SOFTWARE CASE STUDY

MySchoolMobileApp

A React Native School Management Platform

An in-depth look at the architecture, features and technical decisions behind a mobile companion app built for tutors and administrators — covering academics, attendance, exams, results, homework, messaging and virtual classrooms.

Attendance · Exams & Results · Homework · Timetable · Messaging

Prepared as a technical & functional overview of the mobile client

1. School & Academic Structure

Academic Organization Hierarchy

The app exposes a multi-level hierarchy that mirrors the backend's school structure:

- **Class** — Top-level academic grouping. Users can search and browse all classes. Each class contains one or more sections. The app fetches the full classes list for cascading dropdowns throughout the UI.
- **Section** — Sub-division within a class. Sections are loaded dynamically based on the selected class. Display names are normalized (e.g., "Section A" → "A") for clean UI presentation.
- **Subject** — Individual subjects scoped to a tutor. The app fetches subjects assigned to the logged-in tutor. Subjects carry metadata such as subject type (Theory/Practical) and whether they are compulsory.

Supporting Master Data

- **Period** — Configurable time slots that define the school day. The Time Table screen fetches the active period list and renders each period as a row in the weekly grid, with start/end times formatted to HH:MM.
- **Room** — Physical room inventory managed on the backend for timetable slot creation.

2. User & Identity Management

Authentication

- Mobile login endpoint accepts companyName, userName, and password.
- On success, the backend returns a JWT token alongside user profile data (userId, role, tutorId, activeAcademicYearId).
- The token is persisted in AsyncStorage and attached to every outgoing request via Axios interceptors.
- Logout clears all stored credentials and resets the navigation stack to the login screen.

Role-Based Access

- The app is role-aware. The JWT payload carries a role field (e.g., Tutor). Screens and features are scoped accordingly — for example, subjects are fetched via getSubjectsByTutor(tutorId), meaning a tutor only sees their own subjects.
- The 401 interceptor automatically triggers logout if the token becomes invalid or expires, providing a seamless re-authentication flow.

Session Management

- A useRef-based approach holds mutable session data (tutorId, activeAcademicYearId) to avoid stale closures in async callbacks.
- Token initialization runs on app boot via initToken(), reading the persisted token from storage before any API calls are made.

3. Student Management

Student Directory

- Students are fetched by class, section, and optional search key. The endpoint supports server-side pagination with configurable page sizes.
- The list screen renders students in a card-based FlatList with roll number, class, admission date, email, and guardian contact.

A detail modal opens on card press, organizing student data into four sections:

- **Personal Information** — Date of birth, gender, blood group, religion, CNIC
- **Contact Information** — Email, phone, address, post code
- **Academic Information** — Class, section, admission date, transport assignment
- **Guardian Information** — Primary and secondary guardian names, contacts, CNICs, and occupations

Search & Pagination

- Server-side search filters students by name.
- Pagination uses onEndReached with a trailing loadingMore indicator to append results without disrupting the scroll position.

4. Tutor (Teacher) Management

Tutor Profile & Subject Assignment

- The app identifies the tutor through the stored tutorId from the login token.
- The Subjects screen loads only subjects assigned to the current tutor, reinforcing the one-tutor-to-many-subjects relationship.
- Class-subject-tutor assignments are consumed implicitly when fetching exam subject mappings and homework lists.

5. Timetable & Scheduling

Weekly Timetable

The Time Table screen fetches two datasets in parallel using Promise.all:

- **Period list** — All configured time slots with start/end times and break flags.
- **Monthly timetable slots** — Subject assignments for each day/period in the current month.
- Days are extracted dynamically from the API's date array (e.g., "Monday", "Tuesday").
- Break periods are visually distinguished from regular subject slots.
- The grid renders a TimeTable component with time on one axis and days on the other, producing a standard school timetable view.

Special Considerations

- The timetable is scoped by class and section, ensuring tutors only see schedules relevant to their assigned classes.
- A print affordance exists in the header for physical copies.

6. Attendance

Daily Attendance Marking

- Attendance is filtered by class, section, and date using a cascading dropdown system. Changing the class reloads sections; changing the section reloads students.
- Each student row presents three toggle buttons: **P** (Present), **A** (Absent), **L** (Leave).
- Live summary cards display real-time counts for Present, Absent, and Leave.
- "Mark All Present" provides a one-tap bulk action for the entire roster.
- The save action submits the full attendance payload to the backend.
- Monthly attendance summary data is also available via a dedicated endpoint.

Online Class Attendance

The backend tracks manual and automated attendance metrics for virtual classes. The mobile client surfaces this through the attendance module.

7. Examinations

Exam Management

- Exams are loaded as a master list and mapped per class/section through cascading dropdowns.
- Each exam entry expands to reveal subject-level details including schedule, room assignment, and invigilator.
- **Paper status tracking** allows tutors to update the lifecycle state of each exam paper (e.g., Scheduled → In Progress → Completed). The status is updated via a dedicated PUT endpoint that targets the specific exam-subject mapping.
- A **Grade Reference** popup displays the configured grade scale (A+ through F) with color-coded badges and percentage ranges.

Exam Staff Assignment

The backend supports assigning invigilators, collectors, checkers, and re-checkers to exam papers. Duty role types are fetched from master data.

Paper Status Flow	Description
Scheduled	Exam paper is planned and assigned but not yet started.
In Progress	The paper is currently being conducted / under evaluation.
Completed	The paper's lifecycle is finished and results can proceed.

8. Results & Grading

Result Entry & Viewing

- Results are queried by class, section, and exam. The response includes per-student totals and a subject-level breakdown.
- Each result card can be expanded to show obtained marks, total marks, and pass/fail/absent status for every subject.
- Percentage and aggregate marks are computed server-side and returned in the payload.
- **Grade scales** are configurable via the backend (grade name, minimum percentage, maximum percentage), and the app renders them in a reference popup during exam management.

Bulk Operations

The backend exposes a bulk-save endpoint for entry operations, enabling rapid marks entry across an entire class section.

9. Homework

Assignment & Lifecycle

Homework supports full CRUD with the following workflow states:

Step	Workflow State
1	Not Started
2	Assigned
3	Due Today
4	Overdue
5	Submitted
6	Reviewed
7	Graded

- Tutors filter homework by class, section, subject, and date range. Search is supported on the server side.
- Status tabs display counts with badges: All, Assigned, Due Today, Overdue.

Context actions per assignment:

- **Edit** — Navigate to the assignment editor with pre-filled data.
- **Duplicate** — Creates a copy of the assignment via a dedicated endpoint.
- **Delete** — Triggers a confirmation modal before calling the delete API.

Metadata

Each homework card shows class, section, subject, assigned date, due date, and total marks.

10. Communication & Collaboration

Real-Time Messaging

The app fetches two parallel datasets on the messages screen:

- **Inbox** — Recent message threads with online status indicators.
- **Users list** — Available contacts for initiating new conversations.
- Tapping a contact opens the chat screen, which loads the full conversation history and enables sending new messages via SignalR-backed endpoints.
- Search filters both inbox and user lists by display name.

Notifications & Announcements

- Announcements are fetched with search and pagination, allowing tutors to browse school-wide communications.
- The dashboard bell icon navigates directly to the announcements list.

11. Online / Virtual Classes

Scheduling & Attendance

- Tutors can create online classes with class/subject mapping, date/time, duration, platform name, and meeting link.
- All upcoming teacher online classes are fetched and displayed in a dedicated list.
- Attendance tracking for virtual sessions is supported through the attendance module.

12. API Surface

The mobile client communicates with a RESTful backend (inventstarts.com) through a structured services layer. Key endpoint categories observed in the client:

Category	Endpoints
Authentication	/Account/login-user
Master Data	/Class/..., /Section/..., /Subject/..., /Period/..., /Master/...
Student	/Student/get-student-by-search-and-pagination
Attendance	/Attendance/get-student-attendance-by-class, /Attendance/add-and-update-student-attendance, /Attendance/get-student-monthly-attendance
Homework	/Homework/get-homework-list, /Homework/get-homework-status-counts, /Homework/add-homework, /Homework/update-homework, /Homework/duplicate-homework, /Homework/delete-homework
Exam	/Exam/get-exams-list, /Exam/get-exams-subject-mapping-list-by-class, /Exam/update-paper-status
Result	/Result/get-class-results, /Result/get-result-entry-context, /Result/bulk-save-results
Timetable	/TimeTable/get-monthly-timetable-slots, /Period/get-period-detail-list
Message	/Message/inbox, /Message/get-users-list, /Message/chat/{userId}, /Message/send-message
Online Class	/OnlineClass/get-teacher-online-classes, /OnlineClass/add-online-class
Announcement	/Announcement/get-announcement-by-search-and-pagination
Grade	/Grade/get-grades

The client uses a mix of POST (for search/pagination payloads) and GET (for resource lookups) with query string parameters where appropriate.

13. Key Technical Decisions

Decision	Rationale
Axios interceptors for auth	Eliminates boilerplate token attachment on every service call; provides centralized 401 handling.
useRef for session values	Maintains fresh references to async values (tutorId, academicYearId, deleteId) across renders without causing re-subscriptions.
useCallback on fetch functions	Stabilizes function identity for FlatList onEndReached, preventing duplicate network requests.
Promise.all for parallel fetches	Reduces waterfall latency when loading the timetable (periods + slots simultaneously).
FlatList pagination pattern	Handles large student, class, and homework lists efficiently via onEndReached with page/total tracking.
Custom component library (MiniCard, StatsCard, FilterBar, SearchBar)	Avoids heavy third-party UI kits; ensures 100% brand consistency across all screens.
LinearGradient headers	Provides brand-aligned gradient branding on every screen without manual color application.
Service layer isolation	All API logic lives in src/services/. Screens consume only typed function exports, making the UI layer easy to test and refactor.
navigationRef + imperative navigation	Enables programmatic navigation from interceptors (e.g., logout → reset to Login) and anywhere else in the app.
SafeAreaProvider wrapper	Ensures consistent safe-area handling on all devices.

14. Conclusion

MySchoolMobileApp is a feature-complete React Native application that surfaces the core academic operations of a school management system. It follows a clean architectural pattern with a dedicated API layer, reusable UI components, and centralized theming.

The app is built for performance with pagination, parallel data fetching, and efficient state management. It serves as a solid mobile companion for tutors and administrators, with room for future expansion into finance, transport, and analytics modules.

End of Case Study