# Invoice Processing System

A modern Angular 19 application built with PrimeNG 19 for managing invoice approval workflows. This application provides a comprehensive solution for RBI and Media Agencies to manage publications, submit invoices, and track approvals.

## Features

### 🏠 Dashboard

* **Invoicing Portal**: Central hub for managing all cases
* **Case Overview**: View all cases with filtering and search capabilities
* **Status Tracking**: Real-time status updates with color-coded badges
* **Quick Actions**: Create new cases directly from the dashboard

### 📝 Case Management

* **Case Creation**: Comprehensive form for creating new cases with:
  + Department and section selection
  + Category and case type configuration
  + Media mix selection with automatic ceiling calculations
  + Campaign and vendor details
  + Budget management
  + Free text and structured field support
* **Case Details**: Detailed view showing:
  + Complete case information
  + Media mix breakdown with visual cards
  + Status indicators
  + Approval workflow actions

### ✅ Approval Workflow

* **Multi-step Approval**: Support for different approval stages
* **Action Buttons**: Approve, Reject, or Send Back cases
* **Status Management**: Automatic status updates with notifications
* **Real-time Updates**: Immediate UI updates after actions

### 🎨 User Interface

* **Modern Design**: Clean, professional interface matching RBI branding
* **Responsive Layout**: Works seamlessly on desktop and mobile devices
* **PrimeNG Components**: Enterprise-grade UI components
* **Consistent Styling**: Unified design system throughout the application

## Technology Stack

* **Frontend Framework**: Angular 19
* **UI Library**: PrimeNG 19
* **Icons**: PrimeIcons
* **Styling**: SCSS with custom themes
* **State Management**: RxJS Observables
* **Routing**: Angular Router
* **Build Tool**: Angular CLI with Vite

## Project Structure

src/

├── app/

│ ├── layout/ # Layout components

│ │ ├── header/ # Application header

│ │ ├── sidebar/ # Navigation sidebar

│ │ └── main-layout/ # Main layout wrapper

│ ├── pages/ # Feature pages

│ │ ├── dashboard/ # Dashboard page

│ │ ├── case-list/ # Case listing page

│ │ ├── case-create/ # Case creation page

│ │ └── case-detail/ # Case detail page

│ ├── services/ # Business logic services

│ │ └── case.service.ts # Case management service

│ ├── app.component.\* # Root component

│ ├── app.config.ts # Application configuration

│ └── app.routes.ts # Routing configuration

├── assets/ # Static assets

└── styles.scss # Global styles

## Getting Started

### Prerequisites

* Node.js (v18 or higher)
* npm or yarn
* Angular CLI 19

### Installation

1. **Clone the repository**

git clone <repository-url>

cd invoice-processing-system

1. **Install dependencies**

npm install

1. **Start the development server**

ng serve

1. **Open your browser**
   Navigate to http://localhost:4200

### Build for Production

ng build --configuration production

The build artifacts will be stored in the dist/ directory.

## API Integration

The application is designed to be easily integrated with backend APIs. The CaseService provides a clean interface for:

### Service Methods

* getCases(): Retrieve all cases
* getCaseById(id): Get specific case details
* createCase(case): Create new case
* updateCaseStatus(id, status): Update case status
* cases$: Observable for real-time case updates

### Integration Points

Replace the mock data in src/app/services/case.service.ts with actual HTTP calls:

// Example API integration

getCases(): Observable<Case[]> {

return this.http.get<Case[]>('/api/cases');

}

createCase(caseData: Partial<Case>): Observable<Case> {

return this.http.post<Case>('/api/cases', caseData);

}

updateCaseStatus(id: string, status: string): Observable<Case> {

return this.http.patch<Case>(`/api/cases/${id}`, { status });

}

## Data Models

### Case Interface

interface Case {

id: string;

departmentName: string;

section: string;

category: string;

caseType: string;

campaignName: string;

vendorName: string;

fundType: string;

totalBudget: string;

status: string;

freeText: string;

mediaDetails: MediaDetail[];

dateCreated: Date;

budget: number;

}

### Media Detail Interface

interface MediaDetail {

type: string;

amount: number;

}

## Customization

### Styling

* Modify src/styles.scss for global styles
* Update component-specific styles in respective .scss files
* Customize PrimeNG theme by modifying CSS variables

### Adding New Features

1. Create new components using Angular CLI: ng generate component feature-name
2. Add routes in src/app/app.routes.ts
3. Update the service layer for data management
4. Add navigation links in the sidebar component

## Testing

### Unit Tests

ng test

### End-to-End Tests

ng e2e

### Manual Testing Checklist

* Dashboard loads with case list
* Case creation form works correctly
* Case details view displays properly
* Approval workflow functions as expected
* Navigation between pages works
* Responsive design on mobile devices
* Status updates reflect in real-time

## Browser Support

* Chrome (latest)
* Firefox (latest)
* Safari (latest)
* Edge (latest)

## Performance Considerations

* Lazy loading for feature modules
* OnPush change detection strategy
* Optimized bundle size with tree shaking
* Efficient data structures and algorithms

## Security Features

* Input validation and sanitization
* XSS protection
* CSRF protection (when integrated with backend)
* Secure routing guards

## Deployment

### Development

ng serve --host 0.0.0.0 --port 4200

### Production

ng build --configuration production

Deploy the contents of dist/ directory to your web server.

## Contributing

1. Follow Angular style guide
2. Use conventional commit messages
3. Write unit tests for new features
4. Update documentation as needed

## License

This project is proprietary software developed for RBI (Reserve Bank of India).

## Support

For technical support or questions, please contact the development team.

**Built with ❤️ using Angular 19 and PrimeNG 19**