Mobile Application Development Plan - School Management System

Executive Summary

This plan outlines the development strategy for completing the React Native mobile application that integrates with the existing School Management System backend.

Current State Analysis

Technology Stack (Already Implemented)

- **Framework:** React Native 0.73.11 with TypeScript
- **State Management:** Redux Toolkit with Redux Persist
- **Navigation:** React Navigation v6 (native-stack, bottom-tabs, drawer)
- **UI Library:** React Native Paper (Material Design)
- **API Client:** Axios with interceptors
- **Forms:** Formik + Yup validation
- **Storage:** AsyncStorage, MMKV (encrypted)
- **Push Notifications:** Firebase Cloud Messaging

Implementation Status

✅ **Fully Implemented:**

- Multi-tenant authentication flow (TenantSelectionScreen → LoginScreen)
- JWT token management with auto-refresh
- Redux store with 10 slices (auth, user, tenant, student, attendance, exam, fee, library, transport, notification)
- API client with request/response interceptors
- 11 API service modules (auth, student, attendance, exam, fee, library, transport, communication, staff, academic)
- Type definitions for all entities (800+ lines in models.ts)
- Role-based navigation (9 user types supported)
- Dashboard screens for all 4 primary roles (Admin, Teacher, Student, Parent)
- Common components (Button, Card, Input, LoadingSpinner, EmptyState, Header, ScreenWrapper)
- Constants system (colors, fonts, spacing, validation patterns)

⚠️ **Partially Implemented (UI exists, needs API integration):**

- Attendance marking interface
- Exam schedule viewing
- Fee payment flow
- Library book catalog
- Transport tracking

❌ **Not Yet Implemented:**

- Complete attendance marking workflow
- Exam creation and marks entry
- Fee payment gateway integration
- Library issue/return workflows
- Transport real-time tracking
- Communication/notice creation
- Timetable viewing and management
- Profile management features
- Settings screens

Backend Integration Points

API Base URL

- **Development:** `http://localhost:8000/api/v1/`
- **Production:** `https://api.schoolmgmt.com/api/v1/`

Multi-Tenancy Implementation

- **Tenant Resolution:** `X-Tenant-Subdomain` header required on all requests
- **Schema Isolation:** PostgreSQL schema-per-tenant architecture
- **Tenant Selection:** Stored in Redux state and persisted locally

Available Backend Endpoints

The backend provides comprehensive RESTful APIs across 10 modules:

1. Authentication (`/auth/`) - 10+ endpoints
2. Students (`/students/`) - 15+ endpoints
3. Academics (`/academics/`) - 8+ endpoints
4. Attendance (`/attendance/`) - 12+ endpoints
5. Finance (`/finance/`) - 15+ endpoints
6. Examinations (`/examinations/`) - 8+ endpoints
7. Communication (`/communication/`) - 6+ endpoints
8. Library (`/library/`) - 8+ endpoints
9. Transport (`/transport/`) - 8+ endpoints
10. Staff (`/staff/`) - 6+ endpoints

Development Strategy

User Preferences Summary

Based on the requirements discussion:

- **Approach:** Horizontal development (one feature across all roles before moving to next)
- **Timeline:** 20+ weeks for comprehensive implementation with all features
- **Payment:** Mock payment gateway for testing (no real transactions initially)
- **Offline:** Online-only with graceful degradation and cached data viewing

Phase Structure Overview

Development will follow horizontal implementation across all user roles, organized into 6 comprehensive phases over 20+ weeks:

Phase 1: Dashboard Enhancement - All Roles (Weeks 1-4)

Goal: Complete and integrate dashboards for all 9 user roles with real backend data

Deliverables:

1. **Super Admin Dashboard** (Step 1 - Global Oversight)
    - Platform-wide tenant monitoring
    - Active users count per school
    - Revenue analytics
    - System health indicators
    - Quick actions: Add new school, view reports
2. **Principal Dashboard** (Step 4 - Academic Identity Setup)
    - Academic year selector
    - School statistics (total students, staff, classes)
    - Board configuration shortcut
    - Grading system overview
    - Pending approvals counter
3. **Admin Dashboard** (Step 5 - Operations Hub) (Currently AdminDashboard.tsx)
    - Refine existing dashboard with real API calls
    - Module shortcuts: Admissions, Scheduling, Fee Collection
    - Approval workflows widget
    - Recent activities feed
4. **Teacher Dashboard** (Step 8 - Modern Teacher Workspace) (Currently TeacherDashboard.tsx at 369 lines)
    - **File:** `src/screens/Dashboard/TeacherDashboard.tsx`
    - Replace Alert.alert calls (lines 220, 227, 234, 241) with actual navigation
    - Implement teacher-specific API endpoint: `GET /staff/teachers/me/dashboard/`
    - Add classroom activation flow: Fetch assigned classes with student rosters
    - Fix mock data fallback (lines 89-96) - use real backend or show empty state
    - Implement quick actions routing:
        - Mark Attendance → AttendanceStack.MarkAttendance
        - Enter Marks → ExamStack.MarksEntry
        - My Timetable → AcademicStack.Timetable
5. **Student Dashboard** (Step 12 - Digital Student Experience) (Currently StudentDashboard.tsx)
    - Today's timetable widget
    - Pending homework counter with drill-down
    - Recent grades showcase
    - Attendance percentage circular progress

   - Quick access: Digital ID, Submit Homework

6. **Parent Dashboard** (Step 11 - Family Hub) (Currently ParentDashboard.tsx)
   - **File:** `src/screens/Dashboard/ParentDashboard.tsx`
   - Multi-child selector: Horizontal scroll chips with avatar
   - Per-child statistics: Attendance, fee status, recent grades
   - Critical alerts section (overdue fees, low attendance)
   - Quick actions: Pay Fees, Track Bus, Contact Teacher

7. **Accountant Dashboard**
   - Daily fee collection summary
   - Pending payments counter
   - Recent transactions list
   - Quick actions: Record Payment, Generate Report

8. **Librarian Dashboard**
   - Books issued today
   - Overdue books counter
   - Recent returns
   - Quick actions: Issue Book, Return Book, Add New Book

9. **Transport Manager Dashboard**
   - Active routes status
   - Vehicles on duty
   - Today's incidents/delays
   - Quick actions: Track Vehicle, Update Route, Manage Drivers

Technical Tasks:

- Convert MainNavigator tabs to Stack Navigators for drill-down
- Implement deep linking configuration
- Add pull-to-refresh for all dashboards
- Create reusable dashboard components: StatCard, QuickActionCard, AlertBanner
- Set up navigation parameter types for all stacks

Files Modified:

- `src/navigation/MainNavigator.tsx` (convert to nested stacks)
- `src/screens/Dashboard/TeacherDashboard.tsx` (replace mocks, add navigation)
- `src/screens/Dashboard/ParentDashboard.tsx` (multi-child selector)
- `src/screens/Dashboard/AdminDashboard.tsx` (real API integration)
- **NEW:** `src/screens/Dashboard/SuperAdminDashboard.tsx`
- **NEW:** `src/screens/Dashboard/PrincipalDashboard.tsx`
- **NEW:** `src/screens/Dashboard/AccountantDashboard.tsx`
- **NEW:** `src/screens/Dashboard/LibrarianDashboard.tsx`
- **NEW:** `src/screens/Dashboard/TransportManagerDashboard.tsx`

Phase 2: Authentication & User Management - All Roles (Weeks 5-7)

Goal: Complete authentication flows, profile management, and notification system for all users

Deliverables:

1. **Enhanced Authentication**
   - Email verification flow (Step 10 - Parent Onboarding component)
   - Biometric authentication (Face ID/Touch ID) for app unlock
   - Password reset with OTP verification
   - "Remember Me" functionality
   - Session management and auto-logout
2. **Profile Management - Role-Based**
   - **Teacher Profile:** Subjects taught, assigned classes, qualifications
   - **Student Profile:** Admission details, parents info, health records, documents
   - **Parent Profile:** Children linked, contact preferences
   - **Admin Profile:** Role, permissions, access level
   - Profile photo upload and cropping
   - Contact information update
3. **Settings & Preferences**
   - Notification preferences (per category: Attendance, Fees, Exams, Notices)
   - Language selection (English, Hindi)
   - Theme toggle (Light/Dark mode foundation)
   - Privacy settings
   - App version and diagnostics
4. **Notification System**
   - Firebase Cloud Messaging setup (Android + iOS)
   - In-app notification center with tabs (Unread, All, Archived)
   - Push notification handlers (foreground, background, killed state)
   - Deep linking from notifications
   - Notification badges on tab icons
   - Rich notifications with images and action buttons
5. **Help & Support Center** (Step 14)
   - FAQ accordion (searchable)
   - Video tutorials (embedded YouTube/Vimeo)
   - Contact school admin (in-app form submission)
   - Feedback submission
   - App tour/onboarding for first-time users

Technical Tasks:

- Integrate Firebase SDK for FCM

- Implement biometric service wrapper (react-native-biometrics)
- Create notification service with deep link routing
- Build profile update API integration
- Implement settings persistence with Redux + AsyncStorage
- Add image picker and cropper for profile photos

Files Created:

- `src/services/firebase.service.ts` (FCM setup and handlers)
- `src/services/biometric.service.ts` (Biometric authentication)
- `src/screens/Auth/EmailVerificationScreen.tsx`
- `src/screens/Auth/BiometricSetupScreen.tsx`
- `src/screens/Profile/EditProfileScreen.tsx`
- `src/screens/Profile/SettingsScreen.tsx`
- `src/screens/Profile/NotificationCenterScreen.tsx`
- `src/screens/Profile/ChangePasswordScreen.tsx`
- `src/screens/Support/HelpCenterScreen.tsx`
- `src/screens/Support/FAQScreen.tsx`
- `src/screens/Support/FeedbackScreen.tsx`
- `src/config/firebase.config.ts`

Files Modified:

- `src/store/slices/authSlice.ts` (add biometric state)
- `src/store/slices/notificationSlice.ts` (FCM token, preferences)
- `src/services/api/authService.ts` (email verification, biometric methods)

Phase 3: Academic Setup & Student Management - Admin Focus (Weeks 8-11)

Goal: Complete student admission workflow, academic structure setup, and enrollment management

Deliverables:

1. **Principal's Academic Identity Setup** (Step 4)
   - Board selection (CBSE, ICSE, State boards, International)
   - Grading system configuration: Percentage-based vs Grade-based (A+, A, B, C)
   - Academic year creation wizard (start date, end date, terms)
   - Academic calendar setup (terms, holidays, exam schedules)
2. **School Structure Management** (Admin)
   - Class creation (1-12, Nursery, KG, etc.)
   - Section management (A, B, C per class)
   - Subject allocation per class
   - Teacher assignment to classes and subjects
   - Classroom allocation
3. **Student Admission Form** (Step 6 - Multi-Step Form)
   - **Step 1: Student Information**

        - Personal details (name, DOB, gender, blood group, Aadhaar)
        - Nationality, religion, caste category
        - Photo upload
    - **Step 2: Guardian Information**
        - Father's details (name, occupation, phone, email, Aadhaar)
        - Mother's details (name, occupation, phone, email, Aadhaar)
        - Guardian selection if different from parents
        - Emergency contact
    - **Step 3: Previous School Details**
        - School name, board, class last attended
        - Transfer certificate upload
        - Reason for transfer
    - **Step 4: Documents Upload**
        - Birth certificate
        - Transfer certificate (if applicable)
        - Aadhaar card
        - Recent photograph
        - Caste certificate (if applicable)
    - Draft saving functionality
    - Form validation with react-hook-form + zod

4. **Admission Approval Workflow** (Admin)

    - Pending admissions list with filters
    - Admission detail view with document previewer
    - Approve/Reject actions with remarks
    - Bulk approval for multiple students
    - Auto-generate admission number on approval
    - Email/SMS notification to parents on approval

5. **Student Enrollment Management**

    - Enroll approved students to class and section
    - Subject selection (for higher classes with optional subjects)
    - Fee structure assignment
    - Generate student ID card
    - Parent account creation and linking

6. **Student Profile & Records**

    - Comprehensive student profile view
    - Document management (view, download, delete, re-upload)
    - Health records tracking (vaccinations, allergies, medical conditions)
    - Parent information view and edit
    - Academic history
    - Disciplinary records (if any)
    - Student transfer workflow

7. **Student Search & Management**

- Advanced search (name, admission number, class, section, status)
- Bulk operations (promote students, transfer section)
- Export student list (CSV/Excel)
- Student status management (Active, Alumni, Transferred, Expelled)

Technical Tasks:

- Implement multi-step form with progress indicator
- Build document upload with preview and validation
- Create approval workflow with state machine
- Implement student search with debouncing and filters
- Build PDF viewer for documents (react-native-pdf)
- Implement bulk actions with confirmation dialogs

Files Created:

- `src/screens/Admin/StudentAdmissionFormScreen.tsx` (Multi-step form)
- `src/screens/Admin/AdmissionApprovalScreen.tsx` (Approval workflow)
- `src/screens/Admin/StudentEnrollmentScreen.tsx` (Enroll to class)
- `src/screens/Admin/PrincipalSetupScreen.tsx` (Academic setup wizard)
- `src/screens/Admin/AcademicStructureScreen.tsx` (Class/section management)
- `src/screens/Student/StudentProfileScreen.tsx` (Enhanced profile)
- `src/screens/Student/StudentDocumentsScreen.tsx` (Document viewer)
- `src/screens/Student/StudentSearchScreen.tsx` (Advanced search)
- `src/components/features/student/AdmissionFormSteps/` (Step components)
- `src/components/features/student/StudentCard.tsx`
- `src/components/features/student/DocumentViewer.tsx`

Files Modified:

- `src/store/slices/studentSlice.ts` (Add admission workflow, enrollment actions)
- `src/services/api/studentService.ts` (Add admission, approval, enrollment endpoints)
- `src/services/api/academicService.ts` (Add class, subject, academic year methods)

Phase 4: Attendance & Timetable - Teacher, Student, Parent (Weeks 12-15)

Goal: Complete attendance marking, leave management, timetable viewing, and QR-based check-in

Deliverables:

1. **Classroom Activation** (Step 7 - Teacher)
    - View assigned classes (from `/academic/classes/?teacher={id}`)
    - Student roster per class with photos and roll numbers
    - Quick stats: Total students, present today, attendance percentage
    - Quick actions: Mark Attendance, View Timetable, Contact Parents
2. **Attendance Marking - Three Modes** (Step 8 - Teacher)
    - **Mode 1: Bulk Class Attendance (Default)**

    - Grid view of all students with photos
    - Default status: PRESENT (tap to cycle: Present → Absent → Late → Half Day → Leave)
    - Visual color coding: Green (Present), Red (Absent), Orange (Late), Blue (Half Day), Gray (Leave)
    - Single "Submit Attendance" button with summary preview
    - Offline capability: Queue submissions, sync when online

  - **Mode 2: Individual Student Attendance**

    - List view with dropdown per student
    - Optional remarks field per student
    - Suitable for corrections or partial attendance marking

  - **Mode 3: QR-Based Attendance (Automated)**

    - Student scans QR code at school entry
    - System auto-marks as Present with timestamp
    - Late arrivals flagged if after configured time
    - Security guards can use scanner app

3. **Digital Student ID Card** (Step 12 - Student)

   - QR code generation with HMAC signature
   - Encoded data: Student ID, admission number, name, class, validity period
   - 24-hour validity with auto-renewal
   - Biometric unlock option before showing QR
   - Offline-available (cached in AsyncStorage)
   - School logo watermark
   - Student photo and basic details display

4. **Attendance Viewing & Reports**

   - **Teacher:** Class-wise attendance summary (today, this week, this month)
   - **Student:** Personal attendance history with calendar view
   - **Parent:** Child's attendance with percentage and alerts
   - **Admin:** School-wide attendance dashboard with defaulter list
   - Filters: Date range, class, section, status
   - Export attendance reports (CSV/PDF)

5. **Leave Management**

   - **Student/Parent:** Submit leave request with reason and dates
   - **Teacher/Admin:** Approve/Reject leave requests
   - Leave types: Sick, Casual, Emergency, Other
   - Attachment support (medical certificate)
   - Auto-mark attendance as "Leave" on approval
   - Leave balance tracking (if applicable)
   - Email/SMS notification on approval/rejection

6. **Holiday Calendar**

    - School-wide holiday list (national, regional, school-specific)
    - Festival/occasion display
    - Add/Edit/Delete holidays (Admin only)
    - Monthly calendar view with highlighted holidays
    - Sync with timetable (no classes on holidays)

7. **Timetable Management**

    - **Admin:** Create and assign timetables per class/section
    - Period-wise schedule (time slots, subjects, teachers, rooms)
    - **Teacher:** View personal timetable (all classes they teach)
    - **Student:** View class timetable with period details
    - **Parent:** View child's timetable
    - Weekly grid view (Monday-Saturday)
    - Color-coded by subject
    - Free periods indication
    - Substitute teacher assignment

Technical Tasks:

- Implement QR code generation using react-native-qrcode-svg
- Build QR scanner with signature verification (react-native-camera or expo-barcode-scanner)
- Create attendance grid component with touch optimization
- Implement calendar component with date selection
- Build timetable grid with horizontal/vertical scrolling
- Add online/offline sync queue for attendance
- Implement leave approval state machine

Files Created:

- `src/screens/Teacher/ClassroomActivationScreen.tsx` (Step 7)
- `src/screens/Attendance/MarkAttendanceScreen.tsx` (Bulk + Individual modes)
- `src/screens/Attendance/QRAttendanceScannerScreen.tsx` (Security/Admin)
- `src/screens/Student/DigitalIDCardScreen.tsx` (Step 12 - QR display)
- `src/screens/Attendance/AttendanceHistoryScreen.tsx` (Student/Parent view)
- `src/screens/Attendance/AttendanceSummaryScreen.tsx` (Teacher/Admin)
- `src/screens/Attendance/SubmitLeaveRequestScreen.tsx` (Student/Parent)
- `src/screens/Attendance/LeaveApprovalScreen.tsx` (Teacher/Admin)
- `src/screens/Attendance/HolidayCalendarScreen.tsx` (All roles)
- `src/screens/Academic/TimetableScreen.tsx` (All roles)
- `src/screens/Academic/CreateTimetableScreen.tsx` (Admin)
- `src/components/features/attendance/BulkAttendanceGrid.tsx`
- `src/components/features/attendance/AttendanceCalendar.tsx`
- `src/components/features/attendance/AttendanceStatusBadge.tsx`
- `src/components/features/student/DigitalIDCard.tsx`
- `src/components/features/timetable/TimetableGrid.tsx`
- `src/components/features/timetable/PeriodCard.tsx`
- `src/services/qr.service.ts` (QR generation and verification)

Files Modified:

- `src/store/slices/attendanceSlice.ts` (Add QR state, offline queue, leave management)
- `src/services/api/attendanceService.ts` (Add QR check-in, bulk mark, leave endpoints)
- `src/services/api/academicService.ts` (Add timetable CRUD methods)

Phase 5: Examinations & Grading - Teacher, Student, Parent (Weeks 16-18)

Goal: Complete exam management, marks entry, grade calculation, and result publishing

Deliverables:

1. **Exam Management** (Admin)
   - Create exam: Name, type (Formative, Summative, Term, Final), date range
   - Exam schedule: Subject-wise date and time slots
   - Assign classes and sections to exams
   - Set total marks per subject
   - Configure grading scale (percentage or grade-based)
   - Publish exam schedule to teachers/students
2. **Teacher's Gradebook** (Step 9 - Marks Entry)
   - **Grid View (Primary Mode):**
     - Spreadsheet-like interface (students in rows, subjects in columns)
     - Fixed first column (student names with photos)
     - Horizontal scroll for subjects
     - Real-time validation (max marks check, numeric only)
     - Total marks calculation
     - Percentage/Grade auto-calculation
     - Color coding: Red for fail, Green for pass, Yellow for borderline
   - **Form View (Alternative Mode):**
     - Student-by-student marks entry
     - Subject dropdown selection
     - Suitable for corrections or partial entry
   - **Features:**
     - Save draft every 2 minutes (auto-save)
     - Publish marks (two-step: draft → published)
     - Bulk copy marks from previous exam (for re-tests)
     - Absent marking (AB) for students who missed exam
     - Remarks/comments per student
     - Undo/Redo functionality
3. **Grade Calculation & Result Generation**
   - Auto-calculate total marks, percentage, grade
   - Apply grading scale: A+ (90-100), A (80-89), B (70-79), etc.

   - Class rank calculation (optional)
   - Subject-wise performance analysis
   - Generate result summary (pass/fail, class average)
   - Bulk result publishing with confirmation

4. **Result Viewing**

   - **Student:** View own marks and grade
   - **Parent:** View child's results with subject-wise breakdown
   - **Teacher:** View class-wise results with analytics
   - **Admin:** School-wide result dashboard with pass percentage
   - Graphical representation: Bar charts for subject comparison
   - Historical performance tracking (compare across terms)
   - Topper list display

5. **Report Card Generation**

   - PDF report card generation (backend API)
   - Customizable template per school (logo, layout)
   - Include: Student info, subject marks, total, percentage, grade, rank, remarks
   - Teacher signatures placeholder
   - Download and share via email/WhatsApp
   - Print-ready format

6. **Exam Notifications**

   - Notify students/parents when exam schedule is published
   - Remind students 1 day before exam
   - Notify parents when results are published
   - Alert parents if child fails in any subject

Technical Tasks:

- Build marks entry grid with optimized rendering (react-native-flash-list)
- Implement auto-save with debouncing (save every 2 minutes or on field blur)
- Create grade calculation logic with configurable scales
- Integrate PDF generation library for report cards
- Build result analytics dashboard with charts (victory-native)
- Implement marks entry validation with field-level error display

Files Created:

- `src/screens/Exam/ExamManagementScreen.tsx` (Admin - create/edit exams)
- `src/screens/Exam/ExamScheduleScreen.tsx` (All roles - view schedule)
- `src/screens/Exam/MarksEntryScreen.tsx` (Step 9 - Teacher gradebook)
- `src/screens/Exam/MarksEntryGridView.tsx` (Spreadsheet mode)
- `src/screens/Exam/MarksEntryFormView.tsx` (Form mode)
- `src/screens/Exam/ResultViewScreen.tsx` (Student/Parent - view results)
- `src/screens/Exam/ReportCardScreen.tsx` (Download/view report card)
- `src/screens/Exam/ResultAnalyticsScreen.tsx` (Teacher/Admin - analytics)

- `src/components/features/exam/MarksEntryGrid.tsx` (Grid component)
- `src/components/features/exam/GradeDisplay.tsx` (Grade badge)
- `src/components/features/exam/ResultCard.tsx` (Result summary card)
- `src/components/features/exam/PerformanceChart.tsx` (Subject comparison)

Files Modified:

- `src/store/slices/examSlice.ts` (Add marks entry state, draft management, result cache)
- `src/services/api/examService.ts` (Add marks CRUD, result publishing, report card download)

Phase 6: Fee Management & Payments - Parent, Student, Accountant (Weeks 19-21)

Goal: Complete fee structure setup, payment processing, and financial tracking

Deliverables:

1. **Fee Structure Management** (Admin/Accountant)
   - Create fee categories (Tuition, Transport, Library, Lab, Sports, etc.)
   - Set fee structure per class (different fees for different classes)
   - Define payment installments (Term-wise, Monthly, Quarterly)
   - Set due dates and late fee penalties
   - Assign fee structure to students (bulk or individual)
   - Configure discounts (sibling discount, merit scholarship, financial aid)
2. **Fee Viewing** (Parent/Student)
   - View assigned fee structure with breakdown
   - Outstanding balance display
   - Paid amount vs total amount progress bar
   - Installment schedule with due dates
   - Late fee calculation if overdue
   - Payment history with receipts
3. **Payment Flow** (Parent/Student)
   - **Payment Options:**
     - Pay full balance
     - Pay next installment
     - Pay custom amount (with min/max validation)
   - **Mock Payment Gateway Integration:**
     - Simulated Razorpay/Stripe checkout flow
     - Test card numbers for different scenarios (success, failure, pending)
     - Payment confirmation screen
     - Auto-generate receipt on successful payment
   - **Payment Methods (Mock):**
     - Credit/Debit Card (simulated)

        - UPI (simulated)
        - Net Banking (simulated)
        - Cash (offline - recorded by accountant)

    - **Receipt Generation:**

        - Digital receipt with transaction ID
        - PDF download option
        - Email receipt to parent
        - Include: Student info, payment date, amount, mode, receipt number

4. **Payment Recording** (Accountant)

    - Record offline payments (cash, cheque, bank transfer)
    - Upload payment proof (bank slip, cheque photo)
    - Manual receipt generation
    - Adjust fee (waiver, discount application)
    - Mark fee as paid/partially paid/overdue
    - Refund processing

5. **Fee Reports & Analytics** (Accountant/Admin)

    - Daily collection report
    - Outstanding fees report (defaulter list)
    - Class-wise fee collection summary
    - Payment mode distribution
    - Revenue projection
    - Export reports (CSV/Excel/PDF)

6. **Fee Reminders & Alerts**

    - Auto-remind parents 3 days before due date
    - Overdue fee alert (daily notifications)
    - Payment confirmation notification
    - SMS/Email reminders (backend integration)

Technical Tasks:

- Create mock payment service mimicking Razorpay API
- Implement payment state machine (Pending → Processing → Success/Failed)
- Build receipt PDF generator
- Create fee calculation logic with late fees and discounts
- Implement payment history with infinite scroll
- Build financial charts and analytics dashboard

Files Created:

- `src/screens/Fee/FeeStructureScreen.tsx` (View fee breakdown)
- `src/screens/Fee/PaymentFlowScreen.tsx` (Step-by-step payment)
- `src/screens/Fee/PaymentOptionsScreen.tsx` (Select payment mode)
- `src/screens/Fee/PaymentConfirmationScreen.tsx` (Success/Failure)

- `src/screens/Fee/PaymentHistoryScreen.tsx` (Past payments with receipts)
- `src/screens/Fee/ReceiptViewScreen.tsx` (Digital receipt)
- `src/screens/Fee/RecordPaymentScreen.tsx` (Accountant - offline payments)
- `src/screens/Fee/FeeReportsScreen.tsx` (Accountant/Admin - analytics)
- `src/screens/Fee/OutstandingFeesScreen.tsx` (Defaulter list)
- `src/components/features/fee/FeeBreakdownCard.tsx`
- `src/components/features/fee/PaymentMethodSelector.tsx`
- `src/components/features/fee/ReceiptViewer.tsx`
- `src/components/features/fee/InstallmentSchedule.tsx`
- `src/services/payment.service.ts` (Mock payment gateway)

Files Modified:

- `src/store/slices/feeSlice.ts` (Add payment state, receipt cache, installment tracking)
- `src/services/api/feeService.ts` (Add payment order creation, verification, receipt download)

Phase 7: Extended Features - Library, Transport, Communication (Weeks 22-25)

Goal: Complete library management, transport tracking, communication tools, and homework submission

Deliverables:

1. **Library Management** (Librarian Focus)
   - **Book Catalog:**
     - Browse books by category (Fiction, Non-Fiction, Textbooks, Reference, etc.)
     - Search books by title, author, ISBN, publisher
     - Book details: Cover image, synopsis, availability status
     - Filter by: Category, author, publication year, language
     - Add new books (Librarian only)
     - Edit book details, update stock quantity
   - **Book Issue/Return:**
     - Issue book to student (scan student ID or search by name)
     - Set due date (default: 14 days)
     - Check student's issue limit (max 3 books at a time)
     - Return book with condition notes
     - Calculate overdue fine if applicable
     - Renewal option (extend due date)
   - **Book Reservations:**
     - Student can reserve currently issued books
     - Queue management (FIFO)
     - Notify student when book is available
   - **Overdue Tracking:**
     - List of overdue books with student details

        - Auto-calculate fine (₹5 per day)
        - Send overdue reminders to students
        - Fine payment integration with fee module
    - **Library Reports:**
        - Most issued books
        - Student-wise issue history
        - Overdue summary
        - Stock availability report

2. **Student Homework Submission** (Step 13)
    - View assigned homework per subject
    - Homework details: Subject, topic, due date, instructions, attachments (reference materials)
    - Upload homework solution:
        - Image upload (photo of handwritten work)
        - PDF upload (typed assignments)
        - Multiple file support (max 10MB per file)
    - Text submission for essay-type assignments
    - Submission status: Pending, Submitted, Graded, Resubmit
    - View teacher's feedback and marks
    - Resubmission if requested by teacher
    - Submission history
3. **Homework Management** (Teacher)
    - Create homework assignment
    - Set due date and attach reference materials
    - Assign to class/section or individual students
    - View submissions with download option
    - Grade homework and provide feedback
    - Request resubmission if needed
    - Track submission rate
4. **Transport Tracking** (Parent/Transport Manager)
    - View assigned bus route with stops
    - Live bus location on map (polling every 30 seconds)
    - ETA calculation to student's stop
    - Stop sequence indicator (upcoming stops highlighted)
    - Driver contact button (call directly)
    - Route schedule (pickup and drop timings)
    - Delay notifications
    - Absence marking (inform driver if student won't board)
5. **Transport Management** (Transport Manager)
    - Vehicle management (add, edit, status: Active, Maintenance, Out of Service)
    - Route creation with stop sequence

    - Driver assignment to vehicles
    - Student route allocation (assign students to routes)
    - Attendance tracking (who boarded, who missed)
    - Fuel and maintenance logs
    - Route optimization suggestions

6. **Communication Tools** (All Roles)

    - **Notices:**

        - Create notice (Admin/Principal/Teacher)
        - Target audience: All, Specific class, Specific role
        - Priority: High, Medium, Low
        - Attach files (circulars, PDFs)
        - View notices (role-based filtering)
        - Mark as read
        - Archive old notices

    - **Events:**

        - Create event (Admin/Principal)
        - Event types: Sports Day, Annual Function, Parent-Teacher Meeting, Holiday, Exam
        - Set date, time, venue, description
        - Attach event banner image
        - RSVP functionality (for PTM)
        - Event reminders
        - View upcoming events in calendar

    - **Announcements:**

        - Emergency announcements (push notification + in-app banner)
        - School closure, exam postponement, etc.

Technical Tasks:

- Implement react-native-maps for transport tracking
- Build polling mechanism for live location updates (30-second interval)
- Create file upload component with progress indicator
- Implement PDF viewer for homework and notices
- Build infinite scroll for library catalog
- Create calendar component for events
- Implement image picker with multi-select

Files Created:

- `src/screens/Library/BookCatalogScreen.tsx` (Browse books)
- `src/screens/Library/BookDetailScreen.tsx` (Book info)
- `src/screens/Library/IssueBookScreen.tsx` (Librarian)
- `src/screens/Library/ReturnBookScreen.tsx` (Librarian)
- `src/screens/Library/MyBooksScreen.tsx` (Student - issued books)

- `src/screens/Library/BookReservationScreen.tsx` (Student)
- `src/screens/Library/OverdueTrackerScreen.tsx` (Librarian)
- `src/screens/Student/HomeworkScreen.tsx` (Step 13 - view assignments)
- `src/screens/Student/HomeworkSubmissionScreen.tsx` (Upload solution)
- `src/screens/Teacher/CreateHomeworkScreen.tsx` (Assign homework)
- `src/screens/Teacher/GradeHomeworkScreen.tsx` (Review submissions)
- `src/screens/Transport/BusTrackingScreen.tsx` (Parent - live map)
- `src/screens/Transport/MyRouteScreen.tsx` (Student/Parent - route details)
- `src/screens/Transport/VehicleManagementScreen.tsx` (Transport Manager)
- `src/screens/Transport/RouteManagementScreen.tsx` (Transport Manager)
- `src/screens/Communication/NoticesScreen.tsx` (View notices)
- `src/screens/Communication/CreateNoticeScreen.tsx` (Admin/Teacher)
- `src/screens/Communication/EventsScreen.tsx` (View events)
- `src/screens/Communication/CreateEventScreen.tsx` (Admin)
- `src/components/features/library/BookCard.tsx`
- `src/components/features/library/IssuedBookItem.tsx`
- `src/components/features/transport/RouteMap.tsx`
- `src/components/features/transport/StopMarker.tsx`
- `src/components/features/communication/NoticeCard.tsx`
- `src/components/features/communication/EventCard.tsx`

Files Modified:

- `src/store/slices/librarySlice.ts` (Enhance with reservations, overdue tracking)
- `src/store/slices/transportSlice.ts` (Add live tracking state, route cache)
- `src/services/api/libraryService.ts` (Add reservation, overdue fine endpoints)
- `src/services/api/transportService.ts` (Add live location endpoint)
- `src/services/api/communicationService.ts` (Add notice CRUD, event CRUD)
- **NEW:** `src/services/api/homeworkService.ts` (Homework assignment and submission)

Phase 8: Platform Admin & Tenant Management (Weeks 26-27)

Goal: Complete Super Admin features for multi-tenancy and platform oversight (Steps 1-3)

Deliverables:

1. **Super Admin Dashboard** (Step 1 - Global Oversight)

   - Platform-wide statistics:
     - Total schools (tenants)
     - Total active users
     - Total students across all schools
     - Monthly revenue (subscription fees)
   - Recent tenant activities
   - System health indicators (API uptime, database status)
   - Quick actions: Add new school, view logs, manage subscriptions

2. **Tenant Setup Wizard** (Step 2 - New School Onboarding)

- **Step 1: School Basic Info**
    - School name, code, address, contact details
    - Subdomain selection (e.g., `hillview.schoolmgmt.com`)
    - Custom domain option
    - School logo upload
- **Step 2: Subscription Plan**
    - Select plan (Basic, Standard, Premium)
    - Features per plan: Max students, modules enabled
    - Billing cycle (Monthly, Quarterly, Annual)
    - Trial period option
- **Step 3: Admin Account**
    - Create principal/admin account
    - Email, phone, password
    - Send welcome email with credentials
- **Step 4: Module Configuration**
    - Enable/Disable modules: Library, Transport, Online Payments, Biometric Attendance
    - Set preferences: Academic board, grading system, fee structure template
- **Step 5: Deployment**
    - Trigger backend schema creation
    - Seed initial data (sample classes, subjects)
    - Activate tenant
    - Show deployment progress

3. **Deployment Success Tracking** (Step 3)
    - Tenant status dashboard
    - Setup completion checklist:
        - Schema created ✓
        - Admin account activated ✓
        - Classes configured ✓
        - First student enrolled ✓
    - Health checks: Database connectivity, API response time
    - Quick fixes for failed deployments
4. **Tenant Management** (Super Admin)
    - List all tenants with filters (active, trial, expired)
    - View tenant details and usage statistics
    - Suspend/Resume tenant
    - Delete tenant (with confirmation and data export)
    - Upgrade/Downgrade subscription
    - Extend trial period

5. **System Audit Logs** (Step 15 - Super Admin & School Admin)
   - **Super Admin Logs:** Platform-level events
     - Tenant creation, deletion, suspension
     - Super admin login attempts
     - Schema migrations
   - **School Admin Logs:** School-level events
     - User login/logout
     - Student admission, transfer, deletion
     - Fee payment, marks entry, attendance marking
     - Settings changes
   - **Filters:** User, action type, date range, severity
   - **Export:** Download logs as CSV

Technical Tasks:

- Implement multi-tenant aware API calls (Super Admin can switch context)
- Build tenant provisioning backend integration
- Create deployment progress tracker
- Implement audit log collection and storage
- Build filterable log viewer with pagination
- Add data export functionality

Files Created:

- `src/screens/SuperAdmin/PlatformDashboardScreen.tsx` (Step 1)
- `src/screens/SuperAdmin/TenantSetupWizardScreen.tsx` (Step 2)
- `src/screens/SuperAdmin/DeploymentTrackerScreen.tsx` (Step 3)
- `src/screens/SuperAdmin/TenantManagementScreen.tsx`
- `src/screens/SuperAdmin/TenantDetailScreen.tsx`
- `src/screens/Admin/AuditLogsScreen.tsx` (Step 15)
- `src/components/features/admin/TenantCard.tsx`
- `src/components/features/admin/DeploymentProgress.tsx`
- `src/components/features/admin/AuditLogItem.tsx`

Files Modified:

- `src/services/api/tenantService.ts` (Add tenant CRUD, deployment endpoints)
- `src/services/api/auditService.ts` (NEW - Log fetching and filtering)

Phase 9: Polish, Testing & Deployment Prep (Weeks 28-30)

Goal: Finalize app quality, fix bugs, optimize performance, prepare for app store submission

Deliverables:

1. **UI/UX Polish**

- Consistent loading states (skeleton screens)
- Empty states with illustrations and call-to-action
- Error states with retry buttons
- Success animations (Lottie animations for key actions)
- Haptic feedback on important actions
- Smooth screen transitions (shared element transitions)
- Dark mode support (extend theme system)

2. **Performance Optimization**
    - Image optimization (compress, lazy load, cache)
    - List rendering optimization (FlashList for all long lists)
    - Code splitting and lazy loading for heavy screens
    - Bundle size analysis and reduction (remove unused dependencies)
    - Memory leak detection and fixes
    - API response caching strategy refinement
    - Reduce app startup time (❤️ seconds)
3. **Accessibility Enhancements**
    - Screen reader support (label all interactive elements)
    - Touch target sizes (min 44x44 points)
    - Color contrast compliance (WCAG AA)
    - Dynamic font sizing support
    - Voice-over testing on iOS
    - TalkBack testing on Android
4. **Security Hardening**
    - Implement certificate pinning for API calls
    - Enable code obfuscation (ProGuard for Android, SwiftShield for iOS)
    - Secure sensitive data storage (encrypt with MMKV)
    - Prevent screenshots in sensitive screens (payment, student ID)
    - Input validation and sanitization
    - Penetration testing
5. **Testing**
    - Unit tests for critical business logic (Redux actions, utilities)
    - Integration tests for API services (mock backend responses)
    - E2E tests for critical flows (Detox):
        - Login → Dashboard → Mark Attendance
        - Login → Fee Payment → Success
        - Login → View Results
    - Performance testing (measure FPS, memory usage)
    - Device compatibility testing (Android 7+, iOS 13+)
6. **App Store Preparation**
    - App icon design (Android adaptive icon + iOS icon)

    - Screenshots for all screen sizes (iPhone, iPad, Android phones, tablets)
    - App description and keywords
    - Privacy policy and terms of service
    - App Store Connect setup (iOS)
    - Google Play Console setup (Android)
    - Beta testing (TestFlight for iOS, Internal Testing for Android)

7. **Documentation**
    - User manual (PDF) for each role
    - Admin guide (school setup, tenant management)
    - API integration guide (if needed for custom integrations)
    - Troubleshooting guide
    - Release notes

Technical Tasks:

- Run `npm audit` and fix security vulnerabilities
- Optimize bundle size using metro bundler config
- Set up Sentry or Firebase Crashlytics for crash reporting
- Configure Firebase Analytics for usage tracking
- Create production build configurations (.env.production)
- Set up CI/CD pipeline (GitHub Actions, Bitrise, or Codemagic)
- Generate signed APK/AAB (Android) and IPA (iOS)

Files Modified:

- All screens (add loading skeletons, empty states, error handling)
- `android/app/build.gradle` (ProGuard configuration)
- `ios/Podfile` (production dependencies)
- `app.json` (version bump, app metadata)
- `package.json` (remove unused dependencies)

Critical Files to Modify

Navigation (High Priority)

File: `src/navigation/MainNavigator.tsx` (141 lines, currently flat bottom tabs)

- **Current State:** Simple bottom tab navigator with role-based tab visibility
- **Required Changes:** Convert each tab to a Stack Navigator for drill-down flows
- **Example:**
    - Attendance tab → AttendanceStack → [AttendanceList, MarkAttendance, AttendanceDetail]
    - Exams tab → ExamStack → [ExamList, MarksEntry, ResultView]
- **Implementation Pattern:** Each Stack Navigator should be defined separately and mounted as tab screen

File: `src/navigation/AppNavigator.tsx` (needs creation or enhancement)

- Add deep linking configuration

- Configure notification-based navigation
- Implement protected route logic for role-specific screens

API Client (Minor Enhancements Needed)

File: `src/services/api/client.ts` (267 lines, already robust)

- **Current State:** ✅ Excellent implementation with:
  - JWT token refresh with queue management (lines 57-105)
  - Multi-tenant header injection (lines 35-39)
  - Error handling with status code mapping (lines 134-182)
- **Needed Enhancements:**
  - Add request retry logic for network failures (3 retries with exponential backoff)
  - Implement request deduplication for concurrent identical requests
  - Add upload progress callback support (already has basic structure at line 211)

Dashboard Screens (Medium Priority)

File: `src/screens/Dashboard/TeacherDashboard.tsx` (369 lines)

- **Current State:**
  - UI structure complete with stat cards and quick actions
  - Mock data fallback (lines 89-96)
  - API integration partially implemented (lines 43-97)
- **Required Changes:**
  - Replace Alert.alert calls with actual navigation (lines 220, 227, 234, 241)
  - Implement proper teacher-specific API endpoints (currently using generic class queries)
  - Add classroom activation flow (fetch assigned classes)
  - Fix attendance pending calculation logic (line 82 has mock logic)

File: `src/screens/Dashboard/ParentDashboard.tsx` (needs enhancement)

- Implement multi-child selector with horizontal scroll chips
- Add fee payment CTA button
- Integrate transport tracking link

Redux Slices (enhance existing 10 slices)

- `src/store/slices/authSlice.ts` - Add biometric auth state, email verification status
- `src/store/slices/studentSlice.ts` - Add admission workflow actions, multi-child management for parents
- `src/store/slices/attendanceSlice.ts` - Add QR check-in state, offline queue for pending sync
- `src/store/slices/examSlice.ts` - Add marks entry draft state, grade calculation cache
- `src/store/slices/feeSlice.ts` - Add payment gateway state, receipt cache
- `src/store/slices/notificationSlice.ts` - Add FCM token management, notification preferences
- **NEW:** `src/store/slices/uiSlice.ts` - Global loading states, modal management, toast notifications
- **NEW:** `src/store/slices/syncSlice.ts` - Offline sync queue, retry logic state

API Services (enhance existing 11 services)

- `src/services/api/authService.ts` - Add email verification, biometric credential storage
- `src/services/api/studentService.ts` - Add admission workflow, approval, transfer methods
- `src/services/api/attendanceService.ts` - Add QR check-in endpoint, bulk mark endpoint
- `src/services/api/examService.ts` - Add marks entry batch endpoint, result publishing
- `src/services/api/feeService.ts` - Add payment order creation, verification
- `src/services/api/academicService.ts` - Add teacher's assigned classes endpoint
- `src/services/api/notificationService.ts` - Add FCM token registration, preferences update

New Services (Create from scratch)

- `src/services/biometric.service.ts` - Biometric authentication wrapper (react-native-biometrics)
- `src/services/payment.service.ts` - Razorpay integration wrapper
- `src/services/firebase.service.ts` - FCM setup, notification handlers, deep link routing
- `src/services/qr.service.ts` - QR code generation (student ID), scanning logic with signature verification
- `src/services/sync.service.ts` - Offline queue management, retry logic, conflict resolution

New Screens (Priority by Phase)

Phase 1 (Teacher Flow):

- `src/screens/Attendance/MarkAttendanceScreen.tsx` - Bulk grid + individual modes
- `src/screens/Exam/MarksEntryScreen.tsx` - Spreadsheet-like grid view
- `src/screens/Academic/TimetableScreen.tsx` - Weekly schedule view
- `src/screens/Teacher/ClassroomActivationScreen.tsx` - Assigned classes with student roster

Phase 2 (Student/Parent Flow):

- `src/screens/Student/DigitalIDCardScreen.tsx` - QR code display with biometric unlock
- `src/screens/Student/HomeworkSubmissionScreen.tsx` - File upload interface
- `src/screens/Parent/ParentOnboardingScreen.tsx` - First-launch walkthrough
- `src/screens/Fee/PaymentFlowScreen.tsx` - Razorpay integration
- `src/screens/Fee/PaymentHistoryScreen.tsx` - Receipt viewer with download

Phase 3 (Admin Flow):

- `src/screens/Admin/StudentAdmissionFormScreen.tsx` - Multi-step form with validation
- `src/screens/Admin/PrincipalSetupScreen.tsx` - Academic board configuration
- `src/screens/Admin/AdminOperationsHubScreen.tsx` - Approval workflows dashboard

Phase 4 (Advanced Features):

- `src/screens/Transport/BusTrackingScreen.tsx` - Live map with react-native-maps
- `src/screens/Support/HelpCenterScreen.tsx` - FAQ accordion
- `src/screens/Admin/AuditLogsScreen.tsx` - Filterable activity log

Configuration Files

- `src/config/payment.config.ts` - Razorpay API keys (dev/prod)
- `src/config/firebase.config.ts` - Firebase project configuration

- `android/app/google-services.json` - Firebase Android config (from Firebase Console)
- `ios/GoogleService-Info.plist` - Firebase iOS config (from Firebase Console)
- `.env` files - Environment-specific variables (API_URL, RAZORPAY_KEY, etc.)

Platform-Specific Implementations

Android

- Permissions: CAMERA (QR), BIOMETRIC, LOCATION (transport tracking)
- Firebase Cloud Messaging configuration
- Razorpay SDK integration
- ProGuard rules for production builds

iOS

- Info.plist permissions (Camera, FaceID, Location)
- APNs certificate configuration
- App Transport Security exceptions for development
- Keychain storage for biometric credentials

Testing Strategy

Unit Testing

- Redux actions/reducers
- API service methods
- Utility functions
- Form validation

Integration Testing

- Authentication flows
- API integration points
- Payment gateway flows
- Navigation flows

E2E Testing (Detox)

- Critical user journeys per role
- Offline/online sync
- Push notification handling

Verification & Quality Assurance

Functional Testing Checklist

1. **Authentication:** Login, logout, token refresh, tenant selection
2. **Student Management:** Admission, enrollment, document upload, profile viewing
3. **Attendance:** Mark attendance, leave requests, QR check-in, summaries
4. **Examinations:** View schedule, enter marks, generate results

5. **Fees:** View fees, make payments, download invoices
6. **Library:** Browse books, issue/return, reservations
7. **Transport:** View routes, track vehicles, allocation
8. **Communication:** View notices/events, receive notifications
9. **Profile:** Update profile, change password, settings

Performance Testing

- API response times (<2s for critical operations)
- List rendering performance (1000+ items with FlashList)
- Image loading optimization
- Offline mode sync speed
- App startup time (<3s)

Security Testing

- Token storage security (MMKV encryption)
- Biometric authentication
- Network request encryption (HTTPS only)
- Input validation
- SQL injection prevention (parameterized queries)
- XSS prevention in web views

Accessibility Testing

- Screen reader support
- Touch target sizes (min 44x44 points)
- Color contrast ratios (WCAG AA)
- Dynamic font sizing

Deployment Strategy

Environment Configuration

- **Development:** localhost backend, debug builds
- **Staging:** Staging API, TestFlight/Internal Testing builds
- **Production:** Production API, App Store/Play Store builds

Release Process

1. Version bump (semantic versioning)
2. Changelog generation
3. Build generation (Android: AAB, iOS: IPA)
4. Internal testing
5. Beta testing (TestFlight, Internal Testing)
6. Production release
7. Monitoring and hotfix readiness

App Store Requirements

- **Android:** Minimum SDK 24 (Android 7.0), Target SDK 34
- **iOS:** iOS 13.0+, Xcode 15+
- Privacy policy URL
- App descriptions and screenshots
- COPPA compliance (children's data protection)

Maintenance & Future Enhancements

Post-Launch Monitoring

- Crash reporting (Sentry/Firebase Crashlytics)
- Analytics (Firebase Analytics)
- User feedback collection
- Performance monitoring

Future Feature Roadmap

- Offline-first architecture with sync queue
- Real-time chat between teachers/parents
- Video calling integration
- AI-powered attendance via facial recognition
- Homework submission with file attachments
- Online examination module
- Parent-teacher meeting scheduling
- School bus live tracking with ETA
- Multilingual support (Hindi, regional languages)
- Dark mode implementation

Risk Mitigation

Technical Risks

- **Payment gateway failures:** Implement retry mechanism, manual reconciliation
- **Offline sync conflicts:** Last-write-wins with conflict resolution UI
- **Token expiration during operation:** Queue failed requests, retry after refresh
- **Large data sets:** Implement pagination, lazy loading, virtualized lists
- **Device fragmentation:** Test on multiple Android/iOS versions, use React Native defaults

Operational Risks

- **API downtime:** Implement circuit breaker pattern, graceful degradation
- **Database schema changes:** Version API endpoints, backward compatibility
- **Breaking changes in dependencies:** Lock dependency versions, test upgrades in staging

Success Metrics

Development KPIs

- Code coverage: >70%

- API response time: <2s (95th percentile)
- Crash-free sessions: >99%
- App size: <50MB

User Experience KPIs

- Daily active users: 60% of enrolled students/parents
- Session duration: >5 minutes
- Feature adoption: 80% of users using attendance/fees modules
- App rating: >4.2/5.0

Verification & Testing Strategy

End-to-End Testing Approach

Phase-by-Phase Verification:

Phase 1 Verification (Dashboards):

- Test login as each user type (9 roles)
- Verify dashboard loads with correct widgets per role
- Test navigation from dashboard quick actions to respective screens
- Verify pull-to-refresh functionality
- Check role-based tab visibility

Phase 2 Verification (Authentication & Notifications):

- Test email verification flow
- Test biometric authentication setup and login
- Test push notifications (foreground, background, killed state)
- Test deep linking from notifications
- Verify notification center shows all notifications
- Test profile photo upload and update

Phase 3 Verification (Academic Setup & Admissions):

- Test student admission form (all 4 steps)
- Test document upload (PDF, images)
- Test admission approval workflow
- Test student enrollment to class
- Test principal's academic setup wizard
- Verify student search and filtering

Phase 4 Verification (Attendance & Timetable):

- Test bulk attendance marking (tap to cycle statuses)
- Test QR-based attendance (generate QR, scan QR)
- Test leave request submission and approval
- Test timetable viewing for all roles
- Verify attendance sync (mark offline, auto-sync when online)

- Test holiday calendar

Phase 5 Verification (Exams & Grading):

- Test exam creation and schedule publishing
- Test marks entry grid (spreadsheet view)
- Test grade calculation and auto-save
- Test result publishing and viewing
- Test report card download (PDF)
- Verify marks validation (max marks check)

Phase 6 Verification (Fee Management):

- Test fee structure assignment
- Test mock payment flow (full, installment, custom amount)
- Test payment success/failure scenarios
- Test receipt generation and download
- Test payment history viewing
- Verify accountant can record offline payments

Phase 7 Verification (Extended Features):

- Test library book issue and return
- Test homework submission (file upload)
- Test transport live tracking (map view)
- Test notice creation and viewing
- Test event creation and RSVP
- Verify ETA calculation for bus tracking

Phase 8 Verification (Platform Admin):

- Test tenant setup wizard (Super Admin)
- Test deployment tracker
- Test audit log viewing and filtering
- Verify multi-tenancy isolation
- Test tenant suspension/reactivation

Phase 9 Verification (Final Polish):

- Performance testing (FPS, memory, app startup time)
- Security testing (token storage, API encryption)
- Accessibility testing (screen reader, touch targets)
- Cross-device testing (Android 7-14, iOS 13-17)
- Beta testing feedback incorporation

Manual Testing Checklist

- [ ] All 9 user types can login successfully
- [ ] Multi-tenant header is sent with all API requests
- [ ] Token refresh works automatically on 401

- [ ] Offline attendance marking queues and syncs when online
- [ ] Push notifications navigate to correct screens
- [ ] QR code student ID works offline
- [ ] Mock payment flow completes successfully
- [ ] File uploads (documents, homework) work within 10MB limit
- [ ] All dashboards load with real data (no mock fallbacks)
- [ ] Deep linking from notifications works
- [ ] Biometric authentication unlocks app
- [ ] Report cards download as PDF
- [ ] Transport map shows live location
- [ ] Dark mode (if implemented) works across all screens
- [ ] App handles network errors gracefully (offline mode)
- [ ] App size is under 50MB (excluding assets)

Automated Testing

Unit Tests (Jest + React Native Testing Library):

```
npm test
```

- Test Redux actions and reducers
- Test utility functions (validation, formatting)
- Test component rendering and interactions

Integration Tests:

```
npm run test:integration
```

- Mock API responses with axios-mock-adapter
- Test complete flows (login → dashboard → feature)

E2E Tests (Detox):

```
detox test --configuration ios.sim.debug
detox test --configuration android.emu.debug
```

- Test critical user journeys
- Test offline/online transitions
- Test notification handling

Performance Benchmarks

Target Metrics:

- App startup time: ❤️ seconds (cold start)
- Screen transition: <300ms
- API response handling: <500ms to display data
- List rendering (1000 items): 60 FPS with FlashList
- Memory usage: <150MB (iOS), <200MB (Android)
- APK/IPA size: <50MB (without large assets)

Measurement Tools:

- React Native Performance Monitor (FPS, JS thread load)
- Xcode Instruments (iOS memory profiling)
- Android Studio Profiler (Android memory, CPU)
- Flipper Network Plugin (API call inspection)

Timeline Summary

Comprehensive Development Timeline (30 weeks):

Phase	Duration	Weeks	Deliverables
Phase 1	4 weeks	1-4	All 9 role-based dashboards with real API integration
Phase 2	3 weeks	5-7	Authentication, notifications, profile management, help center
Phase 3	4 weeks	8-11	Student admission, academic setup, enrollment, search
Phase 4	4 weeks	12-15	Attendance marking (3 modes), QR ID, timetable, leave management
Phase 5	3 weeks	16-18	Exam management, marks entry, grading, report cards
Phase 6	3 weeks	19-21	Fee structure, mock payments, receipts, financial reports
Phase 7	4 weeks	22-25	Library, homework submission, transport tracking, communication
Phase 8	2 weeks	26-27	Tenant management, deployment wizard, audit logs
Phase 9	3 weeks	28-30	UI polish, performance optimization, testing, app store prep
Total	**30 weeks**		**All 15 Golden Thread steps + Advanced features**

Post-Launch (Weeks 31+):

- Monitor crash reports and user feedback
- Hotfix deployment as needed
- Plan next iteration features (AI attendance, video calling, etc.)

Key Implementation Recommendations

Architecture Best Practices

1. **Navigation Structure:**

- Use nested Stack Navigators within each bottom tab
- Implement deep linking for all major screens
- Type-safe navigation with TypeScript

2. **State Management:**
   - Persist only auth, user, and tenant data (Redux Persist)
   - Use online-first approach with cached fallbacks
   - Implement graceful degradation on network errors
3. **API Integration:**
   - Leverage existing robust API client (client.ts is production-ready)
   - Add request retry logic for transient failures
   - Implement request deduplication for concurrent calls
4. **Performance:**
   - Use FlashList for all lists with 50+ items
   - Implement image caching with FastImage
   - Lazy load heavy screens
   - Optimize bundle size (code splitting)
5. **Security:**
   - Store tokens in encrypted storage (MMKV)
   - Use HTTPS only (enforce in API client)
   - Implement certificate pinning for production
   - Validate all user inputs on both client and server
6. **User Experience:**
   - Show loading skeletons (avoid blank screens)
   - Implement optimistic updates for quick feedback
   - Add haptic feedback for key actions
   - Use consistent spacing and colors from constants

Development Workflow

1. **Branch Strategy:**
   - `main` - production-ready code
   - `develop` - integration branch
   - `feature/*` - individual features
   - `hotfix/*` - urgent production fixes
2. **Code Review Process:**
   - All changes require PR review
   - Run linter (ESLint) and formatter (Prettier) before commit
   - Pre-commit hooks with Husky
   - CI/CD pipeline runs tests on every PR

3. **Release Process:**
    - Semantic versioning (v1.0.0, v1.1.0, v2.0.0)
    - Changelog generation from commit messages
    - Beta testing via TestFlight/Internal Testing
    - Staged rollout (10% → 50% → 100%)

Risk Management

Risk	**Impact**	**Mitigation**
Backend API changes breaking mobile	High	Version API endpoints (/api/v1/, /api/v2/), maintain backward compatibility
Mock payment not realistic enough	Medium	Use actual test mode of Razorpay (sandbox) instead of fully mocked flow
Online-only approach fails in low-connectivity schools	High	Implement selective offline (attendance, homework viewing) with sync queue
30-week timeline too long for stakeholders	Medium	Deliver incremental releases (Phase 1 MVP in 4 weeks, Phase 4 core features in 15 weeks)
App store rejection	High	Follow platform guidelines strictly, thorough testing, privacy policy
Performance issues on older devices	Medium	Set minimum SDK versions (Android 7+, iOS 13+), test on low-end devices

---

Success Criteria

Technical Metrics

- ✅ All 15 Golden Thread steps implemented
- ✅ Code coverage >70%
- ✅ Crash-free sessions >99%
- ✅ API response time <2s (95th percentile)
- ✅ App startup time <3s
- ✅ App size <50MB

User Experience Metrics

- ✅ All 9 user roles can complete their primary workflows
- ✅ Multi-child management works seamlessly for parents
- ✅ Teachers can mark attendance in <2 minutes
- ✅ Fee payment completes in <5 steps
- ✅ QR student ID works offline
- ✅ Push notifications deliver within 10 seconds

Business Metrics

- [x] App ready for app store submission
- [x] Beta testing with 3+ schools completed
- [x] User documentation completed for all roles
- [x] Admin can onboard a new school in <15 minutes

Next Steps for Implementation

Immediate Actions (Week 1)

1. **Environment Setup:**

   ```
   cd mobile-app
   npm install
   npx pod-install  # iOS only
   ```

2. **Configuration:**

   - Create `.env.development`, `.env.staging`, `.env.production` files
   - Set `API_BASE_URL`, `RAZORPAY_KEY_ID` (test mode), `FIREBASE_CONFIG`
   - Configure Firebase project and download `google-services.json` (Android) and `GoogleService-Info.plist` (iOS)

3. **Backend Coordination:**

   - Confirm API endpoint availability for all modules
   - Request sample API responses for dashboard endpoints
   - Test multi-tenancy header (`X-Tenant-Subdomain`) requirement
   - Verify JWT token refresh mechanism

4. **Start Development:**

   - **Phase 1, Week 1:** Enhance TeacherDashboard.tsx (replace mocks, add navigation)
   - Convert MainNavigator.tsx tabs to nested stacks
   - Create SuperAdminDashboard.tsx
   - Test navigation flows

Developer Onboarding

Required Skills:

- React Native (hooks, navigation, state management)
- TypeScript
- Redux Toolkit
- REST API integration
- Mobile UI/UX best practices

Recommended Team:

- 2-3 React Native developers

- 1 Backend developer (API support)
- 1 QA engineer
- 1 UI/UX designer (for mockups and assets)

This comprehensive plan provides a roadmap for building a production-ready School Management System mobile application that integrates seamlessly with the existing backend and delivers all 15 Golden Thread steps across 9 user roles over 30 weeks.