# BAB VI
# React *Progressive Web App* (PWA) II

## 6.1. Tujuan

1. Praktikan dapat memahami konsep dasar teknologi *Progressive Web App*
2. Praktikan mampu melakukan proses *deployment project* PWA dengan benar menggunakan layanan *hosting* Vercel
3. Praktikan dapat memahami dan mengimplementasikan Progressive Web App (PWA) menggunakan ReactJS
4. Praktikan dapat memahami konsep, menggunakan, dan mengintegrasikan RESTful API untuk *project* PWA
5. Praktikan dapat membangun Single Page Application (SPA) dengan React Hooks, error handling, state management, dan implementasi localStorage untuk data persistence.

## 6.2. Alat dan Bahan

1. Laptop
2. *Smartphone*
3. Browser
4. Text Editor (disarankan menggunakan Visual Studio Code)
5. Node.js
6. Git
7. Akun Github aktif untuk daftar Vercel

## 6.3. Dasar Teori

### 6.3.1. Progressive Web App (PWA)

Progressive Web App (PWA) adalah aplikasi web yang menggunakan pendekatan *progressive enhancement* untuk memberikan pengalaman seperti aplikasi *native*. PWA dapat diinstal pada perangkat pengguna, bekerja secara *offline* melalui *service worker*, dan menggunakan web *app manifest* untuk mengontrol perilaku instalasi serta tampilan. Tujuan utama PWA adalah menghadirkan aplikasi web yang andal, cepat, dan menarik, namun tetap berbasis teknologi web standar seperti HTML, CSS, dan JavaScript.

*(Sumber: https://developer.mozilla.org/en-US/docs/Web/Progressive_web_apps)*

### 6.3.2. React JS

React JS adalah pustaka JavaScript untuk membangun antarmuka pengguna (*user interface*). React menggunakan pendekatan *component-based* sehingga setiap bagian tampilan dipecah menjadi komponen yang dapat digunakan kembali. React juga menerapkan konsep *declarative programming*, di mana pengembang hanya perlu mendefinisikan bagaimana tampilan seharusnya pada suatu kondisi, dan React akan mengelola perubahan data secara otomatis melalui *Virtual* DOM. Tujuan utama React adalah memudahkan pengembang membuat antarmuka dinamis dan interaktif dengan struktur kode yang modular dan mudah dikelola.

*(Sumber: https://react.dev)*

### 6.3.3. Vite

Vite (diucapkan /vit/, dari bahasa Prancis yang berarti “cepat”) adalah sebuah *build tool* modern yang berfungsi untuk menjalankan dan membangun proyek aplikasi web *front-end*. Tujuan utama Vite adalah memberikan pengalaman pengembangan yang jauh lebih cepat dan efisien dibandingkan dengan alat-alat generasi sebelumnya. Vite memanfaatkan *native* ES modules pada browser modern untuk mempercepat waktu startup saat pengembangan, serta menggunakan *Rollup* untuk proses *build* produksi yang optimal.

*(Sumber: https://vite.dev/guide/)*

### 6.3.4. API

*Application Programming Interface* (API) adalah sekumpulan aturan dan protokol yang memungkinkan aplikasi atau sistem perangkat lunak berkomunikasi dan bertukar data. API mendefinisikan cara komponen perangkat lunak saling berinteraksi melalui permintaan (*request*) dan respons (*response*) yang terstruktur. API banyak digunakan untuk menghubungkan *frontend* dan *backend*, misalnya melalui layanan RESTful atau GraphQL, sehingga data dapat dikirim dan diterima secara efisien antar sistem.

*(Sumber: https://www.ibm.com/think/topics/api)*

### 6.3.5. Axios

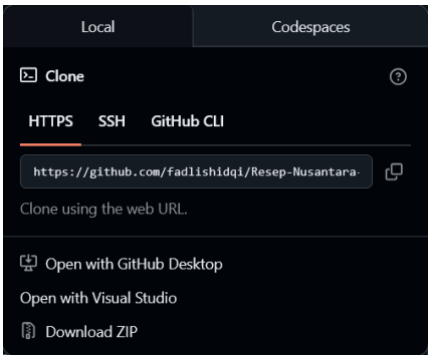
Axios adalah pustaka JavaScript berbasis *Promise* yang digunakan untuk melakukan permintaan HTTP dari browser maupun Node.js. Axios mendukung berbagai metode seperti GET, POST, PUT, dan DELETE, serta menyediakan fitur *interceptors*, *request cancellation*, dan transformasi data otomatis (misalnya format JSON). Keunggulan utama Axios adalah kemudahan integrasi dengan *frontend framework* seperti React, kemampuan penanganan kesalahan (*error handling*) yang baik, serta dukungan penuh untuk operasi asinkron menggunakan *async* dan *await*.

*(Sumber: https://axios-http.com/docs/intro)*

## 6.4. Langkah Kerja

### 6.4.1. *Clone Repository* Github Modul 4

1. Buka *link* berikut Link Repository Modul 4
2. Klik tombol *Code* kemudian *copy link* yang tampil




Gambar 6. 1 *Clone Repository* Github Modul 4

3. Buka *terminal* dan ikuti *command* di bawah ini untuk *clone* dan membuka *project* pada IDE

```
> git clone https://github.com/fadlishidqi/Resep-Nusantara-Modul4.git modul-lima
> cd modul-lima
> code .
```


Gambar 6. 2 Membuka *project* hasil *clone* pada *local*

4. Apabila *project* modul 4 masih kalian simpan, lewati langkah di atas

### 6.4.2. Integrasi *Frontend* PWA dengan API

1. Baca dokumentasi API yang ada pada *link* berikut Dokumentasi API
2. Buat *file* konfigurasi API pada src/config/api.js sebagai berikut

```
import axios from 'axios';

// Base URL configuration
const BASE_URL = import.meta.env.VITE_API_BASE_URL;

// Create axios instance
const apiClient = axios.create({
  baseURL: BASE_URL,
```

```
  headers: {
    'Content-Type': 'application/json',
  },
  timeout: 10000,
});

// Request interceptor
apiClient.interceptors.request.use(
  (config) => {
    return config;
  },
  (error) => {
    return Promise.reject(error);
  }
);

// Response interceptor
apiClient.interceptors.response.use(
  (response) => {
    return response.data;
  },
  (error) => {
    const errorMessage = error.response?.data?.message ||
error.message || 'An error occurred';
    return Promise.reject(error.response?.data || error);
  }
);

export { apiClient, BASE_URL };
```

3. Buat *environment file* (.env) pada *root project* sebagai berikut

```
VITE_API_BASE_URL=https://modlima.fuadfakhruz.id
```

4. Buat *custom hooks* (hooks/usexxx.js) untuk menggunakan *services*

useRecipes.js

```
import { useState, useEffect, useCallback } from 'react';
import recipeService from '../services/recipeService';

/**
 * Custom hook for fetching recipes
 * @param {Object} params - Query parameters
 *  @returns  {Object}  -  {  recipes,  loading,  error,
pagination, refetch }
 */
export function useRecipes(params = {}) {
  const [recipes, setRecipes] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const [pagination, setPagination] = useState(null);

  const fetchRecipes = useCallback(async () => {
```

```
    try {
      setLoading(true);
      setError(null);
      const response = await
recipeService.getRecipes(params);

      if (response.success) {
        setRecipes(response.data || []);
        setPagination(response.pagination || null);
      } else {
        setError(response.message || 'Failed to fetch
recipes');
      }
    } catch (err) {
      setError(err.message || 'An error occurred while
fetching recipes');
      setRecipes([]);
    } finally {
      setLoading(false);
    }
  }, [JSON.stringify(params)]);

  useEffect(() => {
    fetchRecipes();
  }, [fetchRecipes]);

  return {
    recipes,
    loading,
    error,
    pagination,
    refetch: fetchRecipes,
  };
}

/**
 * Custom hook for fetching a single recipe
 * @param {string} id - Recipe ID
 * @returns {Object} - { recipe, loading, error, refetch
}
 */
export function useRecipe(id) {
  const [recipe, setRecipe] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  const fetchRecipe = useCallback(async () => {
    if (!id) {
      setLoading(false);
      return;
    }

    try {
      setLoading(true);
      setError(null);
```

```
      const response = await
recipeService.getRecipeById(id);

      if (response.success) {
        setRecipe(response.data);
      } else {
        setError(response.message || 'Failed to fetch
recipe');
      }
    } catch (err) {
      setError(err.message || 'An error occurred while
fetching recipe');
      setRecipe(null);
    } finally {
      setLoading(false);
    }
  }, [id]);

  useEffect(() => {
    fetchRecipe();
  }, [fetchRecipe]);

  return {
    recipe,
    loading,
    error,
    refetch: fetchRecipe,
  };
}
```

useReviews.js

```
import { useState, useEffect, useCallback } from 'react';
import reviewService from '../services/reviewService';

/**
 * Custom hook for fetching reviews
 * @param {string} recipeId - Recipe ID
 * @returns {Object} - { reviews, loading, error, refetch
}
 */
export function useReviews(recipeId) {
  const [reviews, setReviews] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  const fetchReviews = useCallback(async () => {
    if (!recipeId) {
      setLoading(false);
      return;
    }

    try {
      setLoading(true);
      setError(null);
```

```
      const response = await
reviewService.getReviews(recipeId);

      if (response.success) {
        setReviews(response.data || []);
      } else {
        setError(response.message || 'Failed to fetch
reviews');
      }
    } catch (err) {
      setError(err.message || 'An error occurred while
fetching reviews');
      setReviews([]);
    } finally {
      setLoading(false);
    }
  }, [recipeId]);

  useEffect(() => {
    fetchReviews();
  }, [fetchReviews]);

  return {
    reviews,
    loading,
    error,
    refetch: fetchReviews,
  };
}

/**
 * Custom hook for creating a review
 * @returns {Object} - { createReview, loading, error,
success }
 */
export function useCreateReview() {
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(null);
  const [success, setSuccess] = useState(false);

  const createReview = async (recipeId, reviewData) => {
    try {
      setLoading(true);
      setError(null);
      setSuccess(false);

      const response = await
reviewService.createReview(recipeId, reviewData);

      if (response.success) {
        setSuccess(true);
        return response;
      } else {
        setError(response.message || 'Failed to create
review');
        return null;
```

```
      }
    } catch (err) {
      setError(err.message || 'An error occurred while
creating review');
      return null;
    } finally {
      setLoading(false);
    }
  };

  return {
    createReview,
    loading,
    error,
    success,
  };
}
```

useFavorites.js

```
import { useState, useEffect, useCallback } from 'react';
import                favoriteService                from
'../services/favoriteService';
import userService from '../services/userService';

/**
 * Get user identifier from localStorage or generate new
one
 */
const getUserIdentifier = () => {
  return userService.getUserIdentifier();
};

/**
 * Custom hook for fetching favorites
 * @returns {Object} - { favorites, loading, error,
refetch }
 */
export function useFavorites() {
  const [favorites, setFavorites] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const userIdentifier = getUserIdentifier();

  const fetchFavorites = useCallback(async () => {
    try {
      setLoading(true);
      setError(null);
      const          response          =          await
favoriteService.getFavorites(userIdentifier);

      if (response.success) {
        setFavorites(response.data || []);
      } else {
```

```
        setError(response.message || 'Failed to fetch
favorites');
      }
    } catch (err) {
      setError(err.message || 'An error occurred while
fetching favorites');
      setFavorites([]);
    } finally {
      setLoading(false);
    }
  }, [userIdentifier]);

  useEffect(() => {
    fetchFavorites();
  }, [fetchFavorites]);

  return {
    favorites,
    loading,
    error,
    refetch: fetchFavorites,
  };
}

/**
 * Custom hook for toggling favorites
 * @returns {Object} - { toggleFavorite, loading, error }
 */
export function useToggleFavorite() {
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState(null);
  const userIdentifier = getUserIdentifier();

  const toggleFavorite = async (recipeId) => {
    try {
      setLoading(true);
      setError(null);

      const          response          =          await
favoriteService.toggleFavorite({
        recipe_id: recipeId,
        user_identifier: userIdentifier,
      });

      if (response.success) {
        return response.data;
      } else {
        setError(response.message || 'Failed to toggle
favorite');
        return null;
      }
    } catch (err) {
      setError(err.message || 'An error occurred while
toggling favorite');
      return null;
    } finally {
```

```
      setLoading(false);
    }
  };

  return {
    toggleFavorite,
    loading,
    error,
  };
}

/**
 * Custom hook to check if a recipe is favorited
 * @param {string} recipeId - Recipe ID
 *   @returns   {Object}   -   {   isFavorited,   loading,
toggleFavorite }
 */
export function useIsFavorited(recipeId) {
  const { favorites, loading: fetchLoading, refetch } =
useFavorites();
  const { toggleFavorite: toggle, loading: toggleLoading
} = useToggleFavorite();

  const isFavorited = favorites.some(fav => fav.id ===
recipeId);

  const toggleFavorite = async () => {
    const result = await toggle(recipeId);
    if (result) {
      await refetch();
    }
    return result;
  };

  return {
    isFavorited,
    loading: fetchLoading || toggleLoading,
    toggleFavorite,
  };
}

export { getUserIdentifier };
```

5. Implementasi *services* (src/services/xxxService.js) untuk membuat *method fetch* API sesuai dengan *endpoint*

recipeService.js

```
import { apiClient } from '../config/api';

class RecipeService {
  /**
   * Get all recipes with optional filters
   * @param {Object} params - Query parameters
```

```
   * @param {number} params.page - Page number (default:
1)
   * @param {number} params.limit - Items per page
(default: 10)
   * @param {string} params.category - Filter by
category: 'makanan' | 'minuman'
   * @param {string} params.difficulty - Filter by
difficulty: 'mudah' | 'sedang' | 'sulit'
   * @param {string} params.search - Search in
name/description
   * @param {string} params.sort_by - Sort by field
(default: 'created_at')
   * @param {string} params.order - Sort order: 'asc' |
'desc' (default: 'desc')
   * @returns {Promise}
   */
  async getRecipes(params = {}) {
    try {
      const response = await
apiClient.get('/api/v1/recipes', { params });
      return response;
    } catch (error) {
      throw error;
    }
  }

  /**
   * Get recipe by ID
   * @param {string} id - Recipe ID
   * @returns {Promise}
   */
  async getRecipeById(id) {
    try {
      const response = await
apiClient.get(`/api/v1/recipes/${id}`);
      return response;
    } catch (error) {
      throw error;
    }
  }

  /**
   * Create new recipe
   * @param {Object} recipeData - Recipe data
   * @returns {Promise}
   */
  async createRecipe(recipeData) {
    try {
      const response = await
apiClient.post('/api/v1/recipes', recipeData);
      return response;
    } catch (error) {
      throw error;
    }
  }
```

```
  /**
   * Update existing recipe (full replacement)
   * @param {string} id - Recipe ID
   * @param {Object} recipeData - Complete recipe data
(all fields required)
   * @returns {Promise}
   */
  async updateRecipe(id, recipeData) {
    try {
      const response = await
apiClient.put(`/api/v1/recipes/${id}`, recipeData);
      return response;
    } catch (error) {
      throw error;
    }
  }

  /**
   * Partially update recipe (only send fields to update)
   * @param {string} id - Recipe ID
   * @param {Object} partialData - Partial recipe data
(only fields to update)
   * @returns {Promise}
   */
  async patchRecipe(id, partialData) {
    try {
      const response = await
apiClient.patch(`/api/v1/recipes/${id}`, partialData);
      return response;
    } catch (error) {
      throw error;
    }
  }

  /**
   * Delete recipe
   * @param {string} id - Recipe ID
   * @returns {Promise}
   */
  async deleteRecipe(id) {
    try {
      const response = await
apiClient.delete(`/api/v1/recipes/${id}`);
      return response;
    } catch (error) {
      throw error;
    }
  }
}

export default new RecipeService();
```

reviewService.js

```
import { apiClient } from '../config/api';

class ReviewService {
  /**
   * Get all reviews for a recipe
   * @param {string} recipeId - Recipe ID
   * @returns {Promise}
   */
  async getReviews(recipeId) {
    try {
      const response = await
apiClient.get(`/api/v1/recipes/${recipeId}/reviews`);
      return response;
    } catch (error) {
      throw error;
    }
  }

  /**
   * Create review for a recipe
   * @param {string} recipeId - Recipe ID
   * @param {Object} reviewData - Review data
   * @param {string} reviewData.user_identifier - User
identifier
   * @param {number} reviewData.rating - Rating (1-5)
   * @param {string} reviewData.comment - Review comment
(optional)
   * @returns {Promise}
   */
  async createReview(recipeId, reviewData) {
    try {
      const response = await
apiClient.post(`/api/v1/recipes/${recipeId}/reviews`,
reviewData);
      return response;
    } catch (error) {
      throw error;
    }
  }

  /**
   * Update existing review
   * @param {string} reviewId - Review ID
   * @param {Object} reviewData - Updated review data
   * @param {number} reviewData.rating - Rating (1-5)
   * @param {string} reviewData.comment - Review comment
(optional)
   * @returns {Promise}
   */
  async updateReview(reviewId, reviewData) {
    try {
      const response = await
apiClient.put(`/api/v1/reviews/${reviewId}`, reviewData);
      return response;
```

```
    } catch (error) {
      throw error;
    }
  }

  /**
   * Delete review
   * @param {string} reviewId - Review ID
   * @returns {Promise}
   */
  async deleteReview(reviewId) {
    try {
      const response = await
apiClient.delete(`/api/v1/reviews/${reviewId}`);
      return response;
    } catch (error) {
      throw error;
    }
  }
}

export default new ReviewService();
```

uploadService.js

```
import axios from 'axios';
import { BASE_URL } from '../config/api';

class UploadService {
  /**
   * Upload recipe image to MinIO
   * @param {File} file - Image file to upload
   * @returns {Promise}
   */
  async uploadImage(file) {
    try {
      // Validate file
      if (!file) {
        throw new Error('No file provided');
      }

      // Validate file type
      const allowedTypes = ['image/jpeg', 'image/jpg',
'image/png', 'image/webp'];
      if (!allowedTypes.includes(file.type)) {
        throw new Error('Invalid file type. Allowed:
.jpg, .jpeg, .png, .webp');
      }

      // Validate file size (5MB)
      const maxSize = 5 * 1024 * 1024; // 5MB in bytes
      if (file.size > maxSize) {
        throw new Error('File size exceeds 5MB limit');
      }
```

```
      // Create form data
      const formData = new FormData();
      formData.append('image', file);

      // Upload to server
      const response = await
axios.post(`${BASE_URL}/api/v1/upload`, formData, {
        headers: {
          'Content-Type': 'multipart/form-data',
        },
        timeout: 30000, // 30 seconds for upload
      });

      return response.data;
    } catch (error) {
      throw error;
    }
  }
}

export default new UploadService();
```

favoriteService.js

```
import { apiClient } from '../config/api';
class FavoriteService {
  /**
   * Get all favorite recipes by user identifier
   * @param {string} userIdentifier - User identifier
   * @returns {Promise}
   */
  async getFavorites(userIdentifier) {
    try {
      const response = await
apiClient.get('/api/v1/favorites', {
        params: { user_identifier: userIdentifier }
      });
      return response;
    } catch (error) {
      throw error;
    }
  }

  /**
   * Toggle favorite (add if not exists, remove if
exists)
   * @param {Object} data - Favorite data
   * @param {string} data.recipe_id - Recipe ID
   * @param {string} data.user_identifier - User
identifier
   * @returns {Promise}
   */
  async toggleFavorite(data) {
    try {
```

```
      const response = await
apiClient.post('/api/v1/favorites/toggle', data);
      return response;
    } catch (error) {
      throw error;
    }
  }
}

export default new FavoriteService();
```

userService.js

```
const USER_PROFILE_KEY = 'user_profile';
const USER_IDENTIFIER_KEY = 'user_identifier';
/**
 * Get or generate user identifier
 */
export const getUserIdentifier = () => {
  let userId = localStorage.getItem(USER_IDENTIFIER_KEY);
  if (!userId) {
    userId =
`user_${Date.now()}_${Math.random().toString(36).substr(2
, 9)}`;
    localStorage.setItem(USER_IDENTIFIER_KEY, userId);
  }
  return userId;
};

/**
 * Get user profile from localStorage
 */
export const getUserProfile = () => {
  try {
    const profile =
localStorage.getItem(USER_PROFILE_KEY);
    if (profile) {
      return JSON.parse(profile);
    }

    // Return default profile with user identifier
    return {
      username: 'Pengguna',
      avatar: null,
      bio: '',
      userId: getUserIdentifier()
    };
  } catch (error) {
    return {
      username: 'Pengguna',
      avatar: null,
      bio: '',
      userId: getUserIdentifier()
    };
  }
```

```
};

/**
 * Save user profile to localStorage
 */
export const saveUserProfile = (profile) => {
  try {
    const userId = getUserIdentifier();
    const profileData = {
      ...profile,
      userId,
      updatedAt: new Date().toISOString()
    };
    localStorage.setItem(USER_PROFILE_KEY,
JSON.stringify(profileData));
    return { success: true, data: profileData };
  } catch (error) {
    return { success: false, message: error.message };
  }
};

/**
 * Update user avatar
 */
export const updateAvatar = (avatarBase64) => {
  try {
    const profile = getUserProfile();
    profile.avatar = avatarBase64;
    return saveUserProfile(profile);
  } catch (error) {
    return { success: false, message: error.message };
  }
};

/**
 * Update username
 */
export const updateUsername = (username) => {
  try {
    const profile = getUserProfile();
    profile.username = username.trim() || 'Pengguna';
    return saveUserProfile(profile);
  } catch (error) {
    return { success: false, message: error.message };
  }
};

/**
 * Update bio
 */
export const updateBio = (bio) => {
  try {
    const profile = getUserProfile();
    profile.bio = bio.trim();
    return saveUserProfile(profile);
  } catch (error) {
```

```
    return { success: false, message: error.message };
  }
};

export default {
  getUserIdentifier,
  getUserProfile,
  saveUserProfile,
  updateAvatar,
  updateUsername,
  updateBio
};
```

6. Lanjut ke *folder* src/pages kemudian edit tiap *file* yang ada didalamnya dengan kode di bawah ini

CreateRecipePage.jsx

```
// src/pages/CreateRecipePage.jsx
import { useState, useEffect } from 'react';
import { ArrowLeft, Upload, X, Plus, Image as ImageIcon,
Loader, Save, AlertCircle, CheckCircle } from 'lucide-
react';
import recipeService from '../services/recipeService';
import uploadService from '../services/uploadService';
import { saveDraft, loadDraft, deleteDraft, hasDraft,
getDraftTimestamp, formatDraftTime } from
'../utils/draftStorage';
import ConfirmModal from
'../components/modals/ConfirmModal';

export default function CreateRecipePage({ onBack,
onSuccess }) {
  // Step state: 'upload' or 'form'
  const [currentStep, setCurrentStep] =
useState('upload');

  const [formData, setFormData] = useState({
    name: '',
    category: 'makanan',
    description: '',
    prep_time: '',
    cook_time: '',
    servings: '',
    difficulty: 'mudah',
    is_featured: false,
  });

  const [imageFile, setImageFile] = useState(null);
  const [imagePreview, setImagePreview] = useState(null);
  const [uploadedImageUrl, setUploadedImageUrl] =
useState('');
  const [ingredients, setIngredients] = useState([{ name:
'', quantity: '' }]);
  const [steps, setSteps] = useState(['']);
```

```
  const [uploading, setUploading] = useState(false);
  const [creating, setCreating] = useState(false);
  const [error, setError] = useState('');
  const [showDraftModal, setShowDraftModal] =
useState(false);
  const [draftTimestamp, setDraftTimestamp] =
useState(null);
  const [autoSaveEnabled, setAutoSaveEnabled] =
useState(true);

  // Check for existing draft on mount
  useEffect(() => {
    if (hasDraft('create')) {
      const timestamp = getDraftTimestamp('create');
      setDraftTimestamp(timestamp);
      setShowDraftModal(true);
    }
  }, []);

  // Auto-save draft every 30 seconds (only when in form
step)
  useEffect(() => {
    if (!autoSaveEnabled || currentStep !== 'form')
return;

    const interval = setInterval(() => {
      handleSaveDraft();
    }, 30000); // 30 seconds

    return () => clearInterval(interval);
  }, [formData, ingredients, steps, autoSaveEnabled,
currentStep]);

  // Load draft
  const handleLoadDraft = () => {
    const draft = loadDraft('create');
    if (draft) {
      setFormData(draft.formData || formData);
      setIngredients(draft.ingredients || ingredients);
      setSteps(draft.steps || steps);
      if (draft.uploadedImageUrl) {
        setUploadedImageUrl(draft.uploadedImageUrl);
        setImagePreview(draft.uploadedImageUrl);
        setCurrentStep('form'); // Go to form if image
was uploaded
      }
      setShowDraftModal(false);
      alert('Draft berhasil dimuat!');
    }
  };

  // Discard draft
  const handleDiscardDraft = () => {
    deleteDraft('create');
    setShowDraftModal(false);
```

```
    setDraftTimestamp(null);
  };

  // Save draft manually
  const handleSaveDraft = () => {
    const draftData = {
      formData,
      ingredients,
      steps,
      uploadedImageUrl,
    };

    const success = saveDraft(draftData, 'create');
    if (success) {
      const timestamp = getDraftTimestamp('create');
      setDraftTimestamp(timestamp);
    }
  };

  // Handle image selection
  const handleImageChange = (e) => {
    const file = e.target.files[0];
    if (!file) return;

    // Validate file size (max 5MB)
    if (file.size > 5 * 1024 * 1024) {
      setError('Ukuran file maksimal 5MB');
      return;
    }

    // Validate file type
    const allowedTypes = ['image/jpeg', 'image/jpg',
'image/png', 'image/webp'];
    if (!allowedTypes.includes(file.type)) {
      setError('Format file harus .jpg, .jpeg, .png, atau
.webp');
      return;
    }

    setImageFile(file);
    setImagePreview(URL.createObjectURL(file));
    setError('');
  };

  // Upload image to server
  const handleUploadImage = async () => {
    if (!imageFile) {
      setError('Pilih gambar terlebih dahulu');
      return;
    }

    try {
      setUploading(true);
      setError('');
```

```
      const uploadResult = await
uploadService.uploadImage(imageFile);

      if (uploadResult.success) {
        setUploadedImageUrl(uploadResult.data.url);
        setCurrentStep('form');
        alert('✅ Gambar berhasil diupload! Silakan isi
form resep.');
      } else {
        throw new Error(uploadResult.error || 'Gagal
upload gambar');
      }
    } catch (err) {
      setError(err.message || 'Gagal mengupload gambar');
    } finally {
      setUploading(false);
    }
  };

  // Change uploaded image
  const handleChangeImage = () => {
    setCurrentStep('upload');
    setImageFile(null);
    setImagePreview(null);
    setUploadedImageUrl('');
    if (imagePreview) {
      URL.revokeObjectURL(imagePreview);
    }
  };

  // Remove image
  const handleRemoveImage = () => {
    setImageFile(null);
    setImagePreview(null);
    if (imagePreview) {
      URL.revokeObjectURL(imagePreview);
    }
  };

  // Handle form field changes
  const handleChange = (e) => {
    const { name, value, type, checked } = e.target;
    setFormData(prev => ({
      ...prev,
      [name]: type === 'checkbox' ? checked : value
    }));
  };

  // Handle ingredient changes
  const handleIngredientChange = (index, field, value) =>
{
    const newIngredients = [...ingredients];
    newIngredients[index][field] = value;
    setIngredients(newIngredients);
  };
```

```
  const addIngredient = () => {
    setIngredients([...ingredients, { name: '', quantity:
'' }]);
  };

  const removeIngredient = (index) => {
    if (ingredients.length > 1) {
      setIngredients(ingredients.filter((_, i) => i !==
index));
    }
  };

  // Handle step changes
  const handleStepChange = (index, value) => {
    const newSteps = [...steps];
    newSteps[index] = value;
    setSteps(newSteps);
  };

  const addStep = () => {
    setSteps([...steps, '']);
  };

  const removeStep = (index) => {
    if (steps.length > 1) {
      setSteps(steps.filter((_, i) => i !== index));
    }
  };

  // Validate form
  const validateForm = () => {
    if (!formData.name.trim()) {
      setError('Nama resep wajib diisi');
      return false;
    }

    if (formData.name.trim().length < 3) {
      setError('Nama resep minimal 3 karakter');
      return false;
    }

    if (!formData.prep_time || formData.prep_time <= 0) {
      setError('Waktu persiapan harus lebih dari 0');
      return false;
    }

    if (!formData.cook_time || formData.cook_time <= 0) {
      setError('Waktu memasak harus lebih dari 0');
      return false;
    }

    if (!formData.servings || formData.servings <= 0) {
      setError('Jumlah porsi harus lebih dari 0');
      return false;
    }
```

```
    // Validate ingredients
    const validIngredients = ingredients.filter(ing =>
ing.name.trim() && ing.quantity.trim());
    if (validIngredients.length === 0) {
      setError('Minimal harus ada 1 bahan');
      return false;
    }

    // Validate steps
    const validSteps = steps.filter(step => step.trim());
    if (validSteps.length === 0) {
      setError('Minimal harus ada 1 langkah');
      return false;
    }

    return true;
  };

  // Submit form
  const handleSubmit = async (e) => {
    e.preventDefault();
    setError('');

    if (!uploadedImageUrl) {
      setError('Gambar harus diupload terlebih dahulu');
      return;
    }

    if (!validateForm()) {
      return;
    }

    try {
      setCreating(true);

      // Prepare recipe data with uploaded image URL
      const validIngredients = ingredients.filter(ing =>
ing.name.trim() && ing.quantity.trim());
      const validSteps = steps.filter(step =>
step.trim());

      const recipeData = {
        name: formData.name.trim(),
        category: formData.category,
        description: formData.description.trim(),
        image_url: uploadedImageUrl, // Use uploaded
image URL
        prep_time: parseInt(formData.prep_time),
        cook_time: parseInt(formData.cook_time),
        servings: parseInt(formData.servings),
        difficulty: formData.difficulty,
        is_featured: formData.is_featured,
        ingredients: validIngredients,
        steps: validSteps,
      };
```

```
      // Create recipe
      const result = await
recipeService.createRecipe(recipeData);

      if (result.success) {
        alert('Resep berhasil dibuat!');
        deleteDraft('create'); // Clear draft after
successful creation
        if (onSuccess) {
          onSuccess(result.data);
        } else if (onBack) {
          onBack();
        }
      } else {
        throw new Error(result.message || 'Gagal membuat
resep');
      }
    } catch (err) {
      setError(err.message || 'Terjadi kesalahan saat
membuat resep');
    } finally {
      setCreating(false);
      setUploading(false);
    }
  };

  return (
    <div className="min-h-screen bg-linear-to-br from-
blue-50 via-white to-indigo-50 pb-20 md:pb-8">
      {/* Draft Modal */}
      <ConfirmModal
        isOpen={showDraftModal}
        onClose={handleDiscardDraft}
        onConfirm={handleLoadDraft}
        title="Draft Ditemukan"
        message={`Anda memiliki draft yang disimpan
${draftTimestamp ? formatDraftTime(draftTimestamp) : ''}.
Apakah Anda ingin melanjutkan draft tersebut?`}
        confirmText="Muat Draft"
        cancelText="Mulai Baru"
        variant="info"
      />

      {/* Header */}
      <div className="sticky top-0 z-10 bg-white/80
backdrop-blur-md border-b border-slate-200 shadow-sm">
        <div className="max-w-4xl mx-auto px-4 py-4 flex
items-center justify-between">
          <button
            onClick={onBack}
            className="flex items-center gap-2 text-
slate-700 hover:text-slate-900 transition-colors"
          >
            <ArrowLeft className="w-5 h-5" />
            <span className="font-medium">Kembali</span>
          </button>
```

```
          {/* Draft Info & Save Button */}
          <div className="flex items-center gap-3">
            {draftTimestamp && (
              <div className="hidden md:flex items-center
gap-2 text-sm text-slate-500">
                <Save className="w-4 h-4" />
                <span>Tersimpan
{formatDraftTime(draftTimestamp)}</span>
              </div>
            )}
            <button
              onClick={handleSaveDraft}
              className="flex items-center gap-2 px-3 py-
2 text-slate-700 hover:bg-slate-100 rounded-lg
transition-colors"
              title="Simpan draft"
            >
              <Save className="w-5 h-5" />
              <span className="hidden md:inline text-
sm">Simpan Draft</span>
            </button>
          </div>
        </div>
      </div>

      <main className="max-w-4xl mx-auto px-4 py-8">
        <div className="bg-white/60 backdrop-blur-sm
rounded-3xl p-6 md:p-8 shadow-xl border border-white/40">
          <h1 className="text-3xl font-bold text-slate-
800 mb-2">Buat Resep Baru</h1>
          <div className="flex items-center justify-
between mb-6">
            <p className="text-slate-600">Bagikan resep
favoritmu dengan komunitas</p>
            {autoSaveEnabled && currentStep === 'form' &&
(
              <div className="flex items-center gap-2
text-sm text-green-600">
                <AlertCircle className="w-4 h-4" />
                <span>Auto-save aktif</span>
              </div>
            )}
          </div>

          {/* Step Indicator */}
          <div className="mb-8 flex items-center justify-
center gap-4">
            <div className={`flex items-center gap-2
${currentStep === 'upload' ? 'text-blue-600' : 'text-
green-600'}`}>
              <div className={`w-8 h-8 rounded-full flex
items-center justify-center font-bold ${
                currentStep === 'upload' ? 'bg-blue-600
text-white' : 'bg-green-600 text-white'
              }`}>
```

```
                {currentStep === 'form' ? <CheckCircle
className="w-5 h-5" /> : '1'}
              </div>
              <span className="font-medium">Upload
Gambar</span>
            </div>
            <div className={`h-1 w-16 ${currentStep ===
'form' ? 'bg-green-600' : 'bg-slate-300'}`}></div>
            <div className={`flex items-center gap-2
${currentStep === 'form' ? 'text-blue-600' : 'text-slate-
400'}`}>
              <div className={`w-8 h-8 rounded-full flex
items-center justify-center font-bold ${
                currentStep === 'form' ? 'bg-blue-600
text-white' : 'bg-slate-300 text-slate-600'
              }`}>
                2
              </div>
              <span className="font-medium">Isi
Form</span>
            </div>
          </div>

          {/* Error Message */}
          {error && (
            <div className="mb-6 bg-red-50 border border-
red-200 rounded-xl p-4">
              <p className="text-red-600 text-
sm">{error}</p>
            </div>
          )}

          {/* STEP 1: Image Upload */}
          {currentStep === 'upload' && (
            <div className="space-y-6">
              <div className="text-center mb-6">
                <h2 className="text-xl font-bold text-
slate-800 mb-2">Upload Foto Resep</h2>
                <p className="text-slate-600">Gambar
harus diupload terlebih dahulu sebelum mengisi form</p>
              </div>

              {!imagePreview ? (
                <div className="border-2 border-dashed
border-slate-300 rounded-xl p-12 text-center
hover:border-blue-400 transition-colors">
                  <input
                    type="file"

accept="image/jpeg,image/jpg,image/png,image/webp"
                    onChange={handleImageChange}
                    className="hidden"
                    id="image-upload"
                  />
                  <label
                    htmlFor="image-upload"
```

```
                    className="cursor-pointer flex flex-
col items-center gap-4"
                  >
                    <div className="w-20 h-20 bg-blue-100
rounded-full flex items-center justify-center">
                      <ImageIcon className="w-10 h-10
text-blue-600" />
                    </div>
                    <div>
                      <p className="text-lg text-slate-
700 font-medium">Klik untuk pilih foto</p>
                      <p className="text-sm text-slate-
500 mt-2">
                        Maksimal 5MB (.jpg, .jpeg, .png,
.webp)
                      </p>
                    </div>
                  </label>
                </div>
              ) : (
                <div className="space-y-4">
                  <div className="relative">
                    <img
                      src={imagePreview}
                      alt="Preview"
                      className="w-full h-80 object-cover
rounded-xl shadow-lg"
                    />
                    <button
                      type="button"
                      onClick={handleRemoveImage}
                      className="absolute top-4 right-4
p-2 bg-red-500 text-white rounded-full hover:bg-red-600
transition-colors shadow-lg"
                    >
                      <X className="w-5 h-5" />
                    </button>
                  </div>

                  <button
                    type="button"
                    onClick={handleUploadImage}
                    disabled={uploading}
                    className="w-full py-4 bg-gradient-
to-r from-blue-600 to-indigo-600 text-white font-semibold
rounded-xl hover:from-blue-700 hover:to-indigo-700
disabled:opacity-50 disabled:cursor-not-allowed
transition-all shadow-lg hover:shadow-xl flex items-
center justify-center gap-2"
                  >
                    {uploading ? (
                      <>
                        <Loader className="w-5 h-5
animate-spin" />
                        <span>Mengupload...</span>
                      </>
```

```
                    ) : (
                      <>
                        <Upload className="w-5 h-5" />
                        <span>Upload Gambar &
Lanjutkan</span>
                      </>
                    )}
                  </button>
                </div>
              )}
            </div>
          )}

          {/* STEP 2: Form Fields */}
          {currentStep === 'form' && (
            <form onSubmit={handleSubmit}
className="space-y-8">
              {/* Uploaded Image Preview */}
              <div className="bg-green-50 border border-
green-200 rounded-xl p-4">
                <div className="flex items-center gap-4">
                  <img
                    src={uploadedImageUrl}
                    alt="Uploaded"
                    className="w-24 h-24 object-cover
rounded-lg"
                  />
                  <div className="flex-1">
                    <div className="flex items-center
gap-2 text-green-700 font-medium mb-1">
                      <CheckCircle className="w-5 h-5" />
                      <span>Gambar berhasil
diupload</span>
                    </div>
                    <p className="text-sm text-slate-
600">Silakan isi form di bawah untuk melengkapi resep</p>
                  </div>
                  <button
                    type="button"
                    onClick={handleChangeImage}
                    className="px-4 py-2 text-blue-600
hover:bg-blue-50 rounded-lg transition-colors text-sm
font-medium"
                  >
                    Ganti Gambar
                  </button>
                </div>
              </div>

              {/* Basic Info */}
              <div className="grid md:grid-cols-2 gap-6">
                <div>
                  <label className="block text-sm font-
medium text-slate-700 mb-2">
                    Nama Resep <span className="text-red-
500">*</span>
```

```
                </label>
                <input
                  type="text"
                  name="name"
                  value={formData.name}
                  onChange={handleChange}
                  placeholder="Nasi Goreng Spesial"
                  className="w-full px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500"
                  required
                />
              </div>

              <div>
                <label className="block text-sm font-
medium text-slate-700 mb-2">
                  Kategori <span className="text-red-
500">*</span>
                </label>
                <select
                  name="category"
                  value={formData.category}
                  onChange={handleChange}
                  className="w-full px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500"
                >
                  <option
value="makanan">Makanan</option>
                  <option
value="minuman">Minuman</option>
                </select>
              </div>
            </div>

            {/* Description */}
            <div>
              <label className="block text-sm font-medium
text-slate-700 mb-2">
                Deskripsi
              </label>
              <textarea
                name="description"
                value={formData.description}
                onChange={handleChange}
                placeholder="Ceritakan tentang resep
ini..."
                rows={4}
                className="w-full px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500 resize-none"
              />
            </div>

            {/* Time & Servings */}
```

```
            <div className="grid md:grid-cols-3 gap-6">
              <div>
                <label className="block text-sm font-
medium text-slate-700 mb-2">
                  Waktu Persiapan (menit) <span
className="text-red-500">*</span>
                </label>
                <input
                  type="number"
                  name="prep_time"
                  value={formData.prep_time}
                  onChange={handleChange}
                  placeholder="15"
                  min="1"
                  className="w-full px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500"
                  required
                />
              </div>

              <div>
                <label className="block text-sm font-
medium text-slate-700 mb-2">
                  Waktu Memasak (menit) <span
className="text-red-500">*</span>
                </label>
                <input
                  type="number"
                  name="cook_time"
                  value={formData.cook_time}
                  onChange={handleChange}
                  placeholder="20"
                  min="1"
                  className="w-full px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500"
                  required
                />
              </div>

              <div>
                <label className="block text-sm font-
medium text-slate-700 mb-2">
                  Porsi (orang) <span className="text-
red-500">*</span>
                </label>
                <input
                  type="number"
                  name="servings"
                  value={formData.servings}
                  onChange={handleChange}
                  placeholder="4"
                  min="1"
```

```
                  className="w-full px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500"
                  required
                />
              </div>
            </div>

            {/* Difficulty */}
            <div>
              <label className="block text-sm font-medium
text-slate-700 mb-2">
                Tingkat Kesulitan <span className="text-
red-500">*</span>
              </label>
              <select
                name="difficulty"
                value={formData.difficulty}
                onChange={handleChange}
                className="w-full px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500"
              >
                <option value="mudah">Mudah</option>
                <option value="sedang">Sedang</option>
                <option value="sulit">Sulit</option>
              </select>
            </div>

            {/* Ingredients */}
            <div>
              <label className="block text-sm font-medium
text-slate-700 mb-3">
                Bahan-bahan <span className="text-red-
500">*</span>
              </label>
              <div className="space-y-3">
                {ingredients.map((ingredient, index) => (
                  <div key={index} className="flex gap-
3">
                    <input
                      type="text"
                      value={ingredient.name}
                      onChange={(e) =>
handleIngredientChange(index, 'name', e.target.value)}
                      placeholder="Nama bahan"
                      className="flex-1 px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500"
                    />
                    <input
                      type="text"
                      value={ingredient.quantity}
                      onChange={(e) =>
handleIngredientChange(index, 'quantity',
e.target.value)}
```

```
                      placeholder="Jumlah"
                      className="w-32 px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500"
                    />
                    <button
                      type="button"
                      onClick={() =>
removeIngredient(index)}
                      disabled={ingredients.length === 1}
                      className="p-3 text-red-600
hover:bg-red-50 rounded-xl transition-colors
disabled:opacity-50 disabled:cursor-not-allowed"
                    >
                      <X className="w-5 h-5" />
                    </button>
                  </div>
                ))}
                <button
                  type="button"
                  onClick={addIngredient}
                  className="w-full py-3 border-2 border-
dashed border-slate-300 rounded-xl text-slate-600
hover:border-blue-400 hover:text-blue-600 transition-
colors flex items-center justify-center gap-2"
                >
                  <Plus className="w-5 h-5" />
                  Tambah Bahan
                </button>
              </div>
            </div>

            {/* Steps */}
            <div>
              <label className="block text-sm font-medium
text-slate-700 mb-3">
                Langkah-langkah <span className="text-
red-500">*</span>
              </label>
              <div className="space-y-3">
                {steps.map((step, index) => (
                  <div key={index} className="flex gap-
3">
                    <div className="flex-shrink-0 w-10 h-
10 bg-blue-600 text-white rounded-full flex items-center
justify-center font-bold">
                      {index + 1}
                    </div>
                    <textarea
                      value={step}
                      onChange={(e) =>
handleStepChange(index, e.target.value)}
                      placeholder="Tulis langkah..."
                      rows={2}
```

```
                      className="flex-1 px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500 resize-none"
                    />
                    <button
                      type="button"
                      onClick={() => removeStep(index)}
                      disabled={steps.length === 1}
                      className="p-3 text-red-600
hover:bg-red-50 rounded-xl transition-colors
disabled:opacity-50 disabled:cursor-not-allowed self-
start"
                    >
                      <X className="w-5 h-5" />
                    </button>
                  </div>
                ))}
                <button
                  type="button"
                  onClick={addStep}
                  className="w-full py-3 border-2 border-
dashed border-slate-300 rounded-xl text-slate-600
hover:border-blue-400 hover:text-blue-600 transition-
colors flex items-center justify-center gap-2"
                >
                  <Plus className="w-5 h-5" />
                  Tambah Langkah
                </button>
              </div>
            </div>

            {/* Featured Toggle */}
            <div className="flex items-center gap-3 bg-
blue-50 p-4 rounded-xl">
              <input
                type="checkbox"
                id="is_featured"
                name="is_featured"
                checked={formData.is_featured}
                onChange={handleChange}
                className="w-5 h-5 text-blue-600 rounded
focus:ring-2 focus:ring-blue-500"
              />
              <label htmlFor="is_featured"
className="text-sm text-slate-700 cursor-pointer">
                Tandai sebagai resep unggulan
              </label>
            </div>

            {/* Submit Buttons */}
            <div className="flex flex-col md:flex-row
gap-4 pt-6">
              <button
                type="button"
                onClick={onBack}
                disabled={creating}
```

```
                  className="flex-1 px-6 py-3 border
border-slate-300 text-slate-700 rounded-xl hover:bg-
slate-50 transition-colors font-medium disabled:opacity-
50 disabled:cursor-not-allowed"
                >
                  Batal
                </button>
                <button
                  type="submit"
                  disabled={creating}
                  className="flex-1 px-6 py-3 bg-blue-600
text-white rounded-xl hover:bg-blue-700 transition-colors
font-medium disabled:opacity-50 disabled:cursor-not-
allowed flex items-center justify-center gap-2"
                >
                  {creating ? (
                    <>
                      <Loader className="w-5 h-5 animate-
spin" />
                      Membuat resep...
                    </>
                  ) : (
                    'Buat Resep'
                  )}
                </button>
              </div>
            </form>
            )}
          </div>
        </main>
      </div>
  );
}
```

EditRecipePage.jsx

```
// src/pages/EditRecipePage.jsx
import { useState, useEffect } from 'react';
import { ArrowLeft, Upload, X, Plus, Image as ImageIcon,
Loader } from 'lucide-react';
import recipeService from '../services/recipeService';
import uploadService from '../services/uploadService';

export default function EditRecipePage({ recipeId,
onBack, onSuccess }) {
  const [loading, setLoading] = useState(true);
  const [formData, setFormData] = useState({
    name: '',
    category: 'makanan',
    description: '',
    prep_time: '',
    cook_time: '',
    servings: '',
    difficulty: 'mudah',
```

```
    is_featured: false,
  });

  const [currentImageUrl, setCurrentImageUrl] =
useState('');
  const [imageFile, setImageFile] = useState(null);
  const [imagePreview, setImagePreview] = useState(null);
  const [ingredients, setIngredients] = useState([{ name:
'', quantity: '' }]);
  const [steps, setSteps] = useState(['']);

  const [uploading, setUploading] = useState(false);
  const [updating, setUpdating] = useState(false);
  const [error, setError] = useState('');

  // Load recipe data
  useEffect(() => {
    const loadRecipe = async () => {
      try {
        setLoading(true);
        const result = await
recipeService.getRecipeById(recipeId);

        if (result.success) {
          const recipe = result.data;
          setFormData({
            name: recipe.name || '',
            category: recipe.category || 'makanan',
            description: recipe.description || '',
            prep_time: recipe.prep_time || '',
            cook_time: recipe.cook_time || '',
            servings: recipe.servings || '',
            difficulty: recipe.difficulty || 'mudah',
            is_featured: recipe.is_featured || false,
          });

          setCurrentImageUrl(recipe.image_url || '');
          setIngredients(recipe.ingredients &&
recipe.ingredients.length > 0
            ? recipe.ingredients
            : [{ name: '', quantity: '' }]
          );

          // Convert steps to array of strings if they're
objects
          let stepsArray = [''];
          if (recipe.steps && recipe.steps.length > 0) {
            stepsArray = recipe.steps.map(step => {
              // If step is an object with a 'step'
property, extract it
              if (typeof step === 'object' && step.step)
{
                return step.step;
              }
              // If step is already a string, use it
```

```
              return typeof step === 'string' ? step :
'';
            });
          }
          setSteps(stepsArray);
        } else {
          throw new Error('Gagal memuat data resep');
        }
      } catch (err) {
        setError(err.message || 'Terjadi kesalahan saat
memuat resep');
      } finally {
        setLoading(false);
      }
    };

    if (recipeId) {
      loadRecipe();
    }
  }, [recipeId]);

  // Handle image selection
  const handleImageChange = (e) => {
    const file = e.target.files[0];
    if (!file) return;

    // Validate file size (max 5MB)
    if (file.size > 5 * 1024 * 1024) {
      setError('Ukuran file maksimal 5MB');
      return;
    }

    // Validate file type
    const allowedTypes = ['image/jpeg', 'image/jpg',
'image/png', 'image/webp'];
    if (!allowedTypes.includes(file.type)) {
      setError('Format file harus .jpg, .jpeg, .png, atau
.webp');
      return;
    }

    setImageFile(file);
    setImagePreview(URL.createObjectURL(file));
    setError('');
  };

  // Remove new image (revert to original)
  const handleRemoveNewImage = () => {
    setImageFile(null);
    setImagePreview(null);
    if (imagePreview) {
      URL.revokeObjectURL(imagePreview);
    }
  };

  // Remove original image
```

```
  const handleRemoveOriginalImage = () => {
    setCurrentImageUrl('');
  };

  // Handle form field changes
  const handleChange = (e) => {
    const { name, value, type, checked } = e.target;
    setFormData(prev => ({
      ...prev,
      [name]: type === 'checkbox' ? checked : value
    }));
  };

  // Handle ingredient changes
  const handleIngredientChange = (index, field, value) =>
{
    const newIngredients = [...ingredients];
    newIngredients[index][field] = value;
    setIngredients(newIngredients);
  };

  const addIngredient = () => {
    setIngredients([...ingredients, { name: '', quantity:
'' }]);
  };

  const removeIngredient = (index) => {
    if (ingredients.length > 1) {
      setIngredients(ingredients.filter((_, i) => i !==
index));
    }
  };

  // Handle step changes
  const handleStepChange = (index, value) => {
    const newSteps = [...steps];
    newSteps[index] = value;
    setSteps(newSteps);
  };

  const addStep = () => {
    setSteps([...steps, '']);
  };

  const removeStep = (index) => {
    if (steps.length > 1) {
      setSteps(steps.filter((_, i) => i !== index));
    }
  };

  // Validate form
  const validateForm = () => {
    if (!formData.name.trim()) {
      setError('Nama resep wajib diisi');
      return false;
    }
```

```
    if (formData.name.trim().length < 3) {
      setError('Nama resep minimal 3 karakter');
      return false;
    }

    if (!formData.prep_time || formData.prep_time <= 0) {
      setError('Waktu persiapan harus lebih dari 0');
      return false;
    }

    if (!formData.cook_time || formData.cook_time <= 0) {
      setError('Waktu memasak harus lebih dari 0');
      return false;
    }

    if (!formData.servings || formData.servings <= 0) {
      setError('Jumlah porsi harus lebih dari 0');
      return false;
    }

    // Validate ingredients
    const validIngredients = ingredients.filter(ing =>
ing.name.trim() && ing.quantity.trim());
    if (validIngredients.length === 0) {
      setError('Minimal harus ada 1 bahan');
      return false;
    }

    // Validate steps
    const validSteps = steps.filter(step => {
      // Ensure step is a string before calling trim
      if (typeof step === 'string') {
        return step.trim();
      }
      // If step is an object, check if it has a 'step'
property
      if (typeof step === 'object' && step.step) {
        return step.step.trim();
      }
      return false;
    });
    if (validSteps.length === 0) {
      setError('Minimal harus ada 1 langkah');
      return false;
    }

    return true;
  };

  // Submit form (PATCH - partial update)
  const handleSubmit = async (e) => {
    e.preventDefault();
    setError('');

    if (!validateForm()) {
```

```
      return;
    }

    try {
      setUpdating(true);

      // Prepare update data
      const updateData = {};

      // Step 1: Upload new image if selected
      if (imageFile) {
        setUploading(true);
        const uploadResult = await
uploadService.uploadImage(imageFile);
        if (uploadResult.success) {
          updateData.image_url = uploadResult.data.url;
        } else {
          throw new Error('Gagal upload gambar');
        }
        setUploading(false);
      } else if (!currentImageUrl && !imageFile) {
        // If original image was removed and no new image
        updateData.image_url = '';
      }

      // Step 2: Add other fields
      const validIngredients = ingredients.filter(ing =>
ing.name.trim() && ing.quantity.trim());
      const validSteps = steps.filter(step => {
        if (typeof step === 'string') {
          return step.trim();
        }
        if (typeof step === 'object' && step.step) {
          return step.step.trim();
        }
        return false;
      }).map(step => {
        // Convert to string if it's an object
        if (typeof step === 'object' && step.step) {
          return step.step;
        }
        return step;
      });

      updateData.name = formData.name.trim();
      updateData.category = formData.category;
      updateData.description =
formData.description.trim();
      updateData.prep_time =
parseInt(formData.prep_time);
      updateData.cook_time =
parseInt(formData.cook_time);
      updateData.servings = parseInt(formData.servings);
      updateData.difficulty = formData.difficulty;
      updateData.is_featured = formData.is_featured;
      updateData.ingredients = validIngredients;
```

```
      updateData.steps = validSteps;

      // Step 3: Update recipe using PUT
      const result = await
recipeService.updateRecipe(recipeId, updateData);

      if (result.success) {
        alert('Resep berhasil diperbarui!');
        if (onSuccess) {
          onSuccess(result.data);
        } else if (onBack) {
          onBack();
        }
      } else {
        throw new Error(result.message || 'Gagal
memperbarui resep');
      }
    } catch (err) {
      setError(err.message || 'Terjadi kesalahan saat
memperbarui resep');
    } finally {
      setUpdating(false);
      setUploading(false);
    }
  };

  if (loading) {
    return (
      <div className="min-h-screen bg-gradient-to-br
from-blue-50 via-white to-indigo-50 flex items-center
justify-center">
        <div className="text-center">
          <Loader className="w-12 h-12 text-blue-600
animate-spin mx-auto mb-4" />
          <p className="text-slate-600">Memuat data
resep...</p>
        </div>
      </div>
    );
  }

  return (
    <div className="min-h-screen bg-gradient-to-br from-
blue-50 via-white to-indigo-50 pb-20 md:pb-8">
      {/* Header */}
      <div className="sticky top-0 z-10 bg-white/80
backdrop-blur-md border-b border-slate-200 shadow-sm">
        <div className="max-w-4xl mx-auto px-4 py-4">
          <button
            onClick={onBack}
            className="flex items-center gap-2 text-
slate-700 hover:text-slate-900 transition-colors"
          >
            <ArrowLeft className="w-5 h-5" />
            <span className="font-medium">Kembali</span>
          </button>
```

```
        </div>
      </div>

      <main className="max-w-4xl mx-auto px-4 py-8">
        <div className="bg-white/60 backdrop-blur-sm
rounded-3xl p-6 md:p-8 shadow-xl border border-white/40">
          <h1 className="text-3xl font-bold text-slate-
800 mb-2">Edit Resep</h1>
          <p className="text-slate-600 mb-8">Perbarui
informasi resepmu</p>

          {/* Error Message */}
          {error && (
            <div className="mb-6 bg-red-50 border border-
red-200 rounded-xl p-4">
              <p className="text-red-600 text-
sm">{error}</p>
            </div>
          )}

          <form onSubmit={handleSubmit} className="space-
y-8">
            {/* Image Upload */}
            <div>
              <label className="block text-sm font-medium
text-slate-700 mb-3">
                Foto Resep
              </label>

              {/* Show new image preview if selected */}
              {imagePreview ? (
                <div className="relative">
                  <img
                    src={imagePreview}
                    alt="Preview"
                    className="w-full h-64 object-cover
rounded-xl"
                  />
                  <button
                    type="button"
                    onClick={handleRemoveNewImage}
                    className="absolute top-4 right-4 p-2
bg-red-500 text-white rounded-full hover:bg-red-600
transition-colors shadow-lg"
                  >
                    <X className="w-5 h-5" />
                  </button>
                  <div className="absolute bottom-4 left-
4 px-3 py-1 bg-blue-600 text-white text-sm rounded-full">
                    Gambar Baru
                  </div>
                </div>
              ) : currentImageUrl &&
currentImageUrl.trim() !== '' ? (
                /* Show current image */
                <div className="relative">
```

```
                  <img
                    src={currentImageUrl}
                    alt="Current"
                    className="w-full h-64 object-cover
rounded-xl"
                  />
                  <button
                    type="button"
                    onClick={handleRemoveOriginalImage}
                    className="absolute top-4 right-4 p-2
bg-red-500 text-white rounded-full hover:bg-red-600
transition-colors shadow-lg"
                  >
                    <X className="w-5 h-5" />
                  </button>
                  <div className="absolute bottom-4 left-
4">
                    <input
                      type="file"

accept="image/jpeg,image/jpg,image/png,image/webp"
                      onChange={handleImageChange}
                      className="hidden"
                      id="image-change"
                    />
                    <label
                      htmlFor="image-change"
                      className="px-4 py-2 bg-white/90
backdrop-blur-sm text-slate-700 rounded-full hover:bg-
white transition-colors cursor-pointer text-sm font-
medium shadow-lg inline-flex items-center gap-2"
                    >
                      <Upload className="w-4 h-4" />
                      Ganti Gambar
                    </label>
                  </div>
                </div>
              ) : (
                /* No image - show upload area */
                <div className="border-2 border-dashed
border-slate-300 rounded-xl p-8 text-center hover:border-
blue-400 transition-colors">
                  <input
                    type="file"

accept="image/jpeg,image/jpg,image/png,image/webp"
                    onChange={handleImageChange}
                    className="hidden"
                    id="image-upload"
                  />
                  <label
                    htmlFor="image-upload"
                    className="cursor-pointer flex flex-
col items-center gap-3"
                  >
```

```
                    <div className="w-16 h-16 bg-blue-100
rounded-full flex items-center justify-center">
                      <ImageIcon className="w-8 h-8 text-
blue-600" />
                    </div>
                    <div>
                      <p className="text-slate-700 font-
medium">Klik untuk upload foto</p>
                      <p className="text-sm text-slate-
500 mt-1">
                        Maksimal 5MB (.jpg, .jpeg, .png,
.webp)
                      </p>
                    </div>
                  </label>
                </div>
              )}
            </div>

            {/* Basic Info */}
            <div className="grid md:grid-cols-2 gap-6">
              <div>
                <label className="block text-sm font-
medium text-slate-700 mb-2">
                  Nama Resep <span className="text-red-
500">*</span>
                </label>
                <input
                  type="text"
                  name="name"
                  value={formData.name}
                  onChange={handleChange}
                  placeholder="Nasi Goreng Spesial"
                  className="w-full px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500"
                  required
                />
              </div>

              <div>
                <label className="block text-sm font-
medium text-slate-700 mb-2">
                  Kategori <span className="text-red-
500">*</span>
                </label>
                <select
                  name="category"
                  value={formData.category}
                  onChange={handleChange}
                  className="w-full px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500"
                >
                  <option
value="makanan">Makanan</option>
```

```
                  <option
value="minuman">Minuman</option>
                </select>
              </div>
            </div>

            {/* Description */}
            <div>
              <label className="block text-sm font-medium
text-slate-700 mb-2">
                Deskripsi
              </label>
              <textarea
                name="description"
                value={formData.description}
                onChange={handleChange}
                placeholder="Ceritakan tentang resep
ini..."
                rows={4}
                className="w-full px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500 resize-none"
              />
            </div>

            {/* Time & Servings */}
            <div className="grid md:grid-cols-3 gap-6">
              <div>
                <label className="block text-sm font-
medium text-slate-700 mb-2">
                  Waktu Persiapan (menit) <span
className="text-red-500">*</span>
                </label>
                <input
                  type="number"
                  name="prep_time"
                  value={formData.prep_time}
                  onChange={handleChange}
                  placeholder="15"
                  min="1"
                  className="w-full px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500"
                  required
                />
              </div>

              <div>
                <label className="block text-sm font-
medium text-slate-700 mb-2">
                  Waktu Memasak (menit) <span
className="text-red-500">*</span>
                </label>
                <input
                  type="number"
                  name="cook_time"
```

```
                  value={formData.cook_time}
                  onChange={handleChange}
                  placeholder="20"
                  min="1"
                  className="w-full px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500"
                  required
                />
              </div>

              <div>
                <label className="block text-sm font-
medium text-slate-700 mb-2">
                  Porsi (orang) <span className="text-
red-500">*</span>
                </label>
                <input
                  type="number"
                  name="servings"
                  value={formData.servings}
                  onChange={handleChange}
                  placeholder="4"
                  min="1"
                  className="w-full px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500"
                  required
                />
              </div>
            </div>

            {/* Difficulty */}
            <div>
              <label className="block text-sm font-medium
text-slate-700 mb-2">
                Tingkat Kesulitan <span className="text-
red-500">*</span>
              </label>
              <select
                name="difficulty"
                value={formData.difficulty}
                onChange={handleChange}
                className="w-full px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500"
              >
                <option value="mudah">Mudah</option>
                <option value="sedang">Sedang</option>
                <option value="sulit">Sulit</option>
              </select>
            </div>

            {/* Ingredients */}
            <div>
```

```
              <label className="block text-sm font-medium
text-slate-700 mb-3">
                Bahan-bahan <span className="text-red-
500">*</span>
              </label>
              <div className="space-y-3">
                {ingredients.map((ingredient, index) => (
                  <div key={index} className="flex gap-
3">
                    <input
                      type="text"
                      value={ingredient.name}
                      onChange={(e) =>
handleIngredientChange(index, 'name', e.target.value)}
                      placeholder="Nama bahan"
                      className="flex-1 px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500"
                    />
                    <input
                      type="text"
                      value={ingredient.quantity}
                      onChange={(e) =>
handleIngredientChange(index, 'quantity',
e.target.value)}
                      placeholder="Jumlah"
                      className="w-32 px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500"
                    />
                    <button
                      type="button"
                      onClick={() =>
removeIngredient(index)}
                      disabled={ingredients.length === 1}
                      className="p-3 text-red-600
hover:bg-red-50 rounded-xl transition-colors
disabled:opacity-50 disabled:cursor-not-allowed"
                    >
                      <X className="w-5 h-5" />
                    </button>
                  </div>
                ))}
                <button
                  type="button"
                  onClick={addIngredient}
                  className="w-full py-3 border-2 border-
dashed border-slate-300 rounded-xl text-slate-600
hover:border-blue-400 hover:text-blue-600 transition-
colors flex items-center justify-center gap-2"
                >
                  <Plus className="w-5 h-5" />
                  Tambah Bahan
                </button>
              </div>
            </div>
```

```
            {/* Steps */}
            <div>
              <label className="block text-sm font-medium
text-slate-700 mb-3">
                Langkah-langkah <span className="text-
red-500">*</span>
              </label>
              <div className="space-y-3">
                {steps.map((step, index) => (
                  <div key={index} className="flex gap-
3">
                    <div className="shrink-0 w-10 h-10
bg-blue-600 text-white rounded-full flex items-center
justify-center font-bold">
                      {index + 1}
                    </div>
                    <textarea
                      value={step}
                      onChange={(e) =>
handleStepChange(index, e.target.value)}
                      placeholder="Tulis langkah..."
                      rows={2}
                      className="flex-1 px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500 resize-none"
                    />
                    <button
                      type="button"
                      onClick={() => removeStep(index)}
                      disabled={steps.length === 1}
                      className="p-3 text-red-600
hover:bg-red-50 rounded-xl transition-colors
disabled:opacity-50 disabled:cursor-not-allowed self-
start"
                    >
                      <X className="w-5 h-5" />
                    </button>
                  </div>
                ))}
                <button
                  type="button"
                  onClick={addStep}
                  className="w-full py-3 border-2 border-
dashed border-slate-300 rounded-xl text-slate-600
hover:border-blue-400 hover:text-blue-600 transition-
colors flex items-center justify-center gap-2"
                >
                  <Plus className="w-5 h-5" />
                  Tambah Langkah
                </button>
              </div>
            </div>

            {/* Featured Toggle */}
```

```
            <div className="flex items-center gap-3 bg-
blue-50 p-4 rounded-xl">
              <input
                type="checkbox"
                id="is_featured"
                name="is_featured"
                checked={formData.is_featured}
                onChange={handleChange}
                className="w-5 h-5 text-blue-600 rounded
focus:ring-2 focus:ring-blue-500"
              />
              <label htmlFor="is_featured"
className="text-sm text-slate-700 cursor-pointer">
                Tandai sebagai resep unggulan
              </label>
            </div>

            {/* Submit Buttons */}
            <div className="flex flex-col md:flex-row
gap-4 pt-6">
              <button
                type="button"
                onClick={onBack}
                disabled={updating || uploading}
                className="flex-1 px-6 py-3 border
border-slate-300 text-slate-700 rounded-xl hover:bg-
slate-50 transition-colors font-medium disabled:opacity-
50 disabled:cursor-not-allowed"
              >
                Batal
              </button>
              <button
                type="submit"
                disabled={updating || uploading}
                className="flex-1 px-6 py-3 bg-blue-600
text-white rounded-xl hover:bg-blue-700 transition-colors
font-medium disabled:opacity-50 disabled:cursor-not-
allowed flex items-center justify-center gap-2"
              >
                {uploading ? (
                  <>
                    <Loader className="w-5 h-5 animate-
spin" />
                    Uploading gambar...
                  </>
                ) : updating ? (
                  <>
                    <Loader className="w-5 h-5 animate-
spin" />
                    Memperbarui resep...
                  </>
                ) : (
                  'Simpan Perubahan'
                )}
              </button>
            </div>
```

```
          </form>
        </div>
      </main>
    </div>
  );
}
```

HomePage.jsx

```
// src/pages/HomePage.jsx
import { useRecipes } from '../hooks/useRecipes';
import HeroSection from '../components/home/HeroSection';
import FeaturedMakananSection from
'../components/home/FeaturedMakananSection';
import FeaturedMinumanSection from
'../components/home/FeaturedMinumanSection';

export default function HomePage({ onRecipeClick,
onNavigate }) {
  // Fetch featured makanan (food) recipes from API
  const {
    recipes: featuredMakanan,
    loading: loadingMakanan,
    error: errorMakanan
  } = useRecipes({
    category: 'makanan',
    limit: 3,
    sort_by: 'created_at',
    order: 'desc'
  });

  // Fetch featured minuman (drinks) recipes from API
  const {
    recipes: featuredMinuman,
    loading: loadingMinuman,
    error: errorMinuman
  } = useRecipes({
    category: 'minuman',
    limit: 2,
    sort_by: 'created_at',
    order: 'desc'
  });

  return (
    <div className="min-h-screen bg-gradient-to-br from-
blue-50 via-white to-indigo-50">
      <HeroSection />

      <div className="max-w-7xl mx-auto px-4 sm:px-6
lg:px-8 py-12 space-y-16">
        {/* Featured Makanan Section */}
        <FeaturedMakananSection
          recipes={featuredMakanan}
          loading={loadingMakanan}
          error={errorMakanan}
```

```
          onRecipeClick={onRecipeClick}
          onNavigate={onNavigate}
        />

        {/* Featured Minuman Section */}
        <FeaturedMinumanSection
          recipes={featuredMinuman}
          loading={loadingMinuman}
          error={errorMinuman}
          onRecipeClick={onRecipeClick}
          onNavigate={onNavigate}
        />
      </div>
    </div>
  );
}
```

MakananPage.jsx

```
// src/pages/MakananPage.jsx
import { useState } from 'react';
import { useRecipes } from '../hooks/useRecipes';
import RecipeGrid from
'../components/makanan/RecipeGrid';
import AdvancedFilter from
'../components/common/AdvancedFilter';

export default function MakananPage({ onRecipeClick }) {
  const [searchQuery, setSearchQuery] = useState('');
  const [filters, setFilters] = useState({
    difficulty: '',
    sortBy: 'created_at',
    order: 'desc',
    prepTimeMax: '',
  });
  const [page, setPage] = useState(1);

  // Fetch recipes from API with all filters
  const { recipes, loading, error, pagination, refetch }
= useRecipes({
    category: 'makanan',
    search: searchQuery || undefined,
    difficulty: filters.difficulty || undefined,
    page,
    limit: 12,
    sort_by: filters.sortBy,
    order: filters.order
  });

  const handleSearchChange = (query) => {
    setSearchQuery(query);
    setPage(1); // Reset to first page on search
  };
```

```
  const handleFilterChange = (newFilters) => {
    setFilters(newFilters);
    setPage(1); // Reset to first page on filter change
  };

  // Client-side filter for prep time (since API might
not support it)
  const filteredRecipes = filters.prepTimeMax
    ? recipes.filter(recipe => recipe.prep_time <=
parseInt(filters.prepTimeMax))
    : recipes;

  return (
    <div className="min-h-screen bg-linear-to-br from-
blue-50 via-white to-indigo-50 pb-20 md:pb-8">
      <main className="max-w-7xl mx-auto px-4 md:px-8 py-
8 md:py-12">
        {/* Header */}
        <div className="mb-8 text-center">
          <h1 className="text-3xl md:text-5xl font-bold
text-slate-800 mb-4">
            Resep Makanan
          </h1>
          <p className="text-slate-600 max-w-2xl mx-
auto">
            Temukan berbagai resep makanan nusantara yang
lezat
          </p>
        </div>

        {/* Advanced Filter */}
        <AdvancedFilter
          onSearchChange={handleSearchChange}
          onFilterChange={handleFilterChange}
          initialFilters={{ ...filters, search:
searchQuery }}
        />

        {/* Loading State */}
        {loading && (
          <div className="text-center py-12">
            <div className="animate-spin rounded-full h-
12 w-12 border-b-2 border-indigo-600 mx-auto"></div>
            <p className="mt-4 text-gray-600">Memuat
resep...</p>
          </div>
        )}

        {/* Error State */}
        {error && (
          <div className="text-center py-12">
            <div className="bg-red-50 border border-red-
200 rounded-lg p-6">
              <p className="text-red-600 font-semibold
mb-2">Terjadi Kesalahan</p>
              <p className="text-red-500">{error}</p>
```

```
              <button
                onClick={refetch}
                className="mt-4 px-4 py-2 bg-red-600
text-white rounded-lg hover:bg-red-700"
              >
                Coba Lagi
              </button>
            </div>
          </div>
        )}

        {/* Recipes Grid */}
        {!loading && !error && (
          <>
            {filteredRecipes.length === 0 ? (
              <div className="text-center py-12">
                <p className="text-gray-600 text-lg">
                  Tidak ada resep ditemukan
                </p>
                <p className="text-gray-500 mt-2">
                  Coba ubah filter atau kata kunci
pencarian
                </p>
              </div>
            ) : (
              <RecipeGrid recipes={filteredRecipes}
onRecipeClick={onRecipeClick} />
            )}

            {/* Pagination */}
            {pagination && pagination.total_pages > 1 &&
(
              <div className="mt-12 flex flex-col
md:flex-row items-center justify-center gap-4">
                <button
                  onClick={() => setPage(p => Math.max(1,
p - 1))}
                  disabled={page === 1}
                  className="px-6 py-3 bg-white/80
backdrop-blur border border-slate-300 rounded-xl
hover:bg-blue-50 hover:border-blue-400 disabled:opacity-
50 disabled:cursor-not-allowed transition-all font-medium
text-slate-700"
                >
                  ← Sebelumnya
                </button>

                <div className="flex flex-col md:flex-row
items-center gap-2 bg-white/60 backdrop-blur px-4 py-2
rounded-xl border border-white/40">
                  <span className="text-slate-700 font-
semibold">
                    Halaman {pagination.page} dari
{pagination.total_pages}
                  </span>
```

```
                  <span className="text-slate-500 text-
sm">
                    ({pagination.total} resep)
                  </span>
                </div>

                <button
                  onClick={() => setPage(p => p + 1)}
                  disabled={page ===
pagination.total_pages}
                  className="px-6 py-3 bg-white/80
backdrop-blur border border-slate-300 rounded-xl
hover:bg-blue-50 hover:border-blue-400 disabled:opacity-
50 disabled:cursor-not-allowed transition-all font-medium
text-slate-700"
                >
                  Selanjutnya →
                </button>
              </div>
            )}
          </>
        )}
      </main>
    </div>
  );
}
```

MinumanPage.jsx

```
// src/pages/MinumanPage.jsx
import { useState } from 'react';
import { useRecipes } from '../hooks/useRecipes';
import RecipeGrid from
'../components/minuman/RecipeGrid';
import AdvancedFilter from
'../components/common/AdvancedFilter';

export default function MinumanPage({ onRecipeClick }) {
  const [searchQuery, setSearchQuery] = useState('');
  const [filters, setFilters] = useState({
    difficulty: '',
    sortBy: 'created_at',
    order: 'desc',
    prepTimeMax: '',
  });
  const [page, setPage] = useState(1);

  // Fetch recipes from API with all filters
  const { recipes, loading, error, pagination, refetch }
= useRecipes({
    category: 'minuman',
    search: searchQuery || undefined,
    difficulty: filters.difficulty || undefined,
    page,
```

```
    limit: 12,
    sort_by: filters.sortBy,
    order: filters.order
  });

  const handleSearchChange = (query) => {
    setSearchQuery(query);
    setPage(1); // Reset to first page on search
  };

  const handleFilterChange = (newFilters) => {
    setFilters(newFilters);
    setPage(1); // Reset to first page on filter change
  };

  // Client-side filter for prep time (since API might
not support it)
  const filteredRecipes = filters.prepTimeMax
    ? recipes.filter(recipe => recipe.prep_time <=
parseInt(filters.prepTimeMax))
    : recipes;

  return (
    <div className="min-h-screen bg-linear-to-br from-
green-50 via-white to-cyan-50 pb-20 md:pb-8">
      <main className="max-w-7xl mx-auto px-4 md:px-8 py-
8 md:py-12">
        {/* Header */}
        <div className="mb-8 text-center">
          <h1 className="text-3xl md:text-5xl font-bold
text-slate-800 mb-4">
            Resep Minuman
          </h1>
          <p className="text-slate-600 max-w-2xl mx-
auto">
            Temukan berbagai resep minuman segar dan
nikmat
          </p>
        </div>

        {/* Advanced Filter */}
        <AdvancedFilter
          onSearchChange={handleSearchChange}
          onFilterChange={handleFilterChange}
          initialFilters={{ ...filters, search:
searchQuery }}
        />

        {/* Loading State */}
        {loading && (
          <div className="text-center py-12">
            <div className="animate-spin rounded-full h-
12 w-12 border-b-2 border-green-600 mx-auto"></div>
            <p className="mt-4 text-slate-600">Memuat
resep...</p>
          </div>
```

```
        )}

        {/* Error State */}
        {error && (
          <div className="text-center py-12">
            <div className="bg-red-50 border border-red-
200 rounded-lg p-6">
              <p className="text-red-600 font-semibold
mb-2">Terjadi Kesalahan</p>
              <p className="text-red-500">{error}</p>
              <button
                onClick={refetch}
                className="mt-4 px-4 py-2 bg-red-600
text-white rounded-lg hover:bg-red-700"
              >
                Coba Lagi
              </button>
            </div>
          </div>
        )}

        {/* Recipes Grid */}
        {!loading && !error && (
          <>
            {recipes.length === 0 ? (
              <div className="text-center py-12">
                <p className="text-gray-600 text-lg">
                  Tidak ada resep ditemukan
                </p>
                <p className="text-gray-500 mt-2">
                  Coba ubah filter atau kata kunci
pencarian
                </p>
              </div>
            ) : (
              <RecipeGrid recipes={filteredRecipes}
onRecipeClick={onRecipeClick} />
            )}

            {/* Pagination */}
            {pagination && pagination.total_pages > 1 &&
(
              <div className="mt-12 flex flex-col
md:flex-row items-center justify-center gap-4">
                <button
                  onClick={() => setPage(p => Math.max(1,
p - 1))}
                  disabled={page === 1}
                  className="px-6 py-3 bg-white/80
backdrop-blur border border-slate-300 rounded-xl
hover:bg-green-50 hover:border-green-400
disabled:opacity-50 disabled:cursor-not-allowed
transition-all font-medium text-slate-700"
                >
                  ← Sebelumnya
                </button>
```

```
                <div className="flex flex-col md:flex-row
items-center gap-2 bg-white/60 backdrop-blur px-4 py-2
rounded-xl border border-white/40">
                  <span className="text-slate-700 font-
semibold">
                    Halaman {pagination.page} dari
{pagination.total_pages}
                  </span>
                  <span className="text-slate-500 text-
sm">
                    ({pagination.total} resep)
                  </span>
                </div>

                <button
                  onClick={() => setPage(p => p + 1)}
                  disabled={page ===
pagination.total_pages}
                  className="px-6 py-3 bg-white/80
backdrop-blur border border-slate-300 rounded-xl
hover:bg-green-50 hover:border-green-400
disabled:opacity-50 disabled:cursor-not-allowed
transition-all font-medium text-slate-700"
                >
                  Selanjutnya →
                </button>
              </div>
            )}
          </>
        )}
      </main>
    </div>
  );
}
```

ProfilePage.jsx

```
// src/pages/ProfilePage.jsx
export default function ProfilePage() {
  return (
    <div className="p-4 md:p-8 pb-20 md:pb-8">
      <div className="max-w-7xl mx-auto">
        <h1 className="text-2xl md:text-3xl font-bold
text-gray-800 mb-6">
          Profile Pengguna
        </h1>
        <div className="bg-white rounded-lg shadow-lg p-
6">
          <p className="text-gray-600">
            Konten halaman profile akan diisi di sini...
          </p>
        </div>
      </div>
    </div>
```

```
  );
}
```

SplashScreen.jsx

```
// src/pages/SplashScreen.jsx
import React, { useState, useEffect } from 'react';
import BackgroundPattern from
'../components/splash/BackgroundPattern';
import FloatingElements from
'../components/splash/FloatingElements';
import LogoContainer from
'../components/splash/LogoContainer';
import TitleSection from
'../components/splash/TitleSection';
import LoadingAnimation from
'../components/splash/LoadingAnimation';
import Footer from '../components/splash/Footer';

export default function SplashScreen({ onComplete }) {
  const [progress, setProgress] = useState(0);
  const [isVisible, setIsVisible] = useState(true);
  const [fadeIn, setFadeIn] = useState(false);
  const [fadeOut, setFadeOut] = useState(false);

  useEffect(() => {
    setTimeout(() => {
      setFadeIn(true);
    }, 100);

    const interval = setInterval(() => {
      setProgress((prev) => {
        if (prev >= 100) {
          clearInterval(interval);
          setTimeout(() => {
            setFadeOut(true);
            setTimeout(() => {
              setIsVisible(false);
              setTimeout(() => {
                if (typeof onComplete === 'function')
onComplete();
              }, 100);
            }, 600);
          }, 800);
          return 100;
        }
        const nextProgress = prev + 6;
        return nextProgress > 100 ? 100 : nextProgress;
      });
    }, 120);

    return () => clearInterval(interval);
  }, [onComplete]);

  if (!isVisible) return null;
```

```
  return (
    <div
      className={`fixed inset-0 z-50 flex flex-col items-
center justify-center bg-gradient-to-br from-blue-50 via-
white to-blue-100 px-4 sm:px-6 transition-all duration-
600 ease-out ${
        !fadeIn ? 'opacity-0 scale-95' : 'opacity-100
scale-100'
      } ${
        fadeOut ? 'opacity-0 scale-105' : ''
      }`}
    >

      <BackgroundPattern fadeOut={fadeOut} />
      <FloatingElements fadeOut={fadeOut} />

      <div className={`relative z-10 flex flex-col items-
center justify-center max-w-xs sm:max-w-lg w-full
transition-all duration-800 ${
        !fadeIn ? 'opacity-0 translate-y-8' : 'opacity-
100 translate-y-0'
      } ${
        fadeOut ? 'opacity-0 -translate-y-8' : ''
      }`}>

        <LogoContainer />
        <TitleSection fadeIn={fadeIn} />
        <LoadingAnimation fadeIn={fadeIn}
progress={progress} />

      </div>

      <Footer fadeOut={fadeOut} fadeIn={fadeIn} />
    </div>
  );
}
```

RecipeDetailPage.jsx

```
// src/pages/RecipeDetailPage.jsx
import { useState } from 'react';
import { useRecipe } from '../hooks/useRecipes';
import { useReviews, useCreateReview } from
'../hooks/useReviews';
import { useIsFavorited } from '../hooks/useFavorites';
import { getUserIdentifier } from
'../hooks/useFavorites';
import { formatDate, getDifficultyColor, getStarRating }
from '../utils/helpers';
import { Heart, Clock, Users, ChefHat } from 'lucide-
react';

export default function RecipeDetailPage({ recipeId,
onBack }) {
```

```
  const { recipe, loading: recipeLoading, error:
recipeError } = useRecipe(recipeId);
  const { reviews, loading: reviewsLoading, refetch:
refetchReviews } = useReviews(recipeId);
  const { createReview, loading: createLoading } =
useCreateReview();
  const { isFavorited, loading: favLoading,
toggleFavorite } = useIsFavorited(recipeId);

  const [rating, setRating] = useState(5);
  const [comment, setComment] = useState('');

  const handleSubmitReview = async (e) => {
    e.preventDefault();

    const reviewData = {
      user_identifier: getUserIdentifier(),
      rating,
      comment: comment.trim(),
    };

    const result = await createReview(recipeId,
reviewData);
    if (result) {
      setComment('');
      setRating(5);
      refetchReviews();
    }
  };

  const handleToggleFavorite = async () => {
    await toggleFavorite();
  };

  if (recipeLoading) {
    return (
      <div className="min-h-screen flex items-center
justify-center">
        <div className="text-center">
          <div className="animate-spin rounded-full h-12
w-12 border-b-2 border-indigo-600 mx-auto"></div>
          <p className="mt-4 text-gray-600">Memuat
resep...</p>
        </div>
      </div>
    );
  }

  if (recipeError) {
    return (
      <div className="min-h-screen flex items-center
justify-center">
        <div className="text-center">
          <p className="text-red-600 mb-4">Error:
{recipeError}</p>
          <button
```

```
            onClick={onBack}
            className="px-4 py-2 bg-indigo-600 text-white
rounded-lg hover:bg-indigo-700"
          >
            Kembali
          </button>
        </div>
      </div>
    );
  }

  if (!recipe) {
    return (
      <div className="min-h-screen flex items-center
justify-center">
        <p className="text-gray-600">Resep tidak
ditemukan</p>
      </div>
    );
  }

  return (
    <div className="min-h-screen bg-linear-to-br from-
blue-50 via-white to-indigo-50 pb-20 md:pb-8">
      {/* Header */}
      <div className="bg-white shadow-sm sticky top-0 z-
10">
        <div className="max-w-7xl mx-auto px-4 py-4 flex
items-center justify-between">
          <button
            onClick={onBack}
            className="text-indigo-600 hover:text-indigo-
700 font-medium"
          >
            ← Kembali
          </button>
          <button
            onClick={handleToggleFavorite}
            disabled={favLoading}
            className={`p-2 rounded-full transition-
colors ${
              isFavorited
                ? 'bg-red-100 text-red-600'
                : 'bg-gray-100 text-gray-400'
            }`}
          >
            <Heart className={isFavorited ? 'fill-
current' : ''} />
          </button>
        </div>
      </div>

      <div className="max-w-4xl mx-auto px-4 py-8">
        {/* Recipe Image */}
        <div className="mb-8">
          <img
```

```
            src={recipe.image_url}
            alt={recipe.name}
            className="w-full h-96 object-cover rounded-
2xl shadow-lg"
          />
        </div>

        {/* Recipe Header */}
        <div className="bg-white rounded-2xl shadow-lg p-
6 md:p-8 mb-6">
          <h1 className="text-3xl md:text-4xl font-bold
text-gray-900 mb-4">
            {recipe.name}
          </h1>

          <div className="flex flex-wrap gap-2 mb-4">
            <span className={`px-3 py-1 rounded-full
text-sm font-medium
${getDifficultyColor(recipe.difficulty)}`}>
              {recipe.difficulty}
            </span>
            <span className="px-3 py-1 rounded-full text-
sm font-medium bg-blue-100 text-blue-800">
              {recipe.category}
            </span>
            {recipe.is_featured && (
              <span className="px-3 py-1 rounded-full
text-sm font-medium bg-yellow-100 text-yellow-800">
                ⭐ Featured
              </span>
            )}
          </div>

          <p className="text-gray-600 mb-
6">{recipe.description}</p>

          {/* Recipe Info */}
          <div className="grid grid-cols-2 md:grid-cols-4
gap-4">
            <div className="flex items-center gap-2">
              <Clock className="w-5 h-5 text-indigo-600"
/>
              <div>
                <p className="text-sm text-gray-
500">Persiapan</p>
                <p className="font-
semibold">{recipe.prep_time} menit</p>
              </div>
            </div>
            <div className="flex items-center gap-2">
              <ChefHat className="w-5 h-5 text-indigo-
600" />
              <div>
                <p className="text-sm text-gray-
500">Memasak</p>
```

```
                <p className="font-
semibold">{recipe.cook_time} menit</p>
              </div>
            </div>
            <div className="flex items-center gap-2">
              <Users className="w-5 h-5 text-indigo-600"
/>
              <div>
                <p className="text-sm text-gray-
500">Porsi</p>
                <p className="font-
semibold">{recipe.servings} orang</p>
              </div>
            </div>
            <div className="flex items-center gap-2">
              <Heart className="w-5 h-5 text-indigo-600"
/>
              <div>
                <p className="text-sm text-gray-
500">Rating</p>
                <p className="font-
semibold">{recipe.average_rating?.toFixed(1) || 'N/A'}
({recipe.review_count})</p>
              </div>
            </div>
          </div>
        </div>

        {/* Ingredients */}
        <div className="bg-white rounded-2xl shadow-lg p-
6 md:p-8 mb-6">
          <h2 className="text-2xl font-bold text-gray-900
mb-4">Bahan-bahan</h2>
          <ul className="space-y-2">
            {recipe.ingredients?.map((ingredient, index)
=> (
              <li key={ingredient.id || index}
className="flex items-start">
                <span className="text-indigo-600 mr-
2">•</span>
                <span className="text-gray-700">
                  {ingredient.name} -
{ingredient.quantity}
                </span>
              </li>
            ))}
          </ul>
        </div>

        {/* Steps */}
        <div className="bg-white rounded-2xl shadow-lg p-
6 md:p-8 mb-6">
          <h2 className="text-2xl font-bold text-gray-900
mb-4">Langkah-langkah</h2>
          <ol className="space-y-4">
            {recipe.steps?.map((step) => (
```

```
              <li key={step.id} className="flex gap-4">
                <span className="shrink-0 w-8 h-8 bg-
indigo-600 text-white rounded-full flex items-center
justify-center font-semibold">
                  {step.step_number}
                </span>
                <p className="text-gray-700 flex-1 pt-
1">{step.instruction}</p>
              </li>
            ))}
          </ol>
        </div>

        {/* Reviews Section */}
        <div className="bg-white rounded-2xl shadow-lg p-
6 md:p-8 mb-6">
          <h2 className="text-2xl font-bold text-gray-900
mb-6">Ulasan</h2>

          {/* Create Review Form */}
          <form onSubmit={handleSubmitReview}
className="mb-8 p-4 bg-gray-50 rounded-xl">
            <h3 className="font-semibold text-gray-900
mb-3">Tulis Ulasan</h3>

            {/* Rating Input */}
            <div className="mb-4">
              <label className="block text-sm font-medium
text-gray-700 mb-2">
                Rating
              </label>
              <div className="flex gap-2">
                {[1, 2, 3, 4, 5].map((star) => (
                  <button
                    key={star}
                    type="button"
                    onClick={() => setRating(star)}
                    className={`text-2xl ${
                      star <= rating ? 'text-yellow-400'
: 'text-gray-300'
                    }`}
                  >
                    ★
                  </button>
                ))}
              </div>
            </div>

            {/* Comment Input */}
            <div className="mb-4">
              <label className="block text-sm font-medium
text-gray-700 mb-2">
                Komentar (Opsional)
              </label>
              <textarea
                value={comment}
```

```
                onChange={(e) =>
setComment(e.target.value)}
                maxLength={500}
                rows={3}
                className="w-full px-4 py-2 border
border-gray-300 rounded-lg focus:ring-2 focus:ring-
indigo-500 focus:border-indigo-500"
                placeholder="Bagikan pengalaman Anda
dengan resep ini..."
              />
              <p className="text-sm text-gray-500 mt-1">
                {comment.length}/500 karakter
              </p>
            </div>

            <button
              type="submit"
              disabled={createLoading}
              className="w-full md:w-auto px-6 py-2 bg-
indigo-600 text-white rounded-lg hover:bg-indigo-700
disabled:opacity-50 disabled:cursor-not-allowed"
            >
              {createLoading ? 'Mengirim...' : 'Kirim
Ulasan'}
            </button>
          </form>

          {/* Reviews List */}
          {reviewsLoading ? (
            <p className="text-gray-500">Memuat
ulasan...</p>
          ) : reviews.length === 0 ? (
            <p className="text-gray-500">Belum ada
ulasan</p>
          ) : (
            <div className="space-y-4">
              {reviews.map((review) => (
                <div key={review.id} className="border-b
border-gray-200 pb-4 last:border-b-0">
                  <div className="flex items-center
justify-between mb-2">
                    <div>
                      <p className="font-medium text-
gray-900">{review.user_identifier}</p>
                      <p className="text-sm text-gray-
500">{formatDate(review.created_at)}</p>
                    </div>
                    <div className="text-yellow-400">
                      {getStarRating(review.rating)}
                    </div>
                  </div>
                  {review.comment && (
                    <p className="text-gray-
700">{review.comment}</p>
                  )}
                </div>
```

```
              ))}
            </div>
          )}
        </div>
      </div>
    </div>
  );
}
```

7. Lanjut ke *folder* components, edit tiap *file* yang ada dengan kode ini

common/AdvancedFilter.jsx

```
// src/components/common/AdvancedFilter.jsx
import { useState } from 'react';
import { Filter, X, ChevronDown, SlidersHorizontal } from
'lucide-react';

/**
 * AdvancedFilter Component
 * Enhanced filter panel with multiple options (NO
category filter - already separate pages)
 */
export default function AdvancedFilter({ onFilterChange,
onSearchChange, initialFilters = {} }) {
  const [isOpen, setIsOpen] = useState(false);
  const [searchQuery, setSearchQuery] =
useState(initialFilters.search || '');
  const [filters, setFilters] = useState({
    difficulty: initialFilters.difficulty || '',
    sortBy: initialFilters.sortBy || 'created_at',
    order: initialFilters.order || 'desc',
    prepTimeMax: initialFilters.prepTimeMax || '',
  });

  // Debounce search
  const handleSearchChange = (value) => {
    setSearchQuery(value);
    if (onSearchChange) {
      // Simple debounce
      setTimeout(() => {
        onSearchChange(value);
      }, 300);
    }
  };

  const handleFilterChange = (key, value) => {
    const newFilters = { ...filters, [key]: value };
    setFilters(newFilters);

    // Call parent callback
    if (onFilterChange) {
      onFilterChange(newFilters);
    }
  };
```

```
  const handleReset = () => {
    const resetFilters = {
      difficulty: '',
      sortBy: 'created_at',
      order: 'desc',
      prepTimeMax: '',
    };
    setFilters(resetFilters);
    setSearchQuery('');

    if (onFilterChange) {
      onFilterChange(resetFilters);
    }
    if (onSearchChange) {
      onSearchChange('');
    }
  };

  const activeFilterCount =
Object.values(filters).filter(v =>
    v && v !== 'created_at' && v !== 'desc'
  ).length + (searchQuery ? 1 : 0);

  return (
    <div className="mb-6 space-y-4">
      {/* Search Bar - Always Visible */}
      <div>
        <input
          type="text"
          value={searchQuery}
          onChange={(e) =>
handleSearchChange(e.target.value)}
          placeholder="Cari resep..."
          className="w-full px-4 py-3 border border-
slate-300 rounded-xl focus:ring-2 focus:ring-blue-500
focus:border-blue-500 bg-white shadow-sm"
        />
      </div>

      {/* Filter & Sort Toggle Button */}
      <button
        onClick={() => setIsOpen(!isOpen)}
        className="flex items-center gap-2 px-4 py-2 bg-
white border border-slate-300 rounded-xl hover:bg-slate-
50 transition-all shadow-sm"
      >
        <SlidersHorizontal className="w-5 h-5 text-slate-
600" />
        <span className="font-medium text-slate-
700">Filter & Sort</span>
        {activeFilterCount > 0 && (
          <span className="px-2 py-0.5 bg-blue-500 text-
white text-xs font-bold rounded-full">
            {activeFilterCount}
          </span>
        )}
```

```
        <ChevronDown className={`w-4 h-4 text-slate-500
transition-transform ${isOpen ? 'rotate-180' : ''}`} />
      </button>

      {/* Filter Panel */}
      {isOpen && (
        <div className="bg-white/80 backdrop-blur-sm
border border-slate-200 rounded-2xl p-6 shadow-lg space-
y-4">
          <div className="flex items-center justify-
between mb-4">
            <h3 className="text-lg font-bold text-slate-
800 flex items-center gap-2">
              <Filter className="w-5 h-5" />
              Filter & Sorting
            </h3>
            <button
              onClick={handleReset}
              className="text-sm text-red-600 hover:text-
red-700 font-medium flex items-center gap-1"
            >
              <X className="w-4 h-4" />
              Reset
            </button>
          </div>

          <div className="grid md:grid-cols-2 gap-4">

            {/* Difficulty */}
            <div className="md:col-span-2">
              <label className="block text-sm font-medium
text-slate-700 mb-2">
                Tingkat Kesulitan
              </label>
              <select
                value={filters.difficulty}
                onChange={(e) =>
handleFilterChange('difficulty', e.target.value)}
                className="w-full px-4 py-2.5 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500 bg-white"
              >
                <option value="">Semua Tingkat</option>
                <option value="mudah">Mudah</option>
                <option value="sedang">Sedang</option>
                <option value="sulit">Sulit</option>
              </select>
            </div>

            {/* Prep Time */}
            <div>
              <label className="block text-sm font-medium
text-slate-700 mb-2">
                Waktu Persiapan (maks)
              </label>
              <select
```

```
                value={filters.prepTimeMax}
                onChange={(e) =>
handleFilterChange('prepTimeMax', e.target.value)}
                className="w-full px-4 py-2.5 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500 bg-white"
              >
                <option value="">Semua Durasi</option>
                <option value="15">≤ 15 menit</option>
                <option value="30">≤ 30 menit</option>
                <option value="60">≤ 60 menit</option>
                <option value="120">≤ 2 jam</option>
              </select>
            </div>

            {/* Sort By */}
            <div>
              <label className="block text-sm font-medium
text-slate-700 mb-2">
                Urutkan Berdasarkan
              </label>
              <select
                value={filters.sortBy}
                onChange={(e) =>
handleFilterChange('sortBy', e.target.value)}
                className="w-full px-4 py-2.5 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500 bg-white"
              >
                <option
value="created_at">Terbaru</option>
                <option value="name">Nama (A-Z)</option>
                <option value="prep_time">Waktu
Persiapan</option>
                <option value="cook_time">Waktu
Masak</option>
                <option
value="difficulty">Kesulitan</option>
              </select>
            </div>

            {/* Order */}
            <div>
              <label className="block text-sm font-medium
text-slate-700 mb-2">
                Urutan
              </label>
              <select
                value={filters.order}
                onChange={(e) =>
handleFilterChange('order', e.target.value)}
                className="w-full px-4 py-2.5 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500 bg-white"
              >
```

```
                <option value="desc">Descending (Besar →
Kecil)</option>
                <option value="asc">Ascending (Kecil →
Besar)</option>
              </select>
            </div>
          </div>

          {/* Active Filters Display */}
          {activeFilterCount > 0 && (
            <div className="pt-4 border-t border-slate-
200">
              <p className="text-sm text-slate-600 mb-
2">Filter Aktif:</p>
              <div className="flex flex-wrap gap-2">
                {searchQuery && (
                  <span className="inline-flex items-
center gap-1 px-3 py-1 bg-blue-100 text-blue-700 rounded-
full text-sm">
                    Search: "{searchQuery}"
                    <button onClick={() =>
handleSearchChange('')}>
                      <X className="w-3 h-3" />
                    </button>
                  </span>
                )}
                {filters.difficulty && (
                  <span className="inline-flex items-
center gap-1 px-3 py-1 bg-purple-100 text-purple-700
rounded-full text-sm">
                    {filters.difficulty}
                    <button onClick={() =>
handleFilterChange('difficulty', '')}>
                      <X className="w-3 h-3" />
                    </button>
                  </span>
                )}
                {filters.prepTimeMax && (
                  <span className="inline-flex items-
center gap-1 px-3 py-1 bg-orange-100 text-orange-700
rounded-full text-sm">
                    ≤ {filters.prepTimeMax} min
                    <button onClick={() =>
handleFilterChange('prepTimeMax', '')}>
                      <X className="w-3 h-3" />
                    </button>
                  </span>
                )}
              </div>
            </div>
          )}
        </div>
      )}
    </div>
  );
}
```

common/FavoriteButton.jsx

```
// src/components/common/FavoriteButton.jsx
import { useState, useEffect } from 'react';
import { Heart } from 'lucide-react';

/**
 * FavoriteButton Component
 * Toggles favorite status with localStorage support
 */
export default function FavoriteButton({ recipeId,
onToggle, showCount = false, initialCount = 0, size =
'md' }) {
  const [isFavorited, setIsFavorited] = useState(false);
  const [favoriteCount, setFavoriteCount] =
useState(initialCount);
  const [isAnimating, setIsAnimating] = useState(false);

  // Size variants
  const sizes = {
    sm: 'w-8 h-8',
    md: 'w-10 h-10',
    lg: 'w-12 h-12'
  };

  const iconSizes = {
    sm: 'w-4 h-4',
    md: 'w-5 h-5',
    lg: 'w-6 h-6'
  };

  // Check if recipe is favorited on mount
  useEffect(() => {
    const favorites =
JSON.parse(localStorage.getItem('favorites') || '[]');
    setIsFavorited(favorites.includes(recipeId));
  }, [recipeId]);

  const handleToggle = async (e) => {
    e.stopPropagation(); // Prevent card click

    setIsAnimating(true);
    setTimeout(() => setIsAnimating(false), 300);

    // Toggle in localStorage
    const favorites =
JSON.parse(localStorage.getItem('favorites') || '[]');
    const index = favorites.indexOf(recipeId);

    let newFavoritedState;
    if (index > -1) {
      // Remove from favorites
      favorites.splice(index, 1);
      newFavoritedState = false;
      setFavoriteCount(prev => Math.max(0, prev - 1));
```

```
    } else {
      // Add to favorites
      favorites.push(recipeId);
      newFavoritedState = true;
      setFavoriteCount(prev => prev + 1);
    }

    localStorage.setItem('favorites',
JSON.stringify(favorites));
    setIsFavorited(newFavoritedState);

    // Call parent callback if provided
    if (onToggle) {
      onToggle(recipeId, newFavoritedState);
    }
  };

  return (
    <button
      onClick={handleToggle}
      className={`
        ${sizes[size]} rounded-full flex items-center
justify-center gap-1.5
        transition-all duration-200
        ${isFavorited
          ? 'bg-red-500 hover:bg-red-600 text-white'
          : 'bg-white/90 hover:bg-white text-slate-700
hover:text-red-500'
        }
        backdrop-blur-sm shadow-md hover:shadow-lg
        ${isAnimating ? 'scale-125' : 'scale-100'}
        group
      `}
      title={isFavorited ? 'Remove from favorites' : 'Add
to favorites'}
    >
      <Heart
        className={`
          ${iconSizes[size]}
          transition-all duration-200
          ${isFavorited ? 'fill-current' : ''}
          ${isAnimating ? 'animate-pulse' : ''}
        `}
      />
      {showCount && favoriteCount > 0 && (
        <span className="text-xs font-semibold">
          {favoriteCount > 999 ? '999+' : favoriteCount}
        </span>
      )}
    </button>
  );
}
```

home/FeaturedMakananSection.jsx

```
// src/components/home/FeaturedMakananSection.jsx
import { Clock, Star, ChefHat } from 'lucide-react';
import { useState, useEffect, useRef } from 'react';

export default function FeaturedMakananSection({ recipes,
loading, error, onRecipeClick, onNavigate }) {
  const [visibleMakanan, setVisibleMakanan] =
useState(new Set());
  const makananRefs = useRef([]);

  useEffect(() => {
    const observerMakanan = new
IntersectionObserver((entries) => {
      entries.forEach((entry) => {
        if (entry.isIntersecting) {
          const index =
parseInt(entry.target.dataset.index);
          setTimeout(() => {
            setVisibleMakanan(prev => new
Set(prev).add(index));
          }, index * 200);
        }
      });
    }, { threshold: 0.1 });

    makananRefs.current.forEach((ref, index) => {
      if (ref) {
        ref.dataset.index = index;
        observerMakanan.observe(ref);
      }
    });

    return () => {
      observerMakanan.disconnect();
    };
  }, [recipes]);

  if (loading) {
    return (
      <section>
        <h2 className="text-xl md:text-3xl font-bold
text-slate-800 mb-6">Resep Makanan</h2>
        <div className="text-center py-12">
          <div className="animate-spin rounded-full h-12
w-12 border-b-2 border-indigo-600 mx-auto"></div>
          <p className="mt-4 text-gray-600">Memuat resep
makanan...</p>
        </div>
      </section>
    );
  }

  if (error) {
    return (
      <section>
```

```
        <h2 className="text-xl md:text-3xl font-bold
text-slate-800 mb-6">Resep Makanan</h2>
        <div className="text-center py-12">
          <p className="text-red-600">Error: {error}</p>
        </div>
      </section>
    );
  }

  if (!recipes || recipes.length === 0) {
    return (
      <section>
        <h2 className="text-xl md:text-3xl font-bold
text-slate-800 mb-6">Resep Makanan</h2>
        <div className="text-center py-12">
          <p className="text-gray-600">Belum ada resep
makanan</p>
        </div>
      </section>
    );
  }

  return (
    <section>
      <div className="flex items-center justify-between
mb-6 md:mb-8">
        <h2 className="text-xl md:text-3xl font-bold
text-slate-800">Resep Makanan</h2>
        <button
          onClick={() => onNavigate &&
onNavigate('makanan')}
          className="text-slate-500 hover:text-slate-600
font-medium text-xs md:text-sm transition-colors
duration-200 hover:underline"
        >
          Lihat Semua
        </button>
      </div>

      <div className="grid grid-cols-1 md:grid-cols-3
gap-4 md:gap-8">
        {recipes.map((recipe, index) => (
          <div
            key={recipe.id}
            ref={el => makananRefs.current[index] = el}
            className={`group transform transition-all
duration-700 ${
              visibleMakanan.has(index)
                ? 'translate-y-0 opacity-100'
                : 'translate-y-8 opacity-0'
            }`}
          >
            <div
              onClick={() => onRecipeClick &&
onRecipeClick(recipe.id)}
```

```
              className="relative bg-white/15 backdrop-
blur-xl border border-white/25 rounded-2xl md:rounded-3xl
overflow-hidden shadow-lg md:shadow-2xl shadow-blue-500/5
hover:shadow-blue-500/15 transition-all duration-500
cursor-pointer group-hover:scale-105 group-hover:bg-
white/20">

              <div className="absolute inset-0 bg-linear-
to-br from-white/5 via-transparent to-blue-500/5 opacity-
0 group-hover:opacity-100 transition-opacity duration-
500" />

              {/* Recipe Image*/}
              <div className="relative h-32 md:h-56
overflow-hidden">
                <img
                  src={recipe.image_url}
                  alt={recipe.name}
                  className="w-full h-full object-cover
group-hover:scale-110 transition-transform duration-500"
                />
                <div className="absolute inset-0 bg-
gradient-to-t from-black/20 via-transparent to-
transparent" />
              </div>

              <div className="relative z-10 p-4 md:p-8">
                <div className="flex items-center
justify-between mb-3 md:mb-4">
                  <span className="text-xs font-semibold
text-blue-700 bg-blue-100/90 px-2 md:px-3 py-1 md:py-1.5
rounded-full">
                    Makanan
                  </span>
                  {recipe.average_rating > 0 && (
                    <div className="flex items-center
space-x-1 bg-white/90 px-2 py-1 rounded-full">
                      <Star className="w-3 h-3 md:w-4
md:h-4 text-yellow-500 fill-current" />
                      <span className="text-xs md:text-sm
font-semibold text-slate-700">

{recipe.average_rating.toFixed(1)}
                      </span>
                    </div>
                  )}
                </div>

                <h3 className="font-bold text-slate-800
mb-3 md:mb-4 text-base md:text-xl group-hover:text-blue-
600 transition-colors duration-200 line-clamp-2">
                  {recipe.name}
                </h3>

                <div className="flex items-center
justify-between text-xs md:text-sm text-slate-600">
```

```
                  <div className="flex items-center
space-x-1 md:space-x-2 bg-white/70 px-2 md:px-3 py-1
md:py-2 rounded-full">
                    <Clock className="w-3 h-3 md:w-4
md:h-4" />
                    <span className="font-
medium">{recipe.prep_time || 15} menit</span>
                  </div>
                  <div className="flex items-center
space-x-1 md:space-x-2 bg-white/70 px-2 md:px-3 py-1
md:py-2 rounded-full">
                    <ChefHat className="w-3 h-3 md:w-4
md:h-4" />
                    <span className="font-
medium">{recipe.difficulty || 'mudah'}</span>
                  </div>
                </div>
              </div>
            </div>
          </div>
        ))}
      </div>
    </section>
  );
}
```

home/FeaturedMinumanSection.jsx

```
// src/components/home/FeaturedMinumanSection.jsx
import { Clock, Star, ChefHat, Coffee } from 'lucide-
react';
import { useState, useEffect, useRef } from 'react';

export default function FeaturedMinumanSection({ recipes,
loading, error, onRecipeClick, onNavigate }) {
  const [visibleMinuman, setVisibleMinuman] = useState(new
Set());
  const minumanRefs = useRef([]);

  useEffect(() => {
    const observerMinuman = new
IntersectionObserver((entries) => {
      entries.forEach((entry) => {
        if (entry.isIntersecting) {
          const index =
parseInt(entry.target.dataset.index);
          setTimeout(() => {
            setVisibleMinuman(prev => new
Set(prev).add(index));
          }, index * 250);
        }
      });
    }, { threshold: 0.1 });

    minumanRefs.current.forEach((ref, index) => {
```

```
      if (ref) {
        ref.dataset.index = index;
        observerMinuman.observe(ref);
      }
    });

    return () => {
      observerMinuman.disconnect();
    };
  }, [recipes]);

  if (loading) {
    return (
      <section>
        <h2 className="text-xl md:text-3xl font-bold text-
slate-800 mb-6">Resep Minuman</h2>
        <div className="text-center py-12">
          <div className="animate-spin rounded-full h-12
w-12 border-b-2 border-indigo-600 mx-auto"></div>
          <p className="mt-4 text-gray-600">Memuat resep
minuman...</p>
        </div>
      </section>
    );
  }

  if (error) {
    return (
      <section>
        <h2 className="text-xl md:text-3xl font-bold text-
slate-800 mb-6">Resep Minuman</h2>
        <div className="text-center py-12">
          <p className="text-red-600">Error: {error}</p>
        </div>
      </section>
    );
  }

  if (!recipes || recipes.length === 0) {
    return (
      <section>
        <h2 className="text-xl md:text-3xl font-bold text-
slate-800 mb-6">Resep Minuman</h2>
        <div className="text-center py-12">
          <p className="text-gray-600">Belum ada resep
minuman</p>
        </div>
      </section>
    );
  }

  return (
    <section>
      <div className="flex items-center justify-between
mb-6 md:mb-8">
```

```
        <h2 className="text-xl md:text-3xl font-bold text-
slate-800">Resep Minuman</h2>
        <button
          onClick={() => onNavigate &&
onNavigate('minuman')}
          className="text-slate-500 hover:text-slate-600
font-medium text-xs md:text-sm transition-colors duration-
200 hover:underline"
        >
          Lihat Semua
        </button>
      </div>

      <div className="grid grid-cols-1 md:grid-cols-2 gap-
4 md:gap-8">
        {recipes.map((recipe, index) => (
          <div
            key={recipe.id}
            ref={el => minumanRefs.current[index] = el}
            className={`group transform transition-all
duration-700 ${
              visibleMinuman.has(index)
                ? 'translate-y-0 opacity-100'
                : 'translate-y-8 opacity-0'
            }`}
          >
            <div
              onClick={() => onRecipeClick &&
onRecipeClick(recipe.id)}
              className="relative bg-white/15 backdrop-
blur-xl border border-white/25 rounded-2xl md:rounded-3xl
overflow-hidden shadow-lg md:shadow-2xl shadow-indigo-
500/5 hover:shadow-indigo-500/15 transition-all duration-
500 cursor-pointer group-hover:scale-105 group-hover:bg-
white/20">

              <div className="absolute inset-0 bg-linear-
to-br from-white/5 via-transparent to-indigo-500/5
opacity-0 group-hover:opacity-100 transition-opacity
duration-500" />

              <div className="flex">
                {/* Recipe Image */}
                <div className="h-29 w-28 md:h-48 md:w-48
flex-shrink-0 overflow-hidden">
                  <img
                    src={recipe.image_url}
                    alt={recipe.name}
                    className="w-full h-full object-cover
group-hover:scale-110 transition-transform duration-500"
                  />
                </div>

                <div className="relative z-10 p-4 md:p-8
flex-1 flex flex-col justify-center">
```

```
                  <div className="flex items-center
justify-between mb-2 md:mb-4">
                    <span className="text-xs font-semibold
text-indigo-700 bg-indigo-100/90 px-2 md:px-3 py-1 md:py-
1.5 rounded-full">
                      Minuman
                    </span>
                    {recipe.average_rating > 0 && (
                      <div className="flex items-center
space-x-1 bg-white/90 px-2 py-1 rounded-full">
                        <Star className="w-3 h-3 md:w-4
md:h-4 text-yellow-500 fill-current" />
                        <span className="text-xs md:text-
sm font-semibold text-slate-700">

{recipe.average_rating.toFixed(1)}
                        </span>
                      </div>
                    )}
                  </div>

                  <h3 className="font-bold text-slate-800
mb-2 md:mb-4 text-sm md:text-xl group-hover:text-indigo-
600 transition-colors duration-200 line-clamp-2">
                    {recipe.name}
                  </h3>

                  <div className="flex items-center
justify-between text-xs md:text-sm text-slate-600">
                    <div className="flex items-center
space-x-1 md:space-x-2 bg-white/70 px-2 md:px-3 py-1
md:py-2 rounded-full">
                      <Clock className="w-3 h-3 md:w-4
md:h-4" />
                      <span className="font-
medium">{recipe.prep_time || 10} menit</span>
                    </div>
                    <div className="flex items-center
space-x-1 md:space-x-2 bg-white/70 px-2 md:px-3 py-1
md:py-2 rounded-full">
                      <Coffee className="w-3 h-3 md:w-4
md:h-4" />
                      <span className="font-
medium">{recipe.difficulty || 'mudah'}</span>
                    </div>
                  </div>
                </div>
              </div>
            </div>
          </div>
        ))}
      </div>
    </section>
  );
}
```

makanan/RecipeGrid.jsx

```
// src/components/makanan/RecipeGrid.jsx
import { Clock, Star, ChefHat } from 'lucide-react';
import { useState, useEffect, useRef } from 'react';
import FavoriteButton from '../common/FavoriteButton';

export default function RecipeGrid({ recipes,
onRecipeClick }) {
  const [visibleCards, setVisibleCards] = useState(new
Set());
  const cardRefs = useRef([]);

  useEffect(() => {

    cardRefs.current = cardRefs.current.slice(0,
recipes.length);

    const observer = new IntersectionObserver((entries)
=> {
      entries.forEach((entry) => {
        if (entry.isIntersecting) {
          const index =
parseInt(entry.target.dataset.index);

          setTimeout(() => {
            setVisibleCards(prev => new
Set(prev).add(index));
          }, (index % 3) * 150);
        }
      });
    }, { threshold: 0.1 });

    cardRefs.current.forEach((ref, index) => {
      if (ref) {
        ref.dataset.index = index;
        observer.observe(ref);
      }
    });

    return () => {
      observer.disconnect();
    };
  }, [recipes]);

  return (
    <section>
       <h1 className="text-3xl md:text-5xl font-bold
text-slate-800 text-center mb-4">
        Jelajahi Resep Makanan
      </h1>
      <p className="text-center text-slate-500 max-w-2xl
mx-auto mb-8">
        Temukan inspirasi masakan Nusantara favoritmu.
Dari hidangan utama hingga camilan, semua ada di sini.
      </p>
```

```
      <div className="grid grid-cols-1 md:grid-cols-2
lg:grid-cols-3 gap-4 md:gap-8">
        {recipes.map((recipe, index) => (
          <div
            key={recipe.id}
            ref={el => cardRefs.current[index] = el}
            className={`group transform transition-all
duration-700 ${
              visibleCards.has(index)
                ? 'translate-y-0 opacity-100'
                : 'translate-y-8 opacity-0'
            }`}
          >

            <div
              onClick={() => onRecipeClick &&
onRecipeClick(recipe.id)}
              className="relative bg-white/15 backdrop-
blur-xl border border-white/25 rounded-2xl md:rounded-3xl
overflow-hidden shadow-lg md:shadow-2xl shadow-blue-500/5
hover:shadow-blue-500/15 transition-all duration-500
cursor-pointer group-hover:scale-105 group-hover:bg-
white/20">
              <div className="absolute inset-0 bg-
gradient-to-br from-white/5 via-transparent to-blue-500/5
opacity-0 group-hover:opacity-100 transition-opacity
duration-500" />
              <div className="relative h-32 md:h-56
overflow-hidden">
                <img
                  src={recipe.image_url}
                  alt={recipe.name}
                  className="w-full h-full object-cover
group-hover:scale-110 transition-transform duration-500"
                />
                <div className="absolute inset-0 bg-
gradient-to-t from-black/20 via-transparent to-
transparent" />

                {/* Favorite Button */}
                <div className="absolute top-3 right-3 z-
10">
                  <FavoriteButton recipeId={recipe.id}
size="sm" />
                </div>
              </div>
              <div className="relative z-10 p-4 md:p-8">
                <div className="flex items-center
justify-between mb-3 md:mb-4">
                  <span className="text-xs font-semibold
text-blue-700 bg-blue-100/90 px-2 md:px-3 py-1 md:py-1.5
rounded-full">
                    Makanan
                  </span>
                  {recipe.average_rating > 0 && (
```

```
                    <div className="flex items-center
space-x-1 bg-white/90 px-2 py-1 rounded-full">
                      <Star className="w-3 h-3 md:w-4
md:h-4 text-yellow-500 fill-current" />
                      <span className="text-xs md:text-sm
font-semibold text-slate-
700">{recipe.average_rating.toFixed(1)}</span>
                    </div>
                  )}
                </div>
                <h3 className="font-bold text-slate-800
mb-3 md:mb-4 text-base md:text-xl group-hover:text-blue-
600 transition-colors duration-200 line-clamp-2">
                  {recipe.name}
                </h3>
                <div className="flex items-center
justify-between text-xs md:text-sm text-slate-600">
                  <div className="flex items-center
space-x-1 md:space-x-2 bg-white/70 px-2 md:px-3 py-1
md:py-2 rounded-full">
                    <Clock className="w-3 h-3 md:w-4
md:h-4" />
                    <span className="font-
medium">{recipe.prep_time}</span>
                  </div>
                  <div className="flex items-center
space-x-1 md:space-x-2 bg-white/70 px-2 md:px-3 py-1
md:py-2 rounded-full">
                    <ChefHat className="w-3 h-3 md:w-4
md:h-4" />
                    <span className="font-
medium">{recipe.difficulty}</span>
                  </div>
                </div>
              </div>
            </div>
          </div>
        ))}
      </div>
      {recipes.length === 0 && (
        <div className="text-center py-16">
            <p className="text-slate-500">Resep tidak
ditemukan. Coba kata kunci lain.</p>
        </div>
      )}
    </section>
  );
}
```

minuman/RecipeGrid.jsx

```
// src/components/minuman/RecipeGrid.jsx
import { Clock, Star, Coffee, ChefHat } from 'lucide-
react';
import { useState, useEffect, useRef } from 'react';
```

```
import FavoriteButton from '../common/FavoriteButton';

export default function RecipeGrid({ recipes,
onRecipeClick }) {
  const [visibleCards, setVisibleCards] = useState(new
Set());
  const cardRefs = useRef([]);

  useEffect(() => {
    cardRefs.current = cardRefs.current.slice(0,
recipes.length);

    const observer = new IntersectionObserver((entries)
=> {
      entries.forEach((entry) => {
        if (entry.isIntersecting) {
          const index =
parseInt(entry.target.dataset.index);
          setTimeout(() => {
            setVisibleCards(prev => new
Set(prev).add(index));
          }, (index % 3) * 150);
        }
      });
    }, { threshold: 0.1 });

    cardRefs.current.forEach((ref, index) => {
      if (ref) {
        ref.dataset.index = index;
        observer.observe(ref);
      }
    });

    return () => {
      observer.disconnect();
    };
  }, [recipes]);

  return (
    <section>
      <h1 className="text-3xl md:text-5xl font-bold text-
slate-800 text-center mb-4">
        Jelajahi Resep Minuman
      </h1>
      <p className="text-center text-slate-500 max-w-2xl
mx-auto mb-8">
        Temukan minuman segar, hangat, dan kekinian.
Mulai dari kopi hingga jus buah, semua ada di sini.
      </p>
      <div className="grid grid-cols-1 md:grid-cols-2
lg:grid-cols-3 gap-4 md:gap-8">
        {recipes.map((recipe, index) => (
          <div
            key={recipe.id}
            ref={el => cardRefs.current[index] = el}
```

```
            className={`group transform transition-all
duration-700 ${
              visibleCards.has(index)
                ? 'translate-y-0 opacity-100'
                : 'translate-y-8 opacity-0'
            }`}
          >
            {/* Card structure is consistent, only the
tag is changed */}
            <div
              onClick={() => onRecipeClick &&
onRecipeClick(recipe.id)}
              className="relative bg-white/15 backdrop-
blur-xl border border-white/25 rounded-2xl md:rounded-3xl
overflow-hidden shadow-lg md:shadow-2xl shadow-green-
500/5 hover:shadow-green-500/15 transition-all duration-
500 cursor-pointer group-hover:scale-105 group-hover:bg-
white/20">
              <div className="absolute inset-0 bg-
gradient-to-br from-white/5 via-transparent to-green-
500/5 opacity-0 group-hover:opacity-100 transition-
opacity duration-500" />
              <div className="relative h-32 md:h-56
overflow-hidden">
                <img
                  src={recipe.image_url}
                  alt={recipe.name}
                  className="w-full h-full object-cover
group-hover:scale-110 transition-transform duration-500"
                />
                <div className="absolute inset-0 bg-
gradient-to-t from-black/20 via-transparent to-
transparent" />

                {/* Favorite Button */}
                <div className="absolute top-3 right-3 z-
10">
                  <FavoriteButton recipeId={recipe.id}
size="sm" />
                </div>
              </div>
              <div className="relative z-10 p-4 md:p-8">
                <div className="flex items-center
justify-between mb-3 md:mb-4">
                  {/* Changed tag color from blue to
green */}
                  <span className="text-xs font-semibold
text-green-700 bg-green-100/90 px-2 md:px-3 py-1 md:py-
1.5 rounded-full">
                    Minuman
                  </span>
                  {recipe.average_rating > 0 && (
                    <div className="flex items-center
space-x-1 bg-white/90 px-2 py-1 rounded-full">
                      <Star className="w-3 h-3 md:w-4
md:h-4 text-yellow-500 fill-current" />
```

```
                      <span className="text-xs md:text-sm
font-semibold text-slate-
700">{recipe.average_rating.toFixed(1)}</span>
                    </div>
                  )}
                </div>
                <h3 className="font-bold text-slate-800
mb-3 md:mb-4 text-base md:text-xl group-hover:text-green-
600 transition-colors duration-200 line-clamp-2">
                  {recipe.name}
                </h3>
                <div className="flex items-center
justify-between text-xs md:text-sm text-slate-600">
                  <div className="flex items-center
space-x-1 md:space-x-2 bg-white/70 px-2 md:px-3 py-1
md:py-2 rounded-full">
                    <Clock className="w-3 h-3 md:w-4
md:h-4" />
                    <span className="font-
medium">{recipe.prep_time}</span>
                  </div>
                  <div className="flex items-center
space-x-1 md:space-x-2 bg-white/70 px-2 md:px-3 py-1
md:py-2 rounded-full">
                    <ChefHat className="w-3 h-3 md:w-4
md:h-4" />
                    <span className="font-
medium">{recipe.difficulty}</span>
                  </div>
                </div>
              </div>
            </div>
          </div>
        ))}
      </div>
       {recipes.length === 0 && (
         <div className="text-center py-16">
             <p className="text-slate-500">Minuman tidak
ditemukan. Coba kata kunci lain.</p>
        </div>
      )}
    </section>
  );
}
```

modals/ConfirmModal.jsx

```
// src/components/modals/ConfirmModal.jsx
import { AlertTriangle, X } from 'lucide-react';

export default function ConfirmModal({
  isOpen,
  onClose,
  onConfirm,
  title = 'Konfirmasi',
```

```
  message = 'Apakah Anda yakin?',
  confirmText = 'Ya, Lanjutkan',
  cancelText = 'Batal',
  variant = 'danger', // 'danger' | 'warning' | 'info'
  isLoading = false,
}) {
  if (!isOpen) return null;

  const variantStyles = {
    danger: {
      icon: 'bg-red-100 text-red-600',
      button: 'bg-red-600 hover:bg-red-700',
    },
    warning: {
      icon: 'bg-yellow-100 text-yellow-600',
      button: 'bg-yellow-600 hover:bg-yellow-700',
    },
    info: {
      icon: 'bg-blue-100 text-blue-600',
      button: 'bg-blue-600 hover:bg-blue-700',
    },
  };

  const currentVariant = variantStyles[variant] ||
variantStyles.danger;

  return (
    <div className="fixed inset-0 z-50 flex items-center
justify-center p-4 bg-black/50 backdrop-blur-sm animate-
fadeIn">
      <div className="bg-white rounded-2xl shadow-2xl
max-w-md w-full animate-slideUp">
        {/* Header */}
        <div className="flex items-center justify-between
p-6 border-b border-slate-200">
          <div className="flex items-center gap-3">
            <div className={`w-10 h-10 rounded-full flex
items-center justify-center ${currentVariant.icon}`}>
              <AlertTriangle className="w-5 h-5" />
            </div>
            <h3 className="text-xl font-bold text-slate-
800">{title}</h3>
          </div>
          <button
            onClick={onClose}
            disabled={isLoading}
            className="text-slate-400 hover:text-slate-
600 transition-colors disabled:opacity-50"
          >
            <X className="w-6 h-6" />
          </button>
        </div>

        {/* Content */}
        <div className="p-6">
```

```
          <p className="text-slate-600 leading-
relaxed">{message}</p>
        </div>

        {/* Actions */}
        <div className="flex gap-3 p-6 bg-slate-50
rounded-b-2xl">
          <button
            onClick={onClose}
            disabled={isLoading}
            className="flex-1 px-4 py-3 border border-
slate-300 text-slate-700 rounded-xl hover:bg-white
transition-colors font-medium disabled:opacity-50
disabled:cursor-not-allowed"
          >
            {cancelText}
          </button>
          <button
            onClick={onConfirm}
            disabled={isLoading}
            className={`flex-1 px-4 py-3 text-white
rounded-xl transition-colors font-medium
disabled:opacity-50 disabled:cursor-not-allowed
${currentVariant.button}`}
          >
            {isLoading ? 'Memproses...' : confirmText}
          </button>
        </div>
      </div>
    </div>
  );
}
```

navbar/DesktopNavbar.jsx

```
// src/components/DesktopNavbar.jsx
import { Plus } from 'lucide-react';
import logoUrl from '../../assets/LOGORN.png';

export default function DesktopNavbar({ currentPage,
onNavigate, onCreateRecipe }) {
  const navItems = [
    { id: 'home', label: 'Beranda' },
    { id: 'makanan', label: 'Makanan' },
    { id: 'minuman', label: 'Minuman' },
    { id: 'profile', label: 'Profile' }
  ];

  return (
    <nav className="hidden md:block shadow-lg border-b
border-blue-100 sticky top-0 z-50 backdrop-blur-sm bg-
white/95">
      <div className="max-w-7xl mx-auto px-4 sm:px-6
lg:px-8">
```

```
        <div className="flex justify-between items-center
h-20">

          {/* Logo */}
          <div className="flex items-center space-x-4">
            <div className="relative group">
              <img
                src={logoUrl}
                alt="Resep Nusantara Logo"
                className="w-12 h-12 object-contain
filter drop-shadow-md transform transition-transform
duration-300 group-hover:scale-110"
              />
              {/* Decorative particles */}
              <div className="absolute -top-0.5 -right-
0.5 w-1.5 h-1.5 bg-blue-400 rounded-full animate-ping
opacity-60" />
              <div className="absolute -bottom-0.5 -left-
0.5 w-1 h-1 bg-blue-300 rounded-full animate-ping
opacity-50" style={{ animationDelay: '300ms' }} />
            </div>
            <div>
              <h1 className="text-2xl font-bold bg-
gradient-to-r from-slate-800 via-slate-700 to-slate-800
bg-clip-text text-transparent">
                Resep
              </h1>
              <h2 className="text-base font-semibold bg-
gradient-to-r from-blue-600 via-blue-500 to-blue-400 bg-
clip-text text-transparent -mt-1">
                Nusantara
              </h2>
            </div>
          </div>

          {/* Navigation Links */}
          <div className="flex items-center space-x-10">
            {navItems.map((item) => (
              <button
                key={item.id}
                onClick={() => onNavigate(item.id)}
                className={`px-4 py-3 text-base font-
medium transition-all duration-200 border-b-2 ${
                  currentPage === item.id
                    ? 'text-blue-600 border-blue-500'
                    : 'text-slate-600 border-transparent
hover:text-blue-500 hover:border-blue-300'
                }`}
              >
                {item.label}
              </button>
            ))}

            {/* Buat Resep Button */}
            <button
              onClick={onCreateRecipe}
```

```
              className="flex items-center gap-2 px-5 py-
2.5 bg-gradient-to-r from-blue-600 to-indigo-600 text-
white rounded-xl hover:from-blue-700 hover:to-indigo-700
transition-all duration-200 shadow-md hover:shadow-lg
transform hover:scale-105 font-medium"
            >
              <Plus className="w-5 h-5" />
              <span>Buat Resep</span>
            </button>
          </div>

        </div>
      </div>
    </nav>
  );
}
```

navbar/MobileNavbar.jsx

```
// src/components/MobileNavbar.jsx
import { Home, ChefHat, Coffee, User, Plus } from
'lucide-react';

export default function MobileNavbar({ currentPage,
onNavigate, onCreateRecipe }) {
  const navItems = [
    { id: 'home', label: 'Beranda', icon: Home },
    { id: 'makanan', label: 'Makanan', icon: ChefHat },
    { id: 'minuman', label: 'Minuman', icon: Coffee },
    { id: 'profile', label: 'Profile', icon: User }
  ];

  return (
    <>
      {/* Floating Create Button */}
      {onCreateRecipe && (
        <button
          onClick={onCreateRecipe}
          className="md:hidden fixed bottom-20 right-4 z-
50 w-14 h-14 bg-blue-600 text-white rounded-full shadow-
lg hover:bg-blue-700 transition-all hover:scale-110 flex
items-center justify-center"
        >
          <Plus className="w-6 h-6" />
        </button>
      )}

      {/* Bottom Navigation */}
      <nav className="md:hidden fixed bottom-0 left-0
right-0 bg-white border-t border-gray-100 px-4 py-1 z-
50">
      <div className="flex items-center justify-around
max-w-sm mx-auto">
        {navItems.map((item) => {
```

```
          const IconComponent = item.icon;
          const isActive = currentPage === item.id;

          return (
            <button
              key={item.id}
              onClick={() => onNavigate(item.id)}
              className={`flex flex-col items-center py-2
px-3 transition-colors duration-200 ${
                isActive ? 'text-blue-600' : 'text-gray-
400'
              }`}
            >
              <IconComponent
                size={20}
                className="mb-1"
                strokeWidth={isActive ? 2 : 1.5}
              />
              <span className="text-xs font-medium">
                {item.label}
              </span>
            </button>
          );
        })}
      </div>
    </nav>
    </>
  );
}
```

recipe/RecipeDetail.jsx

```
// src/components/recipe/RecipeDetail.jsx
import { useState } from 'react';
import { useRecipe } from '../../hooks/useRecipes';
import { useReviews, useCreateReview } from
'../../hooks/useReviews';
import { useIsFavorited } from
'../../hooks/useFavorites';
import { getUserIdentifier } from
'../../hooks/useFavorites';
import { formatDate, getDifficultyColor, getStarRating }
from '../../utils/helpers';
import { ArrowLeft, Heart, Clock, Users, ChefHat, Star,
Send, Edit, Trash2 } from 'lucide-react';
import recipeService from '../../services/recipeService';
import ConfirmModal from '../modals/ConfirmModal';
import FavoriteButton from '../common/FavoriteButton';
import userService from '../../services/userService';

export default function RecipeDetail({ recipeId, onBack,
onEdit, category = 'makanan' }) {
  const { recipe, loading: recipeLoading, error:
recipeError } = useRecipe(recipeId);
```

```
  const { reviews, loading: reviewsLoading, refetch:
refetchReviews } = useReviews(recipeId);
  const { createReview, loading: createLoading } =
useCreateReview();
  const { isFavorited, loading: favLoading,
toggleFavorite } = useIsFavorited(recipeId);

  const [rating, setRating] = useState(5);
  const [comment, setComment] = useState('');
  const [showReviewForm, setShowReviewForm] =
useState(false);
  const [showDeleteModal, setShowDeleteModal] =
useState(false);
  const [deleting, setDeleting] = useState(false);

  const categoryColors = {
    makanan: {
      primary: 'blue',
      gradient: 'from-blue-50 via-white to-indigo-50',
      text: 'text-blue-700',
      bg: 'bg-blue-100',
      border: 'border-blue-400',
      hover: 'hover:bg-blue-50',
      ring: 'ring-blue-500'
    },
    minuman: {
      primary: 'green',
      gradient: 'from-green-50 via-white to-cyan-50',
      text: 'text-green-700',
      bg: 'bg-green-100',
      border: 'border-green-400',
      hover: 'hover:bg-green-50',
      ring: 'ring-green-500'
    }
  };

  const colors = categoryColors[category] ||
categoryColors.makanan;

  const handleSubmitReview = async (e) => {
    e.preventDefault();

    // Get username from user profile
    const userProfile = userService.getUserProfile();

    const reviewData = {
      user_identifier: userProfile.username ||
getUserIdentifier(),
      rating,
      comment: comment.trim(),
    };

    const success = await createReview(recipeId,
reviewData);

    if (success) {
```

```
      setComment('');
      setRating(5);
      setShowReviewForm(false);
      refetchReviews();
    }
  };

  const handleToggleFavorite = async () => {
    await toggleFavorite();
  };

  const handleDeleteRecipe = async () => {
    try {
      setDeleting(true);
      const result = await
recipeService.deleteRecipe(recipeId);

      if (result.success) {
        alert('Resep berhasil dihapus!');
        setShowDeleteModal(false);
        if (onBack) {
          onBack();
        }
      } else {
        throw new Error(result.message || 'Gagal
menghapus resep');
      }
    } catch (err) {
      console.error('Delete recipe error:', err);
      alert(err.message || 'Terjadi kesalahan saat
menghapus resep');
    } finally {
      setDeleting(false);
    }
  };

  if (recipeLoading) {
    return (
      <div className="min-h-screen flex items-center
justify-center">
        <div className="text-center">
          <div className={`animate-spin rounded-full h-12
w-12 border-b-2 border-${colors.primary}-600 mx-
auto`}></div>
          <p className="mt-4 text-slate-600">Memuat
resep...</p>
        </div>
      </div>
    );
  }

  if (recipeError) {
    return (
      <div className="min-h-screen flex items-center
justify-center p-4">
        <div className="text-center max-w-md">
```

```
          <div className="bg-red-50 border border-red-200
rounded-2xl p-6">
            <p className="text-red-600 font-semibold mb-
2">Terjadi Kesalahan</p>
            <p className="text-red-500 mb-
4">{recipeError}</p>
            <button
              onClick={onBack}
              className="px-4 py-2 bg-red-600 text-white
rounded-lg hover:bg-red-700 transition-colors"
            >
              Kembali
            </button>
          </div>
        </div>
      </div>
    );
  }

  if (!recipe) {
    return (
      <div className="min-h-screen flex items-center
justify-center p-4">
        <div className="text-center">
          <p className="text-slate-600">Resep tidak
ditemukan</p>
          <button
            onClick={onBack}
            className={`mt-4 px-4 py-2 bg-
${colors.primary}-600 text-white rounded-lg hover:bg-
${colors.primary}-700 transition-colors`}
          >
            Kembali
          </button>
        </div>
      </div>
    );
  }

  return (
    <div className={`min-h-screen bg-gradient-to-br
${colors.gradient} pb-20 md:pb-8`}>
      {/* Delete Confirmation Modal */}
      <ConfirmModal
        isOpen={showDeleteModal}
        onClose={() => setShowDeleteModal(false)}
        onConfirm={handleDeleteRecipe}
        title="Hapus Resep"
        message={`Apakah Anda yakin ingin menghapus resep
"${recipe?.name}"? Tindakan ini tidak dapat dibatalkan.`}
        confirmText="Ya, Hapus"
        cancelText="Batal"
        variant="danger"
        isLoading={deleting}
      />
```

```
      {/* Header */}
      <div className="sticky top-0 z-10 bg-white/80
backdrop-blur-md border-b border-slate-200 shadow-sm">
        <div className="max-w-4xl mx-auto px-4 py-4 flex
items-center justify-between">
          <button
            onClick={onBack}
            className="flex items-center gap-2 text-
slate-700 hover:text-slate-900 transition-colors"
          >
            <ArrowLeft className="w-5 h-5" />
            <span className="font-medium">Kembali</span>
          </button>

          {/* Action Buttons */}
          {onEdit && (
            <div className="flex gap-2">
              <button
                onClick={() => {
                  console.log('🖱️ Edit button clicked in
RecipeDetail');
                  console.log('📝 Recipe ID:',
recipeId);
                  console.log('🔧 onEdit function:',
onEdit);
                  onEdit(recipeId);
                }}
                className="flex items-center gap-2 px-4
py-2 bg-blue-600 text-white rounded-lg hover:bg-blue-700
transition-colors"
              >
                <Edit className="w-4 h-4" />
                <span className="hidden
md:inline">Edit</span>
              </button>
              <button
                onClick={() => setShowDeleteModal(true)}
                className="flex items-center gap-2 px-4
py-2 bg-red-600 text-white rounded-lg hover:bg-red-700
transition-colors"
              >
                <Trash2 className="w-4 h-4" />
                <span className="hidden
md:inline">Hapus</span>
              </button>
            </div>
          )}
        </div>
      </div>

      <main className="max-w-5xl mx-auto px-4 py-8">
        {/* Recipe Header */}
        <div className="bg-white/60 backdrop-blur-sm
rounded-3xl overflow-hidden shadow-xl border border-
white/40 mb-8">
          {/* Hero Image */}
```

```
          <div className="relative h-64 md:h-96 overflow-
hidden">
            <img
              src={recipe.image_url}
              alt={recipe.name}
              className="w-full h-full object-cover"
            />
            <div className="absolute inset-0 bg-gradient-
to-t from-black/60 via-transparent to-transparent" />

            {/* Favorite Button - Use component */}
            <div className="absolute top-4 right-4 z-10">
              <FavoriteButton recipeId={recipeId}
size="lg" />
            </div>

            {/* Category Badge */}
            <div className="absolute bottom-4 left-4">
              <span className={`${colors.text}
${colors.bg} px-4 py-2 rounded-full text-sm font-
semibold`}>
                {category === 'makanan' ? 'Makanan' :
'Minuman'}
              </span>
            </div>
          </div>

          {/* Recipe Info */}
          <div className="p-6 md:p-8">
            <h1 className="text-3xl md:text-4xl font-bold
text-slate-800 mb-4">
              {recipe.name}
            </h1>

            {recipe.description && (
              <p className="text-slate-600 text-lg mb-6
leading-relaxed">
                {recipe.description}
              </p>
            )}

            {/* Stats Grid */}
            <div className="grid grid-cols-2 md:grid-
cols-4 gap-4">
              <div className="bg-white/70 backdrop-blur
p-4 rounded-xl border border-white/60 text-center">
                <Clock className={`w-6 h-6 mx-auto mb-2
text-${colors.primary}-600`} />
                <p className="text-xs text-slate-500 mb-
1">Persiapan</p>
                <p className="font-semibold text-slate-
700">{recipe.prep_time}</p>
              </div>
              <div className="bg-white/70 backdrop-blur
p-4 rounded-xl border border-white/60 text-center">
```

```
                <Clock className={`w-6 h-6 mx-auto mb-2
text-${colors.primary}-600`} />
                <p className="text-xs text-slate-500 mb-
1">Memasak</p>
                <p className="font-semibold text-slate-
700">{recipe.cook_time} menit</p>
              </div>
              <div className="bg-white/70 backdrop-blur
p-4 rounded-xl border border-white/60 text-center">
                <Users className={`w-6 h-6 mx-auto mb-2
text-${colors.primary}-600`} />
                <p className="text-xs text-slate-500 mb-
1">Porsi</p>
                <p className="font-semibold text-slate-
700">{recipe.servings} orang</p>
              </div>
              <div className="bg-white/70 backdrop-blur
p-4 rounded-xl border border-white/60 text-center">
                <ChefHat className={`w-6 h-6 mx-auto mb-2
text-${colors.primary}-600`} />
                <p className="text-xs text-slate-500 mb-
1">Kesulitan</p>
                <p className={`font-semibold capitalize
${getDifficultyColor(recipe.difficulty)}`}>
                  {recipe.difficulty}
                </p>
              </div>
            </div>

            {/* Rating */}
            {recipe.average_rating > 0 && (
              <div className="mt-6 flex items-center gap-
4 bg-amber-50 border border-amber-200 rounded-xl p-4">
                <div className="flex items-center gap-1">
                  {[1, 2, 3, 4, 5].map((star) => (
                    <Star
                      key={star}
                      className={`w-5 h-5 ${
                        star <=
Math.round(recipe.average_rating)
                          ? 'text-amber-500 fill-current'
                          : 'text-slate-300'
                      }`}
                    />
                  ))}
                </div>
                <div>
                  <p className="text-lg font-bold text-
slate-800">
                    {recipe.average_rating.toFixed(1)}
                  </p>
                  <p className="text-xs text-slate-500">
                    {recipe.review_count} ulasan
                  </p>
                </div>
              </div>
```

```
            )}
          </div>
        </div>

        {/* Ingredients & Steps */}
        <div className="grid md:grid-cols-2 gap-8 mb-8">
          {/* Ingredients */}
          <div className="bg-white/60 backdrop-blur-sm
rounded-3xl p-6 md:p-8 shadow-xl border border-white/40">
            <h2 className="text-2xl font-bold text-slate-
800 mb-6 flex items-center gap-3">
              <div className={`w-10 h-10 rounded-full bg-
${colors.primary}-100 flex items-center justify-center`}>
                <span className={`text-${colors.primary}-
600 text-xl`}>🍳</span>
              </div>
              Bahan-bahan
            </h2>
            <ul className="space-y-3">
              {recipe.ingredients?.map((ingredient) => (
                <li
                  key={ingredient.id}
                  className="flex items-start gap-3 bg-
white/50 p-3 rounded-xl border border-white/60"
                >
                  <span className={`text-
${colors.primary}-600 mt-1`}>•</span>
                  <div>
                    <p className="font-medium text-slate-
700">{ingredient.name}</p>
                    <p className="text-sm text-slate-
500">{ingredient.quantity}</p>
                  </div>
                </li>
              ))}
            </ul>
          </div>

          {/* Steps */}
          <div className="bg-white/60 backdrop-blur-sm
rounded-3xl p-6 md:p-8 shadow-xl border border-white/40">
            <h2 className="text-2xl font-bold text-slate-
800 mb-6 flex items-center gap-3">
              <div className={`w-10 h-10 rounded-full bg-
${colors.primary}-100 flex items-center justify-center`}>
                <span className={`text-${colors.primary}-
600 text-xl`}>👨‍🍳</span>
              </div>
              Langkah-langkah
            </h2>
            <ol className="space-y-4">
              {recipe.steps?.map((step) => (
                <li
                  key={step.id}
                  className="flex gap-4 bg-white/50 p-4
rounded-xl border border-white/60"
```

```
                >
                  <div className={`flex-shrink-0 w-8 h-8
rounded-full bg-${colors.primary}-600 text-white flex
items-center justify-center font-bold text-sm`}>
                    {step.step_number}
                  </div>
                  <p className="text-slate-700 leading-
relaxed pt-1">
                    {step.instruction}
                  </p>
                </li>
              ))}
            </ol>
          </div>
        </div>

        {/* Reviews Section */}
        <div className="bg-white/60 backdrop-blur-sm
rounded-3xl p-6 md:p-8 shadow-xl border border-white/40">
          <div className="flex items-center justify-
between mb-6">
            <h2 className="text-2xl font-bold text-slate-
800">
              Ulasan ({reviews?.length || 0})
            </h2>
            <button
              onClick={() =>
setShowReviewForm(!showReviewForm)}
              className={`px-4 py-2 bg-${colors.primary}-
600 text-white rounded-xl hover:bg-${colors.primary}-700
transition-colors font-medium`}
            >
              {showReviewForm ? 'Batal' : 'Tulis Ulasan'}
            </button>
          </div>

          {/* Review Form */}
          {showReviewForm && (
            <form onSubmit={handleSubmitReview}
className="mb-8 bg-white/70 rounded-2xl p-6 border
border-white/60">
              <div className="mb-4">
                <label className="block text-sm font-
medium text-slate-700 mb-2">
                  Rating
                </label>
                <div className="flex gap-2">
                  {[1, 2, 3, 4, 5].map((star) => (
                    <button
                      key={star}
                      type="button"
                      onClick={() => setRating(star)}
                      className="focus:outline-none"
                    >
                      <Star
                        className={`w-8 h-8 ${
```

```
                            star <= rating
                              ? 'text-amber-500 fill-
current'
                              : 'text-slate-300'
                          } hover:scale-110 transition-
transform`}
                        />
                      </button>
                    ))}
                  </div>
                </div>

                <div className="mb-4">
                  <label className="block text-sm font-
medium text-slate-700 mb-2">
                    Komentar
                  </label>
                  <textarea
                    value={comment}
                    onChange={(e) =>
setComment(e.target.value)}
                    placeholder="Bagikan pengalaman Anda
dengan resep ini..."
                    rows={4}
                    className="w-full px-4 py-3 border
border-slate-300 rounded-xl focus:ring-2 focus:ring-blue-
500 focus:border-blue-500 bg-white resize-none"
                  />
                </div>

                <button
                  type="submit"
                  disabled={createLoading ||
!comment.trim()}
                  className={`w-full md:w-auto px-6 py-3
bg-${colors.primary}-600 text-white rounded-xl hover:bg-
${colors.primary}-700 transition-colors font-medium
disabled:opacity-50 disabled:cursor-not-allowed flex
items-center justify-center gap-2`}
                >
                  <Send className="w-4 h-4" />
                  {createLoading ? 'Mengirim...' : 'Kirim
Ulasan'}
                </button>
              </form>
            )}

            {/* Reviews List */}
            <div className="space-y-4">
              {reviewsLoading ? (
                <div className="text-center py-8">
                  <div className={`animate-spin rounded-
full h-8 w-8 border-b-2 border-${colors.primary}-600 mx-
auto`}></div>
                </div>
              ) : reviews && reviews.length > 0 ? (
```

```
              reviews.map((review) => (
                <div
                  key={review.id}
                  className="bg-white/70 rounded-2xl p-6
border border-white/60"
                >
                  <div className="flex items-start
justify-between mb-3">
                    <div>
                      <p className="font-semibold text-
slate-800">
                        {review.user_identifier}
                      </p>
                      <div className="flex items-center
gap-1 mt-1">
                        {[1, 2, 3, 4, 5].map((star) => (
                          <Star
                            key={star}
                            className={`w-4 h-4 ${
                              star <= review.rating
                                ? 'text-amber-500 fill-
current'
                                : 'text-slate-300'
                            }`}
                          />
                        ))}
                      </div>
                    </div>
                    <p className="text-sm text-slate-
500">
                      {formatDate(review.created_at)}
                    </p>
                  </div>
                  {review.comment && (
                    <p className="text-slate-700 leading-
relaxed">
                      {review.comment}
                    </p>
                  )}
                </div>
              ))
            ) : (
              <div className="text-center py-8">
                <p className="text-slate-500">Belum ada
ulasan untuk resep ini.</p>
                <p className="text-slate-400 text-sm mt-
2">
                  Jadilah yang pertama memberikan ulasan!
                </p>
              </div>
            )}
          </div>
        </div>
      </main>
    </div>
  );
```

```
}
```

8. *Update file* main.jsx untuk handle operasi CRUD yang baru ditambahkan

Main.jsx

```
// src/main.jsx
import { StrictMode, useState } from 'react'
import { createRoot } from 'react-dom/client'
import SplashScreen from './pages/SplashScreen';
import HomePage from './pages/HomePage';
import MakananPage from './pages/MakananPage';
import MinumanPage from './pages/MinumanPage';
import ProfilePage from './pages/ProfilePage';
import CreateRecipePage from './pages/CreateRecipePage';
import EditRecipePage from './pages/EditRecipePage';
import RecipeDetail from
'./components/recipe/RecipeDetail';
import DesktopNavbar from
'./components/navbar/DesktopNavbar';
import MobileNavbar from
'./components/navbar/MobileNavbar';
import './index.css'
import PWABadge from './PWABadge';

function AppRoot() {
  const [showSplash, setShowSplash] = useState(true);
  const [currentPage, setCurrentPage] = useState('home');
  const [mode, setMode] = useState('list'); // 'list',
'detail', 'create', 'edit'
  const [selectedRecipeId, setSelectedRecipeId] =
useState(null);
  const [selectedCategory, setSelectedCategory] =
useState('makanan');
  const [editingRecipeId, setEditingRecipeId] =
useState(null);

  const handleSplashComplete = () => {
    setShowSplash(false);
  };

  const handleNavigation = (page) => {
    setCurrentPage(page);
    setMode('list');
    setSelectedRecipeId(null);
    setEditingRecipeId(null);
  };

  const handleCreateRecipe = () => {
    setMode('create');
  };

  const handleRecipeClick = (recipeId, category) => {
    setSelectedRecipeId(recipeId);
    setSelectedCategory(category || currentPage);
```

```
    setMode('detail');
  };

  const handleEditRecipe = (recipeId) => {
    console.log('🔧 Edit button clicked! Recipe ID:',
recipeId);
    setEditingRecipeId(recipeId);
    setMode('edit');
    console.log('✅ Mode changed to: edit');
  };

  const handleBack = () => {
    setMode('list');
    setSelectedRecipeId(null);
    setEditingRecipeId(null);
  };

  const handleCreateSuccess = (newRecipe) => {
    alert('Resep berhasil dibuat!');
    setMode('list');
    // Optionally navigate to the new recipe's category
    if (newRecipe && newRecipe.category) {
      setCurrentPage(newRecipe.category);
    }
  };

  const handleEditSuccess = (updatedRecipe) => {
    alert('Resep berhasil diperbarui!');
    setMode('list');
  };

  const renderCurrentPage = () => {
    // Show Create Recipe Page
    if (mode === 'create') {
      return (
        <CreateRecipePage
          onBack={handleBack}
          onSuccess={handleCreateSuccess}
        />
      );
    }

    // Show Edit Recipe Page
    if (mode === 'edit') {
      return (
        <EditRecipePage
          recipeId={editingRecipeId}
          onBack={handleBack}
          onSuccess={handleEditSuccess}
        />
      );
    }

    // Show Recipe Detail
    if (mode === 'detail') {
      return (
```

```
        <RecipeDetail
          recipeId={selectedRecipeId}
          category={selectedCategory}
          onBack={handleBack}
          onEdit={handleEditRecipe}
        />
      );
    }

    // Show List Pages
    switch (currentPage) {
      case 'home':
        return <HomePage
onRecipeClick={handleRecipeClick}
onNavigate={handleNavigation} />;
      case 'makanan':
        return <MakananPage
onRecipeClick={handleRecipeClick} />;
      case 'minuman':
        return <MinumanPage
onRecipeClick={handleRecipeClick} />;
      case 'profile':
        return <ProfilePage
onRecipeClick={handleRecipeClick} />;
      default:
        return <HomePage
onRecipeClick={handleRecipeClick}
onNavigate={handleNavigation} />;
    }
  };

  if (showSplash) {
    return <SplashScreen
onComplete={handleSplashComplete} />;
  }

  return (
    <div className="min-h-screen bg-gray-50">
      {/* Only show navbar in list mode */}
      {mode === 'list' && (
        <>
          <DesktopNavbar
            currentPage={currentPage}
            onNavigate={handleNavigation}
            onCreateRecipe={handleCreateRecipe}
          />
          <MobileNavbar
            currentPage={currentPage}
            onNavigate={handleNavigation}
            onCreateRecipe={handleCreateRecipe}
          />
        </>
      )}

      {/* Main Content */}
      <main className="min-h-screen">
```

```
        {renderCurrentPage()}
      </main>

      <PWABadge />
    </div>
  );
}

createRoot(document.getElementById('root')).render(
  <StrictMode>
    <AppRoot />
  </StrictMode>,
)
```

9. Tambahkan helper di dalam folder src

Utils/helpers.js

```
// src/utils/helpers.js

/**
 * Format date to readable string
 * @param {string} dateString - ISO date string
 * @returns {string} - Formatted date
 */
export function formatDate(dateString) {
  if (!dateString) return '';

  const date = new Date(dateString);
  const options = {
    year: 'numeric',
    month: 'long',
    day: 'numeric',
  };

  return date.toLocaleDateString('id-ID', options);
}

/**
 * Format date to relative time (e.g., "2 hours ago")
 * @param {string} dateString - ISO date string
 * @returns {string} - Relative time string
 */
export function formatRelativeTime(dateString) {
  if (!dateString) return '';

  const date = new Date(dateString);
  const now = new Date();
  const diffInSeconds = Math.floor((now - date) / 1000);

  if (diffInSeconds < 60) {
    return 'Baru saja';
  }

  const diffInMinutes = Math.floor(diffInSeconds / 60);
```

```
  if (diffInMinutes < 60) {
    return `${diffInMinutes} menit yang lalu`;
  }

  const diffInHours = Math.floor(diffInMinutes / 60);
  if (diffInHours < 24) {
    return `${diffInHours} jam yang lalu`;
  }

  const diffInDays = Math.floor(diffInHours / 24);
  if (diffInDays < 7) {
    return `${diffInDays} hari yang lalu`;
  }

  const diffInWeeks = Math.floor(diffInDays / 7);
  if (diffInWeeks < 4) {
    return `${diffInWeeks} minggu yang lalu`;
  }

  const diffInMonths = Math.floor(diffInDays / 30);
  if (diffInMonths < 12) {
    return `${diffInMonths} bulan yang lalu`;
  }

  const diffInYears = Math.floor(diffInDays / 365);
  return `${diffInYears} tahun yang lalu`;
}

/**
 * Get difficulty badge color
 * @param {string} difficulty - Recipe difficulty
 * @returns {string} - Tailwind color classes
 */
export function getDifficultyColor(difficulty) {
  const colors = {
    mudah: 'bg-green-100 text-green-800',
    sedang: 'bg-yellow-100 text-yellow-800',
    sulit: 'bg-red-100 text-red-800',
  };
  return colors[difficulty?.toLowerCase()] || 'bg-gray-
100 text-gray-800';
}

/**
 * Get category icon/emoji
 * @param {string} category - Recipe category
 * @returns {string} - Emoji
 */
export function getCategoryEmoji(category) {
  const emojis = {
    makanan: '🍲',
    minuman: '🥤',
  };
  return emojis[category?.toLowerCase()] || '🍽️';
}
```

```
/**
 * Validate rating (1-5)
 * @param {number} rating - Rating value
 * @returns {boolean} - Is valid
 */
export function isValidRating(rating) {
  return typeof rating === 'number' && rating >= 1 &&
rating <= 5;
}

/**
 * Generate star rating display
 * @param {number} rating - Rating value (0-5)
 * @returns {string} - Star string
 */
export function getStarRating(rating) {
  const fullStars = Math.floor(rating);
  const hasHalfStar = rating % 1 >= 0.5;
  const emptyStars = 5 - fullStars - (hasHalfStar ? 1 :
0);

  return '★'.repeat(fullStars) +
         (hasHalfStar ? '✨' : '') +
         '☆'.repeat(emptyStars);
}

/**
 * Truncate text to specific length
 * @param {string} text - Text to truncate
 * @param {number} length - Max length
 * @returns {string} - Truncated text
 */
export function truncateText(text, length = 100) {
  if (!text) return '';
  if (text.length <= length) return text;
  return text.substring(0, length) + '...';
}

/**
 * Debounce function
 * @param {Function} func - Function to debounce
 * @param {number} wait - Wait time in ms
 * @returns {Function} - Debounced function
 */
export function debounce(func, wait = 300) {
  let timeout;
  return function executedFunction(...args) {
    const later = () => {
      clearTimeout(timeout);
      func(...args);
    };
    clearTimeout(timeout);
    timeout = setTimeout(later, wait);
  };
}
```

Utils/draftStorage.js

```
// src/utils/draftStorage.js

const DRAFT_KEY_PREFIX = 'recipe_draft_';
const DRAFT_TIMESTAMP_KEY = 'recipe_draft_timestamp';

/**
 * Save recipe draft to localStorage
 * @param {Object} draftData - Recipe draft data
 * @param {string} draftId - Unique identifier for the
draft (e.g., 'create' or recipeId for edit)
 */
export const saveDraft = (draftData, draftId = 'create')
=> {
  try {
    const key = `${DRAFT_KEY_PREFIX}${draftId}`;
    const timestamp = new Date().toISOString();

    const draft = {
      ...draftData,
      savedAt: timestamp,
    };

    localStorage.setItem(key, JSON.stringify(draft));
    localStorage.setItem(`${key}_${DRAFT_TIMESTAMP_KEY}`,
timestamp);

    console.log(`Draft saved: ${draftId} at
${timestamp}`);
    return true;
  } catch (error) {
    console.error('Error saving draft:', error);
    return false;
  }
};

/**
 * Load recipe draft from localStorage
 * @param {string} draftId - Unique identifier for the
draft
 * @returns {Object|null} Draft data or null if not found
 */
export const loadDraft = (draftId = 'create') => {
  try {
    const key = `${DRAFT_KEY_PREFIX}${draftId}`;
    const draftJson = localStorage.getItem(key);

    if (!draftJson) {
      return null;
    }

    const draft = JSON.parse(draftJson);
    console.log(`Draft loaded: ${draftId}`);
    return draft;
```

```
  } catch (error) {
    console.error('Error loading draft:', error);
    return null;
  }
};

/**
 * Delete recipe draft from localStorage
 * @param {string} draftId - Unique identifier for the
draft
 */
export const deleteDraft = (draftId = 'create') => {
  try {
    const key = `${DRAFT_KEY_PREFIX}${draftId}`;
    localStorage.removeItem(key);

localStorage.removeItem(`${key}_${DRAFT_TIMESTAMP_KEY}`);

    console.log(`Draft deleted: ${draftId}`);
    return true;
  } catch (error) {
    console.error('Error deleting draft:', error);
    return false;
  }
};

/**
 * Check if a draft exists
 * @param {string} draftId - Unique identifier for the
draft
 * @returns {boolean} True if draft exists
 */
export const hasDraft = (draftId = 'create') => {
  try {
    const key = `${DRAFT_KEY_PREFIX}${draftId}`;
    return localStorage.getItem(key) !== null;
  } catch (error) {
    console.error('Error checking draft:', error);
    return false;
  }
};

/**
 * Get draft timestamp
 * @param {string} draftId - Unique identifier for the
draft
 * @returns {string|null} ISO timestamp or null
 */
export const getDraftTimestamp = (draftId = 'create') =>
{
  try {
    const key = `${DRAFT_KEY_PREFIX}${draftId}`;
    return
localStorage.getItem(`${key}_${DRAFT_TIMESTAMP_KEY}`);
  } catch (error) {
```

```
    console.error('Error getting draft timestamp:',
error);
    return null;
  }
};

/**
 * Get all draft IDs
 * @returns {string[]} Array of draft IDs
 */
export const getAllDraftIds = () => {
  try {
    const keys = Object.keys(localStorage);
    const draftKeys = keys.filter(key =>
key.startsWith(DRAFT_KEY_PREFIX) &&
!key.includes(DRAFT_TIMESTAMP_KEY));
    return draftKeys.map(key =>
key.replace(DRAFT_KEY_PREFIX, ''));
  } catch (error) {
    console.error('Error getting all draft IDs:', error);
    return [];
  }
};

/**
 * Clear all recipe drafts
 */
export const clearAllDrafts = () => {
  try {
    const draftIds = getAllDraftIds();
    draftIds.forEach(id => deleteDraft(id));
    console.log(`Cleared ${draftIds.length} drafts`);
    return true;
  } catch (error) {
    console.error('Error clearing all drafts:', error);
    return false;
  }
};

/**
 * Format draft timestamp for display
 * @param {string} timestamp - ISO timestamp
 * @returns {string} Formatted time ago
 */
export const formatDraftTime = (timestamp) => {
  if (!timestamp) return '';

  const now = new Date();
  const savedTime = new Date(timestamp);
  const diffMs = now - savedTime;
  const diffMins = Math.floor(diffMs / 60000);

  if (diffMins < 1) return 'Baru saja';
  if (diffMins < 60) return `${diffMins} menit yang
lalu`;
```

```
  const diffHours = Math.floor(diffMins / 60);
  if (diffHours < 24) return `${diffHours} jam yang
lalu`;

  const diffDays = Math.floor(diffHours / 24);
  if (diffDays < 7) return `${diffDays} hari yang lalu`;

  return savedTime.toLocaleDateString('id-ID');
};
```

10. Atur *icon*/logo untuk PWA

Index.html

```
<!doctype html>
<html lang="id">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" href="/favicon.ico" sizes="48x48">
    <link rel="apple-touch-icon" href="/apple-touch-icon-
180x180.png">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0" />

    <!-- PWA Meta Tags -->
    <meta name="theme-color" content="#ffffff" />
    <meta name="description" content="Resep Makanan dan
Minuman Indonesia" />
    <meta name="mobile-web-app-capable" content="yes" />
    <meta name="apple-mobile-web-app-capable"
content="yes" />
    <meta name="apple-mobile-web-app-status-bar-style"
content="default" />
    <meta name="apple-mobile-web-app-title"
content="Resep Nusantara" />

    <title>Resep Nusantara</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

Vite.config.js

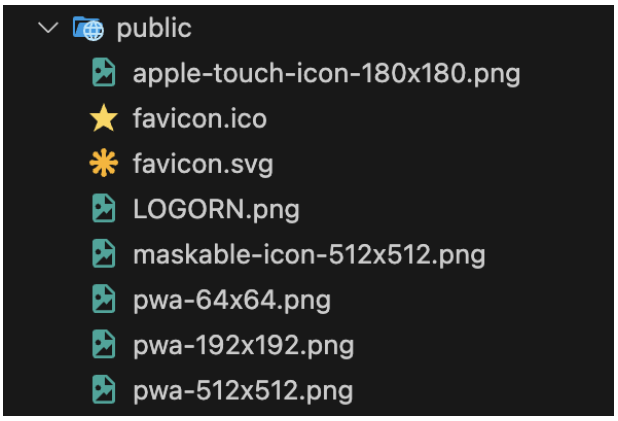
```
import { VitePWA } from 'vite-plugin-pwa';
import { defineConfig } from 'vite'
import tailwindcss from '@tailwindcss/vite'
import react from '@vitejs/plugin-react'

// https://vitejs.dev/config/
export default defineConfig({
  plugins: [react(), VitePWA(), tailwindcss(),({
```

```
    registerType: 'autoUpdate',
    includeAssets: ['favicon.ico', 'robots.txt', 'apple-
touch-icon.png', 'LOGORN.png'],
    injectRegister: false,

    pwaAssets: {
      disabled: false,
      config: true,
    },

    manifest: {
      name: 'Resep Nusantara',
      short_name: 'Resep Nusantara',
      description: 'Aplikasi Resep Makanan dan Minuman
Khas Indonesia',
      theme_color: '#2563eb',
      background_color: '#ffffff',
      display: 'standalone',
      scope: '/',
      start_url: '/',
      orientation: 'portrait',
      icons: [
        {
          src: '/pwa-64x64.png',
          sizes: '64x64',
          type: 'image/png'
        },
        {
          src: '/pwa-192x192.png',
          sizes: '192x192',
          type: 'image/png'
        },
        {
          src: '/pwa-512x512.png',
          sizes: '512x512',
          type: 'image/png',
          purpose: 'any'
        },
        {
          src: '/maskable-icon-512x512.png',
          sizes: '512x512',
          type: 'image/png',
          purpose: 'maskable'
        }
      ]
    },

    workbox: {
      globPatterns: ['**/*.{js,css,html,svg,png,ico}'],
      cleanupOutdatedCaches: true,
      clientsClaim: true,
    },

    devOptions: {
      enabled: false,
      navigateFallback: 'index.html',
```

```
      suppressWarnings: true,
      type: 'module',
    },
  })],
})
```

Icon PWA Resep Nusantara

Link *download* icon




Gambar 6. 3 *file icon* PWA

11. *Install* semua *dependencies* yang dibutuhkan dan cek PWA kalian

```
npm install lucide-react
npm install -D vite-plugin-pwa
npm install tailwindcss @tailwindcss/vite
npm install axios
```

Mode pengembangan

```
npm run dev
```

Mode Produksi

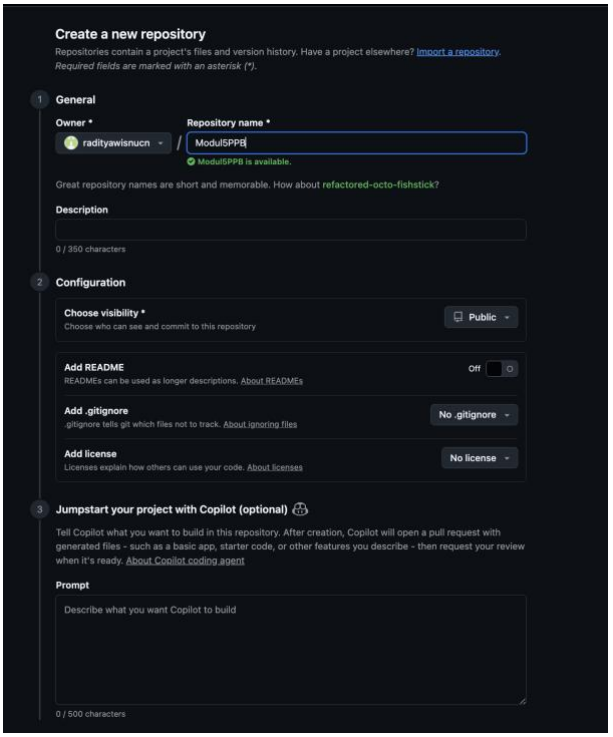
```
npm run build
```

### 6.4.3. *Deploy* PWA ke Vercel

1. Hapus *remote origin* lama (*remote repository* modul 4 sebelumnya) dengan *command* sebagai berikut

```
>  git remote remove origin
```

2. Buat *repository* baru pada Github




Gambar 6. 4 Repository baru untuk modul 5

3. *Copy link repository* yang telah dibuat dan masukkan *command* berikut

```
>  git remote add origin [link repo]
```

4. *Staged* perubahan kode yang ada pada *project* dengan *command* berikut

```
>  git add .
```

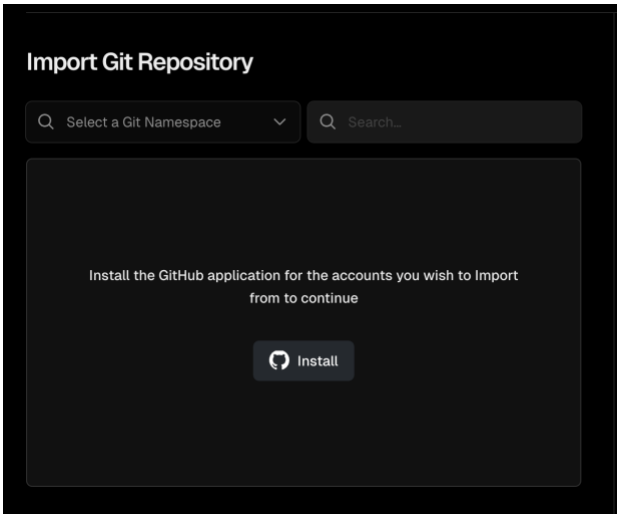
5. *Commit* perubahan dengan *command* berikut

```
>  git commit -m "[isi dengan pesan kalian]"
```

6. *Push project* ke *repository* Github dengan *command* berikut

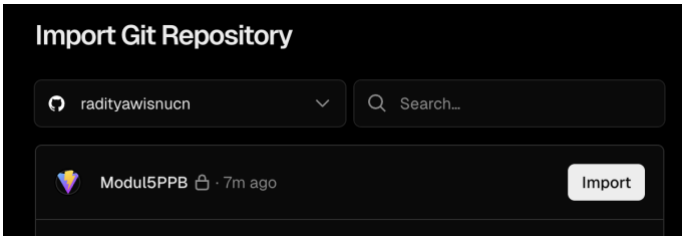
```
>  git push -u origin main
```

7. Buka Vercel dan *login* menggunakan Github kemudian *install* Github *app* untuk mengakses *repository* kalian



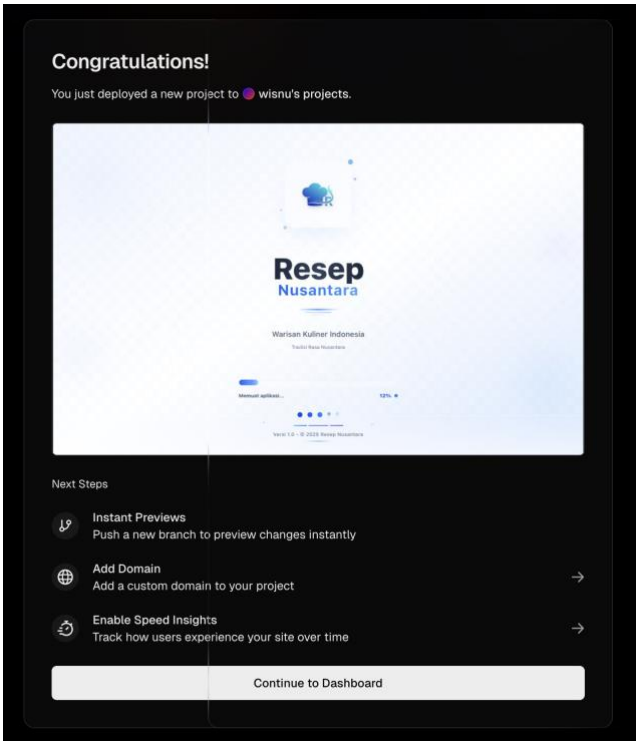

Gambar 6. 5 Import GitHub Repository

8. Pilih *repository* Github



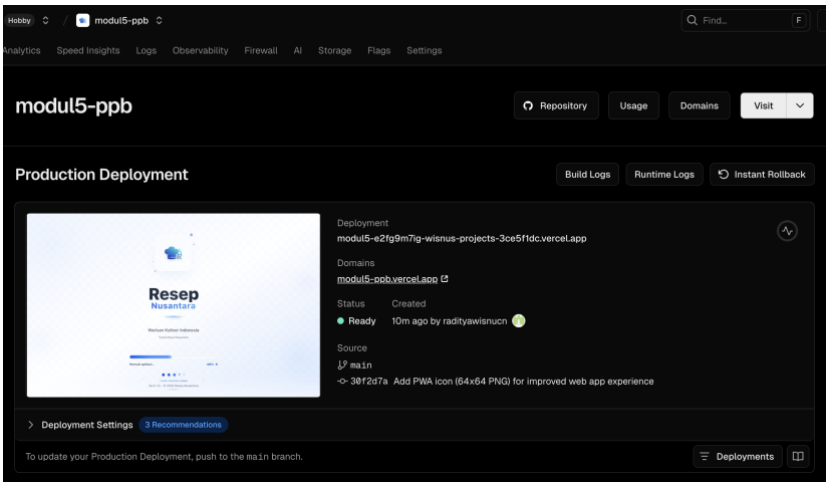

Gambar 6. 6 Pilih *repository* yang ingin di-*deploy*

9. *Import repository* percobaan dan PWA akan otomatis ter-*deploy*




Gambar 6. 7 PWA berhasil diunggah dan dapat diakses

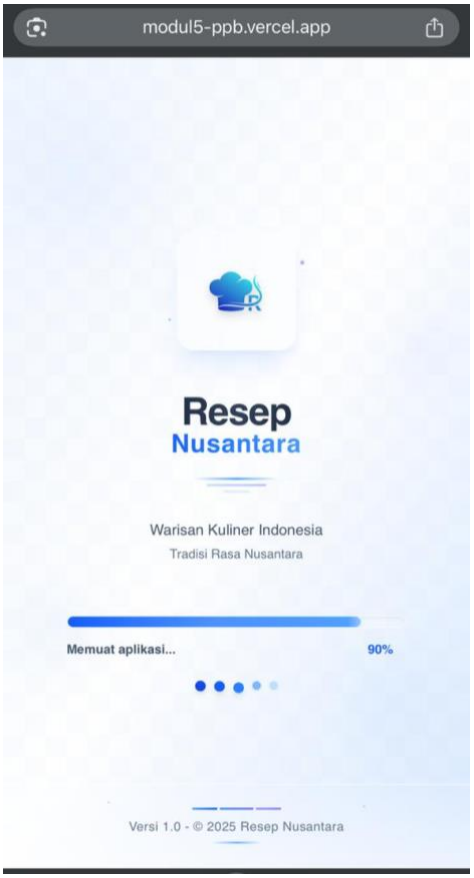
10. PWA bisa dicek di *domain* yang diberikan Vercel




Gambar 6. 8 Tampilan PWA pada Vercel

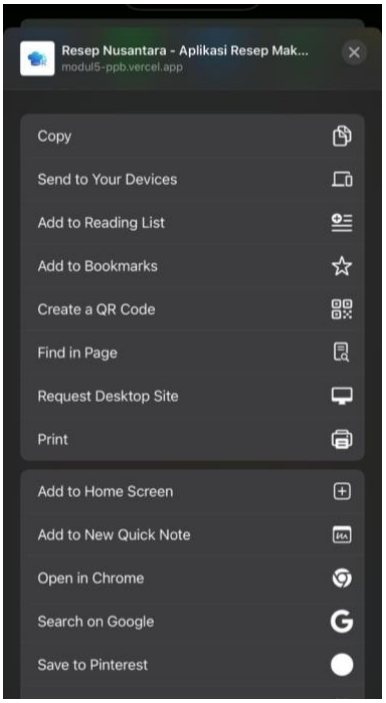
### 6.4.4. PWA pada Mobile

1. Buka *link* PWA yang telah di-*deploy*



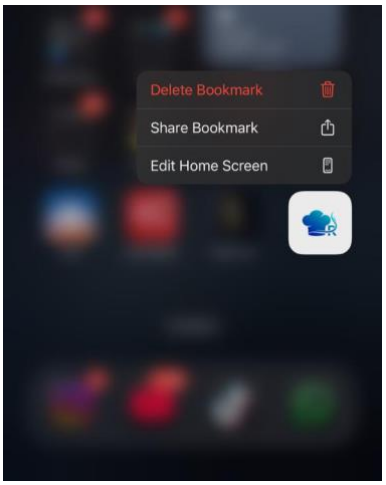

Gambar 6. 9 Buka *link* PWA

2. Klik *icon share* yang terletak di samping *link*, lalu *scroll* dan pilih “Add to Home Screen”




Gambar 6. 10 Add to Home Screen

3. PWA akan tampil pada *smartphone* kalian




Gambar 6. 11 Tampilan PWA pada *smartphone*