BiteFlow - Advanced Restaurant Management System

Overview

BiteFlow is an innovative restaurant management platform built with Flutter that revolutionizes the dining experience. It combines powerful features like real-time payment processing through Stripe, collaborative dining with QR code table sharing, and comprehensive restaurant management capabilities.

Table of Contents

Key Features

Advanced Payment Processing

- **Stripe Integration**
  - Real-time payment processing
  - Secure payment handling
  - Multiple payment methods support
  - Transaction history tracking
  - Payment status monitoring

Collaborative Dining

- **QR Code Integration**
  - Dynamic QR code generation
  - Instant table joining
  - Real-time order synchronization
  - Group order management
  - Split bill functionality

Bill Splitting System

- **Multiple Splitting Methods**
  - Equal split functionality
  - Item-based splitting
  - Percentage-based division
  - Individual payment tracking

    - Split history maintenance

Restaurant Management

- **Menu Management**
    - Dynamic category organization
    - Real-time menu updates
    - Image management
    - Price control
    - Item availability tracking

Order Processing

- **Real-time Order Management**
    - Live order tracking
    - Status updates
    - Kitchen notifications
    - Order history
    - Special instructions handling

Marketing Tools

- **Promotional Management**
    - Offer creation
    - Campaign tracking
    - Customer targeting
    - Discount management
    - Performance analytics

Screenshots

Database Details

Authentication

We use **Firebase Authentication** to manage user access and permissions. Below are the key details for authentication:

- **Authentication Provider**: Firebase Auth
- **Supported Methods**:
    - Email/Password Authentication
    - Google Sign-In

Firestore Collections

The following collections are used in the **Firestore Database**:

Collection Name	Purpose	Fields

Collection Name	Purpose	Fields
`users`	Stores user profiles	`id`, `name`, `email`, `role`, `fcmToken`, `unseenOfferCount`
`clients`	Stores client-specific data	`id`, `name`, `email`, `orderIds`, `fcmToken`, `unseenOfferCount`
`managers`	Stores manager-specific data	`id`, `name`, `email`, `restaurantId`
`restaurants`	Holds restaurant details	`id`, `name`, `managerId`, `location`, `rating`, `reviewCount`, `description`, `isTableAvailable`, `imageUrl`
`menu`	Contains menu items for each restaurant	`id`, `title`, `price`, `categoryId`, `restaurantId`, `discountPercentage`, `description`, `rating`, `imageUrl`
`categories`	Manages item categories	`id`, `title`, `restaurantId`
`orders`	Tracks orders placed by users	`id`, `status`, `totalAmount`, `items`, `orderClientsPayment`, `paymentMethod`, `restaurantId`, `orderNumber`
`order_items`	Contains individual order items	`id`, `title`, `price`, `quantity`, `notes`, `discountPercentage`, `participants`, `categoryId`, `restaurantId`
`promotional_offers`	Holds promotional offers managed by restaurants	`id`, `restaurantId`, `restaurantName`, `title`, `description`, `imageUrl`, `startDate`, `endDate`, `discount`, `isActive`
`carts`	Tracks group carts with participants	`id`, `restaurantId`, `creatorId`, `participants`, `items`, `isDeleted`
`comments`	Stores user comments and ratings for restaurants	`id`, `userId`, `restaurantId`, `text`, `rating`, `createdAt`
`notifications`	Manages offer notifications for users	`id`, `title`, `endDate`
`payments`	Tracks user payments for orders	`id`, `userId`, `isPaid`, `amount`

Database Configuration

- **Database Type**: Firestore (NoSQL)
- **Region**: `europe-west1` *(adjust as per your project configuration)*
- **Data Consistency**: Strong consistency using Firestore's document-based structure.
- **Security Rules**: Firestore security rules limit read/write access to authorized users.

Firestore and Firebase Storage Rules

```
rules_version = '2';

service cloud.firestore {
  match /databases/{database}/documents {

    match /{document=**} {
      allow read, write: if request.time < timestamp.date(2024, 12, 18);
    }
  }
}

service firebase.storage {
  match /b/{bucket}/o {

    match /{allPaths=**} {
      allow read, write: if request.time < timestamp.date(2025, 1, 10);
    }

    // Development-specific rule
    match /uploads/{fileName} {
      allow read: if true;
      allow write: if true;  // For development only
    }
  }
}
```

Technical Architecture

Core Technologies

Frontend:

- Flutter Framework
- Provider State Management
- GetIt Dependency Injection
- Custom Theme System

Backend:

- Firebase Authentication
- Cloud Firestore
- Firebase Cloud Functions
- Firebase Cloud Messaging
- Firebase Storage

Payment Processing:

- Stripe SDK Integration
- Secure Payment Gateway
- Transaction Management

Project Structure

```
lib/
├── core/
│ ├── constants/
│ │ ├── api_constants.dart
│ │ ├── business_constants.dart
│ │ ├── firestore_collections.dart
│ │ ├── navbar_constants.dart
│ │ └── theme_constants.dart
│ ├── providers/
│ │ ├── notification_provider.dart
│ │ └── user_provider.dart
│ └── utils/
│ ├── auth_helper.dart
│ ├── price_calculator.dart
│ ├── result.dart
│ └── status_icon_color.dart
│
├── dummy_data/
├── models/
│ ├── cart.dart
│ ├── category.dart
│ ├── client.dart
│ ├── comment.dart
│ ├── dialog_models.dart
│ ├── item.dart
│ ├── manager.dart
│ ├── menu_item.dart
│ ├── offer_notification.dart
│ ├── order_clients_payment.dart
│ ├── order_full_clients_payment.dart
│ ├── order_item_participant.dart
│ ├── order_item.dart
│ ├── order.dart
│ ├── promotional_offer.dart
│ ├── restaurant.dart
│ └── user.dart
│
├── services/
├── viewmodels/
│ ├── base_model.dart
│ ├── cart_item_view_model.dart
│ ├── cart_view_model.dart
│ ├── client_offers_view_model.dart
│ ├── client_orders_view_model.dart
│ ├── entry_point_view_model.dart
│ ├── feedback_view_model.dart
│ ├── home_view_model.dart
│ ├── image_view_model.dart
│ ├── login_view_model.dart
│ ├── manager_create_item_view_model.dart
│ ├── manager_menu_view_model.dart
│ ├── manager_offers_view_model.dart
│ ├── manager_orders_details_view_model.dart
│ ├── manager_orders_view_model.dart
│ ├── manager_promotional_offers_view_model.dart
│ ├── menu_view_model.dart
│ ├── mode_view_model.dart
│ ├── order_view_model.dart
│ ├── payment_view_model.dart
│ ├── profile_view_model.dart
│ ├── rating_view_model.dart
│ ├── restaurant_onboarding_view_model.dart
│ └── signup_view_model.dart
│
├── views/
│ ├── screens/
│ │ ├── feedback/
│ │ │ ├── feedback_screen.dart
│ │ │ └── feedback_view.dart
│ │ ├── home/
│ │ │ ├── home_screen.dart
│ │ │ └── home_view.dart
│ │ ├── login/
│ │ │ ├── components/
│ │ │ ├── login_screen.dart
│ │ │ └── login_view.dart
│ │ ├── manager_menu/
│ │ │ ├── components/
│ │ │ ├── manager_menu_screen.dart
│ │ │ └── manager_menu_view.dart
│ │ ├── manager_orders/
│ │ │ ├── components/
│ │ │ │ ├── order_bottom_sheet.dart
│ │ │ │ ├── order_details.dart
│ │ │ │ ├── order_update_status.dart
│ │ │ │ ├── orders_list.dart
│ │ │ │ └── track_payments.dart
│ │ │ ├── manager_orders_screen.dart
│ │ │ └── manager_orders_view.dart
│ │ ├── manager_promotional_offers/
│ │ │ ├── add_promotional_offer_screen.dart
│ │ │ ├── manager_promotional_offers_screen.dart
│ │ │ └── manager_promotional_offers_view.dart
│ │ ├── menu/
│ │ │ ├── menu_screen.dart
│ │ │ └── menu_view.dart
│ │ ├── order_details/
│ │ │ ├── client_orders_list.dart
│ │ │ ├── order_details_screen.dart
│ │ │ ├── order_details_view.dart
│ │ │ ├── orders_screen.dart
│ │ │ └── orders_view.dart
│ │ ├── payment/
│ │ │ ├── payment_test_screen.dart
│ │ │ └── payment_test_view.dart
│ │ ├── profile/
│ │ │ ├── profile_screen.dart
│ │ │ └── profile_view.dart
│ │ ├── rating/
│ │ │ ├── rating_screen.dart
│ │ │ └── rating_view.dart
│ │ ├── restaurant_onboarding/
│ │ │ ├── components/
│ │ │ ├── restaurant_onboarding_screen.dart
│ │ │ └── restaurant_onboarding_view.dart
│ │ ├── search/
│ │ │ └── search_view.dart
│ │ ├── signup/
│ │ │ ├── components/
│ │ │ ├── signup_screen.dart
│ │ │ └── signup_view.dart
│ │ └── split_bill/
│ │ └── split_screen.dart
│ │
│ ├── theme/
│ │ ├── biteflow_theme.dart
│ │ ├── button_theme.dart
│ │ ├── checkbox_themedata.dart
│ │ ├── input_decoration_theme.dart
│ │ └── theme_data.dart
│ └── widgets/
│ ├── auth/
│ ├── cart/
│ ├── dialogues/
│ ├── home/
│ ├── menu/
│ ├── order/
│ ├── rating/
│ ├── user/
│ └── utils.dart
│
├── animated_splash_screen.dart
├── firebase_notifications.dart
├── firebase_options.dart
├── locator.dart
└── main.dart
```

Implementation Details

Payment Integration

```
class PaymentService {
  Future<PaymentIntent> createPaymentIntent(double amount) async {
    try {
      final response = await _stripe.createPaymentIntent(
        amount: amount.toInt() * 100,
        currency: 'USD',
        paymentMethodTypes: ['card'],
      );
      return PaymentIntent.fromJson(response.data);
    } catch (e) {
      throw PaymentException(message: 'Failed to create payment intent');
    }
  }
}
```

QR Code Implementation

```
class QRCodeService {
  String generateOrderQR(String orderId) {
    final data = {
      'orderId': orderId,
      'timestamp': DateTime.now().toIso8601String(),
      'restaurantId': restaurantId,
    };
    return jsonEncode(data);
  }

  Future<void> joinOrder(String qrData) async {
    final decodedData = jsonDecode(qrData);
    // Join order logic
  }
}
```

Installation Guide

Prerequisites

- Flutter SDK (Latest stable version)
- Firebase CLI
- Node.js & npm
- Stripe Account
- Android Studio / VS Code

Setup Steps

1. Clone the repository

```
git clone https://github.com/your-username/biteflow.git
cd biteflow
```

2. Install dependencies

```
flutter pub get
```

3. Configure Firebase

```
firebase init
# Configure Firebase services
firebase deploy
```

4. Configure Stripe

- Create a `.env` file:

```
STRIPE_PUBLISHABLE_KEY=your_publishable_key
STRIPE_SECRET_KEY=your_secret_key
```

5. Run the application

```
flutter run
```

Configuration

Environment Variables

```
# .env configuration
STRIPE_PUBLISHABLE_KEY=pk_test_...
STRIPE_SECRET_KEY=sk_test_...
```

Firebase Configuration

```
await Firebase.initializeApp(
  options: DefaultFirebaseOptions.currentPlatform,
);
```

Usage Guide

Payment Processing

1. Create payment intent
2. Present payment sheet
3. Handle payment result
4. Update order status

QR Code Sharing

1. Generate QR code for order
2. Share with other users
3. Scan and join order
4. Synchronize order details

Contributing

Development Process

1. Fork the repository
2. Create feature branch
3. Implement changes
4. Submit pull request

Code Standards

- Follow Flutter style guide
- Write unit tests
- Document new features
- Update README as needed

License

This project is licensed under the MIT License - see the LICENSE file for details.

Support

Acknowledgments

- Dr. Milad Ghantos - Project Supervisor
- Team Members:
  - Abdelrahman Mohamed Salah
  - Ahmad Hoseiny AlShahhat
  - Mohamed Hassan Samy
  - Mohamed Khaled Fouad
  - Youssef Mohamed Shawky
