# **Recipe PWA Technical Documentation**

## **1. Summary of Design and Architecture**

### **Design Philosophy**

The Recipe PWA was built with three core principles:

* Intuitive user interface and experience
* Caching for offline functionalities
* SEO and Performance optimization.

### **UI/UX Design Decisions**

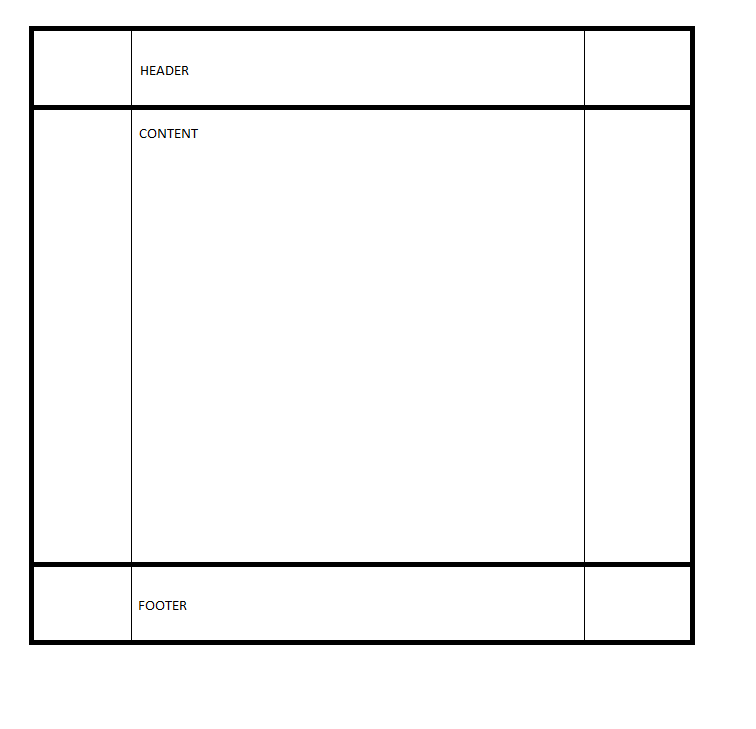
* **Color Scheme**:
  + Primary: Gold colour as highlight and accent color
  + Background: Gray color for content areas
  + Text: Light/Dark Gray color for optimal readability
* **Layout Structure**:
  + Responsive system using CSS Grid and Flexbox
  + Sidebar filter panel that collapses on mobile
  + Card-based recipe presentation for visual appeal

### **Website Logic Flow**

* A - user opens the website and the website checks for network
* B - if there is a network available, it caches the request from the api and if there’s no network, the app checks IndexedDB for caches.
* C - if data is found from a network or cache, they are displayed to the user. Otherwise, a network error is shown.
* D - users can filter available recipes, save recipes to favourites.
* E - all users requests are cached new or updated on the database depending on the actions.

### **Website Layout Architecture**

The website uses a simple layout architecture where each page consists of a header, a content(main) and a footer. All the sections are capped 1280px to avoid stretching the website on a big screen. The header and footer are at a fixed content height and the main fills the remaining area (note: the main can extend base on the content inside it).



https://stackoverflow.com/questions/50167396/trying-to-create-a-flex-based-layout-of-header-content-and-footer

## **2. Testing Documentation**

## **Issue 1: IndexedDB Cache on Recipe Fetched by ID**

* **Description**: Recipes fetched by ID were not cached to IndexedDB
* **Steps to Reproduce**:
  1. Click on any recipe fetched on Discover Page
  2. Go offline and refreshed
  3. Network error produced
* **Resolution**: Cast ID params from string type to number type

*export* *default* function RecipeId({ *params* }: { params: { id: string } }) {

const recipeQuery = useRecipeDetails(Number(*params*?.id))

const { data: recipe\_data } = recipeQuery || {}

## **Issue 2: Service Worker Image Cache**

* **Description**: Service worker failed to cache images from Spoonacular APIi
* **Tools Used**: Chrome Dev Tool
* **Steps to Reproduce**:

1. Fetch recipes or single recipe
2. Go offline and refresh
3. Alternative name showed on the images

* **Resolution**: Added cache config to PWA configuration

{

urlPattern: /*^*https:\/\/spoonacular\.com\/cdn\/.\**$*/,

handler: 'CacheFirst',

options: {

cacheName: 'spoonacular-images-cache',

expiration: {

maxEntries: 50,

maxAgeSeconds: 7 \* 24 \* 60 \* 60, *// 7 days*

},

},

},

## **Issue 3: Filtered Recipe Pagination**

* **Description**: Load more recipe not cached
* **Tools Used**: Chrome Dev Tool
* **Steps to Reproduce**:

1. Fetch recipes on Discover Page
2. Click Load More Button to load more recipes
3. Go offline and restart
4. Network error produced

* **Resolution**: Added pageParam(offset) to cache key

private createCacheKey(*filters*: RecipeFilters, *pageParam*?: number): string {

*return* `${*filters*.type || ''}-${*filters*.cuisine || ''}

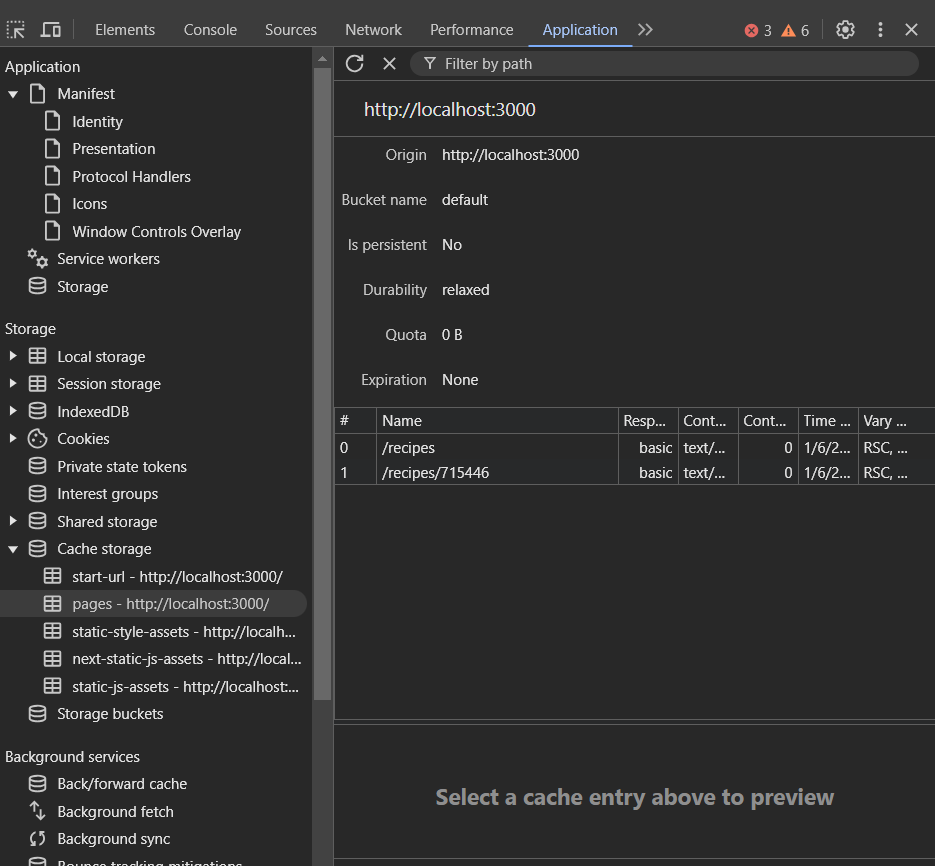
-${*filters*.diet || ''}-${*pageParam* || 0}`

}

### 

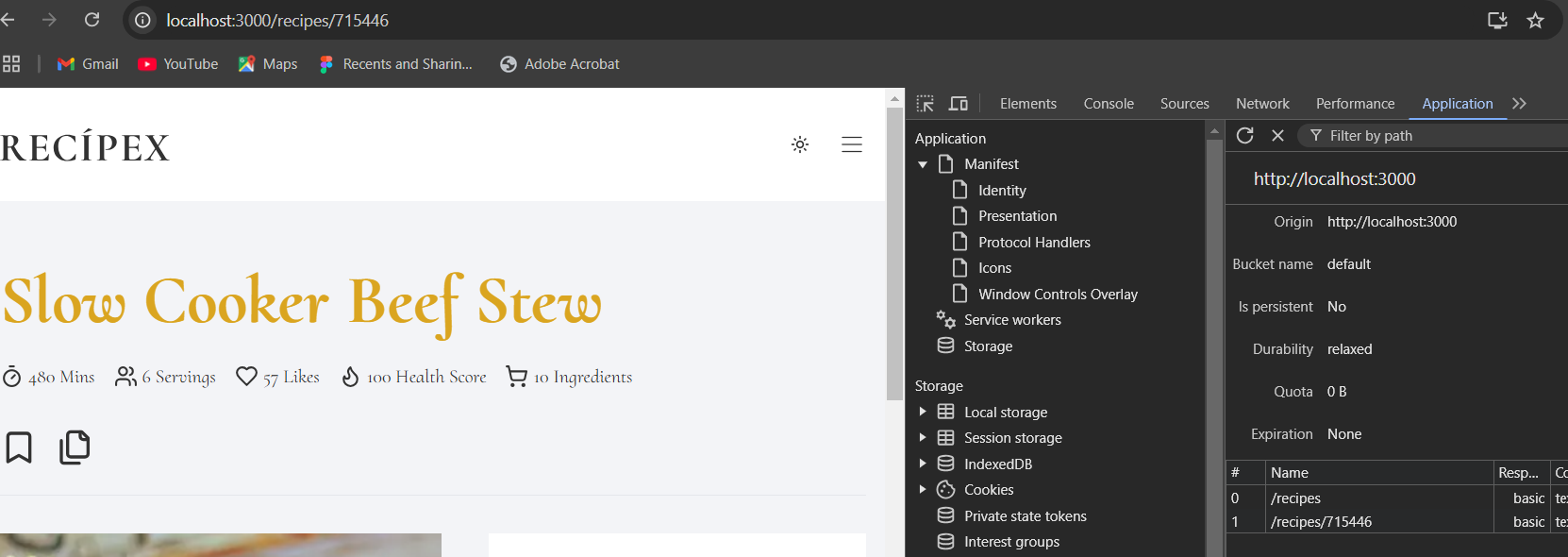
## **3. Screenshots and Feature Verification**

1. Offline Functionality

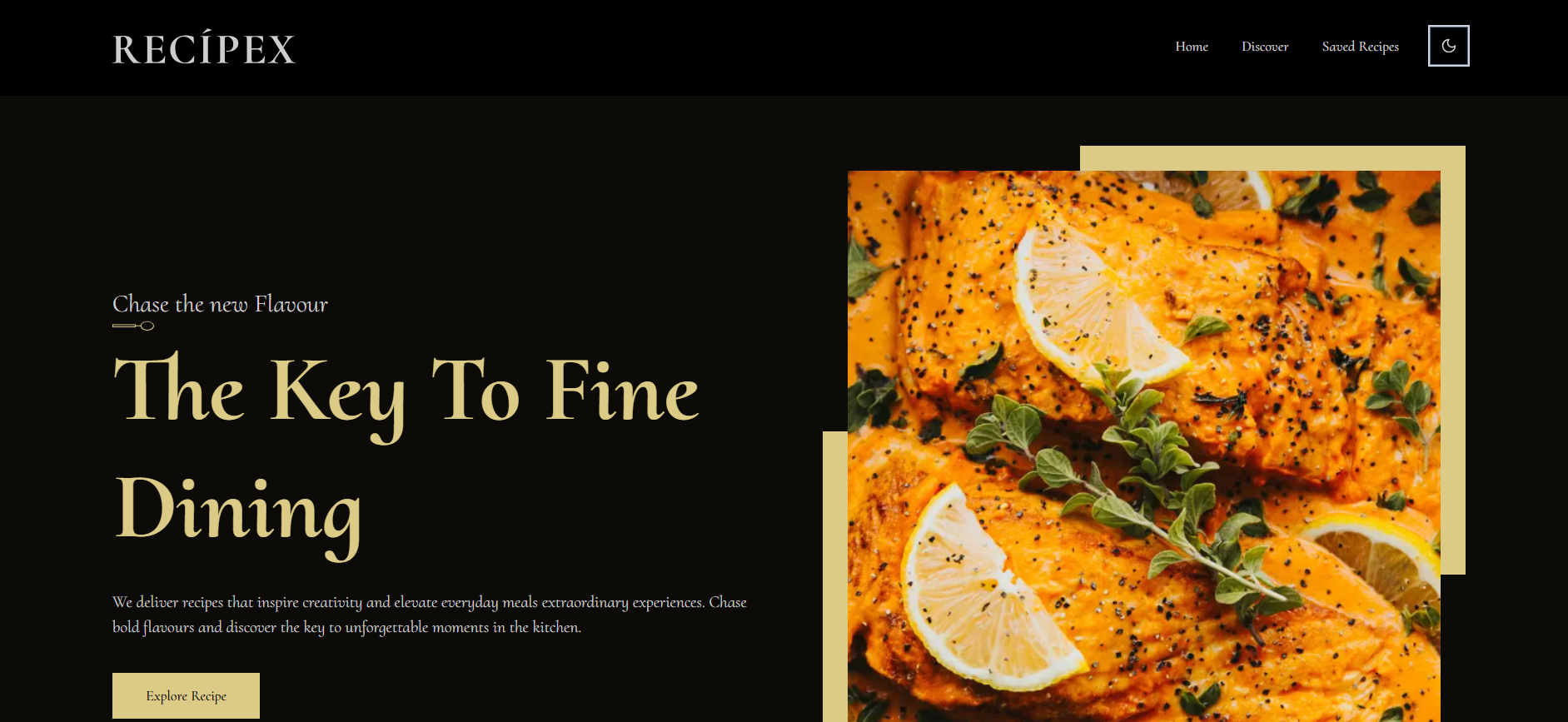


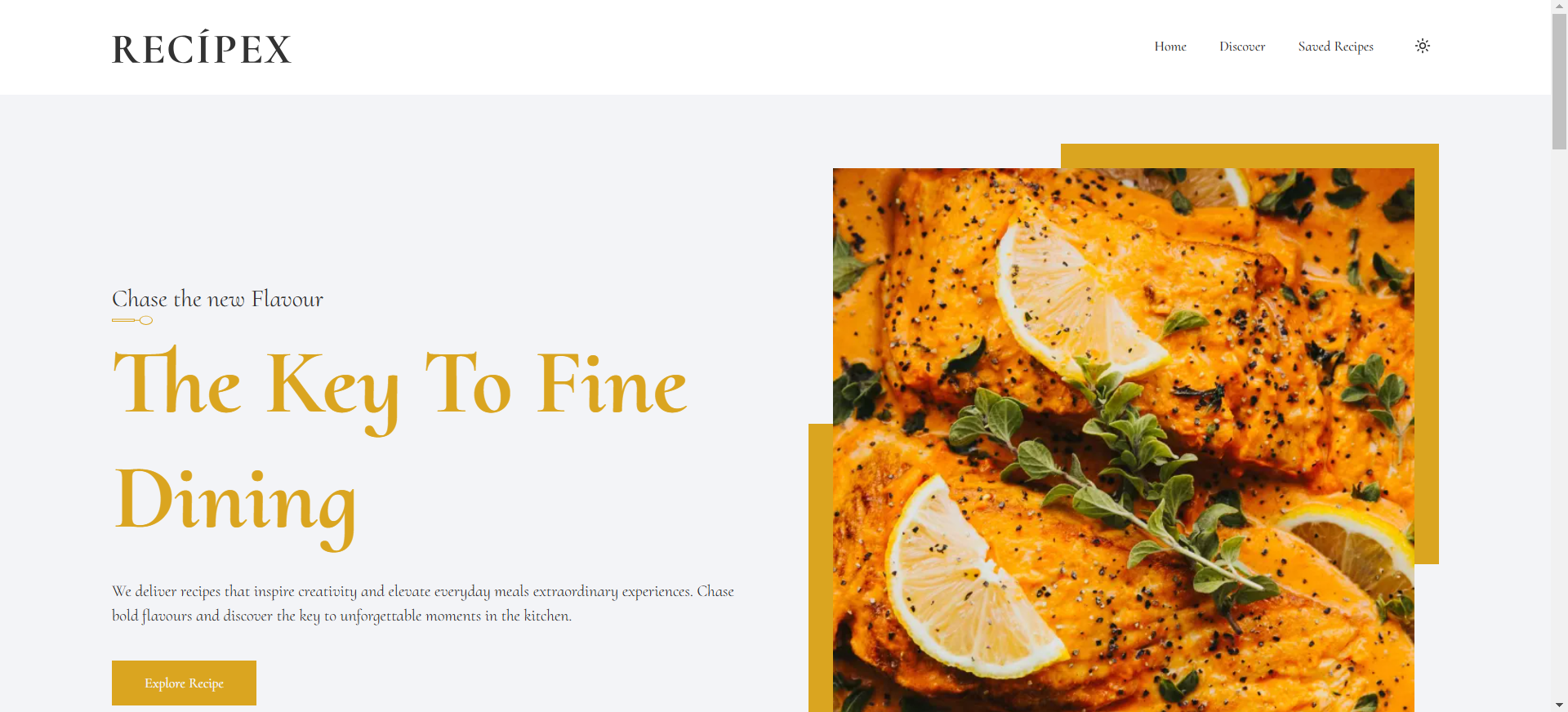
Demonstrating that the service work is fully registered and pages loaded are cached in the storage

1. PWA Installation Show Image

Shows installation prompt on compatible devices and Verifies proper manifest implementation

1. Light and Dark mode

Dark Mode

Light Mode

## **4. Development Reflection**

### **Testing Impact**

The testing process revealed several critical insights:

1. Importance of real device testing across platforms
2. Need for robust error handling in offline scenarios
3. Value of automated testing for cache consistency

### **Best Practices Identified**

* Implement progressive enhancement
* Use feature detection rather than browser detection
* Maintain clear separation of concerns between data and UI layers
* Implement comprehensive error boundaries
* Use type-safe interfaces for all data operations

### **Future Improvements**

* Implement background sync for offline changes
* Add push notifications for recipe updates
* Enhance offline search capabilities

The development of this PWA has reinforced the importance of thorough testing and documentation in creating reliable, user-friendly applications. The challenges encountered led to more robust solutions and better understanding of PWA architecture best practices.

## 