Technical Assessment: Mobile Application Developer (Flutter)

Objective

Design and develop a **Flutter-based e-Signature mobile application** that allows users to upload documents, place interactive fields (signature, text, checkbox, date), export/import field configurations as JSON, and generate a finalized signed PDF.

This assessment is designed to evaluate:

- Flutter development skills
- UI/UX implementation
- Drag & drop interactions
- PDF rendering and generation
- Data modeling and JSON handling
- Firebase authentication and integration
- Code quality and architectural decisions

UI & Functionality Reference (Mandatory)

To understand the expected **workflow, UX behavior, and baseline functionality**, candidates **must review** the following app:

Reference App:

https://play.google.com/store/apps/details?id=com.deedsign.mobile

Reference Usage Rules

- This app is **only for reference**

- Pixel-perfect UI replication is **NOT required**
- Focus on:
  - Overall flow
  - Field placement behavior
  - Signing experience
  - PDF export results

Tech Stack (Required)

- **Flutter (latest stable)**
- **Dart**
- **Firebase Authentication**
- **Firebase Firestore or Realtime Database**
- Any Flutter packages required for:
  - PDF rendering & generation
  - Drag & drop interaction
  - File upload & preview

Functional Requirements

1. Authentication (Firebase)

- Implement authentication using **Firebase Authentication**
- Email & password login/signup is sufficient
- Authentication state must persist between sessions
- Non-authenticated users must not access document features

---

2. Document Upload

- Allow users to upload:

    - **PDF**
    - **DOCX**
- Store files locally or in Firebase Storage
- After successful upload, redirect to **Document Editor Screen**
- Display document preview before editing

---

3. Document Editor – Field Placement

After upload, users must be able to prepare the document for signing.

Supported Fields

- Signature
- Textbox
- Checkbox
- Date field

Field Requirements

- Users can add multiple fields of any type
- Fields must support:
    - Drag & drop positioning
    - Placement anywhere on the document
- Each field must have:
    - Unique ID / name
    - Field type
    - X & Y position (relative to document)
    - Width & height

---

4. Export & Import Field Configuration (JSON)

- Users must be able to **export** field configurations as JSON

- JSON structure must include:
  - Field ID
  - Field type
  - Position
  - Size

Example JSON:

```
{
   "fields":
   [
      {
         "id": "signature_1",
         "type": "signature",
         "x": 120,
         "y": 340,
         "width": 180,
         "height": 60
      }
   ]
}
```

- Users must be able to **import** this JSON
- Imported JSON must restore:
  - All fields
  - Correct positions and sizes
- Invalid JSON should be handled gracefully

---

5. Publish Document

- User can **publish** the document once field setup is complete
- Once published:

    - Field positions become locked
    - Document enters **signing mode**
- Editing fields after publishing must not be allowed

---

6. Signing Mode – Fill Fields

- Users can enter values for all defined fields:
    - Draw or upload a signature
    - Enter text
    - Toggle checkboxes
    - Select date
- Required fields must be validated before submission

---

7. Final PDF Generation

- After completing all fields:
    - Generate a final **PDF**
- The PDF must contain:
    - Original document content
    - All filled field values rendered at correct positions
- User must be able to:
    - Preview the final PDF
    - Save it locally

Non-Functional Requirements

Code Quality

- Clean, readable, maintainable code
- Proper separation of UI, logic, and services

- Meaningful naming conventions

Architecture

- Any state management approach is acceptable (Provider, Riverpod, Bloc, etc.)
- App structure should be scalable and modular

Error Handling

- Handle:
  - Upload failures
  - Auth failures
  - Invalid JSON imports
- App must not crash under normal failure scenarios

UI/UX

- Simple, intuitive interface
- Clear user flow:
  - Login → Upload → Edit → Publish → Sign → Export PDF

Deliverables

- Git repository link (GitHub / GitLab / Bitbucket)
- `README.md` containing:
  - Setup instructions
  - Firebase configuration steps
  - App architecture overview
  - List of used Flutter packages
- APK file (optional but recommended)

Evaluation Criteria

Area	Weight
Flutter UI & UX	20%
Drag & Drop & Field Handling	20%
PDF Generation Accuracy	20%
Firebase Authentication	15%
JSON Import/Export	15%
Code Quality & Architecture	10%

Bonus (Optional)

- Multiple signers support
- Zoom & pan on document
- Field resizing
- Dark mode
- Unit or widget tests

Time Expectation

- **3–5 days**
- Focus on **correctness and functionality**
- UI polish is secondary