ANNEX 2

Application Flow Diagram

Flick MVP - iOS Application

October 8, 2025

---

1. Overview

This document describes the overall application flow, user journeys, data interactions, and backend architecture for the Flick MVP iOS application. It provides a comprehensive view of how different modules interact with each other and with the backend infrastructure.

2. System Architecture

Layer	Components	Technology
Presentation Layer	iOS Application (iPhone & iPad)	Flutter/React Native
API Layer	RESTful API Gateway	Node.js + Express.js
Business Logic Layer	Authentication, QR Processing, Trading Logic, Gamification Engine	Node.js
Data Layer	User DB, Lighter DB, Transaction DB	Supabase/Firebase
Storage Layer	Images, Documents, Backups	AWS S3 / Firebase Storage
Infrastructure Layer	Cloud Hosting, Load Balancing	AWS/GCP/Vercel

3. Key User Journey Flows

3.1 User Registration & Onboarding Flow

Step	Action	System Response
1	User downloads app from App Store	App launches with splash screen
2	User sees welcome screen	Display registration options (Email/Google OAuth)
3	User selects registration method	Navigate to respective auth flow
4	User provides credentials	Backend validates and creates account
5	Email verification sent (if email signup)	User verifies email via link
6	User completes profile (username, avatar)	Data stored in User DB
7	Onboarding tutorial displayed	User learns key features
8	User lands on home screen	Display personal collection (empty initially)

3.2 QR Scanning & Lighter Registration Flow

Step	Action	System Response
1	User taps "Scan QR" button	Camera opens with QR scanner overlay
2	User scans QR code on lighter	App validates QR code format
3	QR validated successfully	Backend checks if lighter already registered
4	If new lighter: show registration form	User enters lighter details (brand, color, notes)
5	User takes photo of lighter	Photo uploaded to cloud storage
6	User confirms registration	Backend creates lighter record, links to user
7	Success notification displayed	Lighter added to user collection
8	Points awarded for registration	Gamification system updates user score

3.3 Lost & Found Workflow

Step	Action	System Response
1	User reports lighter as lost	Opens "Report Lost" form with lighter details
2	User provides description & last known location	Data saved to Lost & Found DB
3	Lost lighter appears in community feed	Other users can see lost item listing
4	Another user finds and scans the lighter	System detects lighter is marked as lost
5	Finder sees "This lighter is lost" notification	Option to contact owner displayed
6	Finder initiates contact	In-app message sent to original owner
7	Owner and finder coordinate return	Chat system facilitates communication
8	Owner confirms return	Lighter marked as recovered, finder awarded points

3.4 Trading Workflow

Step	Action	System Response
1	User browses marketplace	Display available lighters for trade
2	User selects desired lighter	Show lighter details and owner profile
3	User sends trade request	Select lighter from own collection to offer
4	Trade request sent	Owner receives push notification
5	Owner reviews trade offer	Can accept, reject, or counter-offer
6	If accepted: trade initiated	Backend updates ownership records
7	Both users confirm trade completion	Ownership transferred in database
8	Trade recorded in history	Both users awarded trade points

4. Data Flow Architecture

4.1 Authentication & Authorization Flow

→ User initiates login (Email/Google OAuth)
→ iOS app sends credentials to API Gateway
→ API validates credentials against User DB
→ If valid: JWT token generated and returned
→ iOS app stores JWT token securely (Keychain)
→ All subsequent API calls include JWT token in header
→ API validates token on each request
→ Token expires after 24 hours, refresh required

4.2 QR Code Validation & Registration Flow

→ User scans QR code via iOS camera
→ App extracts unique lighter ID from QR code
→ App sends lighter ID to backend API for validation
→ Backend checks if ID format is valid
→ Backend queries Lighter DB to check if already registered
→ If new: create new lighter record with user_id linkage
→ Upload lighter photo to cloud storage (S3/Firebase)
→ Update User DB to link lighter to user's collection
→ Return success response with lighter details to app

4.3 Real-Time Messaging Flow (Lost & Found / Trading)

→ User initiates chat (from lost & found or trade screen)
→ App sends message via API to backend
→ Backend stores message in Messages DB
→ Backend sends push notification to recipient via FCM
→ Recipient's app receives push notification
→ Recipient opens chat, app fetches message history from API
→ Real-time updates via WebSocket or polling (every 5 seconds)
→ All messages encrypted in transit (HTTPS)

5. Database Schema Overview

5.1 Core Database Tables

Table Name	Key Fields	Purpose
users	user_id, email, username, avatar_url, created_at, points, level	Store user account information
lighters	lighter_id, qr_code, owner_id, brand, color, photo_url, registered_at	Store registered lighter data
trades	trade_id, requester_id, owner_id, lighter_offered_id, lighter_requested_id, status, created_at	Manage trade requests and history
lost_found	report_id, lighter_id, reporter_id, status, description, last_seen_location, reported_at	Track lost and found reports
messages	message_id, sender_id, recipient_id, content, timestamp, read_status	In-app messaging system
achievements	achievement_id, user_id, badge_type, earned_at	Store user achievements and badges
notifications	notification_id, user_id, type, content, read_status, created_at	Push notification records

6. Key API Endpoints

Endpoint	Method	Purpose
/api/auth/register	POST	User registration
/api/auth/login	POST	User login
/api/auth/oauth/google	POST	Google OAuth login
/api/lighters/register	POST	Register new lighter
/api/lighters/my-collection	GET	Fetch user's lighter collection
/api/lighters/marketplace	GET	Browse available lighters for trade
/api/trades/request	POST	Send trade request
/api/trades/respond	PUT	Accept/reject trade
/api/lost-found/report	POST	Report lighter as lost
/api/lost-found/list	GET	Get list of lost lighters
/api/messages/send	POST	Send message

/api/messages/conversation/:userId	GET	Fetch conversation history
/api/users/profile/:userId	GET	Get user profile
/api/users/leaderboard	GET	Fetch leaderboard
/api/notifications/send	POST	Send push notification

7. Security & Performance Considerations

7.1 Security Measures

- HTTPS encryption for all API communication
- JWT token-based authentication with secure storage
- Password hashing using bcrypt (salt rounds: 10)
- OAuth 2.0 implementation for Google Sign-In
- Input validation and sanitization on all endpoints
- Rate limiting to prevent API abuse (100 requests/minute per user)
- SQL injection prevention through parameterized queries
- XSS protection through content security policies
- Regular security audits and penetration testing

7.2 Performance Optimization

- Database indexing on frequently queried fields (user_id, lighter_id)
- Image compression before upload (max 1MB per image)
- Caching frequently accessed data (user profiles, leaderboard)
- Lazy loading for lighter collections (paginated results)
- CDN for static assets and images
- API response time target: < 200ms for most endpoints
- Background processing for non-critical tasks (analytics, email)
- Optimized database queries with proper joins
- Load balancing for high-traffic scenarios

8. Deployment Architecture

iOS App: Deployed to Apple App Store via Xcode + App Store Connect
Backend API: Hosted on AWS EC2 / Google Cloud Run / Vercel
Database: Supabase (managed PostgreSQL) or Firebase Realtime Database
File Storage: AWS S3 or Firebase Storage for user-uploaded images
Admin Dashboard: Deployed on Vercel or Netlify (static hosting)
Landing Website: Deployed on Vercel/Netlify with custom domain
CI/CD Pipeline: GitHub Actions for automated testing and deployment
Monitoring: AWS CloudWatch / Firebase Analytics for performance tracking
Backup Strategy: Daily automated database backups to secure cloud storage

---

This document forms Annex 2 of the Application Development Agreement between Good Monkeys LLC and CodeFlow Studios, dated October 8, 2025.