



# INTERACTIVE MOBILE APP FOR STREAMLINED RESTAURANT SERVICES

Ali Madiyar

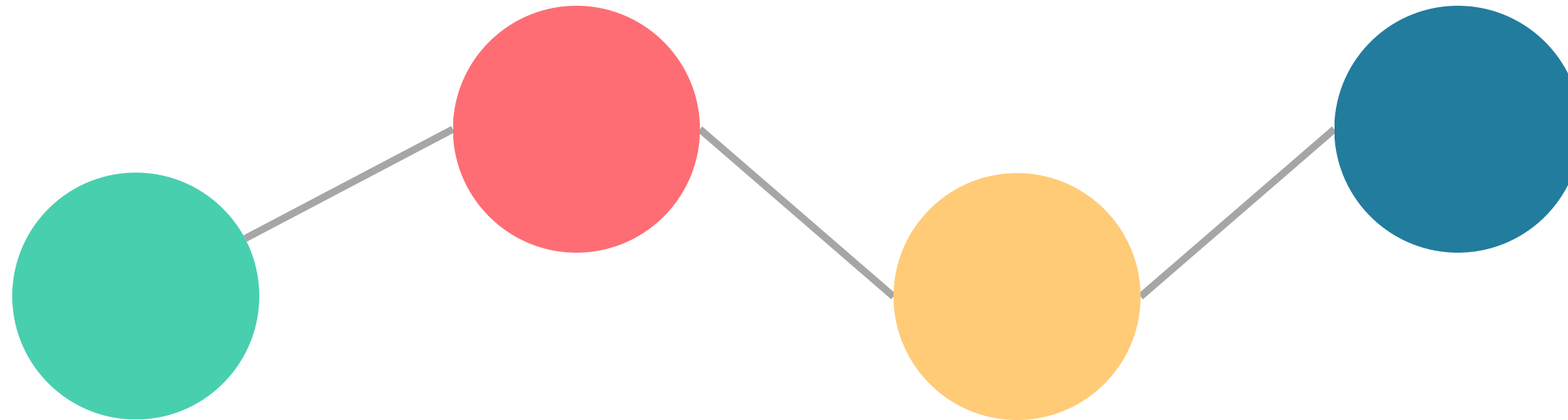
# PROBLEM STATEMENT

## Order Errors

Manual order-taking increases the likelihood of miscommunication and mistakes.

## Lack of Feedback Mechanisms:

Limited or no structured way for customers to provide meaningful feedback.



## Communication Delays

Waitstaff often face delays in taking orders and delivering them to the kitchen.

## Inefficient Payments:

Customers often experience delays during the payment process.



# SOLUTION OVERVIEW

## 01 - SEAMLESS MENU BROWSING

Customers can browse digital menus on their devices.

## 02 - ORDER PLACEMENT

Orders are placed directly from the app, reducing waitstaff involvement.

## 03 - INTEGRATED PAYMENTS

Secure, instant payments through multiple payment gateways.

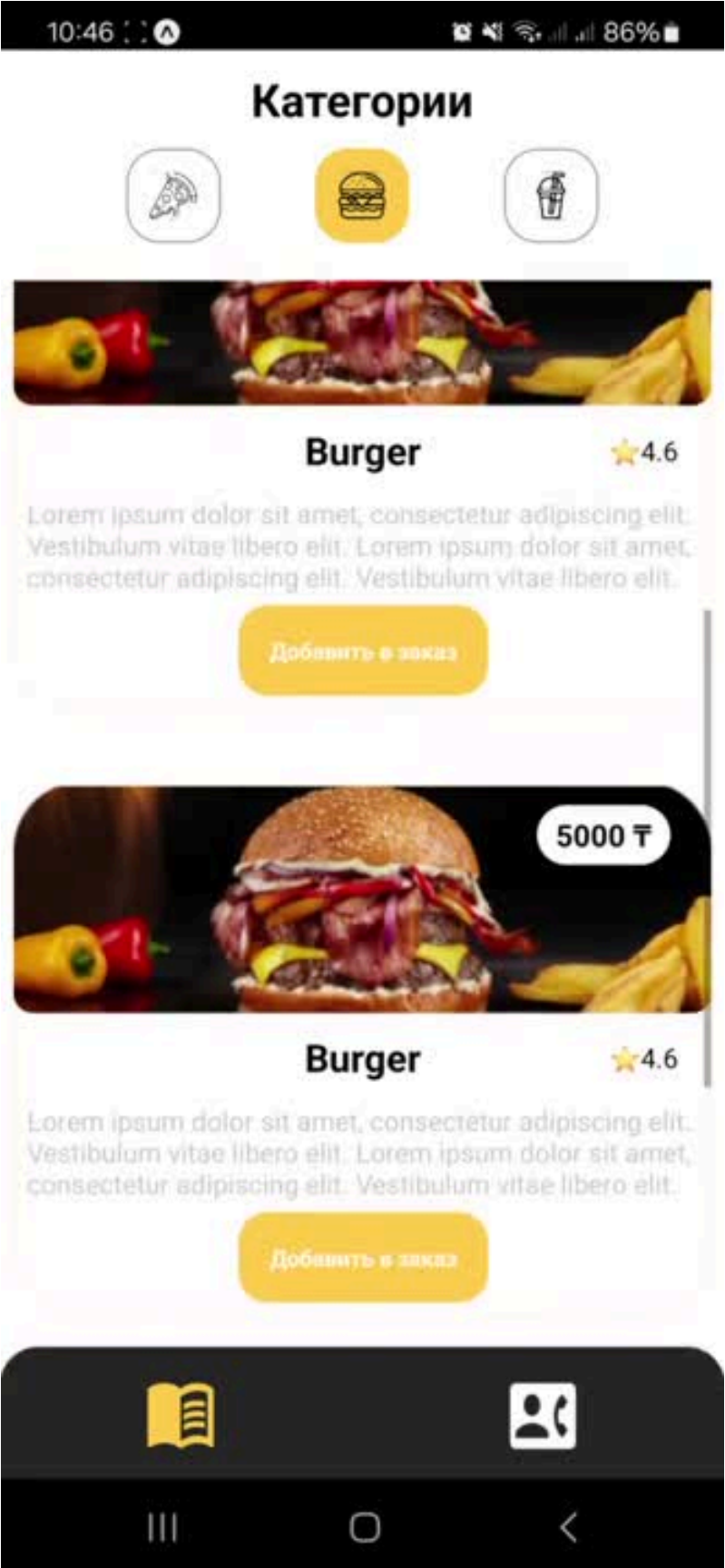
## 04 - FEEDBACK MECHANISM

Easy-to-use feedback forms for continuous improvement.



The background features four abstract geometric patterns in the corners. Top-left: A series of parallel diagonal lines in a light blue-grey color, with a curved line segment on the right. Top-right: A cluster of semi-circles in yellow, dark blue, red, and teal. Bottom-left: A cluster of semi-circles in red, teal, dark blue, and red. Bottom-right: A series of parallel diagonal lines in a light blue-grey color, with a curved line segment on the left.

DEMO



# TECHNICAL APPROACH

## 1. Frontend (React Native with Expo)

- React Native: A powerful framework for building cross-platform mobile applications with a single codebase for iOS and Android.
- Expo: Simplifies the development and deployment process with built-in tools for debugging, testing, and building applications efficiently.
- Navigation: react-navigation enables seamless screen transitions and structured navigation using stack and tab navigators.

## 2. State Management (React Context API)

- React Context API: Used to manage global application state without relying on external libraries like Redux.
- State persistence and sharing across components are managed using useContext and useReducer hooks, ensuring smooth data flow and centralized state management.

## 3. Local Data Storage

- All data is managed directly within the application state (Context API) without external databases or backend services.
- State updates are handled locally, and data persists during the app's runtime.

# CHALLENGES

## 01 - NAVIGATION

- Problem: Structuring complex navigation hierarchies with multiple stack and tab navigators while maintaining clarity and avoiding deep nesting issues.
- Solution: Used react-navigation to design clean and modular navigation structures. Stack navigators were employed for screen-to-screen flows, while tab navigators were used for main dashboard-level views.




## 02 - STATE MANAGEMENT

- Problem: Managing and sharing state across multiple screens and components without causing performance bottlenecks or unnecessary re-renders.
- Solution: Implemented the React Context API effectively with useContext and useReducer hooks. This ensured a centralized state management system, minimized redundant state updates, and allowed seamless communication between different parts of the app.





## 03 - NAVIGATION

- Problem: Structuring complex navigation hierarchies with multiple stack and tab navigators while maintaining clarity and avoiding deep nesting issues.
  - Solution: Used react-navigation to design clean and modular navigation structures. Stack navigators were employed for screen-to-screen flows, while tab navigators were used for main dashboard-level views.
- 



# IMPACT

## **Current Impact:**

- Faster order processing and reduced waiting times.
- Enhanced accuracy in order fulfillment.
- Improved customer feedback collection and analysis.

## **Future Enhancements:**

- AI-Powered Dish Recommendations: Personalized suggestions based on customer history.
- Table Reservation Integration: Enable customers to book tables in advance.
- Scalability: Adapt the app for use in larger restaurant chains.

**THANK YOU FOR  
YOUR  
ATTENTION!**