MINOR PROJECT REPORT

ON

"MindMix"

(A Modern Flutter Guessing Game with Firebase Backend)

SUBMITTED BY:

Ayush Kumar Thakur

UID- 24MCA20322(5'B)

UNDER THE GUIDANCE OF: Ms.

Winky Bhatiya

ON**'November, 2025'**

University Institute of Computing

Chandigarh University

Mohali,Punjab, 140413

CERTIFICATE

This is to certify that the project report entitled **"MindMix: A Modern Flutter Guessing Game with Firebase Backend"** has been successfully completed and submitted by **Ayush Kumar Thakur (UID-24MCA20322)** of Chandigarh University - UIC, in partial fulfillment of the requirements for the subject **AMAD (Application Modeling and Development) (24CAP-705)**, under the guidance and supervision of **Ms. Winky Bhatiya** during the academic year 2024-2026.

This work embodies a full-stack mobile application. The student has developed a cross-platform application using **Flutter** and integrated real-time backend services using **Firebase**. The project focuses on building a functional, user-centered, and scalable mobile game, complete with user authentication and a cloud database.

------------------------------   ------------------------------

SIGNATURE

Dr. Krishan Tuli
HEAD OF THE DEPARTMENT
(University Institute of Computing)

SIGNATURE

Ms. Indu Sharma
PROJECT SUPERVISOR
(Assistant professor)
(University Institute of Computing)

ABSTRACT

The project titled **"MindMix: A Modern Flutter Guessing Game with Firebase Backend"** documents the development of a full-stack, cross-platform mobile application. The primary objective was to build an engaging and modern memory-based game from an initial UI sketch, implementing both a sophisticated frontend in Flutter and a robust backend using Google's Firebase platform.

The application features a suite of mini-games (Color Guess, Number Guess, and Alphabet Guess) based on a core "memorize, mix, and find" logic. This challenges the user's short-term memory and provides an engaging user experience.

The development process followed an iterative methodology. It began with translating a static UI drawing into a functional Flutter application, followed by implementing complex game logic and state management using `setState`. The application was then refactored for a modern, minimalist UI, incorporating features like light/dark theme toggling.

The final and most critical phase involved integrating a cloud backend. **Firebase Authentication** was implemented to manage user registration (Sign Up) and login (Login) via Email/Password. **Cloud Firestore** was used as a NoSQL database to persistently store user profile information (such as name and email), linking it to their unique authentication UID. The app securely manages user sessions, displaying the correct interface based on the user's auth state.

This project serves as a comprehensive case study in full-stack mobile development, demonstrating the power of Flutter for building beautiful UIs and Firebase for providing a scalable, serverless backend.

1. Introduction and Project Aim

Project Title:

MindMix: ***A Modern Flutter Guessing Game with Firebase Backend***

Overview:

The primary aim of this project was to design, develop, and deploy a complete, full-stack mobile application from a basic UI concept. The project bridges the gap between a simple frontend prototype and a real-world, data-driven application by integrating a powerful cloud backend for user management and data persistence.

Project Description:

"MindMix" is a mobile application, built with Flutter, that hosts a collection of memory-based guessing games. The core gameplay loop, "Memorize, Mix, and Find," requires users to remember the initial state of game cards (Colors, Numbers, or Alphabets), press a "MIX" button which shuffles and hides them, and then correctly identify the new location of a target item.

The application is built on a modern, minimalist UI design and is not just a standalone app. It is connected to a powerful **Backend-as-a-Service (BaaS) platform, Google Firebase.** This integration elevates the app from a simple game to a complete service.

Key Features & Objectives:

The core objectives for this project were as follows:

1. **Translate Concept to Code:** To take a hand-drawn UI sketch and translate it into a fully functional, responsive, and aesthetically pleasing mobile application using the Flutter framework.
2. **Implement Complex Game Logic:** To develop a dynamic and reusable game logic for multiple game types (Color, Number, Alphabet) based on state management.
3. **Integrate Firebase Backend:** To implement a secure and scalable backend for user management.
    - **Authentication:** Integrate **Firebase Authentication** to provide a complete Email/Password based login and sign-up flow.
    - **Database:** Integrate **Cloud Firestore** to store and retrieve user profile data (Name, Email), linking it to their auth UID.
4. **Manage User Sessions:** To use Firebase's real-time auth state listeners (`authStateChanges`) to manage the user's session, showing the login screen when logged out and the game home screen when logged in.

5. **Develop a Modern UI/UX:** To create a clean, minimalist, and intuitive user interface that includes features like a navigation drawer, light/dark theme support, and a dedicated user profile page.

2. System Architecture and Technology Stack

System Architecture:

The application follows a client-server architecture, utilizing Firebase as a serverless backend (BaaS).

1. **Client (Frontend):** A single Flutter codebase (written in Dart) that runs on both Android and iOS. The client is responsible for rendering the UI, managing all game logic and local state, and communicating with Firebase services.
2. **Backend (BaaS - Firebase):**
   - **Firebase Authentication:** Acts as the entry gate. It handles all user registration and login requests, securely managing user credentials and sessions.
   - **Cloud Firestore:** Acts as the database. It stores non-auth user information in a `users` collection. Each user is given a document, named after their unique `uid` from Firebase Auth, which stores their name and email.

A simple data flow for login is:

```
Flutter App (Login Screen) -> Firebase Auth (Verifies credentials) -> Flutter App
(Listens to auth change) -> Cloud Firestore (Fetches user data) -> Flutter App (Shows
Profile)
```

Technology Stack:

- **Programming Language:** Dart
- **Framework:** Flutter SDK
- **Backend:** Google Firebase (Backend-as-a-Service)
- **Core Services:**
  - `firebase_core`: For initializing the Firebase app.
  - `firebase_auth`: For all user authentication tasks.
  - `cloud_firestore`: For the NoSQL cloud database.
- **IDE:** Android Studio / Visual Studio Code
- **Database Model:** NoSQL (Document-based)

3. Software & Hardware Requirements

Software Requirements:

- **Development:** Flutter SDK (3.x.x), Dart SDK
- **IDE:** Android Studio (v2023.x) or Visual Studio Code
- **Backend:** A Google Firebase Project
- **Flutter Packages:** `firebase_core`, `firebase_auth`, `cloud_firestore`
- **OS:** Windows 10/11, macOS, or Linux
- **Testing:** Android Emulator (API 30+) or a physical device.

Hardware Requirements:

- **RAM:** Minimum 8GB (16GB Recommended)
- **Disk Space:** Minimum 5GB for SDKs and IDEs
- **CPU:** x86_64 architecture, 2nd Gen Intel Core or newer

4. Implementation (Design & Process)

The project was developed in four distinct phases:

Phase 1: Initial UI Prototyping

The project began with the user-provided UI sketch. This drawing was analyzed to identify key components: an `AppBar` with a menu and profile, a main content area (a `GridView`), and a `BottomNavigationBar`. A static Flutter app was built to replicate this layout, using placeholder widgets.

Phase 2: Core Game Logic Implementation

This phase focused on making the app interactive.

1. **Game Development:** The `ColorGuessPage` was built first. A `List` of `Color` objects was used to populate a `GridView`.
2. **"Mix & Find" Logic:** The core logic was developed:
   - **Remember:** The user is shown 4 colored cards. A target color is chosen and stored.
   - **Mix:** When the "MIX" button is pressed, the `List<Color>` is shuffled. A `bool _isMixed` is set to `true`, which rebuilds the UI to show hidden cards (e.g., a grey card with a `?` icon) instead of the colors.
   - **Find:** The UI prompts the user to "Find: [Target Color]".
   - **Guess:** When the user taps a card, their guess is compared to the target. All cards are revealed, and a `SnackBar` (Green for correct, Red for wrong) provides feedback.
   - **Score:** A local `int _score` variable is incremented on a correct guess.

3. **Replication:** This logic was successfully abstracted and applied to the `NumberGuessPage` (using a `List<int>`) and `AlphabetsGuessPage` (using a `List<String>`).

Phase 3: UI/UX Refinement & Features

The initial functional app was visually basic. This phase focused on creating a modern user experience.

- **Login Screen:** A new, "cool" login screen was designed with a gradient background, app icon, and sleek, card-based input fields.
- **Minimalist UI:** The game screens were redesigned. `GameCard` widgets were created for a softer, card-based layout with shadows.
- **Themeing:** A central theme file (`_buildTheme`) was created to manage `ThemeData` for both Light and Dark modes.
- **Navigation:** The `AppBar` and `AppDrawer` were made functional. The drawer now includes a "Dark Theme" toggle switch that updates the app's `ThemeMode`.
- **Pages:** The `ProfilePage` (static) and `ScorePage` were created and linked.

Phase 4: Firebase Backend Integration

This was the final and most complex phase, converting the app to a full-stack application.

1. **Firebase Setup:** A new Firebase project was created. The `flutterfire_cli` was used to configure the Flutter app, which automatically generated the `lib/firebase_options.dart` file.
2. **Dependencies:** `firebase_core`, `firebase_auth`, and `cloud_firestore` were added to `pubspec.yaml`.
3. **Initialization:** The `main()` function was modified to be `async` and initialize Firebase before running the app:

```
void main() async {
  WidgetsFlutterBinding.ensureInitialized();
  await Firebase.initializeApp(
    options: DefaultFirebaseOptions.currentPlatform,
  );
  runApp(const GuessingGameApp());
}
```

4. **Authentication Flow:**
    - The `AuthPage` was completely rebuilt. It now uses `TextEditingControllers` for Email and Password and a toggle to switch between Login and Signup forms.
    - **Signup:** Calls `FirebaseAuth.instance.createUserWithEmailAndPassword()`. On success, it retrieves the `UserCredential` and calls the Firestore function (see below).
    - **Login:** Calls `FirebaseAuth.instance.signInWithEmailAndPassword()`.

- Error handling was added using `try-catch` blocks to show `SnackBar` alerts for "Wrong Password" or "User Not Found."

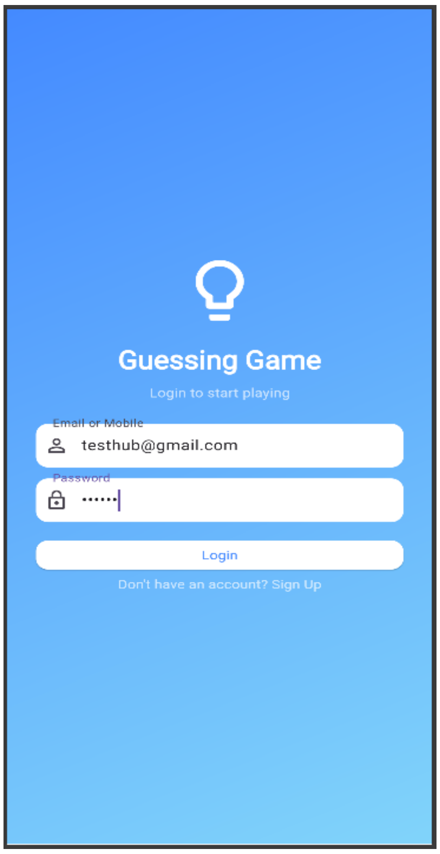
5. **Firestore Database (User Profiles):**
    - When a user signs up successfully, a new function is called to save their data to Firestore.
    - A `users` collection is used. A new document is created *with the same ID as the user's Auth UID*.
    - This document stores the user's name (from the `_nameController`) and email.

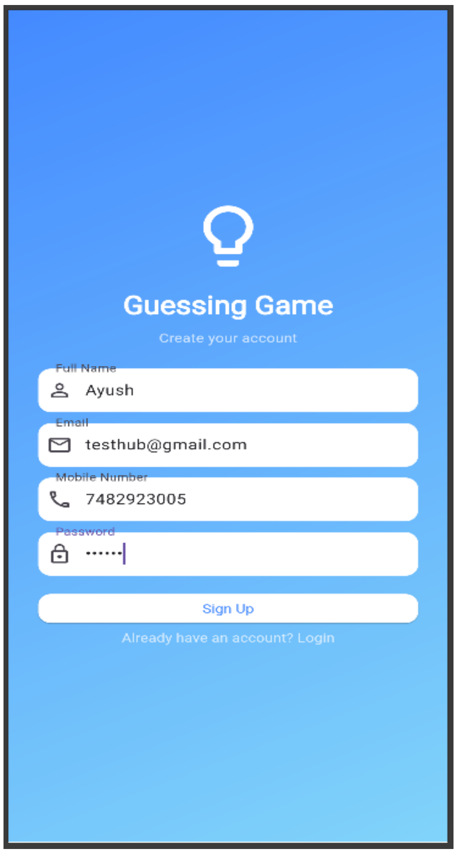
```
// On Signup Success
UserCredential userCredential = await _auth.createUserWithEmailAndPassword(...);
await
FirebaseFirestore.instance.collection('users').doc(userCredential.user!.uid).set({
  'name': _nameController.text,
  'email': _emailController.text,
});
```

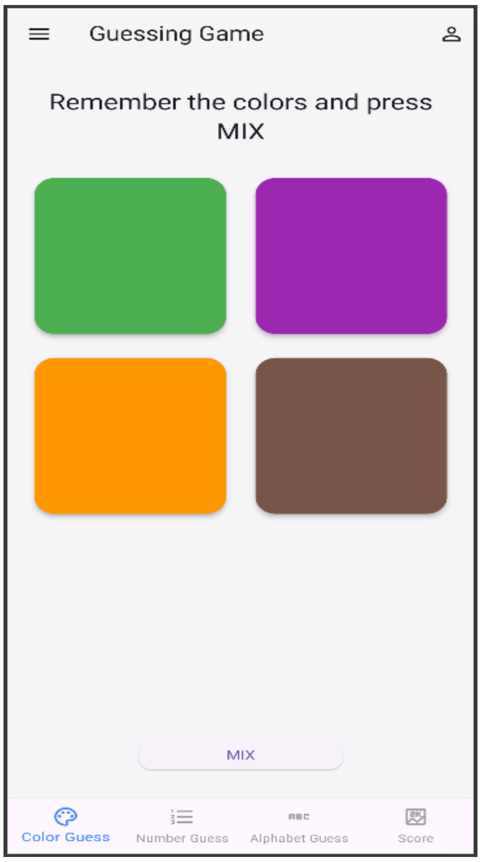
6. **Real-time Session Management:**
    - The `home` of `MaterialApp` was replaced with a `StreamBuilder`.
    - This `StreamBuilder` listens to `FirebaseAuth.instance.authStateChanges()`.
    - **Case 1 (Logged Out):** If `snapshot.hasData` is `false`, the `AuthPage` is shown.
    - **Case 2 (Logged In):** If `snapshot.hasData` is `true`, a `FutureBuilder` is shown. This `FutureBuilder` calls a `_fetchUserData()` function, which uses the `snapshot.data!.uid` to get the user's document from Firestore. Once the data (name, email) is fetched, the `GameHomeScreen` is displayed and the data is passed to the `ProfilePage`.
7. **Profile Page Update:** The `ProfilePage` was updated to receive the `AppUser` object. It now displays the real name and email from Firestore. The "Logout" button was made functional by calling `await FirebaseAuth.instance.signOut()`.

5. Output (Screenshots)

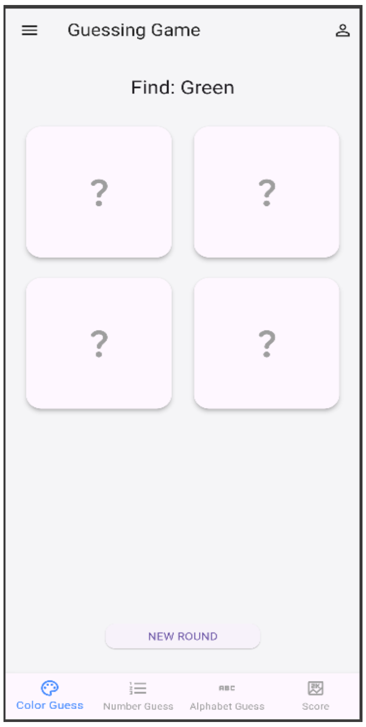
- **Screenshot 1: Modern Login/Signup Screen** (App screenshot showing the final gradient-based login page with fields for Email and Password, and a toggle to switch to the Signup form which includes a "Name" field).
- **Screenshot 2: Main Game Screen (Color Guess)** (App screenshot showing the minimalist game UI. Four `GameCard` widgets are displayed with their colors, and text at the top reads "Remember the colors and press MIX").

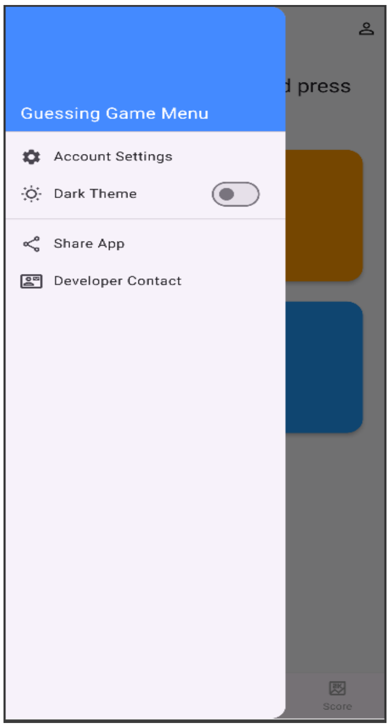


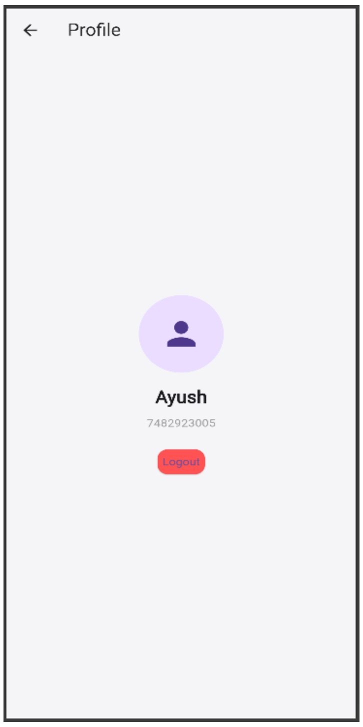




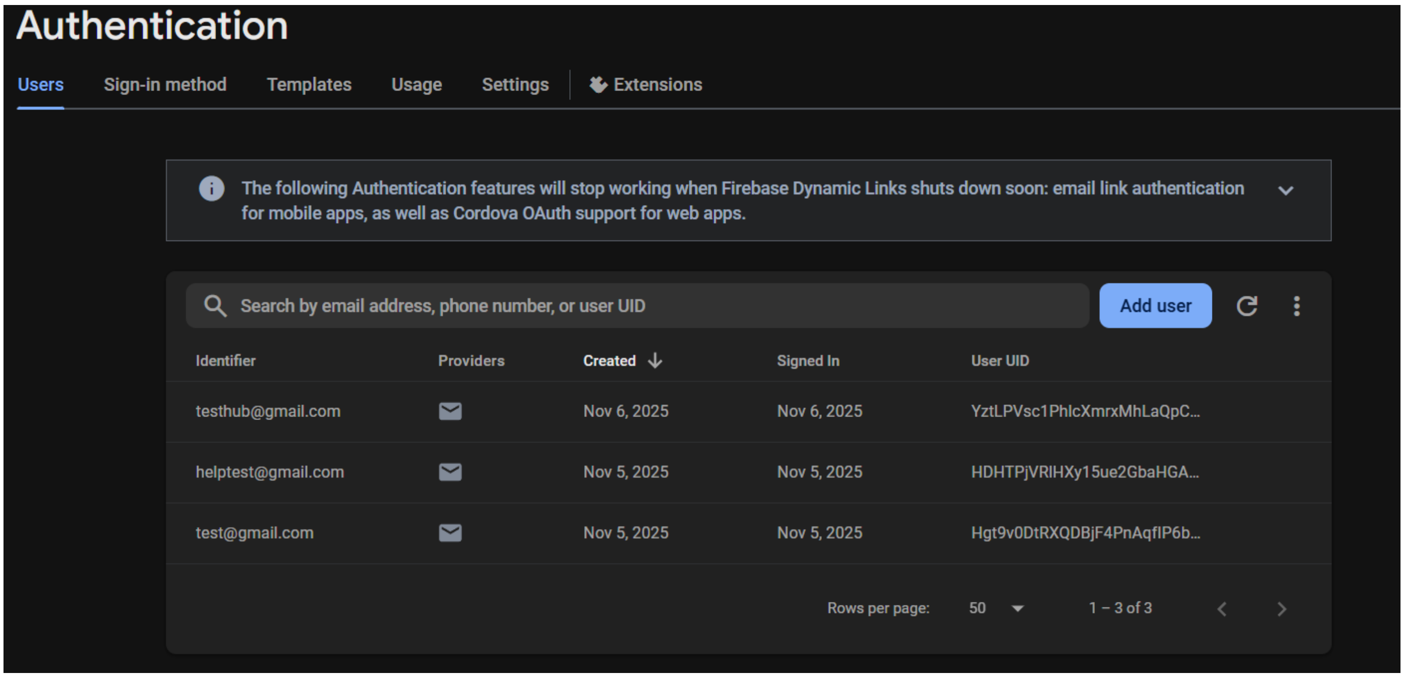
- **Screenshot 3: Game in Progress (Mixed)** (App screenshot showing the four `GameCard` widgets as hidden (grey with a `?` icon). The text now reads "Find: Red" or a similar prompt).
- **Screenshot 4: Navigation Drawer & Profile Page** (A composite image. One part shows the open `AppDrawer` with the "Dark Theme" toggle. The other part shows the `ProfilePage` displaying "Ayush Thakur" and "ayushwork981@gmail.com", with a "Logout" button below. This demonstrates data being fetched from Firestore).







- **Screenshot 5: Firebase Console (Authentication)** (A web browser screenshot of the Firebase Authentication console, showing a list of registered users with their emails and User UIDs).
- **Screenshot 6: Firebase Console (Cloud Firestore)** (A web browser screenshot of the Cloud Firestore database, showing the `users` collection. A document ID matching a User UID is open, revealing the `name: "Ayush Thakur"` and `email: "..."` fields).



The following Authentication features will stop working when Firebase Dynamic Links shuts down soon: email link authentication for mobile apps, as well as Cordova OAuth support for web apps.

Identifier	Providers	Created	Signed In	User UID
testhub@gmail.com	✉	Nov 6, 2025	Nov 6, 2025	YztLPVsc1PhlcXmrxMhLaQpC...
helptest@gmail.com	✉	Nov 5, 2025	Nov 5, 2025	HDHTPjVRIHXy15ue2GbaHGA...
test@gmail.com	✉	Nov 5, 2025	Nov 5, 2025	Hgt9v0DtRXQDBjF4PnAqfIP6b...

Github Link Of this project – https://github.com/aayushthakur001/MindMix

6. Conclusion

This project successfully achieved its goal of developing a full-stack, cross-platform mobile application. Starting from a simple UI sketch, a complete and interactive game, "MindMix," was built using Flutter. The application's functionality was significantly enhanced by integrating Firebase, transforming it from a local-only app to a cloud-connected service.

The implementation of Firebase Authentication provides a secure and standard way for user management, while Cloud Firestore offers a scalable and real-time database for storing user profiles. The use of `StreamBuilder` for auth state management proved to be an efficient way to handle user sessions.

The project demonstrates a comprehensive understanding of the mobile development lifecycle, from UI design and frontend logic to backend integration and data management. The final product is a scalable, modern, and engaging mobile game that is ready for further expansion.

7. Learning Outcomes

Through the course of this project, the following practical skills and concepts were learned:

1. **Full-Stack Development:** Gained practical experience in building a full-stack application, managing both the client-side (Flutter) and server-side (Firebase).
2. **Flutter UI & State Management:** Mastered the translation of UI designs into Flutter widgets and managed application state (game logic, theme) using `setState`.
3. **Firebase Authentication:** Learned to implement a complete user authentication system (register, login, logout, and session management) using `firebase_auth`.
4. **Cloud Firestore (NoSQL):** Gained proficiency in performing Create and Read (CR) operations on a NoSQL database, including structuring data in collections and documents.
5. **Firebase Integration:** Understood how to configure a Flutter app with Firebase using `flutterfire_cli` and manage multiple Firebase services.
6. **Real-time State Listening:** Implemented real-time listeners (`StreamBuilder` on `authStateChanges`) to create a reactive app that responds instantly to auth state.
7. **Iterative Development:** Practiced an iterative development process, starting with a simple prototype and progressively adding features and refactoring the UI.

8. References

1. **Flutter Official Documentation:** https://flutter.dev/docs
2. **Firebase for Flutter Documentation (FlutterFire):** https://firebase.flutter.dev/
3. **Firebase Authentication Documentation:** https://firebase.google.com/docs/auth
4. **Cloud Firestore Documentation:** https://firebase.google.com/docs/firestore
5. **Dart Language Documentation:** https://dart.dev/guides
6. **Stack Overflow:** For various implementation-specific troubleshooting.