COURSE-CORRECT APP DOCUMETATION

Table of Contents

[2 INTRODUCTION 2](#_Toc204864663)

[2.1 PROJECT NAME AND TYPE 2](#_Toc204864664)

[2.2 DESCRIPTION 2](#_Toc204864665)

[2.3 Target Users 2](#_Toc204864666)

[2.4 OBJECTIVES 3](#_Toc204864667)

[3 SYSTEM ARCHITECTURE 3](#_Toc204864668)

[3.1 ARCHITECTURE OVERVIEW 3](#_Toc204864669)

[3.2 ARCHITECTURE LAYERS 5](#_Toc204864670)

[3.3 OPERATIONAL FLOW 6](#_Toc204864671)

[3.4 DESIGN JUSTIFICATION 6](#_Toc204864672)

[4 TECHNOLOGY STACK 7](#_Toc204864673)

[4.1 FRONTEND 7](#_Toc204864674)

[4.2 BACKEND (BaaS - SUPABASE) 8](#_Toc204864675)

[4.3 SUPPORTING LIBRARIES AND TOOLS 8](#_Toc204864676)

# INTRODUCTION

## PROJECT NAME AND TYPE

Course-Correct a Mobile Productivity Application (React Native/Expo + Supabase)

## DESCRIPTION

Course-Correct is a modular productivity tool designed for students to manage academic tasks, notes, schedules, and quizzes all in one place. The goal is to reduce the number of disconnected apps students use by integrating:

* **Course Notes** (tag-based, searchable notes)
* **Planner** (schedule tasks with date, time, and recurrence)
* **To-Do List** (daily task tracker with filters)
* **Quiz** (interactive learning via Open Trivia API)

## Target Users

* University and college students
* Self-learners using MOOCs or tutorials
* Students preparing for exams (e.g. WASSCE, SATs, etc.)

## OBJECTIVES

* Deliver a minimal yet complete academic organizer
* Enable cloud-synced and persistent user data
* Integrate secure user authentication and profile management
* Offering an offline-capable architecture in future iterations

# SYSTEM ARCHITECTURE

The system architecture for the Course-Correct application follows a client–backend service model, using **Supa base** as the BaaS (Backend-as-a-Service) platform and **React Native** for the frontend.

## ARCHITECTURE OVERVIEW

The system is organized into three primary layers:

1. **Frontend Layer (React Native)**  
   This is the mobile user interface built using React Native with Expo. It manages all user interactions, form inputs, navigation between screens, and local UI states such as loading and error displays.
2. **Middleware Layer (Supabase Client SDK)**  
   The Supabase JavaScript client connects the mobile app to backend services using secure RESTful API calls and real-time subscriptions. It handles authentication, database queries, file uploads, and responses.
3. **Backend Layer (Supabase Services)**  
   This layer includes:
   * **Authentication** for handling user sign-up, login, logout, and secure token management using JWT (JSON Web Tokens).
   * **Database** powered by PostgreSQL for storing all app data such as notes, tasks, quizzes, and user profiles.
   * **Storage** for handling file uploads, such as profile pictures.
   * **Row-Level Security (RLS)** and database policies to ensure that each user can only access their own data.

React Native Frontend

(Expo + Custom UI Screens)

Supabase Client SDK

(REST & Realtime Queries

Supabase Services

Authentication (JWT)

Realtime Postgres DB

Row-Level Security (RLS)

## ARCHITECTURE LAYERS

1. Presentation **Layer**  
   The presentation layer is developed using React Native and provides the user interface of the application. It is responsible for rendering screens such as dashboard, planner, notes, quiz interface, authentication screens, and profile settings. It manages how users interact with the app and ensures a clean, consistent experience.
2. **Logic Layer**  
   The application logic resides within React hooks and component-level state management. This layer handles tasks such as form validation, local coaching, input handling, navigation logic, conditional rendering, and UI feedback like loading indicators or error messages. It acts as the intermediary between user actions and data services.
3. **Data Access Layer**  
   This layer is responsible for communicating with the backend services through the Supabase JavaScript SDK. It sends and receives data via RESTful API calls or real-time subscriptions. All CRUD operations on notes, tasks, and profile data are managed here. This layer abstracts away direct backend interaction from the UI.
4. **Backend Layer**  
   The backend services are hosted entirely on Supabase. These services include:

* **Authentication**: A secure email and password-based authentication system with session handling using JSON Web Tokens (JWT).
* **Database**: A PostgreSQL database stores structured information such as user profiles, course notes, planner tasks, to-dos, and quiz metadata.
* **Storage**: A secure storage bucket holds user-uploaded content, including profile images. Access to files is scoped to each user.  
  Supabase’s services allow for rapid backend deployment with robust scaling and minimal configuration.

1. **Security Layer**  
   Supabase enforces **Row-Level Security (RLS)** on all critical database tables. This ensures that users can only read or modify records that are associated with their own user ID. Each table implements fine-grained access control policies defined at the SQL level. JWT tokens from authenticated sessions are verified automatically with each request, providing both stateless security and scalability. Profile updates and media uploads are scope to individual users and cannot be accessed across a cross-account.

## OPERATIONAL FLOW

When a user installs and opens the Course-Correct app, the system checks for an active session using Supabase’s authentication state. If a session exists, the user is directed to the main dashboard; otherwise, they are prompted to register or log in. Upon authentication, a session token is issued and all further interactions with Supabase services are scoped to that session.

Users interact with modules such as the Course Notes, Planner, or To-Do List. Each interaction — such as adding a note or marking a task as complete — triggers a call to the Supabase backend, where the data is securely stored, retrieved, or updated. At no point does the client handle raw credentials or direct database access; all operations are abstracted and secured through the Supabase client SDK.

## DESIGN JUSTIFICATION

This architectural approach ensures a lightweight and scalable system with minimal backend maintenance. By using Supabase as the BaaS, development time is reduced significantly, while offering enterprise-grade security features and built-in support for future scalability, including real-time updates and offline support.

# TECHNOLOGY STACK

The Course-Correct application is built using a carefully selected set of modern technologies and tools, chosen for their reliability, ease of integration, and support for rapid mobile development. The stack comprises both frontend and backend components, supported by a robust toolchain and open-source libraries.

## FRONTEND

* **React Native (with Expo):**  
  Used to build a cross-platform mobile application that runs on both Android and iOS. Expo simplifies the development workflow with features like hot-reloading, push notifications, and OTA updates.
* **React Navigation:**  
  Provides screen navigation through stack and tab navigators for seamless user experience across modules (e.g. Home, Courses, Planner, Profile).
* **React Native Vector Icons (Ionicons):**  
  Adds visual context and polish through a wide range of vector-based icons, improving usability and accessibility.
* **React Native Paper (optional):**  
  A UI component library that supports Material Design patterns. Useful for structuring clean and accessible interfaces.
* **JavaScript (ES6+):**  
  The programming language used throughout the app to handle logic, navigation, API interactions, and state management using hooks like useState and useEffect.

## BACKEND (BaaS - SUPABASE)

* **Supabase Auth:**  
  Provides secure email/password authentication and session management using JSON Web Tokens (JWT). Handles registration, login, password reset, and session tracking.
* **Supabase Database (PostgreSQL):**  
  A relational database that stores structured data such as user profiles, notes, planner tasks, to-do items, and quiz metadata. The database is secured using Row-Level Security (RLS) policies.
* **Supabase Storage:**  
  Used to store and serve media files, including user-uploaded profile pictures. Storage access is tightly scoped to authenticated user sessions.
* **Supabase Client SDK:**  
  A JavaScript-based client that enables API calls to Supabase services from within the mobile application. It handles real-time queries, mutations, and authentication natively.

## SUPPORTING LIBRARIES AND TOOLS

* **react-native-community/datetimepicker:**  
  Enables users to select dates and times for planner tasks using a native platform-compatible UI.
* **react-native-picker/picker:**  
  Adds dropdown selection menus for features like task recurrence or tag categorization.
* **Toast Messages (react-native-toast-message):**  
  Displays real-time feedback messages for actions like login failure, successful submission, or validation errors.
* **Expo Image Picker (Media Access):**  
  Enables profile image uploads using the device’s gallery or camera, integrated with Supabase Storage.

# DATABASE DESIGN

The backend of the Course-Correct application is powered by **Supabase PostgreSQL**, which offers a scalable, relational data model with robust support for row-level security. The database schema was designed to support user-specific data segregation, efficient querying, and extensibility for future features such as reminders or course categorization.

## DESIGN PRINCIPLES

The design of the database followed these core principles:

* **Data normalization** to reduce redundancy and improve consistency.
* **User-centric ownership** of data to ensure every record is tied to a specific authenticated user.
* **Scalability** to support growing feature sets like tagging, media uploads, and real-time updates.
* **Security-first access** using Supabase’s Row-Level Security (RLS), ensuring strict access boundaries for user data.

## CORE TABLES AND THEIR ROLES

1. **Profiles Table**  
   This table extends the default Supabase Auth user system by storing additional user information such as first name, last name, and profile picture URL. Each profile record is linked to the unique id from the Supabase Auth system, serving as a foreign key across all user-generated content.
2. **Course Notes Table**  
   This table stores all notes created by a user. Each note contains a title, content body, optional tags (which serve as categories), and a status marker to indicate whether the note is complete or in progress. Notes are stored with a user\_id field to ensure data isolation. Timestamp fields capture when each note was created or modified.
3. **Planner Tasks Table**  
   This table holds scheduled academic tasks or reminders. Each entry includes a title, a datetime field indicating when the task is due, and a recurrence field to represent patterns such as daily, weekly, or one-time tasks. Tasks are ordered chronologically and tied to the user via a user\_id.
4. **To-Do Items Table**  
   The to-do list is a lightweight, checklist-style table that stores short-term tasks. Each record includes the content of the task, a boolean field to indicate whether it is completed, and a timestamp to sort items by creation or due date. All items are scoped by user\_id.
5. **Quiz Metadata (Optional/Future Scope)**  
   Though quizzes currently rely on external APIs, a future schema extension may include a quizzes table for storing saved questions, user scores, or custom quiz sets. This would support offline usage and allow performance tracking over time.

## SECURITY MODEL (ROW-LEVEL SECURITY)

Every core table is protected by **Row-Level Security (RLS)** policies. These policies enforce that:

* A user can only select, insert, update, or delete records where the user\_id matches their authenticated session ID.
* Profile updates are scoped to the user's own profile row.
* Storage objects (e.g., profile images) are tied to unique folders named after the user ID, preventing cross-access.

These constraints are defined at the database level using SQL policy statements and remain in effect regardless of the API call origin, providing a robust, secure data environment.

## DATA TYPES AND CONSTRAINTS

Each table uses UUIDs as primary keys to ensure uniqueness and compatibility with Supabase’s authentication system. Fields such as created\_at and updated\_at are timestamped using default PostgreSQL functions. Boolean flags are used for task completion and note status. Arrays are used where appropriate is, for example, the tags field in notes uses a text array to support multi-tag filtering.

# FEATURES OVERVIEW

The Course-Correct application is designed around five core functional modules. Each module addresses a key aspect of academic productivity and is accessible through a centralized dashboard. The design philosophy emphasizes clarity, ease of use, and cross-functionality.

## USER AUTHENTICATION AND PROFILE MANAGEMENT

Users register and log in using a secure email and password authentication system powered by Supabase Auth. Upon successful authentication, a user profile is created in the profiles table to store additional information including first name, last name, and a profile picture.

Features:

* User registration with email verification
* Login and logout functionality
* Password update
* Edit profile (first name, last name, image upload)
* Session persistence via JWT

## DASHBOARD

The dashboard serves as the home screen once a user is authenticated. It provides a greeting, displays the user’s name, and presents quick-access tiles to the major modules of the app.

Features:

* Personalized greeting message
* Navigation to Course Notes, Planner, Todo, and Quiz screens
* Clean, grid-based tile layout for core actions
* Compact header with profile avatar and notification icon (optional)

## COURSE NOTES

The Course Notes module allows students to write, categorize, and manage their learning content. Notes are saved in the cloud and can be edited or deleted at any time. A tagging system allows users to organize notes by subject or theme.

Features:

* Create, edit, and delete notes
* Add titles, tags, and content
* Mark notes as completed
* Filter notes by tag
* Search notes by title or content (planned enhancement)
* Swipe-to-edit and swipe-to-delete interactions

## PLANNER

The Planner module is a scheduling tool that enables users to set up tasks with a specified date, time, and recurrence pattern. It functions like a lightweight calendar tailored for students.

Features:

* Add new planner tasks with datetime picker
* Select recurrence (One-time, Daily, Weekly)
* View tasks ordered by due date
* Swipe to delete or edit tasks
* Visual cues for upcoming or overdue tasks

## TO-DO LIST

The To-Do module serves as a quick task tracker for daily academic or personal goals. Tasks are kept minimalistic, with optional filters to view pending, completed, or all tasks.

Features:

* Add short-form to-do items
* Mark tasks as completed with checkbox
* Delete tasks via swipe gesture
* Filter tasks by completion status
* Sort tasks by most recent

## QUIZ MODULE

The Quiz feature uses the Open Trivia Database API to provide general knowledge questions for practice and cognitive stimulation. Though currently limited to external quizzes, future updates may support academic-focused quizzes and offline access.

Features:

* Fetch random trivia questions from Open Trivia API
* Multiple-choice answers
* Immediate feedback on correct/incorrect answers
* Plans for quiz result storage and performance tracking

# USER INTERFACE WALKTHROUGH

The Course-Correct application adopts a clean, minimal interface design with consistent navigation and feedback. The user interface (UI) was built using React Native components and styled for accessibility and clarity. Below is a walkthrough of the key UI screens, each accompanied by a description of their functionality and user interactions.

## SPLASH SCREEN

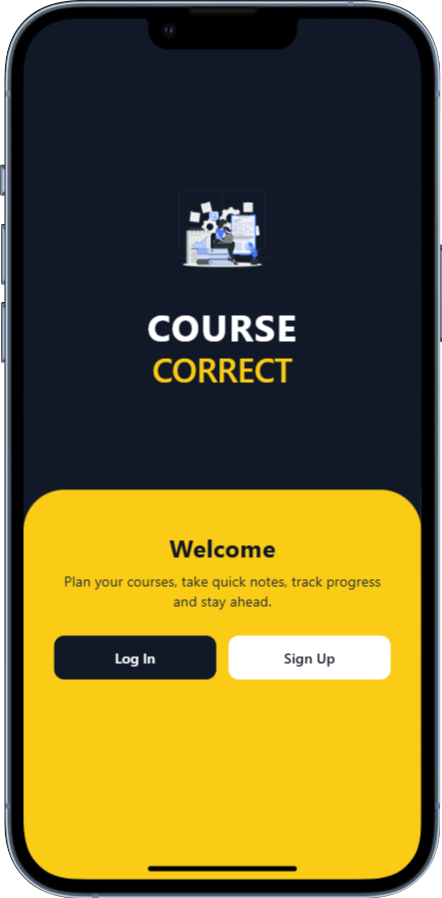
This is the initial screen shown when the app launches. It includes a fade-in animation with the app’s logo and performs a session check to determine whether to direct the user to the dashboard or landing page.

A screen shot of a cell phone

AI-generated content may be incorrect.

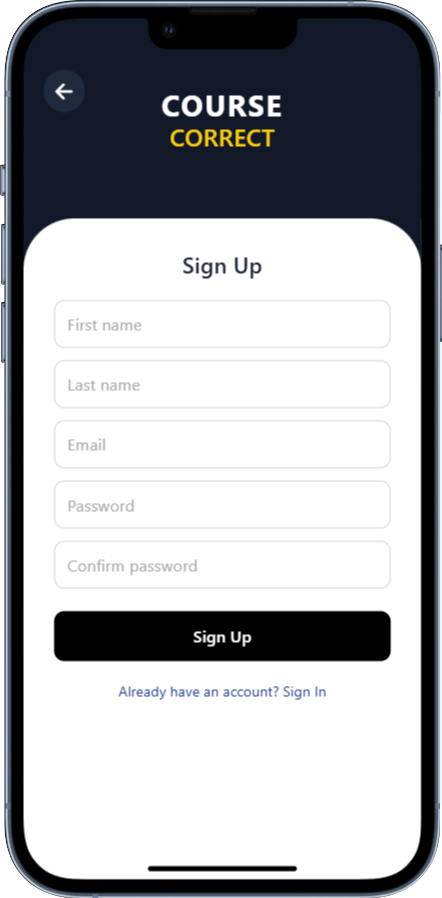
## LANDING SCREEN

For users who are not logged in, the landing screen provides access to the login and registration pages. It acts as the entry point to the app for first-time users.



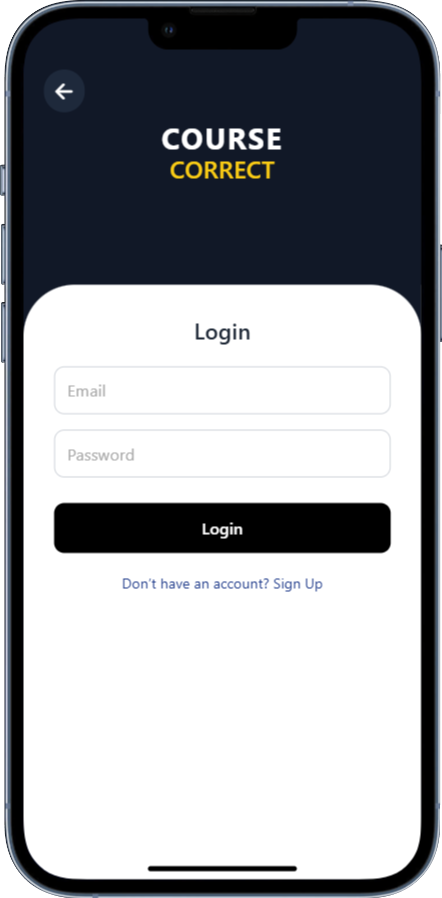
## REGISTER SCREEN

This screen allows new users to create an account using their email, password, and personal information (first name and last name). Input validation ensures that passwords match and fields are not left empty.



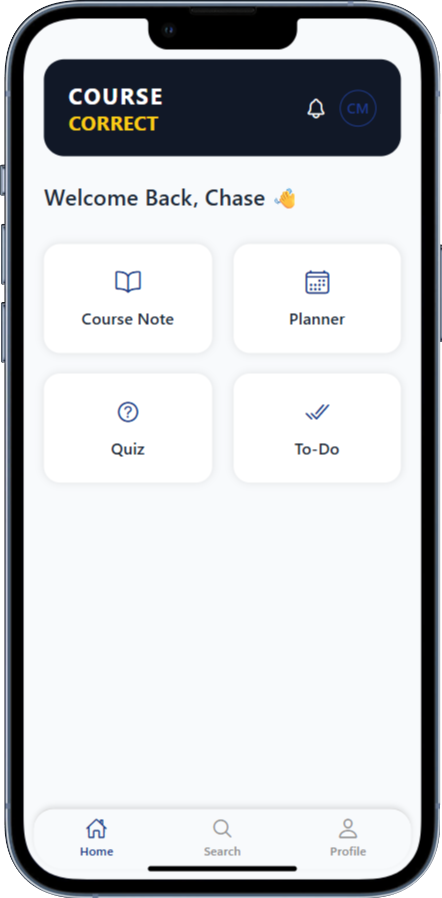
## LOGIN SCREEN

Registered users log in via this screen. Error messages appear for incorrect credentials or unverified accounts. Successful login redirects users to the dashboard.



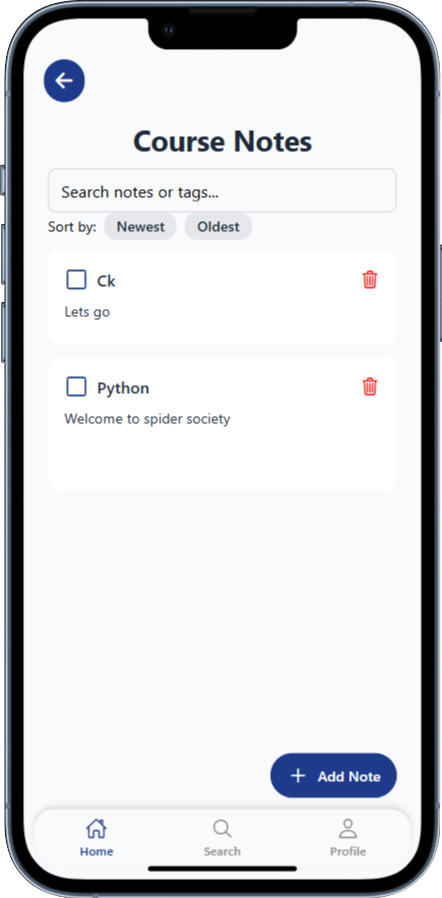
## DASHBOARD

The dashboard is the central navigation hub of the app. It includes a personalized greeting, action tiles for each module (Course Notes, Planner, Quiz, To-do), and a floating menu for additional features.



## COURSE NOTES

This module allows users to create and manage learning notes. Users can add tags, search for notes by title, and mark notes as done. Notes can be edited or deleted via swipe gestures.



## PLANNER

The Planner allows users to schedule academic tasks using a date and time picker. Tasks may be recurring (daily/weekly) and can be updated or removed.

A screenshot of a phone

AI-generated content may be incorrect.

## TO-DO LIST

A lightweight task manager for personal and academic checklist items. Tasks can be added, marked as completed, filtered by status, and deleted via swipe.

A screenshot of a phone

AI-generated content may be incorrect.

## QUIZ

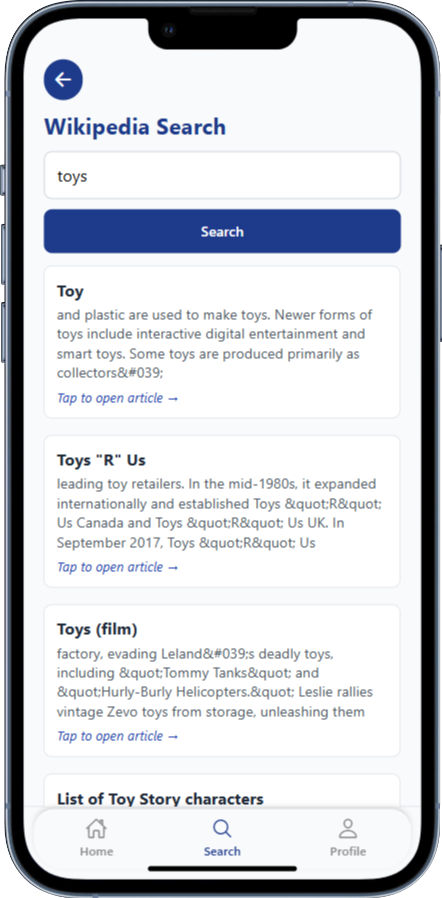
This screen fetches and displays random trivia questions. Users select from multiple-choice options and receive immediate feedback.

A screenshot of a cell phone

AI-generated content may be incorrect.

## SEARCH PAGE

The Search module allows users to explore academic topics and general knowledge using Wikipedia’s API. It serves as a lightweight research assistant directly within the app, helping students quickly access summaries and definitions without leaving the platform.



## PROFILE SCREEN

Displays the user’s full name, email, profile image and stats of your dashboard. From here, users can upload a new profile photo and log out.

A screenshot of a phone

AI-generated content may be incorrect.

# AUTHENTICATION AND SECURITY

Authentication and data security are central to the design of the Course-Correct application. The system leverages Supabase's built-in authentication and authorization features to ensure that only authorized users can access and manipulate their data. Security is enforced at every level of architecture, from login credentials to row-level database access.

## AUTHENTICATION SYSTEM

The app uses **Supabase Auth**, which provides a secure email and password authentication flow. Upon successful login or registration, Supabase issues a **JSON Web Token (JWT)**, which is used to authenticate all subsequent API requests. The session is maintained in memory on the client and is refreshed automatically.

**Key authentication features include:**

* Email/password registration and login
* Session persistence with auto-refresh
* Password change and reset capability
* Secure logout with token invalidation

## ROLE-BASED ACCESS CONTROL

Supabase Auth automatically links each authenticated user to a unique identifier (user\_id), which is then used throughout the application to scope access to resources such as notes, tasks, or profile information.

All records in the application’s core tables include a user\_id foreign key. This identifier is used in queries to ensure that each user only accesses their own data.

## ROW-LEVEL SECURITY(RLS)

To enforce data isolation at the database level, **Row-Level Security (RLS)** policies are applied to all user-scoped tables including:

* profiles
* course\_notes
* planner\_tasks
* todos

These policies ensure that:

* A user can only SELECT, INSERT, UPDATE, or DELETE rows where the user\_id matches their own.
* Even if a user attempts to tamper with requests at the network level, Supabase will reject queries that violate these policies.

## STORAGE SECURITY

All uploaded media (such as profile images) is stored in Supabase’s Storage module, organized in user-specific buckets or folder paths. Access rules prevent users from reading or modifying any file that does not belong to them.

## FRONTEND SAFEGUARDS

The React Native app includes additional client-side validations and user experience guards:

* Empty field validation for all forms
* Password matching during registration and change
* Error messages for failed logins, invalid inputs, or network issues
* Confirmation modals for sensitive actions like deleting notes or tasks

# TESTING AND DEBUG LOGS

Testing of the Course-Correct application was conducted across multiple functional modules, with the goal of ensuring component reliability, correct backend integration, and consistent user experience across platforms. Logs were generated through the Expo CLI and in-device testing, and critical issues were resolved before feature confirmation.

## TESTING OBJECTIVES

The main objectives of testing included the following:

* Verifying that all features work as intended under normal and edge-case conditions.
* Ensuring that all user-generated data is stored, retrieved, and updated correctly in the Supabase database.
* Validating that row-level security is enforced and unauthorized access is blocked.
* Confirming session persistence and re-authentication flows.
* Identifying performance bottlenecks and interface bugs, especially in gesture and form-handling logic.

## FUNCTIONAL TESTING RESULTS

Each feature was tested in isolation and within complete user flows. Below are the results by module.

**Authentication**

* Registration and login operations function correctly.
* Error handling displays informative feedback for invalid credentials or duplicate accounts.
* Sessions persist correctly across app restarts.

**Profile Management**

* Users can update their names and upload a profile image.
* Password changes are securely processed and enforced immediately.

**Dashboard Navigation**

* All navigation links (bottom tab and action tiles) function as expected.
* Greeting message reflects the current user’s name upon login.

**Course Notes**

* Users can create, update, and delete notes successfully.
* Tag-based filtering works as expected.
* Swipe gestures for edit and delete behavior correctly on supported devices.

**Planner**

* Tasks can be scheduled with date, time, and recurrence options.
* Tasks are sorted by their scheduled time.
* Edit and delete operations reflect instantly in the database.

**To-Do List**

* Users can mark tasks as completed and filter by completion status.
* Sorting and swipe-to-delete are operational and intuitive.

**Quiz**

* Quiz questions are fetched from the Open Trivia API.
* User selections trigger immediate correctness feedback.
* Network failure during fetch is handled gracefully.

**Search (Wikipedia API)**

* User-entered terms return relevant summaries.
* Errors for empty input or no results are handled clearly.

## PLATFORM AND DEVICE TESTING

Testing was conducted on the following devices using Expo Go:

* Android 12 (Samsung Galaxy A32, Google Pixel 4a)
* Android 10 (Infinix Hot)
* iOS 15 (iPhone XR via Expo iOS simulator)

No significant platform inconsistencies were identified. Minor variation was observed in keyboard overlay behavior between Android and iOS.

## COMMON DEBUG SCENARIOS AND FIXES

* Issue: Profile name changes were not reflected immediately.  
  Resolution: Implemented a data refresh after save.
* Issue: Swipe-to-delete did not work on the Planner screen.  
  Resolution: Adjusted the layout and gesture handling logic.
* Issue: Notes modal retained previous input when reopened.  
  Resolution: Reset local state upon modal dismissal.
* Issue: “Email already in use” error was not shown clearly.  
  Resolution: Introduced custom toast message with early termination of the flow.

# APPENDICES

This section contains supplementary materials that support the content presented in earlier chapters. It includes visual references, code excerpts, and technical specifications for developers, testers, or stakeholders reviewing the Course-Correct application.

## EXTERNAL APIs USED

* **Supabase Auth, Database, Storage** – https://supabase.com/docs
* **Wikipedia API** – https://www.mediawiki.org/wiki/API:Main\_page
* **Open Trivia Database** – https://opentdb.com/api\_config.php

## DEVELOPMENT ENVIRONMENT

 React Native via Expo CLI

 Node.js (v18+)

 Visual Studio Code (recommended editor)

 Supabase project hosted at: https://reyikvrxcocgpshxfuwf.supabase.co