Of course. Here is the updated project prompt, revised to incorporate the dual authentication strategy, the single React Native for Web codebase architecture, and the email verification process.

Wawa Cafe Web & Mobile App Development Project

Project Overview

Build a comprehensive food ordering platform for **Wawa Cafe**. This project will consist of two main components:

1. A **unified client application** for both mobile (iOS/Android) and web, built from a single **React Native for Web** codebase.
2. A **separate and standalone backend API** built with Node.js.

The application will facilitate table service, pickup, and delivery orders with multiple payment options and a modern, secure authentication system.

Core Features Required

1. User Interface & Experience

- **Welcome Screen**: Wawa Cafe branding with order type selection (responsive for web and mobile).
- **Authentication Options**:
  - **Create Account**: Email and password registration.
  - **Email Verification**: A mandatory 4-digit PIN verification step during sign-up to ensure email validity.
  - **Sign In**: For existing, verified users.
  - **Continue as Guest**: Anonymous checkout for streamlined ordering.
- **Order Type Selection**:
  - Dine-in (table service)
  - Pickup
  - Delivery
- **Menu Navigation**: Intuitive category-based Browse, adapted for both touch and mouse input.
- **Shopping Cart**: Real-time order summary with quantity adjustments.
- **Order Tracking**: Live status updates and estimated wait times.

PROF

2. Menu Categories & Items

Drinks Category:

- Beer (with subcategories: Local, Imported, Craft)
- Wine (Red, White, Rosé, Sparkling)
- Soft Drinks (Sodas, Juices, Water, Coffee, Tea)

Food Category:

- Starters/Appetizers
- Main Courses
- Desserts

Menu Item Details:

- High-quality images
- Detailed descriptions
- Pricing
- Customization options (size, extras, dietary preferences)
- Availability status

3. Ordering System

- **Table Service**: Table number input/QR code scanning.
- **Pickup Orders**: Customer details and preferred pickup time.
- **Delivery Orders**: Address input with delivery radius validation.
- **Special Instructions**: Text field for dietary requirements or preferences.
- **Order Modification**: Ability to edit orders before confirmation.

4. Payment Integration

Implement multiple payment gateways:

- **Google Pay** (Android)
- **Apple Pay** (iOS)
- **Flutterwave** (card payments, bank transfers)
- **Cryptocurrency payments** (Bitcoin, Ethereum, stablecoins)
- Payment security with encryption and tokenization.

5. Order Management & Communication

- **Order Confirmation**: Immediate confirmation with order number.
- **Wait Time Estimation**: Dynamic timing based on kitchen load.
- **Status Updates**: Real-time notifications (Order received → Preparing → Ready → Delivered).
- **Random Rewards System**:
  - Configurable spend thresholds trigger random benefits.
  - Rewards include: discount coupons, free items, bonus points.
  - Surprise rewards notification system.
- **Issue Resolution**:
  - Contact restaurant button
  - Report order issues
  - Request modifications
  - Cancel order (if within time window)

6. Admin Dashboard

- **Inventory Management**:
  - Add/edit/remove menu items

    - Stock level tracking and alerts
    - Bulk import/export functionality
    - Real-time availability updates
- **Order Management**:
    - Live order queue with status updates
    - Kitchen display system integration
    - Order history and analytics
- **Customer Management**:
    - User accounts and profiles
    - Order history per customer
    - Loyalty program management
- **Rewards Configuration**:
    - Set spend thresholds for random benefits
    - Configure reward types and probabilities
    - Track reward redemptions
- **Analytics Dashboard**:
    - Sales reports and trends
    - Popular items analysis
    - Peak hour insights
    - Revenue tracking
- **Settings Management**:
    - Restaurant hours and availability
    - Delivery radius and fees
    - Payment method toggles

Technical Requirements

Frontend (Unified Web & Mobile App)

A single, unified codebase using **React Native for Web** will be developed to target iOS, Android, and modern web browsers.

PROF

Guiding Principles:

- **Design "Mobile-First" but "Web-Aware"**: Components and layouts will be designed primarily for a mobile experience but must be responsive and adapt gracefully to larger desktop screens and mouse/keyboard interaction.
- **Abstract Your Logic**: Business logic, API calls, and state management will be kept separate from the UI layer to maximize code reuse across all platforms.
- **Implement Platform-Specific Code**: Utilize framework features (e.g., `Platform.select` API, `.web.tsx` / `.native.tsx` file extensions) to handle features unique to a platform, such as secure token storage on mobile vs. session management on the web.

Key Technologies:

- **Framework**: React Native with Expo or React Native CLI.
- **Navigation**: React Navigation v6 for screen transitions.
- **State Management**: Redux Toolkit or Context API.

- **UI Components**: Native Base, React Native Elements, or custom components.
- **Maps Integration**: React Native Maps for delivery addresses.
- **Push Notifications**: Firebase Cloud Messaging (for mobile).
- **Payment SDKs**: Integration with payment provider SDKs.
- **Image Handling**: React Native Fast Image for menu photos.
- **Form Validation**: Formik or React Hook Form.

Backend (Separate Node.js Project)

The backend will be developed as a completely separate and standalone Node.js project, providing a RESTful API for all client applications.

Key Technologies & Architecture:

- **Framework**: Express.js with TypeScript.
- **Database**: MongoDB with Mongoose ODM.
- **Authentication**: A **Dual-Pronged Approach** to authentication will be implemented:
    - **Web App (Session-Based)**: Secure, `HttpOnly`, `SameSite=Strict` session cookies managed by `express-session` with a `connect-mongo` store.
    - **Mobile App (Token-Based)**: JWTs (short-lived access tokens + long-lived secure refresh tokens) signed with an asymmetric algorithm (e.g., **RS256**). Refresh tokens will be stored securely on the device (Keychain for iOS, EncryptedSharedPreferences for Android).
    - **Unified RBAC**: A single middleware will protect routes based on user roles (`admin`, `user`) by checking for roles in either the JWT payload or the session data.
- **Email Verification**: Integration with a service like SendGrid or Nodemailer to dispatch 4-digit verification PINs during user registration.
- **API Structure**: RESTful API with proper status codes.
- **Payment Processing**: Secure webhook handling for payment confirmations.
- **Real-time Updates**: Socket.io for order status updates.
- **File Storage**: AWS S3 or Cloudinary for menu images.
- **Email/SMS**: SendGrid for receipts, Twilio for SMS notifications.
- **Admin Panel**: The admin dashboard will be built as part of the unified **React Native for Web** application, creating a web-specific view that reuses existing components, API services, and state management logic from the core app.

PROF

Database Schema (MongoDB Collections)

Users Collection:

- User ID, name, email (unique, verified), password (hashed), phone, addresses, payment methods, total spent, rewards earned.

(Other collections remain the same: Menu Items, Orders, Payments, Rewards, Reward Rules, Inventory)

User & Order Flows

User Registration & Verification Flow:

1. Open app/website → Select "Create Account".
2. Enter email and password → Submit.
3. Backend sends a 4-digit PIN to the provided email.
4. User is prompted to enter the 4-digit PIN.
5. User enters PIN → Backend verifies it.
6. On success, the account is marked as verified and the user is logged in.

Dine-in Order Flow (Web & Mobile):

1. Open app/website → Select "Dine-in".
2. Scan QR code or enter table number.
3. Browse menu → Add items to cart.
4. Review order → Select payment method.
5. Complete payment → Receive confirmation.
6. Track order status → Food delivered to table.

Delivery Order Flow (Web & Mobile):

1. Open app/website → Select "Delivery".
2. Enter/confirm delivery address.
3. Browse menu → Customize items.
4. Checkout → Choose payment method.
5. Order confirmation → Track delivery.
6. Receive order → Rate experience.

Deployment Requirements

- **Web Application**: Deploy the static web build to a service like Vercel, Netlify, or AWS S3/CloudFront.
- **Mobile Apps**: Deploy to Google Play Store and Apple App Store.
- **Backend**: AWS, Google Cloud, or DigitalOcean deployment.
- **Domain**: Custom domain for API endpoints.
- **SSL Certificates**: HTTPS encryption for all communications.
- **Monitoring**: Application performance monitoring (APM).
- **Analytics**: User behavior tracking for business insights.

PROF

Testing Strategy

- **Unit Tests**: Component and function testing
- **Integration Tests**: API endpoint testing
- **End-to-End Tests**: Complete user journey testing
- **Payment Testing**: Test all payment methods in sandbox mode
- **Performance Testing**: Load testing for peak hours
- **Security Testing**: Penetration testing for vulnerabilities

Launch Preparation

- **Beta Testing**: Internal testing with restaurant staff

- **Soft Launch**: Limited customer testing
- **Staff Training**: Restaurant team training on order management
- **Marketing Materials**: App store descriptions and screenshots
- **Customer Support**: Help documentation and support channels

Success Metrics

- Order completion rate
- Payment success rate
- Average order value
- Customer retention rate
- App store ratings
- Order processing time
- Customer satisfaction scores

Budget Considerations

- Development team costs (frontend, backend, mobile developers)
- Third-party service fees (payment gateways, cloud hosting)
- App store fees and annual developer accounts
- Marketing and promotion costs
- Ongoing maintenance and updates