Video Feed App – Specifications Document

# Created by : Ahmed SILINI

## 📋 Table of Contents

1. Project Overview
2. Functional Requirements
3. Non-Functional Requirements
4. Technical Architecture
5. Data Models
6. User Interface Specifications
7. Firebase Configuration
8. Development Timeline
9. Testing Approach
10. Deployment & Delivery
11. Success Criteria

# 🎯 Project Overview

**Project Name:** Flutter Video Feed App - TikTok-Style Social Media Application

**Duration:** 4 days (Internship Assignment)

**Platform:** Android & iOS (Flutter)

**Scope:** Mobile application with vertically scrollable video feed, smart caching, user authentication, and social interactions

# ⚙ Functional Requirements

## F1 - User Authentication (High Priority)

 Anonymous authentication by default

 Email/password registration and login (Bonus)  Persistent authentication across sessions

 Secure logout functionality

## F2 - Video Feed Display (High Priority)

 Full-screen video playback in vertical scroll format  Automatic play/pause when videos come into view

 Smooth swipe transitions between videos  Basic video controls (play/pause overlay)

## F3 - Smart Video Caching (High Priority)

 Cache 3-video window (previous, current, next)  Background downloading of upcoming videos

 Automatic cache cleanup and storage management  Offline playback for cached content

## F4 - Social Interactions (Medium Priority)

 Like/dislike buttons with visual feedback  Real-time counter updates

 User interaction persistence in Firebase

 Visual indication of previous user actions

## F5 - Video Metadata (Medium Priority)

 Display video title, creator info, duration  Like/dislike counters

 Synchronization with Firestore database

## F6 - Comment System (Bonus - Low Priority)

 Comment input and display interface  Real-time comment updates

 User attribution for comments

# 🚀 Non-Functional Requirements

## Performance

 Video loading: < 2 seconds for cached videos  Smooth 60fps scrolling transitions

 Memory usage: < 200MB peak  App startup: < 3 seconds

## Reliability

 Network failure recovery with retry mechanisms  Offline functionality for cached content

 Graceful error handling with user-friendly messages  Data consistency between local cache and Firebase

## Security

 HTTPS/TLS encryption for all communications  Firebase security rules implementation

 Secure authentication token management  Input validation and sanitization

## Usability

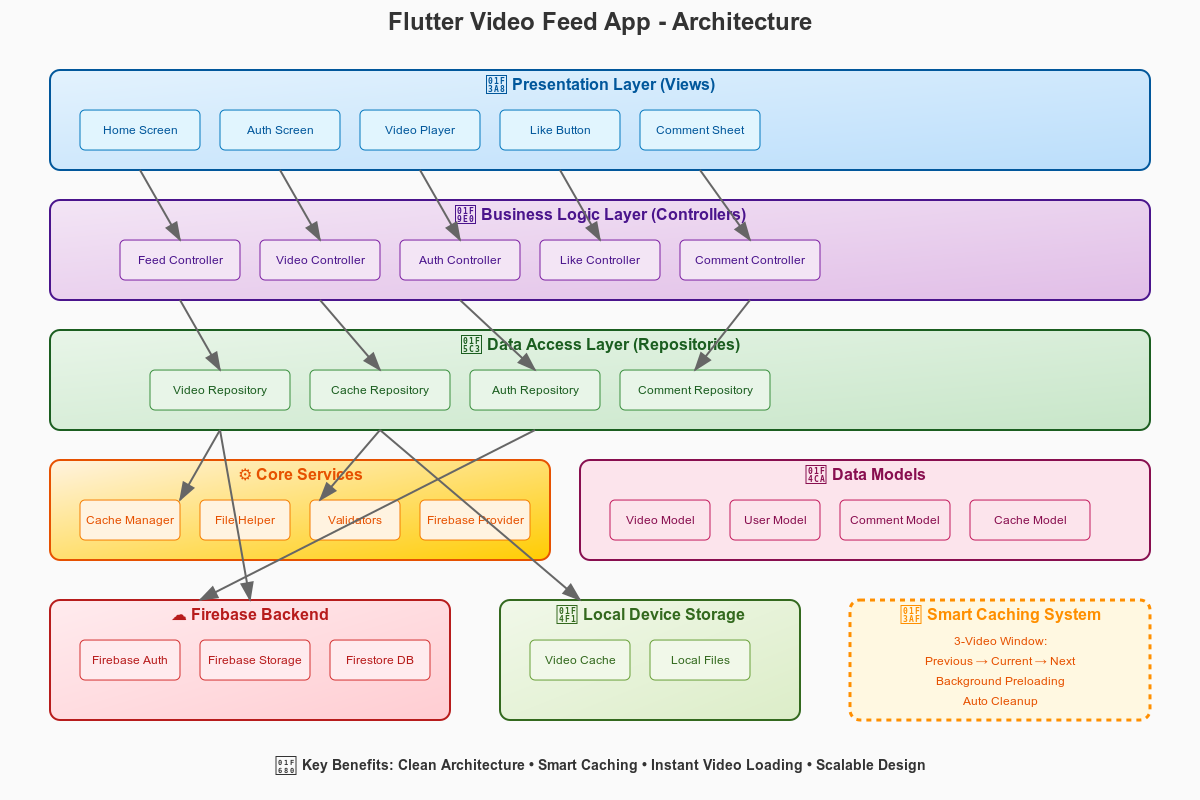
 Intuitive swipe navigation controls  Consistent UI design patterns

 Loading states and progress indicators

 Responsive design for different screen sizes

# 🏗 Technical Architecture

## Architecture Pattern



## Layer Structure

1. **Presentation Layer** - UI components and user interactions
2. **Business Logic Layer** - State management and business rules (Provider)
3. **Data Access Layer** - Repositories and API calls
4. **Core Services Layer** - Utilities and cross-cutting concerns

## Smart Caching System

 **3-Video Window:** Previous → Current → Next

 **Background Preloading:** Automatic download of upcoming videos

 **Memory Optimization:** Intelligent cleanup and storage management

📊 **Data Models**

## Video Model

 id, url, title

 likes, dislikes counters

 createdAt timestamp

 thumbnailUrl, duration

## User Model

 uid, email, displayName  isAnonymous flag

 createdAt, lastLoginAt timestamps

## Comment Model (Bonus)

 id, videoId, userId  content, createdAt

 likes counter, parentCommentId

## Video Cache Model

 videoId, localPath

 cachedAt timestamp

 fileSize, download progress

 Cache management met

# 🎨 User Interface Specifications

## Main Video Feed Screen

 **Layout:** Full-screen vertical PageView

 **Navigation:** Swipe gestures (up/down)

### Overlay Elements:

 Video title (bottom-left)

 Like/dislike buttons (bottom-right)  Progress indicator (top)

 Loading spinner (center)

## Authentication Screen

 Centered form with app branding  Email/password input fields

 Login/register buttons with anonymous option  Error message display

## Design System

 **Theme:** Dark theme optimized for video content

 **Typography:** Roboto font family

 **Icons:** Material Design icons

 **Animations:** Smooth transitions and loading states

# 🔥 Firebase Configuration

## Required Services

### Firebase Authentication

 Anonymous authentication (enabled)

 Email/password authentication (bonus)

### Firestore Database Collections:

//vviiddeeooss//{{vviiddeeooIIdd}}

├├──── uurl,,lr ttitle,,elti llikes,,seki ddislikessekilsi

├├──── ccreatedAt,,tAdetaer dduration,,noitaru tthumbnailUrllrUlianbmuh

//uusseerrss//{{uusseerrIIdd}}

├├──── eemail,,liam ddisplayName,,emaNyalpsi iisAnonymoussuomynonAs

├├──── ccreatedAt,,tAdetaer llastLoginAttAnigoLtsa

//uusseerriinntteerraaccttiioonnss//{{uusseerrIIdd}}//vviiddeeooss//{{vviiddeeooIIdd}}

├├──── iisLiked,,dekiLs iisDisliked,,dekilsiDs ttimestamppmatsemi

//ccoommmmeennttss//{{ccoommmmeennttIIdd}} ((BBoonnuuss))

├├──── vvideoId,,dIoedi uuserId,,dIres ccontenttnetno

├├──── ccreatedAt,,tAdetaer llikes,,seki pparentCommentIddItnemmoCtnera

### Firebase Storage Structure:

//vviiddeeooss//

├├──── vvideo1.mp4,,4pm.1oedi vvideo2.mp4,,4pm.2oedi vvideo3.mp4 pm.3oedi

//tthhuummbbnnaaiillss// ((OOppttiioonnaall))

├├──── vvideo11oedi tthumb.jpg,,gpj.bmuh vvideo22oedi tthumb.jpg gpj.bmuh

## Security Rules

 Videos readable by all authenticated users  User interactions private to each user

 Proper write permissions with validation

📅 **Development Timeline Day 1: Foundation Setup**

## Day 2: Specifications Document

**Day 3: Core Features**

## Day 4: Polish

**Day 5: Bonus**

# 🧪 Testing Approach

## Basic Testing Strategy

 **Manual Testing:** Test core features on device/emulator

### Key Areas to Test:

 Video playback and scrolling  Authentication flow

 Like/dislike functionality  Caching behavior

 Network connectivity scenarios

## Simple Testing Checklist

 ✅ Videos load and play correctly

 ✅ Smooth scrolling between videos

 ✅ Like/dislike buttons work

 ✅ Authentication persists across app restarts

 ✅ App handles poor network conditions

 ✅ Memory usage stays reasonable

# 🚀 Deployment & Delivery

## Development Environment

 **IDE:** VS Code or Android Studio

 **Flutter Version:** 3.0+

 **Target Platforms:** Android 5.0+, iOS 11.0+

## Delivery Requirements

 ✅ Source code in public GitHub repository

 ✅ Comprehensive README.md with setup instructions

 ✅ Firebase configuration guide

 ✅ Architecture documentation

 ✅ Working demo on physical device or emulator

## Repository Structure

fflluutttteerr--vviiddeeoo--ffeeeedd//

├├────

├├────

RREEAADDMMEE..mmdd

lliibb//

││ ├├────

││ ├├────

││ └└────

ccoorree// ffeeaattuurreess// mmaaiinn..ddaarrtt

├├────

├├────

├├────

└└────

aasssseettss// aannddrrooiidd// iiooss// ppuubbssppeecc..yyaammll

# ✅ Success Criteria

## Minimum Viable Product (MVP)

 ✅ User authentication (anonymous)

 ✅ Vertical video feed with smooth scrolling

 ✅ Video playback from Firebase Storage

 ✅ Basic caching functionality

 ✅ Like/dislike interactions

 ✅ Firebase Firestore integration

## Complete Solution

 ✅ Smart 3-video caching system

 ✅ Background preloading

 ✅ Memory optimization

 ✅ Error handling and offline support

 ✅ Clean, documented code  
 ✅ Professional README documentation