Teka Platform: A Comprehensive Feature and Architectural Blueprint

Section 1: Platform Vision and Strategic Positioning

This document outlines the comprehensive feature set, system architecture, and strategic framework for the Teka platform. Teka is conceived as a hyperlocal, two-sided marketplace designed to connect independent culinary entrepreneurs—home chefs—with local clients seeking authentic, homemade meals. The platform's architecture and features are engineered not merely to facilitate transactions, but to cultivate a robust, community-driven ecosystem built on trust, quality, and personal connection.

1.1 The Market Opportunity: Beyond Food Delivery

The contemporary food delivery market is dominated by aggregators like Uber Eats and DoorDash, which primarily serve as logistics and marketing channels for established restaurants.[1] While this market is mature, a significant and largely untapped segment exists: the home chef.[2] This segment represents a powerful alignment with burgeoning consumer trends favoring hyperlocalism, artisanal products, and authentic culinary experiences that mass-market chains cannot replicate.[2]

Teka's strategic position is to be the definitive platform for this niche. It will not compete on the same terms as traditional delivery apps. Instead, it will differentiate itself by offering a curated selection of unique, high-quality meals prepared by passionate local cooks. The platform's value is rooted in storytelling and authenticity, transforming the act of ordering food from a simple transaction into a meaningful connection with the person behind the meal.[1] This approach taps into a powerful emotional driver; consumers are more likely to remain loyal to businesses with which they feel a personal connection.[2]

1.2 Core Value Proposition

The platform's success is predicated on delivering distinct and compelling value to its three key stakeholders: clients, chefs, and the platform itself.

- **For Clients:** Teka offers exclusive access to a diverse and rotating menu of homemade meals that are unavailable through conventional channels. It provides a trusted and convenient way to discover local culinary talent, enjoy authentic food from various cultures, and support small, local entrepreneurs. The emphasis on chef profiles, reviews, and direct communication fosters a sense of community and trust that is absent in anonymous, transaction-focused platforms.
- **For Chefs:** Teka provides a comprehensive "business-in-a-box" solution, empowering chefs to launch and manage their own food business with minimal upfront investment and operational overhead.[2] The platform abstracts away the complexities of building a web presence, managing orders, processing payments, and coordinating logistics. By providing a full suite of tools—from a customizable digital storefront to sales analytics—Teka allows chefs to focus on their core passion: cooking.[3] This model effectively crowdsources the supply chain, leveraging a decentralized network of pre-equipped kitchens and eliminating the need for capital-intensive infrastructure.[2]
- **For the Platform:** The business model is designed for scalable and sustainable growth. By focusing on the home chef segment, Teka can achieve significantly lower customer acquisition costs (CAC). Unlike platforms that must invest heavily in marketing to attract customers to large restaurants, Teka leverages the inherent local networks of its chefs.[2] Each chef who joins the platform brings a pre-existing community of friends, neighbors, and social media followers, creating a powerful, organic, and high-ROI marketing channel.[2] This community-driven growth, combined with high user retention fueled by personal connections, creates a defensible market position.

1.3 The Trust-First Framework

In a marketplace connecting individual producers with consumers, trust is the fundamental currency. Teka must be engineered from the ground up to establish and maintain this trust. This principle informs the design of every feature, from the initial chef onboarding process to the post-order review system. Key elements of this framework include:

- **Vetted Onboarding:** A rigorous but streamlined process for verifying chef identity and ensuring compliance with local food safety regulations.[1]
- **Transparent Profiles:** Rich chef profiles that go beyond a menu, showcasing the chef's story, culinary philosophy, and even videos of their process, fostering a personal

connection.[1]

- **Social Proof:** A robust, two-way review and rating system that allows both clients and chefs to build a reputation on the platform, creating a self-regulating community of quality.[2]
- **Secure Transactions:** Guaranteed and secure payments managed through a trusted third-party processor like Stripe, ensuring chefs are paid reliably and client financial data is protected.[3]

The platform's growth strategy is fundamentally intertwined with its ability to empower chefs as its primary marketers. The most valuable asset is not a centralized advertising budget but the decentralized, authentic reach of its culinary partners. Home chefs naturally engage in word-of-mouth promotion through local community channels like WhatsApp groups and Instagram stories.[2] Therefore, the platform's feature set must be designed to amplify these efforts. Tools such as easily shareable menu links, seamless social media integration on chef profiles, and a built-in promotional code generator are not ancillary features; they are core components of the platform's growth engine.[3] The success of Teka is directly proportional to how effectively it equips its chefs to be its most powerful brand ambassadors.

Section 2: The Client Experience Portal

The client-facing portal is designed for frictionless discovery, intuitive ordering, and a transparent, trust-building post-purchase experience. The user journey is architected to maximize engagement and conversion by removing barriers to entry, allowing users to explore the full value of the platform before committing to creating an account.

2.1 Discovery and Browsing (Unauthenticated Access)

The initial interaction with Teka must be immediate, visually engaging, and highly relevant to the user's context.

- **Location-Based Chef Discovery:** Upon landing on the platform's homepage, the system will use Geo-IP detection or browser location services to automatically identify the user's approximate location and display a curated list of nearby chefs. A prominent, persistent search bar will allow users to enter a specific address or postal code for more precise, hyperlocal results, which is a key differentiator from platforms with a broader, less personal focus.[2]

- **Interactive Map and List View:** The user will be presented with two primary modes of discovery: an interactive map displaying pins for each available chef, and a more detailed filterable list view. The list view will serve as the main discovery interface, showcasing each chef with a high-quality lead photo, their name, primary cuisine type, a price range indicator (e.g., $, $$, $$$), and their average star rating.
- **Advanced Search and Filtering:** To empower users to find exactly what they are looking for, a comprehensive set of filtering options will be available. Users can refine the list of chefs based on criteria such as:
    - **Cuisine Type:** (e.g., Italian, Mexican, Indian, Vegan)
    - **Price Point:** Matching the price range indicators.
    - **Dietary Accommodations:** Specific tags for options like Vegetarian, Gluten-Smart, Keto-Friendly, or Calorie-Conscious, similar to established meal kit services.[6]
    - **Chef Rating:** Filtering by average star rating (e.g., 4 stars and above).
    - **Availability:** A toggle to show only chefs who are currently "online" and accepting orders.
- **Viewing Chef Profiles:** Clicking on a chef's card will navigate the user to their detailed profile page. This page is a critical touchpoint for building trust and telling the chef's story. It will feature all the elements configured by the chef (as detailed in Section 3.1), including their personal bio, a gallery of food images, short videos, and the integrated social media iframe that allows exploration of their social presence without leaving the Teka platform.
- **Menu Exploration:** The chef's menu will be prominently displayed on their profile page. Each menu item will be presented with a high-resolution image, a compelling description, a clear price, and a list of key ingredients or potential allergens.

2.2 The Ordering Funnel

The process of selecting items and completing a purchase is optimized for simplicity and speed, guiding the user from browsing to checkout with minimal friction.

- **Adding to Cart:** Each menu item will have a clear "Add to Cart" button. If an item has customizable options (e.g., choice of protein, spice level), clicking the button will trigger a modal pop-up where the user can make their selections before adding the item to their cart. This adheres to the specified UI pattern of using modals to handle CRUD-like operations and maintain a fluid, single-page feel.
- **The Shopping Cart:** The user's cart will be persistently accessible, often as a slide-out sidebar or a pop-up, allowing them to review their selected items, adjust quantities, or remove items at any point. The cart will display a running subtotal, exclusive of delivery fees and taxes, which will be calculated at checkout.
- **Streamlined Checkout and Account Creation:** The checkout process is a strategically

designed funnel. The decision to allow unauthenticated browsing and delay account creation until the point of purchase is a deliberate strategy to maximize user acquisition. It lowers the initial barrier to entry, enabling potential customers to fully engage with the platform's core offering—the unique food from local chefs—before being asked for any personal commitment. This aligns with a natural user journey of exploration followed by decision-making.[7] When the user has a cart full of desired items, the perceived value is high, making the "cost" of creating an account (a moment of time) negligible and significantly increasing the likelihood of conversion.
Upon clicking "Proceed to Checkout," the system will check for user authentication.
    - **For Guest Users:** A simple, unobtrusive form will appear, requiring only an email address and a password to create an account. This minimalist approach ensures the final step is as frictionless as possible.
    - **Checkout Flow:** Once authenticated, the user will be guided through a single-page or simple multi-step checkout process to provide a delivery address, a contact phone number, and payment details.
- **Secure Payment Gateway Integration:** The platform will integrate with a PCI-compliant payment gateway such as Stripe or PayPal.[9] This ensures that all transactions are secure and that sensitive credit card information is never stored on Teka's servers. The integration will support major credit and debit cards, and for an enhanced user experience, will also offer express checkout options via mobile wallets like Apple Pay and Google Pay.[4]

2.3 The Authenticated Client Dashboard

After creating an account and placing an order, the client gains access to a personalized dashboard, which serves as their central hub for managing all platform activities.

- **Real-Time Order Tracking:** This is a critical feature for managing customer expectations and reducing support inquiries. The dashboard will feature a section for "Current Orders" displaying the real-time status of each order. The status will be updated automatically as the chef progresses through the fulfillment process, with clear stages such as Order Placed, Confirmed by Chef, In the Kitchen, Ready for Pickup/Delivery, Out for Delivery, and Delivered.[4]
- **Order History:** A comprehensive archive of all past orders will be available to the client.[4] Each entry will show the chef, the items ordered, the total cost, and the date of the order. This feature provides convenience, allowing a client to easily re-order a favorite meal with a single click.
- **Profile and Address Management:** Clients will have the ability to manage their personal information, including their name, email address, and password. They can also save and manage multiple delivery addresses (e.g., "Home," "Work") to expedite the checkout

process for future orders.
- **Reviews and Ratings System:** Following the completion of an order (marked as Delivered), the client will receive a prompt (via email and on their dashboard) to leave a rating (typically on a 1-5 star scale) and a written review for the chef. This user-generated content is the lifeblood of the platform's trust framework, providing essential social proof for other clients and valuable feedback for chefs.[2]

Section 3: The Chef Empowerment Portal

The Chef Portal is the operational core for Teka's supply side. It is designed not merely as a set of administrative tools, but as an empowering "business-in-a-box" platform. Its purpose is to provide chefs with everything they need to build their brand, manage their operations, and grow their business, allowing them to focus on their culinary craft.

3.1 Onboarding and Profile Configuration

The initial experience for a chef joining the platform must be welcoming, efficient, and professional, instilling confidence from the very first step.

- **Seamless Onboarding:** A guided, step-by-step onboarding wizard will walk new chefs through the registration process. The goal is to minimize friction while collecting all necessary information, enabling a chef to go from signup to live on the platform within 24-48 hours.[2] This process will include:
  - **Account Creation:** Basic personal and business information.
  - **Verification:** Secure upload of identity documents and any locally required food handling certifications or business licenses to ensure compliance and build platform integrity.[2]
  - **Payment Setup:** Integration with Stripe Connect or a similar service to securely link the chef's bank account for automated payouts, without Teka ever handling sensitive banking details directly.[3]
- **Dynamic Profile Management:** The chef's profile is their digital storefront and primary marketing tool. The portal will provide an intuitive interface for managing all aspects of their public-facing presence.
  - **Personal Story:** A rich text editor for chefs to share their personal narrative—their culinary background, their passion for cooking, and the stories behind their signature dishes. This is a key differentiator that fosters a personal connection with clients.[1]

    - **Media Uploads:** Easy-to-use tools for uploading a professional profile picture, a header image for their page, a gallery of high-quality food photography, and—as a key feature—short video clips. These videos can showcase their cooking process, their kitchen environment, or a personal introduction, adding a dynamic and trustworthy element to their profile.
    - **Social Media Integration:** The profile will support two forms of social media integration as specified. Firstly, chefs can add direct links to their external social media pages (e.g., Instagram, Facebook, TikTok). Secondly, the profile will feature an iframe component. This allows a chef to embed their Instagram feed directly onto their Teka profile, enabling clients to browse their latest posts and stories without navigating away from the platform, thereby increasing engagement and time-on-site.

It is critical to recognize that the iframe feature, while powerful for engagement, introduces a significant business risk. A chef's social media feed may contain calls to action that encourage clients to transact outside the Teka ecosystem (e.g., "DM me to order," links to personal payment methods), leading to platform leakage and lost commissions. This technical feature must therefore be coupled with a robust operational policy. The Chef Terms of Service must explicitly prohibit any form of off-platform solicitation. Furthermore, the administrative portal (Section 4) should include tools for monitoring or flagging profiles whose embedded content violates these terms. This illustrates the essential interplay between product features and platform governance in a marketplace model.

3.2 Business Operations Management

This suite of tools gives chefs granular control over their menu, schedule, and availability, allowing them to run their business on their own terms.

- **Menu Management (CRUD via Modals):** The portal will feature an intuitive dashboard for managing the chef's menu. All Create, Read, Update, and Delete (CRUD) operations will be executed via modal pop-ups, as per the design specification. This creates a fluid, single-page application experience, avoiding disruptive page reloads. When creating or editing a menu item, the chef will have fields for:
    - Item Name
    - Detailed Description
    - Price
    - High-Resolution Photo Upload
    - List of Ingredients and Allergen Information
    - Categorization (e.g., Appetizer, Main Course, Dessert)
    - Customization Options (e.g., "Spice Level: Mild, Medium, Hot").
- **Availability and Scheduling:** Flexibility is key for home chefs. The portal will provide

powerful yet simple scheduling tools.

  - **Working Hours:** A visual, calendar-based interface for setting recurring weekly availability (e.g., Wednesdays and Fridays from 5:00 PM to 9:00 PM).
  - **Availability Toggle:** A prominent, one-click toggle switch on the main dashboard that allows a chef to instantly set their status to "Online" (accepting orders) or "Offline" (temporarily unavailable). This is essential for managing real-time capacity, handling unexpected events, or taking a break during a busy service.
  - **Date-Specific Overrides:** The ability to block out specific dates for vacations, holidays, or personal time, which will override their standard weekly schedule.

3.3 Order Fulfillment Dashboard

This is the operational command center where chefs manage the entire lifecycle of an order from acceptance to completion.

- **New Order Notifications:** When a new order is placed by a client, the chef's dashboard will trigger an immediate, unmissable notification, which may include both a visual alert and an audible chime to ensure no order is missed.
- **Order Management Pipeline:** Orders will be displayed in a clear, actionable format, such as a Kanban-style board or a filterable list. This allows the chef to visually track orders as they move through the fulfillment stages:
  - **New:** Awaiting chef confirmation.
  - **Accepted:** Confirmed by the chef and in the queue.
  - **In Progress:** The meal is currently being prepared.
  - Ready for Pickup/Delivery: The order is packaged and awaiting dispatch.
    Each status update made by the chef will automatically trigger a corresponding update in the client's order tracking view.
- **Accept/Reject Functionality:** Chefs must retain control over their workflow. For each new order, they will have the option to "Accept" or "Reject" it within a predefined time window (e.g., 15 minutes).[4] A rejection might be necessary due to running out of a specific ingredient or reaching maximum production capacity. If an order is rejected, the client is notified immediately, and the payment authorization is voided.

3.4 Analytics and Payouts

Providing clear, accessible data on business performance and finances is crucial for empowering chefs and fostering a professional mindset.

- **Sales Dashboard:** A simple and visual analytics dashboard will provide at-a-glance insights into business performance.[4] It will feature key metrics such as:
  - Total Sales (by day, week, month)
  - Number of Orders
  - Average Order Value
  - Top-Selling Menu Items
  - Total Earnings after platform commission.
- **Payout Management:** This section provides full transparency into the chef's finances. It will display:
  - Current Account Balance (funds processed but not yet paid out).
  - A schedule of upcoming automatic payouts (e.g., weekly direct deposits).[3]
  - A complete history of all past payouts.
  - An option for manual withdrawal of available funds, subject to a processing period.[3]

Section 4: The Administrative Control Platform

The Administrative Platform is the central nervous system of the Teka marketplace. It is not merely a back-office tool but a strategic hub for risk mitigation, quality control, community management, and business intelligence. Built upon the robust and extensible Django Admin interface, this portal provides the necessary levers to manage the platform's health, ensure its integrity, and guide its growth. The features within this portal are designed to directly address the unique operational and legal challenges inherent in a home-chef marketplace, such as food safety, service consistency, and legal compliance.[2]

4.1 User and Content Management

Maintaining a safe, trusted, and high-quality community is paramount. The admin portal provides the tools to curate and manage the platform's user base.

- **Chef Account Management:** This module oversees the entire chef lifecycle, from application to ongoing performance management. It is the platform's primary tool for quality assurance. Administrators will be able to:
  - Review and process new chef applications.
  - Verify submitted documents, such as identity verification and food handling certificates.
  - Approve or reject applications with templated email notifications.

    - View a comprehensive profile for each chef, including their performance metrics (average rating, order acceptance rate, on-time fulfillment).
    - Suspend or deactivate chef accounts in response to policy violations or consistently poor reviews, thereby protecting the platform's reputation.
- **Client Account Management:** Provides standard administrative capabilities for managing the client user base, including the ability to:
    - View client profiles and order histories.
    - Assist with customer support issues, such as password resets or account inquiries.
    - Manage accounts in cases of fraudulent activity or violations of terms of service.
- **Content Moderation:** A system for reviewing and moderating all user-generated content to ensure it aligns with Teka's community guidelines. This includes:
    - Reviewing chef profile text, photos, and videos for appropriateness.
    - Moderating client reviews to remove spam, abusive language, or content that violates privacy, while preserving genuine feedback.

4.2 Marketplace Oversight

These tools provide a high-level view of platform operations, enabling administrators to monitor activity in real-time and intervene when necessary.

- **Platform-wide Order Dashboard:** A real-time, centralized dashboard displaying all active orders across the entire platform. This view is essential for troubleshooting and support, allowing administrators to filter orders by status (placed, in_progress, etc.), by chef, or by geographic region.
- **Dispute Resolution System:** A structured ticketing system for managing conflicts between clients and chefs. When a client or chef reports an issue (e.g., a problem with order quality, a no-show delivery), a dispute ticket is created. This system allows administrators to:
    - View all communication between the involved parties.
    - Act as a neutral mediator to find a resolution.
    - Process refunds or issue platform credits as needed.
    - Track dispute patterns to identify problematic users or areas for platform improvement. This system is a direct mechanism for managing liability and maintaining customer trust when exceptions occur.
- **Commission and Fee Management:** A configuration panel where platform owners can set and adjust the financial parameters of the marketplace. This includes setting the platform's commission percentage on each order or defining alternative models, such as a tiered monthly subscription fee for chefs.[3]

4.3 Analytics and Reporting

Data-driven decision-making is critical for sustainable growth. The admin portal will feature a suite of analytics and reporting tools for comprehensive business intelligence.

- **Key Performance Indicator (KPI) Dashboard:** A high-level dashboard that provides a snapshot of the platform's overall health, tracking essential business metrics such as:
  - **Growth:** New client signups, new chef applications, and activated chefs.
  - **Engagement:** Total orders processed (daily, weekly, monthly), Gross Merchandise Volume (GMV).
  - **Financials:** Total platform revenue (commissions).
  - **Retention:** Client and chef churn rates.
- **Financial Reporting:** Tools dedicated to managing the platform's financial operations.[12] Administrators can:
  - Generate reports on total commissions earned over specific periods.
  - Oversee and verify the automated payout processes to chefs.
  - Reconcile transaction records from the payment gateway (e.g., Stripe) with the platform's internal order database.

Section 5: System Architecture and Intercommunication

This section details the technical blueprint for the Teka platform, outlining the chosen technology stack, database architecture, data flow logic, and the Application Programming Interface (API) that will serve as the communication backbone for all client applications, including the web portal and future mobile apps.

5.1 Core Backend Architecture (Django)

The backend is the engine of the platform, responsible for all business logic, data processing, and user management.

- **Technology:** The platform will be built using the **Django** framework in **Python**. Django is selected for its "batteries-included" philosophy, which provides a robust, secure, and scalable foundation. Its powerful Object-Relational Mapper (ORM), built-in security features (protecting against common threats like CSRF and XSS), and comprehensive, extensible admin interface make it an ideal choice for rapid and reliable development.[9]

- **Database: PostgreSQL** is the recommended relational database. It is known for its reliability, data integrity, and powerful feature set. Crucially, its robust support for geospatial data types and queries (via PostGIS) is essential for implementing the platform's core location-based chef discovery features efficiently.[9]
- **Database Schema Overview:** The data will be structured around a set of core Django models that represent the key entities in the marketplace:
    - User: Extending Django's built-in AbstractUser model to include roles (client, chef, admin).
    - ChefProfile: A model with a one-to-one relationship to a User with the chef role. It will store all chef-specific information, such as bio, profile picture URL, video URLs, social media links, location (latitude/longitude), and availability status (a boolean field).
    - MenuItem: A model with a foreign key to ChefProfile, containing details like name, description, price, photo URL, and ingredients.
    - Order: The central transactional model, with foreign keys to the client (User) and the ChefProfile. It will store the delivery address, total cost, and the current order status (e.g., placed, confirmed, in_progress, ready, delivered, cancelled).
    - OrderItem: A linking table between Order and MenuItem, storing the quantity and any selected customizations for each item in an order.
    - Review: A model with foreign keys to an Order, the client (User), and the ChefProfile, storing the star rating and review text.

5.2 Data Flow and Communication

The three portals (Client, Chef, Admin) are not separate applications but distinct views into a single, centralized system. They communicate and stay synchronized by reading from and writing to the central PostgreSQL database, primarily via the GraphQL API.

- **The Order Lifecycle Data Flow:** The process of placing and fulfilling an order illustrates the system's interconnectedness [12]:
    1. **Order Placement:** A client on the web or mobile app finalizes their cart and submits a createOrder request to the GraphQL API.
    2. **Backend Processing:** The Django backend receives the request, validates the data, creates a new Order instance in the database with a status of placed, and processes the payment through the integrated payment gateway.
    3. **Chef Notification:** Upon successful order creation, the backend triggers a real-time event. This can be implemented using a technology like **Django Channels** with

WebSockets. The event is pushed to the specific Chef's dashboard, causing a new order notification to appear instantly without requiring a page refresh.

4. **Status Updates:** The chef updates the order's status on their dashboard (e.g., from placed to confirmed). This action sends an updateOrderStatus request to the GraphQL API.
5. **Database Update:** The backend updates the status field of the corresponding Order object in the database.
6. **Client Notification:** This database change triggers another real-time event, which is pushed via WebSockets to the client's dashboard, updating their order tracking interface to reflect the new status (e.g., "Your order has been confirmed!").
7. **Admin Oversight:** Throughout this process, an administrator can view the current state and history of the Order object at any time through the Django Admin interface.

5.3 The GraphQL API Layer

To support both the web application and a native mobile application, a flexible and efficient API is required. GraphQL is the preferred choice for this architecture.

- **Rationale:** GraphQL is selected over traditional REST APIs for several key reasons. Its primary advantage is the ability for the client to request exactly the data it needs in a single network request, and nothing more. This eliminates the problems of over-fetching (receiving unnecessary data) and under-fetching (needing to make multiple requests to gather all required data), which is particularly beneficial for mobile applications on potentially slow or unreliable networks.[15] Furthermore, GraphQL's strongly typed schema serves as comprehensive, self-documenting contract between the frontend and backend, improving developer productivity and reducing integration errors.[16]
- **Implementation:** The GraphQL API will be implemented in Django using the **graphene-django** library. This library provides seamless integration with the Django ORM, allowing for the rapid creation of GraphQL types directly from Django models (DjangoObjectType) and simplifying the implementation of queries and mutations.[15]
- **Schema Design:** The API schema will be organized around core queries (for reading data) and mutations (for writing or modifying data).
  - **Core Queries:**
    - chefsNearMe(lat: Float!, long: Float!, radius: Int):
    - chefProfile(id: ID!): ChefType
    - menuForChef(chefId: ID!):
    - myOrders: (Authenticated clients only)
    - chefOrders: (Authenticated chefs only)
  - **Core Mutations:**
    - registerClient(email: String!, password: String!): AuthPayload

    - createOrder(chefId: ID!, items: [OrderItemInput!]!): OrderType
    - updateOrderStatus(orderId: ID!, status: String!): OrderType
    - submitReview(orderId: ID!, rating: Int!, text: String): ReviewType
    - updateChefAvailability(isAvailable: Boolean!): ChefType
    - updateMenuItem(itemId: ID!, name: String, price: Float): MenuItemType

5.4 Frontend Implementation Strategy

The frontend strategy balances rapid development with a modern, responsive user experience.

- **Web Application:** The primary web portal will be built using standard **Django Templates**. This server-side rendering approach is SEO-friendly, secure, and leverages Django's built-in templating engine for fast development.
- **Styling:** The **Bootstrap** CSS framework will be used to create a responsive, mobile-first design system. This ensures a consistent look and feel across the platform and significantly reduces the time required for custom styling.
- **Dynamic UI (Modal-Based CRUD):** To meet the requirement for a dynamic user experience without building a full single-page application (SPA), client-side **JavaScript** will be used to enhance the Django templates. When a user needs to perform a CRUD operation (e.g., editing a menu item), the flow will be as follows:
  1. The user clicks an "Edit" button.
  2. A JavaScript event handler prevents the default page navigation.
  3. An AJAX (fetch) request is sent to a dedicated Django view or API endpoint to retrieve the data for the specific item.
  4. The JavaScript populates a Bootstrap modal with a form containing the retrieved data.
  5. Upon submission, another AJAX request sends the updated form data to the backend to be saved.
  6. On success, the modal is closed, and the UI is updated in place to reflect the change, all without a full page reload. This approach, potentially augmented with a lightweight library like HTMX or Alpine.js, provides a modern user experience while retaining the simplicity and development speed of a server-rendered application.

Table 5.1: Technology Stack Summary

Component	Technology	Rationale
Backend Framework	Django (Python)	Robust, secure, scalable, and includes a powerful ORM and admin interface for rapid development.[13]
Database	PostgreSQL	High-performance, reliable relational database with strong support for geospatial queries (PostGIS).[9]
API Layer	GraphQL (with graphene-django)	Enables efficient data fetching for mobile clients, reduces network requests, and provides a strongly typed schema.[15]
Web Frontend	Django Templates + JavaScript	Server-side rendering for SEO and simplicity, enhanced with JavaScript for dynamic UI elements like modals.[13]
Styling	Bootstrap	A comprehensive, mobile-first CSS framework for building responsive and consistent user interfaces quickly.
Real-time Comms	Django Channels (WebSockets)	Provides real-time notifications for new orders and status updates to client and chef dashboards.
Hosting	AWS / Google Cloud / Azure	Scalable, reliable, and secure cloud infrastructure with a wide range of managed services.[9]

Payment Gateway	Stripe (Connect)	Secure, PCI-compliant payment processing and simplified payout management for chefs.[3]

Table 5.2: Core GraphQL API Endpoint Overview

Operation Type	**Endpoint Name**	**Description**	**Key Arguments**	**Returns**
Query	chefsNearMe	Fetches a list of available chefs within a specified geographic radius.	lat, long, radius	List
Query	chefProfile	Retrieves the complete public profile for a single chef, including their menu.	id	ChefType
Query	myOrders	Fetches the order history for the currently authenticated client.	(None)	List
Mutation	createOrder	Places a new food order from a specific chef with a list of items.	chefId, items, deliveryAddress	OrderType

Mutation	updateOrderStatus	Allows a chef to update the status of an order they have received.	orderId, status	OrderType
Mutation	submitReview	Allows a client to submit a rating and review for a completed order.	orderId, rating, text	ReviewType
Mutation	updateChefAvailability	Toggles the chef's online/offline status for accepting new orders.	isAvailable (Boolean)	ChefType

Conclusion

The Teka platform, as detailed in this blueprint, is strategically positioned to capture the growing market for authentic, homemade food. Its success hinges on a tripartite foundation: a client experience optimized for frictionless discovery and trust; a chef portal designed for empowerment and operational simplicity; and a robust, scalable technical architecture that can support both a modern web application and future mobile expansion.

The decision to build on a Django and PostgreSQL backend provides a secure and reliable core, while the adoption of a GraphQL API is a forward-looking choice that prioritizes mobile performance and developer efficiency. The frontend strategy balances rapid development with a dynamic user experience, leveraging Django templates enhanced with JavaScript for a responsive and interactive interface.

Beyond the technology, the platform's long-term viability is intrinsically linked to its ability to foster a genuine community. By prioritizing features that build trust—such as detailed chef stories, transparent reviews, and seamless communication—Teka can create a loyal user base and a network of empowered culinary entrepreneurs. The administrative tools are designed not just for management but as a crucial system for risk mitigation and quality assurance, safeguarding the platform's integrity as it scales.

This document serves as a comprehensive guide for the development of the Teka platform. By adhering to the principles and specifications outlined herein, the development team can construct a powerful, differentiated, and sustainable marketplace that redefines the local food landscape.

Works cited

1. Seeking Feedback on New Homemade Food Marketplace App : r/Startup_Ideas - Reddit, accessed on October 29, 2025, https://www.reddit.com/r/Startup_Ideas/comments/1d08b7r/seeking_feedback_on_new_homemade_food_marketplace/
2. Home Chefs in Your Food Delivery App- A Market Opportunity, accessed on October 29, 2025, https://oyelabs.com/home-chefs-in-your-food-delivery-app-an-opportunity/
3. Cookin - The easiest way to sell your food online, accessed on October 29, 2025, https://www.cookin.com/
4. On-Demand Home Cooked Meals Delivery Apps Development, accessed on October 29, 2025, https://www.flightslogic.com/on-demand-home-cooked-meals-delivery-apps-development.php
5. Online Food Ordering System | Use Case Diagram Template, accessed on October 29, 2025, https://online.visual-paradigm.com/diagrams/templates/use-case-diagram/online-food-ordering-system-/
6. Home Chef: Meal Delivery Service - Fresh Weekly Meal Kit Delivery, accessed on October 29, 2025, https://www.homechef.com/
7. Food ordering app - User journey diagram example - Gleek.io, accessed on October 29, 2025, https://www.gleek.io/templates/food-ordering-app
8. Food Ordering App Graphical Flow - Swayam Infotech, accessed on October 29, 2025, https://www.swayaminfotech.com/food-ordering-app-graphical-flow/
9. Tech Stack for Food Delivery App Development - Top 10+ Tools - Enatega, accessed on October 29, 2025, https://enatega.com/tech-stack-for-food-delivery-app-development/
10. Restaurant Technology: Picking the Right Tech Stack - Uber Eats for Merchants, accessed on October 29, 2025, https://merchants.ubereats.com/us/en/resources/articles/restaurant-technology-tech-stack/
11. Is Homemade the Right Platform for Your Home Food Business? Honest Review 2025, accessed on October 29, 2025, https://www.homemadechefs.com/post/is-homemade-the-right-platform-for-your-home-foodbusiness-an-honest-review-for-the-netherlands
12. DFD for Food Ordering System - GeeksforGeeks, accessed on October 29, 2025,

https://www.geeksforgeeks.org/software-engineering/dfd-for-food-ordering-system/
13. How to Create an Online Food Ordering System? 7 Simple Steps, accessed on October 29, 2025, https://enatega.com/how-to-create-an-online-food-ordering-system/
14. Understanding Data Flow Diagram for Food Ordering System - Hashmato, accessed on October 29, 2025, https://hashmato.com/data-flow-diagram-food-ordering-system/
15. How to Build a GraphQL API in Django - freeCodeCamp, accessed on October 29, 2025, https://www.freecodecamp.org/news/how-to-build-a-graphql-api-in-django/
16. Working with Django and GraphQL: A Beginner's Guide | by Farad Alam - Medium, accessed on October 29, 2025, https://medium.com/@farad.dev/working-with-django-and-graphql-a-beginners-guide-6556f7c3d7a7
17. Django Graphene: The REST to GraphQL Migration Guide - FullStack Labs, accessed on October 29, 2025, https://www.fullstack.com/labs/resources/blog/django-graphene-from-rest-to-graphql