SyncIncorp Crèche Management Mobile Application – Functionality Checklist

Core Functional Requirements

1. Parent & Child Management

Parents can register securely via the app.

Each child profile includes personal info, guardian details, allergies, and emergency contacts.

Administrators can update and manage child records in real time.

All sensitive data stored in compliance with GDPR/POPIA.

Attendance Tracking

Real-time attendance recording (check-in/check-out).

Automatic synchronization with backend (Firestore + WebSocket).

Instant push notifications to parents when attendance status changes.

Offline attendance recording (auto-sync once reconnected).

Media Consent System

Explicit consent required before any child's media is displayed.

Granular permission settings (photos, videos, activity posts).

Consent history and audit trail maintained.

Parents can revoke consent at any time.

Access control ensures only authorized users view media.

Meal Ordering & Payments

Parents can view meal plans and pre-order meals.

Stripe API integrated for secure payments.

Confirmation receipts sent to parent dashboards and email.

Dietary restrictions automatically flagged in meal selection.

Refunds or order changes handled securely within app.

Event & Calendar Management

Crèche admins can create and update upcoming events.

Events automatically sync with parents' Google Calendars.

Push notifications for new or changed events.

RSVP and event reminders integrated.

Communication Tools

One-way communication from administrators to parents (announcements, newsletters).

Read receipts or confirmation indicators for messages.

Notifications delivered through Firebase Cloud Messaging.

Progress Reporting

Teachers can log progress and milestone reports.

Parents can view progress summaries in the dashboard.

Reports stored securely and accessible only to authorized users.

2. User Experience (UX/UI) Requirements

Dashboard & Navigation

Centralized dashboard showing real-time child status and updates.

Gestalt-based grouping for logical layout and visual clarity.

Quick-access buttons for Attendance | Media | Messages | Events | Meals.

Child-friendly visual design with warm, trust-building color palette.

Accessibility & Usability

Simple, intuitive navigation (minimum 3 taps to any key function).

Readable font sizes and contrast meeting WCAG 2.1 standards.

Multi-language support (if applicable).

Responsive design across Android and iOS.

3. System Architecture & Technical Requirements

Frontend (Flutter)

Built using Flutter (cross-platform).

State management via Provider or Bloc.

Navigation implemented through Navigator 2.0.

Lazy loading for images/media.

Offline data caching (SQLite).

Backend (Node.js + Firebase)

Express.js server for API handling.

Firebase Auth for secure login and role-based access.

Firestore database for real-time updates.

Firebase Storage for consent-controlled media files.

Redis caching for session management.

WebSocket for instant updates.

Integration Services

Stripe API for payments.

Google Calendar API for event syncing.

Firebase Cloud Messaging for notifications.

4. Performance & Optimization

API responses compressed (GZIP).

Background sync for non-critical updates.

Local caching of frequently accessed data.

Progressive media loading for bandwidth efficiency.

App meets responsiveness benchmark (<200ms interaction delay).

Performance monitoring with real-time metrics.

5. Security & Compliance

Data Protection

All transmissions encrypted with TLS 1.3.

Privacy-by-design approach implemented.

Minimal personal data collected (data minimization).

Right-to-erasure functionality implemented.

Role-based access control (Admin / Teacher / Parent).

Audit & Backup

All data changes logged with timestamps.

Daily automated backups with geo-redundant storage.

Regular security vulnerability scanning.

Legal Compliance

Fully compliant with GDPR and POPIA.

Explicit consent and opt-out capabilities.

Adherence to child data protection regulations.

6. CI/CD & Quality Assurance

Continuous Integration (GitHub Actions)

Automated unit tests executed on every commit.

Code quality enforced (ESLint/Dart Analyzer).

Security vulnerability scanning before deployment

Build & Deployment

Android (AAB/APK) and iOS (IPA) packages generated automatically.

Staging deployment for testing via Firebase App Distribution.

Production deployment to Play Store & App Store.

Database migration scripts executed automatically.

Testing Requirements

Minimum 85% code coverage for unit and integration tests.

User Acceptance Testing (UAT) conducted with parent/teacher samples.

Performance stress testing under peak loads.

Cross-device and OS version compatibility verified.

7. Offline & Emergency Capabilities

App remains partially functional offline (attendance, contact info, forms).

Automatic data sync once reconnected.

Access to emergency contact info without connectivity.

Notification queuing for delivery when back online.

8. Documentation & Support

Technical documentation (API endpoints, architecture diagrams).

User manuals for parents, teachers, and admins.

Privacy policy and data use agreement visible in-app.

Support/helpdesk contact or chatbot integrated.

Version control and changelog tracking.

9. Acceptance Criteria Summary

For the app to be considered 100% functional, it must meet all requirements across core features, UX/UI, technical implementation, security, compliance, and QA benchmarks. It must achieve 85%+ test coverage, pass all UAT feedback, and maintain full GDPR/POPIA compliance.