# 🧠 SmartExplain 🔍✨

## 📱 Project Summary

SmartExplain is a React Native app where users can log in with Google, take or upload a photo, and receive an AI-generated explanation of the image. It uses Firebase for authentication, storage, and database, and a Node.js backend to process the image with an external AI service.

**SmartExplain** is a React Native (Expo) app using the **MVVM architecture**, where users:

1. 🔐 Sign in via **Google (Firebase Auth)**
2. 📸 Take or upload a photo
3. ☁️ Upload the image to **Firebase Storage**
4. 🔗 Retrieve the downloadURL
5. 🧠 Send the URL to a **Node.js backend** that uses an **AI API**
6. 📖 Get and display an **encyclopedic explanation**
7. 🗃️ Save metadata in **Firebase Realtime Database**:

## 🔄 Information Flow

1. User logs in
2. Selects or takes a photo
3. Image is uploaded to Firebase Storage
4. downloadURL is retrieved
5. Metadata is saved in Firebase Realtime DB
6. URL is sent to backend via Axios
7. Backend uses AI to analyze it
8. Explanation is returned and displayed

## 🧩 MVVM Architecture Overview

| **Layer** | **Responsibility** |
| --- | --- |
| **Model** | Data models and Firebase/Axios logic |
| **ViewModel** | Coordinates login, upload, backend communication, and data saving |
| **View** | Screens and UI reacting to state |

## ✅ Covered Requirements

| **Requirement** | **Fulfilled by** |
| --- | --- |
| 📱 Mobile sensor | Camera/gallery via image-picker |
| 🔐 Authentication | Firebase Google Auth |
| ☁️ Cloud Storage | Firebase Storage |
| 🧠 AI Backend | Node.js backend + Google Vision API |
| 🗃️ Database | Firebase Realtime DB |
| 🔁 Client-server | Axios between frontend and backend |
| 📋 Agile process | 5 Sprints, Kanban board, demo video, GitHub |

## 📁 Backend Folder Structure

/routes

└── analyzeRoutes.js # POST /analyze-image

/controllers

└── analyzeController.js # Logic to process imageURL and return result

/services

└── aiService.js # Google Vision or AI API handler

.env # API keys and secrets

└── server.js # Express server

# 🧠 Project Structure

## 📁 Frontend Folder Structure

/frontend

/assets

└── icon.png, placeholder.jpg # UI images

/models

└── userModel.js # User data structure

└── imageModel.js # Image metadata structure

/viewmodels

└── authViewModel.js # Login/logout logic

└── imageViewModel.js # Upload, save metadata, backend request

/contexts

└── AuthContext.js # Global auth state

└── ImageContext.js # Current image and result

/views

└── LoginScreen.js # Login screen

└── HomeScreen.js # Image selector and preview

└── ExplanationScreen.js # Display AI result

/components

└── ImageSelector.js # Camera/Gallery picker

└── ExplanationCard.js # Result UI card

/services

└── axios.js # Axios setup

└── firebase.js # Firebase config (auth, db, storage)

App.js # Entry point + context providers

# **⚙️ Sprint 0 – Project Setup (Infraestructura base)**

**Goal:** Set up entire project structure + initial UI  
**Frontend tasks:**

* Create Express project
* Initialize React Native + Expo project
* Create folder structure
* Set up navigation and context providers
* Create visual skeletons for:
* LoginScreen.js
* HomeScreen.js
* ExplanationScreen.js
* ImageSelector.js
* ExplanationCard.js

**Backend tasks:**

* Create Express project
* Setup folder structure (routes, controllers, services)
* Create base route /analyze-image
* Add .env, install needed dependencies
* Confirm Express server runs

# **🟢 Sprint 1 – Google Login + Image Selection**

**Goal:** Allow user login and prepare image input

* Upload image to Firebase Storage
* Implement Google Auth (authViewModel.js)
* Store login in AuthContext.js
* Connect LoginScreen.js to ViewModel
* Implement ImageSelector.js to take or pick an image
* Preview image in HomeScreen.js

# **🟡 Sprint 1 – Upload + Metadata Save**

**Goal:** Save image and its metadata

* Upload image to Firebase Storage
* Retrieve downloadURL
* Save metadata to Realtime Database via imageViewModel.js
* Use imageModel.js for structure

# **🔵 Sprint 3 – AI Integration + Delivery**

**Goal:** Generate explanation from AI and polish the app

* Send imageURL to backend using Axios
* Backend (analyzeController.js) calls AI API via aiService.js
* Return explanation as JSON
* Display it using ExplanationCard.js in ExplanationScreen.js

# **🟣 Sprint 3 – Final Delivery**

**Goal:** Complete project and deliver

* Improve UI (styles, layout)
* Add loading states and error handling
* Record demo (≤10 min)
* Export Trello/Notion board screenshots
* Push GitHub repo
* Submit final delivery via LearnPoint