**EatKaro**
Food Delivery & Restaurant Management Platform
Comprehensive Project Documentation

# Table of Contents

# Project Overview

EatKaro is a modern, scalable food delivery and restaurant management platform, connecting customers and restaurants with real-time features, analytics, and a seamless user experience.

# Project Objectives

• Customer-centric experience: intuitive ordering, real-time tracking, secure payments.

• Restaurant empowerment: menu management, analytics, direct customer communication.

• Platform performance: high availability, security, and real-time updates.

# Key Features

• Secure authentication (users & restaurants, role-based)

• Real-time order tracking and notifications

• Interactive chat system

• PDF bill generation

• Wishlist, cart, reviews, coupons

• Analytics dashboard for restaurants

• Multiple payment methods

# Technology Stack

## Frontend

HTML5, CSS3, JavaScript (ES6+), Tailwind CSS, Responsive Design

## Backend

Node.js, Express.js, Firebase (Realtime DB, Auth, Storage), Cloudinary, Stripe

## Other

PDF generation, Email service, Chart.js for analytics

# Architecture Overview

EatKaro uses a modular architecture with a Node.js/Express backend, Firebase for real-time data and authentication, and a modern JavaScript frontend. Static assets are served from /public, and all business logic is split between user and restaurant flows.

# Workflow

## Customer Workflow

1. Registration & Login (email/password, role selection)

2. Profile setup (personal details, address)

3. Browse restaurants and menus, add to cart/wishlist

4. Apply coupons, proceed to checkout

5. Select payment method, place order

6. Real-time order tracking, chat with restaurant

7. Receive notifications, download bill, rate & review

## Restaurant Workflow

1. Registration & Login (restaurant role)

2. Restaurant profile setup, menu management

3. Receive and manage orders in real-time

4. Update order status, chat with customers

5. View analytics, manage reviews, generate reports

## Technical Workflow

• Authentication via Firebase Auth

• Real-time data sync via Firebase Realtime Database

• Payments processed via Stripe API

• Images managed via Cloudinary

• Notifications and chat via Firebase and custom logic

## Workflow Diagram

+------------------+ +------------------+ +------------------+
| Registration | --> | Profile Setup | --> | Browse Menu |
+------------------+ +------------------+ +------------------+
| | |
v v v
+------------------+ +------------------+ +------------------+
| Apply Coupon | --> | Checkout | --> | Place Order |
+------------------+ +------------------+ +------------------+
| | |
v v v
+------------------+ +------------------+ +------------------+
| Track Order | --> | Chat | --> | Rate & Review |
+------------------+ +------------------+ +------------------+

## Restaurant Workflow Diagram

+------------------+ +------------------+ +------------------+
| Registration | --> | Profile Setup | --> | Menu Management|
+------------------+ +------------------+ +------------------+
| | |
v v v
+------------------+ +------------------+ +------------------+
| Receive Orders | --> | Update Status | --> | Chat with Users|
+------------------+ +------------------+ +------------------+
| | |
v v v
+------------------+ +------------------+ +------------------+
| View Analytics | --> | Manage Reviews | --> | Generate Reports|
+------------------+ +------------------+ +------------------+

## Technical Workflow Diagram

+------------------+ +------------------+ +------------------+
| Firebase Auth | --> | Realtime DB | --> | Stripe API |
+------------------+ +------------------+ +------------------+
| | |
v v v
+------------------+ +------------------+ +------------------+
| Cloudinary | --> | Notifications | --> | Custom Logic |
+------------------+ +------------------+ +------------------+

# Frontend

• index.html: Landing/auth page with role selection

• user-dashboard.html: Customer dashboard (browse, order, chat, wishlist, etc.)

• seller-dashboard.html: Restaurant dashboard (orders, menu, chat, analytics)

• seller-analytics.html: In-depth analytics for restaurants

• checkout.html: Checkout and payment flow

• JS: user-dashboard.js, seller-dashboard.js, auth.js, checkout.js, seller-analytics.js

• CSS: style.css, Tailwind for utility classes

# Backend

• server.js: Express server, serves static files, handles routing

• server/payment.js: Stripe payment integration

• All API endpoints are RESTful and secured

# Authentication & Security

• Firebase Auth for secure login/registration

• Role-based access (user/restaurant)

• Session management, input validation, password hashing (bcryptjs)

• CORS, XSS, CSRF protection

# Database Schema

See below for a detailed JSON schema of all collections.

# EatKaro Database Schema Documentation
## 📊 Database Overview
EatKaro uses Firebase Realtime Database with the following main collections and their relationships.
## 🔑 Collections Structure
### 1. Users Collection
```json
{
"users": {
"{userId}": {
"profile": {
"name": "string",
"email": "string",
"phone": "string",
"profilePicture": "string (URL)",
"role": "string (customer/restaurant)",
"createdAt": "timestamp",
"lastLogin": "timestamp",
"isVerified": "boolean"
},
"addresses": {
"{addressId}": {
"type": "string (home/work/other)",
"address": "string",
"landmark": "string",
"city": "string",
"state": "string",
"pincode": "string",
"isDefault": "boolean"
}
},
"paymentMethods": {
"{paymentId}": {
"type": "string (card/upi/wallet)",
"details": "object",
"isDefault": "boolean",
"lastUsed": "timestamp"
}
},
"favorites": {
"{restaurantId}": "boolean"
},
"cart": {
"items": {
"{itemId}": {
"restaurantId": "string",
"name": "string",
"quantity": "number",
"price": "number",
"specialInstructions": "string"
}
},
"total": "number",
"lastUpdated": "timestamp"
}
}
}
}
```
### 2. Restaurants Collection
```json
{
"restaurants": {
"{restaurantId}": {
"profile": {
"name": "string",
"description": "string",
"logo": "string (URL)",
"coverImage": "string (URL)",
"cuisine": ["string"],
"address": {
"street": "string",
"city": "string",
"state": "string",
"pincode": "string",
"coordinates": {
"latitude": "number",
"longitude": "number"
}
},
"contact": {
"phone": "string",
"email": "string"
},
"timings": {
"openingTime": "string",
"closingTime": "string",
"isOpen": "boolean"
},
"rating": "number",
"totalRatings": "number",
"isActive": "boolean",
"createdAt": "timestamp"
},
"menu": {
"categories": {
"{categoryId}": {
"name": "string",
"description": "string",
"image": "string (URL)",
"isActive": "boolean"
}
},
"items": {
"{itemId}": {
"name": "string",
"description": "string",
"price": "number",
"categoryId": "string",
"image": "string (URL)",
"isVeg": "boolean",
"isAvailable": "boolean",
"preparationTime": "number",
"customizations": {
"{optionId}": {
"name": "string",
"options": {
"{optionValueId}": {
"name": "string",
"price": "number"
}
}
}
}
}
}
},
"analytics": {
"daily": {
"{date}": {
"orders": "number",
"revenue": "number",
"ratings": "number"
}
},
"monthly": {
"{monthYear}": {
"orders": "number",
"revenue": "number",
"ratings": "number"
}
}
}
}
}
}
```
### 3. Orders Collection
```json
{
"orders": {
"{orderId}": {
"customerId": "string",
"restaurantId": "string",
"items": {
"{itemId}": {
"name": "string",
"quantity": "number",
"price": "number",
"customizations": {
"{optionId}": {
"name": "string",
"selected": "string",
"price": "number"
}
}
}
},
"delivery": {
"address": {
"type": "string",
"address": "string",
"landmark": "string",
"city": "string",
"state": "string",
"pincode": "string"
},
"instructions": "string",
"estimatedTime": "timestamp"
},
"payment": {
"method": "string",
"status": "string (pending/completed/failed)",
"amount": "number",
"transactionId": "string",
"paidAt": "timestamp"
},
"status": {
"current": "string (pending/accepted/preparing/ready/delivering/delivered/cancelled)",
"history": [{
"status": "string",
"timestamp": "timestamp",
"note": "string"
}]
},
"total": {
"subtotal": "number",
"tax": "number",
"deliveryFee": "number",
"discount": "number",
"final": "number"
},
"createdAt": "timestamp",
"updatedAt": "timestamp"
}
}
}
```
### 4. Reviews Collection
```json
{
"reviews": {
"{reviewId}": {
"orderId": "string",
"customerId": "string",
"restaurantId": "string",
"rating": "number",
"comment": "string",
"images": ["string (URL)"],
"createdAt": "timestamp",
"response": {
"comment": "string",
"respondedAt": "timestamp"
}
}
}
}
```
### 5. Chats Collection
```json
{
"chats": {
"{chatId}": {
"orderId": "string",
"participants": {
"customerId": "string",
"restaurantId": "string"
},
"messages": {
"{messageId}": {
"senderId": "string",
"content": "string",
"type": "string (text/image)",
"timestamp": "timestamp",
"status": "string (sent/delivered/read)"
}
},
"lastMessage": {
"content": "string",
"timestamp": "timestamp"
},
"createdAt": "timestamp",
"updatedAt": "timestamp"
}
}
}
```
### 6. Notifications Collection
```json
{
"notifications": {
"{userId}": {
"{notificationId}": {
"type": "string (order/chat/promotion)",
"title": "string",
"message": "string",
"data": "object",
"isRead": "boolean",
"createdAt": "timestamp"
}
}
}
}
```
## 🔒 Security Rules
```json
{
"rules": {
"users": {
"$uid": {
".read": "$uid === auth.uid",
".write": "$uid === auth.uid",
"profile": {
".read": true,
".write": "$uid === auth.uid"
}
}
},
"restaurants": {
"$restaurantId": {
".read": true,
".write": "auth != null && root.child('users').child(auth.uid).child('profile/role').val() === 'restaurant'"
}
},
"orders": {
"$orderId": {
".read": "auth != null && (data.child('customerId').val() === auth.uid || data.child('restaurantId').val() === auth.uid)",
".write": "auth != null && (data.child('customerId').val() === auth.uid || data.child('restaurantId').val() === auth.uid)"
}
},
"reviews": {
"$reviewId": {
".read": true,
".write": "auth != null && data.child('customerId').val() === auth.uid"
}
},
"chats": {
"$chatId": {
".read": "auth != null && data.child('participants').child(auth.uid).exists()",
".write": "auth != null && data.child('participants').child(auth.uid).exists()"
}
},
"notifications": {
"$userId": {
".read": "$userId === auth.uid",
".write": "$userId === auth.uid"
}
}
}
}
```
## 📈 Indexes
1. \*\*Orders Collection\*\*
- `customerId` + `createdAt`
- `restaurantId` + `createdAt`
- `status.current` + `createdAt`
2. \*\*Restaurants Collection\*\*
- `cuisine` + `rating`
- `isActive` + `rating`
3. \*\*Reviews Collection\*\*
- `restaurantId` + `createdAt`
- `customerId` + `createdAt`
4. \*\*Chats Collection\*\*
- `orderId` + `updatedAt`
- `participants` + `updatedAt`

# Analytics & Reporting

• Seller dashboard includes sales, revenue, order status, and item popularity charts

• Reports can be exported (CSV, PDF)

# Notifications & Real-Time Features

• Real-time chat between users and restaurants

• Push notifications for order status, new messages, promotions

# Deployment & Environment

1. Clone the repo, run npm install

2. Set up .env and Firebase/Cloudinary credentials

3. npm run dev to start locally

4. Deploy static files to Firebase Hosting or similar

# Contribution Guidelines

• Fork the repo, create a feature branch, submit a pull request

• Follow code style and add documentation/comments

# Contact & Support

For support, email support@eatkaro.com or open an issue on GitHub.

*Made with ❤️ by the EatKaro Team*

**Follow us on social media:**
Twitter: @EatKaro
Facebook: EatKaro
Instagram: @EatKaro