COMPREHENSIVE LOCAL TESTING PLAN

School Management System Mobile Application

Executive Summary

This testing plan provides a robust strategy for locally testing the React Native mobile application with the Django backend using Android Studio emulator. The plan covers environment setup, compatibility testing with the existing web application, and comprehensive functional testing across all 9 user roles.

Testing Environment Overview

Technology Stack Compatibility

Mobile Application:

- React Native 0.73.11 with TypeScript
- Redux Toolkit + Redux Persist
- Axios HTTP client with JWT interceptors
- Multi-tenant header support (`X-Tenant-Subdomain`)

Backend Application:

- Django 4.2.7 + Django REST Framework 3.14.0
- PostgreSQL with schema-per-tenant architecture
- JWT Authentication (SimpleJWT)
- OpenAPI/Swagger documentation
- **Two Mock Tenants Available:**
  - Demo High School (subdomain: `demo`, schema: `school_demo`)
  - Veda Vidyalaya (subdomain: `veda`, schema: `tenant_veda_vidyalaya`)

Compatibility Assessment: ✅ Fully Compatible

- Both use JWT authentication with access/refresh tokens
- Mobile app configured to send `X-Tenant-Subdomain` header
- Backend accepts multi-tenant header and routes to correct schema
- API endpoints match mobile service layer expectations
- CORS configured to accept localhost requests

Phase 1: Environment Setup (Day 1)

1.1 Backend Setup

Prerequisites:

- PostgreSQL 14+ installed and running
- Python 3.11+ installed

- Redis server installed (optional for Celery tasks)

Setup Steps:

```
# Navigate to backend directory
cd "G:\School Mgmt System\backend"

# Create virtual environment
python -m venv venv

# Activate virtual environment
# Windows:
venv\Scripts\activate

# Install dependencies
pip install -r requirements/base.txt

# Copy environment configuration
copy .env.example .env

# Edit .env file with these settings:
DJANGO_SETTINGS_MODULE=config.settings.development
DEBUG=True
SECRET_KEY=django-insecure-local-testing-key-12345

# Database Configuration
DB_NAME=school_management
DB_USER=postgres
DB_PASSWORD=postgres
DB_HOST=localhost
DB_PORT=5432

# CORS for Mobile Testing
CORS_ALLOW_ALL_ORIGINS=True

# Create PostgreSQL database
createdb school_management

# Run migrations
python manage.py migrate

# Load mock data for demo tenant
python setup_demo_test_accounts.py
python create_demo_student_v2.py

# Start development server
python manage.py runserver 0.0.0.0:8000
```

Verification:

- Open browser: `http://localhost:8000/api/docs/`

- Verify Swagger UI loads with all endpoints
- Test login endpoint with:

  ```
  POST /api/v1/auth/login/
  Headers: {"X-Tenant-Subdomain": "demo"}
  Body: {"email": "teacher@demo.com", "password": "School@123"}
  ```

- Verify you receive access and refresh tokens

Available Test Accounts (Demo Tenant):

```
Teacher: teacher@demo.com / School@123
Students: std.1@demo.school to std.100@demo.school / School@123
Admin: admin@demo.com / School@123
```

1.2 Android Studio Setup

Prerequisites:

- Android Studio Hedgehog (2023.1.1) or later
- Java JDK 17+
- Android SDK Platform 34
- Android SDK Build-Tools 34.0.0
- Android Emulator

Installation Steps:

1. **Download and Install Android Studio:**
   - Download from: https://developer.android.com/studio
   - Install with default settings
   - Install Android SDK Platform 34 via SDK Manager
2. **Configure Environment Variables:**

   ```
   # Windows - Add to System Environment Variables:
   ANDROID_HOME=C:\Users\<YourUsername>\AppData\Local\Android\Sdk

   # Add to PATH:
   %ANDROID_HOME%\platform-tools
   %ANDROID_HOME%\emulator
   %ANDROID_HOME%\tools
   %ANDROID_HOME%\tools\bin

   # Verify installation:
   adb --version
   ```

3. **Create Android Virtual Device (AVD):**
    - Open Android Studio → Tools → Device Manager
    - Click "Create Device"
    - Select: Pixel 5 (or Pixel 7) - 6.0" display
    - System Image: Android 14.0 (API 34) - Download if needed
    - AVD Name: `Pixel_5_API_34`
    - Advanced Settings:
        - RAM: 2048 MB (minimum)
        - Internal Storage: 2048 MB
        - Enable Hardware Keyboard
        - Graphics: Hardware - GLES 2.0
    - Click Finish
4. **Start Emulator:**

```
# List available AVDs
emulator -list-avds

# Start emulator (from command line)
emulator -avd Pixel_5_API_34

# Or start from Android Studio Device Manager
```

Emulator Network Configuration:

- Emulator uses `10.0.2.2` to access host's `localhost`
- This is already configured in `.env` file: `API_BASE_URL=http://10.0.2.2:8000/api/v1`

1.3 Mobile App Setup

Prerequisites:

- Node.js 18+ installed
- npm or yarn package manager

Setup Steps:

```
# Navigate to mobile app directory
cd "G:\School Mgmt System\mobile-app"

# Install dependencies
npm install

# Verify .env.development configuration
# File: G:\School Mgmt System\mobile-app\.env.development
# Should contain:
API_BASE_URL=http://10.0.2.2:8000/api/v1  # For Android emulator
API_TIMEOUT=30000
ENABLE_DEBUG=true
LOG_LEVEL=debug
USE_MOCK_PAYMENT=true

# Start Metro bundler
npm start

# In a new terminal, build and run Android app
npm run android

# This will:
# 1. Build the APK
# 2. Install on running emulator
# 3. Launch the app
```

Troubleshooting:

If build fails with "SDK not found":

```
# Create local.properties in android folder
cd android
echo "sdk.dir=C:\\Users\\<YourUsername>\\AppData\\Local\\Android\\Sdk" >
local.properties
cd ..
npm run android
```

If Metro bundler cache issues:

```
npm start -- --reset-cache
```

If emulator connection issues:

```
adb devices  # Should show emulator
adb reverse tcp:8000 tcp:8000  # Forward port to access localhost directly
# Then use API_BASE_URL=http://localhost:8000/api/v1 in .env
```

1.4 Verify End-to-End Connection

Test Connectivity:

1. **Backend Running Check:**

  - Browser: `http://localhost:8000/api/docs/`
  - Should see Swagger UI

2. **Emulator Network Check:**
   - Open Chrome browser in emulator
   - Navigate to: `http://10.0.2.2:8000/api/docs/`
   - Should load Swagger UI (confirms network connectivity)

3. **Mobile App Login Test:**
   - Launch mobile app on emulator
   - Select tenant: "Demo High School" (subdomain: demo)
   - Enter credentials: `teacher@demo.com` / `School@123`
   - Should successfully login and see Teacher Dashboard

4. **Verify API Calls:**
   - In terminal running backend, you should see API requests:

```
[16/Jan/2024 10:30:15] "POST /api/v1/auth/login/ HTTP/1.1" 200 1234
[16/Jan/2024 10:30:16] "GET /api/v1/auth/users/me/ HTTP/1.1" 200 567
```

Phase 2: Backend Compatibility Testing (Day 2)

2.1 Multi-Tenancy Testing

TC-MT-01: Tenant Selection

```
Steps:
1. Launch app
2. View tenant selection screen
3. Select "Demo High School"
4. Verify subdomain "demo" is stored in Redux
5. Login as teacher@demo.com
6. Verify X-Tenant-Subdomain header is sent with all API requests

Expected:
- Backend receives X-Tenant-Subdomain: demo header
- Queries execute against school_demo schema
- Only demo school data returned

Verification:
- Check backend terminal logs for header
- Check Redux state: state.tenant.currentTenant.subdomain === "demo"
```

2.2 Authentication Flow Testing

TC-AUTH-01: Login Success

```
Steps:
1. Select Demo tenant
2. Enter: teacher@demo.com / School@123
3. Tap "Login"

Expected:
- API POST /api/v1/auth/login/ returns 200
- Response contains access, refresh tokens
- User object contains:
  - user_type: TEACHER
  - permissions array
  - roles array
  - staff_id populated
- Redux auth slice stores tokens
- Navigation to TeacherDashboard

Verification:
Backend response:
{
  "access": "eyJ0eXAiOiJKV1Q...",
  "refresh": "eyJ0eXAiOiJKV1Q...",
  "user": {
    "user_type": "TEACHER",
    "email": "teacher@demo.com",
    "permissions": ["students.view", "attendance.mark", ...]
  }
}
```

TC-AUTH-03: Token Refresh

```
Steps:
1. Login successfully
2. Wait for access token to expire (15 minutes default)
3. Make any API call (navigate to Students screen)

Expected:
- Initial API call fails with 401
- Axios interceptor catches 401
- Automatically calls POST /api/v1/auth/refresh/ with refresh token
- Gets new access token
- Retries original API call with new token
- Students screen loads successfully
```

Phase 3: Functional Testing by User Role (Days 3-5)

3.1 Teacher Role Testing

Login: teacher@demo.com / School@123

TC-TEACHER-03: Mark Attendance - Bulk Mode

```
Steps:
1. Navigate to Attendance → Mark Attendance
2. Select class: 10A
3. Select date: Today
4. View student grid (all default to PRESENT)
5. Tap student cards to cycle: Present → Absent → Late → Leave
6. Set 2 students as Absent, 1 as Late
7. Tap "Submit Attendance"
8. Confirm in dialog

Expected:
- Grid shows all 40 students with photos
- Status cycles with color coding:
  - Green (Present)
  - Red (Absent)
  - Orange (Late)
  - Gray (Leave)
- Summary shows: 37 Present, 2 Absent, 1 Late
- POST request sends attendance array
- Success message shown
- Attendance recorded in backend

Verification:
POST /api/v1/attendance/student-attendance/
Body: {
  "class_id": "uuid",
  "date": "2024-01-16",
  "attendance_records": [
    {"student_id": "uuid1", "status": "PRESENT"},
    {"student_id": "uuid2", "status": "ABSENT"},
    ...
  ]
}
Response: 201 Created
```

3.2 Student Role Testing

Login: std.1@demo.school / School@123

TC-STUDENT-02: Digital ID Card with QR

```
Steps:
1. Tap "View Digital ID" from dashboard
2. View digital ID card

Expected:
```

```
- Shows student photo
- Student details (name, admission number, class, blood group)
- QR code displayed (center of card)
- QR code encodes:
  - Student ID (UUID)
  - Admission number
  - Timestamp (for 24h validity)
  - HMAC signature for security
- Works offline (cached in AsyncStorage)

Verification:
- QR generation uses react-native-qrcode-svg
- Data format:
  {
    "student_id": "uuid",
    "admission_no": "D-1-A-001",
    "timestamp": 1705395600,
    "signature": "hmac_sha256_hash"
  }
```

3.3 Parent Role Testing

Login: parent@demo.com / School@123

TC-PARENT-03: Pay Fees

```
Steps:
1. Navigate to Fees → Pending Payments
2. View fee breakdown (Tuition: ₹15,000, Transport: ₹3,000, Total: ₹18,000)
3. Tap "Pay Now"
4. Select "Pay Full Amount" (₹18,000)
5. Choose payment method: UPI (Mock)
6. Enter UPI ID: test@okaxis
7. Tap "Proceed to Pay"
8. Mock payment success screen

Expected:
- Fee structure shows accurate breakdown
- Payment flow uses mock Razorpay
- Test UPI ID: test@okaxis (success)
- Test UPI ID: fail@okaxis (failure)
- On success:
  - Payment recorded
  - Receipt generated (PDF)
  - Fee balance updates
  - SMS/Email confirmation (backend)
```

3.4 Admin Role Testing

TC-ADMIN-02: Student Admission - Multi-Step Form

```
Steps:
1. Tap "Add Student" from dashboard
2. Fill Step 1: Student Info (Name, DOB, Gender, Blood Group, Photo)
3. Fill Step 2: Guardian Info (Father, Mother details)
4. Fill Step 3: Previous School (School name, TC upload)
5. Fill Step 4: Documents (Birth Certificate, Aadhaar, Photo)
6. Review & Submit
7. Tap "Submit Application"

Expected:
- Form validates each step before allowing next
- Draft saves automatically every 2 minutes
- File uploads show progress (max 5MB per file)
- On submit:
  - Application created with status: PENDING
  - Admin notified
  - Application ID generated: ADM-2024-001
```

3.5 Super Admin Role Testing

TC-SUPERADMIN-02: Tenant Setup Wizard

```
Steps:
1. Tap "Add New School"
2. Step 1: School Info (Name, Code, Subdomain, Logo)
3. Step 2: Subscription Plan (Premium, ₹25,000/month)
4. Step 3: Admin Account (Principal details, auto-generate password)
5. Step 4: Preferences (CBSE, Percentage, Asia/Kolkata)
6. Review & Deploy
7. Tap "Create School"

Expected:
- Deployment progress screen shows:
  - Creating database schema ✓
  - Setting up tables ✓
  - Creating admin account ✓
  - Seeding initial data ✓
  - Activating tenant ✓
- Deployment completes in 30-60 seconds
- Welcome email sent to principal
```

Phase 4: Integration Testing (Day 6)

4.1 End-to-End User Journey: Complete Attendance Flow

```
Actors: Teacher, Student, Parent, Admin
```

```
1. Teacher marks attendance (38 Present, 2 Absent)
2. Student (Ayan - absent) submits leave request with medical certificate
3. Teacher approves leave request
4. System updates Ayan's status: Absent → Leave (Approved)
5. Parent receives notification: "Leave request approved"
6. Admin views attendance report showing correct counts

Verification Points:
- Attendance marked: ✓
- Leave request created: ✓
- Leave approved: ✓
- Attendance status updated: ✓
- Parent notified: ✓
- Report accurate: ✓
```

Phase 5: Performance Testing (Day 7)

5.1 App Performance Metrics

TC-PERF-01: App Startup Time

```
Test:
1. Close app completely (force stop)
2. Clear app cache
3. Launch app
4. Measure time from tap to login screen visible

Target: < 3 seconds (cold start)

Measurement:
- Use React Native Performance Monitor
- Record 5 cold starts, calculate average
```

TC-PERF-03: List Rendering Performance

```
Test:
1. Navigate to Students list (100 students)
2. Scroll quickly through entire list
3. Monitor FPS (frames per second)

Target: 60 FPS (smooth scrolling)

Measurement:
- Enable React Native Performance Monitor (Cmd+D → Show Perf Monitor)
- Observe FPS while scrolling
- JS thread should stay < 80% usage
```

Phase 6: Security Testing (Day 8)

6.1 Authentication Security

TC-SEC-01: Token Storage Security

```
Test:
1. Login as teacher
2. Inspect AsyncStorage/MMKV

Expected:
- Access token stored encrypted
- Refresh token stored encrypted
- Sensitive data (passwords) never stored

Verification:
- Check AsyncStorage keys
- Verify tokens are encrypted (not plain text)
- If using MMKV, verify encryption enabled
```

TC-SEC-03: Session Timeout

```
Test:
1. Login as teacher
2. Keep app open but idle for 15 minutes
3. Attempt to make an API call

Expected:
- Access token expires after 15 minutes
- App automatically refreshes token
- If refresh fails (refresh token expired):
  - Logout user
  - Navigate to login screen
  - Message: "Session expired. Please login again."
```

Phase 7: Cross-Device Compatibility (Day 9)

7.1 Android Version Testing

Test Matrix:

Android Version	API Level	Test Status
Android 7.0 (Nougat)	24	✅ Test required (min supported)
Android 10.0	29	✅ Test required
Android 11.0	30	✅ Test required

Android Version	API Level	Test Status
Android 14.0	34	✅ Primary test device

7.2 Screen Size Testing

Device Type	Screen Size	Resolution	AVD Configuration
Small Phone	4.7"	720x1280	Pixel 2
Medium Phone	6.0"	1080x2340	Pixel 5
Large Phone	6.7"	1440x3200	Pixel 7 Pro
Tablet	10.1"	1920x1200	Pixel Tablet

Phase 8: Accessibility Testing (Day 10)

8.1 Screen Reader Support

TC-ACCESS-01: TalkBack (Android)

```
Test:
1. Enable TalkBack: Settings → Accessibility → TalkBack → ON
2. Launch app
3. Navigate using TalkBack gestures
4. Login as teacher
5. Navigate to Students screen

Expected:
- All interactive elements have labels
- Screen reader announces element purpose
- Forms announce field labels and hints
- Lists announce item counts
- Navigation announces screen changes
```

TC-ACCESS-03: Touch Target Sizes

```
Minimum Size: 44x44 points (iOS) / 48x48dp (Android)

Check:
- Buttons: Should be at least 44x44
- List items: Entire row tappable
- Tab bar icons: Large enough (48x48)
- Form inputs: Adequate height (48+)
```

Phase 9: Error Handling & Edge Cases (Day 11)

9.1 Network Error Scenarios

TC-ERROR-01: No Internet Connection

```
Test:
1. Login as teacher (while online)
2. Navigate to Dashboard
3. Disable network (airplane mode)
4. Attempt to navigate to Students screen

Expected:
- Error screen shown: "No internet connection"
- Offline indicator banner at top
- Option to "Retry" or "View Cached Data"
- No app crash
```

TC-ERROR-03: 500 Server Error

```
Test:
1. Cause backend to return 500 error
2. Make API call from mobile

Expected:
- Error screen: "Something went wrong on our end"
- Technical details hidden from user
- Option to "Retry" or "Contact Support"
- Error logged to crash reporting (Sentry)
```

Phase 10: Regression Testing (Day 12)

10.1 Smoke Test Suite

Quick regression checks after any code change:

```
✓ TC-SMOKE-01: App launches successfully
✓ TC-SMOKE-02: Login works for all user types
✓ TC-SMOKE-03: Dashboard loads for each role
✓ TC-SMOKE-04: Navigation between tabs works
✓ TC-SMOKE-05: API calls return data
✓ TC-SMOKE-06: Logout works
✓ TC-SMOKE-07: No crashes during basic navigation
```

Run Time: ~10 minutes

Test Deliverables

Daily Test Summary Template

```
Date: 16 Jan 2024
Tester: QA Engineer Name
Environment: Android Emulator Pixel 5 API 34
Backend: localhost:8000 (Demo tenant)

Tests Executed: 50
Passed: 45
Failed: 3
Blocked: 2

Failed Tests:
1. TC-TEACHER-05: Marks entry validation
   - Issue: Max marks validation not working
   - Severity: Medium
   - Status: Bug logged
```

Test Environment Checklist

Backend Setup:

- [ ] PostgreSQL installed and running
- [ ] Backend server running on port 8000
- [ ] Demo tenant configured (subdomain: demo)
- [ ] Test accounts created
- [ ] Mock data loaded (100+ students)
- [ ] Swagger UI accessible

Android Studio Setup:

- [ ] Android Studio installed
- [ ] SDK Platform 34 installed
- [ ] AVD created (Pixel 5 API 34)
- [ ] Emulator starts successfully
- [ ] ADB working
- [ ] Environment variables set (ANDROID_HOME)

Mobile App Setup:

- [ ] Node.js 18+ installed
- [ ] Dependencies installed (npm install)
- [ ] .env.development configured
- [ ] Metro bundler starts (npm start)
- [ ] App builds successfully (npm run android)
- [ ] App launches without crashes

Network Configuration:

- [ ] Backend accessible from emulator (10.0.2.2:8000)
- [ ] API calls successful
- [ ] CORS allows emulator origin
- [ ] FCM configured

Success Criteria

Testing is considered successful when:

✅ All critical user journeys work end-to-end ✅ No critical or high-severity bugs remain ✅ App works on Android 7.0+ (API 24+) ✅ Performance targets met (startup < 3s, navigation < 300ms) ✅ No memory leaks detected ✅ Security tests pass (token encryption, HTTPS) ✅ Multi-tenancy isolation verified ✅ Offline mode handles network issues gracefully ✅ Push notifications work in all app states ✅ Accessibility standards met ✅ All 9 user roles can complete their primary tasks ✅ Pass rate > 90% on regression testing ✅ No crashes during 2-hour continuous usage test

Ready for Production When:

- [ ] All test phases completed
- [ ] Test report shows > 95% pass rate
- [ ] No critical bugs open
- [ ] Performance benchmarks met
- [ ] Security audit passed
- [ ] Beta testing completed
- [ ] User acceptance testing (UAT) passed
- [ ] Production environment configured

Next Steps After Testing

1. **Bug Fixes:** Address all critical and high-priority bugs
2. **Performance Optimization:** Optimize slow screens, reduce bundle size
3. **Beta Testing:** Deploy to TestFlight/Internal Testing
4. **Production Preparation:** Configure production backend and keys
5. **Release:** Submit to App Store and Google Play Store

This comprehensive testing plan ensures the School Management System mobile application is thoroughly tested, performant, secure, and ready for production deployment.