Service Request Module Documentation

Ray International App

Hijaz Muhammed

31/07/2025

Contents

1. Overview

The Service Request Module is a comprehensive form management system within the Ray International App that enables users to create, edit, view, and manage service requests. Built with Flutter and utilizing GetX for state management, it supports multi-entry forms, file attachments, digital signatures, real-time validation, and offline persistence.

Key Features

- Multi-entry form management with dynamic table entries
- File attachment support for documents and images
- Digital signature integration using signature pad
- Real-time data validation and form state management
- Offline draft saving and data persistence
- CRUD operations (Create, Read, Update, Delete)
- Responsive UI with modern design patterns

2. Architecture

2.1. Directory Structure

```
lib/screens/forms/service_request/
 form/
    service_request_new_form_screen.dart
    service_request_form.dart
    widgets/
       service_request_table.dart
       widget_main_attachment.dart
       widget_attachment.dart
       widget_signature.dart
       widget_save_button.dart
       widget_add_button.dart
    view_models/
        post_form.dart
 list/
    service_request_list_screen.dart
    widgets/
        list_item.dart
        widget_appBar.dart
        widget_appBar_new.dart
 controllers/
     service_request_form_controller.dart
     service_request_list_controller.dart
```

2.2. Design Patterns

- **GetX MVC**: Separation of concerns via Models, Views, Controllers
- **Widget Composition**: Reusable, modular UI components
- **Reactive State**: Observable-driven UI updates
- **RESTful API Integration**: Multipart form data support
- **Local Storage**: Offline draft management

3. Core Components

3.1. ServiceRequestNewFormScreen

File: `form/service`$_r$`equestt_new_form_screen.dart`

Orchestrates the creation and editing of service requests, managing table entries, attachments, signatures, and drafts.

Key Methods

```
void updateTableEntry(Map formData, int index);
void preserveFormState();
void loadDraft(ServiceRequestSummaryModel draft);
Future trySave();
```

3.2. ServiceRequestFormController

File: `controllers/service`$_r$`equest_f orm_c ontroller.dartart`

Manages form state, data fetching, validation, attachments, signatures, and API calls.

Key Properties

```
TextEditingController companyCodeController;
RxList companies = [].obs;
RxBool isLoadingCcompanies = false.obs;
```

3.3. ServiceRequestTable Widget

File: `widgets/service_request_table.dart`

Renders a scrollable table with edit/delete actions, attachment previews, and signature display. Columns include Company Code, Project, Description, Cost Code, Rate, Quantity, Amounts, Tax, VAT, Total, and Actions.

4. Data Models

4.1. ServiceRequestFormModel

```
class ServiceRequestFormModel {
  int? iTransId;
  String? dDate;
  String? sNarration;
  String? sAttachment;
  int? iUser;
  int? bDraft;
  int? bWeb;
  List? body;
}
```

4.2. ServiceRequestBody

```
class ServiceRequestBody {
  int? iSlNo;
  int? iCompany;
  int? iProject;
  String? sDescription;
  int? iParticulars;
  int? iEmployee;
  int? iCostCode;
  double? nRate;
  double? nQty;
  double? nGross;
  double? nNet;
  int? iTaxCode;
  String? sVat;
  double? nTotal;
  String? sAttachment;
  String? sSignature;
  String? sEmployee;
}
```

4.3. ServiceRequestSummaryModel

```
class ServiceRequestSummaryModel {
  int? iTransId;
  String? sDocNo;
  String? date;
  String? sNarration;
  String? status;
  int? iCreatedBy;
}
```

5. API Integration

5.1. Form Submission API

Endpoint: POST /api/service-request
Supports multipart form data for JSON, file attachments, and signature.

```
Future apiPostServiceRequest({
  required ServiceRequestFormModel data,
  required List images,
  required String? signature,
})
```

5.2. List Retrieval API

Endpoint: GET /api/service-request/summary
Returns user-specific request summaries.

5.3. Supporting Dropdown APIs

- Companies: /api/companies
- Projects: /api/projects
- Employees: /api/employee
- Cost Codes, Tax Codes, Particulars

6. User Interface

6.1. Main Form Screen

Features header fields, dynamic table, main attachments, and Save/Draft controls with progress indicators.

6.2. Dialog Form

Modal editor with real-time validation, API-driven dropdowns, entry attachments, and signature capture.

6.3. List Screen

Card-based layout, pull-to-refresh, search/filter, and quick edit/delete actions.

7. Form Management

7.1. Entry Operations

- Add Entry: Inserts a new blank row.
- Edit Entry: Opens pre-filled dialog.
- Delete Entry: Confirms and removes.
- Update Entry: Deep copies attachments.

7.2. Validation Rules

- Required: Company, Project, Description
- Numeric: Rate, Quantity, Amounts
- Date format: YYYY-MM-DD
- File: Type and size constraints

7.3. Draft Persistence

Auto-save drafts locally, load on reopen, support offline use.

8. File Management

8.1. Attachments

Multi-file selection via FilePicker, thumbnails, size checks, compression, and duplicate prevention.

8.2. Storage

Temporary directories, timestamped filenames, safe copying.

8.3. Signature

SignaturePad integration, shared across entries, preview and re-capture.

9. State Management

9.1. GetX Controllers

ServiceRequestFormController and ServiceRequestListController manage reactive state and API interactions.

9.2. Data Flow

$$UserAction \rightarrow Controller \rightarrow APICall \rightarrow StateUpdate \rightarrow UIRefresh$$

9.3. Persistence

Controller lifecycle management, offline draft load, state retention on navigation.

10. Features & Functionality

10.1. Create Service Request

1. Fill header (date, narration).
2. Add main attachments.
3. Add table entries with attachments and signatures.
4. Validate and submit.

10.2. Edit Service Request

1. Load existing data.
2. Modify fields/attachments.
3. Preserve attachments.
4. Submit updates.

10.3. View Service Request List

Chronological cards, search/filter, status tracking, quick actions.

10.4. Delete Service Request

Confirmation dialog, server delete, list refresh, error handling.

11. Technical Implementation

11.1. Form Architecture

Dialog-based editing orchestrated by main screen for modularity.

11.2. File Handling

```
// FilePicker:
FilePickerResult? result = await FilePicker.platform.pickFiles(allowMultiple: true);
// Unique naming:
String stamp = DateFormat("HHmmssddMMyyyy").format(DateTime.now());
File newFile = await source.copy("${tempDir.path}/main_$stamp$ext");
```

11.3. API Pattern

Multipart REST, error logging, retry mechanisms, progress indicators.

11.4. State Code Example

```
RxList> entries = >[].obs;
RxBool isLoading = false.obs;
RxBool isFormValid = false.obs;
```

12. Error Handling

12.1. Network Errors

Connection timeouts, server errors, retries, user-friendly messages.

12.2. Validation Errors

Required fields, data types, file constraints, real-time feedback.

12.3. File Errors

Permission denied, not found, corruption detection.

12.4. API Errors

HTTP status codes, JSON parsing, authentication failures.

13. Performance Considerations

13.1. Memory Management

Controller disposal, image compression, list virtualization, cleanup.

13.2. Network Optimization

Batch calls, caching, progressive loading, connection pooling.

13.3. UI Performance

const constructors, lazy loading, animation disposal.

13.4. File Performance

Size limits, background processing, thumbnail generation, temp cleanup.

14. Conclusion

The Service Request Module provides a robust, scalable, and user-centric solution for managing service requests in the Ray International App. Its modular architecture, reactive GetX state management, comprehensive error handling, and performance optimizations align with Flutter best practices, ensuring maintainability, extensibility, and seamless user experiences.