UNIVERSITY OF BUEA

FACULTY OF ENGINEERING AND TECHNOLOGY

Department of Computer Engineering

CEF 440:

INTERNET PROGRAMMING AND MOBILE PROGRAMMING

Task 6 Report — Database design and Implementation

Project Title: Design and Implementation of a Road Sign and Road State Mobile Notification Application (SafePath)

GROUP 1 MEMBERS	MATRICULE
ARREY-TABOT PASCALINE	FE22A151
TANUI NORBERT TANGIE	FE22A306
OROCKTAKANG MANYI	FE22A293
NGATTA GEORGE TABOT	FE22A259
FOMECHE ABONGACHU SIDNEY	FE22A218

1. Core Data Elements & Attributes

Data Elements

The SafePath application requires the following core data elements to support its functionality:

- **User**: Name, email, location (optional), preferred settings (e.g., notification preferences).
- **RoadSign**: Type, icon, meaning, category (e.g., warning, regulatory).
- **TrafficAlert**: Location, description, severity (high, moderate, low), timestamp.
- **Report**: User-generated reports including type (accident, hazard), location, description, timestamp.
- **PetrolStation**: Location, name, distance from user (optional).

Entity	Primary Key	Attributes	Special Notes
User	userId	name, email, role, locationPrefs, createdAt	Central account entity
Alert	alertId	type, severity, timestamp, location, description	Real-time safety notifications
RoadSign	signId	title, iconURL, category, meaning, addedAt	Traffic sign repository
Report	reportId	userId (FK), photoURL, location, status, alertId (FK, nullable), signId (FK, nullable)	Incident documentation
Feedback	feedbackId	userId (FK), message, rating, createdAt	User experience records

2. Conceptual Design

Key Relationships

1. **User → Report** (1-to-many)
    a. A user submits multiple reports
    b. *Constraint*: Reports require a user (NOT NULL FK)
2. **User → Feedback** (1-to-many)
    a. A user provides multiple feedback entries
    b. *Constraint*: Feedback persists if user is deleted (nullable FK)
3. **Alert → Report** (0-to-many)
    a. Alerts may trigger multiple reports
    b. *Optional*: Reports can exist without alerts
4. **RoadSign → Report** (0-to-many)
    a. Signs may be referenced in multiple reports
    b. *Optional*: Reports can be sign-agnostic

Design Principles

- **Minimalist Approach**: Only essential attributes retained
- **Flexible Reporting**: Optional alert/sign associations
- **Data Persistence**: Feedback survives user deletion

3. ER Diagram Explanation

Visual Components

• **Rectangles**: Represent entities (User, Alert, Report, RoadSign, Feedback)

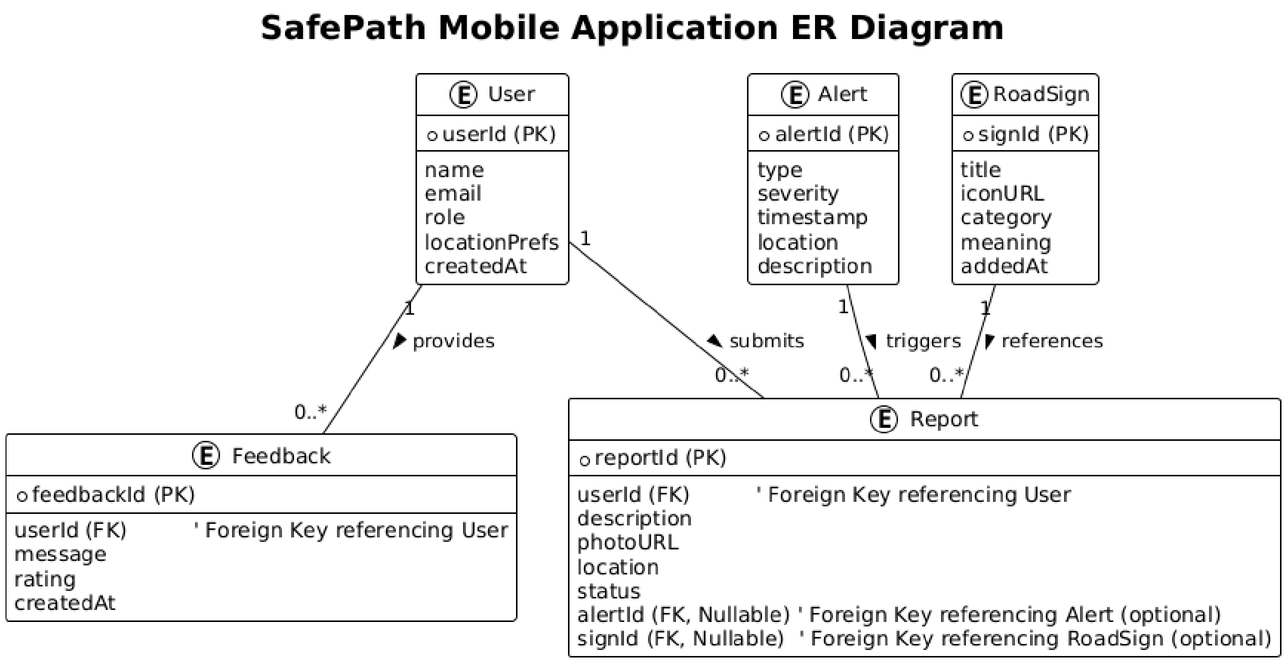
• **Diamonds**: Relationship types (Submit, Provide, Trigger, Reference)

• **Cardinality Notation**:

○ 1 (single) ○ * (many) ○
`0..*` (optional many)

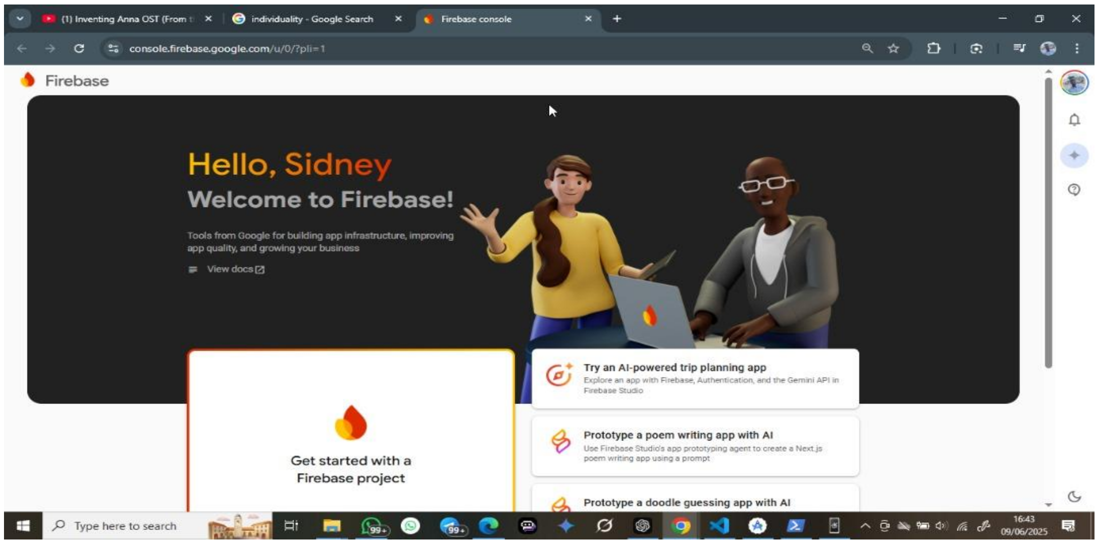
Key Diagram Features

- **User Centrality**: All relationships originate from User
- **Optional Links**: Dashed lines to Alert/RoadSign in Reports
- **Time Tracking**: createdAt fields in all user-generated content



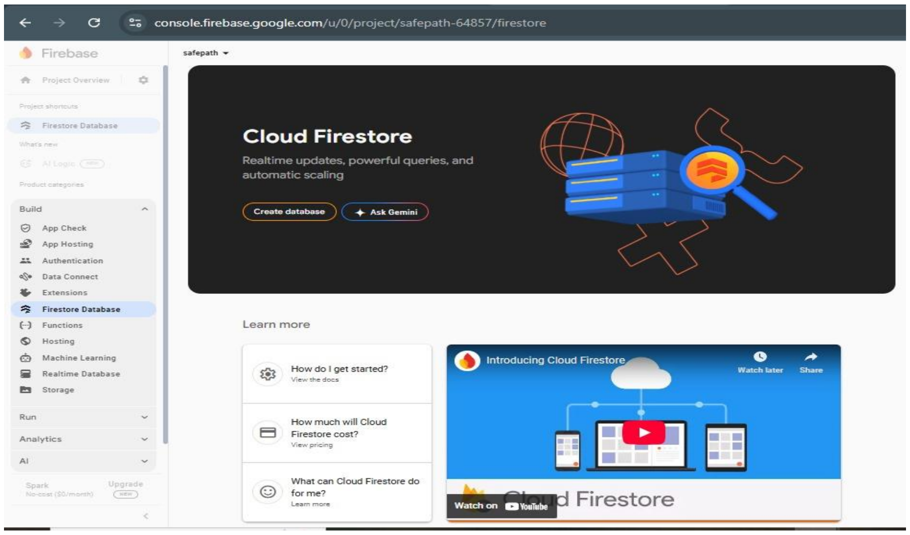
4. Database Implementation

Database Implementation (in Firebase)


Firebase
Hello, Sidney
Welcome to Firebase!
Tools from Google for building app infrastructure, improving app quality, and growing your business
View docs
Get started with a Firebase project
Try an AI-powered trip planning app
Prototype a poem writing app with AI
Prototype a doodle guessing app with AI

The database was set up in the Firebase Console as follows:

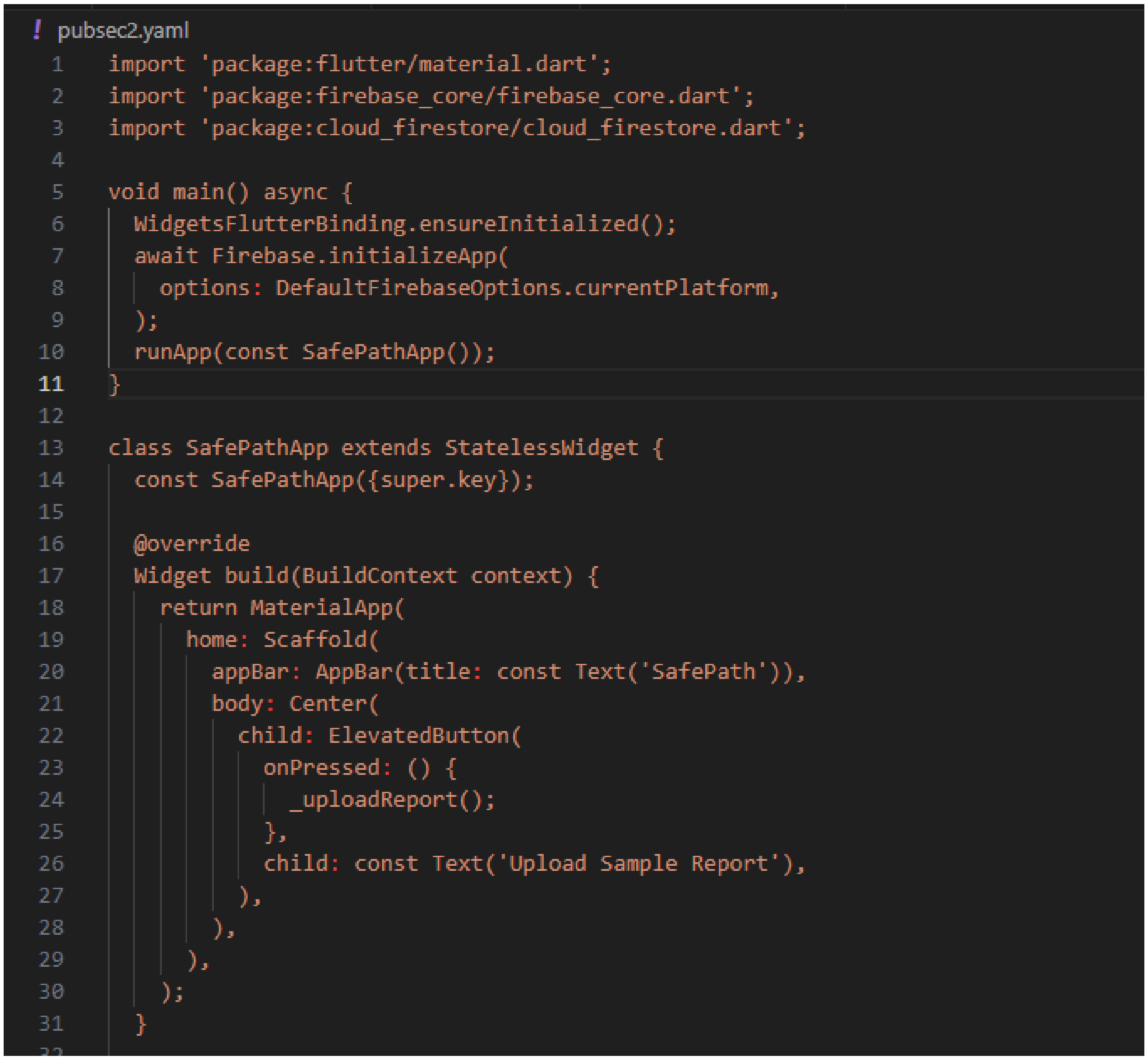
1. Created a new Firebase project named SafePath.
2. Initialized Cloud Firestore with security rules set to allow authenticated writes.
3. Created the following collections with sample documents:
   - users: { "email": "user@gmail.com", "name": "John Doe" }
   - alerts: { "type": "Accident", "location": "Buea Checkpoint", "severity": "High", "timestamp": "2025-06-09T18:00:00Z" }
   - reports: { "type": "Hazard", "location": "Main Street", "description": "Pothole reported" }
   - road_signs: { "type": "Stop", "meaning": "Mandatory stop", "category": "Regulatory" }



5. Backend Implementation

Backend Implementation (Flutter + Firebase)

The backend leverages Flutter with Firebase integration for real-time data management. The following dependencies were added to pubspec.yaml



pubsec2.yaml

```dart
import 'package:flutter/material.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:cloud_firestore/cloud_firestore.dart';

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

class SafePathApp extends StatelessWidget {
  const SafePathApp({super.key});

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text('SafePath')),
        body: Center(
          child: ElevatedButton(
            onPressed: () {
              _uploadReport();
            },
            child: const Text('Upload Sample Report'),
          ),
        ),
      ),
    );
  }
```

6. Connecting the Database to the Backend

Connecting Database to Backend

Firebase Configuration

- The Firebase project was configured using the FlutterFire CLI:
    1. Ran dart pub global activate flutterfire_cli and flutterfire configure to generate firebase_options.dart.
    2. Integrated the generated options into main.dart.

Conclusion

The database design and implementation for SafePath using Firebase Cloud Firestore provide a robust foundation for real-time data management. The integration with Flutter ensures seamless connectivity, enabling features like report submission and alert delivery. The next steps include testing with real user data and adding authentication.