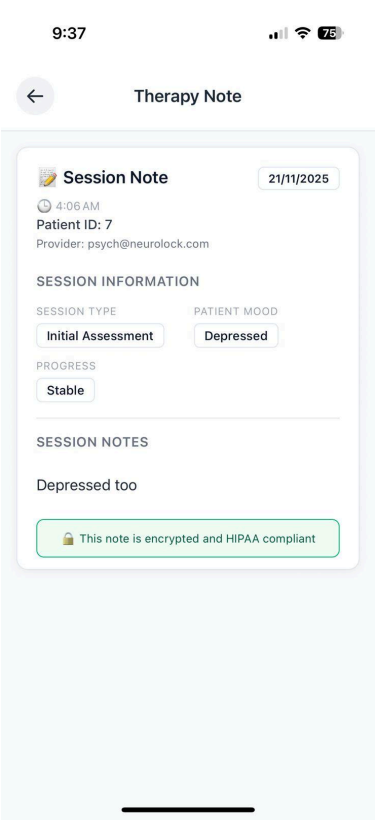
- Psychologist dashboard
  - Patient lists tailored to mental health workflows, assessment creation, and notes.
  - Can add new patients, new therapy notes, and new assessments
  - Can update existing information
  - Cannot view/prescribe medications

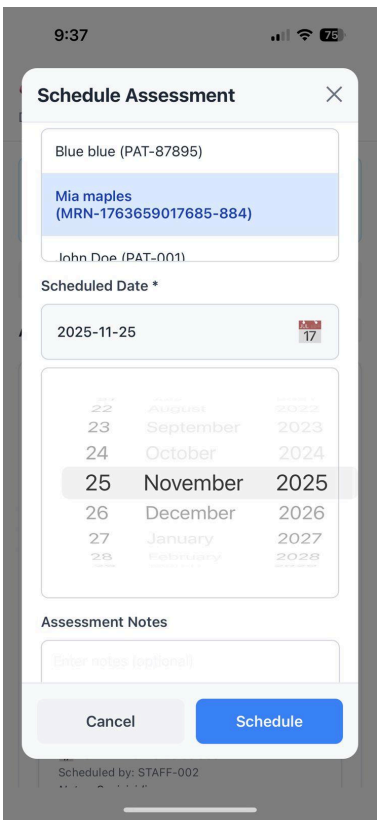

9:36
Psychologist Dashboard
Dr. STAFF-002
Login successful!
Limited to therapy session logs and progress tracking only.
Patients
Notes
Assessments
Patients
+ New Patient
Search patients...
Blue blue
PAT-87895
Condition
Pending
Last Session
2025-11-20
Recent Assessment
N/A
Prescription information restricted
Mia maples
MRN-1763659017685-884
Condition
No diagnosis


9:36
Psychologist Dashboard
Dr. STAFF-002
Access Level: Psychologist
Limited to therapy session logs and progress tracking only.
Patients
Notes
Assessments
Therapy Notes
New Note
Access to therapy session notes, treatment plans, and psychological observations.
Mia maples
21/11/2025
By: psych@neurolock.com
No content
4:03 AM
Encrypted
Blue blue
20/11/2025
By: psych@neurolock.com
No content
5:57 PM
Encrypted
Emily Davis
20/11/2025
By: psych@neurolock.com
No content

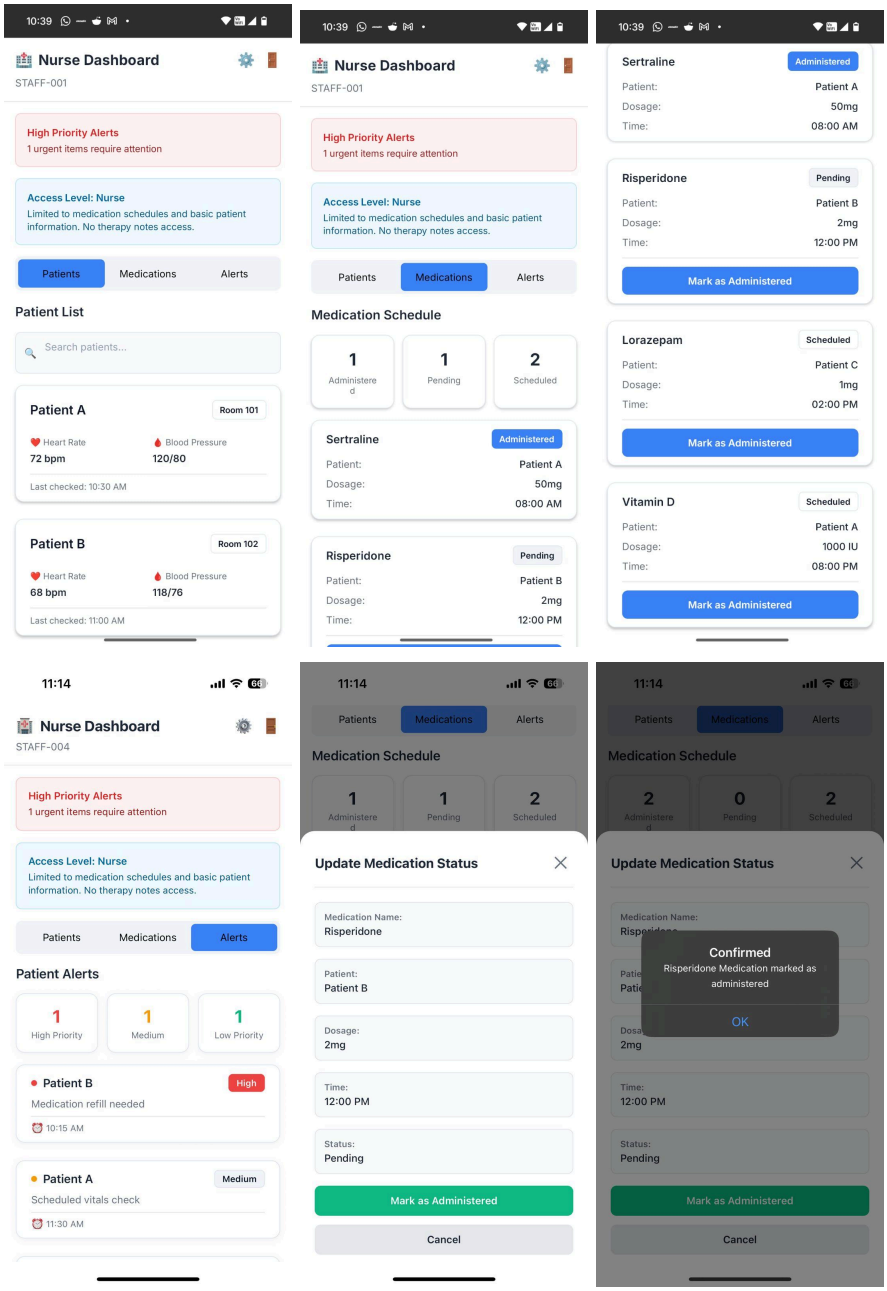

9:36
Psychologist Dashboard
Dr. STAFF-002
Access Level: Psychologist
Limited to therapy session logs and progress tracking only.
Patients
Notes
Assessments
Assessments
New Assessment
Scheduled Assessments
GAD-7
scheduled
Patient: John Doe
2026-11-20T18:30:00.000Z
Scheduled by: STAFF-002
Notes: Djjd
MMPI-2
scheduled
Patient: Mia maples
2026-11-23T18:30:00.000Z
Scheduled by: STAFF-002
Notes: Seee
Beck Depression Inventory
scheduled
Patient: Blue blue
2027-12-19T18:30:00.000Z
Scheduled by: STAFF-002


9:36
New Therapy Note
Editing
Session Type *
Initial Assessment
Follow-up
Crisis Intervention
Group Therapy
Patient Mood/State
Calm
Anxious
Depressed
Agitated
Cooperative
Progress Assessment
Improved
Stable
Declined
No Change
Session Notes *
Depressed too
End-to-End Encrypted
HIPAA Compliant · Auto-lock Enabled

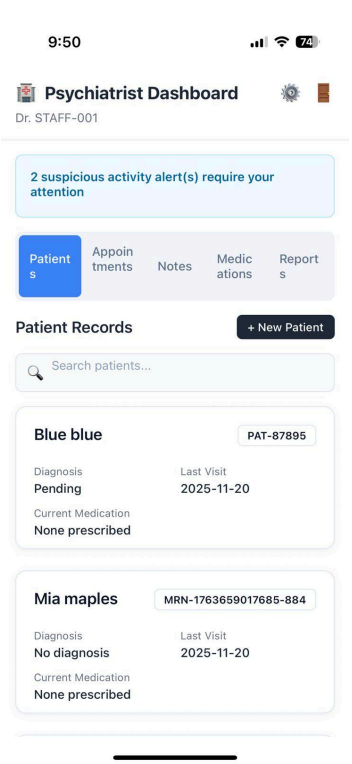

9:37
Therapy Note
Session Note
21/11/2025
4:06 AM
Patient ID: 7
Provider: psych@neurolock.com
SESSION INFORMATION
SESSION TYPE
Initial Assessment
PATIENT MOOD
Depressed
PROGRESS
Stable
SESSION NOTES
Depressed too
This note is encrypted and HIPAA compliant

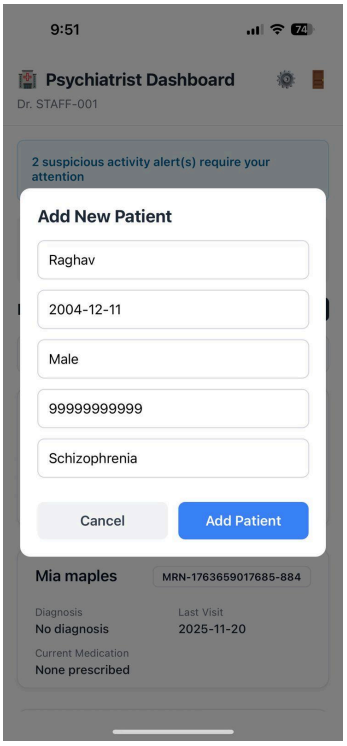

9:37
Schedule Assessment
Blue blue (PAT-87895)
Mia maples (MRN-1763659017685-884)
John Doe (PAT-001)
Scheduled Date *
2025-11-25
22 August 2022
23 September 2023
24 October 2024
25 November 2025
26 December 2026
27 January 2027
28 February 2028
Assessment Notes
Enter notes (optional)
Cancel
Schedule
Scheduled by: STAFF-002

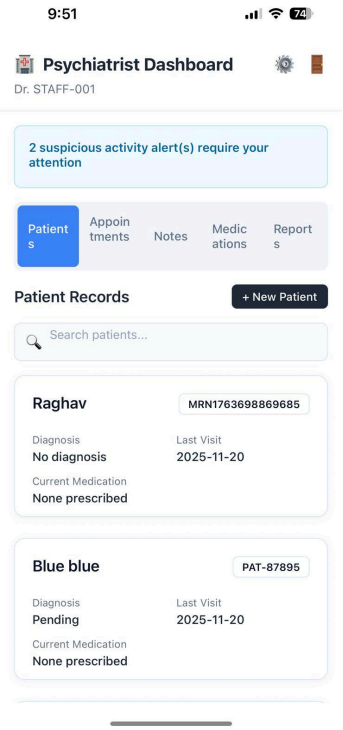
- Nurse dashboard
  - Tasks list
  - Patient list : view each patients medication schedule
  - Medication management
  - Can complete task and administrate it.
  - Medication tasks are given by the psychiatrist.

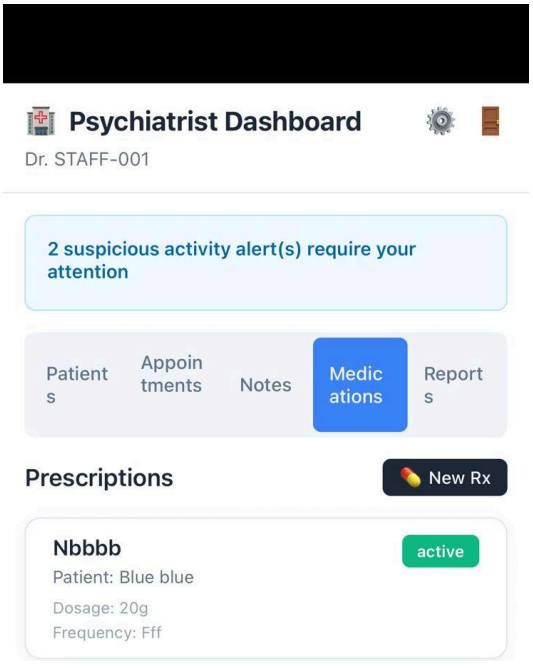


- Psychiatrist
  - Patient lists tailored to mental health workflows, assessment creation, and notes.Can also add prescriptions. Highest level of visibility.
  - Can add new patients
  - Can add appointments
  - Can see notes from psychologist
  - Can add and prescribe medicines

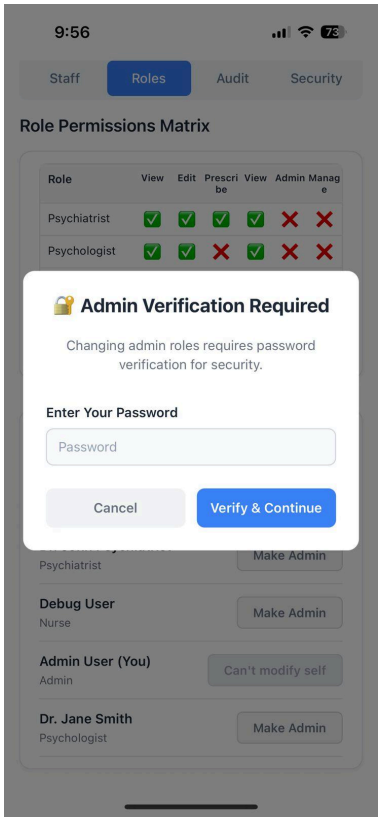


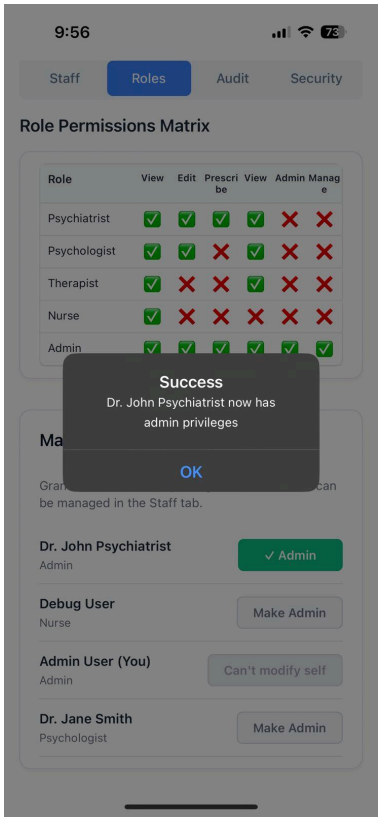


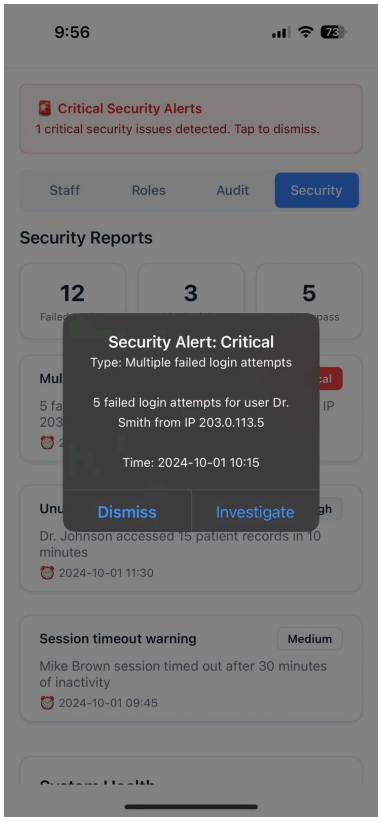


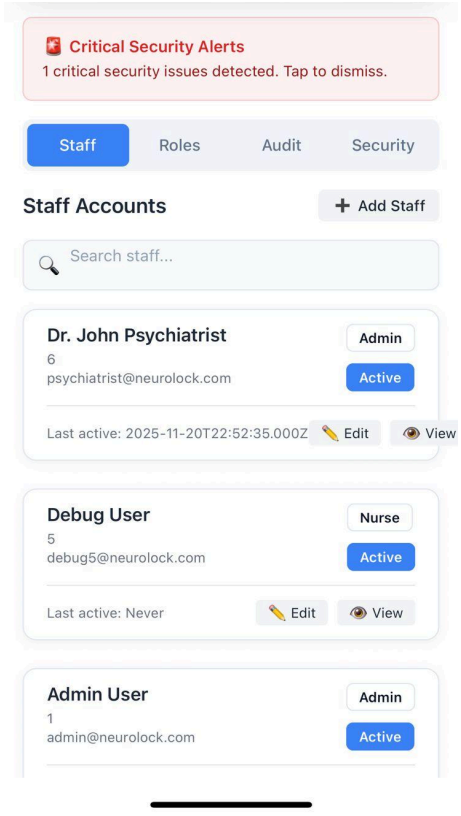


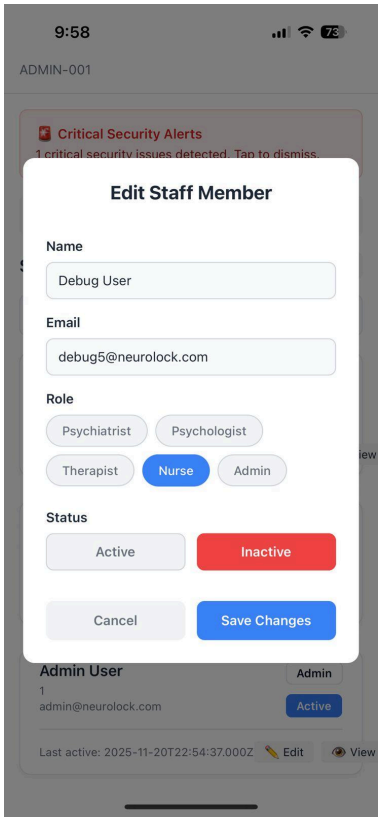
- Admin dashboard







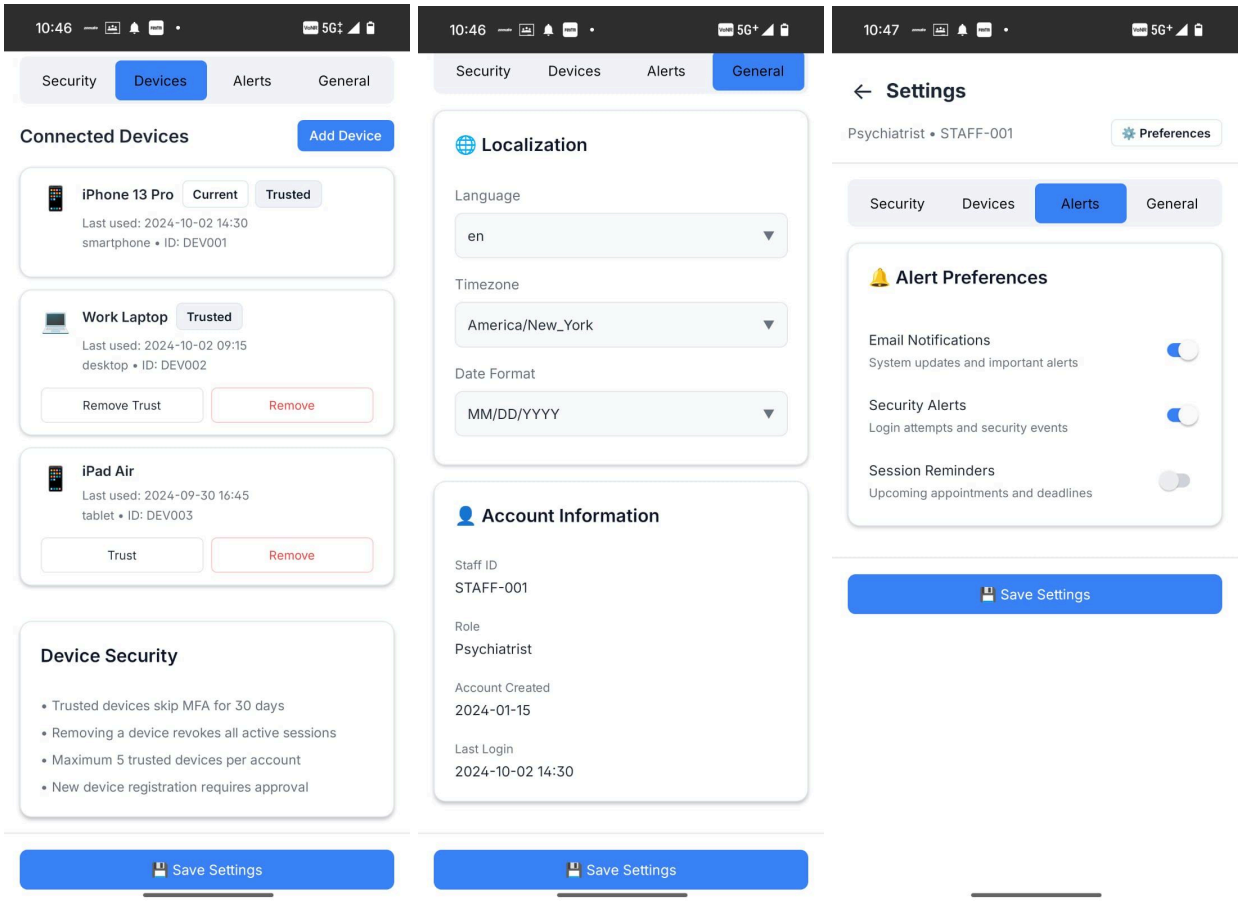




- User and role management operations, system status indicators, and audits.
- Creating new admin requires authentication again
- Editting staff requires authentication
- Security errors can be investigated
- Can check logs who added what from staff at what time
- Add new staff
- Update staff details
- Check security prospects

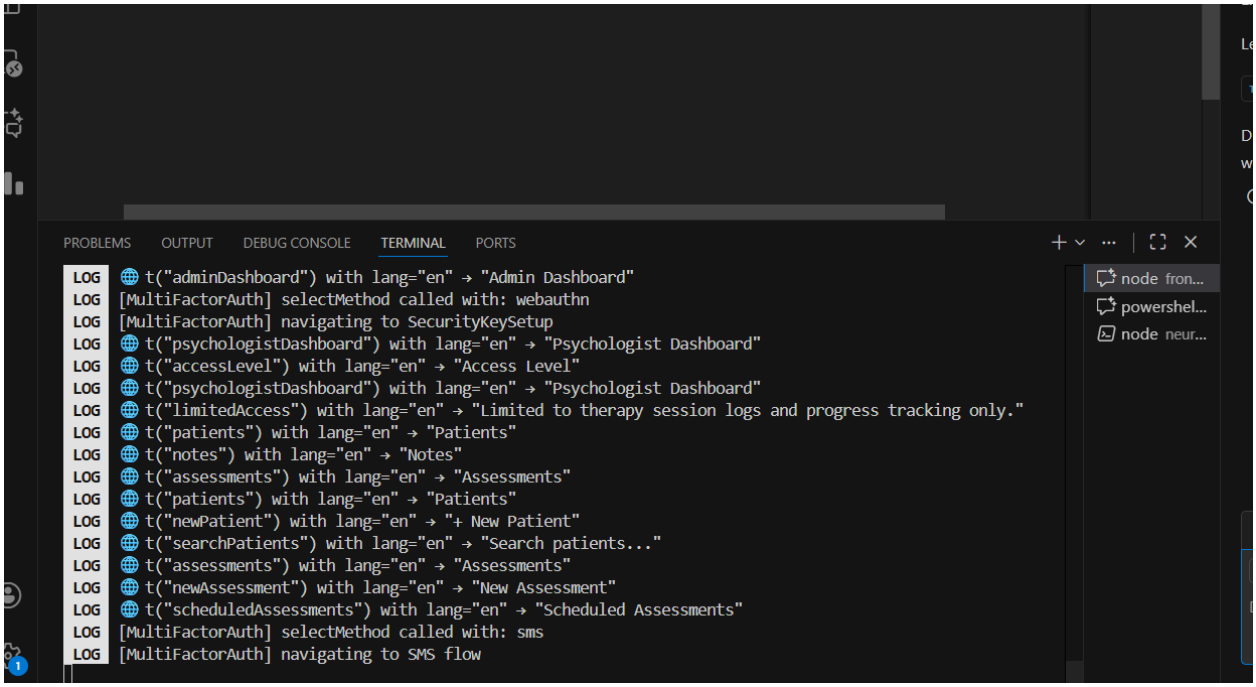
Security & Settings

- Backup codes and account recovery flows.
- Language switching readiness (several LANGUAGE_* docs indicate multi-language capability); UI labels are prepared for translation.
- Check connected devices and localization
- Alert preferences
- Language changing works.



Backend features

- Authentication and session management
    - Password hashing (bcrypt), JWT tokens for API authentication, and token expiry.



- Database migrations & rollback
    - Scripts: npm run migrate and npm run rollback (implemented under src/infra/db).
- REST APIs for staff and patient operations
    - The backend provides endpoints for registering and logging in staff, managing devices, and reading and writi patient records. Inspect src/ for endpoint list and route implementation.
- Tests and linting
    - Jest is configured (see jest.config.ts) and npm run test runs the test suite. Linting is available (npm run lint).

How to explore these features?

- Start backend with a seeded development DB (if seed scripts exist) and run Expo frontend.
- Use the app flows to:
    - Create a staff user, verify email, configure MFA and biometrics.
    - Register a device and try login/logout across devices.

    - Create a patient record and complete an assessment in offline mode, then reconnect and confirm sync.
    - Switch languages and review UI text placeholders (HARDCODED_TEXT_ANALYSIS* files and translation docs).

Developer notes

- Most components are TypeScript; open frontend/components/ to inspect props and navigation usage.
- Services and contexts (likely under frontend/services/ and frontend/context/) handle API communication and auth state.

4) Assumptions made while building the app

These assumptions guided design and implementation decisions.

- **Deployment context**: The app was developed for an internal hospital environment with staff users (not open to public signups). Authentication and access are intended to be limited by network and firewall rules. So there isn't any "sign in" option and instead there is role-based login.
- **Single-tenant by default**: The backend assumes a single hospital instance; multi-tenancy features are not implemented unless explicitly added.
- **OTP verification**: Although we have built screens for MFA using OTP generation and connection with gmail/sms/google authenticator, the logic for the same hasn't been implemented given that third-party apps usage is restricted in the scope of this project.
- **Secure transport provided by infra**: The code assumes HTTPS termination (reverse proxy/load balancer) in production; dev runs on HTTP during local testing.
- **Database choice**: Migration scripts support common SQL engines and both pg and mysql2 exist as dependencies; the project assumes a relational DB is available.
- **Device trust model**: Device registration and biometric flows are convenience layers; ultimate security is enforced server-side via tokens and session validations.
- **Translation readiness**: The app is designed to be translation-ready, but some text may still be hard-coded until translations are added.
- **Minimal external integrations**: The initial implementation focuses on internal workflows; integrations with third-party EHRs or SSO providers are out of scope unless added later.

5) Current observed security vulnerabilities by us

Below is a prioritized list of observed weaknesses (based on our repeated testing and general best-practice checks) and a concrete remediation plan for each.

Critical / High priority

- Secrets in environment and lack of secret management
    - Observation: The project uses dotenv for local config. If .env files or secrets are checked into source control, they are at risk.
    - Risk: Secret leakage, credential compromise.
    - Future Fix: Adopt a secrets manager (HashiCorp Vault, AWS Secrets Manager, Azure KeyVault) for production. Ensure .env is in .gitignore. Use CI secret injection for builds.

- No enforced HTTPS in app or token transport
    - Observation: Dev runs HTTP; production must use HTTPS.
    - Risk: Token interception, session hijacking.
    - Future Fix: Enforce HTTPS at the reverse proxy. Redirect HTTP to HTTPS. Use HSTS and secure cookie flags. Ensure mobile app endpoints use HTTPS.

- Authentication protections and rate limiting missing
    - Observation: No express middleware found that enforces rate limiting or account lockouts beyond UI screens.
    - Risk: Brute-force logins, credential stuffing.
    - Future Fix: Add express-rate-limit or an API gateway WAF; enforce account lockouts and exponential backoff, CAPTCHA on suspicious attempts, and monitoring of failed logins.

- JWT token handling and rotation concerns
    - Observation: JSON Web Tokens (JWTs) are used; ensure expiry and rotation patterns are correct.
    - Risk: Long-lived tokens can be abused.
    - Future Fix: Use short-lived access tokens, refresh tokens with rotation, store refresh tokens securely server-side and invalidate on logout/device deregistration.

Medium priority

- Input validation and parameterized DB queries
    - Observation: Without a dedicated ORM or validation library, injection risks exist.
    - Risk: SQL injection and malformed data saving.

    - Fix: Use parameterized queries or an ORM (Prisma, Sequelize, TypeORM) and validate incoming data at the API boundary (use Zod, Joi, or express-validator). Review src/infra/db for any raw query usage and refactor.

- Password policy and bcrypt cost
    - Observation: bcrypt is used but cost factor should be checked.
    - Risk: Weak passwords or insufficient hashing cost.
    - Fix: Enforce strong password rules (min length, complexity), use bcrypt cost >= 12 (evaluate per infra), and block common weak passwords via a denylist.

- Missing CSRF/XSS mitigations for web
    - Observation: The codebase includes web entry (expo web) and may expose endpoints.
    - Risk: For browser-based clients, CSRF/XSS threats can exist.
    - Fix: Serve CSP, sanitize outputs, use same-site cookie attributes, and CSRF tokens for state-changing requests when supporting web clients.

- Insufficient logging, monitoring, and audit trails
    - Observation: No centralized audit or structured logging subsystem documented.
    - Risk: Hard to detect breaches and comply with audits.
    - Fix: Add structured logging (winston/pino), integrate with a log aggregator (ELK/Cloud provider), and ensure audit events for auth and patient data operations.

Lower priority / Operational

- Outdated dependencies and supply chain risk
    - Observation: Several dependencies (Expo/react-native/ts) may have newer versions.
    - Risk: Known vulnerabilities in transitive dependencies.
    - Fix: Run npm audit, add Dependabot or Snyk, update dependencies regularly, and patch critical findings immediately.
- Lack of RBAC and least-privilege controls
    - Observation: Role-specific dashboards exist but server-side enforcement must be verified.
    - Risk: Horizontal privilege escalation.

    - Fix: Implement server-side RBAC checks for every protected endpoint. Create role matrices and tests that assert permissions.

- Data-at-rest encryption and secure storage
    - Observation: Local backups or device storage may not be encrypted.
    - Risk: Sensitive patient data exposure on lost/stolen devices or backup media.
    - Fix: Use secure storage for secrets on mobile (SecureStore on Expo), encrypt local databases at rest, and ensure backups are encrypted.

6) More fixes needed to be made in APK

- Currently, SMS and Mail do not work for having OTP being sent to them for password recovery, as these emails and numbers do not exist for verification, realistically.
- Devices in the setting work only locally
- For future purpose, we have also created a static therapist dashboard accessed via therapist@neurolock.com
- We hope to make this app better and develop further.