Software Requirements Specification (SRS)

Mobile Application for Juria Cargo Management System

Version: 1.0

Technology: Flutter

---

1. Introduction

1.1 Purpose

This document specifies the requirements for developing a mobile application for the Juria Cargo Management System. The mobile app will serve customers and drivers with essential cargo tracking and management functionalities.

1.2 Scope

The mobile application will provide limited functionality compared to the web system, focusing on: - **Customer Module**: Registration, authentication, order management, and tracking - **Driver Module**: Order management, package handling, and delivery coordination

1.3 System Overview

The existing web system is built on: - **Backend**: CodeIgniter 4 (PHP) - **Database**: MySQL - **Architecture**: MVC Pattern - **Authentication**: Session-based with role-based access control

---

2. System Architecture

2.1 Existing Web System Analysis

Based on the codebase analysis: - **Framework**: CodeIgniter 4 - **Database**: MySQL (juria) - **Controllers**: 50+ controllers handling different modules - **Models**: Customer, Order, User, Pickup Request, etc. - **Authentication**: Role-based with user groups and privileges

2.2 Mobile App Architecture

- **Frontend**: Flutter (Cross-platform)
- **Backend**: Existing CodeIgniter 4 APIs (to be extended)
- **Database**: Existing MySQL database
- **Communication**: RESTful APIs with JSON responses
- **Authentication**: JWT tokens or session-based authentication

3. Functional Requirements

3.1 Customer Module

3.1.1 Authentication & Registration

Frontend Functions:

- `registerUser()` - Handle user registration form
- `loginUser()` - Handle user authentication
- `logoutUser()` - Clear user session and data
- `validateForm()` - Validate registration/login forms
- `uploadPassportPhoto()` - Handle passport photo upload

Backend API Functions:

- `POST /client/register` - Register new customer
- `POST /login` - Authenticate user
- `GET /logout` - Logout user
- `POST /forgot_password` - Initiate password reset
- `POST /reset_password` - Reset password with OTP

3.1.2 Dashboard

Frontend Functions:

- `getDashboardData()` - Fetch dashboard statistics
- `displayOrderSummary()` - Show order summary cards
- `navigateToOrders()` - Navigate to orders screen
- `refreshDashboard()` - Pull-to-refresh functionality

Backend API Functions:

- `GET /dashboard` - Get dashboard data
- `GET /profile/customer` - Get customer profile data

3.1.3 Orders Management

Frontend Functions:

- `createNewOrder()` - Create new pickup request
- `getOrderHistory()` - Fetch customer's order history
- `viewOrderDetails()` - Display detailed order information
- `calculatePrice()` - Calculate shipping price
- `trackOrder()` - Track order status
- `cancelOrder()` - Cancel pending orders

Backend API Functions:

- `POST /create_pickup_request` - Create new pickup request
- `GET /pickup_request_history` - Get order history
- `POST /cal_price` - Calculate shipping price
- `GET /getWaybillDetails/{id}` - Get waybill details
- `POST /search_waybill_history` - Search order history

3.1.4 Order Tracking

Frontend Functions:

- `trackOrderStatus()` - Get real-time order status
- `displayTrackingHistory()` - Show order progress
- `getDeliveryUpdates()` - Fetch delivery notifications
- `showOrderLocation()` - Display order location on map

Backend API Functions:

- `GET /track_orders` - Get order tracking information
- `GET /report/order_report/get_order_history` - Get detailed order history

3.1.5 Profile Management

Frontend Functions:

- `viewProfile()` - Display user profile
- `updateProfile()` - Update profile information
- `changePassword()` - Change user password
- `uploadProfilePicture()` - Update profile photo

Backend API Functions:

- `GET /profile/customer` - Get customer profile
- `POST /profile/customer/update` - Update customer profile
- `POST /my_profile/change_password` - Change password
- `POST /my_profile/upload_profile_picture` - Upload profile picture

3.2 Driver Module

3.2.1 Authentication

Frontend Functions:

- `driverLogin()` - Driver authentication
- `validateDriverCredentials()` - Validate driver login
- `logoutDriver()` - Clear driver session

Backend API Functions:

- `POST /login` - Authenticate driver (using existing login endpoint)
- `GET /logout` - Logout driver

3.2.2 Pending Orders Management

Frontend Functions:

- `getPendingOrders()` - Fetch assigned pending orders
- `viewOrderDetails()` - View detailed order information

- `acceptOrder()` - Accept pickup order
- `updatePackageDetails()` - Edit package information
- `markOrderAsPicked()` - Mark order as picked up

Backend API Functions:

- `GET /rider` - Get pending orders for driver
- `GET /rider/view_con_order/{id}` - View order details
- `POST /rider/pickup_conform` - Confirm pickup
- `POST /rider/update_package` - Update package details

3.2.3 Picked Orders Management

Frontend Functions:

- `getPickedOrders()` - Fetch picked orders
- `generateInvoice()` - Generate delivery invoice
- `viewInvoice()` - Display invoice details
- `handoverToWarehouse()` - Mark handover to warehouse
- `printInvoice()` - Print/share invoice

Backend API Functions:

- `GET /rider/picked` - Get picked orders
- `POST /rider/handover_wh` - Handover to warehouse
- `GET /print_pickup_request` - Generate printable invoice
- `GET /print_driver_pickup_request` - Generate driver invoice

3.2.4 Reports

Frontend Functions:

- `generateDailyReport()` - Generate daily pickup report
- `viewDeliveryHistory()` - View delivery history

- `exportReport()` - Export reports in various formats

Backend API Functions:

- `GET /report/order_report/get_order_history` - Get order reports
- `GET /report/order_report/collected_orders` - Get collected orders report

---

4. Non-Functional Requirements

4.1 Performance

- App launch time: < 3 seconds
- API response time: < 2 seconds
- Image loading: Progressive loading with placeholders
- Offline capability for basic features

4.2 Security

- JWT token-based authentication
- Data encryption in transit (HTTPS)
- Local data encryption
- Session timeout: 30 minutes of inactivity
- Biometric authentication support

4.3 Usability

- Intuitive Material Design/Cupertino UI
- Multi-language support (English/Sinhala)
- Accessibility compliance
- Dark/Light theme support

4.4 Compatibility

- iOS 12+ and Android 8+ support
- Screen sizes from 4.7" to 12.9"
- Both portrait and landscape orientations

---

7. User Interface Requirements

7.1 Customer App Screens

1. **Authentication Screens**
   - Login Screen

- Registration Screen (Multi-step form)
- Forgot Password Screen

2. **Main Screens**
   - Dashboard (Order statistics, quick actions)
   - New Order Screen (Pickup request form)
   - Order History Screen (List with search/filter)
   - Order Details Screen (Tracking, package details)
   - Profile Screen (Personal information management)
3. **Supporting Screens**
   - Order Tracking Screen (Real-time status)
   - Settings Screen (Preferences, notifications)
   - Help/Support Screen

7.2 Driver App Screens

1. **Authentication Screens**
   - Login Screen
2. **Main Screens**
   - Driver Dashboard (Daily statistics)
   - Pending Orders Screen (List of assigned orders)
   - Order Details Screen (Customer info, packages)
   - Picked Orders Screen (Completed pickups)
   - Invoice Screen (Generate/view invoices)
3. **Supporting Screens**
   - Reports Screen (Daily/weekly reports)
   - Profile Screen (Driver information)
   - Settings Screen

---

8. Development Phases

Phase 1: Core Development

- Set up Flutter project structure
- Implement authentication system
- Develop basic UI components
- Integrate with existing APIs

Phase 2: Customer Module

- Complete customer registration/login
- Implement dashboard and order management
- Add order tracking functionality

- Profile management features

Phase 3: Driver Module

- Develop driver interface
- Implement order pickup workflow
- Add invoice generation
- Driver reporting system

Phase 4: Testing & Deployment

- Unit and integration testing
- User acceptance testing
- Performance optimization
- App store deployment

---

9. Technical Considerations

9.1 Flutter Dependencies

```yaml
dependencies:
  flutter:
    sdk: flutter
  http: ^0.13.5 # API calls
  shared_preferences: ^2.0.15 # Local storage
  provider: ^6.0.3 # State management
  image_picker: ^0.8.6 # Photo capture
  geolocator: ^9.0.2 # Location services
  local_auth: ^2.1.6 # Biometric authentication
  flutter_secure_storage: ^9.0.0 # Secure storage
```

9.2 Backend Modifications Required

- Add JWT authentication support
- Create mobile-specific API endpoints
- Implement proper error handling for mobile responses
- Add file upload handling for mobile
- Create simplified response formats

9.3 Database Considerations

- No database schema changes required
- Utilize existing tables: users, customers, orders, pickup_requests
- May need additional fields for mobile-specific features

---

10. Security Requirements

10.1 Authentication Security

- Implement JWT with refresh tokens

- Add rate limiting for login attempts
- Use bcrypt for password hashing
- Implement account lockout mechanism

10.2 Data Security

- Encrypt sensitive data in local storage
- Use certificate pinning for API calls
- Implement proper session management
- Add biometric authentication option

10.3 API Security

- Validate all input parameters
- Use HTTPS for all communications
- Implement proper CORS policies
- Add request/response logging

---

11. Testing Strategy

11.1 Unit Testing

- Test all business logic functions
- Mock API responses for offline testing
- Test data validation and formatting
- Test error handling scenarios

11.2 Integration Testing

- Test API integration
- Test authentication flow
- Test file upload/download
- Test real-time data updates

11.3 User Acceptance Testing

- Test with real customers and drivers
- Validate user workflows
- Performance testing under load
- Usability testing on different devices

---

12. Deployment & Maintenance

12.1 Deployment Strategy

- **Android**: Google Play Store (Internal testing → Production)
- **iOS**: Apple App Store (TestFlight → App Store)
- **Backend**: Deploy API extensions to existing server

- **Database**: No migration required

12.2 Maintenance Requirements

- Regular security updates
- API endpoint monitoring
- User feedback integration
- Performance optimization
- Feature updates based on user needs

---

13. Success Criteria

13.1 Customer Success Metrics

- User registration completion rate > 80%
- Order creation completion rate > 90%
- User retention rate > 70% after 30 days
- Average session duration > 5 minutes

13.2 Driver Success Metrics

- Order pickup confirmation rate > 95%
- Invoice generation accuracy > 98%
- Daily active driver rate > 80%
- Driver satisfaction score > 4/5

13.3 Technical Success Metrics

- App crash rate < 1%
- API response time < 2 seconds
- App store rating > 4.0
- Zero critical security vulnerabilities

This SRS document provides comprehensive specifications for developing the Juria Cargo Management mobile application using Flutter, ensuring seamless integration with the existing web system.