# Prompt for Codex: Ultra-Modern Bakery Website (Bilingual, GTA VI-Inspired Style)

**IMPORTANT:** All content and visuals must be original – *do not copy* any text, images, or code from Rockstar Games or any other source. Use the Grand Theft Auto VI website only as design inspiration (style and vibe), **without** referencing it or any Rockstar intellectual property in the code or text. Use only royalty-free or original images (e.g. from free stock libraries or placeholders) for the bakery. The website’s primary theme color should be a rich **purple** (instead of GTA VI’s colors), giving the site a unique identity while maintaining a flashy, modern aesthetic.

## Project Overview

Create a **full multi-section website** for a fictional local bakery that has a long-standing family tradition. The site should be an **“ultra” modern interactive web experience** – visually striking, highly polished, and reminiscent of a AAA game promo site in style (inspired by Rockstar’s GTA VI site). This means the design should be bold and immersive, with dynamic content, animations, and a sleek user interface. The website must be **bilingual**, supporting both **English and Serbian** languages with a toggle switch for language selection. All text content (bakery name, headings, descriptions) should be provided in both languages (you can invent appropriate translations for Serbian content). The final product will consist of **HTML, CSS, and JavaScript** code (and image assets) that implements the entire site. The code should be well-structured, clean, and commented where helpful.

## Design & Thematic Style

* **Color Scheme:** Use a **dark theme** background (e.g. black or deep charcoal) to evoke a cinematic, high-contrast look. Overlay this with **purple accents** (e.g. #800080 or other shades of purple) for highlights, buttons, and interactive elements. The purple should be vivid to create a neon-like glow effect, reminiscent of a neon sign or nightlife (fitting a Vice City-inspired aesthetic, but applied to a bakery context). Complementary colors (white or light gray text on dark backgrounds, lighter purple or pink highlights if needed) should ensure good contrast and readability.
* **Typography:** Choose modern, bold fonts (preferably from Google Fonts or other free sources). For example, use a **decorative or stylized display font** for the bakery’s logo/name and section titles (to give a unique identity, possibly a cursive or retro neon-style font to echo a sign). Use a clean **sans-serif font** for body text and smaller UI elements, for readability (similar to how GTA VI site uses clean fonts for content). Ensure the fonts you choose support Serbian Cyrillic characters as well as English (since Serbian content may use Cyrillic script if appropriate, or you can use Latin script for Serbian if preferred – be consistent).
* **Imagery & Media:** Integrate high-quality visuals of bakery goods to make the site engaging. Use **copyright-free images** of bread, pastries, and bakery scenes. Ideally, incorporate:
* A **hero background image or short video** loop showing an appetizing baking scene (e.g. close-up of bread coming out of an oven, flour being dusted, or a display of pastries). This can autoplay on mute in the hero section for a dynamic effect (similar to game sites that have looping video backgrounds). If video is not available, a static high-resolution image with a subtle animated overlay (like a slow zoom or CSS animation) is acceptable.
* A **gallery or slideshow** of various baked goods (loaves, cakes, pastries) in a dedicated section. These images should be visually consistent and free to use. If real images cannot be embedded directly by the AI, use placeholders or references (e.g. image paths or example URLs from Unsplash/Pexels). Provide descriptive alt text for each image (important for accessibility).
* **Layout and Style:** Follow a **slick, one-page** design (with multiple sections) or a few linked pages, using full-width sections that showcase content boldly (like a promotional site). Each section should have a distinct feel but cohesive style (consistent color scheme and fonts). Use large heading text, immersive visuals, and **stylistic elements** inspired by the GTA VI site (for example, section headers that resemble the “Explore [Location]” style but for bakery content, or hover effects and transitions that feel interactive and modern). However, *do not use any game-related imagery* – all content should be about the bakery. Use CSS effects like gradients, glows, or transitions to enhance the purple neon theme (e.g., glowing borders or text shadows on hover for buttons in purple).

## Content Structure & Sections

Design the website with clear sections, each fitting on a typical screen viewport (for a scrolling single-page experience). Ensure smooth navigation between sections (either via scroll or menu links). The main sections to include are:

1. **Header & Navigation:** A persistent **top navigation bar** (fixed to top on scroll). This should include the bakery’s **logo/name** on the left and navigation links to each major section (e.g., Home, About, Products, Gallery, Contact). Also include the **language toggle control** on the right side of the nav (details on language toggle below). The nav bar should have a transparent or dark background initially, and if possible, switch to a solid background or add a slight shadow once the page scrolls (to ensure it remains visible over content, similar to how modern sites handle sticky headers). On mobile view, implement a responsive menu (e.g., a hamburger icon that can toggle the nav links).
2. **Hero Section (Home):** This is the top section that creates a strong first impression. Use a **full-screen hero** design with a dark overlay on a background video or image of bakery items. Overlay the bakery’s **name and tagline** prominently. For example:
3. Bakery Name: *“Purple Oven Bakery”* (you can invent a catchy name; make sure to also provide a Serbian version if the name would differ or a translated tagline).
4. Tagline example (English): *“Baking Tradition Meets Modern Flavor”*; (Serbian: e.g., “**Tradicija pečenja u modernom ruhu**”). The tagline should emphasize heritage and quality.  
   Use large, stylish font for the name and a slightly smaller, contrasting font for the tagline. Optionally, include a brief **call-to-action (CTA)** button in the hero, such as “View Our Breads” or “Explore Menu” (and a Serbian version, e.g. “Pogledaj ponudu”), which scrolls down or links to the Products section. Also, consider a small **scroll indicator** (like a down arrow) to encourage users to scroll for more content (this element can pulsate or bounce subtly).
5. **About Us / Story Section:** A section that tells the **story of the bakery**, highlighting its *family tradition and heritage*. This section should have a warm, inviting feel. Include a heading like “Our Story” (Serbian: “Naša Priča”) and 1-3 paragraphs of descriptive text (in both languages) that you will **invent**. For example, describe that the bakery was founded decades ago by the current owner’s grandparents, has been using secret family recipes, won local awards, etc. Use rich, engaging language to make the story interesting. You can include a photo of the bakery or the founders here (placeholder if necessary, with alt text like “Founders of the bakery” or “Old bakery shop in 19xx”). Ensure the text is wrapped in appropriate <p> tags and is not too lengthy for design balance (break into multiple paragraphs if needed for readability).
6. **Products / Specialties Section:** A showcase of the bakery’s key **products or specialties**. This can be presented in an attractive grid or slider format. Include a section title like “Our Specialties” or “What We Bake” (Serbian: e.g., “Naši Specijaliteti”). For each featured product category or item, provide:
7. **Name of the item** (e.g., Sourdough Bread – with Serbian equivalent “Domaći hleb od kiselog testa”), possibly styled as a subtitle.
8. **Image**: a high-quality image of that product (with alt text, e.g., “Crusty sourdough loaf”). Use a few different items: bread loaf, pastries like croissants or burek, cakes or cookies – to show variety.
9. **Description**: one or two sentences describing what makes it special (e.g., “A rustic sourdough loaf fermented for 24 hours for rich flavor and texture.” / “Rustični hleb od kiselog testa fermentisan 24 sata za bogat ukus i teksturu.”). Invent appetizing descriptions.  
   Arrange these in a visually appealing layout – for example, a three-column grid on desktop (each column with an image and text beneath or beside it), that collapses to single column on mobile. You can also implement a **carousel/slider** if that enhances the design (user can click through slides of different products). Ensure text is legible on any overlay on images (use a slight overlay or contrasting caption background if placing text over images).
10. **Gallery Section:** An interactive **image gallery** showing various scenes and products from the bakery. This section can be grid-based thumbnails that, when clicked, open a larger view (you can use a lightbox effect or simply link to the image in a new tab). Title the section “Gallery” (Serbian: “Galerija”). Include at least 6–8 images: examples might be close-ups of pastries, the bakery storefront, customers inside the shop, bakers at work, etc. Since the AI cannot actually provide real images, use placeholders (e.g., <img src="assets/img/bread1.jpg" alt="Fresh bread on shelf"> etc., and the user will later replace these with actual image files). The key is to ensure the HTML structure is there for an image grid. Use consistent image sizing or aspect ratio for a clean look, and add some CSS spacing (margin/padding) between images or a nice hover effect (like slight enlarge or shadow on hover).
11. **Contact / Visit Us Section:** A section for **contact information and call-to-action**. Title could be “Contact Us” or “Visit Us” (Serbian: “Kontakt” or “Posetite Nas”). Content to include:
12. **Address and Map:** Provide a fictitious address for the bakery (e.g., “123 Baker Street, Novi Sad, Serbia”) and perhaps embed a Google Map or an image of a map (if possible, use an <iframe> with Google Maps embed for realism, otherwise a placeholder).
13. **Phone number and email:** e.g., “Phone: +381 21 555 1234”, “Email: info@purpleoven.rs” (make sure to include a mailto: link for the email in HTML).
14. **Business hours:** list the opening hours on days of the week.
15. Optionally, a **contact form** for users to send a message (fields: name, email, message). Since this is a static site, the form won’t actually submit to a server in this context, but you can include the form HTML and perhaps a JavaScript alert or console log on submit as a placeholder. Label the inputs clearly (and provide Serbian labels too when in Serbian mode).  
    If using a form, include basic validation for required fields via HTML5 attributes or a bit of JS. The contact section can have a simpler background (maybe a dark purple solid background to contrast with the gallery images above, with light text) and clear, easy-to-read fonts.
16. **Footer:** A footer that remains at the bottom with small text. It should include the bakery name (or logo) again and a copyright notice (e.g., “© 2025 Purple Oven Bakery. All rights reserved.” – provide a Serbian version as well, e.g., “Sva prava zadržana.”). You might also list a couple of social media icon links (use font-awesome or SVGs for a Facebook/Instagram icon with dummy links), as many modern sites do. If using external icons or fonts, ensure they are referenced properly (and are free to use). The footer can have a slightly different shade of the theme (maybe a bit lighter or darker) to distinguish it from the main content, but should align with the overall style.

## Language Toggle (Bilingual Implementation)

Implement a **language toggle** so that the site’s text content can switch between English and Serbian seamlessly, without reloading the page. This could be a dropdown (with options “EN” and “SR”) or two buttons, or a toggle switch UI in the header (for example, a simple <select> element or two flag icons). The control should be intuitive (label with aria-label for accessibility, e.g., “Select Language”). Use the following approach to handle multilingual content:

* **Content Storage:** Prepare all textual content in both languages. A convenient method is to use a JavaScript object or JSON structure that holds translations for each text element. For example, create an object like:
* const translations = {  
   en: {   
   heroTitle: "Purple Oven Bakery",  
   heroTagline: "Baking Tradition Meets Modern Flavor",  
   navAbout: "About Us",  
   // ... (all other text strings)  
   contactPhoneLabel: "Phone",  
   contactHoursLabel: "Working Hours",  
   // etc.  
   },  
   sr: {  
   heroTitle: "Pekara Purpurna Peć",  
   heroTagline: "Tradicija pečenja u modernom ruhu",  
   navAbout: "O nama",  
   // ... Serbian translations for all strings  
   contactPhoneLabel: "Telefon",  
   contactHoursLabel: "Radno vreme",  
   // etc.  
   }  
  };
* Each key corresponds to a piece of UI text (navigation labels, section headings, paragraph texts, form labels, etc.). Use meaningful keys to map to elements (you may use data-i18n attributes in HTML to mark elements and match keys, or direct references via IDs).
* **Applying Language:** Write a JavaScript function to switch language. For instance, when the user selects “SR” from the language dropdown, the script should iterate through all text elements and replace their inner text with the Serbian version from the translations object. If using data-i18n attributes, you can store a key on each element (e.g., <h2 data-i18n="aboutTitle">About Us</h2>) and on toggle, set element.textContent = translations[selLang][element.dataset.i18n]. Alternatively, maintain references to specific element IDs and change their text accordingly (document.getElementById("hero-tagline").innerText = translations[selLang].heroTagline, etc.). Ensure **all visible text** on the site changes when language switches, including navigation menu items, section headings, paragraphs, form labels, button text, footer text, etc.
* **Default Language:** The site can default to one language (e.g., English) on first load. Optionally, you can remember the user’s choice using localStorage (store a key like selectedLang and on page load, initialize the content to that language if present). This is not required but is a nice touch (the reference ElectroGroup site does this).
* **Direction/Alphabet:** For Serbian, you can use either Latin or Cyrillic script. If using Cyrillic for authenticity, ensure the font supports it. (For simplicity, using Serbian Latin script is acceptable, as it is commonly used online in Serbia.) Adjust the <html lang=""> attribute dynamically if needed (e.g., set lang="sr" for Serbian, lang="en" for English for accessibility and SEO).
* **Testing:** Verify that toggling the language updates all text and does not break the layout (some strings might be longer in one language, so design should accommodate that). Also ensure that the toggle UI itself (like the dropdown or buttons) is easy to use and visible in both language states.

## Animations & Interactive Features

To give the site an **“ultra-modern” feel**, implement engaging but not overwhelming animations and effects, inspired by high-end promo websites:  
- **Preloader Splash:** Create a preloader screen that appears while the site assets load (especially if using large images/videos). For example, a centered bakery logo or icon (perhaps an oven or bread icon) that might be shown with a loading spinner or a simple fade-in animation. You can animate the logo or a progress bar for 2-3 seconds, then transition to the homepage once content is ready. The preloader can have a plain background or a subtle animation (e.g., an overlay color with the logo). Use JavaScript (e.g., setTimeout) to remove the preloader div after a short delay or once the main content is fully loaded.  
- **Hero Animations:** In the hero section, consider animating the entrance of text. For example, the bakery name could appear letter-by-letter with a slight delay (creating a typing or fading-in effect), followed by the tagline sliding in or fading. The CTA button can also gently fade in or pop in after a delay. Use CSS keyframes or JS for these timed effects. Also, if a video background is used, ensure it plays smoothly (use the autoplay muted loop playsinline attributes for mobile compatibility if it’s an HTML5 video). If using just an image, perhaps add a **parallax effect** on scroll (where the background image moves slightly slower than foreground content) to create depth – this can be done with CSS background-attachment: fixed for simple parallax or a bit of JS for more control.  
- **On-Scroll Reveal:** For sections below the hero, implement a subtle **on-scroll reveal** animation. For instance, as the user scrolls to the About or Products section, fade in the content or slide it in from the bottom. You can add a CSS class like “.reveal” that has opacity: 0 and a transform (like translateY(20px)), and then use a script to add a class “.reveal--visible” when the section scrolls into view (setting opacity to 1 and transform to none with a transition). This adds a polished touch so content doesn’t just abruptly appear. Ensure to trigger these only once per section to avoid repetition on scroll up/down (or make it repeatable if you prefer that effect).  
- **Interactive Elements:** Make interactive elements respond to user actions: - Navigation links: highlight the current section in the nav as active when scrolling (this can be done by updating classes on scroll or using IntersectionObserver to detect which section is in view). Also, smooth-scroll to sections when nav links are clicked (use window.scrollTo with behavior smooth, or CSS scroll-behavior: smooth on the document).  
- Buttons and links: Use hover effects (e.g., a slight glow or color change for the purple buttons, maybe a box-shadow neon glow effect on hover to enhance the theme).  
- Image gallery: if possible, implement a lightbox – when an image is clicked, open a modal with the full-size image and a dark backdrop, and allow closing it (simple JS to display a overlay div with the image). This gives a more immersive viewing of photos.  
- **Responsive Design:** Ensure all interactive features work on mobile/touch devices as well. For example, the hover effects should have a fallback (mobile users can still tap images to open them, etc.), and the navigation menu should collapse to a hamburger menu on small screens. Implement the hamburger toggle with a bit of JS (add/remove a class to show/hide the menu links), and ensure the language selector is accessible on mobile too (maybe as part of the menu or a fixed element). Use CSS media queries to adjust font sizes, layout (e.g., switch to single-column layouts on narrow screens), and tap-friendly button sizes.

## Technical Requirements and Code Structure

* **HTML:** Use semantic HTML5 elements for structure (e.g., <header>, <nav>, <section>, <footer>). Ensure the markup is properly nested and valid. Include comments in the HTML to delineate sections (for readability of code). The site can be contained in a single HTML file (one-page design with sections) or multiple HTML files (one for each page/section) if that fits better – but one-page is preferred for a seamless scroll experience.
* **CSS:** Create a separate CSS stylesheet (e.g., styles.css) for all styling. Organize it with section comments (/ *Hero Section* /, / *Nav* /, etc.). Use the chosen color scheme and typography consistently via CSS variables or classes. Include styles for the responsive behavior (media queries for different breakpoints, e.g., adjustments at 768px, 1024px widths, etc.). All visual aspects described (colors, layout, animations) should be implemented in CSS. Leverage Flexbox or CSS Grid for layout (for instance, grid for the gallery, flex for aligning items in hero). Use @keyframes for any custom animations (like text fade-in) and CSS transitions for smooth hover and toggle effects. Ensure to hide the preloader by default after load (you can add a CSS rule to hide it when a certain class is added via JS). Also style the language dropdown or buttons nicely to match the site (the language selector should be unobtrusive yet visible in the header).
* **JavaScript:** Use a main JavaScript file (e.g., script.js) to handle interactive behavior: language toggle functionality, preloader removal, scroll-based effects, mobile menu toggle, gallery lightbox, etc. Write vanilla JS (no need for jQuery or external libraries, unless absolutely needed for complex effects – but it should be doable with plain JS). Structure the script so that it runs after DOM content is loaded (DOMContentLoaded event or place script at end of body). Use functions to keep code organized (e.g., initLanguageToggle(), initGalleryLightbox(), etc.). Comment the JavaScript to explain key sections (especially the translation logic and any non-trivial calculations for animations).
* **Performance Considerations:** Optimize as needed – for example, if using large images/videos, ensure they are not excessively large (in a real scenario). Use proper <meta viewport> in the HTML head for mobile scaling. All links and interactive controls should have appropriate ARIA labels or roles for accessibility (e.g., aria-label on the language switch, alt text on images, form inputs with labels).
* **Output format:** Provide the complete code for the website. This includes the index.html (with all sections and content in English and Serbian text), styles.css, and script.js. If multiple files cannot be shown simultaneously in this prompt, you can present the HTML first (with links to CSS/JS), followed by the content of CSS and JS in separate sections. Ensure that when assembled, the code runs as a functional website. Double-check that all references (image sources, script and link hrefs) are consistent (e.g., if you assume a directory structure like assets/img/…, mention that). All text content should be filled in (no lorem ipsum; instead, use the fictional but coherent bakery content as described). For any dynamic strings in JS (like the translations object), make sure they exactly match the content in the HTML default language so that switching works correctly.

By following this specification, the AI (Codex) should generate an **impressive, fully-functional bakery website** that captures the flashy, engaging style of a Rockstar game promo site, while featuring a charming family-run bakery and its delicious offerings. The result should be an immersive bilingual website, with a rich purple-themed design, smooth interactivity, and all required sections completed in detail.