SmartBudget — Full Project Folder (Ready for GitHub)

This document contains the complete project folder layout and the full contents of each file required to create the SmartBudget repository. The Android package name used throughout is **com.smartbudget**.

File: README.md

```
# SmartBudget

Student project: Personal budgeting app (Android + Node backend).

## Features implemented
- User registration & login (passwords hashed on server)
- Settings screen (theme, currency)
- Add expenses (local Room DB)
- Offline mode (Room)
- REST API (Node/Express) with MongoDB Atlas
- Unit tests & GitHub Actions CI

## Running the backend locally
1. `cd backend`
2. `npm install`
3. Copy `.env.sample` to `.env` and set `MONGO_URI` and `JWT_SECRET`
4. `npm run dev`

## Running the Android app
1. Open the `android-app` folder with Android Studio
2. Replace `ServiceBuilder.BASE_URL` in `ServiceBuilder.kt` with your backend URL
3. Build & run on a device (minSdk 24)

## CI
A GitHub Action workflow is included at `.github/workflows/android-ci.yml`. It builds the app and runs unit tes

## Demo video
Add a hosted video link here showing:
- Registration & login (show DB in MongoDB Atlas)
- Change settings
- Add expense (show saved locally and synced if online)
- Tests & GitHub Actions status
```

File: ai-use.txt

```
AI Tools Used (example):

During development I used ChatGPT for:
1. Generating boilerplate code for a Node/Express backend with authentication (bcrypt + JWT).
2. Producing Kotlin Retrofit and Room code snippets for the Android client.
3. Creating a README, demo script, and GitHub Actions workflow.

All AI-generated snippets were reviewed, adapted and tested locally. Secrets (MONGO_URI, JWT_SECRET) are stored
```

File: demo-script.md

```
Checklist to record:
- Show backend deployed and /api/auth reachable (Postman/browser).
- Show MongoDB Atlas and a user document after registration.
- On phone: register -> login.
- Open settings -> change theme/currency -> show stored change.
- Add expense -> show local DB updated.
- Demonstrate offline add (airplane mode) -> saved locally.
- Show GitHub repo commits and GitHub Actions run status.

Voiceover: Introduce yourself, explain each step, mention where data is stored (MongoDB Atlas), and point out t
```

File: diagrams/README.txt

```
Diagrams are included in the diagrams/ folder. If images are available, they are attached at the end of this PD
```

File: .github/workflows/android-ci.yml

```
name: Android CI

on:
  push:
    branches: [ main, master ]
  pull_request:
    branches: [ main, master ]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Set up JDK 11
        uses: actions/setup-java@v4
        with:
          java-version: '11'
          distribution: 'temurin'

      - name: Cache Gradle
        uses: actions/cache@v4
        with:
          path: |
            ~/.gradle/caches
            ~/.gradle/wrapper
          key: ${{ runner.os }}-gradle-${{ hashFiles('**/*.gradle*', '**/gradle-wrapper.properties') }}

      - name: Build with Gradle
        run: ./gradlew -p android-app assembleDebug --no-daemon

      - name: Run unit tests
        run: ./gradlew -p android-app testDebugUnitTest --no-daemon
```

File: backend/package.json

```
{
  "name": "smartbudget-api",
  "version": "1.0.0",
  "main": "index.js",
  "license": "MIT",
  "scripts": {
    "start": "node index.js",
    "dev": "nodemon index.js"
  },
  "dependencies": {
    "bcrypt": "^5.1.0",
    "body-parser": "^1.20.0",
    "cors": "^2.8.5",
    "dotenv": "^16.0.0",
    "express": "^4.18.2",
    "jsonwebtoken": "^9.0.0",
    "mongoose": "^6.7.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.20"
  }
}
```

File: backend/.env.sample

```
PORT=5000
MONGO_URI=your_mongo_atlas_connection_string
JWT_SECRET=ChangeThisSecret
```

File: backend/index.js

```
require('dotenv').config();
const express = require('express');
const mongoose = require('mongoose');
const bodyParser = require('body-parser');
const cors = require('cors');
const authRoutes = require('./routes/auth');

const app = express();
app.use(cors());
app.use(bodyParser.json());

const mongoUri = process.env.MONGO_URI;
mongoose.connect(mongoUri, { useNewUrlParser: true, useUnifiedTopology: true })
  .then(()=> console.log('MongoDB connected'))
  .catch(err=> console.error('MongoDB connection error:', err));

app.use('/api/auth', authRoutes);

app.get('/', (req, res) => res.json({ ok: true, message: 'SmartBudget API' }));

const PORT = process.env.PORT || 5000;
app.listen(PORT, ()=> console.log(`Server running on ${PORT}`));
```

File: backend/models/User.js

```
const mongoose = require('mongoose');

const UserSchema = new mongoose.Schema({
  name: { type: String },
  email: { type: String, unique: true, required: true },
  passwordHash: { type: String, required: true },
  settings: {
    theme: { type: String, default: "light" },
    currency: { type: String, default: "ZAR" }
  }
}, { timestamps: true });

module.exports = mongoose.model('User', UserSchema);
```

File: backend/routes/auth.js

```
const express = require('express');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const User = require('../models/User');

const router = express.Router();
const JWT_SECRET = process.env.JWT_SECRET || 'supersecretkey';

// Register
router.post('/register', async (req, res) => {
  try {
    const { name, email, password } = req.body;
    const existing = await User.findOne({ email });
    if (existing) return res.status(400).json({ message: 'Email in use' });

    const saltRounds = 10;
    const passwordHash = await bcrypt.hash(password, saltRounds);

    const user = new User({ name, email, passwordHash });
    await user.save();

    const token = jwt.sign({ userId: user._id }, JWT_SECRET, { expiresIn: '7d' });
    res.json({ token, user: { id: user._id, name: user.name, email: user.email } });
  } catch (err) {
    console.error(err);
    res.status(500).json({ message: 'Server error' });
  }
});

// Login
router.post('/login', async (req, res) => {
  try {
    const { email, password } = req.body;
    const user = await User.findOne({ email });
    if (!user) return res.status(401).json({ message: 'Invalid credentials' });

    const ok = await bcrypt.compare(password, user.passwordHash);
    if (!ok) return res.status(401).json({ message: 'Invalid credentials' });

    const token = jwt.sign({ userId: user._id }, JWT_SECRET, { expiresIn: '7d' });
    res.json({ token, user: { id: user._id, name: user.name, email: user.email } });
  } catch (err) {
    console.error(err);
    res.status(500).json({ message: 'Server error' });
  }
});

module.exports = router;
```

File: backend/README.md

````
# SmartBudget Backend

This is a Node/Express backend for the SmartBudget app.

## Setup (local)
1. Copy `.env.sample` to `.env` and fill `MONGO_URI` and `JWT_SECRET`.
2. Install dependencies:
   ```
   npm install
   ```
3. Run locally:
   ```
   npm run dev
   ```

## Deploy
Use a hosting service (Railway, Render, Heroku). Set environment variables on the host.
````

File: android-app/settings.gradle

```
include ':app'
rootProject.name = 'SmartBudget'
```

File: android-app/build.gradle

```
buildscript {
    ext.kotlin_version = "1.8.0"
    repositories { google(); mavenCentral() }
    dependencies {
        classpath "com.android.tools.build:gradle:8.1.1"
        classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:$kotlin_version"
    }
}
allprojects { repositories { google(); mavenCentral() } }
```

File: android-app/app/build.gradle

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
    id 'kotlin-kapt'
}

android {
    compileSdk 34
    defaultConfig {
        applicationId "com.smartbudget.app"
        minSdk 24
        targetSdk 34
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
    }
    buildTypes { release { minifyEnabled false } }
    compileOptions { sourceCompatibility JavaVersion.VERSION_11; targetCompatibility JavaVersion.VERSION_11 }
    kotlinOptions { jvmTarget = '11' }
}

dependencies {
    implementation "org.jetbrains.kotlin:kotlin-stdlib:1.8.0"
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-moshi:2.9.0'
    implementation 'com.squareup.okhttp3:logging-interceptor:4.9.3'
    implementation "androidx.room:room-runtime:2.5.2"
    kapt "androidx.room:room-compiler:2.5.2"
    implementation "androidx.room:room-ktx:2.5.2"
    implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.6.1"
    implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.6.1"
    implementation 'com.google.android.material:material:1.9.0'
    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
}
```

File: android-app/app/src/main/AndroidManifest.xml

```
<manifest package="com.smartbudget.app" xmlns:android="http://schemas.android.com/apk/res/android">
    <uses-permission android:name="android.permission.INTERNET"/>
    <application android:allowBackup="true" android:label="SmartBudget">
        <activity android:name=".ui.DashboardActivity"/>
        <activity android:name=".ui.LoginActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

File: android-app/app/src/main/java/com/smartbudget/app/ui/LoginActivity.kt

```
package com.smartbudget.app.ui

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.EditText
import android.widget.Toast
import com.smartbudget.app.auth.AuthRepository

class LoginActivity : AppCompatActivity() {
    private val repo = AuthRepository()

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_login)

        val etEmail = findViewById<EditText>(R.id.etEmail)
        val etPassword = findViewById<EditText>(R.id.etPassword)
        val btnLogin = findViewById<Button>(R.id.btnLogin)

        btnLogin.setOnClickListener {
            val email = etEmail.text.toString().trim()
            val pass = etPassword.text.toString().trim()
            if (email.isEmpty() || pass.isEmpty()) {
                Toast.makeText(this, "Please fill email & password", Toast.LENGTH_SHORT).show()
                return@setOnClickListener
            }
            repo.login(email, pass) { ok, token ->
                runOnUiThread {
                    if (ok) {
                        getSharedPreferences("smart", MODE_PRIVATE).edit().putString("TOKEN", token).apply()
                        startActivity(Intent(this, DashboardActivity::class.java))
                        finish()
                    } else {
                        Toast.makeText(this, "Login failed: $token", Toast.LENGTH_LONG).show()
                    }
                }
            }
        }
    }
}
```

File: android-app/app/src/main/java/com/smartbudget/app/ui/DashboardActivity.kt

```
package com.smartbudget.app.ui

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import com.smartbudget.app.R

class DashboardActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_dashboard)
    }
}
```

File: android-app/app/src/main/java/com/smartbudget/app/auth/AuthApi.kt

```
package com.smartbudget.app.auth

import retrofit2.Call
import retrofit2.http.Body
import retrofit2.http.POST

data class RegisterRequest(val name: String, val email: String, val password: String)
data class LoginRequest(val email: String, val password: String)
data class UserDto(val id: String, val name: String, val email: String)
data class AuthResponse(val token: String, val user: UserDto)

interface AuthApi {
    @POST("/api/auth/register")
    fun register(@Body req: RegisterRequest): Call<AuthResponse>

    @POST("/api/auth/login")
    fun login(@Body req: LoginRequest): Call<AuthResponse>
}
```

File: android-app/app/src/main/java/com/smartbudget/app/auth/ServiceBuilder.kt

```
package com.smartbudget.app.auth

import okhttp3.OkHttpClient
import okhttp3.logging.HttpLoggingInterceptor
import retrofit2.Retrofit
import retrofit2.converter.moshi.MoshiConverterFactory
import java.util.concurrent.TimeUnit

object ServiceBuilder {
    private const val BASE_URL = "https://YOUR_BACKEND_URL" // <-- replace

    private val logger = HttpLoggingInterceptor().apply { level = HttpLoggingInterceptor.Level.BODY }
    private val client = OkHttpClient.Builder()
        .addInterceptor(logger)
        .connectTimeout(30, TimeUnit.SECONDS)
        .readTimeout(30, TimeUnit.SECONDS)
        .build()

    val retrofit: Retrofit = Retrofit.Builder()
        .baseUrl(BASE_URL)
        .client(client)
        .addConverterFactory(MoshiConverterFactory.create())
        .build()

    val authApi: AuthApi = retrofit.create(AuthApi::class.java)
}
```

File: android-app/app/src/main/java/com/smartbudget/app/auth/AuthRepository.kt

```
package com.smartbudget.app.auth

import retrofit2.Call
import retrofit2.Callback
import retrofit2.Response

class AuthRepository {
    private val api = ServiceBuilder.authApi

    fun register(name: String, email: String, password: String, cb: (Boolean, String?) -> Unit) {
        val req = RegisterRequest(name, email, password)
        api.register(req).enqueue(object : Callback<AuthResponse> {
            override fun onResponse(call: Call<AuthResponse>, response: Response<AuthResponse>) {
                if (response.isSuccessful) { cb(true, response.body()?.token) }
                else { cb(false, response.errorBody()?.string()) }
            }
            override fun onFailure(call: Call<AuthResponse>, t: Throwable) { cb(false, t.message) }
        })
    }

    fun login(email: String, password: String, cb: (Boolean, String?) -> Unit) {
        val req = LoginRequest(email, password)
        api.login(req).enqueue(object : Callback<AuthResponse> {
            override fun onResponse(call: Call<AuthResponse>, response: Response<AuthResponse>) {
                if (response.isSuccessful) { cb(true, response.body()?.token) }
                else { cb(false, response.errorBody()?.string()) }
            }
            override fun onFailure(call: Call<AuthResponse>, t: Throwable) { cb(false, t.message) }
        })
    }
}
```

File: android-app/app/src/main/java/com/smartbudget/app/data/db/Expense.kt

```
package com.smartbudget.app.data.db

import androidx.room.Entity
import androidx.room.PrimaryKey
import java.util.*

@Entity
data class Expense(
    @PrimaryKey val id: String = UUID.randomUUID().toString(),
    val amount: Double,
    val category: String,
    val date: Long
)
```

File: android-app/app/src/main/java/com/smartbudget/app/data/db/ExpenseDao.kt

```
package com.smartbudget.app.data.db

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.Query

@Dao
interface ExpenseDao {
    @Insert
    suspend fun insert(expense: Expense)

    @Query("SELECT * FROM Expense ORDER BY date DESC")
    suspend fun getAll(): List<Expense>
}
```

File: android-app/app/src/main/java/com/smartbudget/app/data/db/AppDatabase.kt

```
package com.smartbudget.app.data.db

import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
import android.content.Context

@Database(entities = [Expense::class], version = 1)
abstract class AppDatabase : RoomDatabase() {
    abstract fun expenseDao(): ExpenseDao

    companion object {
        @Volatile private var instance: AppDatabase? = null
        fun get(context: Context): AppDatabase =
            instance ?: synchronized(this) {
                instance ?: Room.databaseBuilder(context.applicationContext,
                    AppDatabase::class.java, "smartbudget-db").build().also { instance = it }
            }
    }
}
```

File: android-app/app/src/main/res/layout/activity_login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:padding="24dp" android:layout_width="match_parent"
    android:layout_height="match_parent">

    <EditText android:id="@+id/etEmail" android:hint="Email" android:inputType="textEmailAddress"
        android:layout_width="match_parent" android:layout_height="wrap_content"/>
    <EditText android:id="@+id/etPassword" android:hint="Password" android:inputType="textPassword"
        android:layout_width="match_parent" android:layout_height="wrap_content" android:layout_marginTop="8dp"
    <Button android:id="@+id/btnLogin" android:text="Login" android:layout_width="match_parent"
        android:layout_height="wrap_content" android:layout_marginTop="16dp"/>
</LinearLayout>
```

File: android-app/app/src/main/res/layout/activity_dashboard.xml

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">
    <TextView android:text="Welcome to SmartBudget (Dashboard)" android:layout_gravity="center"
        android:layout_width="wrap_content" android:layout_height="wrap_content"/>
</FrameLayout>
```

File: PROJECT_NOTE.txt

```
This folder layout is ready to be copied into a GitHub repository.
Open Android Studio and choose 'Open' -> navigate to android-app folder to import the project.
Remember to:
- Set your backend URL at android-app/app/src/main/java/com/smartbudget/app/auth/ServiceBuilder.kt
- Create a MongoDB Atlas cluster and set MONGO_URI in backend/.env
- Deploy backend to Railway/Render/Heroku if you want it hosted
```