Senior Flutter Engineer

Technical Assessment

Loan Origination System (LOS) - MSME Lending Platform

⏱ Duration: 24 Hours | 💻 Expected Effort: 3 Hours | 📊 Total Points: 100

1. Introduction

Welcome to the Senior Flutter Engineer technical assessment. You will be building a mini Loan Origination System (LOS) mobile application for an NBFC specializing in MSME lending. This assignment evaluates your expertise in Flutter development, state management, clean architecture, animations, and code quality.

Important: This is a time-boxed assignment. Focus on demonstrating clean, production-ready code rather than building every possible feature. Quality over quantity.

2. What We're Looking For

- Clean Architecture implementation with proper separation of concerns
- BLoC pattern for state management (using flutter_bloc package)
- Polished UI with meaningful animations and smooth transitions
- Proper data fetching from provided remote JSON endpoints
- Local database for storing new applications and drafts
- Well-documented, readable, and maintainable code
- Lightweight application with optimized performance

3. Project Requirements

3.1 Application Overview

Build a "LoanEase" mobile application that allows loan officers to manage MSME loan applications. The app should have the following core modules:

Module 1: Authentication & Onboarding

1. **Splash Screen:** Animated splash with logo reveal animation (scale + fade)
2. **Login Screen:** Phone number + OTP based authentication (mock OTP - any 6 digits work)
3. **OTP Verification:** Auto-focus OTP fields with countdown timer animation
4. **Session Management:** Persist login state using secure local storage

Module 2: Dashboard

5. **Statistics Cards:** Animated counter cards showing loan statistics (Total, Approved, Pending, Rejected)
6. **Quick Actions:** Grid of action buttons with subtle press animations

7. **Recent Applications:** Horizontal scrollable list with card flip animation on tap for details preview
8. **Pull to Refresh:** Custom refresh indicator with company branding

Module 3: Loan Application Management

9. **Application List:** Paginated list with search, filter (by status, amount range), and sort functionality
10. **Application Card:** Show applicant name, business type, requested amount, status badge with color coding
11. **Swipe Actions:** Swipe left to reject, swipe right to approve (with confirmation dialog)
12. **List Animation:** Staggered list animation on load, smooth item removal animation

Module 4: New Loan Application (Multi-Step Form)

Implement a 4-step wizard with progress indicator and step transition animations:

- **Step 1 - Business Details:** Business name, type (dropdown), registration number, years in operation
- **Step 2 - Applicant Details:** Name, PAN, Aadhaar (masked input), mobile, email
- **Step 3 - Loan Requirements:** Amount (with slider + input), tenure, purpose (multi-select chips)
- **Step 4 - Review & Submit:** Summary of all entered data with edit option for each section
- **Form Validation:** Real-time validation with animated error messages
- **Draft Save:** Auto-save form progress to local database

Module 5: Application Details Screen

13. **Hero Animation:** Shared element transition from list card to detail header
14. **Collapsible Sections:** Expandable sections for Business Info, Applicant Info, Loan Details
15. **Timeline View:** Application status timeline with animated progress
16. **Action Buttons:** Approve/Reject with animated state change

4. Technical Requirements

4.1 Architecture & Project Structure

Follow Clean Architecture with the following layer structure:

Layer	Responsibilities
Presentation	UI Widgets, BLoCs, Pages, Animations
Domain	Entities, Use Cases, Repository Interfaces
Data	Repository Implementations, Remote & Local Data Sources, Models
Core	Constants, Themes, Utils, Extensions, DI Setup

4.2 State Management

- Use flutter_bloc package exclusively for state management
- Implement proper BLoC/Cubit for each feature module
- Use Equatable for state comparison
- Implement proper loading, success, and error states
- Use BlocObserver for debugging/logging

4.3 Data Layer

Remote Data (GET requests): Fetch data from provided JSON endpoints hosted on gist.

Local Data (POST/CREATE): Store new applications and status updates in local database.

- Use Dio or http package for API calls
- Implement proper error handling and retry logic
- Use repository pattern to abstract data sources
- Merge remote and local data for display (local takes priority for status updates)

4.4 Local Database

- Use Hive or SQLite (sqflite/drift) for local persistence
- Store newly created loan applications
- Store status updates (approve/reject) for existing applications
- Store user session and preferences
- Save draft applications during form filling

4.5 Animations Requirements

Mandatory animations to implement:

1. **Implicit Animations:** AnimatedContainer, AnimatedOpacity, AnimatedSwitcher for UI state changes
2. **Explicit Animations:** At least one custom AnimationController (e.g., splash screen logo)
3. **Hero Animations:** List to detail screen transition
4. **Page Transitions:** Custom page route transitions between screens
5. **Staggered Animations:** List items appearing with delay
6. **Micro-interactions:** Button press feedback, loading states, success/error animations

5. API Endpoints (JSON Data)

Use the following hosted JSON endpoints for GET requests. These are static JSON files - no POST operations supported on remote.

Available Endpoints

Dashboard stats

https://gist.githubusercontent.com/rishimalgwa/4d3d4d0e8e270f4ba8af64a3d4099e5c/raw/bd7d9bf5
0692d284500523ac97f46b93da97aa9f/gistfile1.txt

Loan applications

https://gist.githubusercontent.com/rishimalgwa/d8edc5edadb4e1e06cec67c8748c1939/raw/b266e383
cbb321b02554180639f8e2f8e52b865a/gistfile1.txt

Master data

https://gist.githubusercontent.com/rishimalgwa/5e0764ed7f61d315c7bef83ac8d48ad9/raw/21aef841
435efc1edf48d4479a7adc44de42b35f/gistfile1.txt

User profile

https://gist.githubusercontent.com/rishimalgwa/5b598c4b5744fd1aa0714d8216398e53/raw/3d4ef3eb
a42322599c4db30acfbfbd776f9e53d1/gistfile1.txt

Note: The exact URLs will be provided in the accompanying email. All JSON files are also attached with this document for reference.

Data Flow Architecture

7. **Initial Load:** Fetch loan applications from remote JSON endpoint
8. **New Application:** Save to local database only (with 'local_' prefix in ID)
9. **Status Update:** Store update in local database, merge with remote data on display
10. **Display Logic:** Show remote + local applications, local status updates override remote
11. **Refresh:** Re-fetch remote data, merge with local modifications

5.1 LoanApplication Model

Field	Type	Description
id	String	Unique identifier (UUID)
applicationNumber	String	Display number (e.g., "LOAN-2024-001")
status	Enum	pending | under_review | approved | rejected | disbursed
businessName	String	Name of the MSME business
businessType	Enum	sole_proprietorship | partnership | pvt_ltd | llp
applicantName	String	Full name of the applicant
requestedAmount	double	Loan amount requested (₹50,000 - ₹50,00,000)
tenure	int	Loan tenure in months (6-60)
purpose	List<String>	working_capital | equipment | expansion | inventory
createdAt / updatedAt	DateTime	Timestamps

See attached JSON files for complete model with all fields.

6. Evaluation Criteria & Scoring

Your submission will be evaluated based on the following criteria. **Passing threshold: 70 points out of 100.**

Criteria	Points	Evaluation Focus
CODE ARCHITECTURE (25 Points)		
Clean Architecture Implementation	**10**	Proper layer separation, dependency direction, folder structure
SOLID Principles	**8**	Single responsibility, interface segregation, DI usage
Code Organization & Modularity	**7**	Feature-wise organization, reusable components, barrel files
STATE MANAGEMENT (20 Points)		
BLoC Implementation	**10**	Proper events/states, no business logic in UI, state immutability
State Handling	**6**	Loading/success/error states, proper state emissions
BLoC Best Practices	**4**	BlocProvider scope, Equatable usage, BlocObserver
UI & ANIMATIONS (20 Points)		
Animation Quality	**10**	Smooth 60fps, meaningful animations, proper curves & durations
UI Design & UX	**6**	Clean UI, consistent design system, intuitive interactions
Responsive Design	**4**	Adapts to different screen sizes, no overflow issues
DATA LAYER & PERSISTENCE (15 Points)		
API Integration	**7**	Proper HTTP client setup, error handling, data parsing
Local Database	**8**	Proper schema, CRUD operations, data merge logic
CODE QUALITY (15 Points)		
Code Readability	**5**	Meaningful naming, proper comments, consistent formatting
Error Handling	**5**	Graceful error handling, user feedback, no crashes
Documentation	**5**	README, inline comments, code documentation
PERFORMANCE & OPTIMIZATION (5 Points)		
App Performance	**3**	No jank, efficient builds, proper const usage
Memory & Build Optimization	**2**	Proper dispose, lazy loading, minimal rebuilds
TOTAL	**100**	**Passing Score: 70+**

7. Bonus Points (Up to +15)

Bonus Feature	Points	Difficulty
Unit Tests (≥70% coverage on BLoCs)	**+5**	Medium
Widget Tests for critical flows	**+3**	Medium
Dark Mode support with animated theme switching	**+2**	Easy
Biometric Authentication (fingerprint/face)	**+2**	Easy
Export loan application to PDF	**+2**	Medium
Localization (English + Hindi)	**+1**	Easy

8. AI/LLM Usage Policy

Use of AI tools (ChatGPT, GitHub Copilot, Claude, etc.) is permitted with conditions:

12. **Mandatory Disclosure:** Create an AI_USAGE.md file in the root directory
13. Document which parts of the code were AI-assisted
14. Specify the AI tool used and the prompts given
15. Explain your understanding of the AI-generated code
16. **Non-disclosure will result in disqualification**

Note: *We evaluate your ability to understand and explain the code during the follow-up interview.*

9. Submission Guidelines

9.1 Repository Structure

1. Create a private GitHub repository
2. Use meaningful commit messages (conventional commits preferred)
3. Add the rishimalgwa35@gmail.com email as collaborator

4. Send assignment on same email thread.

9.2 Required Files

- **README.md:** Setup instructions, architecture overview, screenshots/GIFs
- **AI_USAGE.md:** AI tool usage disclosure (mandatory if used)
- **ARCHITECTURE.md:** Brief explanation of architectural decisions
- **APK file:** Debug APK in /apk folder

9.3 Checklist

- App runs without errors on Android
- All mandatory features implemented
- flutter analyze shows no errors
- Code is properly formatted
- Repository access shared with reviewer

10. Timeline

Milestone	Details
Assignment Start	Upon receiving this document
Submission Deadline	24 hours from start time
Questions	Email: [recruiter-email]
Results	Within 3 business days

💡 Pro Tips for Success

• Focus on core features with quality before bonus features
• Commit frequently with meaningful messages
• A polished, working subset beats a buggy complete implementation
• Include GIFs in README to showcase animations
• Test thoroughly before submission

Good luck! 🚀