Projektbeschreibung

Die **Schach-App** in Flutter ermöglicht es Nutzern, Schach zu spielen – sowohl im lokalen Modus als auch gegen den Computer. Außerdem bietet die App die Möglichkeit, online über Firebase mit anderen Spielern in Echtzeit zu spielen. Diese App wurde entwickelt, um den Schachspielern ein flexibles, benutzerfreundliches und interaktives Erlebnis zu bieten, unabhängig davon, ob sie alleine spielen möchten oder sich mit Freunden und anderen Spielern weltweit messen wollen.

Hauptfunktionen:

1. **Lokales Spiel:** Spieler können direkt gegeneinander spielen, indem sie einfach die gleiche App auf einem Gerät nutzen, ohne eine Internetverbindung zu benötigen.
2. **Spiel gegen den Computer:** Die App enthält einen Schach-Computergegner, gegen den der Nutzer auf verschiedenen Schwierigkeitsgraden spielen kann.
3. **Online-Multiplayer:** Über Firebase können Spieler gegeneinander antreten, unabhängig davon, wo sie sich befinden. Firebase ermöglicht die Speicherung von Spielständen und Echtzeit-Updates, damit beide Spieler immer synchron bleiben.

Die App nutzt die Vorteile von **Flutter** für die plattformübergreifende Entwicklung und ermöglicht eine nahtlose Nutzung auf iOS, Android und Web. Firebase wird als Backend verwendet, um Nutzerdaten, Spielstände und Echtzeit-Spiel-Interaktionen zu verwalten.

Ziel des Projekts

Ziel des Projekts ist es, eine moderne, plattformübergreifende Schachanwendung zu entwickeln, die sowohl Einsteigern als auch erfahrenen Spielern ein ansprechendes und intuitives Spielerlebnis bietet. Die App soll die verschiedenen Spielmodi – lokal, gegen den Computer sowie online über das Internet – in einer einzigen, benutzerfreundlichen Oberfläche vereinen.

Im Fokus steht dabei:

- **Nutzerfreundlichkeit:** Eine einfache und übersichtliche Benutzeroberfläche soll den Zugang zum Spiel erleichtern.
- **Vielseitigkeit:** Die App soll flexibel einsetzbar sein – ob für ein schnelles Spiel zu zweit auf einem Gerät, für herausfordernde Partien gegen eine KI oder für Online-Spiele mit Freunden.
- **Technologische Umsetzung:** Durch den Einsatz von Flutter und Firebase sollen moderne Technologien genutzt werden, um eine stabile, performante und skalierbare Anwendung zu schaffen.

Langfristig soll das Projekt eine Grundlage für weitere Funktionen wie Benutzerprofile, Ranglisten, Spielanalyse und Trainingsmodi bieten.

User Storys

Spiel gegen Computer starten	
ID	US001
Priorität	Must-have
Beschreibung	Als Nutzer möchte ich gegen einen Computergegner spielen können, um jederzeit alleine Schach spielen zu können.
Ergänzungen	Schwierigkeitsgrade, Timer optional
Zulieferung	Schach-Engine (z. B. Stockfish), ggf. grafische Icons
Abhängigkeiten	Schachlogik muss vollständig funktionieren
Definition of Done / Akzeptanzkriterium	Nutzer kann Spiel gegen Computer starten, Spiel läuft stabil durch
Entwickleraufgaben	Integration Schach-Engine, Spiellogik, GUI-Anpassung
Offene Fragen	Welche Engine? Soll Spiel speicherbar sein?
Wireframes	Board mit KI-Markierungen

Lokales Spiel zu zweit	
ID	US002
Priorität	Must-have
Beschreibung	Als Nutzer möchte ich lokal zu zweit Schach spielen können, um mit Freunden an einem Gerät zu spielen.
Ergänzungen	Wechsel der Spielrichtung, Timer optional
Zulieferung	keine
Abhängigkeiten	Schachlogik muss vorhanden sein
Definition of Done / Akzeptanzkriterium	Zwei Spieler können abwechselnd Züge machen
Entwickleraufgaben	Lokalen Spielmodus umsetzen, Spielsteuerung zweier Spieler
Offene Fragen	Rückgängig machen erlaubt?
Wireframes	Startmenü mit Auswahl „Lokal spielen“ → Standard-Schachbrett

Nutzer-Login per Firebase	
ID	US003
Priorität	Must-have
Beschreibung	Als Nutzer möchte ich mich online einloggen können, um meine Partien und Statistiken zu speichern.
Ergänzungen	Google/Firebase Auth, eventuell Gastmodus
Zulieferung	Firebase-Zugangsdaten, UI-Design
Abhängigkeiten	Firebase muss konfiguriert sein
Definition of Done / Akzeptanzkriterium	Login-Flow funktioniert, Auth-Status bleibt erhalten
Entwickleraufgaben	Firebase Auth einbinden, Login UI bauen, Fehlerbehandlung
Offene Fragen	Nur E-Mail/Passwort oder auch Google Sign-In?
Wireframes	Login-Screen mit E-Mail und Passwort, Google-Login-Button optional

Online-Spiel gegen andere Nutzer	
ID	US004
Priorität	Must-have
Beschreibung	Als Nutzer möchte ich online gegen andere Spieler in Echtzeit spielen können, um weltweit mit Menschen zu spielen.
Ergänzungen	Spielraum erstellen/beitreten, ELO-System optional
Zulieferung	Firebase Realtime oder Firestore Datenbank-Zugriff
Abhängigkeiten	Login muss vorher funktionieren, Matchmaking ggf.
Definition of Done / Akzeptanzkriterium	Zwei eingeloggte Nutzer können miteinander spielen
Entwickleraufgaben	Spiel-Session synchronisieren, Zugübertragung, Verbindungslogik
Offene Fragen	Matchmaking automatisch oder mit Einladung? Zeitkontrolle?
Wireframes	Liste offener Spiele, Button „Neues Spiel erstellen“

Chat während Online-Partie	
ID	US005
Priorität	Nice-to-have
Beschreibung	Als Spieler möchte ich während einer Online-Partie mit meinem Gegner chatten können, um zu kommunizieren.
Ergänzungen	Emojis, Nachricht löschen optional
Zulieferung	Chat-Komponenten, Firebase Firestore Zugriff
Abhängigkeiten	Online-Spiel muss funktionieren, Nutzer müssen eingeloggt sein
Definition of Done / Akzeptanzkriterium	Nachrichten erscheinen bei beiden Nutzern in Echtzeit
Entwickleraufgaben	Firestore-Integration, UI für Chatfeld, Scroll-Handling
Offene Fragen	Begrenzung der Nachrichtenlänge? Moderation notwendig?
Wireframes	Chatbereich unterhalb des Boards oder als Overlay

Tools

Für die Umsetzung der Schach-App kamen folgende Technologien, Frameworks und Pakete zum Einsatz:

Programmiersprachen & Frameworks

- **Flutter** – UI-Toolkit für plattformübergreifende App-Entwicklung (Android, iOS, Web)
- **Dart** – Programmiersprache zur Entwicklung mit Flutter

Backend & Datenhaltung

- **Firebase**:
    - firebase_core – zur Initialisierung der Firebase-Dienste
    - firebase_auth – für Benutzer-Authentifizierung (Login)
    - cloud_firestore – für die Speicherung und Synchronisation von Online-Spielständen und Chats

Schachlogik & Spielfläche

- squares – zur Darstellung des Schachbretts und der Figuren
- bishop – leistungsfähige Schach-Engine für Spielregeln, KI & Validierung
- square_bishop – verbindet squares mit bishop zur Spiellogik-Integration

State-Management

- flutter_bloc – zur Strukturierung der Geschäftslogik in Blöcken (BLoC-Pattern)
- equatable – für einfache Vergleichbarkeit von Objekten im BLoC-Kontext

Lokale Speicherung

- shared_preferences – zur Speicherung von Einstellungen und lokalen Daten auf dem Gerät

Icons & UI

- cupertino_icons – standardisierte iOS-Icons zur UI-Gestaltung

Gesamtarchitektur der Schach-App

Die Architektur der Schach-App folgt einem **modularen, klar getrennten Aufbau**, der auf modernen Prinzipien wie **State-Management mit BLoC**, **Firebase-Integration**, sowie einer sauberen **Trennung von UI und Logik** basiert.

1. Benutzeroberfläche (Frontend / UI)

Die UI ist vollständig in **Flutter** mit **Dart** umgesetzt und besteht aus verschiedenen Screens und Widgets:

- **Startbildschirm**: Auswahl des Spielmodus (lokal, gegen Computer, online)
- **Schachbrett-Ansicht**: Visualisierung des Spielfelds mit squares-Paket
- **Login- / Registrierseite**: Authentifizierung über Firebase
- **Chatbereich**: Anzeige und Eingabe von Nachrichten während Online-Spielen

Die UI verwendet **flutter_bloc** zur Steuerung und reagiert auf Zustandsänderungen aus der Logikschicht.

2. State-Management (Business-Logik)

Für die Steuerung der App-Logik wird das **BLoC-Pattern (Business Logic Component)** eingesetzt:

- **flutter_bloc** steuert Zustände für Spielverlauf, Authentifizierung und Online-Synchronisation.
- Zustandsklassen (State) und Event-Klassen (Event) strukturieren die Reaktionen auf Nutzereingaben und externe Änderungen (z. B. Firebase-Updates).

3. Schachlogik & Spielverlauf

Die Spiellogik basiert auf drei eng verknüpften Paketen:

- **bishop**: Berechnet gültige Züge, erkennt Matt/Pat, bewertet Spielstellungen (für KI)
- **squares**: Zeichnet das Schachbrett und verarbeitet Nutzereingaben (Drag & Drop)
- **square_bishop**: Brücke zwischen visueller Darstellung (squares) und der Logik (bishop)

Für Spiele gegen den Computer wird die **KI-Logik von bishop** verwendet.

🔹 **4. Datenhaltung & Online-Funktionalität (Firebase)**

Die App nutzt **Firebase** als Backend für Authentifizierung, Datenhaltung und Echtzeit-Kommunikation:

- **firebase_auth**: Benutzerregistrierung, Login, Sitzungserhaltung
- **cloud_firestore**: Speicherung von Online-Spielständen, Zügen, Matchmaking-Informationen und Chatnachrichten
- **firebase_core**: Initialisierung des Firebase-Stacks

Testbeschreibung: GameStorage.validateGameData()

📄 **Testziel**

Ziel dieses Tests ist es, die **Datenvalidierung einer gespeicherten Schachpartie** zu überprüfen. Die Methode validateGameData() stellt sicher, dass ein gespeichertes Spiel im erwarteten Format vorliegt, bevor es geladen wird.

🧪 **Testgegenstand**

Methode:

static bool validateGameData(Map<String, dynamic>? data)

Datei: game_storage.dart

Klasse: GameStorage

Testinhalte & Testfälle

Testfall	**Beschreibung**	**Erwartetes Ergebnis**
1. **Gültige Spieldaten**	Ein vollständiges Spielobjekt mit gültigem FEN, Modus (computer) und Spieler am Zug (isWhiteTurn) wird übergeben.	True
2. **Fehlender FEN-String**	Das Objekt enthält kein fen-Feld.	False
3. **Null als Eingabe**	Die Methode wird mit null aufgerufen.	False
4. **Ungültiger Modus**	Der Modus ist ein unbekannter String (invalid_mode).	False

Ergebnis

Die Methode validateGameData() besteht alle Tests wie erwartet und trägt somit zur **Datensicherheit und Stabilität beim Laden gespeicherter Partien** bei. Fehlerhafte oder unvollständige Spieldaten werden zuverlässig abgewiesen.

Ausblick

In zukünftigen Erweiterungen der Schach-App sind weitere Funktionen geplant. Dazu gehören unter anderem:

- **ELO-System:** Eine Bewertung der Spielstärke jedes registrierten Spielers anhand eines dynamischen ELO-Ratings. Dieses System soll faire Matchmaking-Prozesse unterstützen und die Motivation durch Ranglisten erhöhen.
- **Timer-Funktion:** Einführung von Zeitkontrollen (z. B. Blitz, Schnellschach, klassisch) für Online- und Offline-Spiele. Dadurch werden die Partien spannender und zeitlich reguliert, was insbesondere für kompetitives Spielen wichtig ist.
- **Schachpuzzles:** Integration eines Trainingsbereichs mit täglichen Schachaufgaben (z. B. Matt in 2), um das taktische Verständnis der Nutzer zu fördern. Die Puzzles sollen aus einer externen Datenbank stammen oder von Trainern erstellt werden können.