IPB University
Bogor Indonesia

Sekolah Vokasi

BETTABEAL Guidebook

MASTEAL

Dosen Pembimbing

Aditya Wicaksono, S.Komp., M.Kom.
Gema Parasti Mindara, S.Si., M.Kom.
Endang Purnama Giri S.Kom., M.Kom.
Lathifunnisa Fathonah M.T.

Anggota TIM

Aubrey Nedwin Mantiri	J0403231130
Muhammad Faris Fadhil Islam	J0403231146
Ferizwan Malik Wichaksana	J0403231157
Athala Fazli Maula	J0403231163

DAFTAR ISI

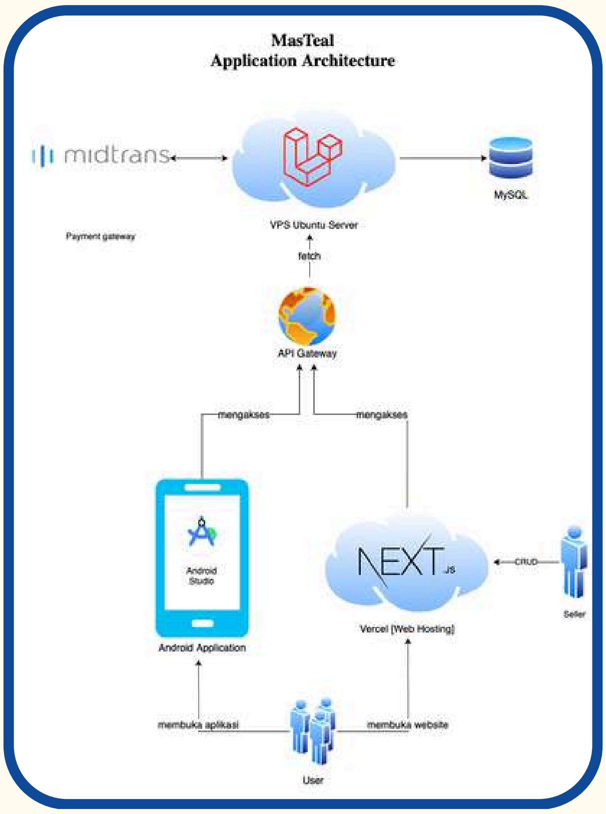
DESKRIPSI *Aplikasi*

BettaBeal adalah aplikasi mobile yang dirancang sebagai platform ecommerce atau marketplace khusus untuk jual beli ikan cupang. Aplikasi ini hadir untuk memfasilitasi pengguna yang ingin menjual atau membeli ikan cupang secara online dengan cara yang mudah, efisien, dan terpercaya.

Sebagai fitur utama, BettaBeal memberikan kemudahan bagi pengguna untuk melakukan transaksi jual beli ikan cupang. Fitur ini dirancang untuk memenuhi kebutuhan baik pemula yang baru memulai hobi memelihara ikan cupang, maupun para pecinta ikan cupang yang sudah berpengalaman. Pengguna dapat dengan mudah menjelajahi berbagai jenis ikan cupang yang tersedia dalam aplikasi, memilih ikan yang sesuai dengan preferensi mereka, dan melakukan transaksi langsung melalui platform ini dengan aman.

Selain menyediakan fitur jual beli, BettaBeal juga menawarkan artikel-artikel informatif yang mencakup panduan perawatan, jenis-jenis ikan cupang, serta tips menjaga kesehatannya. Untuk meningkatkan pengalaman pengguna, setiap artikel dilengkapi dengan fitur komentar yang memungkinkan pengguna berbagi pendapat, bertanya, atau berdiskusi dengan sesama pecinta ikan cupang. Fitur ini menjadikan BettaBeal tidak hanya sebagai platform jual beli dan sumber informasi, tetapi juga sebagai ruang interaksi untuk mempererat komunitas pecinta ikan cupang.

ARSITEKTUR Sistem



Arsitektur sistem ini mendukung aplikasi Android dan website berbasis Next.js yang terintegrasi dengan backend di VPS Ubuntu Server. User dapat mengakses layanan melalui aplikasi Android atau website yang di-hosting di Vercel, sementara seller menggunakan website untuk operasi CRUD. Semua permintaan diteruskan melalui API Gateway ke server backend yang menggunakan Laravel untuk memproses data, terhubung ke database MySQL, dan memproses pembayaran melalui Midtrans. Arsitektur ini memastikan layanan berjalan cepat, aman, dan terintegrasi.

APLIKASI *Mobile*

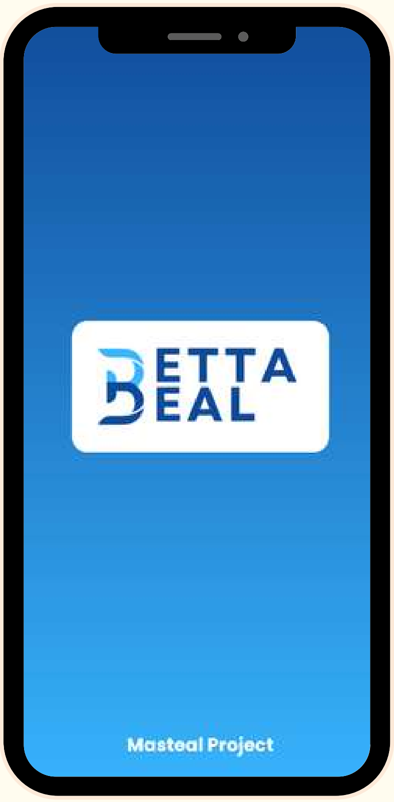
Bettabeal adalah aplikasi mobile e-commerce yang dikembangkan menggunakan Android Studio, dirancang khusus untuk penggemar ikan cupang. Aplikasi ini menawarkan fitur jual beli ikan cupang, perlengkapan ikan cupang, dan pakan, dengan kemudahan pembayaran langsung di dalam aplikasi. Selain itu, Bettabeal juga dilengkapi dengan fitur penampilan artikel informatif seputar ikan cupang, yang memberikan edukasi dan tips bermanfaat bagi pengguna. Dengan desain yang user-friendly dan fitur lengkap, Bettabeal menjadi solusi praktis bagi para pecinta ikan cupang untuk memenuhi kebutuhan mereka dalam satu platform.

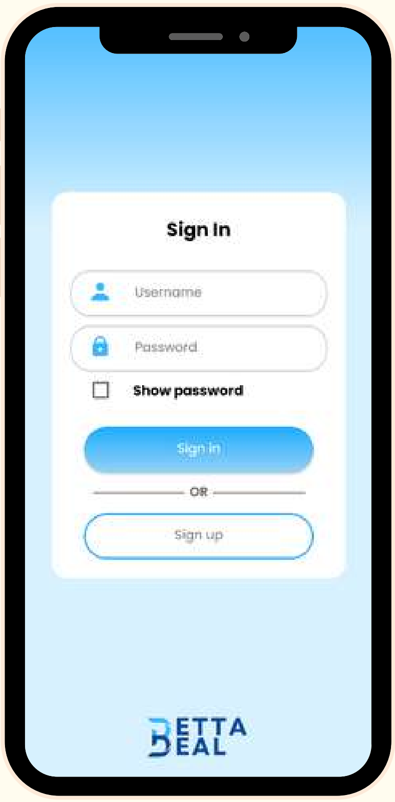
Aplikasi Mobile

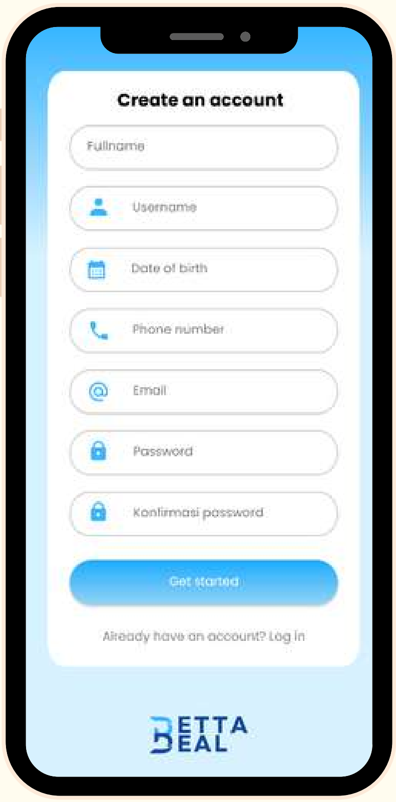
1. Splash screen, login, dan register page

Halaman ini menampilkan tampilan awal saat membuka aplikasi, ditampilkannya splashscreen selama 3 detik, setelah itu diarahkan ke halaman login, dan di halaman login user dapat mengarahkan dirinya ke halaman register untuk mendafrtarkan dan mendapatkan akun agar bisa login di halaman login.

Pageview
Splashscreen - Login page - Register page

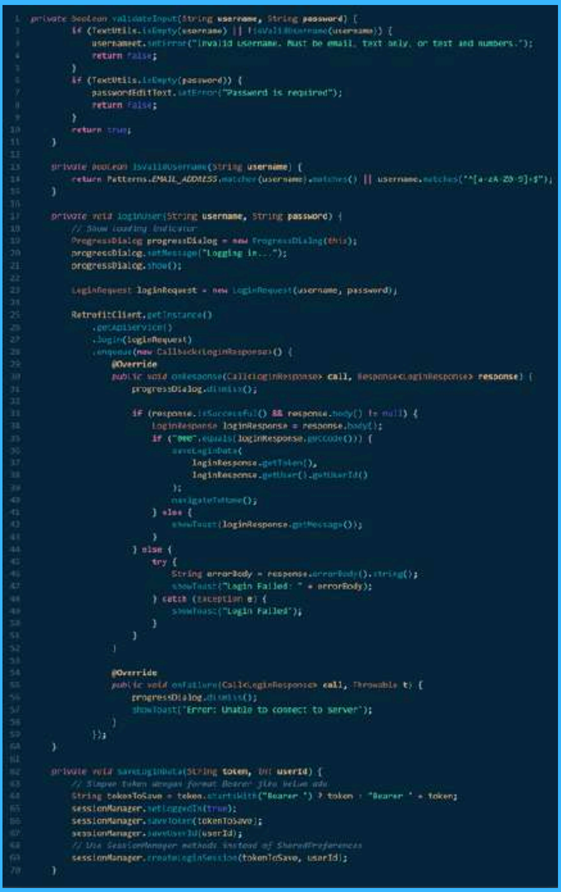




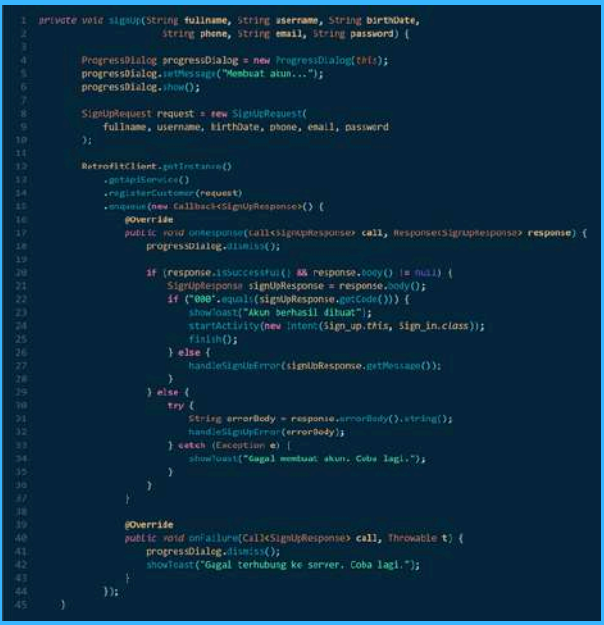


Aplikasi Mobile

Source Code
Login logic - Register logic



```java
private boolean validateInput(String username, String password) {
        if (TextUtils.isEmpty(username) || !isValidUsername(username)) {
            usernameEt.setError("Invalid username. Must be email, text only, or text and numbers.");
            return false;
        }
        if (TextUtils.isEmpty(password)) {
            passwordEditText.setError("Password is required");
            return false;
        }
        return true;
    }

    private boolean isValidUsername(String username) {
        return Patterns.EMAIL_ADDRESS.matcher(username).matches() || username.matches("^[a-zA-Z0-9]+$");
    }

    private void loginUser(String username, String password) {
        // Show loading indicator
        ProgressDialog progressDialog = new ProgressDialog(this);
        progressDialog.setMessage("Logging in...");
        progressDialog.show();

        LoginRequest loginRequest = new LoginRequest(username, password);

        RetrofitClient.getInstance()
            .getApiService()
            .login(loginRequest)
            .enqueue(new Callback<LoginResponse>() {
                @Override
                public void onResponse(Call<LoginResponse> call, Response<LoginResponse> response) {
                    progressDialog.dismiss();

                    if (response.isSuccessful() && response.body() != null) {
                        LoginResponse loginResponse = response.body();
                        if ("000".equals(loginResponse.getCode())) {
                            saveLoginData(
                                loginResponse.getToken(),
                                loginResponse.getUser().getUserId()
                            );
                            navigateToHome();
                        } else {
                            showToast(loginResponse.getMessage());
                        }
                    } else {
                        try {
                            String errorBody = response.errorBody().string();
                            showToast("Login Failed: " + errorBody);
                        } catch (Exception e) {
                            showToast("Login Failed");
                        }
                    }
                }

                @Override
                public void onFailure(Call<LoginResponse> call, Throwable t) {
                    progressDialog.dismiss();
                    showToast("Error: Unable to connect to server");
                }
            });
    }

    private void saveLoginData(String token, int userId) {
        // Simpan token dengan format Bearer jika belum ada
        String tokenToSave = token.startsWith("Bearer ") ? token : "Bearer " + token;
        sessionManager.setLoggedIn(true);
        sessionManager.saveToken(tokenToSave);
        sessionManager.saveUserId(userId);
        // Use SessionManager methods instead of SharedPreferences
        sessionManager.createLoginSession(tokenToSave, userId);
    }
```



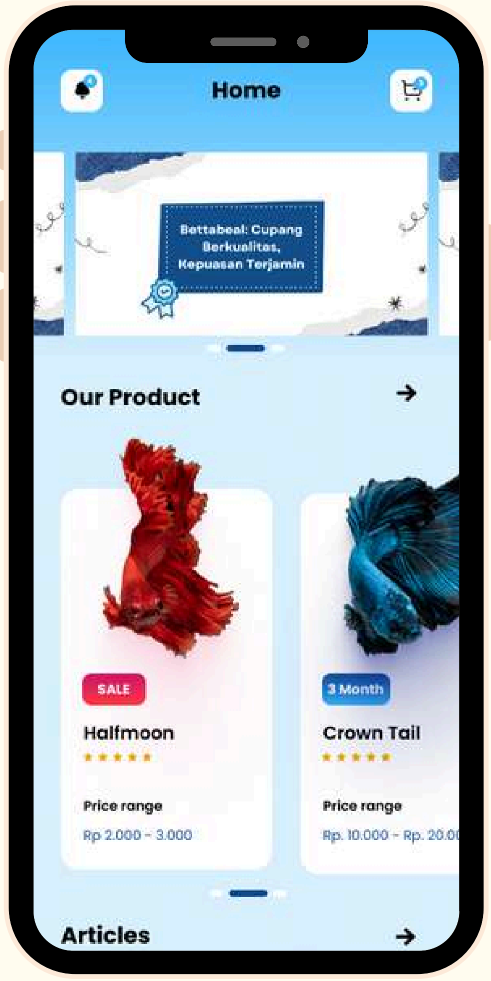
```java
private void signUp(String fullname, String username, String birthDate,
                    String phone, String email, String password) {

        ProgressDialog progressDialog = new ProgressDialog(this);
        progressDialog.setMessage("Membuat akun...");
        progressDialog.show();

        SignUpRequest request = new SignUpRequest(
            fullname, username, birthDate, phone, email, password
        );

        RetrofitClient.getInstance()
            .getApiService()
            .registerCustomer(request)
            .enqueue(new Callback<SignUpResponse>() {
                @Override
                public void onResponse(Call<SignUpResponse> call, Response<SignUpResponse> response) {
                    progressDialog.dismiss();

                    if (response.isSuccessful() && response.body() != null) {
                        SignUpResponse signUpResponse = response.body();
                        if ("000".equals(signUpResponse.getCode())) {
                            showToast("Akun berhasil dibuat");
                            startActivity(new Intent(Sign_up.this, Sign_in.class));
                            finish();
                        } else {
                            handleSignUpError(signUpResponse.getMessage());
                        }
                    } else {
                        try {
                            String errorBody = response.errorBody().string();
                            handleSignUpError(errorBody);
                        } catch (Exception e) {
                            showToast("Gagal membuat akun. Coba lagi.");
                        }
                    }
                }

                @Override
                public void onFailure(Call<SignUpResponse> call, Throwable t) {
                    progressDialog.dismiss();
                    showToast("Gagal terhubung ke server. Coba lagi.");
                }
            });
    }
```

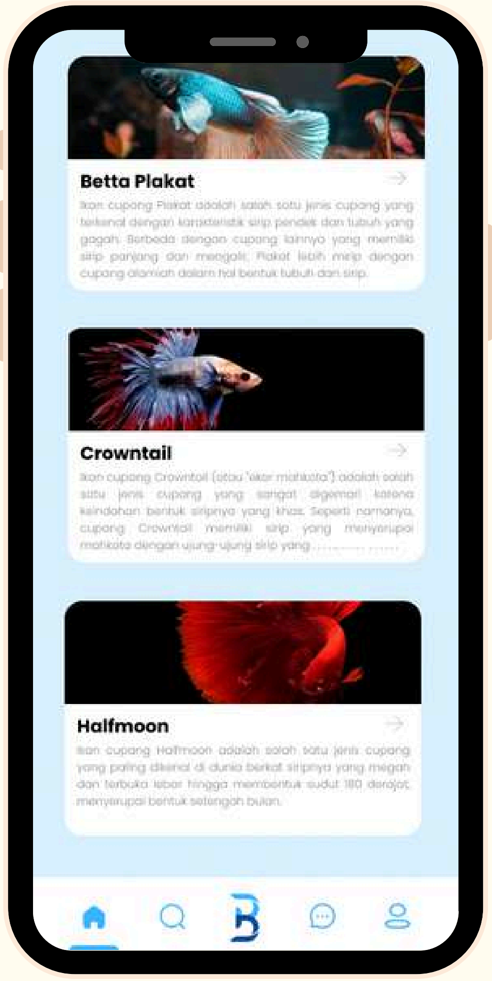
Aplikasi Mobile

2. Homepage

Halaman ini menampilkan tampilan setelah login, disediakan berbagai tombol untuk mengakses fitur yang tersedia di aplikasi. Dapat juga melihat Berita Terkini, cuaca, dll

Pageview
Homepage





Betta Plakat

Ikan cupang Plakat adalah salah satu jenis cupang yang terkenal dengan karakteristik sirip pendek dan tubuh yang gagah. Berbeda dengan cupang lainnya yang memiliki sirip panjang dan megah, Plakat lebih mirip dengan cupang alamiah dalam hal bentuk tubuh dan sirip.

Crowntail

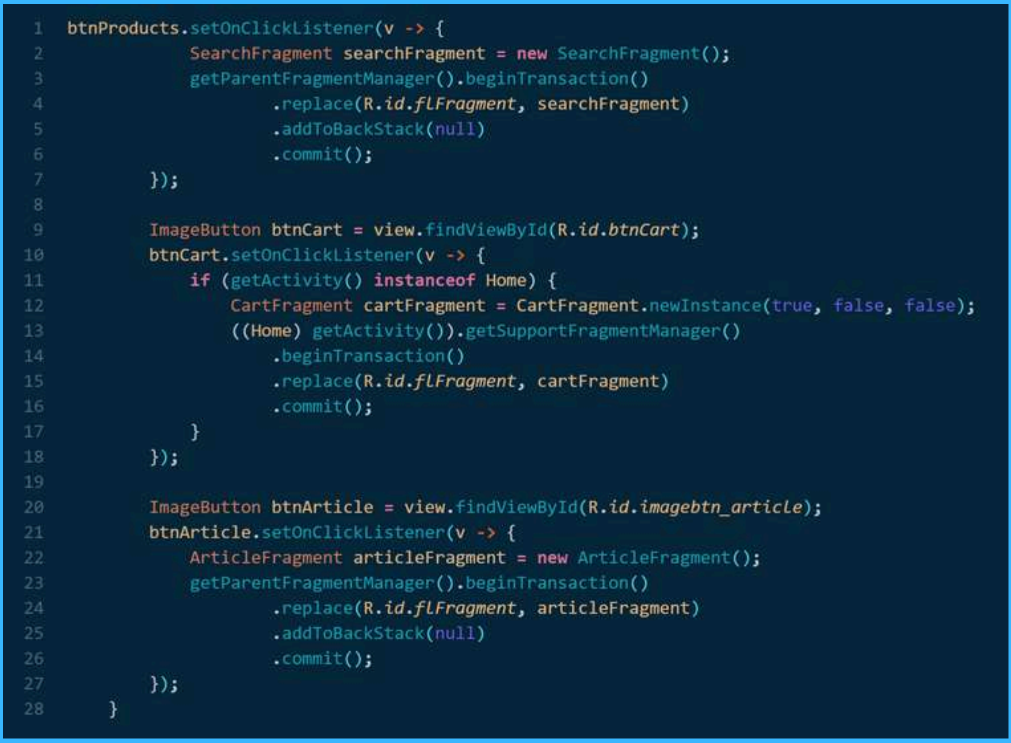
Ikan cupang Crowntail (atau 'ekor mahkota') adalah salah satu jenis cupang yang sangat digemari karena keindahan bentuk siripnya yang khas. Seperti namanya, cupang Crowntail memiliki sirip yang menyerupai mahkota dengan ujung-ujung sirip yang..............

Halfmoon

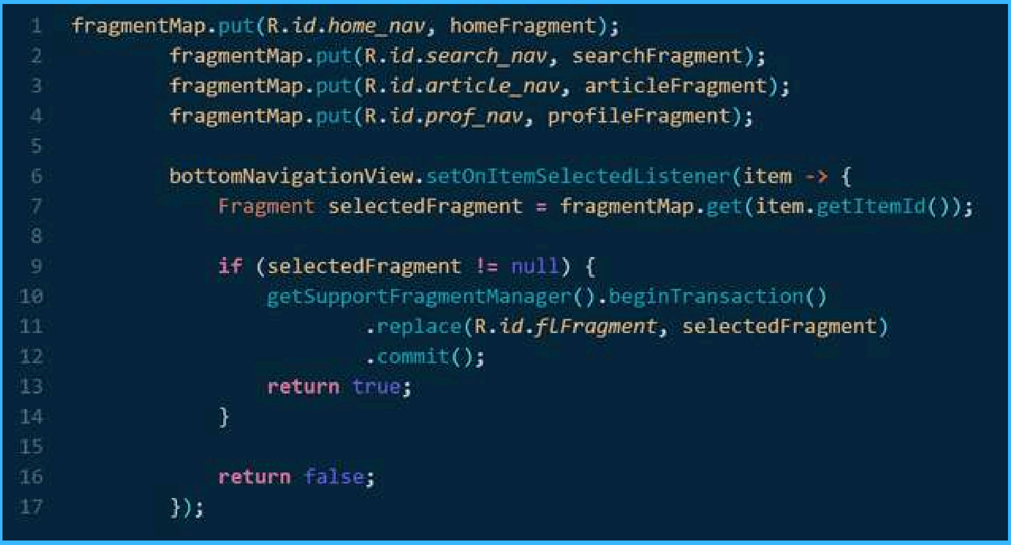
Ikan cupang Halfmoon adalah salah satu jenis cupang yang paling dikenal di dunia berkat siripnya yang megah dan terbuka lebar hingga membentuk sudut 180 derajat, menyerupai bentuk setengah bulan.

Aplikasi Mobile

Source Code
Button page routing - Navigation router



```java
btnProducts.setOnClickListener(v -> {
            SearchFragment searchFragment = new SearchFragment();
            getParentFragmentManager().beginTransaction()
                    .replace(R.id.flFragment, searchFragment)
                    .addToBackStack(null)
                    .commit();
        });

        ImageButton btnCart = view.findViewById(R.id.btnCart);
        btnCart.setOnClickListener(v -> {
            if (getActivity() instanceof Home) {
                CartFragment cartFragment = CartFragment.newInstance(true, false, false);
                ((Home) getActivity()).getSupportFragmentManager()
                    .beginTransaction()
                    .replace(R.id.flFragment, cartFragment)
                    .commit();
            }
        });

        ImageButton btnArticle = view.findViewById(R.id.imagebtn_article);
        btnArticle.setOnClickListener(v -> {
            ArticleFragment articleFragment = new ArticleFragment();
            getParentFragmentManager().beginTransaction()
                    .replace(R.id.flFragment, articleFragment)
                    .addToBackStack(null)
                    .commit();
        });
    }
```



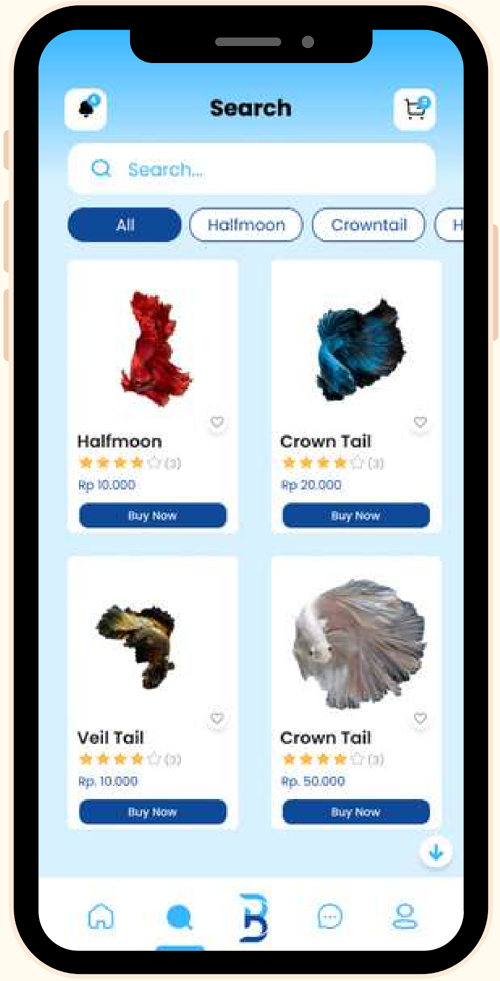
```java
fragmentMap.put(R.id.home_nav, homeFragment);
        fragmentMap.put(R.id.search_nav, searchFragment);
        fragmentMap.put(R.id.article_nav, articleFragment);
        fragmentMap.put(R.id.prof_nav, profileFragment);

        bottomNavigationView.setOnItemSelectedListener(item -> {
            Fragment selectedFragment = fragmentMap.get(item.getItemId());

            if (selectedFragment != null) {
                getSupportFragmentManager().beginTransaction()
                        .replace(R.id.flFragment, selectedFragment)
                        .commit();
                return true;
            }

            return false;
        });
```

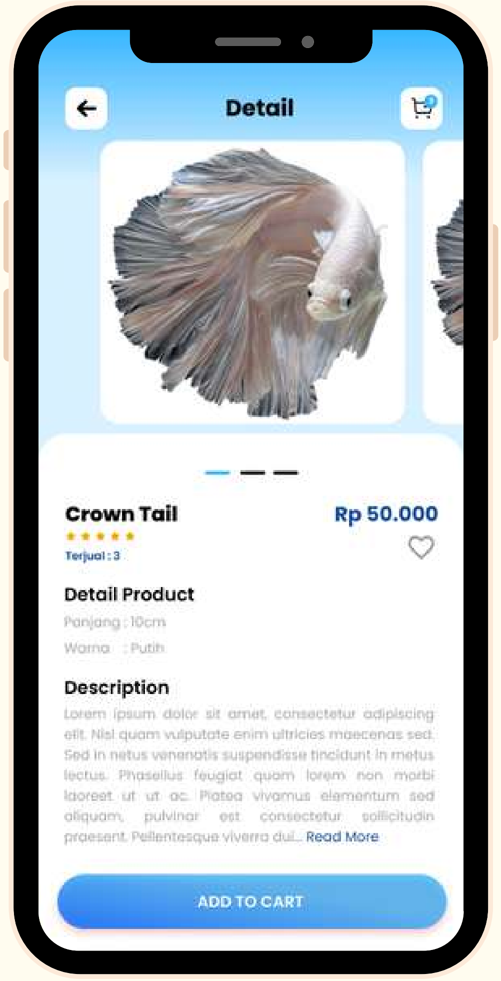
Aplikasi Mobile

3. Product Page

Halaman ini menampilkan tampilan List product. Lalu ketika salah satu Product diklik, akan menampilkan halaman detail product sesuai dengan product yang diklik.

Pageview
Product page - Detail product





Detail Product

Panjang : 10cm

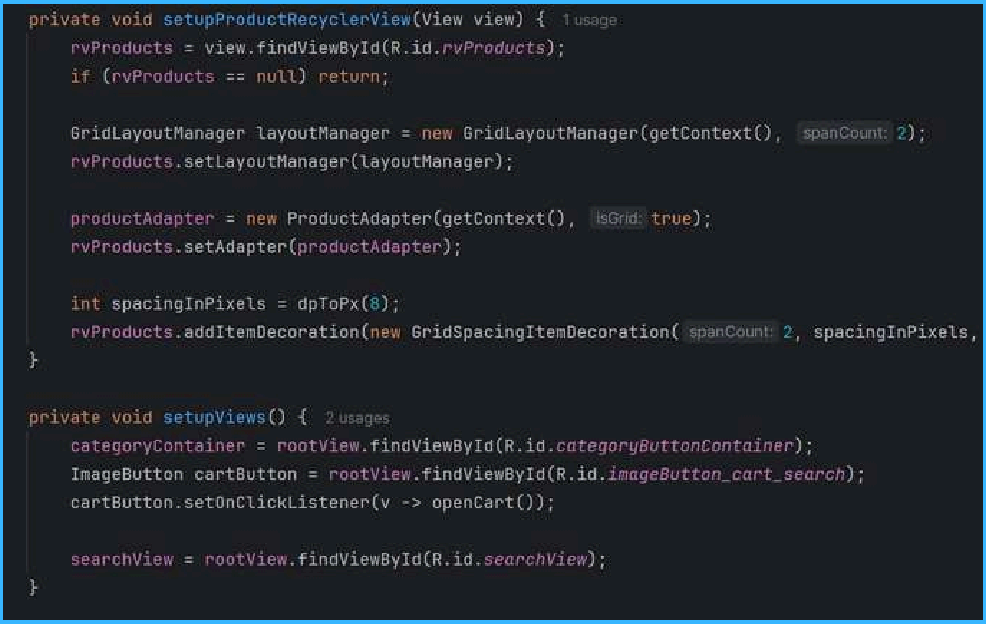
Warna : Putih

Description

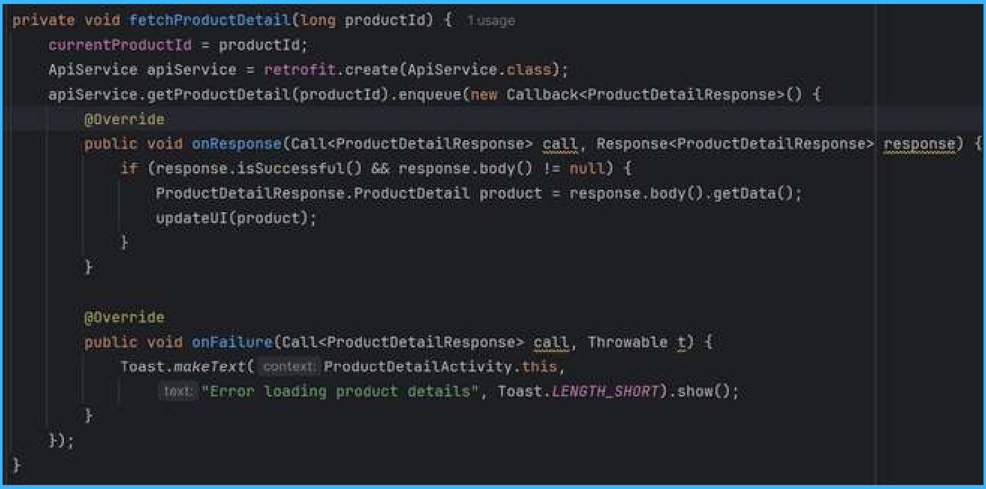
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nisl quam vulputate enim ultricies maecenas sed. Sed in netus venenatis suspendisse tincidunt in metus lectus. Phasellus feugiat quam lorem non morbi laoreet ut ut ac. Platea vivamus elementum sed aliquam, pulvinar est consectetur sollicitudin praesent. Pellentesque viverra dui... Read More

Aplikasi Mobile

Source Code
Product List - Product Detail



```java
private void setupProductRecyclerView(View view) {  1 usage
    rvProducts = view.findViewById(R.id.rvProducts);
    if (rvProducts == null) return;

    GridLayoutManager layoutManager = new GridLayoutManager(getContext(),  spanCount: 2);
    rvProducts.setLayoutManager(layoutManager);

    productAdapter = new ProductAdapter(getContext(),  isGrid: true);
    rvProducts.setAdapter(productAdapter);

    int spacingInPixels = dpToPx(8);
    rvProducts.addItemDecoration(new GridSpacingItemDecoration( spanCount: 2, spacingInPixels,
}

private void setupViews() {  2 usages
    categoryContainer = rootView.findViewById(R.id.categoryButtonContainer);
    ImageButton cartButton = rootView.findViewById(R.id.imageButton_cart_search);
    cartButton.setOnClickListener(v -> openCart());

    searchView = rootView.findViewById(R.id.searchView);
}
```



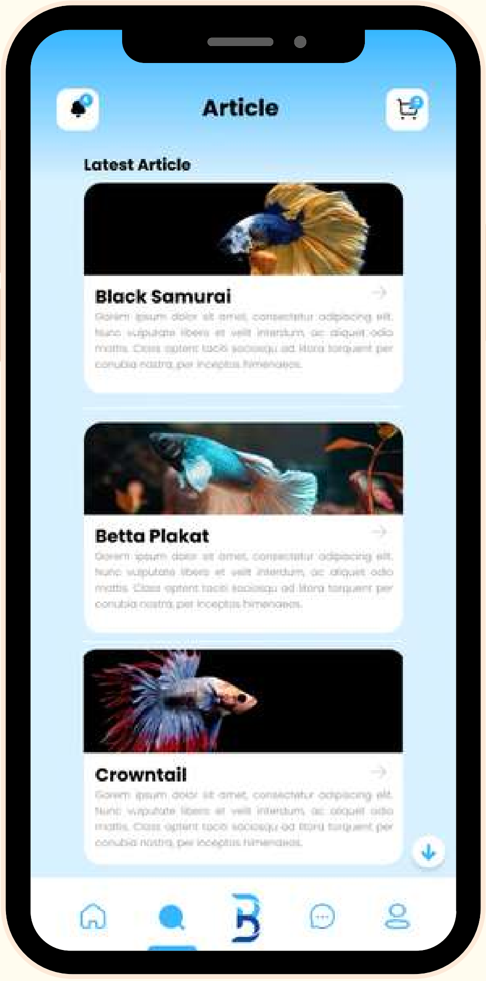
```java
private void fetchProductDetail(long productId) {  1 usage
    currentProductId = productId;
    ApiService apiService = retrofit.create(ApiService.class);
    apiService.getProductDetail(productId).enqueue(new Callback<ProductDetailResponse>() {
        @Override
        public void onResponse(Call<ProductDetailResponse> call, Response<ProductDetailResponse> response) {
            if (response.isSuccessful() && response.body() != null) {
                ProductDetailResponse.ProductDetail product = response.body().getData();
                updateUI(product);
            }
        }

        @Override
        public void onFailure(Call<ProductDetailResponse> call, Throwable t) {
            Toast.makeText( context: ProductDetailActivity.this,
                 text: "Error loading product details", Toast.LENGTH_SHORT).show();
        }
    });
}
```

Aplikasi Mobile

4. Article Page

Halaman ini menampilkan tampilan List Berita. Lalu ketika salah satu Berita diklik, akan menampilkan halaman detail Berita sesuai dengan Berita yang diklik sebelumnya.

Pageview
Article page - Detail article



Article

Latest Article

Black Samurai

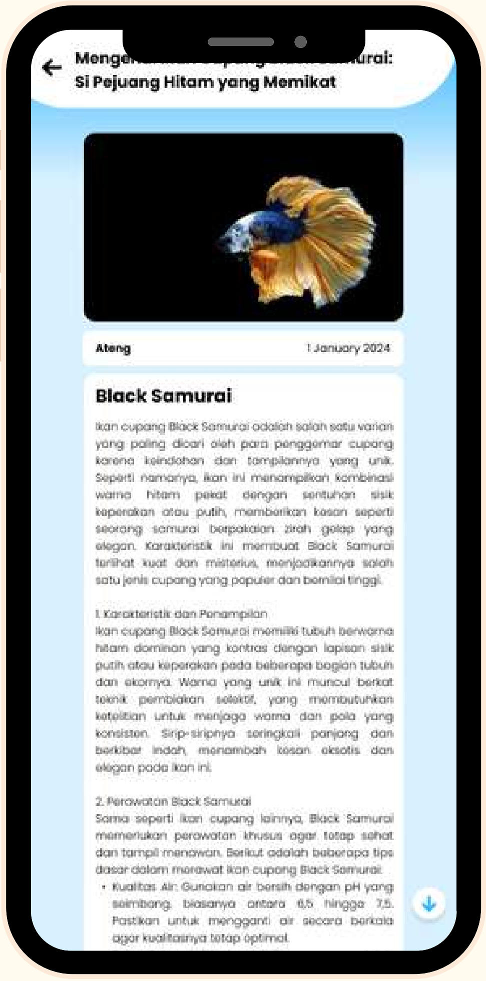
Gorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc vulputate libero et velit interdum, ac aliquet odio mattis. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

Betta Plakat

Gorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc vulputate libero et velit interdum, ac aliquet odio mattis. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.

Crowntail

Gorem ipsum dolor sit amet, consectetur adipiscing elit. Nunc vulputate libero et velit interdum, ac aliquet odio mattis. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos.



Menge[illegible]urai: Si Pejuang Hitam yang Memikat

Ateng 1 January 2024

Black Samurai

Ikan cupang Black Samurai adalah salah satu varian yang paling dicari oleh para penggemar cupang karena keindahan dan tampilannya yang unik. Seperti namanya, ikan ini menampilkan kombinasi warna hitam pekat dengan sentuhan sisik keperakan atau putih, memberikan kesan seperti seorang samurai berpakaian zirah gelap yang elegan. Karakteristik ini membuat Black Samurai terlihat kuat dan misterius, menjadikannya salah satu jenis cupang yang populer dan bernilai tinggi.

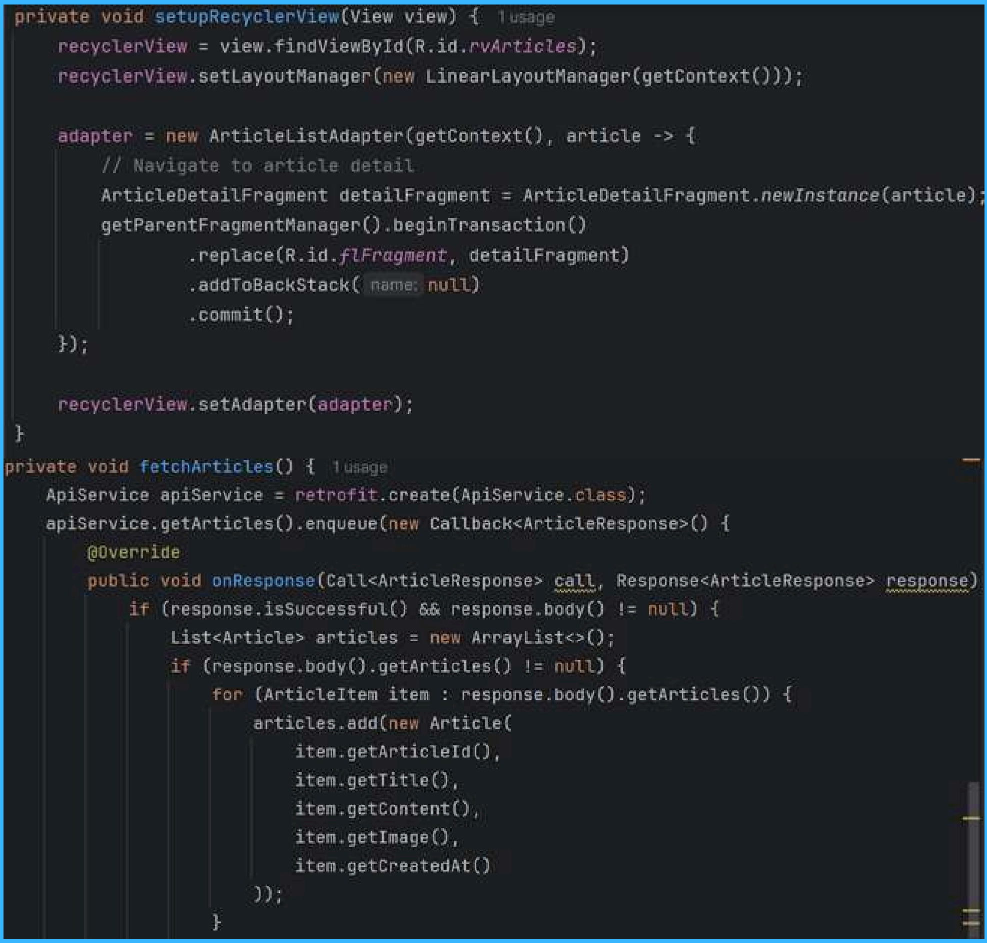
1. Karakteristik dan Penampilan
Ikan cupang Black Samurai memiliki tubuh berwarna hitam dominan yang kontras dengan lapisan sisik putih atau keperakan pada beberapa bagian tubuh dan ekornya. Warna yang unik ini muncul berkat teknik pembiakan selektif, yang membutuhkan ketelitian untuk menjaga warna dan pola yang konsisten. Sirip-siripnya seringkali panjang dan berkibar indah, menambah kesan eksotis dan elegan pada ikan ini.

2. Perawatan Black Samurai
Sama seperti ikan cupang lainnya, Black Samurai memerlukan perawatan khusus agar tetap sehat dan tampil menawan. Berikut adalah beberapa tips dasar dalam merawat ikan cupang Black Samurai:

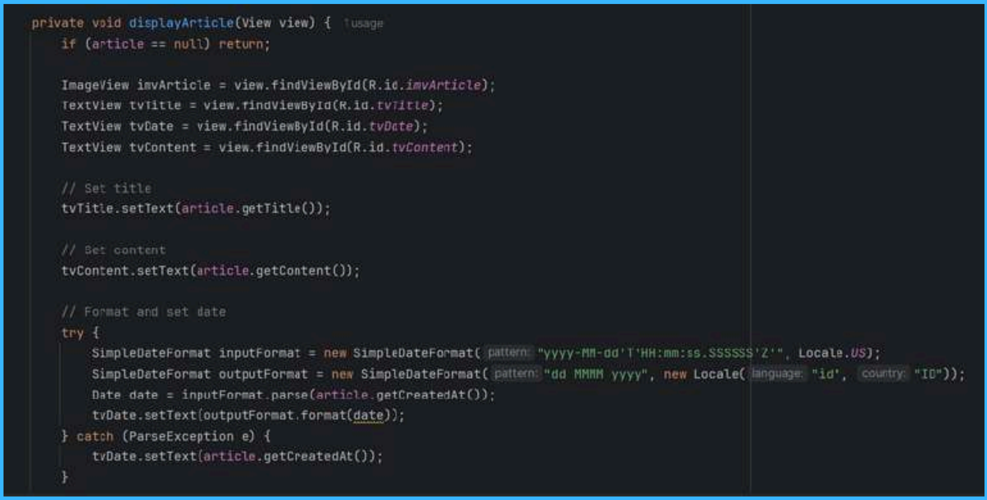
- Kualitas Air: Gunakan air bersih dengan pH yang seimbang, biasanya antara 6,5 hingga 7,5. Pastikan untuk mengganti air secara berkala agar kualitasnya tetap optimal.

Aplikasi Mobile

Source Code
Article List - Article Detail



```java
private void setupRecyclerView(View view) {  1 usage
    recyclerView = view.findViewById(R.id.rvArticles);
    recyclerView.setLayoutManager(new LinearLayoutManager(getContext()));

    adapter = new ArticleListAdapter(getContext(), article -> {
        // Navigate to article detail
        ArticleDetailFragment detailFragment = ArticleDetailFragment.newInstance(article);
        getParentFragmentManager().beginTransaction()
                .replace(R.id.flFragment, detailFragment)
                .addToBackStack( name: null)
                .commit();
    });

    recyclerView.setAdapter(adapter);
}
private void fetchArticles() {  1 usage
    ApiService apiService = retrofit.create(ApiService.class);
    apiService.getArticles().enqueue(new Callback<ArticleResponse>() {
        @Override
        public void onResponse(Call<ArticleResponse> call, Response<ArticleResponse> response)
            if (response.isSuccessful() && response.body() != null) {
                List<Article> articles = new ArrayList<>();
                if (response.body().getArticles() != null) {
                    for (ArticleItem item : response.body().getArticles()) {
                        articles.add(new Article(
                            item.getArticleId(),
                            item.getTitle(),
                            item.getContent(),
                            item.getImage(),
                            item.getCreatedAt()
                        ));
                    }
```



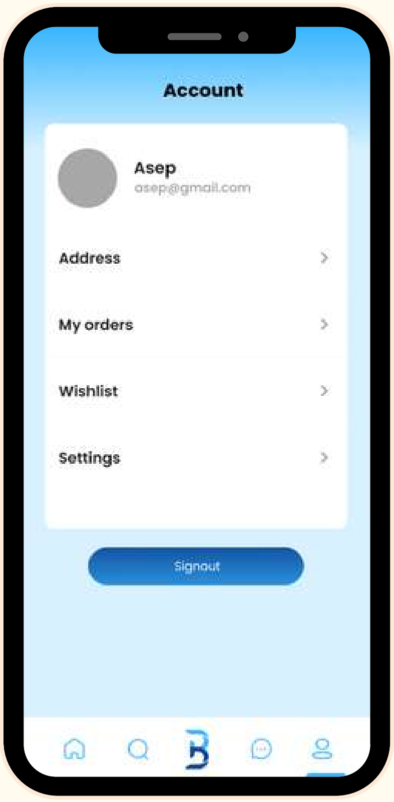
```java
private void displayArticle(View view) {  1 usage
    if (article == null) return;

    ImageView imvArticle = view.findViewById(R.id.imvArticle);
    TextView tvTitle = view.findViewById(R.id.tvTitle);
    TextView tvDate = view.findViewById(R.id.tvDate);
    TextView tvContent = view.findViewById(R.id.tvContent);

    // Set title
    tvTitle.setText(article.getTitle());

    // Set content
    tvContent.setText(article.getContent());

    // Format and set date
    try {
        SimpleDateFormat inputFormat = new SimpleDateFormat( pattern: "yyyy-MM-dd'T'HH:mm:ss.SSSSSS'Z'", Locale.US);
        SimpleDateFormat outputFormat = new SimpleDateFormat( pattern: "dd MMMM yyyy", new Locale( language: "id",  country: "ID"));
        Date date = inputFormat.parse(article.getCreatedAt());
        tvDate.setText(outputFormat.format(date));
    } catch (ParseException e) {
        tvDate.setText(article.getCreatedAt());
    }
```

Aplikasi Mobile

5. Settings Page

Halaman ini menampilkan pengaturan alamat yang mempermudah pengguna dalam mengelola informasi alamat mereka. Pengguna dapat melihat daftar alamat yang tersimpan, menambahkan alamat baru melalui formulir yang tersedia, serta memperbarui alamat sesuai kebutuhan untuk keperluan pengiriman atau lainnya.

Pageview
Settings page - Address - Detail address

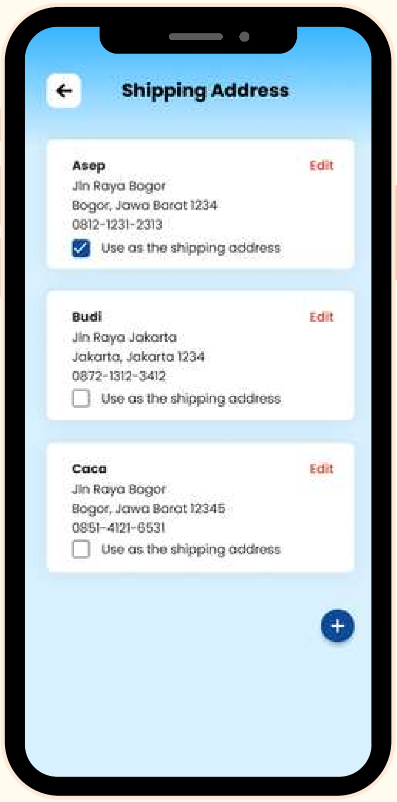


Account

Asep
asep@gmail.com

Address >
My orders >
Wishlist >
Settings >

Signout



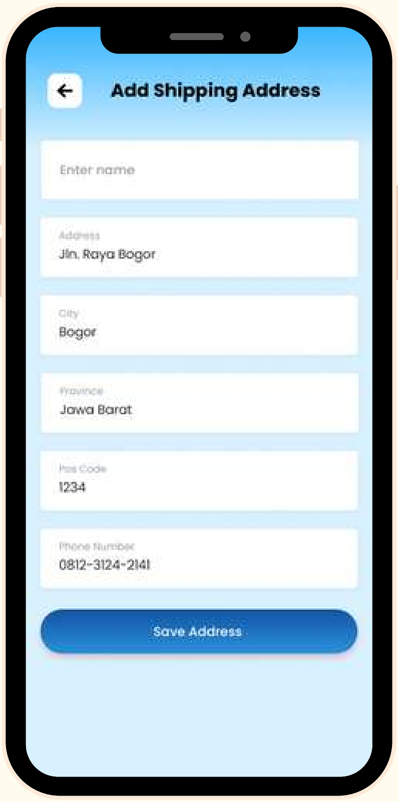
Shipping Address

Asep — Edit
Jln Raya Bogor
Bogor, Jawa Barat 1234
0812-1231-2313
[✓] Use as the shipping address

Budi — Edit
Jln Raya Jakarta
Jakarta, Jakarta 1234
0872-1312-3412
[ ] Use as the shipping address

Caca — Edit
Jln Raya Bogor
Bogor, Jawa Barat 12345
0851-4121-6531
[ ] Use as the shipping address

+



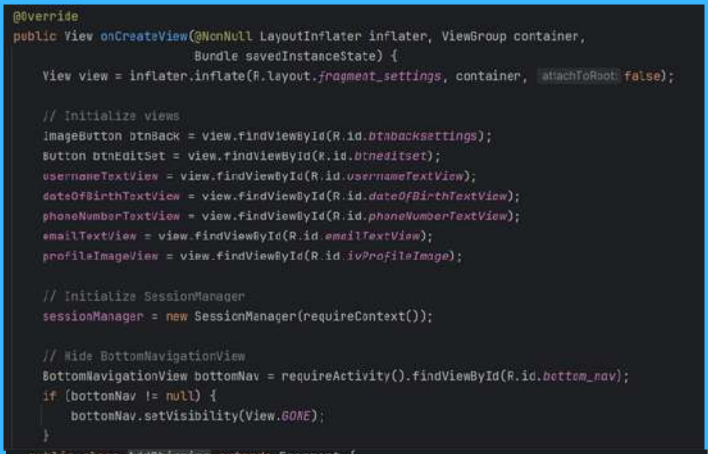
Add Shipping Address

Enter name
Address: Jln. Raya Bogor
City: Bogor
Province: Jawa Barat
Pos Code: 1234
Phone Number: 0812-3124-2141

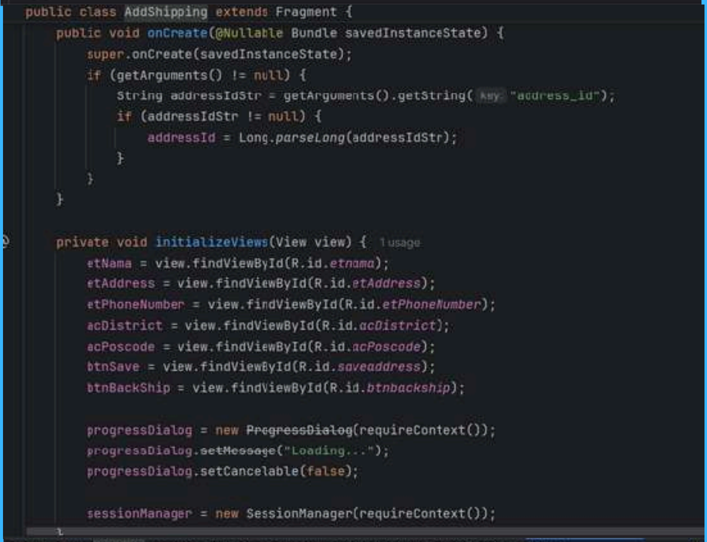
Save Address

Aplikasi Mobile

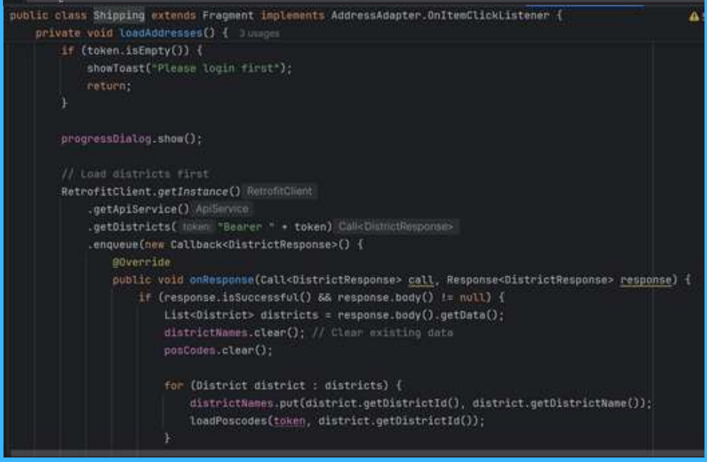
Source Code
Setting Page - Address List - Address Detail



```java
@Override
public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
                         Bundle savedInstanceState) {
    View view = inflater.inflate(R.layout.fragment_settings, container,  attachToRoot: false);

    // Initialize views
    ImageButton btnBack = view.findViewById(R.id.btnbacksettings);
    Button btnEditSet = view.findViewById(R.id.btneditset);
    usernameTextView = view.findViewById(R.id.usernameTextView);
    dateOfBirthTextView = view.findViewById(R.id.dateOfBirthTextView);
    phoneNumberTextView = view.findViewById(R.id.phoneNumberTextView);
    emailTextView = view.findViewById(R.id.emailTextView);
    profileImageView = view.findViewById(R.id.ivProfileImage);

    // Initialize SessionManager
    sessionManager = new SessionManager(requireContext());

    // Hide BottomNavigationView
    BottomNavigationView bottomNav = requireActivity().findViewById(R.id.bottom_nav);
    if (bottomNav != null) {
        bottomNav.setVisibility(View.GONE);
    }
```



```java
public class AddShipping extends Fragment {
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        if (getArguments() != null) {
            String addressIdStr = getArguments().getString( key: "address_id");
            if (addressIdStr != null) {
                addressId = Long.parseLong(addressIdStr);
            }
        }
    }

    private void initializeViews(View view) {  1 usage
        etNama = view.findViewById(R.id.etnama);
        etAddress = view.findViewById(R.id.etAddress);
        etPhoneNumber = view.findViewById(R.id.etPhoneNumber);
        acDistrict = view.findViewById(R.id.acDistrict);
        acPoscode = view.findViewById(R.id.acPoscode);
        btnSave = view.findViewById(R.id.saveaddress);
        btnBackShip = view.findViewById(R.id.btnbackship);

        progressDialog = new ProgressDialog(requireContext());
        progressDialog.setMessage("Loading...");
        progressDialog.setCancelable(false);

        sessionManager = new SessionManager(requireContext());
    }
```



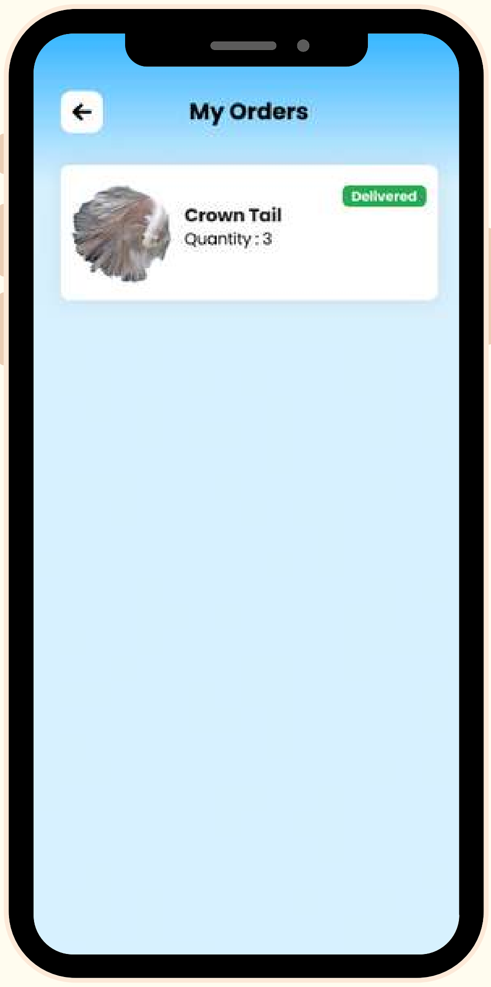
```java
public class Shipping extends Fragment implements AddressAdapter.OnItemClickListener {
    private void loadAddresses() {  3 usages
        if (token.isEmpty()) {
            showToast("Please login first");
            return;
        }

        progressDialog.show();

        // Load districts first
        RetrofitClient.getInstance() RetrofitClient
            .getApiService() ApiService
            .getDistricts( token: "Bearer " + token) Call<DistrictResponse>
            .enqueue(new Callback<DistrictResponse>() {
                @Override
                public void onResponse(Call<DistrictResponse> call, Response<DistrictResponse> response) {
                    if (response.isSuccessful() && response.body() != null) {
                        List<District> districts = response.body().getData();
                        districtNames.clear(); // Clear existing data
                        posCodes.clear();

                        for (District district : districts) {
                            districtNames.put(district.getDistrictId(), district.getDistrictName());
                            loadPoscodes(token, district.getDistrictId());
                        }
```

Aplikasi Mobile

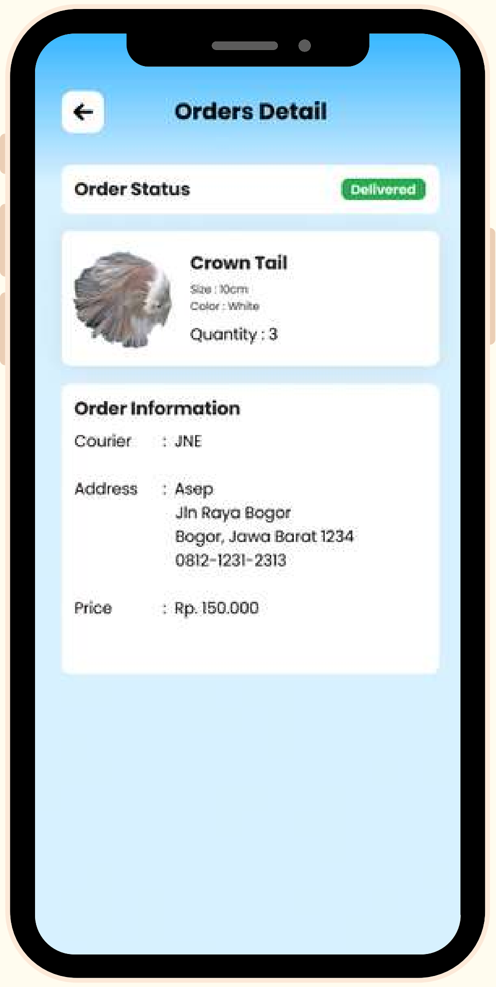
6. Orders Page

Halaman ini menampilkan daftar pesanan yang telah dilakukan oleh pengguna. Pengguna dapat melihat detail dari masing-masing pesanan, termasuk rincian produk yang dipesan, deskripsi atau keterangan produk, serta status terkini dari pesanan tersebut. Fitur ini memudahkan pengguna untuk memantau pesanan mereka dengan jelas dan akurat.

Pageview
Order page - Detail order


My Orders

Crown Tail
Quantity : 3
Delivered


Orders Detail

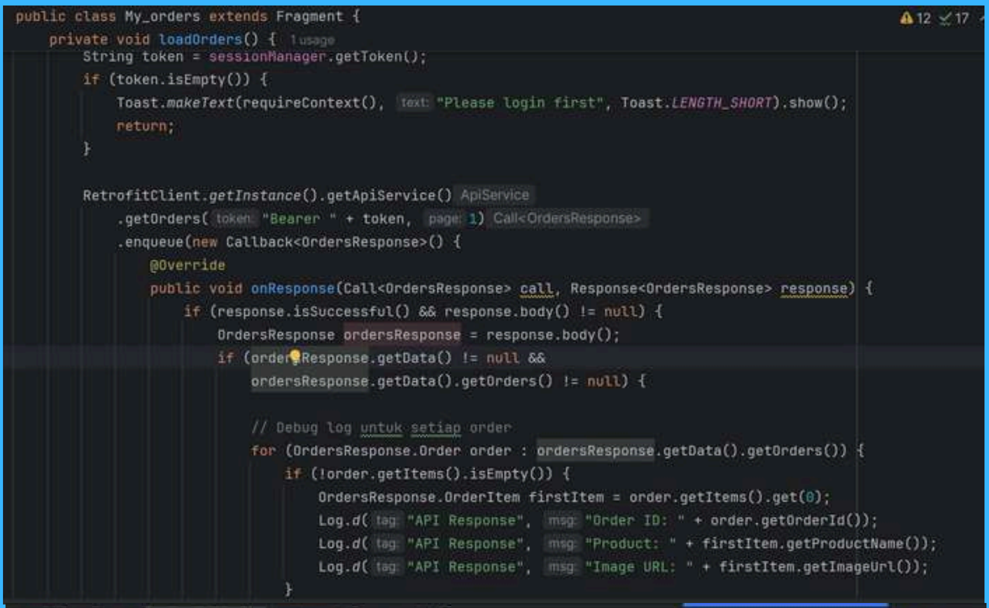
Order Status — Delivered

Crown Tail
Size : 10cm
Color : White
Quantity : 3

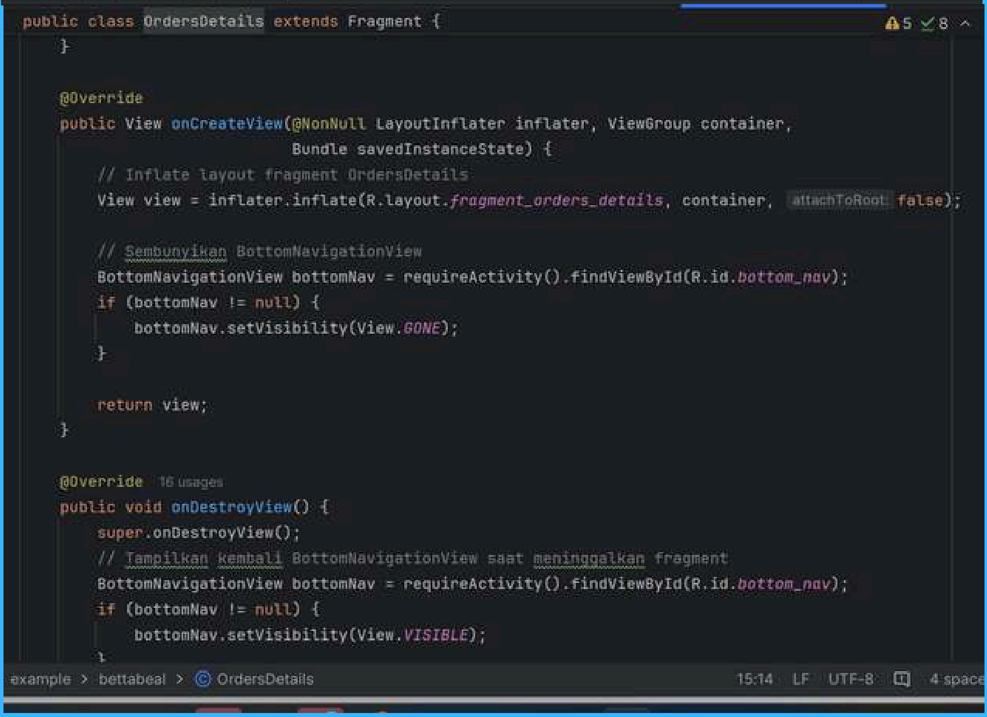
Order Information
Courier : JNE
Address : Asep
Jln Raya Bogor
Bogor, Jawa Barat 1234
0812-1231-2313
Price : Rp. 150.000

Aplikasi Mobile

Source Code
Order page - Detail order



```java
public class My_orders extends Fragment {                                                    ▲12 ✓17
    private void loadOrders() {  1 usage
        String token = sessionManager.getToken();
        if (token.isEmpty()) {
            Toast.makeText(requireContext(),  text: "Please login first", Toast.LENGTH_SHORT).show();
            return;
        }

        RetrofitClient.getInstance().getApiService() ApiService
            .getOrders( token: "Bearer " + token,  page: 1) Call<OrdersResponse>
            .enqueue(new Callback<OrdersResponse>() {
                @Override
                public void onResponse(Call<OrdersResponse> call, Response<OrdersResponse> response) {
                    if (response.isSuccessful() && response.body() != null) {
                        OrdersResponse ordersResponse = response.body();
                        if (ordersResponse.getData() != null &&
                            ordersResponse.getData().getOrders() != null) {

                            // Debug log untuk setiap order
                            for (OrdersResponse.Order order : ordersResponse.getData().getOrders()) {
                                if (!order.getItems().isEmpty()) {
                                    OrdersResponse.OrderItem firstItem = order.getItems().get(0);
                                    Log.d( tag: "API Response",  msg: "Order ID: " + order.getOrderId());
                                    Log.d( tag: "API Response",  msg: "Product: " + firstItem.getProductName());
                                    Log.d( tag: "API Response",  msg: "Image URL: " + firstItem.getImageUrl());
                                }
```



```java
public class OrdersDetails extends Fragment {                                              ▲5 ✓8 ^
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate layout fragment OrdersDetails
        View view = inflater.inflate(R.layout.fragment_orders_details, container,  attachToRoot: false);

        // Sembunyikan BottomNavigationView
        BottomNavigationView bottomNav = requireActivity().findViewById(R.id.bottom_nav);
        if (bottomNav != null) {
            bottomNav.setVisibility(View.GONE);
        }

        return view;
    }

    @Override  16 usages
    public void onDestroyView() {
        super.onDestroyView();
        // Tampilkan kembali BottomNavigationView saat meninggalkan fragment
        BottomNavigationView bottomNav = requireActivity().findViewById(R.id.bottom_nav);
        if (bottomNav != null) {
            bottomNav.setVisibility(View.VISIBLE);
        }
```

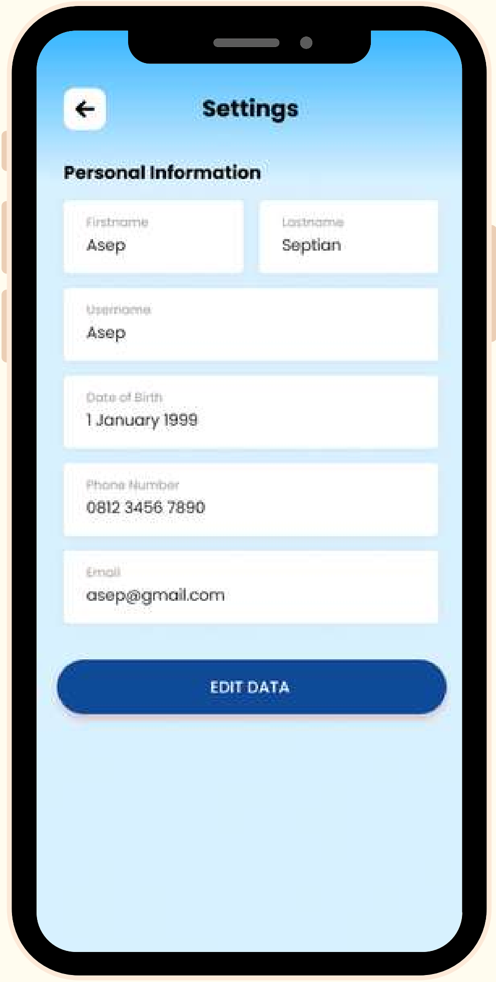
example > bettabeal > OrdersDetails
15:14 LF UTF-8 4 space

Aplikasi Mobile

7. Personal Settings

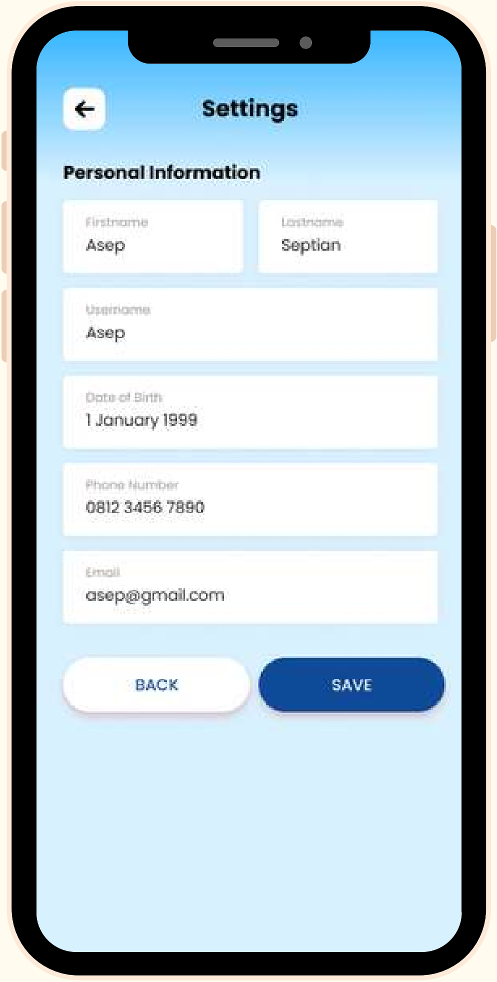
Halaman ini memungkinkan pengguna untuk melihat dan mengedit informasi pribadi mereka, seperti nama, username, tanggal lahir, nomor telepon, dan email. Fitur ini dirancang untuk mempermudah pengguna dalam memperbarui data sesuai kebutuhan.

Pageview
Add Setting - Update Setting


Settings

Personal Information

Firstname: Asep
Lastname: Septian
Username: Asep
Date of Birth: 1 January 1999
Phone Number: 0812 3456 7890
Email: asep@gmail.com

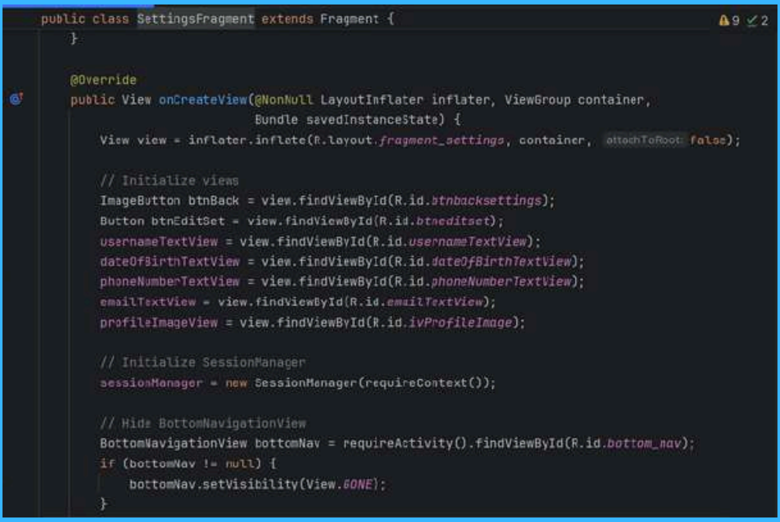

Settings

Personal Information

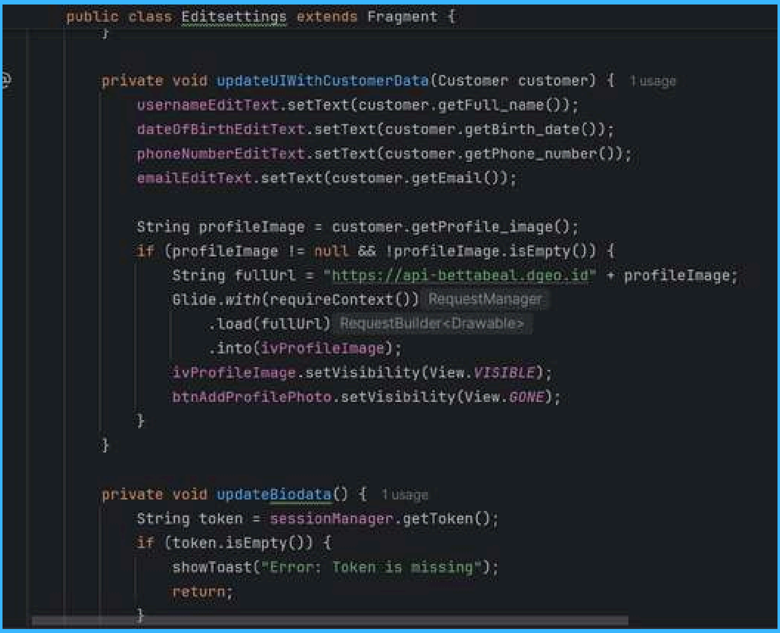
Firstname: Asep
Lastname: Septian
Username: Asep
Date of Birth: 1 January 1999
Phone Number: 0812 3456 7890
Email: asep@gmail.com

Aplikasi Mobile

Source Code
Add Setting - Update Setting

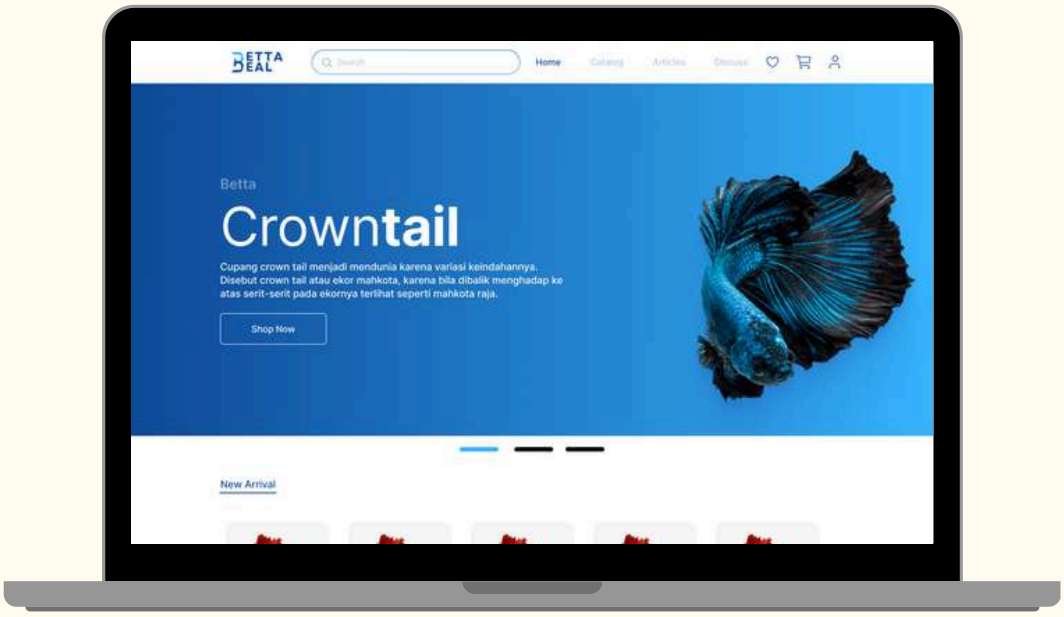


```java
public class SettingsFragment extends Fragment {
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.fragment_settings, container,  attachToRoot: false);

        // Initialize views
        ImageButton btnBack = view.findViewById(R.id.btnbacksettings);
        Button btnEditSet = view.findViewById(R.id.btneditset);
        usernameTextView = view.findViewById(R.id.usernameTextView);
        dateOfBirthTextView = view.findViewById(R.id.dateOfBirthTextView);
        phoneNumberTextView = view.findViewById(R.id.phoneNumberTextView);
        emailTextView = view.findViewById(R.id.emailTextView);
        profileImageView = view.findViewById(R.id.ivProfileImage);

        // Initialize SessionManager
        sessionManager = new SessionManager(requireContext());

        // Hide BottomNavigationView
        BottomNavigationView bottomNav = requireActivity().findViewById(R.id.bottom_nav);
        if (bottomNav != null) {
            bottomNav.setVisibility(View.GONE);
        }
```



```java
public class Editsettings extends Fragment {
    }

    private void updateUIWithCustomerData(Customer customer) {  1 usage
        usernameEditText.setText(customer.getFull_name());
        dateOfBirthEditText.setText(customer.getBirth_date());
        phoneNumberEditText.setText(customer.getPhone_number());
        emailEditText.setText(customer.getEmail());

        String profileImage = customer.getProfile_image();
        if (profileImage != null && !profileImage.isEmpty()) {
            String fullUrl = "https://api-bettabeal.dgeo.id" + profileImage;
            Glide.with(requireContext()) RequestManager
                    .load(fullUrl) RequestBuilder<Drawable>
                    .into(ivProfileImage);
            ivProfileImage.setVisibility(View.VISIBLE);
            btnAddProfilePhoto.setVisibility(View.GONE);
        }
    }

    private void updateBiodata() {  1 usage
        String token = sessionManager.getToken();
        if (token.isEmpty()) {
            showToast("Error: Token is missing");
            return;
        }
```

WEBSITE *Customer*



Betta

Crowntail

Cupang crown tail menjadi mendunia karena variasi keindahannya. Disebut crown tail atau ekor mahkota, karena bila dibalik menghadap ke atas serit-serit pada ekornya terlihat seperti mahkota raja.

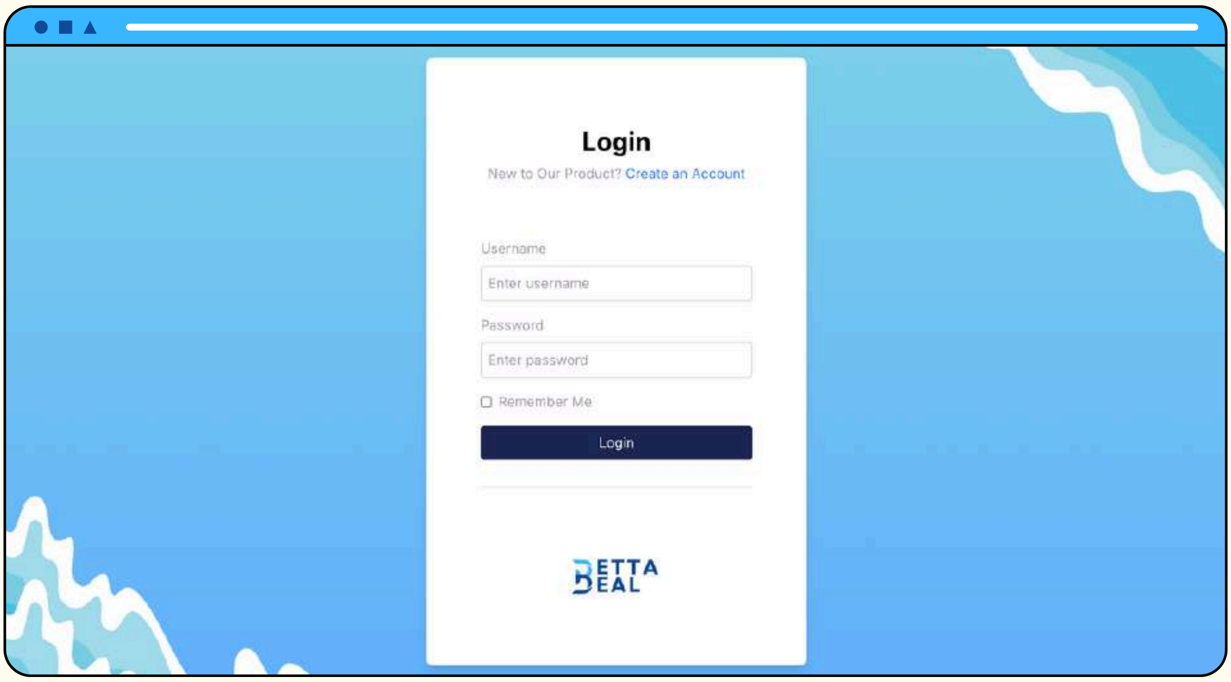
Bettabeal adalah website e-commerce yang dirancang khusus untuk customer pecinta ikan cupang. Website ini menawarkan fitur jual beli ikan cupang, perlengkapan ikan cupang, dan pakan, dengan kemudahan transaksi langsung melalui platform. Selain itu, Bettabeal menyediakan berbagai artikel informatif seputar ikan cupang yang dilengkapi dengan fitur komentar, sehingga pengguna dapat berdiskusi, berbagi pendapat, dan memberikan masukan pada setiap artikel. Dengan antarmuka yang ramah pengguna dan fitur interaktif, Bettabeal menjadi tempat yang ideal bagi customer untuk memenuhi kebutuhan sekaligus memperluas pengetahuan mereka tentang ikan cupang.

Website Customer

1. Login Page

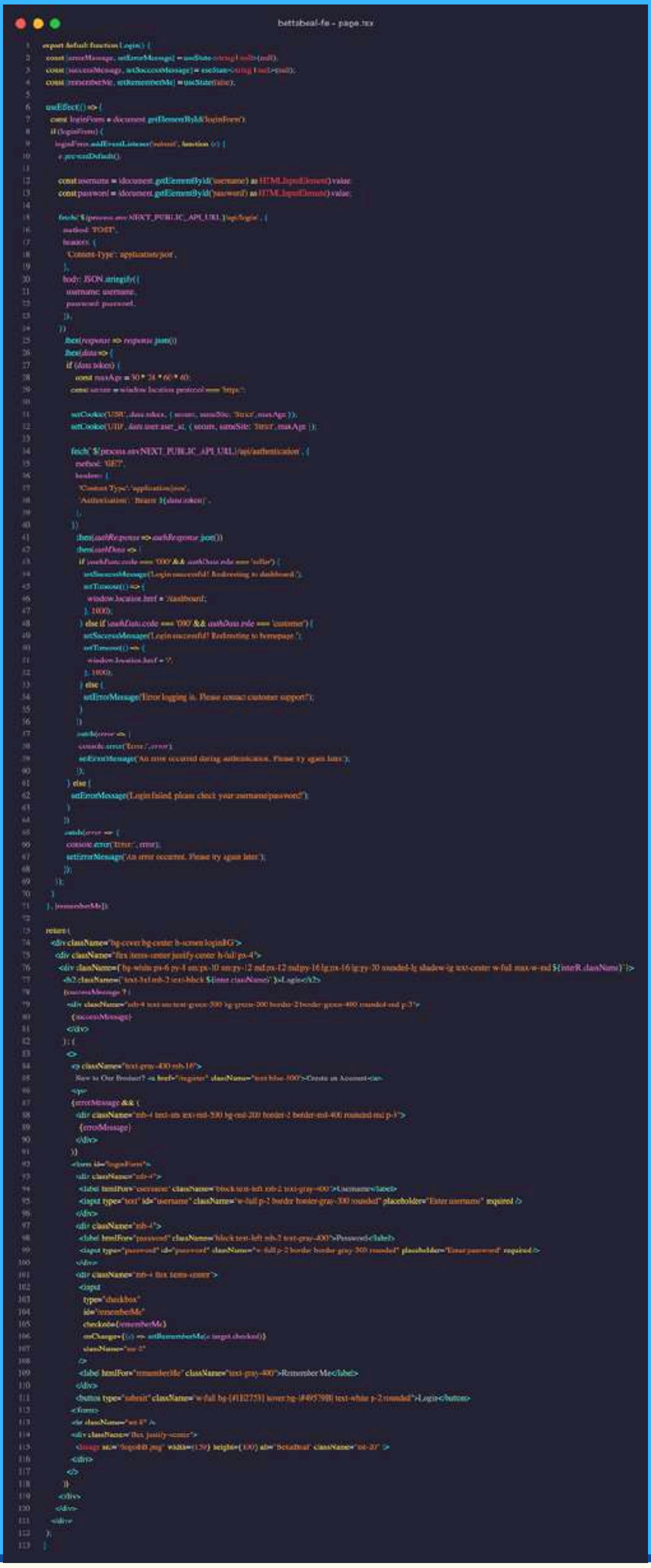
Halaman login ini memungkinkan pengguna untuk masuk ke website dengan memasukkan username dan password yang telah terdaftar. Pengguna yang sudah memiliki akun dapat mengisi kolom username dan password untuk mengakses aplikasi. Jika belum memiliki akun, pengguna dapat diarahkan ke halaman pendaftaran untuk membuat akun baru.

Pageview
Login page



Website Customer

Source Code



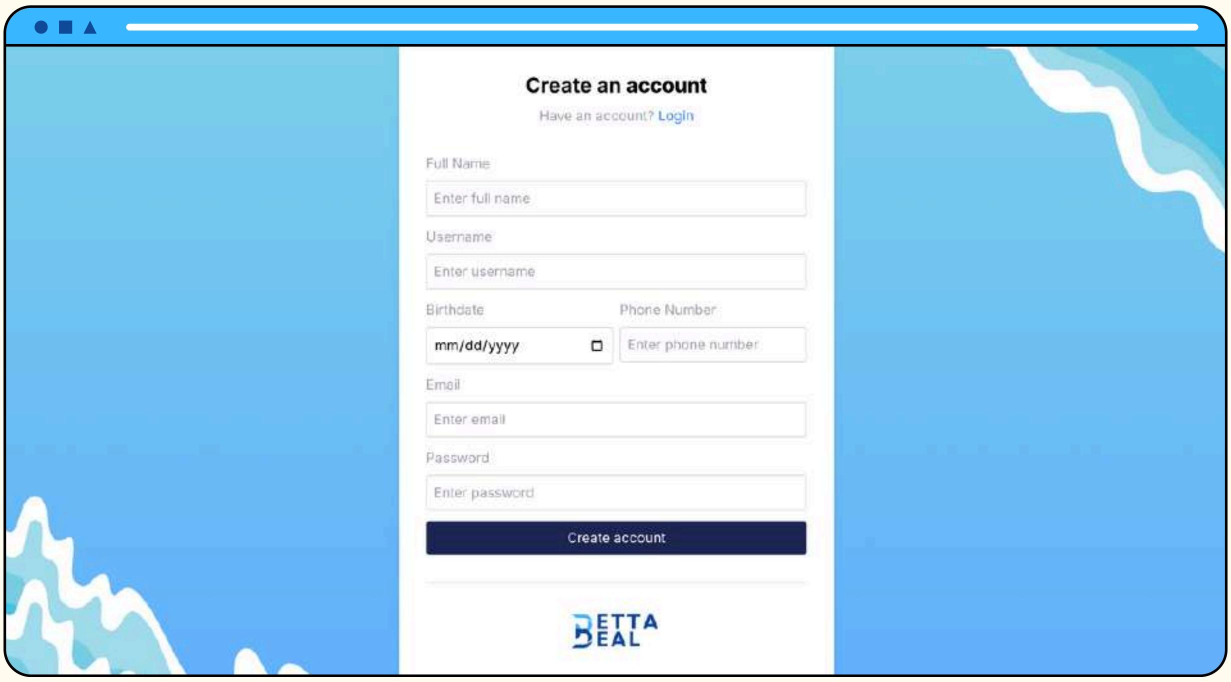
```
export default function Login() {
 const [errorMessage, setErrorMessage] = useState<string | null>(null);
 const [successMessage, setSuccessMessage] = useState<string | null>(null);
 const [rememberMe, setRememberMe] = useState(false);

 useEffect(() => {
  const loginForm = document.getElementById('loginForm');
  if (loginForm) {
   loginForm.addEventListener('submit', function (e) {
    e.preventDefault();

    const username = (document.getElementById('username') as HTMLInputElement).value;
    const password = (document.getElementById('password') as HTMLInputElement).value;

    fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/login`, {
     method: 'POST',
     headers: {
      'Content-Type': 'application/json',
     },
     body: JSON.stringify({
      username: username,
      password: password,
     }),
    })
     .then(response => response.json())
     .then(data => {
     if (data.token) {
       const maxAge = 30 * 24 * 60 * 60;
      const secure = window.location.protocol === 'https:';

      setCookie('USR', data.token, { secure, sameSite: 'Strict', maxAge });
      setCookie('UID', data.user.user_id, { secure, sameSite: 'Strict', maxAge });

      fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/authentication`, {
       method: 'GET',
       headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${data.token}`,
       },
      })
       .then(authResponse => authResponse.json())
       .then(authData => {
       if (authData.code === '000' && authData.role === 'seller') {
        setSuccessMessage('Login successful! Redirecting to dashboard.');
        setTimeout(() => {
         window.location.href = '/dashboard';
        }, 1000);
       } else if (authData.code === '000' && authData.role === 'customer') {
        setSuccessMessage('Login successful! Redirecting to homepage.');
        setTimeout(() => {
         window.location.href = '/';
        }, 1000);
       } else {
        setErrorMessage('Error logging in. Please contact customer support!');
       }
       })
       .catch(error => {
       console.error('Error:', error);
       setErrorMessage('An error occurred during authentication. Please try again later.');
       });
     } else {
      setErrorMessage('Login failed, please check your username/password!');
     }
     })
     .catch(error => {
     console.error('Error:', error);
     setErrorMessage('An error occurred. Please try again later.');
     });
   });
  }
 }, [rememberMe]);

 return (
  <div className="bg-cover bg-center h-screen loginBG">
   <div className="flex items-center justify-center h-full px-4">
    <div className={`bg-white px-6 py-4 sm:px-10 sm:py-12 md:px-12 md:py-16 lg:px-16 lg:py-20 rounded-lg shadow-lg text-center w-full max-w-md ${interR.className}`}>
     <h2 className={`text-3xl mb-2 text-black ${inter.className}`}>Login</h2>
     {successMessage ? (
      <div className="mb-4 text-sm text-green-700 bg-green-200 border-2 border-green-400 rounded-md p-3">
       {successMessage}
      </div>
     ) : (
      <>
       <p className="text-gray-400 mb-10">
        New to Our Product? <a href="/register" className="text-blue-500">Create an Account</a>
       </p>
       {errorMessage && (
        <div className="mb-4 text-sm text-red-500 bg-red-200 border-2 border-red-400 rounded-md p-3">
         {errorMessage}
        </div>
       )}
       <form id="loginForm">
        <div className="mb-4">
         <label htmlFor="username" className="block text-left mb-2 text-gray-400">Username</label>
         <input type="text" id="username" className="w-full p-2 border border-gray-300 rounded" placeholder="Enter username" required />
        </div>
        <div className="mb-4">
         <label htmlFor="password" className="block text-left mb-2 text-gray-400">Password</label>
         <input type="password" id="password" className="w-full p-2 border border-gray-300 rounded" placeholder="Enter password" required />
        </div>
        <div className="mb-4 flex items-center">
         <input
          type="checkbox"
          id="rememberMe"
          checked={rememberMe}
          onChange={(e) => setRememberMe(e.target.checked)}
          className="mr-2"
         />
         <label htmlFor="rememberMe" className="text-gray-400">Remember Me</label>
        </div>
        <button type="submit" className="w-full bg-[#1E2753] hover:bg-[#495788] text-white p-2 rounded">Login</button>
       </form>
       <hr className="mt-8" />
       <div className="flex justify-center">
        <Image src="/logobb.png" width={150} height={100} alt="BetaBeal" className="mt-20" />
       </div>
      </>
     )}
    </div>
   </div>
  </div>
 );
}
```

Website Customer

2. Signup Page

Halaman sign up ini memungkinkan pengguna untuk mendaftar akun baru dengan mengisi informasi lengkap, termasuk nama lengkap, username, tanggal lahir, nomor handphone, email, dan password. Setelah semua kolom diisi, pengguna dapat membuat akun dan melanjutkan untuk mengakses website.

Pageview
Signup page



Create an account

Have an account? Login

Full Name

Enter full name

Username

Enter username

Birthdate

mm/dd/yyyy

Phone Number

Enter phone number

Email

Enter email

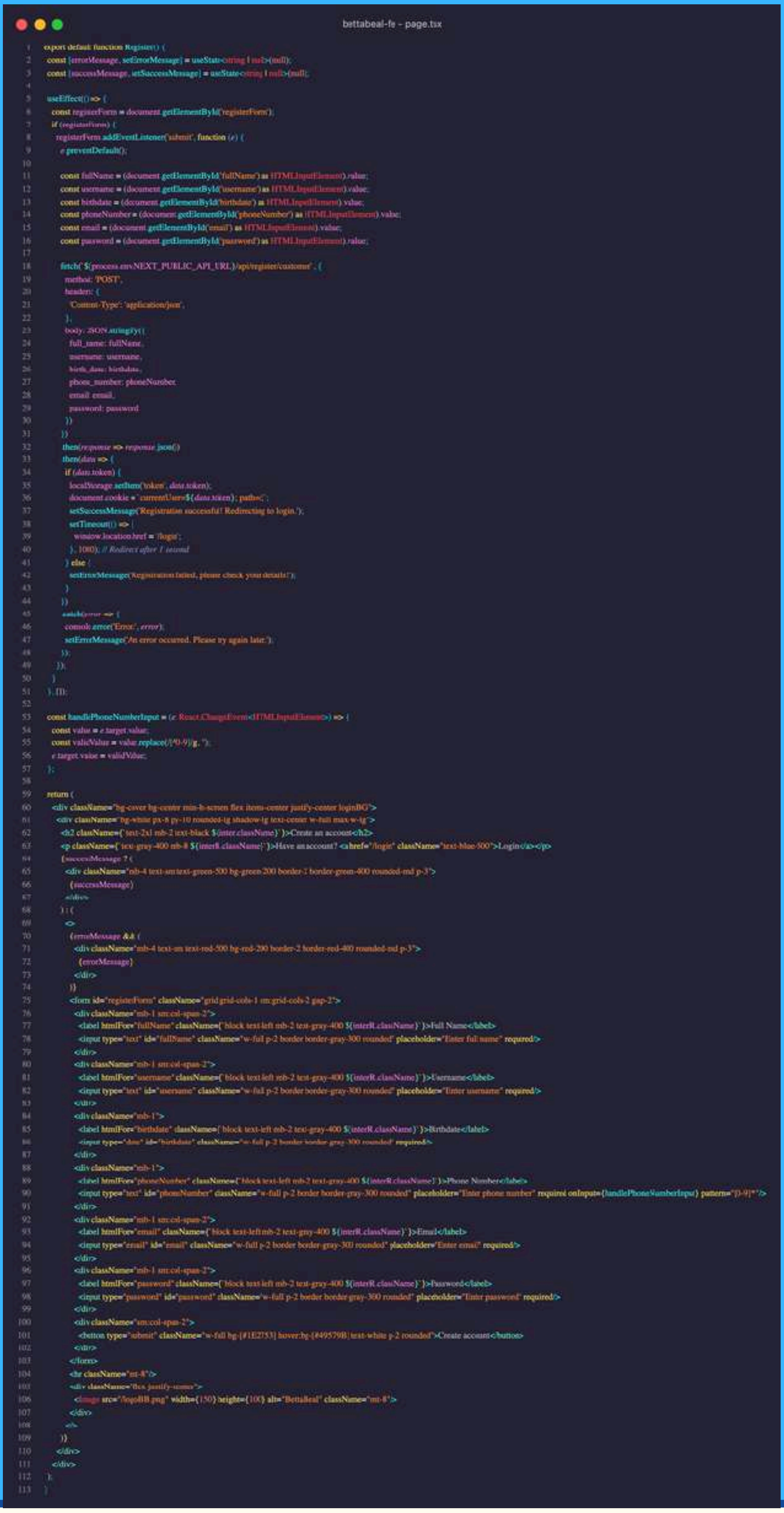
Password

Enter password

Create account

Website Customer

Source Code



bettabeal-fe – page.tsx

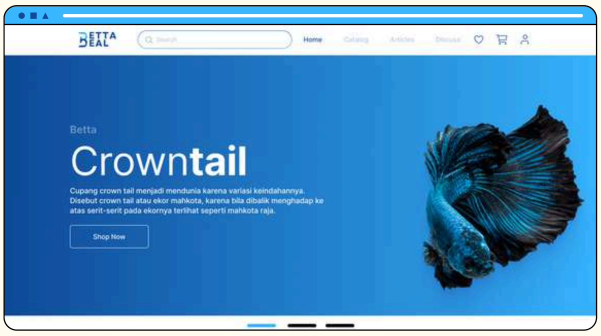
```tsx
export default function Register() {
 const [errorMessage, setErrorMessage] = useState<string | null>(null);
 const [successMessage, setSuccessMessage] = useState<string | null>(null);

 useEffect(() => {
  const registerForm = document.getElementById('registerForm');
  if (registerForm) {
   registerForm.addEventListener('submit', function (e) {
    e.preventDefault();

    const fullName = (document.getElementById('fullName') as HTMLInputElement).value;
    const username = (document.getElementById('username') as HTMLInputElement).value;
    const birthdate = (document.getElementById('birthdate') as HTMLInputElement).value;
    const phoneNumber = (document.getElementById('phoneNumber') as HTMLInputElement).value;
    const email = (document.getElementById('email') as HTMLInputElement).value;
    const password = (document.getElementById('password') as HTMLInputElement).value;

    fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/register/customer`, {
     method: 'POST',
     headers: {
      'Content-Type': 'application/json',
     },
     body: JSON.stringify({
      full_name: fullName,
      username: username,
      birth_date: birthdate,
      phone_number: phoneNumber,
      email: email,
      password: password
     })
    })
    .then(response => response.json())
    .then(data => {
     if (data.token) {
      localStorage.setItem('token', data.token);
      document.cookie = `currentUser=${(data.token}; path=/`;
      setSuccessMessage('Registration successful! Redirecting to login.');
      setTimeout(() => {
       window.location.href = '/login';
      }, 1000); // Redirect after 1 second
     } else {
      setErrorMessage('Registration failed, please check your details!');
     }
    })
    .catch(error => {
     console.error('Error:', error);
     setErrorMessage('An error occurred. Please try again later.');
    });
   });
  }
 }, []);

 const handlePhoneNumberInput = (e: React.ChangeEvent<HTMLInputElement>) => {
  const value = e.target.value;
  const validValue = value.replace(/[^0-9]/g, '');
  e.target.value = validValue;
 };

 return (
  <div className="bg-cover bg-center min-h-screen flex items-center justify-center loginBG">
   <div className="bg-white px-8 py-10 rounded-lg shadow-lg text-center w-full max-w-lg">
    <h2 className={`text-2xl mb-2 text-black ${inter.className}`}>Create an account</h2>
    <p className={`text-gray-400 mb-8 ${interR.className}`}>Have an account? <a href="/login" className="text-blue-500">Login</a></p>
    {successMessage ? (
     <div className="mb-4 text-sm text-green-500 bg-green-200 border-1 border-green-400 rounded-md p-3">
      {successMessage}
     </div>
    ) : (
     <>
      {errorMessage && (
       <div className="mb-4 text-sm text-red-500 bg-red-200 border-2 border-red-400 rounded-md p-3">
        {errorMessage}
       </div>
      )}
      <form id="registerForm" className="grid grid-cols-1 sm:grid-cols-2 gap-2">
       <div className="mb-1 sm:col-span-2">
        <label htmlFor="fullName" className={`block text-left mb-2 text-gray-400 ${interR.className}`}>Full Name</label>
        <input type="text" id="fullName" className="w-full p-2 border border-gray-300 rounded" placeholder="Enter full name" required/>
       </div>
       <div className="mb-1 sm:col-span-2">
        <label htmlFor="username" className={`block text-left mb-2 text-gray-400 ${interR.className}`}>Username</label>
        <input type="text" id="username" className="w-full p-2 border border-gray-300 rounded" placeholder="Enter username" required/>
       </div>
       <div className="mb-1">
        <label htmlFor="birthdate" className={`block text-left mb-2 text-gray-400 ${interR.className}`}>Birthdate</label>
        <input type="date" id="birthdate" className="w-full p-2 border border-gray-300 rounded" required/>
       </div>
       <div className="mb-1">
        <label htmlFor="phoneNumber" className={`block text-left mb-2 text-gray-400 ${interR.className}`}>Phone Number</label>
        <input type="text" id="phoneNumber" className="w-full p-2 border border-gray-300 rounded" placeholder="Enter phone number" required onInput={handlePhoneNumberInput} pattern="[0-9]*"/>
       </div>
       <div className="mb-1 sm:col-span-2">
        <label htmlFor="email" className={`block text-left mb-2 text-gray-400 ${interR.className}`}>Email</label>
        <input type="email" id="email" className="w-full p-2 border border-gray-300 rounded" placeholder="Enter email" required/>
       </div>
       <div className="mb-1 sm:col-span-2">
        <label htmlFor="password" className={`block text-left mb-2 text-gray-400 ${interR.className}`}>Password</label>
        <input type="password" id="password" className="w-full p-2 border border-gray-300 rounded" placeholder="Enter password" required/>
       </div>
       <div className="sm:col-span-2">
        <button type="submit" className="w-full bg-[#1E2753] hover:bg-[#49579B] text-white p-2 rounded">Create account</button>
       </div>
      </form>
      <hr className="mt-8"/>
      <div className="flex justify-center">
       <Image src="/logoBB.png" width={150} height={100} alt="BettaBeal" className="mt-8"/>
      </div>
     </>
    )}
   </div>
  </div>
 );
}
```

Website Customer

3. Homepage

Halaman utama website ini menampilkan berbagai produk unggulan yang tersedia untuk dibeli, serta beberapa artikel terbaru yang memberikan informasi seputar produk dan tips terkait. Pengguna dapat dengan mudah menjelajahi produk-produk yang ditawarkan dan membaca artikel-artikel yang relevan langsung dari halaman utama.

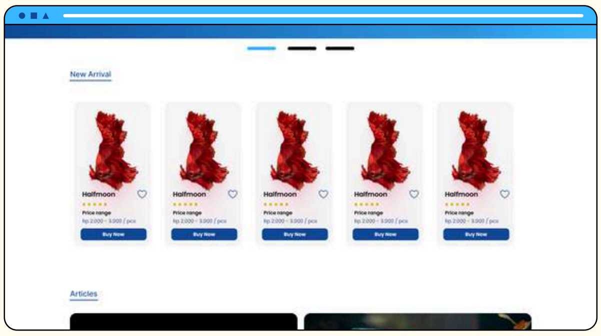
Pageview Homepage

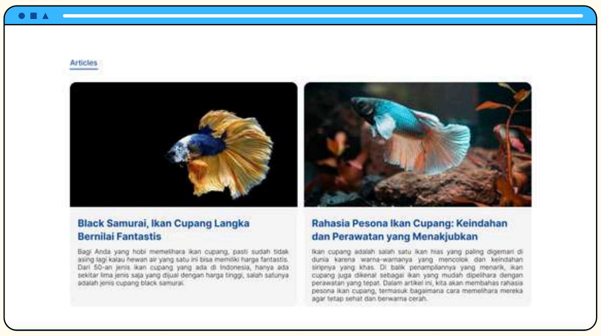


Betta

Crowntail

Cupang crown tail menjadi mendunia karena variasi keindahannya. Disebut crown tail atau ekor mahkota, karena bila dibalik menghadap ke atas serit-serit pada ekornya terlihat seperti mahkota raja.



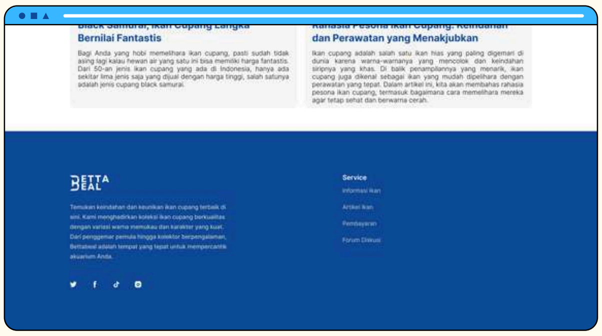


Black Samurai, Ikan Cupang Langka Bernilai Fantastis

Bagi Anda yang hobi memelihara ikan cupang, pasti sudah tidak asing lagi kalau hewan air yang satu ini bisa memiliki harga fantastis. Dari 50-an jenis ikan cupang yang ada di Indonesia, hanya ada sekitar lima jenis saja yang dijual dengan harga tinggi, salah satunya adalah jenis cupang black samurai.

Rahasia Pesona Ikan Cupang: Keindahan dan Perawatan yang Menakjubkan

Ikan cupang adalah salah satu ikan hias yang paling digemari di dunia karena warna-warnanya yang mencolok dan keindahan siripnya yang khas. Di balik penampilannya yang menarik, ikan cupang juga dikenal sebagai ikan yang mudah dipelihara dengan perawatan yang tepat. Dalam artikel ini, kita akan membahas rahasia pesona ikan cupang, termasuk bagaimana cara memelihara mereka agar tetap sehat dan berwarna cerah.



Bernilai Fantastis

Bagi Anda yang hobi memelihara ikan cupang, pasti sudah tidak asing lagi kalau hewan air yang satu ini bisa memiliki harga fantastis. Dari 50-an jenis ikan cupang yang ada di Indonesia, hanya ada sekitar lima jenis saja yang dijual dengan harga tinggi, salah satunya adalah jenis cupang black samurai.

dan Perawatan yang Menakjubkan

Ikan cupang adalah salah satu ikan hias yang paling digemari di dunia karena warna-warnanya yang mencolok dan keindahan siripnya yang khas. Di balik penampilannya yang menarik, ikan cupang juga dikenal sebagai ikan yang mudah dipelihara dengan perawatan yang tepat. Dalam artikel ini, kita akan membahas rahasia pesona ikan cupang, termasuk bagaimana cara memelihara mereka agar tetap sehat dan berwarna cerah.

Temukan keindahan dan keunikan ikan cupang terbaik di sini. Kami menghadirkan koleksi ikan cupang berkualitas dengan variasi warna memukau dan karakter yang kuat. Dari penggemar pemula hingga kolektor berpengalaman, Bettadeal adalah tempat yang tepat untuk mempercantik akuarium Anda.

Website Customer

Source Code

bettabeal-fe - Hero.tsx

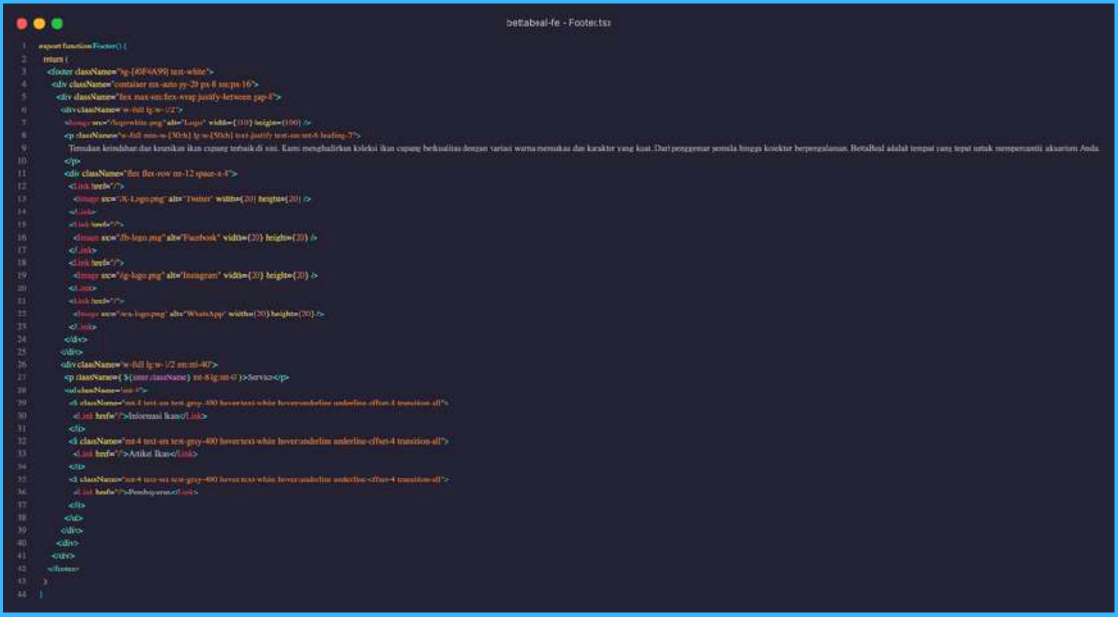
[illegible]

bettabeal-fe - NewArrivals.tsx

[illegible]

bettabeal-fe - ArticleHome.tsx

[illegible]


bettabeal-fe - Footer.tsx

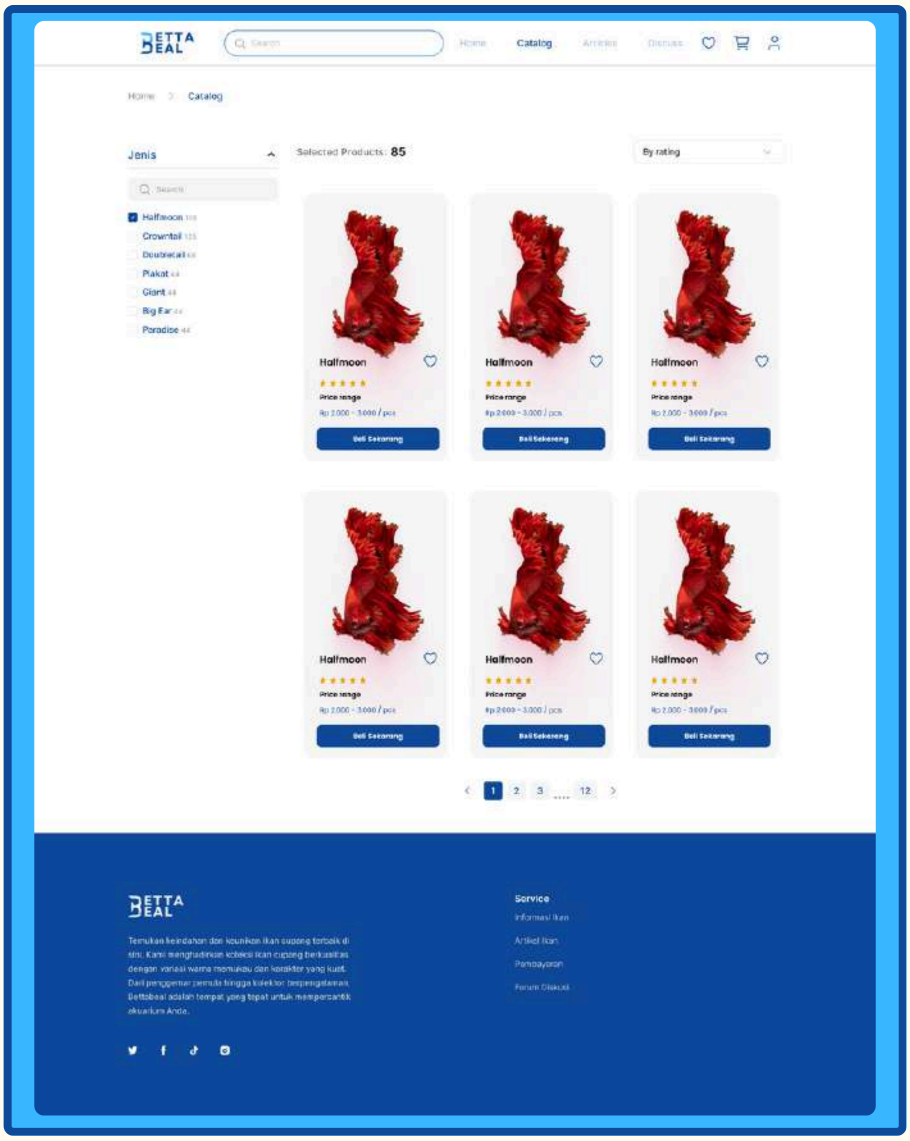
```
export function Footer() {
  return (
    <footer className="bg-[#0F4A99] text-white">
      <div className="container mx-auto py-20 px-8 sm:px-16">
        <div className="flex max-sm:flex-wrap justify-between gap-8">
          <div className="w-full lg:w-1/2">
            <Image src="/logowhite.png" alt="Logo" width={110} height={100} />
            <p className="w-full min-w-[30ch] lg:w-[50ch] text-justify text-sm mt-6 leading-7">
              Temukan keindahan dan keunikan ikan cupang terbaik di sini. Kami menghadirkan koleksi ikan cupang berkualitas dengan variasi warna memukau dan karakter yang kuat. Dari penggemar pemula hingga kolektor berpengalaman, BettaBeal adalah tempat yang tepat untuk mempercantik akuarium Anda.
            </p>
            <div className="flex flex-row mt-12 space-x-4">
              <Link href="/">
                <Image src="/X-Logo.png" alt="Twitter" width={20} height={20} />
              </Link>
              <Link href="/">
                <Image src="/fb-logo.png" alt="Facebook" width={20} height={20} />
              </Link>
              <Link href="/">
                <Image src="/ig-logo.png" alt="Instagram" width={20} height={20} />
              </Link>
              <Link href="/">
                <Image src="/wa-logo.png" alt="WhatsApp" width={20} height={20} />
              </Link>
            </div>
          </div>
          <div className="w-full lg:w-1/2 sm:ml-40">
            <p className={ `${inter.className} mt-8 lg:mt-0` }>Service</p>
            <ul className="mt-4">
              <li className="mt-4 text-sm text-gray-400 hover:text-white hover:underline underline-offset-4 transition-all">
                <Link href="/">Informasi Ikan</Link>
              </li>
              <li className="mt-4 text-sm text-gray-400 hover:text-white hover:underline underline-offset-4 transition-all">
                <Link href="/">Artikel Ikan</Link>
              </li>
              <li className="mt-4 text-sm text-gray-400 hover:text-white hover:underline underline-offset-4 transition-all">
                <Link href="/">Pembayaran</Link>
              </li>
            </ul>
          </div>
        </div>
      </div>
    </footer>
  );
}
```

Website Customer

4. Product Page

Halaman produk ini memungkinkan pengguna untuk melihat berbagai produk yang tersedia, seperti ikan cupang, alat, dan pakan. Pengguna dapat menelusuri produk-produk tersebut dengan mudah, serta memanfaatkan fitur kategori untuk menyaring produk berdasarkan jenis atau kebutuhan, mempermudah pencarian produk yang diinginkan.

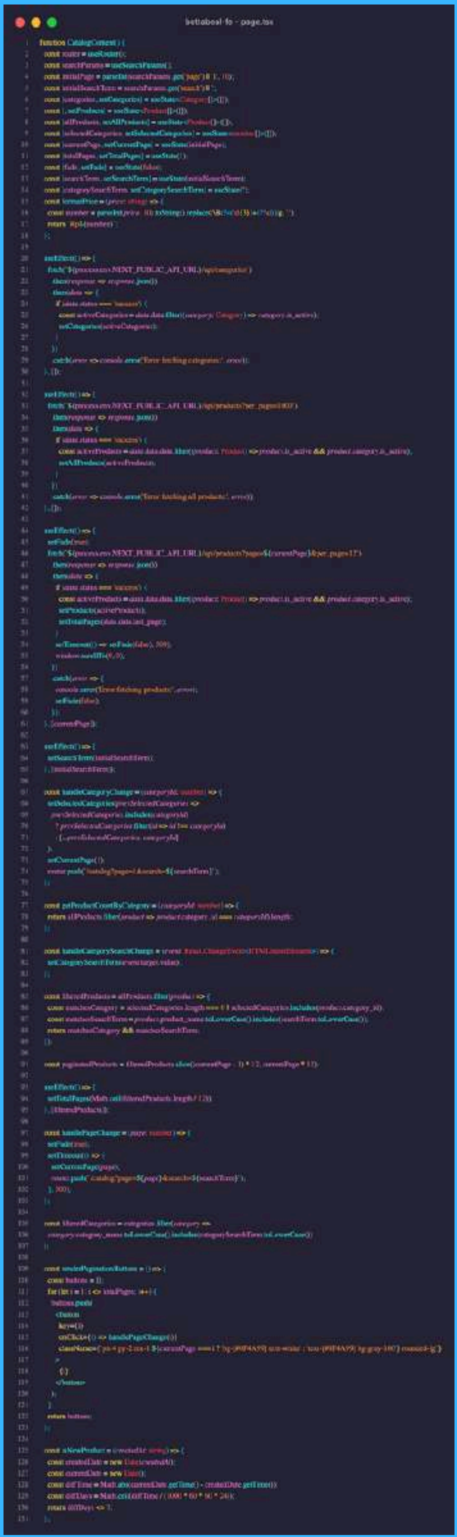
Pageview
Product page



Temukan keindahan dan keunikan ikan cupang terbaik di sini. Kami menghadirkan koleksi ikan cupang berkualitas dengan variasi warna memukau dan karakter yang kuat. Dari penggemar pemula hingga kolektor berpengalaman, Bettabeal adalah tempat yang tepat untuk mempercantik akuarium Anda.

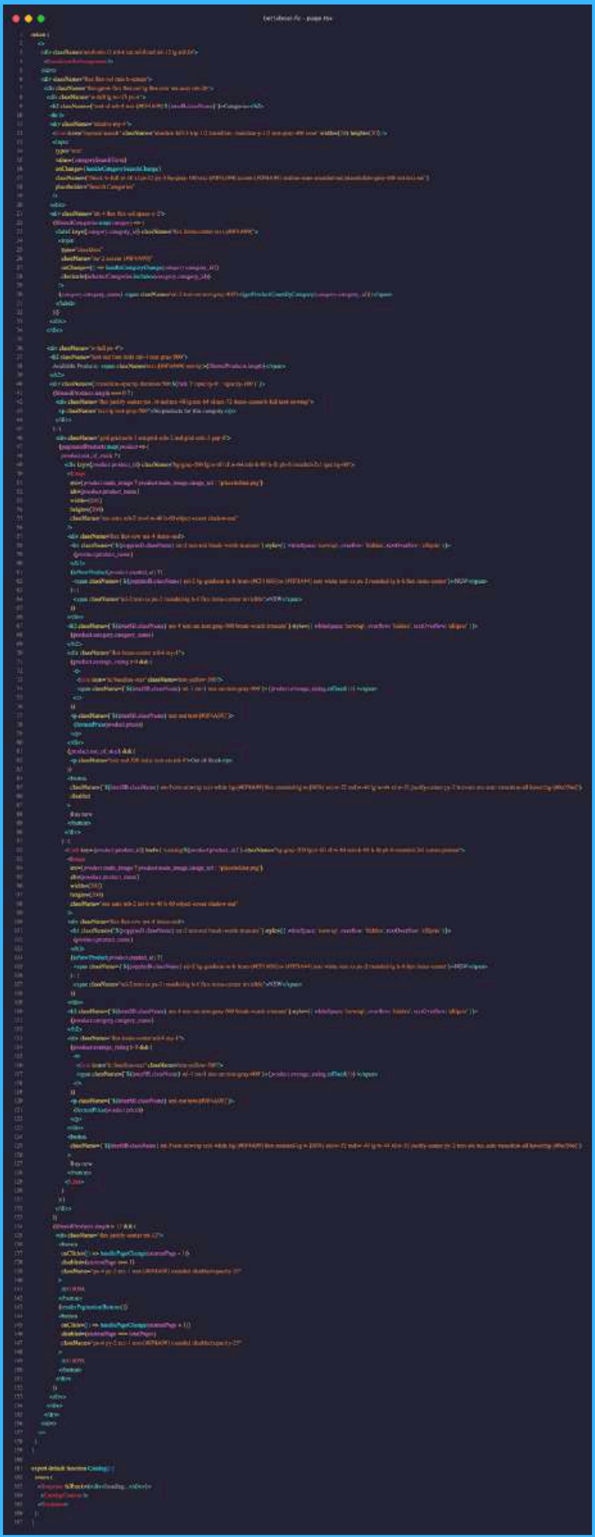
Website Customer

Source Code



bettabeal-fe - page.tsx

```
function CatalogContent() {
  const router = useRouter();
  const searchParams = useSearchParams();
  const initialPage = parseInt(searchParams.get('page') || '1', 10);
  const initialSearchTerm = searchParams.get('search') || '';
  const [categories, setCategories] = useState<Category[]>([]);
  const [, setProducts] = useState<Product[]>([]);
  const [allProducts, setAllProducts] = useState<Product[]>([]);
  const [selectedCategories, setSelectedCategories] = useState<number[]>([]);
  const [currentPage, setCurrentPage] = useState(initialPage);
  const [totalPages, setTotalPages] = useState(1);
  const [fade, setFade] = useState(false);
  const [searchTerm, setSearchTerm] = useState(initialSearchTerm);
  const [categorySearchTerm, setCategorySearchTerm] = useState('');
  const formatPrice = (price: string) => {
   const number = parseInt(price, 10).toString().replace(/\B(?=(\d{3})+(?!\d))/g, '.');
   return `Rp${number}`;
  };

  useEffect(() => {
   fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/categories`)
    .then(response => response.json())
    .then(data => {
     if (data.status === 'success') {
      const activeCategories = data.data.filter((category: Category) => category.is_active);
      setCategories(activeCategories);
     }
    })
    .catch(error => console.error('Error fetching categories:', error));
  }, []);

  useEffect(() => {
   fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/products?per_page=1000`)
    .then(response => response.json())
    .then(data => {
     if (data.status === 'success') {
      const activeProducts = data.data.data.filter((product: Product) => product.is_active && product.category.is_active);
      setAllProducts(activeProducts);
     }
    })
    .catch(error => console.error('Error fetching all products:', error));
  }, []);
[illegible]
```



bettabeal-fe - page.tsx

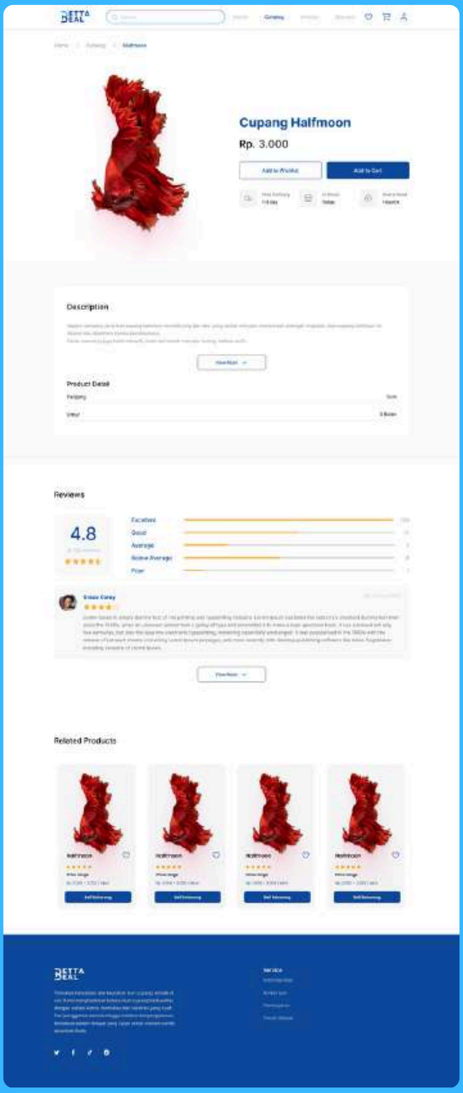
```
[illegible]
```

Website Customer

5. Product Detail Page

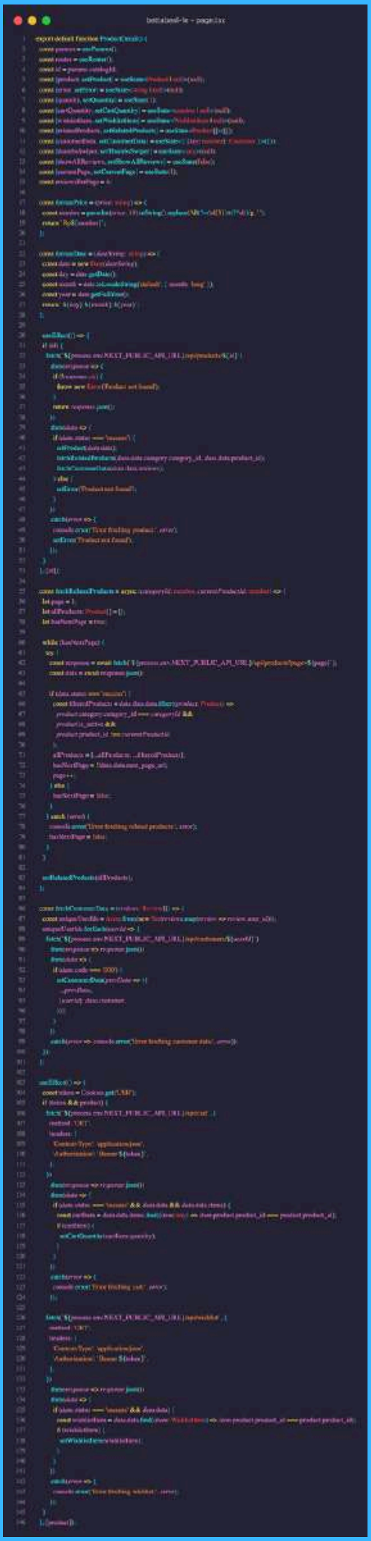
Halaman detail produk ini menampilkan beberapa foto produk dari berbagai sudut, deskripsi lengkap mengenai produk, serta ulasan dari pelanggan yang sudah membeli produk tersebut. Fitur ini memberikan informasi yang jelas dan transparan, membantu pengguna membuat keputusan pembelian yang lebih baik.

Pageview
Product detail page



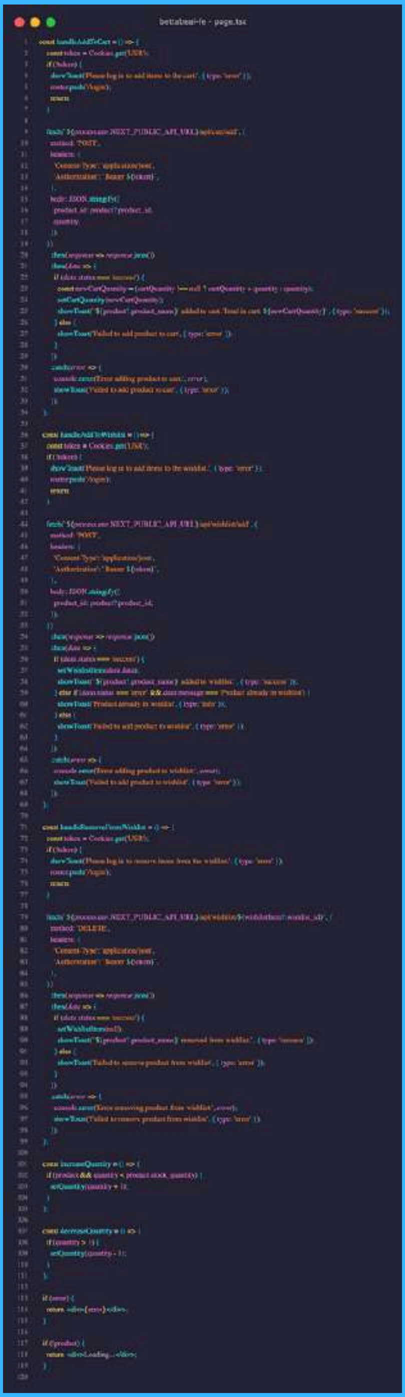
Website Customer

Source Code



betabaezi-fe – page.tsx

```
export default function ProductDetail() {
  const params = useParams();
  const router = useRouter();
  const id = params.catalogId;
  const [product, setProduct] = useState<Product | null>(null);
  const [error, setError] = useState<string | null>(null);
  const [quantity, setQuantity] = useState(1);
  const [cartQuantity, setCartQuantity] = useState<number | null>(null);
  const [wishlistItem, setWishlistItem] = useState<WishlistItem | null>(null);
  const [relatedProducts, setRelatedProducts] = useState<Product[]>([]);
  const [customerData, setCustomerData] = useState<{ [key: number]: Customer }>({});
  const [[illegible]] = useState<[illegible]>(null);
  const [showAllReviews, setShowAllReviews] = useState(false);
  const [currentPage, setCurrentPage] = useState(1);
  const reviewsPerPage = 4;

  const formatPrice = (price: string) => {
    const number = parseInt(price, 10).toString().replace(/\B(?=(\d{3})+(?!\d))/g, ".");
    return `Rp${number}`;
  };

  const formatDate = (dateString: string) => {
    const date = new Date(dateString);
    const day = date.getDate();
    const month = date.toLocaleString("default", { month: "long" });
    const year = date.getFullYear();
    return `${day} ${month} ${year}`;
  };

  useEffect(() => {
    if (id) {
      fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/products/${id}`)
        .then((response) => {
          if (!response.ok) {
            throw new Error("Product not found");
          }
          return response.json();
        })
        .then((data) => {
          if (data.status === "success") {
            setProduct(data.data);
            fetchRelatedProducts(data.data.category.category_id, data.data.product_id);
            fetchCustomerData(data.data.reviews);
          } else {
            setError("Product not found");
          }
        })
        .catch((error) => {
          console.error("Error fetching product:", error);
          setError("Product not found");
        });
    }
  }, [id]);

  const fetchRelatedProducts = async (categoryId: number, currentProductId: number) => {
    let page = 1;
    let allProducts: Product[] = [];
    let hasNextPage = true;

    while (hasNextPage) {
      try {
        const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/products?page=${page}`);
        const data = await response.json();

        if (data.status === "success") {
          const filteredProducts = data.data.data.filter((product: Product) =>
            product.category.category_id === categoryId &&
            product.is_active &&
            product.product_id !== currentProductId
          );
          allProducts = [...allProducts, ...filteredProducts];
          hasNextPage = !!data.data.next_page_url;
          page++;
        } else {
          hasNextPage = false;
        }
      } catch (error) {
        console.error("Error fetching related products:", error);
        hasNextPage = false;
      }
    }

    setRelatedProducts(allProducts);
  };

  const fetchCustomerData = (reviews: Review[]) => {
    const uniqueUserIds = Array.from(new Set(reviews.map(review => review.user_id)));
    uniqueUserIds.forEach(userId => {
      fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/customers/${userId}`)
        .then(response => response.json())
        .then(data => {
          if (data.code === 200) {
            setCustomerData(prevData => ({
              ...prevData,
              [userId]: data.customer
            }));
          }
        })
        .catch(error => console.error("Error fetching customer data:", error));
    });
  };

  useEffect(() => {
    const token = Cookies.get("USER");
    if (token && product) {
      fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/cart`, {
        method: "GET",
        headers: {
          "Content-Type": "application/json",
          "Authorization": `Bearer ${token}`,
        },
      })
        .then((response) => response.json())
        .then((data) => {
          if (data.status === "success" && data.data && data.data.items) {
            const cartItem = data.data.items.find((item: [illegible]) => item.product.product_id === product.product_id);
            if (cartItem) {
              setCartQuantity(cartItem.quantity);
            }
          }
        })
        .catch((error) => {
          console.error("Error fetching cart:", error);
        });

      fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/wishlist`, {
        method: "GET",
        headers: {
          "Content-Type": "application/json",
          "Authorization": `Bearer ${token}`,
        },
      })
        .then((response) => response.json())
        .then((data) => {
          if (data.status === "success" && data.data) {
            const wishlistItem = data.data.find((item: WishlistItem) => item.product.product_id === product.product_id);
            if (wishlistItem) {
              setWishlistItem(wishlistItem);
            }
          }
        })
        .catch((error) => {
          console.error("Error fetching wishlist:", error);
        });
    }
  }, [product]);
```



betabaezi-fe – page.tsx

```
const handleAddToCart = () => {
  const token = Cookies.get("USER");
  if (!token) {
    showToast("Please log in to add items to the cart.", { type: "error" });
    router.push("/login");
    return;
  }

  fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/cart/add`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "Authorization": `Bearer ${token}`,
    },
    body: JSON.stringify({
      product_id: product?.product_id,
      quantity,
    }),
  })
    .then((response) => response.json())
    .then((data) => {
      if (data.status === "success") {
        const newCartQuantity = (cartQuantity !== null ? cartQuantity : 0) + quantity;
        setCartQuantity(newCartQuantity);
        showToast(`${product?.product_name} added to cart. Total in cart: ${newCartQuantity}`, { type: "success" });
      } else {
        showToast("Failed to add product to cart.", { type: "error" });
      }
    })
    .catch((error) => {
      console.error("Error adding product to cart:", error);
      showToast("Failed to add product to cart.", { type: "error" });
    });
};

const handleAddToWishlist = () => {
  const token = Cookies.get("USER");
  if (!token) {
    showToast("Please log in to add items to the wishlist.", { type: "error" });
    router.push("/login");
    return;
  }

  fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/wishlist/add`, {
    method: "POST",
    headers: {
      "Content-Type": "application/json",
      "Authorization": `Bearer ${token}`,
    },
    body: JSON.stringify({
      product_id: product?.product_id,
    }),
  })
    .then((response) => response.json())
    .then((data) => {
      if (data.status === "success") {
        setWishlistItem(data.data);
        showToast(`${product?.product_name} added to wishlist.`, { type: "success" });
      } else if (data.status === "error" && data.message === "Product already in wishlist") {
        showToast("Product already in wishlist.", { type: "info" });
      } else {
        showToast("Failed to add product to wishlist.", { type: "error" });
      }
    })
    .catch((error) => {
      console.error("Error adding product to wishlist:", error);
      showToast("Failed to add product to wishlist.", { type: "error" });
    });
};

const handleRemoveFromWishlist = () => {
  const token = Cookies.get("USER");
  if (!token) {
    showToast("Please log in to remove items from the wishlist.", { type: "error" });
    return;
  }

  fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/wishlist/${wishlistItem?.wishlist_id}`, {
    method: "DELETE",
    headers: {
      "Content-Type": "application/json",
      "Authorization": `Bearer ${token}`,
    },
  })
    .then((response) => response.json())
    .then((data) => {
      if (data.status === "success") {
        setWishlistItem(null);
        showToast(`${product?.product_name} removed from wishlist.`, { type: "success" });
      } else {
        showToast("Failed to remove product from wishlist.", { type: "error" });
      }
    })
    .catch((error) => {
      console.error("Error removing product from wishlist:", error);
      showToast("Failed to remove product from wishlist.", { type: "error" });
    });
};

const increaseQuantity = () => {
  if (product && quantity < product.stock_quantity) {
    setQuantity(quantity + 1);
  }
};

const decreaseQuantity = () => {
  if (quantity > 1) {
    setQuantity(quantity - 1);
  }
};

if (error) {
  return <div>{error}</div>;
}

if (!product) {
  return <div>Loading...</div>;
}
```

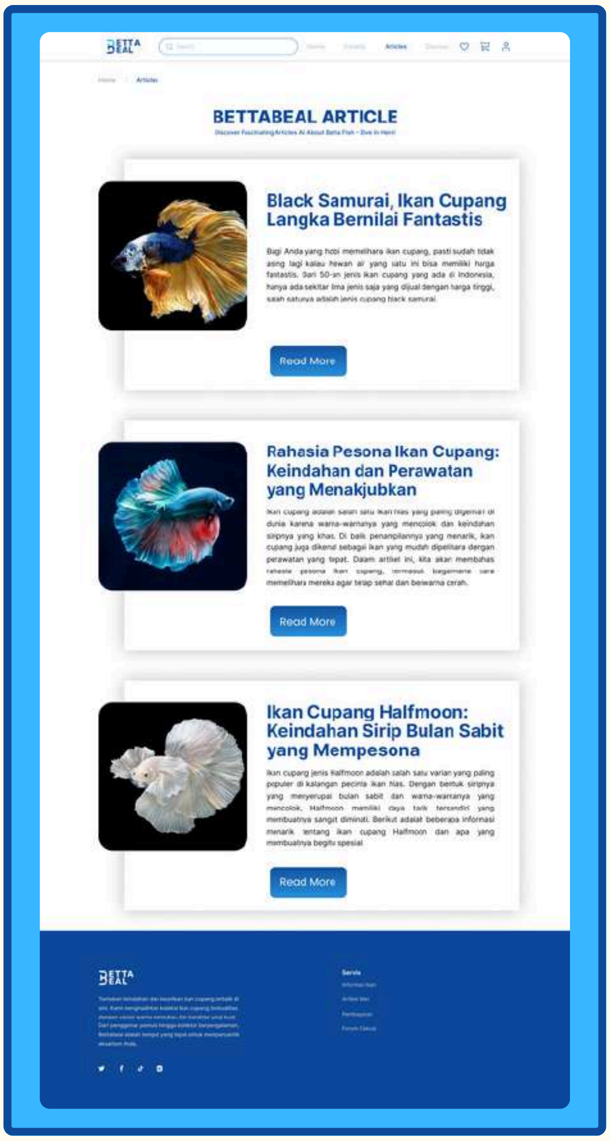
[illegible]

Website Customer

6. Article Page

Halaman artikel ini menampilkan berbagai artikel informatif seputar ikan cupang, mencakup topik seperti jenis-jenis ikan cupang, panduan perawatan, serta tips-tips berguna untuk merawat ikan cupang dengan baik. Pengguna dapat membaca artikel-artikel tersebut untuk menambah wawasan dan pengetahuan tentang hobi memelihara ikan cupang.

Pageview
Article page



BETTABEAL ARTICLE

Discover Fascinating Articles All About Betta Fish - Dive In Here!

Black Samurai, Ikan Cupang Langka Bernilai Fantastis

Bagi Anda yang hobi memelihara ikan cupang, pasti sudah tidak asing lagi kalau hewan air yang satu ini bisa memiliki harga fantastis. Dari 50-an jenis ikan cupang yang ada di Indonesia, hanya ada sekitar lima jenis saja yang dijual dengan harga tinggi, salah satunya adalah jenis cupang black samurai.

Rahasia Pesona Ikan Cupang: Keindahan dan Perawatan yang Menakjubkan

Ikan cupang adalah salah satu ikan hias yang paling digemari di dunia karena warna-warnanya yang mencolok dan keindahan siripnya yang khas. Di balik penampilannya yang menarik, ikan cupang juga dikenal sebagai ikan yang mudah dipelihara dengan perawatan yang tepat. Dalam artikel ini, kita akan membahas rahasia pesona ikan cupang, termasuk bagaimana cara memelihara mereka agar tetap sehat dan berwarna cerah.

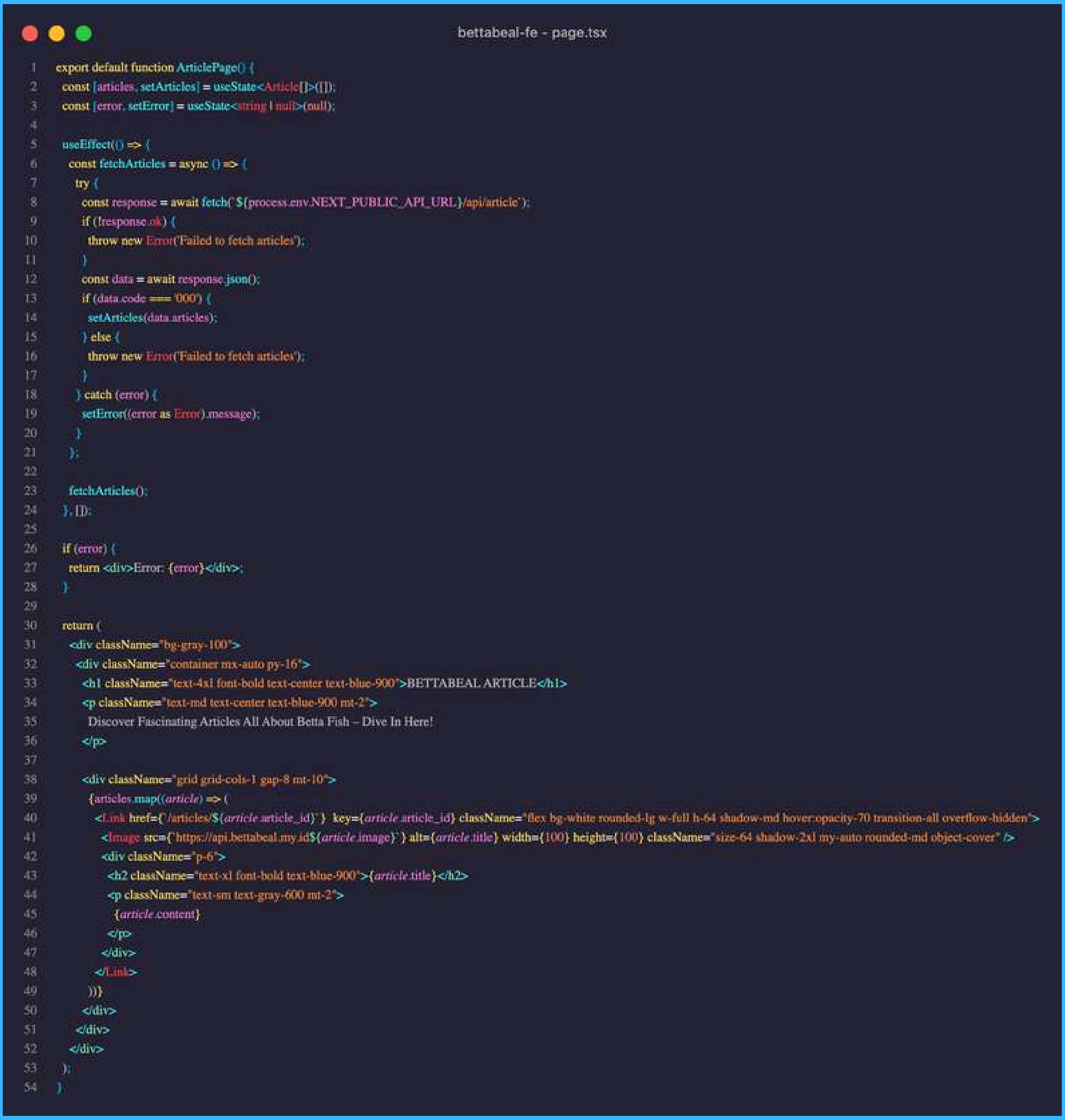
Ikan Cupang Halfmoon: Keindahan Sirip Bulan Sabit yang Mempesona

Ikan cupang jenis Halfmoon adalah salah satu varian yang paling populer di kalangan pecinta ikan hias. Dengan bentuk siripnya yang menyerupai bulan sabit dan warna-warnanya yang mencolok, Halfmoon memiliki daya tarik tersendiri yang membuatnya sangat diminati. Berikut adalah beberapa informasi menarik tentang ikan cupang Halfmoon dan apa yang membuatnya begitu spesial.

[illegible]

Website Customer

Source Code



bettabeal-fe - page.tsx

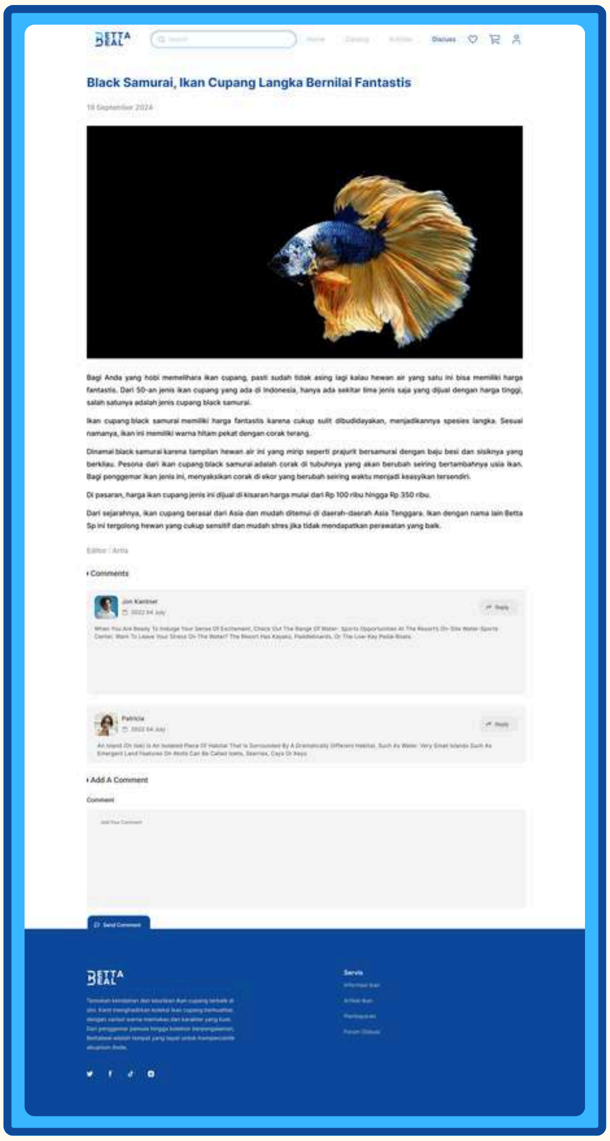
```tsx
export default function ArticlePage() {
  const [articles, setArticles] = useState<Article[]>([]);
  const [error, setError] = useState<string | null>(null);

  useEffect(() => {
    const fetchArticles = async () => {
      try {
        const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/article`);
        if (!response.ok) {
          throw new Error('Failed to fetch articles');
        }
        const data = await response.json();
        if (data.code === '000') {
          setArticles(data.articles);
        } else {
          throw new Error('Failed to fetch articles');
        }
      } catch (error) {
        setError((error as Error).message);
      }
    };

    fetchArticles();
  }, []);

  if (error) {
    return <div>Error: {error}</div>;
  }

  return (
    <div className="bg-gray-100">
      <div className="container mx-auto py-16">
        <h1 className="text-4xl font-bold text-center text-blue-900">BETTABEAL ARTICLE</h1>
        <p className="text-md text-center text-blue-900 mt-2">
          Discover Fascinating Articles All About Betta Fish – Dive In Here!
        </p>

        <div className="grid grid-cols-1 gap-8 mt-10">
          {articles.map((article) => (
            <Link href={`/articles/${article.article_id}`}  key={article.article_id} className="flex bg-white rounded-lg w-full h-64 shadow-md hover:opacity-70 transition-all overflow-hidden">
              <Image src={`https://api.bettabeal.my.id${article.image}`} alt={article.title} width={100} height={100} className="size-64 shadow-2xl my-auto rounded-md object-cover" />
              <div className="p-6">
                <h2 className="text-xl font-bold text-blue-900">{article.title}</h2>
                <p className="text-sm text-gray-600 mt-2">
                  {article.content}
                </p>
              </div>
            </Link>
          ))}
        </div>
      </div>
    </div>
  );
}
```

Website Customer

7. Article Detail Page

Halaman detail artikel ini menampilkan isi artikel lengkap seputar ikan cupang, termasuk informasi penting dan tips yang bermanfaat. Pengguna juga dapat menambahkan komentar pada artikel tersebut untuk berbagi pendapat, bertanya, atau berdiskusi dengan sesama pembaca, memperkaya pengalaman dan interaksi.

**Pageview
Article detail page**



Black Samurai, Ikan Cupang Langka Bernilai Fantastis

19 September 2024

Bagi Anda yang hobi memelihara ikan cupang, pasti sudah tidak asing lagi kalau hewan air yang satu ini bisa memiliki harga fantastis. Dari 50-an jenis ikan cupang yang ada di Indonesia, hanya ada sekitar lima jenis saja yang dijual dengan harga tinggi, salah satunya adalah jenis cupang black samurai.

Ikan cupang black samurai memiliki harga fantastis karena cukup sulit dibudidayakan, menjadikannya spesies langka. Sesuai namanya, ikan ini memiliki warna hitam pekat dengan corak terang.

Dinamai black samurai karena tampilan hewan air ini yang mirip seperti prajurit bersamurai dengan baju besi dan sisiknya yang berkilau. Pesona dari ikan cupang black samurai adalah corak di tubuhnya yang akan berubah seiring bertambahnya usia ikan. Bagi penggemar ikan jenis ini, menyaksikan corak di ekor yang berubah seiring waktu menjadi keasyikan tersendiri.

Di pasaran, harga ikan cupang jenis ini dijual di kisaran harga mulai dari Rp 100 ribu hingga Rp 350 ribu.

Dari sejarahnya, ikan cupang berasal dari Asia dan mudah ditemui di daerah-daerah Asia Tenggara. Ikan dengan nama lain Betta Sp ini tergolong hewan yang cukup sensitif dan mudah stres jika tidak mendapatkan perawatan yang baik.

Editor : Arita

Comments

Jon Kantner
2022 04 July
Reply
When You Are Ready To Indulge Your Sense Of Excitement, Check Out The Range Of Water- Sports Opportunities At The Resort's On-Site Water-Sports Center. Want To Leave Your Stress On The Water? The Resort Has Kayaks, Paddleboards, Or The Low-Key Pedal Boats.

Patricia
2022 04 July
Reply
An Island (Or Isle) Is An Isolated Piece Of Habitat That Is Surrounded By A Dramatically Different Habitat, Such As Water. Very Small Islands Such As Emergent Land Features On Atolls Can Be Called Islets, Skerries, Cays Or Keys.

Add A Comment

Comment

Add Your Comment

Send Comment

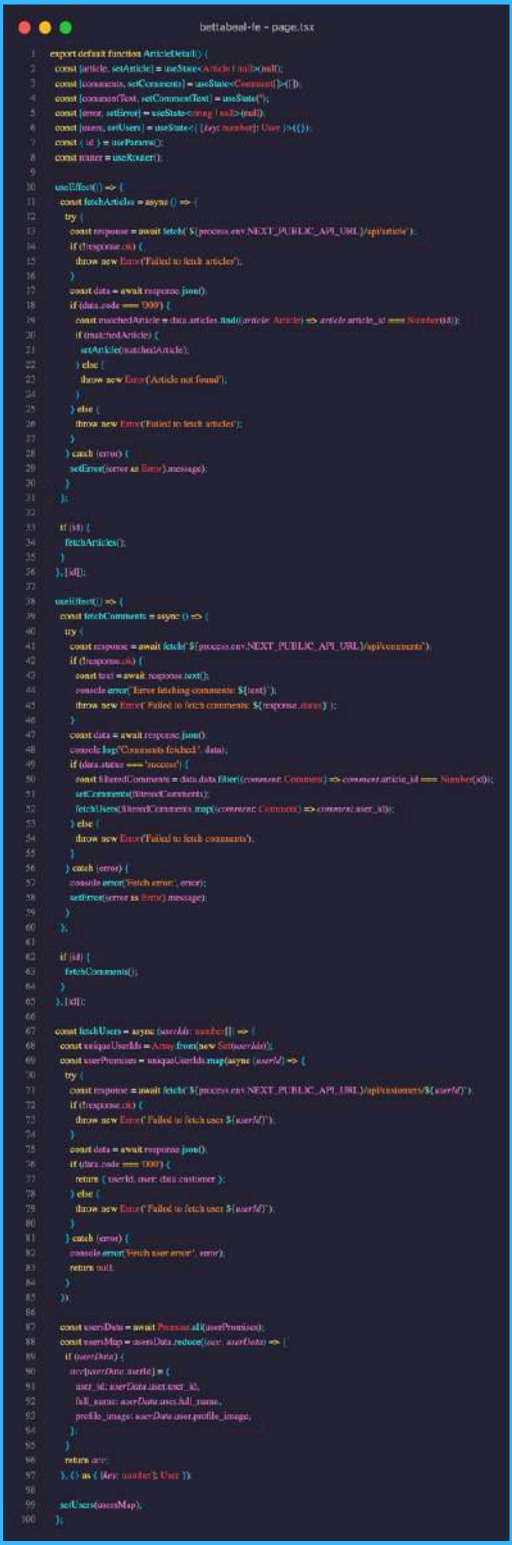
BETTA DEAL

[illegible]

Servis
Informasi Ikan
Artikel Ikan
Pembayaran
Forum Diskusi

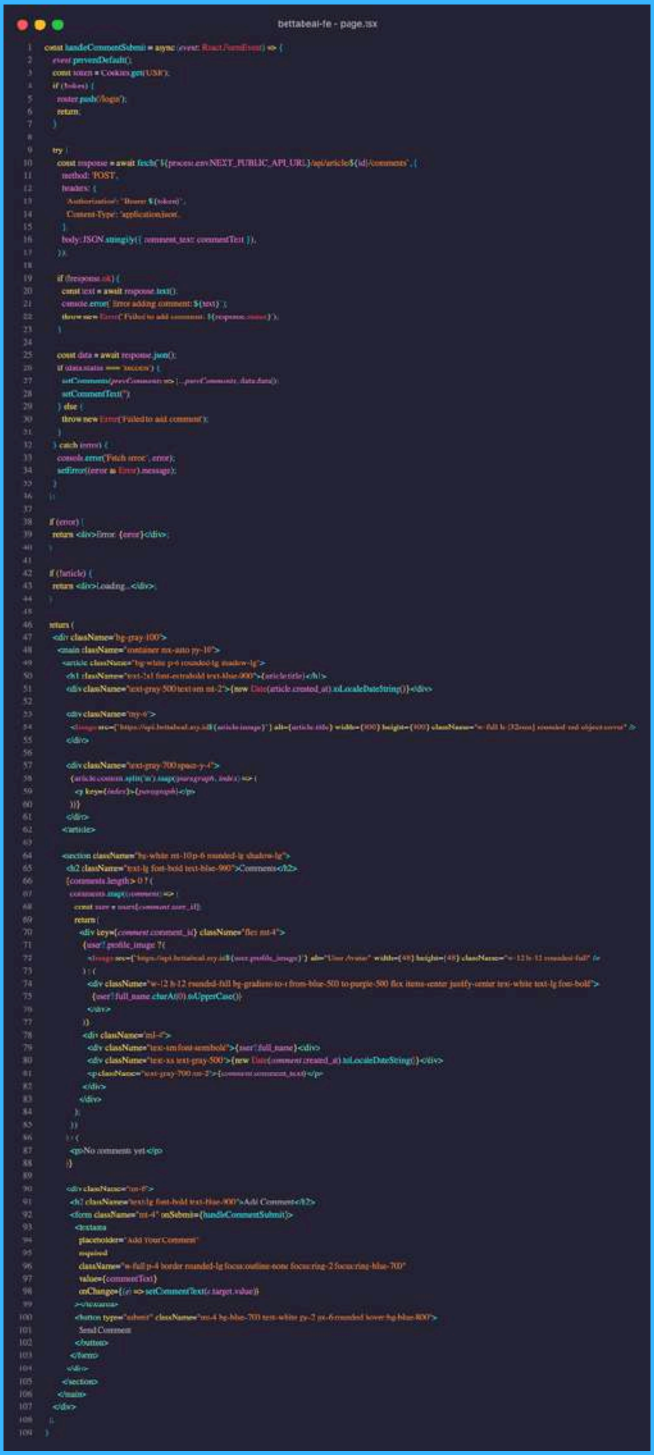
Website Customer

Source Code



bettabeal-fe - page.tsx

```tsx
export default function ArticleDetail() {
 const [article, setArticle] = useState<Article | null>(null);
 const [comments, setComments] = useState<Comment[]>([]);
 const [commentText, setCommentText] = useState('');
 const [error, setError] = useState<string | null>(null);
 const [users, setUsers] = useState<{ [key: number]: User }>({});
 const { id } = useParams();
 const router = useRouter();

 useEffect(() => {
  const fetchArticles = async () => {
   try {
    const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/article`);
    if (!response.ok) {
     throw new Error('Failed to fetch articles');
    }
    const data = await response.json();
    if (data.code === '000') {
     const matchedArticle = data.articles.find((article: Article) => article.article_id === Number(id));
     if (matchedArticle) {
      setArticle(matchedArticle);
     } else {
      throw new Error('Article not found');
     }
    } else {
     throw new Error('Failed to fetch articles');
    }
   } catch (error) {
    setError((error as Error).message);
   }
  };

  if (id) {
   fetchArticles();
  }
 }, [id]);

 useEffect(() => {
  const fetchComments = async () => {
   try {
    const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/comments`);
    if (!response.ok) {
     const text = await response.text();
     console.error(`Error fetching comments: ${text}`);
     throw new Error(`Failed to fetch comments: ${response.status}`);
    }
    const data = await response.json();
    console.log('Comments fetched:', data);
    if (data.status === 'success') {
     const filteredComments = data.data.filter((comment: Comment) => comment.article_id === Number(id));
     setComments(filteredComments);
     fetchUsers(filteredComments.map((comment: Comment) => comment.user_id));
    } else {
     throw new Error('Failed to fetch comments');
    }
   } catch (error) {
    console.error('Fetch error:', error);
    setError((error as Error).message);
   }
  };

  if (id) {
   fetchComments();
  }
 }, [id]);

 const fetchUsers = async (userIds: number[]) => {
  const uniqueUserIds = Array.from(new Set(userIds));
  const userPromises = uniqueUserIds.map(async (userId) => {
   try {
    const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/customers/${userId}`);
    if (!response.ok) {
     throw new Error(`Failed to fetch user ${userId}`);
    }
    const data = await response.json();
    if (data.code === '000') {
     return { userId, user: data.customer };
    } else {
     throw new Error(`Failed to fetch user ${userId}`);
    }
   } catch (error) {
    console.error('Fetch user error:', error);
    return null;
   }
  });

  const usersData = await Promise.all(userPromises);
  const usersMap = usersData.reduce((acc, userData) => {
   if (userData) {
    acc[userData.userId] = {
     user_id: userData.user.user_id,
     full_name: userData.user.full_name,
     profile_image: userData.user.profile_image,
    };
   }
   return acc;
  }, {} as { [key: number]: User });

  setUsers(usersMap);
 };
```



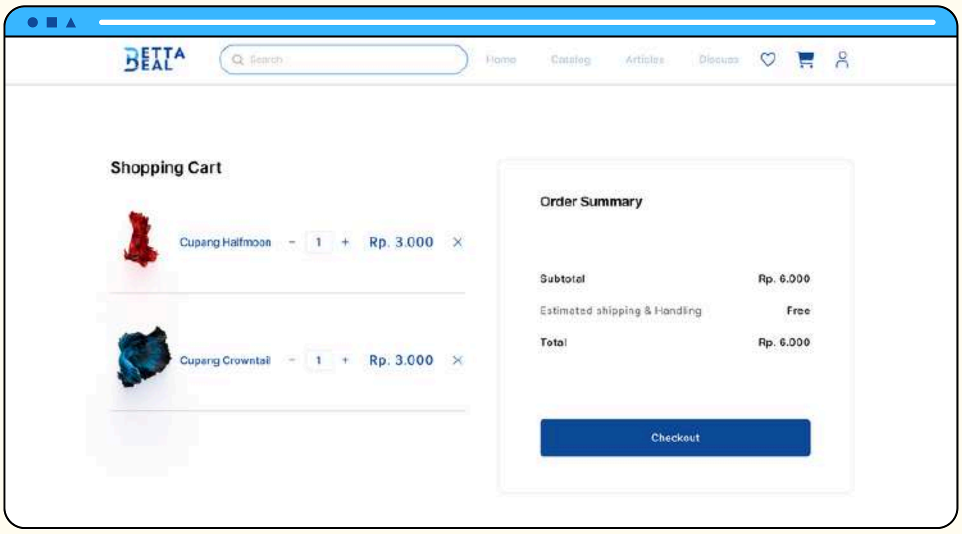
bettabeal-fe - page.tsx

```tsx
const handleCommentSubmit = async (event: React.FormEvent) => {
  event.preventDefault();
  const token = Cookies.get('USR');
  if (!token) {
   router.push('/login');
   return;
  }

  try {
   const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/article/${id}/comments`, {
    method: 'POST',
    headers: {
     'Authorization': `Bearer ${token}`,
     'Content-Type': 'application/json',
    },
    body: JSON.stringify({ comment_text: commentText }),
   });

   if (!response.ok) {
    const text = await response.text();
    console.error(`Error adding comment: ${text}`);
    throw new Error(`Failed to add comment: ${response.status}`);
   }

   const data = await response.json();
   if (data.status === 'success') {
    setComments(prevComments => [...prevComments, data.data]);
    setCommentText('');
   } else {
    throw new Error('Failed to add comment');
   }
  } catch (error) {
   console.error('Fetch error:', error);
   setError((error as Error).message);
  }
 };

 if (error) {
  return <div>Error: {error}</div>;
 }

 if (!article) {
  return <div>Loading...</div>;
 }

 return (
  <div className="bg-gray-100">
   <main className="container mx-auto py-10">
    <article className="bg-white p-6 rounded-lg shadow-lg">
     <h1 className="text-2xl font-extrabold text-blue-900">{article.title}</h1>
     <div className="text-gray-500 text-sm mt-2">{new Date(article.created_at).toLocaleDateString()}</div>

     <div className="my-6">
      <Image src={`https://api.bettabeal.my.id/${article.image}`} alt={article.title} width={800} height={400} className="w-full h-[32rem] rounded-md object-cover" />
     </div>

     <div className="text-gray-700 space-y-4">
      {article.content.split('\n').map((paragraph, index) => (
       <p key={index}>{paragraph}</p>
      ))}
     </div>
    </article>

    <section className="bg-white mt-10 p-6 rounded-lg shadow-lg">
     <h2 className="text-lg font-bold text-blue-900">Comments</h2>
     {comments.length > 0 ? (
      comments.map((comment) => {
       const user = users[comment.user_id];
       return (
        <div key={comment.comment_id} className="flex mt-4">
         {user?.profile_image ? (
          <Image src={`https://api.bettabeal.my.id/${user.profile_image}`} alt="User Avatar" width={48} height={48} className="w-12 h-12 rounded-full" />
         ) : (
          <div className="w-12 h-12 rounded-full bg-gradient-to-r from-blue-500 to-purple-500 flex items-center justify-center text-white text-lg font-bold">
           {user?.full_name.charAt(0).toUpperCase()}
          </div>
         )}
         <div className="ml-4">
          <div className="text-sm font-semibold">{user?.full_name}</div>
          <div className="text-xs text-gray-500">{new Date(comment.created_at).toLocaleDateString()}</div>
          <p className="text-gray-700 mt-2">{comment.comment_text}</p>
         </div>
        </div>
       );
      })
     ) : (
      <p>No comments yet.</p>
     )}

     <div className="mt-6">
      <h2 className="text-lg font-bold text-blue-900">Add Comment</h2>
      <form className="mt-4" onSubmit={handleCommentSubmit}>
       <textarea
        placeholder="Add Your Comment"
        required
        className="w-full p-4 border rounded-lg focus:outline-none focus:ring-2 focus:ring-blue-700"
        value={commentText}
        onChange={(e) => setCommentText(e.target.value)}
       ></textarea>
       <button type="submit" className="mt-4 bg-blue-700 text-white py-2 px-6 rounded hover:bg-blue-800">
        Send Comment
       </button>
      </form>
     </div>
    </section>
   </main>
  </div>
 );
}
```

8. Cart Page

Halaman keranjang ini memungkinkan pengguna untuk melihat produk-produk yang telah mereka tambahkan untuk dibeli. Pengguna dapat melihat rincian produk, seperti nama, jumlah, harga, dan total biaya. Selain itu, pengguna dapat mengedit jumlah produk, menghapus item, atau melanjutkan ke proses checkout untuk menyelesaikan pembelian.

Pageview
Cart page

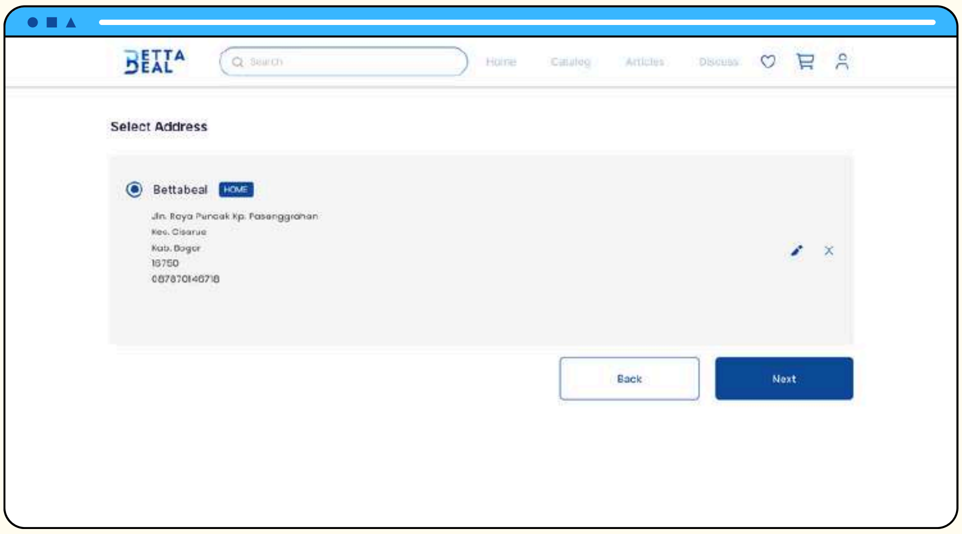


Shopping Cart

Cupang Halfmoon	- 1 +	Rp. 3.000	×
Cupang Crowntail	- 1 +	Rp. 3.000	×

Order Summary

Subtotal	Rp. 6.000
Estimated shipping & Handling	Free
Total	Rp. 6.000



Select Address

Bettabeal HOME
Jln. Raya Puncak Kp. Pasanggrahan
Kec. Cisarua
Kab. Bogor
16750
087870146718

Website Customer

Source Code

Cart

[illegible]

[illegible]

Website Customer

Shipping address

Source Code

bettabeal-fe - page.tsx

```
[illegible]
```

bettabeal-fe - page.tsx

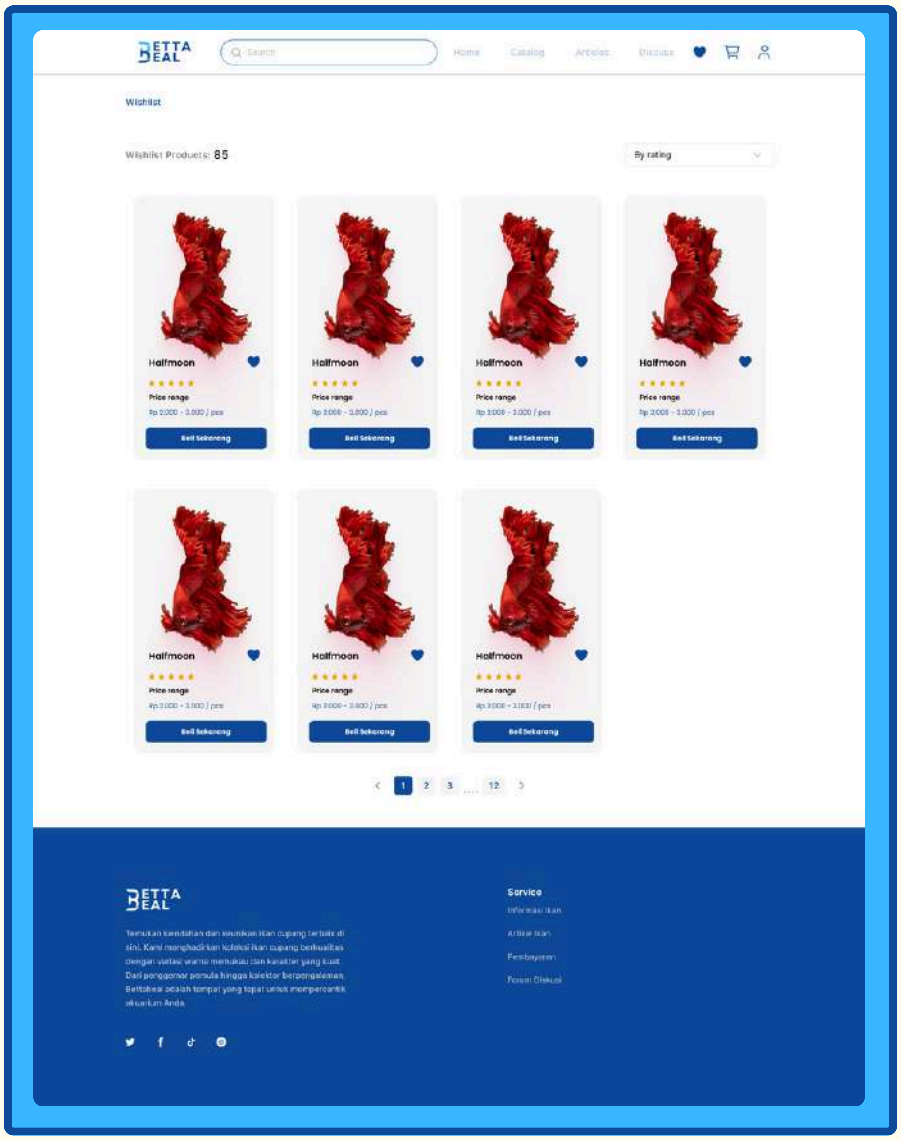
```
[illegible]
```

Website Customer

9. Wishlist Page

Halaman wishlist ini menampilkan daftar produk yang telah ditambahkan pengguna sebagai favorit. Pengguna dapat melihat produk-produk yang mereka minati, mengedit wishlist dengan menambah atau menghapus item, serta melanjutkan untuk membeli produk yang ada di dalam wishlist langsung dari halaman ini.

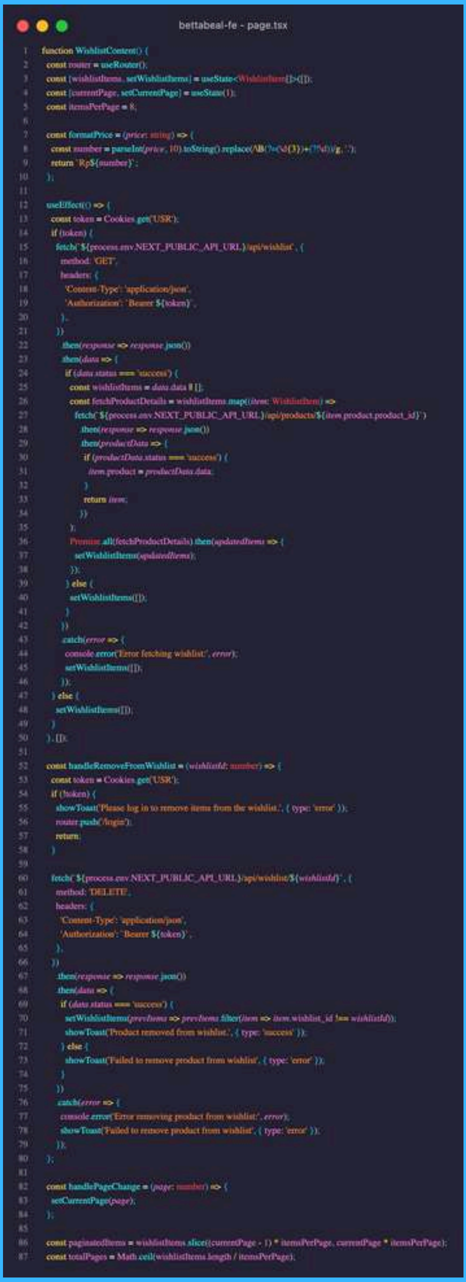
Pageview
Wishlist page



Temukan keindahan dan keunikan ikan cupang terbaik di sini. Kami menghadirkan koleksi ikan cupang berkualitas dengan variasi warna memukau dan karakter yang kuat. Dari penggemar pemula hingga kolektor berpengalaman, Bettabeal adalah tempat yang tepat untuk mempercantik akuarium Anda.

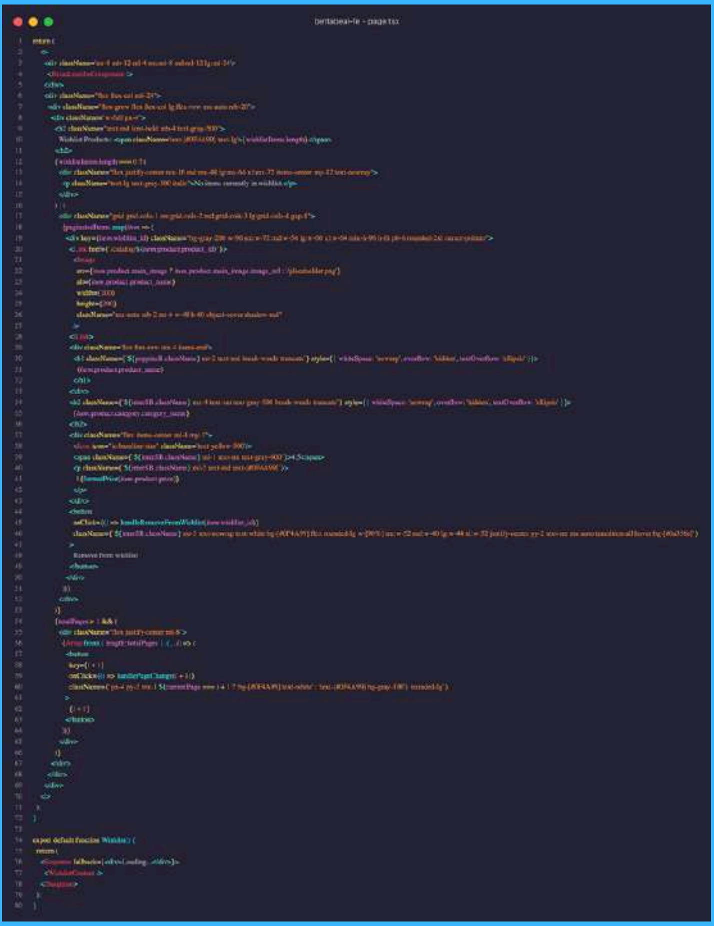
Website Customer

Source Code



bettabeal-fe - page.tsx

```tsx
function WishlistContent() {
  const router = useRouter();
  const [wishlistItems, setWishlistItems] = useState<WishlistItem[]>([]);
  const [currentPage, setCurrentPage] = useState(1);
  const itemsPerPage = 8;

  const formatPrice = (price: string) => {
    const number = parseInt(price, 10).toString().replace(/\B(?=(\d{3})+(?!\d))/g, '.');
    return `Rp${number}`;
  };

  useEffect(() => {
    const token = Cookies.get('USR');
    if (token) {
      fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/wishlist`, {
        method: 'GET',
        headers: {
          'Content-Type': 'application/json',
          'Authorization': `Bearer ${token}`,
        },
      })
        .then(response => response.json())
        .then(data => {
          if (data.status === 'success') {
            const wishlistItems = data.data || [];
            const fetchProductDetails = wishlistItems.map((item: WishlistItem) =>
              fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/products/${item.product.product_id}`)
                .then(response => response.json())
                .then(productData => {
                  if (productData.status === 'success') {
                    item.product = productData.data;
                  }
                  return item;
                })
            );
            Promise.all(fetchProductDetails).then(updatedItems => {
              setWishlistItems(updatedItems);
            });
          } else {
            setWishlistItems([]);
          }
        })
        .catch(error => {
          console.error('Error fetching wishlist:', error);
          setWishlistItems([]);
        });
    } else {
      setWishlistItems([]);
    }
  }, []);

  const handleRemoveFromWishlist = (wishlistId: number) => {
    const token = Cookies.get('USR');
    if (!token) {
      showToast('Please log in to remove items from the wishlist.', { type: 'error' });
      router.push('/login');
      return;
    }

    fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/wishlist/${wishlistId}`, {
      method: 'DELETE',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${token}`,
      },
    })
      .then(response => response.json())
      .then(data => {
        if (data.status === 'success') {
          setWishlistItems(prevItems => prevItems.filter(item => item.wishlist_id !== wishlistId));
          showToast('Product removed from wishlist.', { type: 'success' });
        } else {
          showToast('Failed to remove product from wishlist', { type: 'error' });
        }
      })
      .catch(error => {
        console.error('Error removing product from wishlist:', error);
        showToast('Failed to remove product from wishlist', { type: 'error' });
      });
  };

  const handlePageChange = (page: number) => {
    setCurrentPage(page);
  };

  const paginatedItems = wishlistItems.slice((currentPage - 1) * itemsPerPage, currentPage * itemsPerPage);
  const totalPages = Math.ceil(wishlistItems.length / itemsPerPage);
```



bettabeal-fe - page.tsx

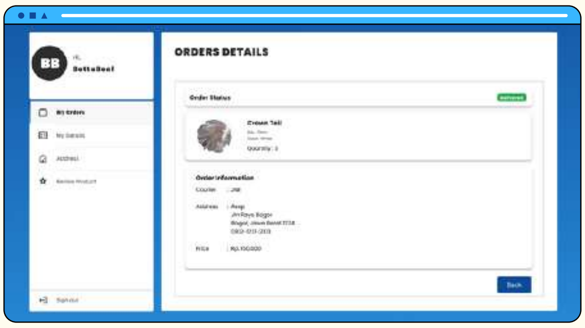
```tsx
  return (
    <>
      <div className="[illegible]">
        <[illegible] />
      </div>
      <div className="flex flex-col [illegible]">
       <div className="flex-grow flex flex-col lg:flex-row [illegible]">
        <div className="w-full px-[illegible]">
         <h2 className="text-md font-bold mb-4 text-gray-[illegible]">
          Wishlist Products: <span className="text-[#[illegible]] text-lg">{wishlistItems.length}</span>
         </h2>
         {wishlistItems.length === 0 ? (
          <div className="flex justify-center [illegible] text-center">
           <p className="text-lg text-gray-300 italic">No items currently in wishlist.</p>
          </div>
         ) : (
          <div className="grid grid-cols-1 sm:grid-cols-2 md:grid-cols-3 lg:grid-cols-4 gap-4">
           {paginatedItems.map(item => (
            <div key={item.wishlist_id} className="[illegible]">
             <Link href={`/catalog/${item.product.product_id}`}>
              <Image
               src={item.product.main_image ? item.product.main_image.image_url : '/placeholder.png'}
               alt={item.product.product_name}
               width={300}
               height={200}
               className="mx-auto mb-2 [illegible] object-cover shadow-md"
              />
             </Link>
             <div className="flex flex-row [illegible] items-end">
              <h1 className={`${poppinsB.className} [illegible] break-words truncate`} style={{ whiteSpace: 'nowrap', overflow: 'hidden', textOverflow: 'ellipsis' }}>
               {item.product.product_name}
              </h1>
             </div>
             <h2 className={`${interSB.className} [illegible] text-gray-500 break-words truncate`} style={{ whiteSpace: 'nowrap', overflow: 'hidden', textOverflow: 'ellipsis' }}>
              {item.product.category.category_name}
             </h2>
             <div className="flex items-center mt-4 [illegible]">
              <Icon icon="[illegible]" className="text-yellow-500" />
              <span className={`${interSB.className} [illegible] text-gray-900`}>4.5</span>
              <p className={`${interSB.className} [illegible] text-[#[illegible]]`}>
               {formatPrice(item.product.price)}
              </p>
             </div>
             <button
              onClick={() => handleRemoveFromWishlist(item.wishlist_id)}
              className={`${interSB.className} [illegible]`}
             >
              Remove from wishlist
             </button>
            </div>
           ))}
          </div>
         )}
         {totalPages > 1 && (
          <div className="flex justify-center mt-8">
           {Array.from({ length: totalPages }, (_, i) => (
            <button
             key={i + 1}
             onClick={() => handlePageChange(i + 1)}
             className={`px-4 py-2 mx-1 ${currentPage === i + 1 ? 'bg-[#[illegible]] text-white' : 'text-[#[illegible]] bg-gray-100'} rounded-lg`}
            >
             {i + 1}
            </button>
           ))}
          </div>
         )}
        </div>
       </div>
      </div>
    </>
  );
}

export default function Wishlist() {
  return (
    <Suspense fallback={<div>Loading...</div>}>
      <WishlistContent />
    </Suspense>
  );
}
```

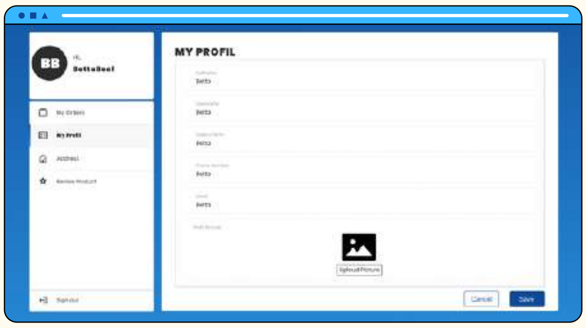
Website Customer

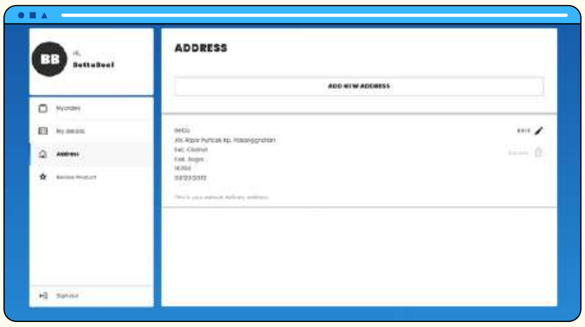
10. Settings Page

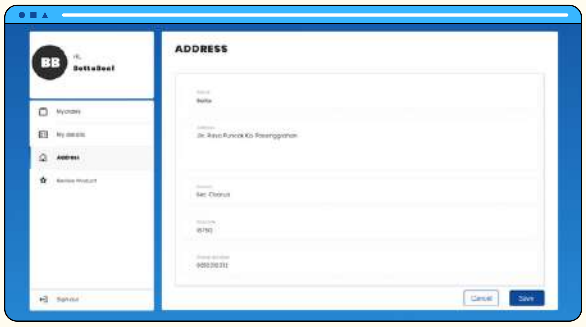
Halaman pengaturan (settings) ini memungkinkan pengguna untuk mengelola berbagai aspek akun mereka. Pengguna dapat melihat riwayat pesanan, menambahkan alamat baru, memberikan ulasan pada produk yang telah dibeli, serta mengelola data pribadi mereka seperti nama, email, dan informasi kontak lainnya. Fitur-fitur ini mempermudah pengguna dalam mengelola akun dan meningkatkan pengalaman mereka dalam menggunakan aplikasi.

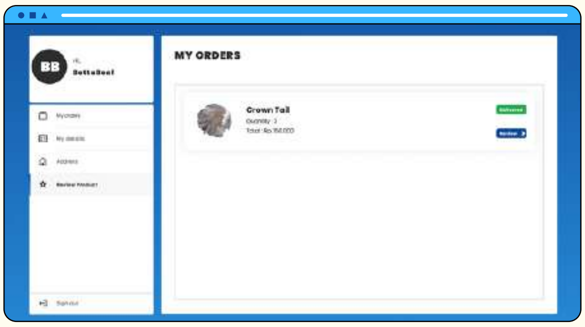
Pageview
Settings page

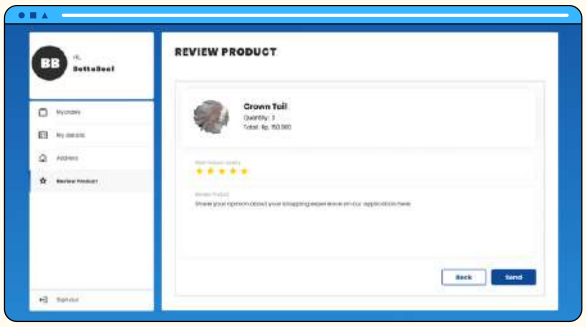












Website Customer

Source Code

Layout

bettabeat-fe – layout.tsx

[illegible]

Profile

bettabeat-fe – page.tsx

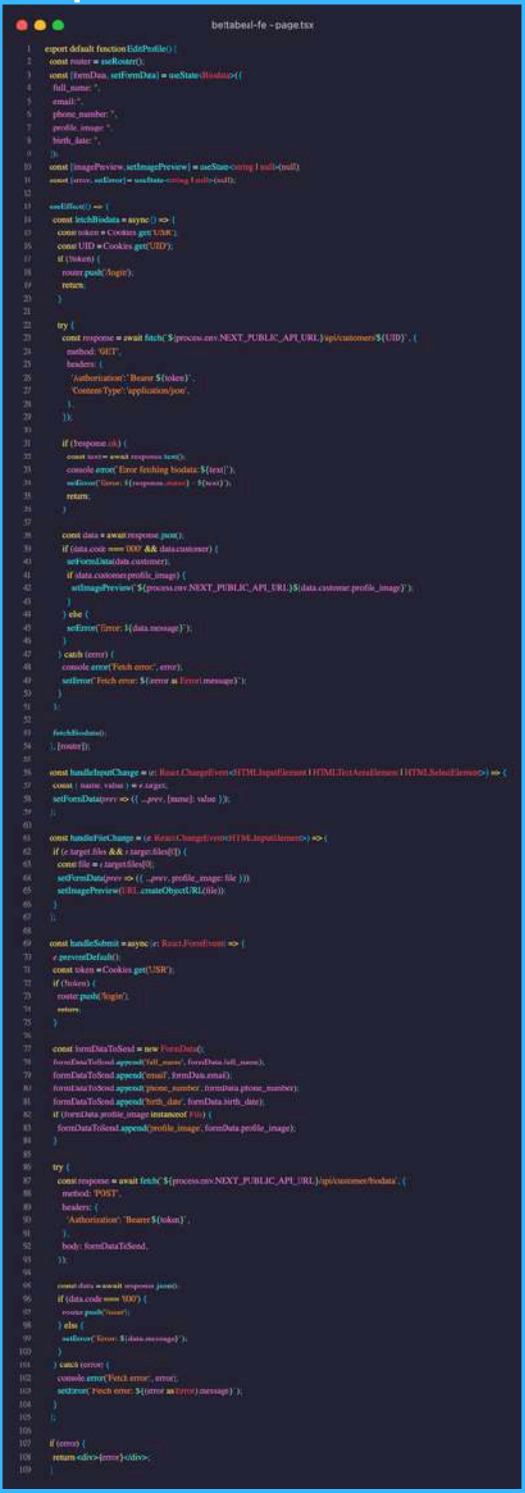
```
export default function Profile() {
  const router = useRouter();
  [illegible]
```

[illegible]

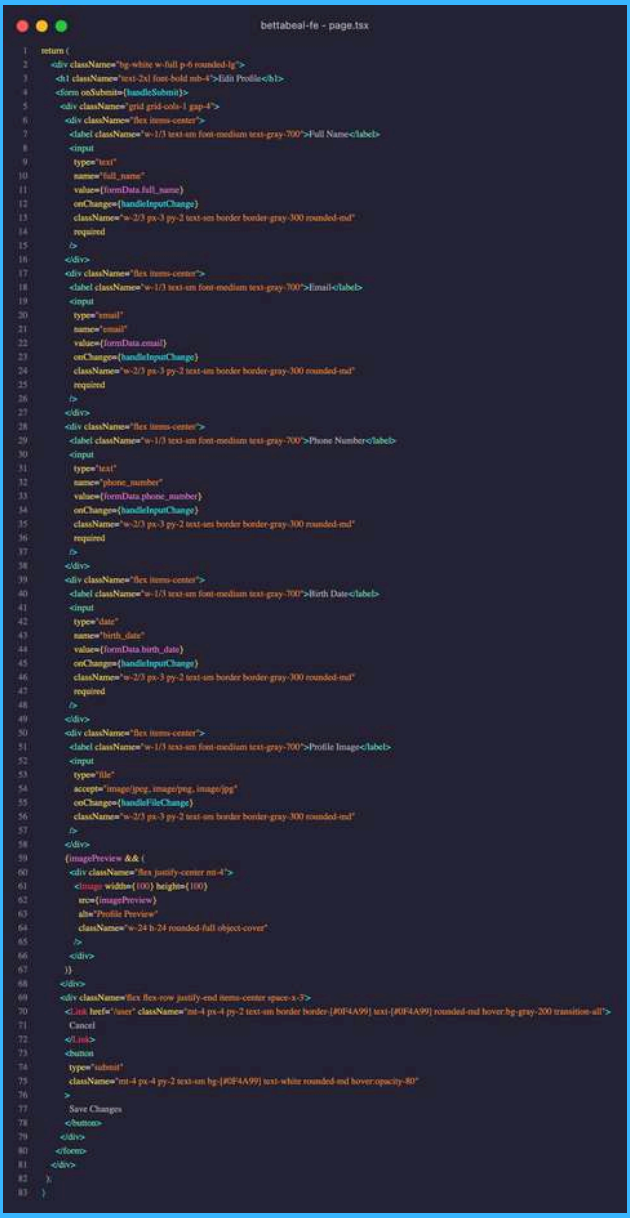
Website Customer

Source Code

Edit profile


bettabeal-fe - page.tsx

```tsx
export default function EditProfile() {
  const router = useRouter();
  const [formData, setFormData] = useState<Biodata>({
    full_name: '',
    email: '',
    phone_number: '',
    profile_image: '',
    birth_date: '',
  });
  const [imagePreview, setImagePreview] = useState<string | null>(null);
  const [error, setError] = useState<string | null>(null);

  useEffect(() => {
    const fetchBiodata = async () => {
      const token = Cookies.get('USR');
      const UID = Cookies.get('UID');
      if (!token) {
        router.push('/login');
        return;
      }

      try {
        const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/customers/${UID}`, {
          method: 'GET',
          headers: {
            'Authorization': `Bearer ${token}`,
            'Content-Type': 'application/json',
          },
        });

        if (!response.ok) {
          const text = await response.text();
          console.error(`Error fetching biodata: ${text}`);
          setError(`Error: ${response.status} - ${text}`);
          return;
        }

        const data = await response.json();
        if (data.code === '000' && data.customer) {
          setFormData(data.customer);
          if (data.customer.profile_image) {
            setImagePreview(`${process.env.NEXT_PUBLIC_API_URL}${data.customer.profile_image}`);
          }
        } else {
          setError(`Error: ${data.message}`);
        }
      } catch (error) {
        console.error('Fetch error:', error);
        setError(`Fetch error: ${(error as Error).message}`);
      }
    };

    fetchBiodata();
  }, [router]);

  const handleInputChange = (e: React.ChangeEvent<HTMLInputElement | HTMLTextAreaElement | HTMLSelectElement>) => {
    const { name, value } = e.target;
    setFormData(prev => ({ ...prev, [name]: value }));
  };

  const handleFileChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    if (e.target.files && e.target.files[0]) {
      const file = e.target.files[0];
      setFormData(prev => ({ ...prev, profile_image: file }));
      setImagePreview(URL.createObjectURL(file));
    }
  };

  const handleSubmit = async (e: React.FormEvent) => {
    e.preventDefault();
    const token = Cookies.get('USR');
    if (!token) {
      router.push('/login');
      return;
    }

    const formDataToSend = new FormData();
    formDataToSend.append('full_name', formData.full_name);
    formDataToSend.append('email', formData.email);
    formDataToSend.append('phone_number', formData.phone_number);
    formDataToSend.append('birth_date', formData.birth_date);
    if (formData.profile_image instanceof File) {
      formDataToSend.append('profile_image', formData.profile_image);
    }

    try {
      const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/customer/biodata`, {
        method: 'POST',
        headers: {
          'Authorization': `Bearer ${token}`,
        },
        body: formDataToSend,
      });

      const data = await response.json();
      if (data.code === '000') {
        router.push('/user');
      } else {
        setError(`Error: ${data.message}`);
      }
    } catch (error) {
      console.error('Fetch error:', error);
      setError(`Fetch error: ${(error as Error).message}`);
    }
  };

  if (error) {
    return <div>{error}</div>;
  }
```

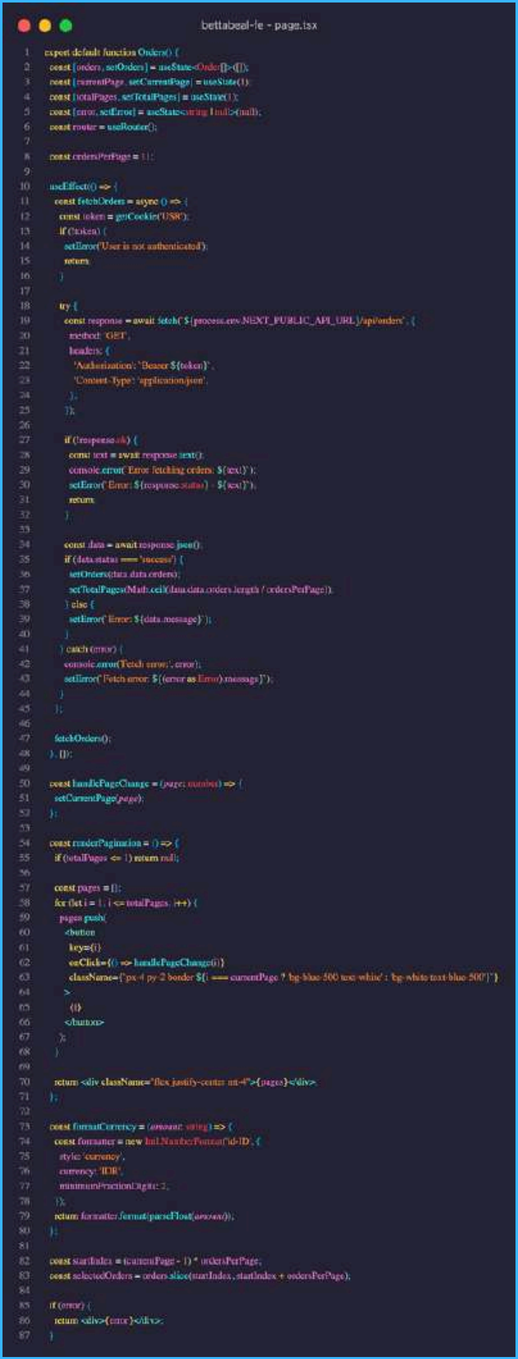

bettabeal-fe - page.tsx

```tsx
return (
  <div className="bg-white w-full p-6 rounded-lg">
    <h1 className="text-2xl font-bold mb-4">Edit Profile</h1>
    <form onSubmit={handleSubmit}>
      <div className="grid grid-cols-1 gap-4">
        <div className="flex items-center">
          <label className="w-1/3 text-sm font-medium text-gray-700">Full Name</label>
          <input
            type="text"
            name="full_name"
            value={formData.full_name}
            onChange={handleInputChange}
            className="w-2/3 px-3 py-2 text-sm border border-gray-300 rounded-md"
            required
          />
        </div>
        <div className="flex items-center">
          <label className="w-1/3 text-sm font-medium text-gray-700">Email</label>
          <input
            type="email"
            name="email"
            value={formData.email}
            onChange={handleInputChange}
            className="w-2/3 px-3 py-2 text-sm border border-gray-300 rounded-md"
            required
          />
        </div>
        <div className="flex items-center">
          <label className="w-1/3 text-sm font-medium text-gray-700">Phone Number</label>
          <input
            type="text"
            name="phone_number"
            value={formData.phone_number}
            onChange={handleInputChange}
            className="w-2/3 px-3 py-2 text-sm border border-gray-300 rounded-md"
            required
          />
        </div>
        <div className="flex items-center">
          <label className="w-1/3 text-sm font-medium text-gray-700">Birth Date</label>
          <input
            type="date"
            name="birth_date"
            value={formData.birth_date}
            onChange={handleInputChange}
            className="w-2/3 px-3 py-2 text-sm border border-gray-300 rounded-md"
            required
          />
        </div>
        <div className="flex items-center">
          <label className="w-1/3 text-sm font-medium text-gray-700">Profile Image</label>
          <input
            type="file"
            accept="image/jpeg, image/png, image/jpg"
            onChange={handleFileChange}
            className="w-2/3 px-3 py-2 text-sm border border-gray-300 rounded-md"
          />
        </div>
        {imagePreview && (
          <div className="flex justify-center mt-4">
            <Image width={100} height={100}
              src={imagePreview}
              alt="Profile Preview"
              className="w-24 h-24 rounded-full object-cover"
            />
          </div>
        )}
      </div>
      <div className="flex flex-row justify-end items-center space-x-3">
        <Link href="/user" className="mt-4 px-4 py-2 text-sm border border-[#0F4A99] text-[#0F4A99] rounded-md hover:bg-gray-200 transition-all">
          Cancel
        </Link>
        <button
          type="submit"
          className="mt-4 px-4 py-2 text-sm bg-[#0F4A99] text-white rounded-md hover:opacity-80"
        >
          Save Changes
        </button>
      </div>
    </form>
  </div>
);
}
```

Website Customer

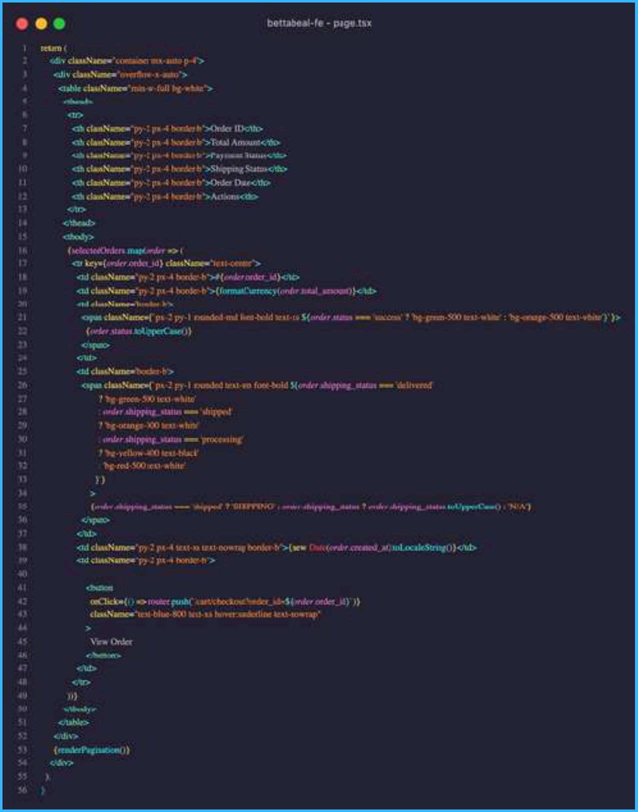
Source Code

Orders



bettabeal-fe - page.tsx

```tsx
export default function Orders() {
 const [orders, setOrders] = useState<Order[]>([]);
 const [currentPage, setCurrentPage] = useState(1);
 const [totalPages, setTotalPages] = useState(1);
 const [error, setError] = useState<string | null>(null);
 const router = useRouter();

 const ordersPerPage = 11;

 useEffect(() => {
  const fetchOrders = async () => {
   const token = getCookie('USR');
   if (!token) {
    setError('User is not authenticated');
    return;
   }

   try {
    const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/orders`, {
     method: 'GET',
     headers: {
      'Authorization': `Bearer ${token}`,
      'Content-Type': 'application/json',
     },
    });

    if (!response.ok) {
     const text = await response.text();
     console.error(`Error fetching orders: ${text}`);
     setError(`Error: ${response.status} - ${text}`);
     return;
    }

    const data = await response.json();
    if (data.status === 'success') {
     setOrders(data.data.orders);
     setTotalPages(Math.ceil(data.data.orders.length / ordersPerPage));
    } else {
     setError(`Error: ${data.message}`);
    }
   } catch (error) {
    console.error('Fetch error:', error);
    setError(`Fetch error: ${(error as Error).message}`);
   }
  };

  fetchOrders();
 }, []);

 const handlePageChange = (page: number) => {
  setCurrentPage(page);
 };

 const renderPagination = () => {
  if (totalPages <= 1) return null;

  const pages = [];
  for (let i = 1; i <= totalPages; i++) {
   pages.push(
    <button
     key={i}
     onClick={() => handlePageChange(i)}
     className={`px-4 py-2 border ${i === currentPage ? 'bg-blue-500 text-white' : 'bg-white text-blue-500'}`}
    >
     {i}
    </button>
   );
  }

  return <div className="flex justify-center mt-4">{pages}</div>;
 };

 const formatCurrency = (amount: string) => {
  const formatter = new Intl.NumberFormat('id-ID', {
   style: 'currency',
   currency: 'IDR',
   minimumFractionDigits: 2,
  });
  return formatter.format(parseFloat(amount));
 };

 const startIndex = (currentPage - 1) * ordersPerPage;
 const selectedOrders = orders.slice(startIndex, startIndex + ordersPerPage);

 if (error) {
  return <div>{error}</div>;
 }
```



bettabeal-fe - page.tsx

```tsx
return (
  <div className="container mx-auto p-4">
   <div className="overflow-x-auto">
    <table className="min-w-full bg-white">
     <thead>
      <tr>
       <th className="py-2 px-4 border-b">Order ID</th>
       <th className="py-2 px-4 border-b">Total Amount</th>
       <th className="py-2 px-4 border-b">Payment Status</th>
       <th className="py-2 px-4 border-b">Shipping Status</th>
       <th className="py-2 px-4 border-b">Order Date</th>
       <th className="py-2 px-4 border-b">Actions</th>
      </tr>
     </thead>
     <tbody>
      {selectedOrders.map(order => (
       <tr key={order.order_id} className="text-center">
        <td className="py-2 px-4 border-b">#{order.order_id}</td>
        <td className="py-2 px-4 border-b">{formatCurrency(order.total_amount)}</td>
        <td className="border-b">
         <span className={`px-2 py-1 rounded-md font-bold text-xs ${order.status === 'success' ? 'bg-green-500 text-white' : 'bg-orange-500 text-white'}`}>
          {order.status.toUpperCase()}
         </span>
        </td>
        <td className="border-b">
         <span className={`px-2 py-1 rounded text-sm font-bold ${order.shipping_status === 'delivered'
             ? 'bg-green-500 text-white'
             : order.shipping_status === 'shipped'
             ? 'bg-orange-300 text-white'
             : order.shipping_status === 'processing'
             ? 'bg-yellow-400 text-black'
             : 'bg-red-500 text-white'
            }`}
          >
          {order.shipping_status === 'shipped' ? 'SHIPPING' : order.shipping_status ? order.shipping_status.toUpperCase() : 'N/A'}
         </span>
        </td>
        <td className="py-2 px-4 text-xs text-nowrap border-b">{new Date(order.created_at).toLocaleString()}</td>
        <td className="py-2 px-4 border-b">

          <button
           onClick={() => router.push(`/cart/checkout?order_id=${order.order_id}`)}
           className="text-blue-800 text-xs hover:underline text-nowrap"
          >
           View Order
          </button>
        </td>
       </tr>
      ))}
     </tbody>
    </table>
   </div>
   {renderPagination()}
  </div>
 );
}
```

Website Customer

Source Code

Reviews

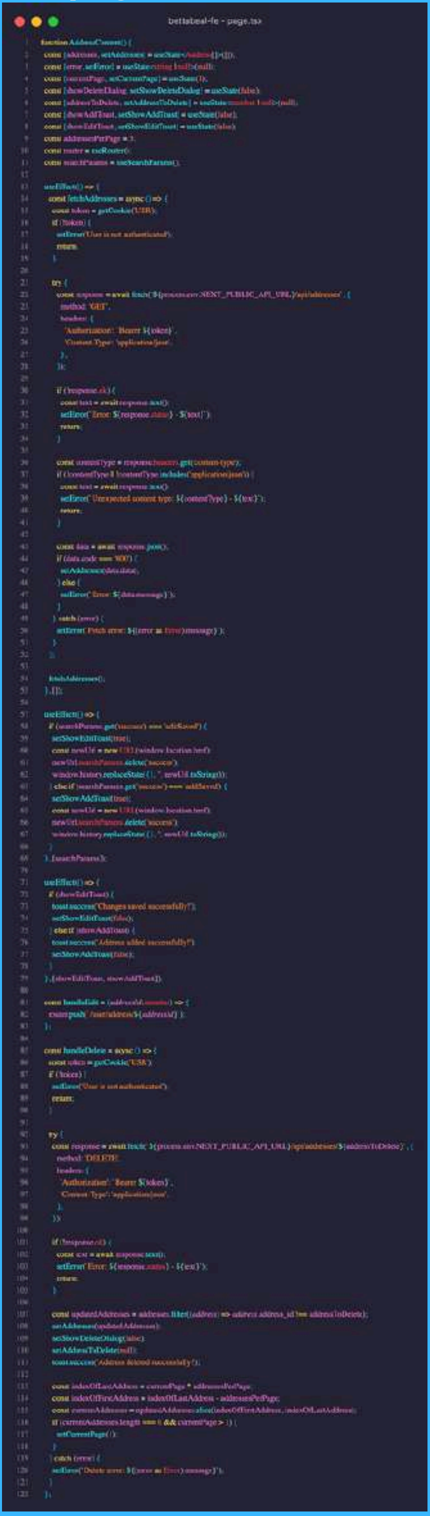
[illegible]

[illegible]

Website Customer

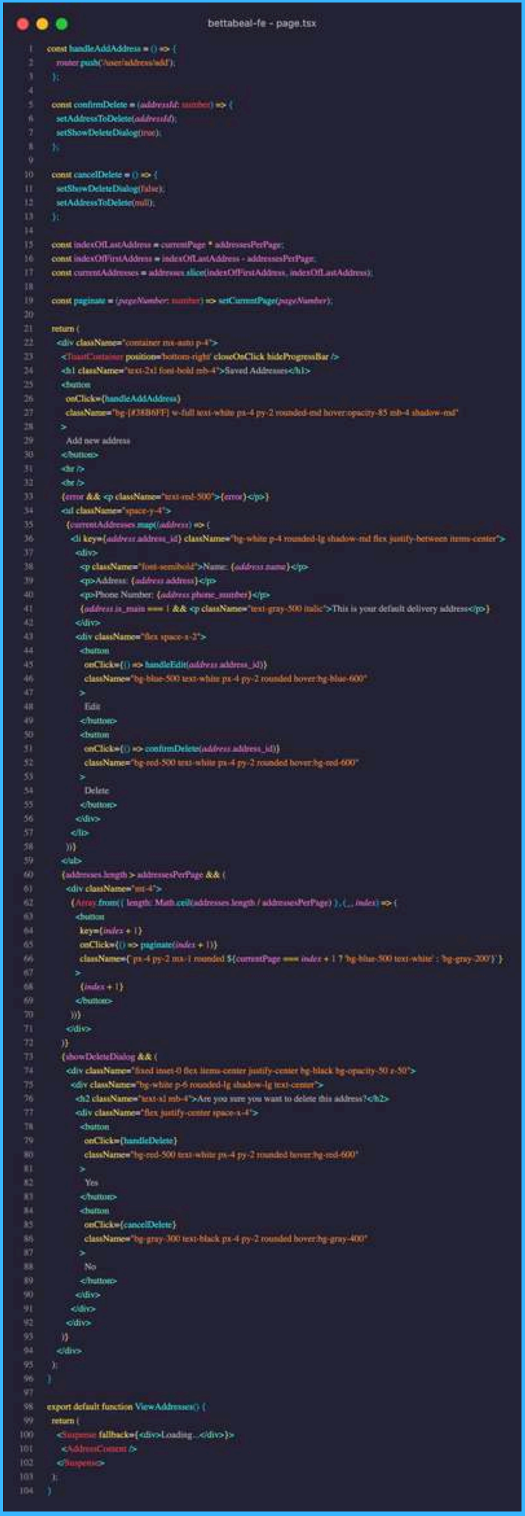
Source Code

Address page



bettabeal-fe - page.tsx

```
function AddressContent() {
  const [addresses, setAddresses] = useState<Address[]>([]);
  const [error, setError] = useState<string | null>(null);
  const [currentPage, setCurrentPage] = useState(1);
  const [showDeleteDialog, setShowDeleteDialog] = useState(false);
  const [addressToDelete, setAddressToDelete] = useState<number | null>(null);
  const [showAddToast, setShowAddToast] = useState(false);
  const [showEditToast, setShowEditToast] = useState(false);
  const addressesPerPage = 3;
  const router = useRouter();
  const searchParams = useSearchParams();

  useEffect(() => {
    const fetchAddresses = async () => {
      const token = getCookie('USR');
      if (!token) {
        setError('User is not authenticated');
        return;
      }

      try {
        const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/addresses`, {
          method: 'GET',
          headers: {
            'Authorization': `Bearer ${token}`,
            'Content-Type': 'application/json',
          },
        });

        if (!response.ok) {
          const text = await response.text();
          setError(`Error: ${response.status} - ${text}`);
          return;
        }

        const contentType = response.headers.get('content-type');
        if (!contentType || !contentType.includes('application/json')) {
          const text = await response.text();
          setError(`Unexpected content type: ${contentType} - ${text}`);
          return;
        }

        const data = await response.json();
        if (data.code === '000') {
          setAddresses(data.data);
        } else {
          setError(`Error: ${data.message}`);
        }
      } catch (error) {
        setError(`Fetch error: ${(error as Error).message}`);
      }
    };

    fetchAddresses();
  }, []);

  useEffect(() => {
    if (searchParams.get('success') === 'editSaved') {
      setShowEditToast(true);
      const newUrl = new URL(window.location.href);
      newUrl.searchParams.delete('success');
      window.history.replaceState({}, '', newUrl.toString());
    } else if (searchParams.get('success') === 'addSaved') {
      setShowAddToast(true);
      const newUrl = new URL(window.location.href);
      newUrl.searchParams.delete('success');
      window.history.replaceState({}, '', newUrl.toString());
    }
  }, [searchParams]);

  useEffect(() => {
    if (showEditToast) {
      toast.success('Changes saved successfully!');
      setShowEditToast(false);
    } else if (showAddToast) {
      toast.success('Address added successfully!');
      setShowAddToast(false);
    }
  }, [showEditToast, showAddToast]);

  const handleEdit = (addressId: number) => {
    router.push(`/user/address/${addressId}`);
  };

  const handleDelete = async () => {
    const token = getCookie('USR');
    if (!token) {
      setError('User is not authenticated');
      return;
    }

    try {
      const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/addresses/${addressToDelete}`, {
        method: 'DELETE',
        headers: {
          'Authorization': `Bearer ${token}`,
          'Content-Type': 'application/json',
        },
      });

      if (!response.ok) {
        const text = await response.text();
        setError(`Error: ${response.status} - ${text}`);
        return;
      }

      const updatedAddresses = addresses.filter((address) => address.address_id !== addressToDelete);
      setAddresses(updatedAddresses);
      setShowDeleteDialog(false);
      setAddressToDelete(null);
      toast.success('Address deleted successfully!');

      const indexOfLastAddress = currentPage * addressesPerPage;
      const indexOfFirstAddress = indexOfLastAddress - addressesPerPage;
      const currentAddresses = updatedAddresses.slice(indexOfFirstAddress, indexOfLastAddress);
      if (currentAddresses.length === 0 && currentPage > 1) {
        setCurrentPage(1);
      }
    } catch (error) {
      setError(`Delete error: ${(error as Error).message}`);
    }
  };
```



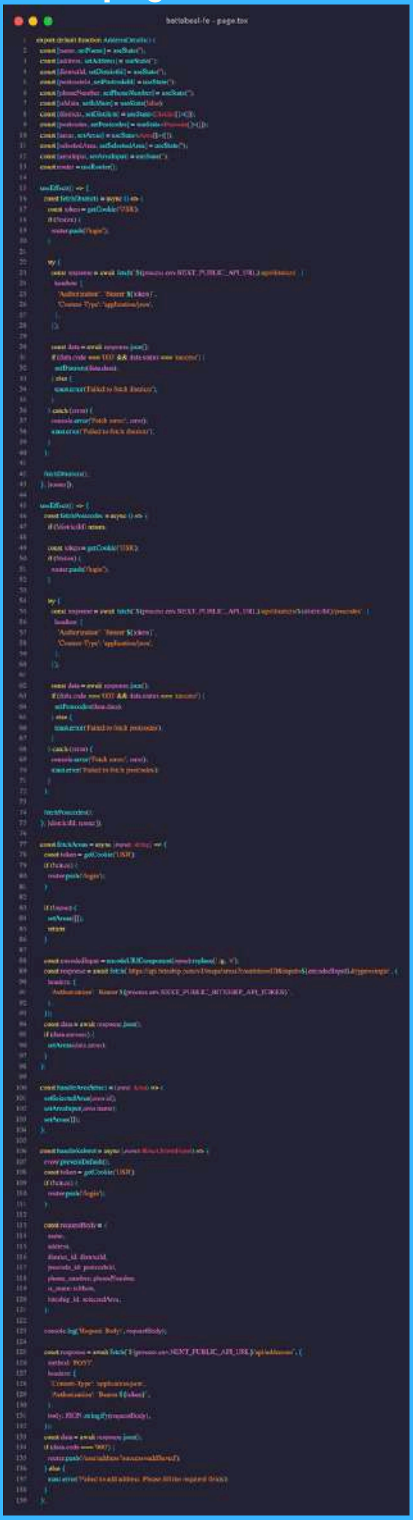
bettabeal-fe - page.tsx

```
const handleAddAddress = () => {
  router.push('/user/address/add');
 };

 const confirmDelete = (addressId: number) => {
  setAddressToDelete(addressId);
  setShowDeleteDialog(true);
 };

 const cancelDelete = () => {
  setShowDeleteDialog(false);
  setAddressToDelete(null);
 };

 const indexOfLastAddress = currentPage * addressesPerPage;
 const indexOfFirstAddress = indexOfLastAddress - addressesPerPage;
 const currentAddresses = addresses.slice(indexOfFirstAddress, indexOfLastAddress);

 const paginate = (pageNumber: number) => setCurrentPage(pageNumber);

 return (
  <div className="container mx-auto p-4">
   <ToastContainer position='bottom-right' closeOnClick hideProgressBar />
   <h1 className="text-2xl font-bold mb-4">Saved Addresses</h1>
   <button
    onClick={handleAddAddress}
    className="bg-[#38B6FF] w-full text-white px-4 py-2 rounded-md hover:opacity-85 mb-4 shadow-md"
   >
    Add new address
   </button>
   <hr />
   <br />
   {error && <p className="text-red-500">{error}</p>}
   <ul className="space-y-4">
    {currentAddresses.map((address) => (
     <li key={address.address_id} className="bg-white p-4 rounded-lg shadow-md flex justify-between items-center">
      <div>
       <p className="font-semibold">Name: {address.name}</p>
       <p>Address: {address.address}</p>
       <p>Phone Number: {address.phone_number}</p>
       {address.is_main === 1 && <p className="text-gray-500 italic">This is your default delivery address</p>}
      </div>
      <div className="flex space-x-2">
       <button
        onClick={() => handleEdit(address.address_id)}
        className="bg-blue-500 text-white px-4 py-2 rounded hover:bg-blue-600"
       >
        Edit
       </button>
       <button
        onClick={() => confirmDelete(address.address_id)}
        className="bg-red-500 text-white px-4 py-2 rounded hover:bg-red-600"
       >
        Delete
       </button>
      </div>
     </li>
    ))}
   </ul>
   {addresses.length > addressesPerPage && (
    <div className="mt-4">
     {Array.from({ length: Math.ceil(addresses.length / addressesPerPage) }, (_, index) => (
      <button
       key={index + 1}
       onClick={() => paginate(index + 1)}
       className={`px-4 py-2 mx-1 rounded ${currentPage === index + 1 ? 'bg-blue-500 text-white' : 'bg-gray-200'}`}
      >
       {index + 1}
      </button>
     ))}
    </div>
   )}
   {showDeleteDialog && (
    <div className="fixed inset-0 flex items-center justify-center bg-black bg-opacity-50 z-50">
     <div className="bg-white p-6 rounded-lg shadow-lg text-center">
      <h2 className="text-xl mb-4">Are you sure you want to delete this address?</h2>
      <div className="flex justify-center space-x-4">
       <button
        onClick={handleDelete}
        className="bg-red-500 text-white px-4 py-2 rounded hover:bg-red-600"
       >
        Yes
       </button>
       <button
        onClick={cancelDelete}
        className="bg-gray-300 text-black px-4 py-2 rounded hover:bg-gray-400"
       >
        No
       </button>
      </div>
     </div>
    </div>
   )}
  </div>
 );
}

export default function ViewAddresses() {
 return (
  <Suspense fallback={<div>Loading...</div>}>
   <AddressContent />
  </Suspense>
 );
}
```

Website Customer

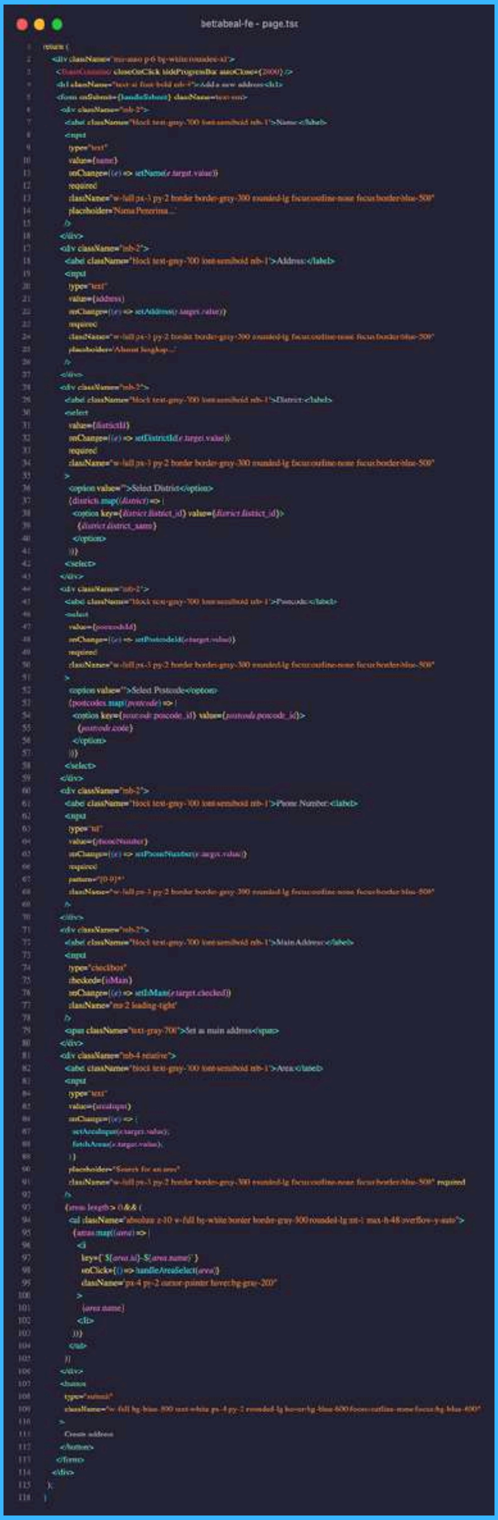
Source Code

Add Address page



betabeal-fe - page.tsx

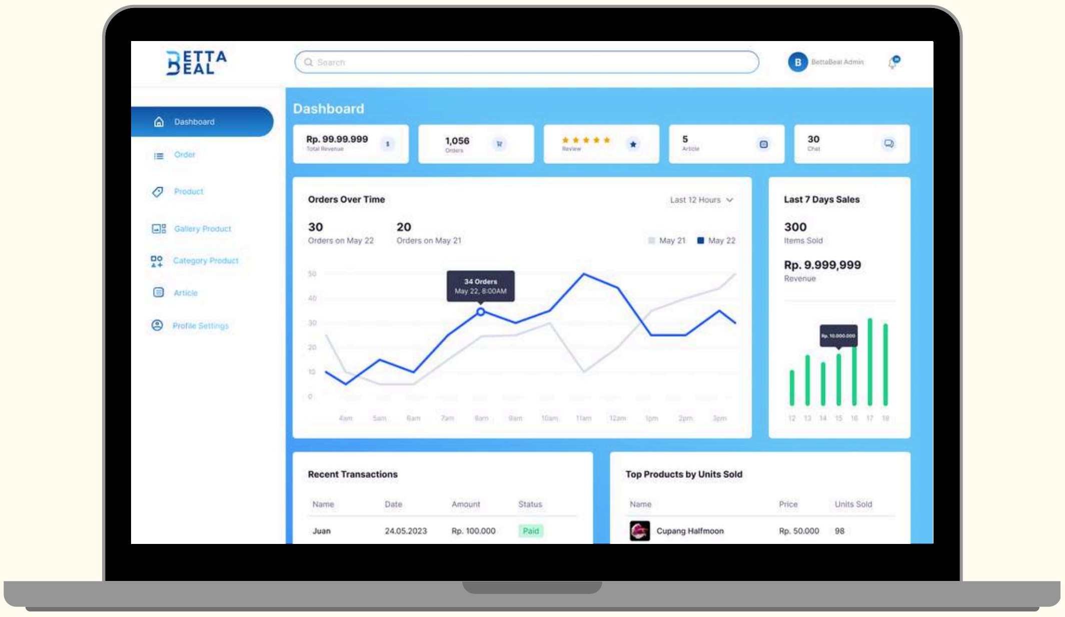
```
export default function AddressDetails() {
  const [name, setName] = useState('');
  const [address, setAddress] = useState('');
  const [districtId, setDistrictId] = useState('');
  const [postcodeId, setPostcodeId] = useState('');
  const [phoneNumber, setPhoneNumber] = useState('');
  const [isMain, setIsMain] = useState(false);
  const [districts, setDistricts] = useState<District[]>([]);
  const [postcodes, setPostcodes] = useState<Postcode[]>([]);
  const [areas, setAreas] = useState<Area[]>([]);
  const [selectedArea, setSelectedArea] = useState('');
  const [areaInput, setAreaInput] = useState('');
  const router = useRouter();

  useEffect(() => {
    const fetchDistricts = async () => {
      const token = getCookie('USR');
      if (!token) {
        router.push('/login');
      }

      try {
        const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/districts`, {
          headers: {
            'Authorization': `Bearer ${token}`,
            'Content-Type': 'application/json',
          },
        });

        const data = await response.json();
        if (data.code === '000' && data.status === 'success') {
          setDistricts(data.data);
        } else {
          toast.error('Failed to fetch districts');
        }
      } catch (error) {
        console.error('Fetch error:', error);
        toast.error('Failed to fetch districts');
      }
    };

    fetchDistricts();
  }, [router]);

  useEffect(() => {
    const fetchPostcodes = async () => {
      if (!districtId) return;

      const token = getCookie('USR');
      if (!token) {
        router.push('/login');
      }

      try {
        const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/districts/${districtId}/postcodes`, {
          headers: {
            'Authorization': `Bearer ${token}`,
            'Content-Type': 'application/json',
          },
        });

        const data = await response.json();
        if (data.code === '000' && data.status === 'success') {
          setPostcodes(data.data);
        } else {
          toast.error('Failed to fetch postcodes');
        }
      } catch (error) {
        console.error('Fetch error:', error);
        toast.error('Failed to fetch postcodes');
      }
    };

    fetchPostcodes();
  }, [districtId, router]);

  const fetchAreas = async (input: string) => {
    const token = getCookie('USR');
    if (!token) {
      router.push('/login');
    }

    if (!input) {
      setAreas([]);
      return;
    }

    const encodedInput = encodeURIComponent(input).replace(/ /g, '+');
    const response = await fetch(`https://api.biteship.com/v1/maps/areas?countries=ID&input=${encodedInput}&type=single`, {
      headers: {
        Authorization: `Bearer ${process.env.NEXT_PUBLIC_BITESHIP_API_TOKEN}`,
      },
    });
    const data = await response.json();
    if (data.success) {
      setAreas(data.areas);
    }
  };

  const handleAreaSelect = (area: Area) => {
    setSelectedArea(area.id);
    setAreaInput(area.name);
    setAreas([]);
  };

  const handleSubmit = async (event: React.FormEvent) => {
    event.preventDefault();
    const token = getCookie('USR');
    if (!token) {
      router.push('/login');
    }

    const requestBody = {
      name,
      address,
      district_id: districtId,
      postcode_id: postcodeId,
      phone_number: phoneNumber,
      is_main: isMain,
      biteship_id: selectedArea,
    };

    console.log('Request Body:', requestBody);

    const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/addresses`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${token}`,
      },
      body: JSON.stringify(requestBody),
    });
    const data = await response.json();
    if (data.code === '000') {
      router.push('/user/address?status=addSuccess');
    } else {
      toast.error('Failed to add address. Please fill the required fields');
    }
  };
```



betabeal-fe - page.tsx

```
  return (
    <div className="mx-auto p-6 bg-white rounded-xl">
      <ToastContainer closeOnClick hideProgressBar autoClose={2000} />
      <h1 className="text-xl font-bold mb-4">Add a new address</h1>
      <form onSubmit={handleSubmit} className="text-sm">
        <div className="mb-2">
          <label className="block text-gray-700 font-semibold mb-1">Name:</label>
          <input
            type="text"
            value={name}
            onChange={(e) => setName(e.target.value)}
            required
            className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:border-blue-500"
            placeholder="Nama Penerima..."
          />
        </div>
        <div className="mb-2">
          <label className="block text-gray-700 font-semibold mb-1">Address:</label>
          <input
            type="text"
            value={address}
            onChange={(e) => setAddress(e.target.value)}
            required
            className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:border-blue-500"
            placeholder="Alamat lengkap..."
          />
        </div>
        <div className="mb-2">
          <label className="block text-gray-700 font-semibold mb-1">District:</label>
          <select
            value={districtId}
            onChange={(e) => setDistrictId(e.target.value)}
            required
            className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:border-blue-500"
          >
            <option value="">Select District</option>
            {districts.map((district) => (
              <option key={district.district_id} value={district.district_id}>
                {district.district_name}
              </option>
            ))}
          </select>
        </div>
        <div className="mb-2">
          <label className="block text-gray-700 font-semibold mb-1">Postcode:</label>
          <select
            value={postcodeId}
            onChange={(e) => setPostcodeId(e.target.value)}
            required
            className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:border-blue-500"
          >
            <option value="">Select Postcode</option>
            {postcodes.map((postcode) => (
              <option key={postcode.postcode_id} value={postcode.postcode_id}>
                {postcode.code}
              </option>
            ))}
          </select>
        </div>
        <div className="mb-2">
          <label className="block text-gray-700 font-semibold mb-1">Phone Number:</label>
          <input
            type="tel"
            value={phoneNumber}
            onChange={(e) => setPhoneNumber(e.target.value)}
            required
            pattern="[0-9]*"
            className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:border-blue-500"
          />
        </div>
        <div className="mb-2">
          <label className="block text-gray-700 font-semibold mb-1">Main Address:</label>
          <input
            type="checkbox"
            checked={isMain}
            onChange={(e) => setIsMain(e.target.checked)}
            className="mr-2 leading-tight"
          />
          <span className="text-gray-700">Set as main address</span>
        </div>
        <div className="mb-4 relative">
          <label className="block text-gray-700 font-semibold mb-1">Area:</label>
          <input
            type="text"
            value={areaInput}
            onChange={(e) => {
              setAreaInput(e.target.value);
              fetchAreas(e.target.value);
            }}
            placeholder="Search for an area"
            className="w-full px-3 py-2 border border-gray-300 rounded-lg focus:outline-none focus:border-blue-500" required
          />
          {areas.length > 0 && (
            <ul className="absolute z-10 w-full bg-white border border-gray-300 rounded-lg mt-1 max-h-48 overflow-y-auto">
              {areas.map((area) => (
                <li
                  key={`${area.id}-${area.name}`}
                  onClick={() => handleAreaSelect(area)}
                  className="px-4 py-2 cursor-pointer hover:bg-gray-200"
                >
                  {area.name}
                </li>
              ))}
            </ul>
          )}
        </div>
        <button
          type="submit"
          className="w-full bg-blue-500 text-white px-4 py-2 rounded-lg hover:bg-blue-600 focus:outline-none focus:bg-blue-600"
        >
          Create address
        </button>
      </form>
    </div>
  );
}
```

DASHBOARD *Admin*



Recent Transactions

Name	Date	Amount	Status
Juan	24.05.2023	Rp. 100.000	Paid

Top Products by Units Sold

Name	Price	Units Sold
Cupang Halfmoon	Rp. 50.000	98

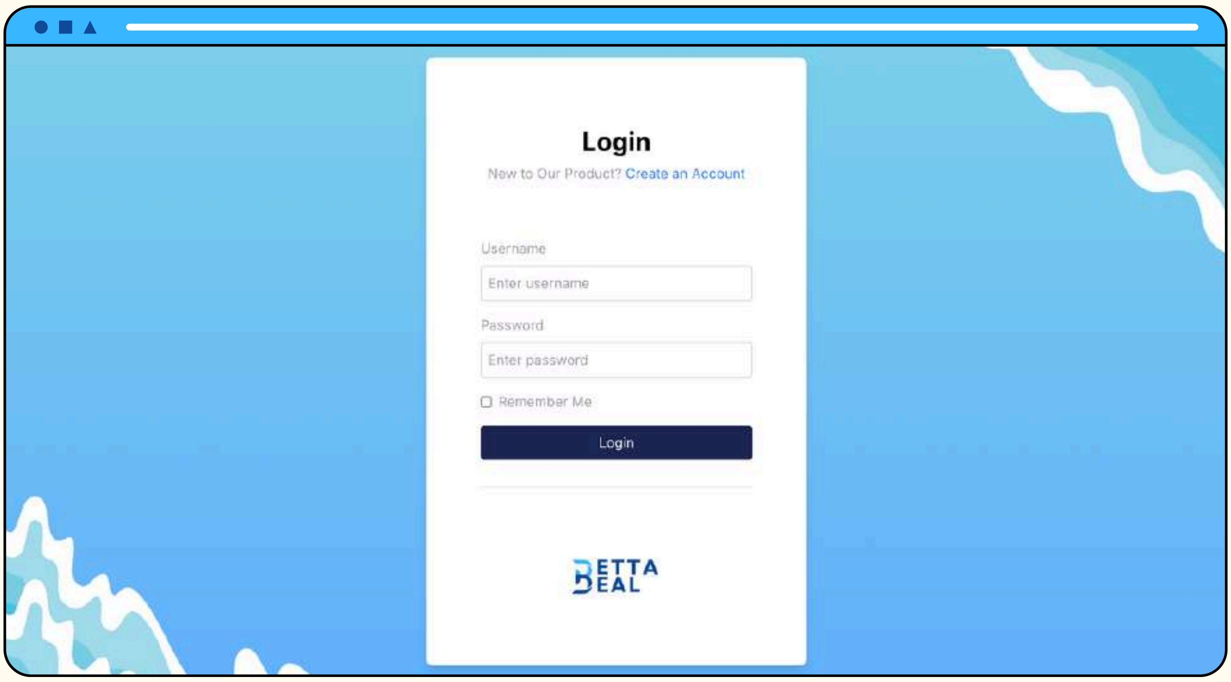
Bettabeal juga dilengkapi dengan sebuah website dashboard admin yang dirancang untuk mempermudah pengelolaan website customer dan aplikasi mobile. Melalui dashboard ini, admin dapat menambahkan, mengedit, atau menghapus produk serta artikel yang akan ditampilkan kepada pengguna. Selain itu, dashboard menyediakan fitur untuk mengelola order secara efisien, termasuk memantau status pesanan dan memproses transaksi. Dengan antarmuka yang intuitif dan fitur yang lengkap, website dashboard ini memberikan kontrol penuh kepada admin untuk memastikan operasional website dan aplikasi berjalan lancar serta memberikan pengalaman terbaik kepada pengguna.

Website Admin

1. Login Page Admin

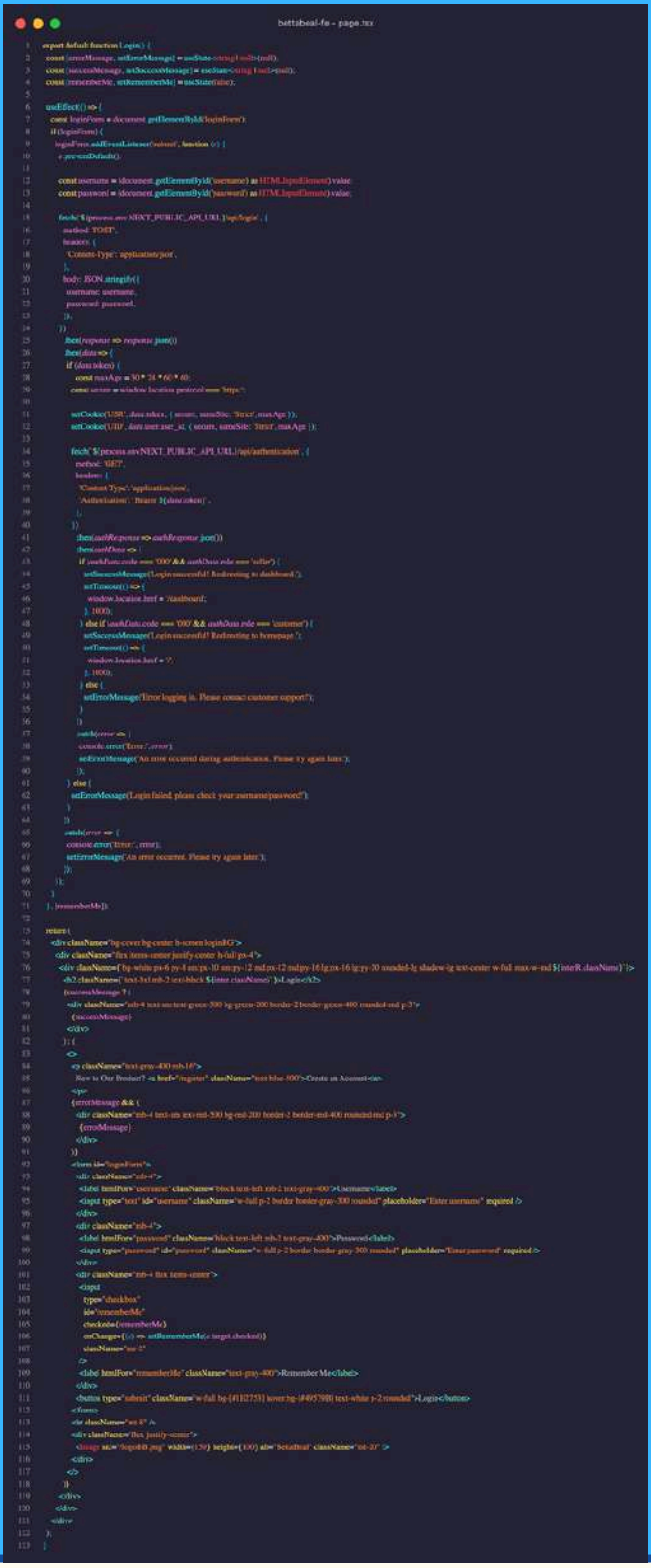
Halaman login admin ini dirancang khusus untuk memungkinkan akses hanya bagi admin. Admin dapat masuk dengan memasukkan username dan password yang telah terdaftar. Setelah berhasil login, admin dapat mengakses fitur-fitur khusus untuk mengelola aplikasi, seperti mengelola produk, artikel, pengguna, dan data lainnya yang hanya dapat diakses oleh admin.

Pageview
Login page



Website Customer

Source Code



bettabeal-fe – page.tsx

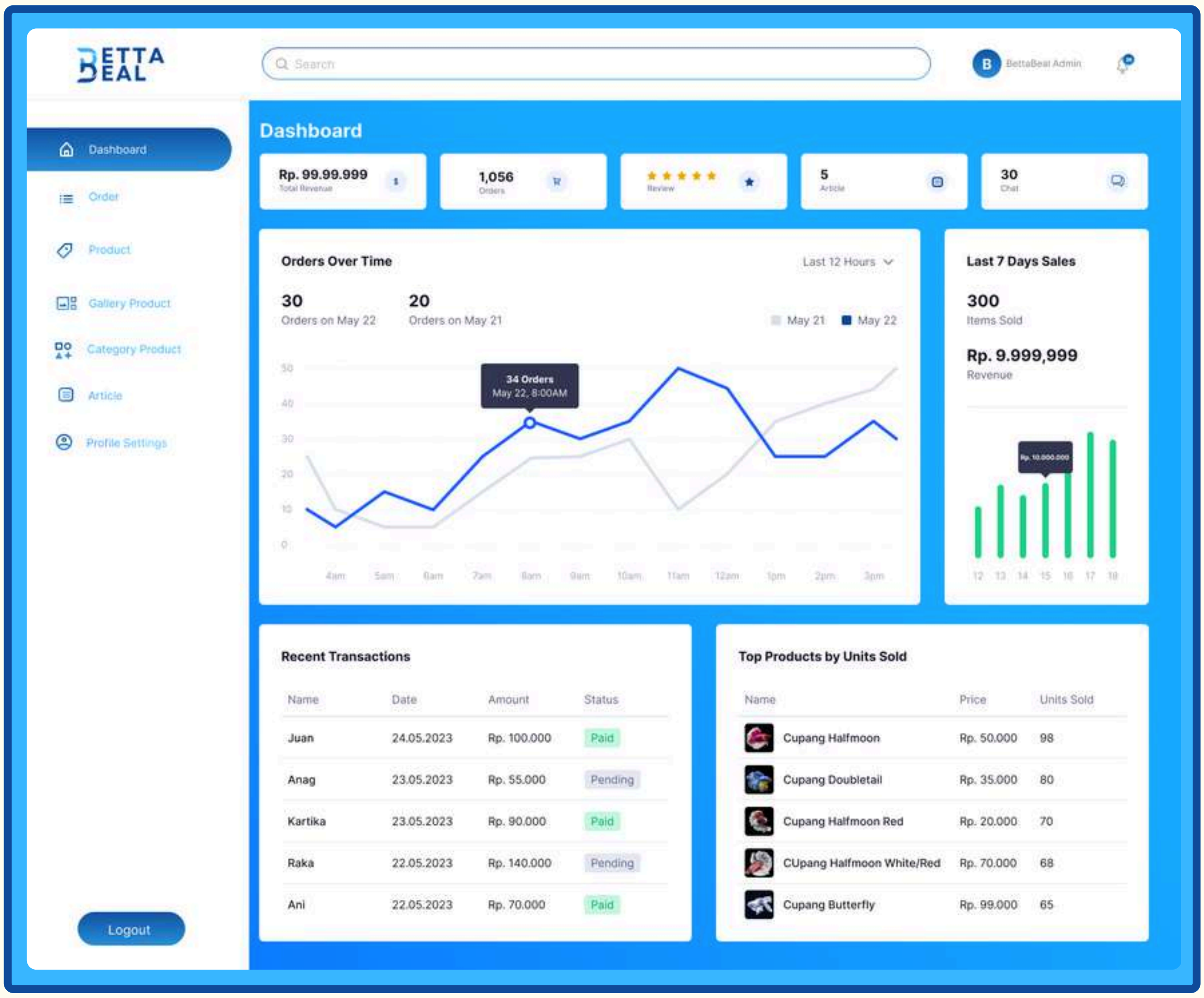
```
export default function Login() {
 const [errorMessage, setErrorMessage] = useState<string | null>(null);
 const [successMessage, setSuccessMessage] = useState<string | null>(null);
 const [rememberMe, setRememberMe] = useState(false);

 useEffect(() => {
  const loginForm = document.getElementById('loginForm');
  if (loginForm) {
   loginForm.addEventListener('submit', function (e) {
    e.preventDefault();

    const username = (document.getElementById('username') as HTMLInputElement).value;
    const password = (document.getElementById('password') as HTMLInputElement).value;

    fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/login`, {
     method: 'POST',
     headers: {
      'Content-Type': 'application/json',
     },
     body: JSON.stringify({
      username: username,
      password: password,
     }),
    })
     .then(response => response.json())
     .then(data => {
     if (data.token) {
       const maxAge = 30 * 24 * 60 * 60;
      const secure = window.location.protocol === 'https:';

      setCookie('USR', data.token, { secure, sameSite: 'Strict', maxAge });
      setCookie('UID', data.user.user_id, { secure, sameSite: 'Strict', maxAge });

      fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/authentication`, {
       method: 'GET',
       headers: {
        'Content-Type': 'application/json',
        'Authorization': `Bearer ${data.token}`,
       },
      })
       .then(authResponse => authResponse.json())
       .then(authData => {
        if (authData.code === '000' && authData.role === 'seller') {
         setSuccessMessage('Login successful! Redirecting to dashboard.');
         setTimeout(() => {
          window.location.href = '/dashboard';
         }, 1000);
        } else if (authData.code === '000' && authData.role === 'customer') {
         setSuccessMessage('Login successful! Redirecting to homepage.');
         setTimeout(() => {
          window.location.href = '/';
         }, 1000);
        } else {
         setErrorMessage('Error logging in. Please contact customer support!');
        }
       })
       .catch(error => {
        console.error('Error:', error);
        setErrorMessage('An error occurred during authentication. Please try again later.');
       });
     } else {
      setErrorMessage('Login failed, please check your username/password!');
     }
    })
    .catch(error => {
     console.error('Error:', error);
     setErrorMessage('An error occurred. Please try again later.');
    });
   });
  }
 }, [rememberMe]);

 return (
  <div className="bg-cover bg-center h-screen loginBG">
   <div className="flex items-center justify-center h-full px-4">
    <div className={`bg-white px-6 py-4 sm:px-10 sm:py-12 md:px-12 md:py-16 lg:px-16 lg:py-20 rounded-lg shadow-lg text-center w-full max-w-md ${interR.className}`}>
     <h2 className={`text-3xl mb-2 text-black ${inter.className}`}>Login</h2>
     {successMessage ? (
      <div className="mb-4 text-sm text-green-500 bg-green-200 border-2 border-green-400 rounded-md p-3">
       {successMessage}
      </div>
     ) : (
      <>
       <p className="text-gray-400 mb-10">
        New to Our Product? <a href="/register" className="text-blue-500">Create an Account</a>
       </p>
       {errorMessage && (
        <div className="mb-4 text-sm text-red-500 bg-red-200 border-2 border-red-400 rounded-md p-3">
         {errorMessage}
        </div>
       )}
       <form id="loginForm">
        <div className="mb-4">
         <label htmlFor="username" className="block text-left mb-2 text-gray-400">Username</label>
         <input type="text" id="username" className="w-full p-2 border border-gray-300 rounded" placeholder="Enter username" required />
        </div>
        <div className="mb-4">
         <label htmlFor="password" className="block text-left mb-2 text-gray-400">Password</label>
         <input type="password" id="password" className="w-full p-2 border border-gray-300 rounded" placeholder="Enter password" required />
        </div>
        <div className="mb-4 flex items-center">
         <input
          type="checkbox"
          id="rememberMe"
          checked={rememberMe}
          onChange={(e) => setRememberMe(e.target.checked)}
          className="mr-2"
         />
         <label htmlFor="rememberMe" className="text-gray-400">Remember Me</label>
        </div>
        <button type="submit" className="w-full bg-[#1B2731] hover:bg-[#495788] text-white p-2 rounded">Login</button>
       </form>
       <hr className="mt-8" />
       <div className="flex justify-center">
        <Image src="/logoBB.png" width={150} height={100} alt="Bettabeal" className="mt-20" />
       </div>
      </>
     )}
    </div>
   </div>
  </div>
 );
}
```

Website Admin

2. Dashboard Admin Page

Halaman dashboard admin ini memberikan gambaran umum yang komprehensif tentang kinerja aplikasi. Admin dapat melihat total revenue, total pesanan yang telah diproses, serta ulasan yang diberikan oleh pelanggan. Selain itu, halaman ini juga menampilkan artikel-artikel yang ada, diagram yang menggambarkan tren pesanan, transaksi terbaru, dan produk terlaris. Semua informasi ini disajikan dalam bentuk yang mudah dipahami untuk membantu admin dalam memantau dan mengelola operasional aplikasi secara efisien.

Pageview
Dashboard admin page



Recent Transactions

Name	Date	Amount	Status
Juan	24.05.2023	Rp. 100.000	Paid
Anag	23.05.2023	Rp. 55.000	Pending
Kartika	23.05.2023	Rp. 90.000	Paid
Raka	22.05.2023	Rp. 140.000	Pending
Ani	22.05.2023	Rp. 70.000	Paid

Top Products by Units Sold

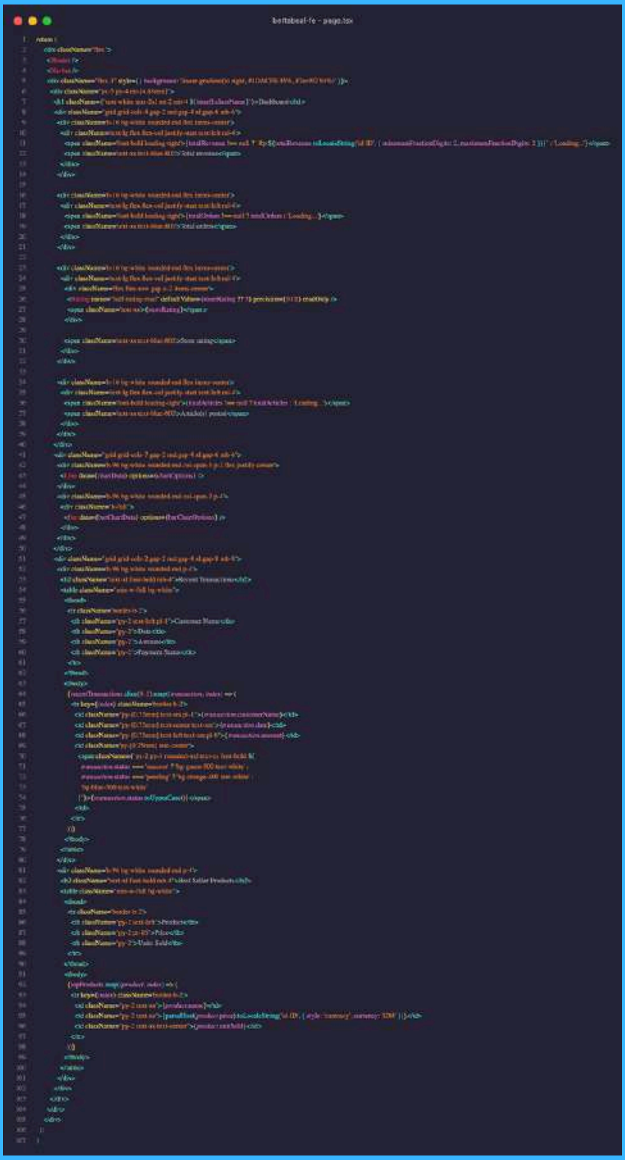
Name	Price	Units Sold
Cupang Halfmoon	Rp. 50.000	98
Cupang Doubletail	Rp. 35.000	80
Cupang Halfmoon Red	Rp. 20.000	70
CUpang Halfmoon White/Red	Rp. 70.000	68
Cupang Butterfly	Rp. 99.000	65

Website Customer

Source Code

```
[illegible]
```

```
[illegible]
```

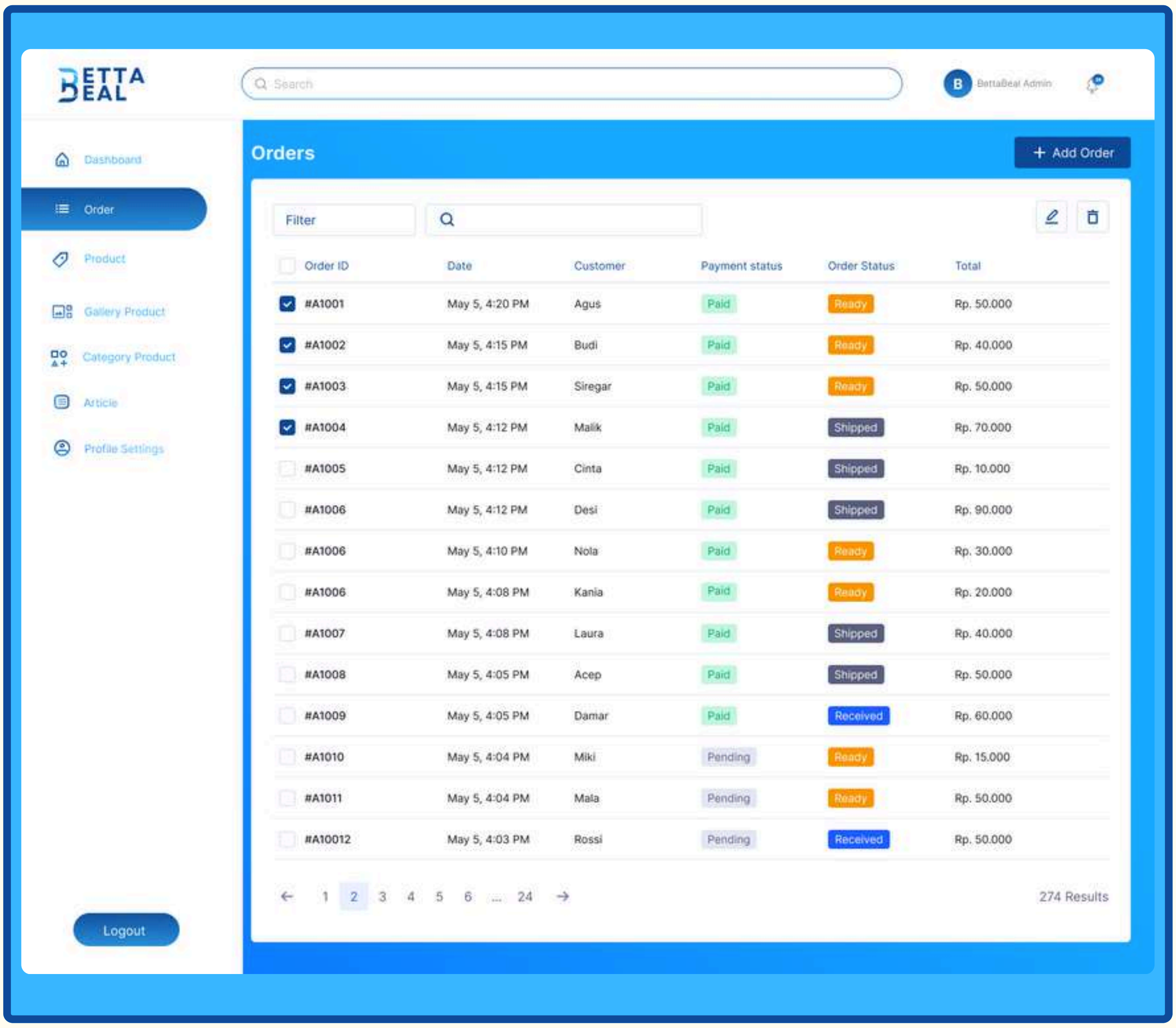

```
[illegible]
```

Website Admin

3. Orders Page

Halaman pesanan ini menampilkan daftar pesanan yang mencakup informasi seperti Order ID, tanggal pembelian, nama pelanggan, status pembayaran, status pesanan, dan total harga pesanan. Admin dapat memantau dan mengelola setiap pesanan dengan melihat rincian tersebut, memastikan kelancaran proses pemesanan dan pembayaran.

Pageview
Dashboard admin page



Orders

	Order ID	Date	Customer	Payment status	Order Status	Total
✓	#A1001	May 5, 4:20 PM	Agus	Paid	Ready	Rp. 50.000
✓	#A1002	May 5, 4:15 PM	Budi	Paid	Ready	Rp. 40.000
✓	#A1003	May 5, 4:15 PM	Siregar	Paid	Ready	Rp. 50.000
✓	#A1004	May 5, 4:12 PM	Malik	Paid	Shipped	Rp. 70.000
	#A1005	May 5, 4:12 PM	Cinta	Paid	Shipped	Rp. 10.000
	#A1006	May 5, 4:12 PM	Desi	Paid	Shipped	Rp. 90.000
	#A1006	May 5, 4:10 PM	Nola	Paid	Ready	Rp. 30.000
	#A1006	May 5, 4:08 PM	Kania	Paid	Ready	Rp. 20.000
	#A1007	May 5, 4:08 PM	Laura	Paid	Shipped	Rp. 40.000
	#A1008	May 5, 4:05 PM	Acep	Paid	Shipped	Rp. 50.000
	#A1009	May 5, 4:05 PM	Damar	Paid	Received	Rp. 60.000
	#A1010	May 5, 4:04 PM	Miki	Pending	Ready	Rp. 15.000
	#A1011	May 5, 4:04 PM	Mala	Pending	Ready	Rp. 50.000
	#A10012	May 5, 4:03 PM	Rossi	Pending	Received	Rp. 50.000

← 1 2 3 4 5 6 ... 24 → 274 Results

Website Customer

Source Code

betlabeal-fe – page.tsx

```
export default function Orders() {
[illegible]
```

betlabeal-fe – page.tsx

```
[illegible]
```

betlabeal-fe – page.tsx

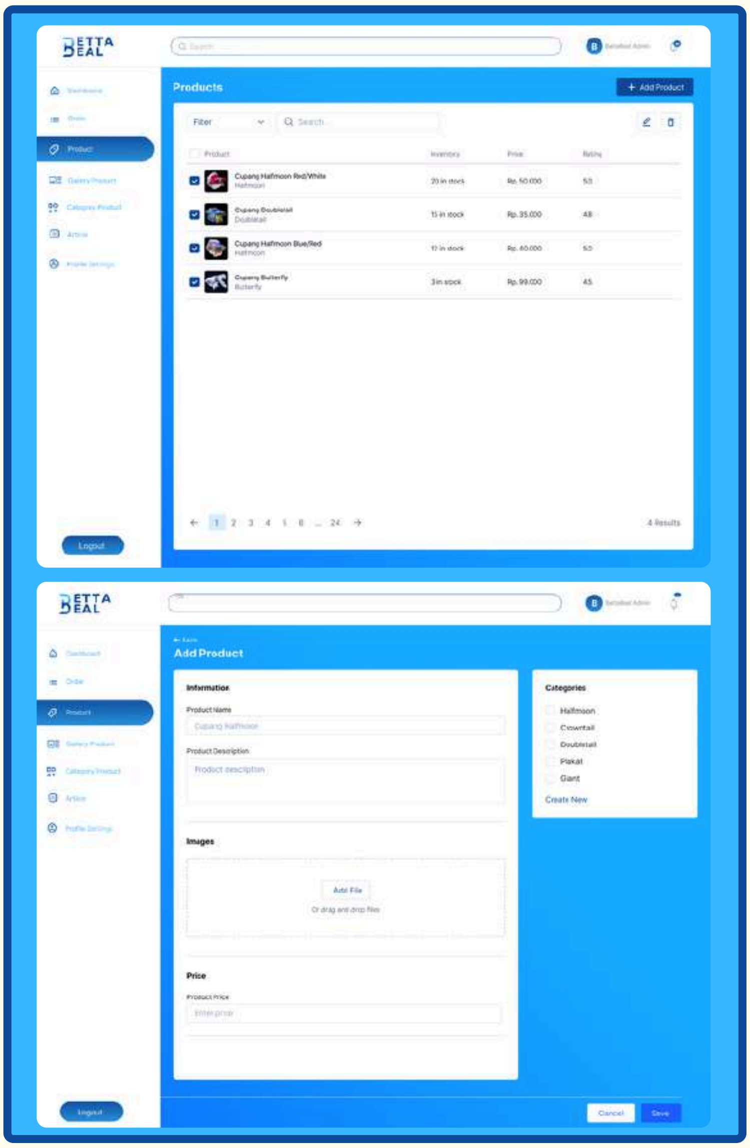
```
[illegible]
```

Website Admin

4. Product Page

Halaman produk admin memungkinkan admin untuk menambah produk baru yang akan ditampilkan di website pelanggan. Admin dapat memasukkan informasi produk, termasuk nama produk, deskripsi lengkap, foto produk, harga, dan kategori produk. Fitur ini mempermudah admin dalam mengelola dan memperbarui produk yang tersedia di platform secara efisien.

Pageview
Dashboard admin page



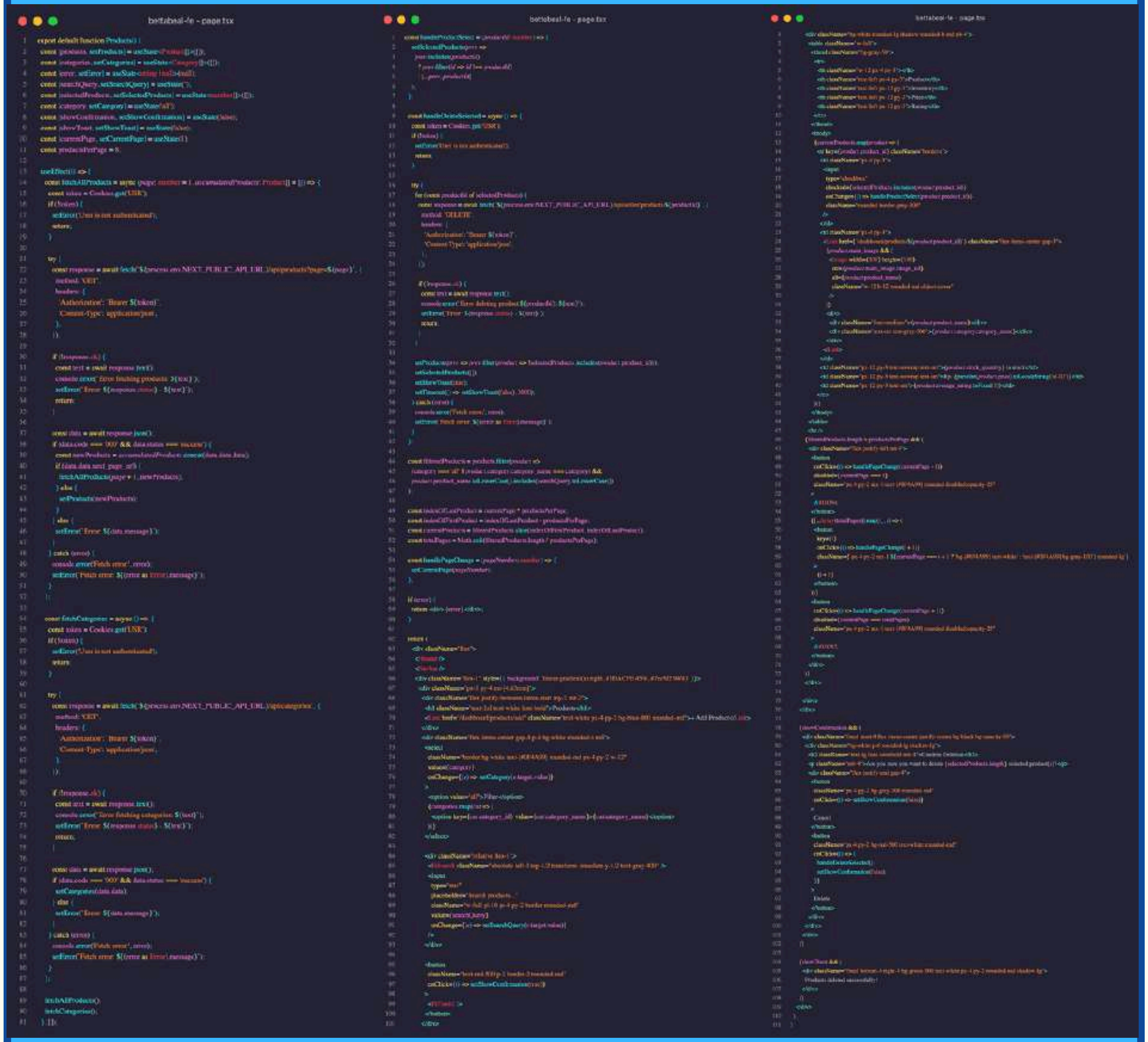
Product	Inventory	Price	Rating
Cupang Halfmoon Red/White Halfmoon	20 in stock	Rp. 50.000	5.0
Cupang Doubletail Doubletail	15 in stock	Rp. 35.000	4.8
Cupang Halfmoon Blue/Red Halfmoon	17 in stock	Rp. 60.000	5.0
Cupang Butterfly Butterfly	3 in stock	Rp. 99.000	4.5

← 1 2 3 4 5 6 ... 24 → 4 Results

Website Customer

Source Code

Product page



bettabeal-fe - page.tsx

```tsx
export default function Products() {
  const [products, setProducts] = useState<Product[]>([]);
  const [categories, setCategories] = useState<Category[]>([]);
  const [error, setError] = useState<string | null>(null);
  const [searchQuery, setSearchQuery] = useState('');
  const [selectedProducts, setSelectedProducts] = useState<number[]>([]);
  const [category, setCategory] = useState('all');
  const [showConfirmation, setShowConfirmation] = useState(false);
  const [showToast, setShowToast] = useState(false);
  const [currentPage, setCurrentPage] = useState(1);
  const productsPerPage = 8;

  useEffect(() => {
    const fetchAllProducts = async (page: number = 1, accumulatedProducts: Product[] = []) => {
      const token = Cookies.get('USR');
      if (!token) {
        setError('User is not authenticated');
        return;
      }

      try {
        const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/products?page=${page}`, {
          method: 'GET',
          headers: {
            'Authorization': `Bearer ${token}`,
            'Content-Type': 'application/json',
          },
        });

        if (!response.ok) {
          const text = await response.text();
          console.error(`Error fetching products: ${text}`);
          setError(`Error: ${response.status} - ${text}`);
          return;
        }

        const data = await response.json();
        if (data.code === '000' && data.status === 'success') {
          const newProducts = accumulatedProducts.concat(data.data.data);
          if (data.data.next_page_url) {
            fetchAllProducts(page + 1, newProducts);
          } else {
            setProducts(newProducts);
          }
        } else {
          setError(`Error: ${data.message}`);
        }
      } catch (error) {
        console.error('Fetch error:', error);
        setError(`Fetch error: ${(error as Error).message}`);
      }
    };

    const fetchCategories = async () => {
      const token = Cookies.get('USR');
      if (!token) {
        setError('User is not authenticated');
        return;
      }

      try {
        const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/categories`, {
          method: 'GET',
          headers: {
            'Authorization': `Bearer ${token}`,
            'Content-Type': 'application/json',
          },
        });

        if (!response.ok) {
          const text = await response.text();
          console.error(`Error fetching categories: ${text}`);
          setError(`Error: ${response.status} - ${text}`);
          return;
        }

        const data = await response.json();
        if (data.code === '000' && data.status === 'success') {
          setCategories(data.data);
        } else {
          setError(`Error: ${data.message}`);
        }
      } catch (error) {
        console.error('Fetch error:', error);
        setError(`Fetch error: ${(error as Error).message}`);
      }
    };

    fetchAllProducts();
    fetchCategories();
  }, []);
```

bettabeal-fe - page.tsx

```tsx
const handleProductSelect = (productId: number) => {
  setSelectedProducts(prev =>
    prev.includes(productId)
      ? prev.filter(id => id !== productId)
      : [...prev, productId]
  );
};

const handleDeleteSelected = async () => {
  const token = Cookies.get('USR');
  if (!token) {
    setError('User is not authenticated');
    return;
  }

  try {
    for (const productId of selectedProducts) {
      const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/seller/products/${productId}`, {
        method: 'DELETE',
        headers: {
          'Authorization': `Bearer ${token}`,
          'Content-Type': 'application/json',
        },
      });

      if (!response.ok) {
        const text = await response.text();
        console.error(`Error deleting product ${productId}: ${text}`);
        setError(`Error: ${response.status} - ${text}`);
        return;
      }
    }

    setProducts(prev => prev.filter(product => !selectedProducts.includes(product.product_id)));
    setSelectedProducts([]);
    setShowToast(true);
    setTimeout(() => setShowToast(false), 3000);
  } catch (error) {
    console.error('Fetch error:', error);
    setError(`Fetch error: ${(error as Error).message}`);
  }
};

const filteredProducts = products.filter(product =>
  (category === 'all' || product.category.category_name === category) &&
  product.product_name.toLowerCase().includes(searchQuery.toLowerCase())
);

const indexOfLastProduct = currentPage * productsPerPage;
const indexOfFirstProduct = indexOfLastProduct - productsPerPage;
const currentProducts = filteredProducts.slice(indexOfFirstProduct, indexOfLastProduct);
const totalPages = Math.ceil(filteredProducts.length / productsPerPage);

const handlePageChange = (pageNumber: number) => {
  setCurrentPage(pageNumber);
};

if (error) {
  return <div>{error}</div>;
}

return (
  <div className="flex">
    <[illegible] />
    <[illegible] />
    <div className="flex-1" style={{ background: 'linear-gradient(to right, #[illegible], #[illegible])' }}>
      <div className="px-[illegible] py-4 [illegible]">
        <div className="flex justify-between items-start mb-[illegible] mt-[illegible]">
          <h1 className="text-2xl text-white font-bold">Products</h1>
          <Link href="/dashboard/products/add" className="text-white px-4 py-2 bg-[illegible] rounded-md">+ Add Product</Link>
        </div>
        <div className="flex items-center gap-4 p-4 bg-white rounded-t-md">
          <select
            className="border bg-white text-[#[illegible]] rounded-md px-4 py-2 w-[illegible]"
            value={category}
            onChange={(e) => setCategory(e.target.value)}
          >
            <option value="all">Filter</option>
            {categories.map((cat) => (
              <option key={cat.category_id} value={cat.category_name}>{cat.category_name}</option>
            ))}
          </select>

          <div className="relative flex-1">
            <[illegible] className="absolute left-3 top-1/2 transform -translate-y-1/2 text-gray-400" />
            <input
              type="text"
              placeholder="Search products..."
              className="w-full pl-10 pr-4 py-2 border rounded-md"
              value={searchQuery}
              onChange={(e) => setSearchQuery(e.target.value)}
            />
          </div>

          <button
            className="text-red-500 p-2 border-2 rounded-md"
            onClick={() => setShowConfirmation(true)}
          >
            <[illegible] />
          </button>
        </div>
```

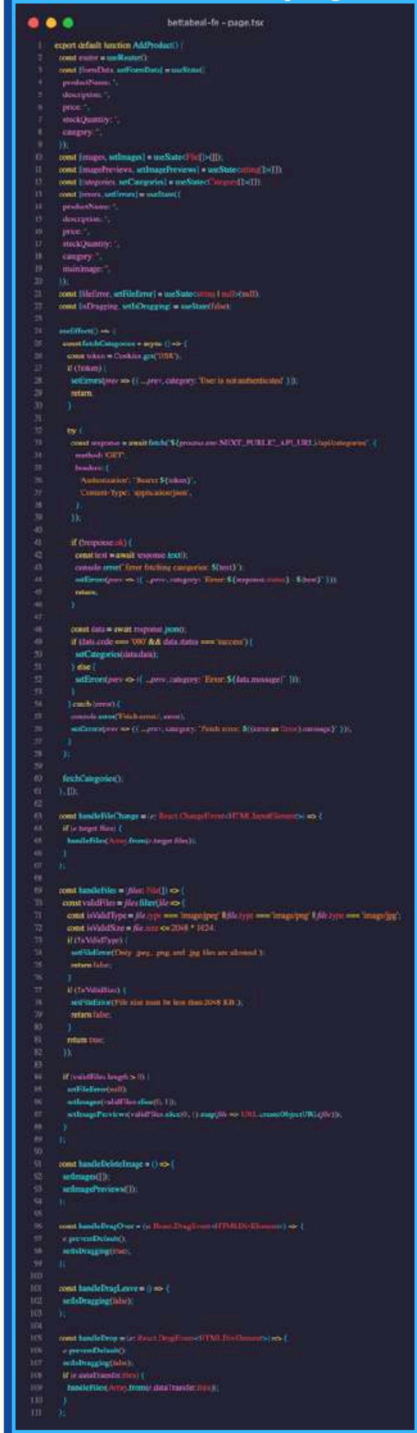
bettabeal-fe - page.tsx

```tsx
<div className="bg-white rounded-b-lg shadow-md overflow-x-auto">
  <table className="w-full">
    <thead className="bg-gray-50">
      <tr>
        <th className="w-12 px-4 py-3"></th>
        <th className="text-left px-4 py-3">Products</th>
        <th className="text-left px-12 py-3">Inventory</th>
        <th className="text-left px-12 py-3">Price</th>
        <th className="text-left px-12 py-3">Rating</th>
      </tr>
    </thead>
    <tbody>
      {currentProducts.map(product => (
        <tr key={product.product_id} className="border-t">
          <td className="px-4 py-3">
            <input
              type="checkbox"
              checked={selectedProducts.includes(product.product_id)}
              onChange={() => handleProductSelect(product.product_id)}
              className="rounded border-gray-300"
            />
          </td>
          <td className="px-4 py-3">
            <Link href={`/dashboard/products/${product.product_id}`} className="flex items-center gap-3">
              {product.main_image && (
                <Image width={100} height={100}
                  src={product.main_image.image_url}
                  alt={product.product_name}
                  className="w-12 h-12 rounded-md object-cover"
                />
              )}
              <div>
                <div className="font-medium">{product.product_name}</div>
                <div className="text-sm text-gray-500">{product.category.category_name}</div>
              </div>
            </Link>
          </td>
          <td className="px-12 py-3 text-nowrap text-left">{product.stock_quantity} in stock</td>
          <td className="px-12 py-3 text-nowrap text-left">Rp. {parseInt(product.price).toLocaleString('id-ID')}</td>
          <td className="px-12 py-3 text-left">{product.average_rating ?? '0.0'}</td>
        </tr>
      ))}
    </tbody>
  </table>
  <hr />
  {filteredProducts.length > productsPerPage && (
    <div className="flex justify-end mt-4">
      <button
        onClick={() => handlePageChange(currentPage - 1)}
        disabled={currentPage === 1}
        className="px-4 py-2 mx-1 text-[#[illegible]] rounded disabled:opacity-25"
      >
        &lt; [illegible]
      </button>
      {[...Array(totalPages)].map((_, i) => (
        <button
          key={i}
          onClick={() => handlePageChange(i + 1)}
          className={`px-4 py-2 mx-1 ${currentPage === i + 1 ? 'bg-[#[illegible]] text-white' : 'text-[#[illegible]] bg-gray-[illegible]'} rounded-lg`}
        >
          {i + 1}
        </button>
      ))}
      <button
        onClick={() => handlePageChange(currentPage + 1)}
        disabled={currentPage === totalPages}
        className="px-4 py-2 mx-1 text-[#[illegible]] rounded disabled:opacity-25"
      >
        [illegible] &gt;
      </button>
    </div>
  )}
</div>

</div>
</div>

{showConfirmation && (
  <div className="fixed inset-0 flex items-center justify-center bg-black bg-opacity-50">
    <div className="bg-white p-6 rounded-lg shadow-lg">
      <h2 className="text-lg font-semibold mb-4">Confirm Deletion</h2>
      <p className="mb-4">Are you sure you want to delete {selectedProducts.length} selected product(s)?</p>
      <div className="flex justify-end gap-4">
        <button
          className="px-4 py-2 bg-gray-300 rounded-md"
          onClick={() => setShowConfirmation(false)}
        >
          Cancel
        </button>
        <button
          className="px-4 py-2 bg-red-500 text-white rounded-md"
          onClick={() => {
            handleDeleteSelected();
            setShowConfirmation(false);
          }}
        >
          Delete
        </button>
      </div>
    </div>
  </div>
)}

{showToast && (
  <div className="fixed bottom-4 right-4 bg-green-500 text-white px-4 py-2 rounded-md shadow-lg">
    Products deleted successfully!
  </div>
)}
</div>
);
}
```

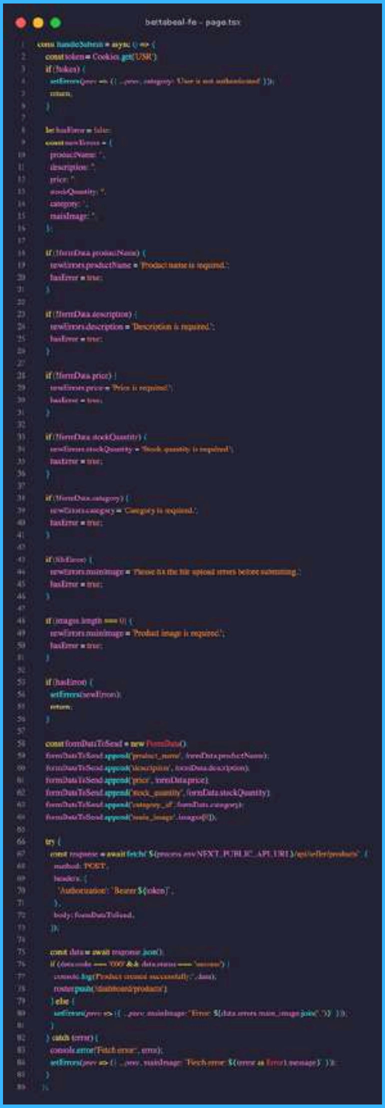
Website Customer

Source Code

Add Product page


bettabeal-fe – page.tsx

```
export default function AddProduct() {
  const router = useRouter();
  const [formData, setFormData] = useState({
    productName: '',
    description: '',
    price: '',
    stockQuantity: '',
    category: '',
  });
  const [images, setImages] = useState<File[]>([]);
  const [imagePreviews, setImagePreviews] = useState<string[]>([]);
  const [categories, setCategories] = useState<Category[]>([]);
  const [errors, setErrors] = useState({
    productName: '',
    description: '',
    price: '',
    stockQuantity: '',
    category: '',
    mainImage: '',
  });
  const [fileError, setFileError] = useState<string | null>(null);
  const [isDragging, setIsDragging] = useState(false);

  useEffect(() => {
    const fetchCategories = async () => {
      const token = Cookies.get('USR');
      if (!token) {
        setErrors(prev => ({ ...prev, category: 'User is not authenticated' }));
        return;
      }

      try {
        const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/categories`, {
          method: 'GET',
          headers: {
            'Authorization': `Bearer ${token}`,
            'Content-Type': 'application/json',
          },
        });

        if (!response.ok) {
          const text = await response.text();
          console.error(`Error fetching categories: ${text}`);
          setErrors(prev => ({ ...prev, category: `Error: ${response.status} - ${text}` }));
          return;
        }

        const data = await response.json();
        if (data.code === '000' && data.status === 'success') {
          setCategories(data.data);
        } else {
          setErrors(prev => ({ ...prev, category: `Error: ${data.message}` }));
        }
      } catch (error) {
        console.error('Fetch error:', error);
        setErrors(prev => ({ ...prev, category: `Fetch error: ${(error as Error).message}` }));
      }
    };

    fetchCategories();
  }, []);

  const handleFileChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    if (e.target.files) {
      handleFiles(Array.from(e.target.files));
    }
  };

  const handleFiles = (files: File[]) => {
    const validFiles = files.filter(file => {
      const isValidType = file.type === 'image/jpeg' || file.type === 'image/png' || file.type === 'image/jpg';
      const isValidSize = file.size <= 2048 * 1024;
      if (!isValidType) {
        setFileError('Only .jpeg, .png, and .jpg files are allowed.');
        return false;
      }
      if (!isValidSize) {
        setFileError('File size must be less than 2048 KB.');
        return false;
      }
      return true;
    });

    if (validFiles.length > 0) {
      setFileError(null);
      setImages(validFiles.slice(0, 1));
      setImagePreviews(validFiles.slice(0, 1).map(file => URL.createObjectURL(file)));
    }
  };

  const handleDeleteImage = () => {
    setImages([]);
    setImagePreviews([]);
  };

  const handleDragOver = (e: React.DragEvent<HTMLDivElement>) => {
    e.preventDefault();
    setIsDragging(true);
  };

  const handleDragLeave = () => {
    setIsDragging(false);
  };

  const handleDrop = (e: React.DragEvent<HTMLDivElement>) => {
    e.preventDefault();
    setIsDragging(false);
    if (e.dataTransfer.files) {
      handleFiles(Array.from(e.dataTransfer.files));
    }
  };
```


bettabeal-fe – page.tsx

```
const handleSubmit = async () => {
  const token = Cookies.get('USR');
  if (!token) {
    setErrors(prev => ({ ...prev, category: 'User is not authenticated' }));
    return;
  }

  let hasError = false;
  const newErrors = {
    productName: '',
    description: '',
    price: '',
    stockQuantity: '',
    category: '',
    mainImage: '',
  };

  if (!formData.productName) {
    newErrors.productName = 'Product name is required.';
    hasError = true;
  }

  if (!formData.description) {
    newErrors.description = 'Description is required.';
    hasError = true;
  }

  if (!formData.price) {
    newErrors.price = 'Price is required.';
    hasError = true;
  }

  if (!formData.stockQuantity) {
    newErrors.stockQuantity = 'Stock quantity is required.';
    hasError = true;
  }

  if (!formData.category) {
    newErrors.category = 'Category is required.';
    hasError = true;
  }

  if (fileError) {
    newErrors.mainImage = 'Please fix the file upload errors before submitting.';
    hasError = true;
  }

  if (images.length === 0) {
    newErrors.mainImage = 'Product image is required.';
    hasError = true;
  }

  if (hasError) {
    setErrors(newErrors);
    return;
  }

  const formDataToSend = new FormData();
  formDataToSend.append('product_name', formData.productName);
  formDataToSend.append('description', formData.description);
  formDataToSend.append('price', formData.price);
  formDataToSend.append('stock_quantity', formData.stockQuantity);
  formDataToSend.append('category_id', formData.category);
  formDataToSend.append('main_image', images[0]);

  try {
    const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/seller/products`, {
      method: 'POST',
      headers: {
        'Authorization': `Bearer ${token}`,
      },
      body: formDataToSend,
    });

    const data = await response.json();
    if (data.code === '000' && data.status === 'success') {
      console.log('Product created successfully:', data);
      router.push('/dashboard/products');
    } else {
      setErrors(prev => ({ ...prev, mainImage: `Error: ${data.errors.main_image.join(', ')}` }));
    }
  } catch (error) {
    console.error('Fetch error:', error);
    setErrors(prev => ({ ...prev, mainImage: `Fetch error: ${(error as Error).message}` }));
  }
};
```

bettabeal-fe – page.tsx

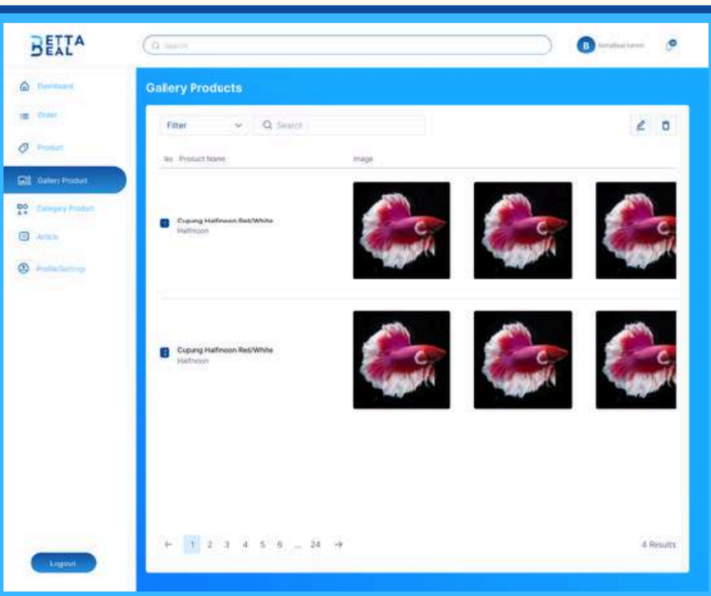
```
return (
  <div className="flex">
    [illegible]
```

Website Admin

5. Gallery Product Page

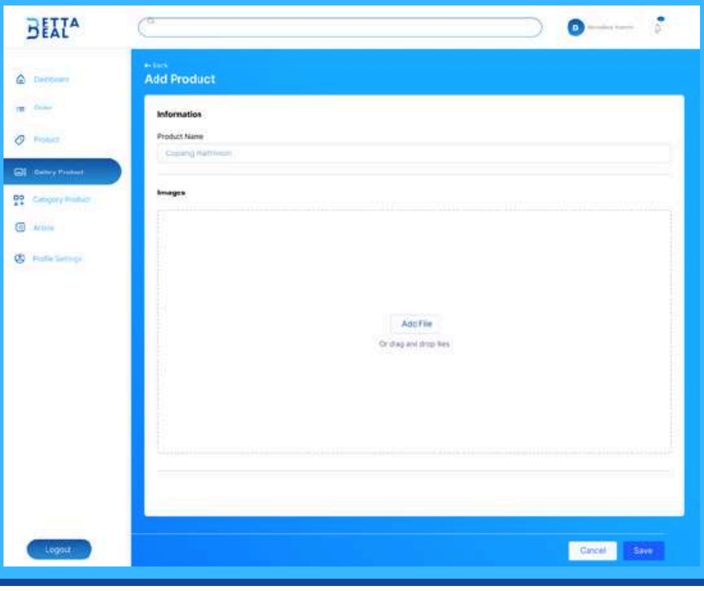
Halaman galeri produk menampilkan berbagai gambar produk yang telah ditambahkan, lengkap dengan nama produk dan beberapa foto yang menggambarkan produk dari berbagai sudut. Gambar-gambar ini akan ditampilkan di website pelanggan, memungkinkan pelanggan melihat detail produk secara lebih jelas dan lengkap sebelum melakukan pembelian.

Pageview
Gallery product page



Gallery Products

No	Product Name	Image
	Cupang Halfmoon Red/White Halfmoon	
	Cupang Halfmoon Red/White Halfmoon	



Add Product

Information

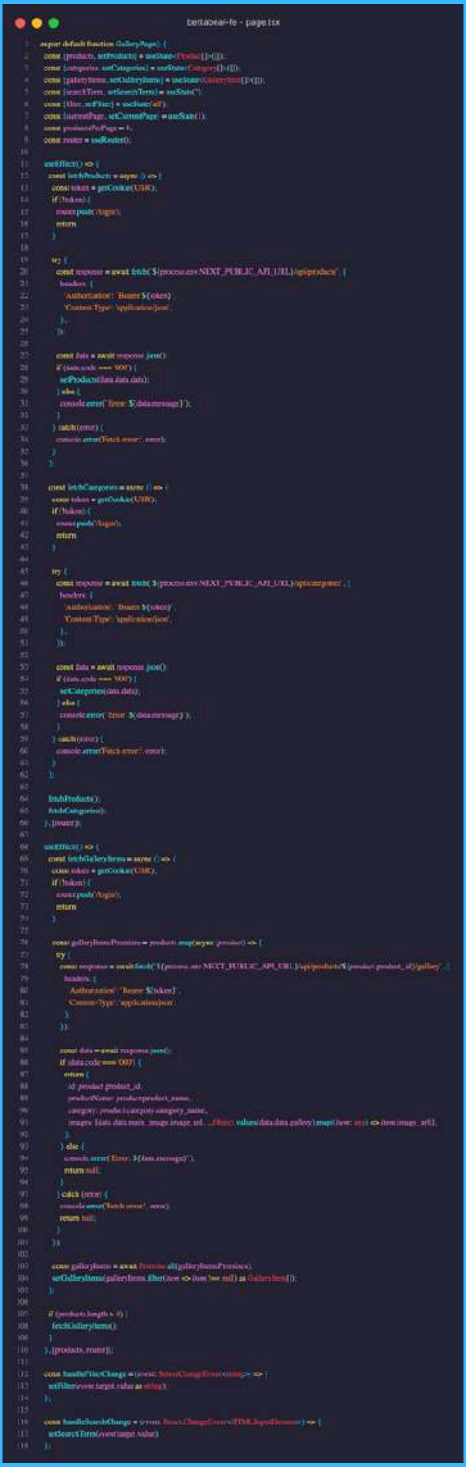
Product Name: Cupang Halfmoon

Images

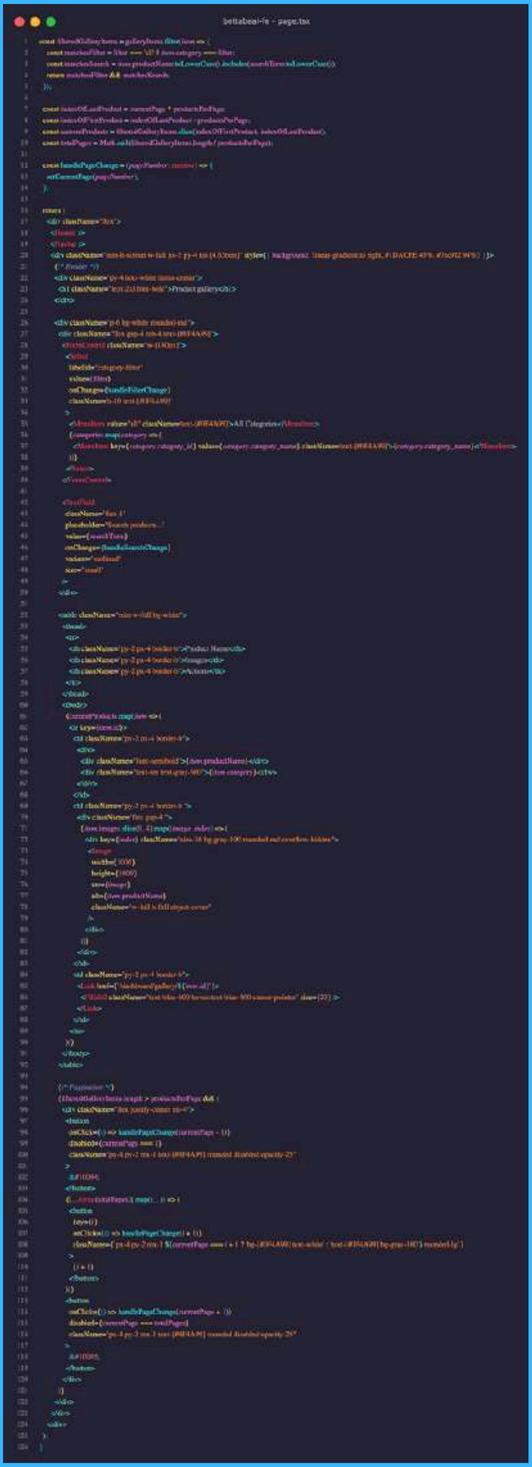
Website Customer

Source Code

Gallery product page



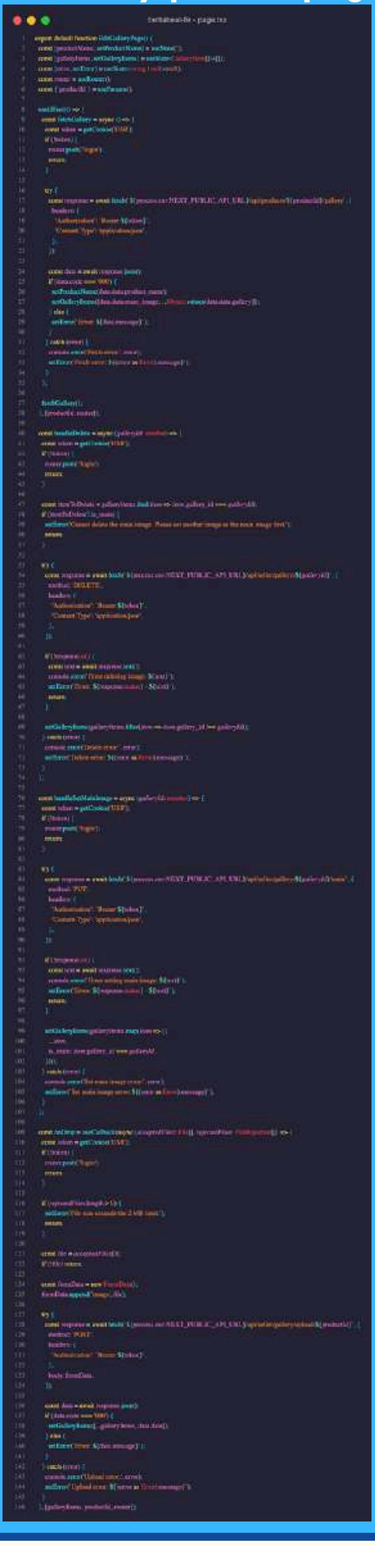
```
export default function GalleryPage() {
  const [products, setProducts] = useState<Product[]>([]);
  const [categories, setCategories] = useState<Category[]>([]);
  const [galleryItems, setGalleryItems] = useState<GalleryItem[]>([]);
  const [searchTerm, setSearchTerm] = useState('');
  const [filter, setFilter] = useState('all');
  const [currentPage, setCurrentPage] = useState(1);
  const productsPerPage = 5;
  const router = useRouter();

  useEffect(() => {
    const fetchProducts = async () => {
      const token = getCookie('UBK');
      if (!token) {
        router.push('/login');
        return;
      }

      try {
        const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/products`, {
          headers: {
            'Authorization': `Bearer ${token}`,
            'Content-Type': 'application/json',
          },
        });

        const data = await response.json();
        if (data.code === '000') {
          setProducts(data.data.data);
        } else {
          console.error(`Error: ${data.message}`);
        }
      } catch (error) {
        console.error('Fetch error:', error);
      }
    };

    const fetchCategories = async () => {
      const token = getCookie('UBK');
      if (!token) {
        router.push('/login');
        return;
      }

      try {
        const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/categories`, {
          headers: {
            'Authorization': `Bearer ${token}`,
            'Content-Type': 'application/json',
          },
        });

        const data = await response.json();
        if (data.code === '000') {
          setCategories(data.data);
        } else {
          console.error(`Error: ${data.message}`);
        }
      } catch (error) {
        console.error('Fetch error:', error);
      }
    };

    fetchProducts();
    fetchCategories();
  }, [router]);

  useEffect(() => {
    const fetchGalleryItems = async () => {
      const token = getCookie('UBK');
      if (!token) {
        router.push('/login');
        return;
      }

      const galleryItemsPromises = products.map(async (product) => {
        try {
          const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/products/${product.product_id}/gallery`, {
            headers: {
              'Authorization': `Bearer ${token}`,
              'Content-Type': 'application/json',
            },
          });

          const data = await response.json();
          if (data.code === '000') {
            return {
              id: product.product_id,
              productName: product.product_name,
              category: product.category.category_name,
              images: [data.data.main_image.image_url, ...Object.values(data.data.gallery).map((item: any) => item.image_url)],
            };
          } else {
            console.error(`Error: ${data.message}`);
            return null;
          }
        } catch (error) {
          console.error('Fetch error:', error);
          return null;
        }
      });

      const galleryItems = await Promise.all(galleryItemsPromises);
      setGalleryItems(galleryItems.filter(item => item !== null) as GalleryItem[]);
    };

    if (products.length > 0) {
      fetchGalleryItems();
    }
  }, [products, router]);

  const handleFilterChange = (event: SelectChangeEvent<string>) => {
    setFilter(event.target.value as string);
  };

  const handleSearchChange = (event: React.ChangeEvent<HTMLInputElement>) => {
    setSearchTerm(event.target.value);
  };
```



```
  const filteredGalleryItems = galleryItems.filter(item => {
    const matchesFilter = filter === 'all' || item.category === filter;
    const matchesSearch = item.productName.toLowerCase().includes(searchTerm.toLowerCase());
    return matchesFilter && matchesSearch;
  });

  const indexOfLastProduct = currentPage * productsPerPage;
  const indexOfFirstProduct = indexOfLastProduct - productsPerPage;
  const currentProducts = filteredGalleryItems.slice(indexOfFirstProduct, indexOfLastProduct);
  const totalPages = Math.ceil(filteredGalleryItems.length / productsPerPage);

  const handlePageChange = (pageNumber: number) => {
    setCurrentPage(pageNumber);
  };

  return (
    <div className="flex">
      <Header />
      <Navbar />
      <div className="min-h-screen w-full px-2 py-4 mt-[4.63rem]" style={{ background: 'linear-gradient(to right, #1DACFE 45%, #7cc9f2 99%)' }}>
        {/* Header */}
        <div className="py-4 text-white items-center">
          <h1 className="text-2xl font-bold">Product gallery</h1>
        </div>

        <div className="p-6 bg-white rounded-md">
          <div className="flex gap-4 mb-4 text-[#0F4A99]">
            <FormControl className="w-[100%]">
              <Select
                labelId="category-filter"
                value={filter}
                onChange={handleFilterChange}
                className="h-10 text-[#0F4A99]"
              >
                <MenuItem value="all" className="text-[#0F4A99]">All Categories</MenuItem>
                {categories.map(category => (
                  <MenuItem key={category.category_id} value={category.category_name} className="text-[#0F4A99]">{category.category_name}</MenuItem>
                ))}
              </Select>
            </FormControl>

            <TextField
              className="flex-1"
              placeholder="Search products..."
              value={searchTerm}
              onChange={handleSearchChange}
              variant="outlined"
              size="small"
            />
          </div>

          <table className="min-w-full bg-white">
            <thead>
              <tr>
                <th className="py-2 px-4 border-b">Product Name</th>
                <th className="py-2 px-4 border-b">Images</th>
                <th className="py-2 px-4 border-b">Actions</th>
              </tr>
            </thead>
            <tbody>
              {currentProducts.map(item => (
                <tr key={item.id}>
                  <td className="py-2 px-4 border-b">
                    <div>
                      <div className="font-semibold">{item.productName}</div>
                      <div className="text-sm text-gray-500">{item.category}</div>
                    </div>
                  </td>
                  <td className="py-2 px-4 border-b">
                    <div className="flex gap-4">
                      {item.images.slice(0, 4).map((image, index) => (
                        <div key={index} className="size-16 bg-gray-100 rounded-md overflow-hidden">
                          <Image
                            width={1000}
                            height={1000}
                            src={image}
                            alt={item.productName}
                            className="w-full h-full object-cover"
                          />
                        </div>
                      ))}
                    </div>
                  </td>
                  <td className="py-2 px-4 border-b">
                    <Link href={`/dashboard/gallery/${item.id}`}>
                      <FiEdit2 className="text-blue-600 hover:text-blue-800 cursor-pointer" size={20} />
                    </Link>
                  </td>
                </tr>
              ))}
            </tbody>
          </table>

          {/* Pagination */}
          {filteredGalleryItems.length > productsPerPage && (
            <div className="flex justify-center mt-4">
              <button
                onClick={() => handlePageChange(currentPage - 1)}
                disabled={currentPage === 1}
                className="px-4 py-2 mx-1 text-[#0F4A99] rounded disabled:opacity-25"
              >
                &#10094;
              </button>
              {[...Array(totalPages)].map((_, i) => (
                <button
                  key={i}
                  onClick={() => handlePageChange(i + 1)}
                  className={`px-4 py-2 mx-1 ${currentPage === i + 1 ? 'bg-[#0F4A99] text-white' : 'text-[#0F4A99] bg-gray-100'} rounded-lg`}
                >
                  {i + 1}
                </button>
              ))}
              <button
                onClick={() => handlePageChange(currentPage + 1)}
                disabled={currentPage === totalPages}
                className="px-4 py-2 mx-1 text-[#0F4A99] rounded disabled:opacity-25"
              >
                &#10095;
              </button>
            </div>
          )}
        </div>
      </div>
    </div>
  );
}
```

Website Customer

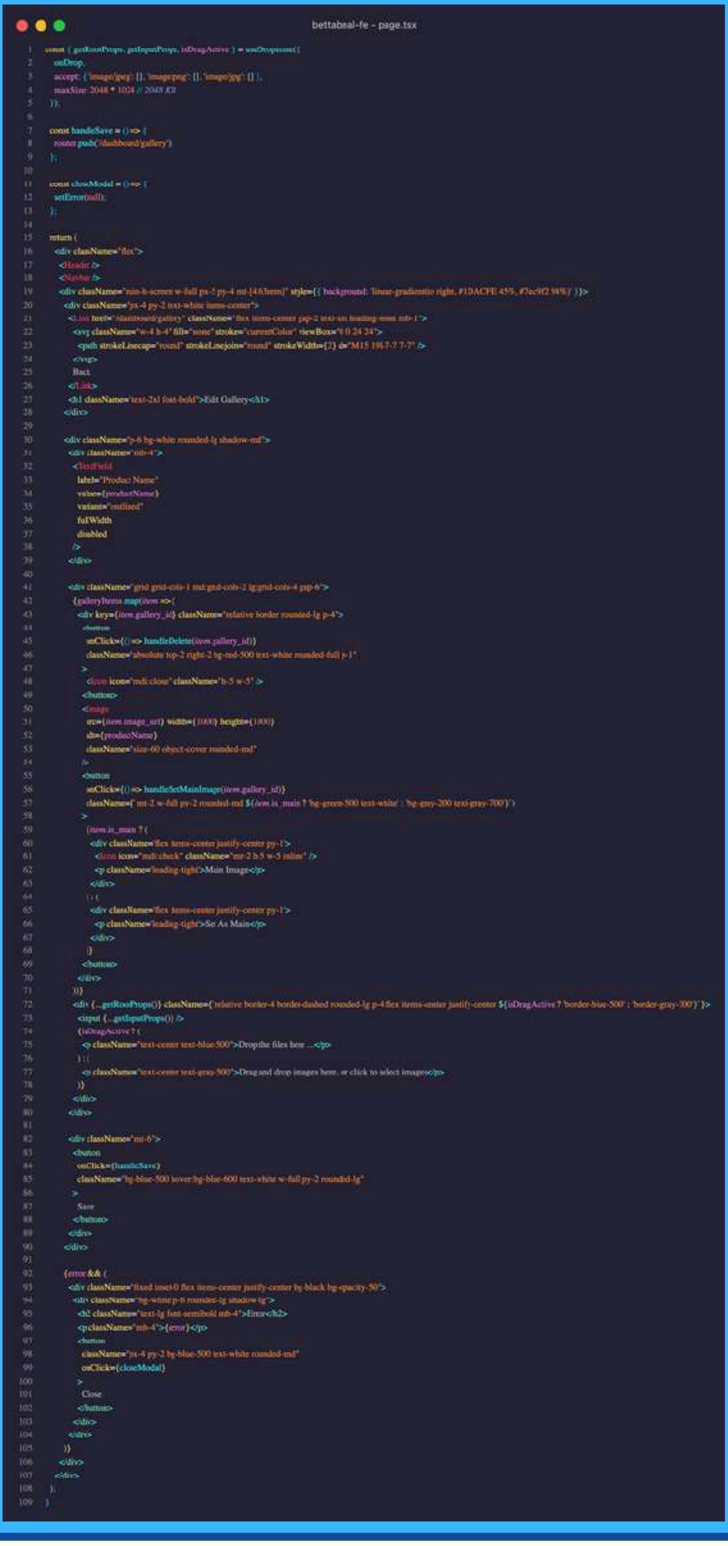
Edit Gallery product page



bettabeal-fe - page.tsx

```
export default function EditGalleryPage() {
  const [productName, setProductName] = useState('');
  const [galleryItems, setGalleryItems] = useState<GalleryItem[]>([]);
  const [error, setError] = useState<string | null>(null);
  const router = useRouter();
  const { productId } = useParams();

  useEffect(() => {
    const fetchGallery = async () => {
      const token = getCookie('USR');
      if (!token) {
        router.push('/login');
        return;
      }

      try {
        const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/products/${productId}/gallery`, {
          headers: {
            'Authorization': `Bearer ${token}`,
            'Content-Type': 'application/json',
          },
        });

        const data = await response.json();
        if (data.code === 200) {
          setProductName(data.data.product_name);
          setGalleryItems([data.data.main_image, ...[illegible] data.data.gallery]);
        } else {
          setError(`Error: ${data.message}`);
        }
      } catch (error) {
        console.error('Fetch error:', error);
        setError(`Fetch error: ${(error as Error).message}`);
      }
    };

    fetchGallery();
  }, [productId, router]);

  const handleDelete = async (galleryId: number) => {
    const token = getCookie('USR');
    if (!token) {
      router.push('/login');
      return;
    }

    const itemToDelete = galleryItems.find(item => item.gallery_id === galleryId);
    if (itemToDelete?.is_main) {
      setError('Cannot delete the main image. Please set another image as the main image first.');
      return;
    }

    try {
      const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/seller/gallery/${galleryId}`, {
        method: 'DELETE',
        headers: {
          'Authorization': `Bearer ${token}`,
          'Content-Type': 'application/json',
        },
      });

      if (!response.ok) {
        const text = await response.text();
        console.error('Error deleting image:', text);
        setError(`Error: ${response.status} - ${text}`);
        return;
      }

      setGalleryItems(galleryItems.filter(item => item.gallery_id !== galleryId));
    } catch (error) {
      console.error('Delete error:', error);
      setError(`Delete error: ${(error as Error).message}`);
    }
  };

  const handleSetMainImage = async (galleryId: number) => {
    const token = getCookie('USR');
    if (!token) {
      router.push('/login');
      return;
    }

    try {
      const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/seller/gallery/${galleryId}/main`, {
        method: 'PUT',
        headers: {
          'Authorization': `Bearer ${token}`,
          'Content-Type': 'application/json',
        },
      });

      if (!response.ok) {
        const text = await response.text();
        console.error('Error setting main image:', text);
        setError(`Error: ${response.status} - ${text}`);
        return;
      }

      setGalleryItems(galleryItems.map(item => ({
        ...item,
        is_main: item.gallery_id === galleryId,
      })));
    } catch (error) {
      console.error('Set main image error:', error);
      setError(`Set main image error: ${(error as Error).message}`);
    }
  };

  const onDrop = useCallback(async (acceptedFiles: File[], rejectedFiles: FileRejection[]) => {
    const token = getCookie('USR');
    if (!token) {
      router.push('/login');
      return;
    }

    if (rejectedFiles.length > 0) {
      setError('File size exceeds the 2 MB limit.');
      return;
    }

    const file = acceptedFiles[0];
    if (!file) return;

    const formData = new FormData();
    formData.append('image', file);

    try {
      const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/seller/gallery/upload/${productId}`, {
        method: 'POST',
        headers: {
          'Authorization': `Bearer ${token}`,
        },
        body: formData,
      });

      const data = await response.json();
      if (data.code === 200) {
        setGalleryItems([...galleryItems, data.data]);
      } else {
        setError(`Error: ${data.message}`);
      }
    } catch (error) {
      console.error('Upload error:', error);
      setError(`Upload error: ${(error as Error).message}`);
    }
  }, [galleryItems, productId, router]);
```

Source Code



bettabeal-fe - page.tsx

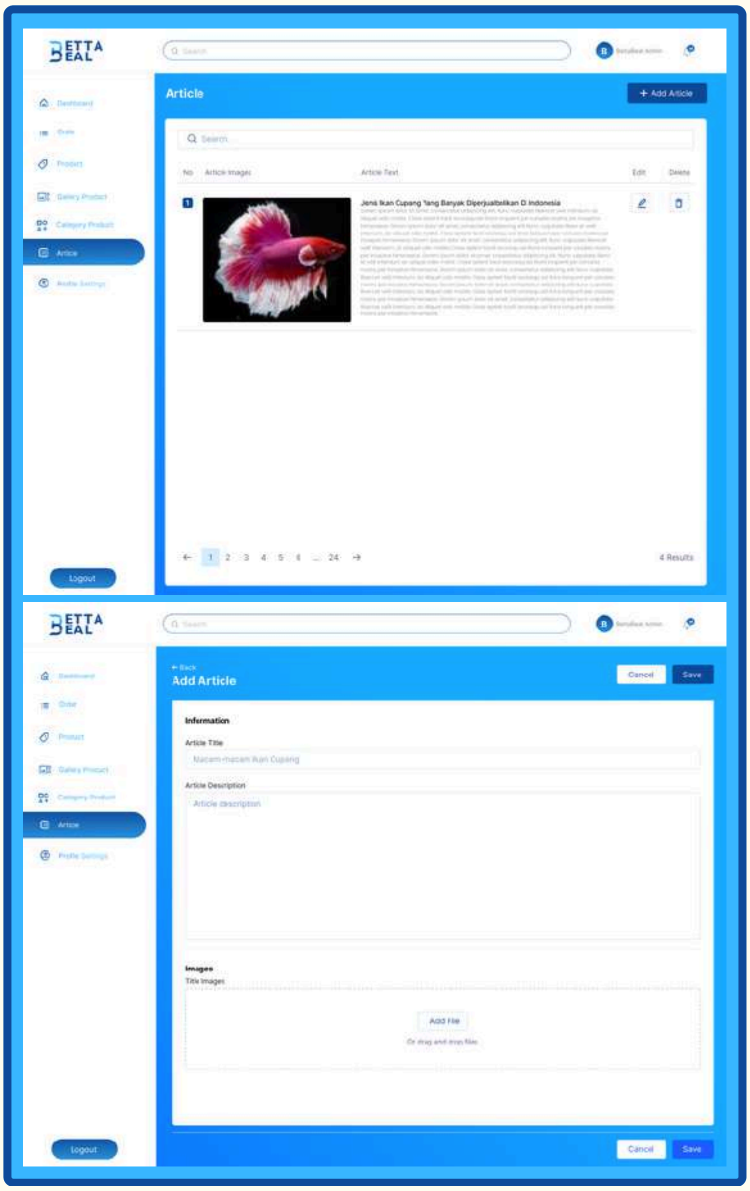
```
const { getRootProps, getInputProps, isDragActive } = useDropzone({
  onDrop,
  accept: { 'image/jpeg': [], 'image/png': [], 'image/jpg': [] },
  maxSize: 2048 * 1024 // 2048 KB
});

const handleSave = () => {
  router.push('/dashboard/gallery');
};

const closeModal = () => {
  setError(null);
};

return (
  <div className="flex">
    <Header />
    <Navbar />
    <div className="min-h-screen w-full px-5 py-4 mt-[4.63rem]" style={{ background: 'linear-gradient(to right, #13ACFE 45%, #7ec9f2 94%)' }}>
      <div className="px-4 py-2 text-white items-center">
        <Link href="/dashboard/gallery" className="flex items-center gap-2 text-sm leading-none mb-1">
          <svg className="w-4 h-4" fill="none" stroke="currentColor" viewBox="0 0 24 24">
            <path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M15 19l-7-7 7-7" />
          </svg>
          Back
        </Link>
        <h1 className="text-2xl font-bold">Edit Gallery</h1>
      </div>

      <div className="p-6 bg-white rounded-lg shadow-md">
        <div className="mb-4">
          <TextField
            label="Product Name"
            value={productName}
            variant="outlined"
            fullWidth
            disabled
          />
        </div>

        <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-4 gap-6">
          {galleryItems.map(item => (
            <div key={item.gallery_id} className="relative border rounded-lg p-4">
              <button
                onClick={() => handleDelete(item.gallery_id)}
                className="absolute top-2 right-2 bg-red-500 text-white rounded-full p-1"
              >
                <Icon icon="mdi:close" className="h-5 w-5" />
              </button>
              <Image
                src={item.image_url} width={1000} height={1000}
                alt={productName}
                className="size-60 object-cover rounded-md"
              />
              <button
                onClick={() => handleSetMainImage(item.gallery_id)}
                className={`mt-2 w-full py-2 rounded-md ${item.is_main ? 'bg-green-500 text-white' : 'bg-gray-200 text-gray-700'}`}
              >
                {item.is_main ? (
                  <div className="flex items-center justify-center py-1">
                    <Icon icon="mdi:check" className="mr-2 h-5 w-5 inline" />
                    <p className="leading-tight">Main Image</p>
                  </div>
                ) : (
                  <div className="flex items-center justify-center py-1">
                    <p className="leading-tight">Set As Main</p>
                  </div>
                )}
              </button>
            </div>
          ))}
          <div {...getRootProps()} className={`relative border-4 border-dashed rounded-lg p-4 flex items-center justify-center ${isDragActive ? 'border-blue-500' : 'border-gray-300'}`}>
            <input {...getInputProps()} />
            {isDragActive ? (
              <p className="text-center text-blue-500">Drop the files here ...</p>
            ) : (
              <p className="text-center text-gray-500">Drag and drop images here, or click to select images</p>
            )}
          </div>
        </div>

        <div className="mt-6">
          <button
            onClick={handleSave}
            className="bg-blue-500 hover:bg-blue-600 text-white w-full py-2 rounded-lg"
          >
            Save
          </button>
        </div>
      </div>

      {error && (
        <div className="fixed inset-0 flex items-center justify-center bg-black bg-opacity-50">
          <div className="bg-white p-6 rounded-lg shadow-lg">
            <h2 className="text-lg font-semibold mb-4">Error</h2>
            <p className="mb-4">{error}</p>
            <button
              className="px-4 py-2 bg-blue-500 text-white rounded-md"
              onClick={closeModal}
            >
              Close
            </button>
          </div>
        </div>
      )}
    </div>
  </div>
);
}
```

Website Admin

6. Article Page

Halaman artikel admin memungkinkan admin untuk menambahkan artikel baru yang akan ditampilkan di website pelanggan. Admin dapat memasukkan foto, judul, dan isi artikel yang berkaitan dengan ikan cupang, serta menyunting atau memperbarui artikel yang sudah ada. Fitur ini mempermudah admin dalam mengelola konten yang relevan dan informatif untuk pelanggan di website.

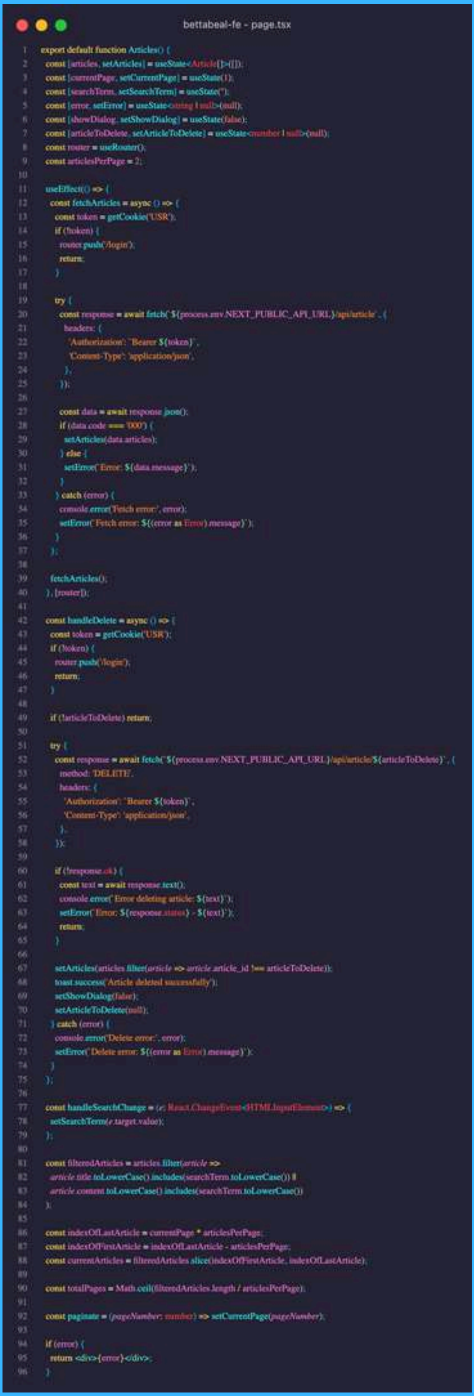
Pageview
Article page



Website Customer

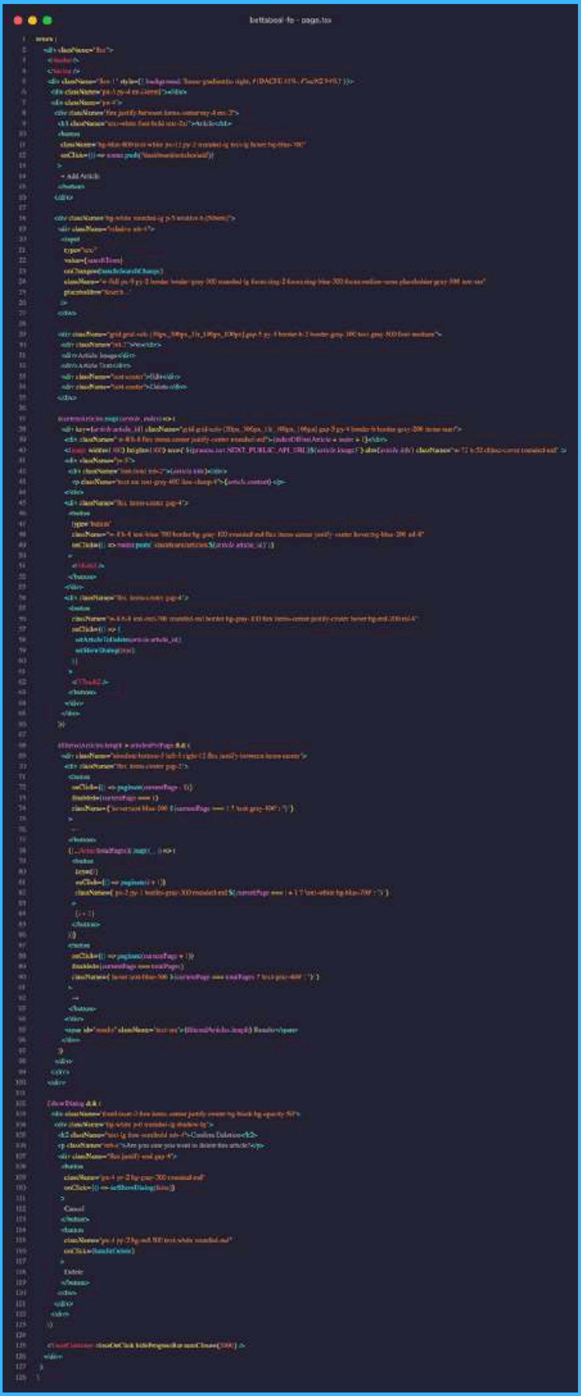
Source Code

Article page



bettabeal-fe - page.tsx

```tsx
export default function Articles() {
  const [articles, setArticles] = useState<Article[]>([]);
  const [currentPage, setCurrentPage] = useState(1);
  const [searchTerm, setSearchTerm] = useState('');
  const [error, setError] = useState<string | null>(null);
  const [showDialog, setShowDialog] = useState(false);
  const [articleToDelete, setArticleToDelete] = useState<number | null>(null);
  const router = useRouter();
  const articlesPerPage = 2;

  useEffect(() => {
    const fetchArticles = async () => {
      const token = getCookie('USR');
      if (!token) {
        router.push('/login');
        return;
      }

      try {
        const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/article`, {
          headers: {
            'Authorization': `Bearer ${token}`,
            'Content-Type': 'application/json',
          },
        });

        const data = await response.json();
        if (data.code === '000') {
          setArticles(data.articles);
        } else {
          setError(`Error: ${data.message}`);
        }
      } catch (error) {
        console.error('Fetch error:', error);
        setError(`Fetch error: ${(error as Error).message}`);
      }
    };

    fetchArticles();
  }, [router]);

  const handleDelete = async () => {
    const token = getCookie('USR');
    if (!token) {
      router.push('/login');
      return;
    }

    if (!articleToDelete) return;

    try {
      const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/article/${articleToDelete}`, {
        method: 'DELETE',
        headers: {
          'Authorization': `Bearer ${token}`,
          'Content-Type': 'application/json',
        },
      });

      if (!response.ok) {
        const text = await response.text();
        console.error(`Error deleting article: ${text}`);
        setError(`Error: ${response.status} - ${text}`);
        return;
      }

      setArticles(articles.filter(article => article.article_id !== articleToDelete));
      toast.success('Article deleted successfully');
      setShowDialog(false);
      setArticleToDelete(null);
    } catch (error) {
      console.error('Delete error:', error);
      setError(`Delete error: ${(error as Error).message}`);
    }
  };

  const handleSearchChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    setSearchTerm(e.target.value);
  };

  const filteredArticles = articles.filter(article =>
    article.title.toLowerCase().includes(searchTerm.toLowerCase()) ||
    article.content.toLowerCase().includes(searchTerm.toLowerCase())
  );

  const indexOfLastArticle = currentPage * articlesPerPage;
  const indexOfFirstArticle = indexOfLastArticle - articlesPerPage;
  const currentArticles = filteredArticles.slice(indexOfFirstArticle, indexOfLastArticle);

  const totalPages = Math.ceil(filteredArticles.length / articlesPerPage);

  const paginate = (pageNumber: number) => setCurrentPage(pageNumber);

  if (error) {
    return <div>{error}</div>;
  }
```



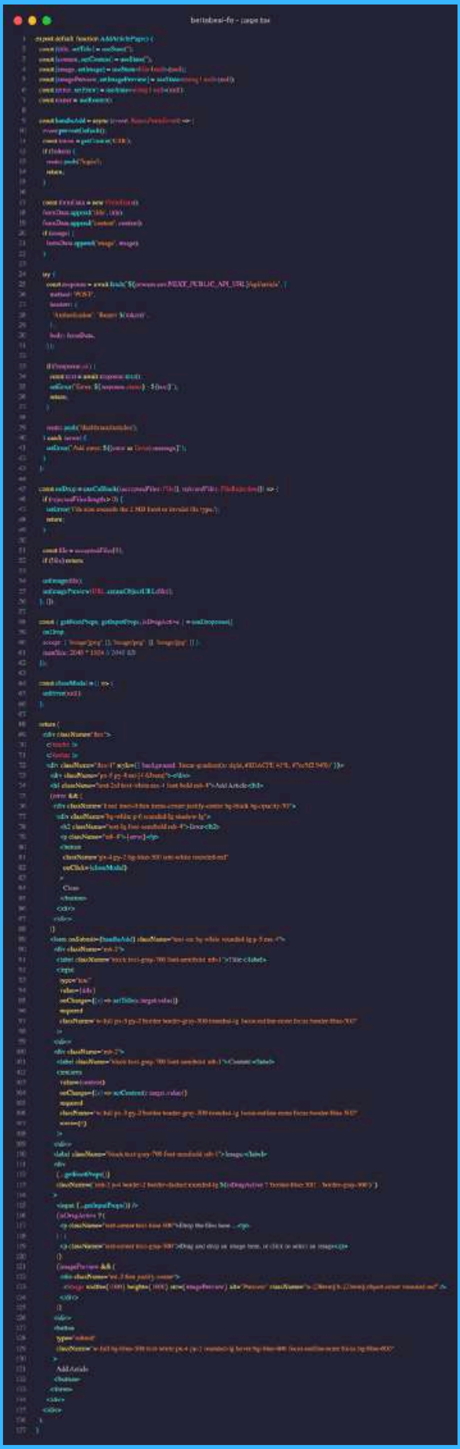
bettabeal-fe - page.tsx

```tsx
  return (
    <div className="flex">
      <[illegible] />
      <[illegible] />
      <div className="flex-1" style={{ background: 'linear-gradient(to right, #[illegible] 15%, #[illegible])' }}>
        <div className="[illegible]"></div>
        <div className="p-6">
          <div className="flex justify-between items-center my-4 mx-2">
            <h1 className="text-white font-bold text-2xl">Article</h1>
            <button
              className="bg-blue-800 text-white px-12 py-2 rounded-lg [illegible] hover:bg-blue-700"
              onClick={() => router.push('/dashboard/articles/add')}
            >
              + Add Article
            </button>
          </div>

          <div className="bg-white rounded-lg p-5 relative h-[50rem]">
            <div className="relative mb-4">
              <input
                type="text"
                value={searchTerm}
                onChange={handleSearchChange}
                className="w-full px-9 py-2 border border-gray-300 rounded-lg focus:ring-2 focus:ring-blue-300 focus:outline-none placeholder-gray-500 text-sm"
                placeholder="Search..."
              />
            </div>

            <div className="grid grid-cols-[50px_300px_1fr_100px_100px] gap-5 py-4 border-b-2 border-gray-100 text-gray-500 font-medium">
              <div className="[illegible]">No</div>
              <div>Article Image</div>
              <div>Article Text</div>
              <div className="text-center">Edit</div>
              <div className="text-center">Delete</div>
            </div>

            {currentArticles.map((article, index) => (
              <div key={article.article_id} className="grid grid-cols-[50px_300px_1fr_100px_100px] gap-5 py-4 border-b border-gray-200 items-start">
                <div className="[illegible]">{indexOfFirstArticle + index + 1}</div>
                <[illegible] width={[illegible]} height={[illegible]} src={`${process.env.NEXT_PUBLIC_API_URL}${article.image}`} alt={article.title} className="[illegible]" />
                <div className="[illegible]">
                  <div className="font-bold mb-2">{article.title}</div>
                  <p className="text-sm text-gray-400 line-clamp-4">{article.content}</p>
                </div>
                <div className="flex items-center gap-4">
                  <button
                    type="button"
                    className="[illegible]"
                    onClick={() => router.push(`/dashboard/articles/${article.article_id}`)}
                  >
                    <[illegible] />
                  </button>
                </div>
                <div className="flex items-center gap-4">
                  <button
                    className="[illegible]"
                    onClick={() => {
                      setArticleToDelete(article.article_id);
                      setShowDialog(true);
                    }}
                  >
                    <[illegible] />
                  </button>
                </div>
              </div>
            ))}

            {filteredArticles.length > articlesPerPage && (
              <div className="absolute bottom-5 left-5 right-12 flex justify-between items-center">
                <div className="flex items-center gap-2">
                  <button
                    onClick={() => paginate(currentPage - 1)}
                    disabled={currentPage === 1}
                    className={`[illegible] ${currentPage === 1 ? 'text-gray-400' : ''}`}
                  >
                    ←
                  </button>
                  {[...Array(totalPages)].map((_, i) => (
                    <button
                      key={i}
                      onClick={() => paginate(i + 1)}
                      className={`px-2 py-1 border-gray-300 rounded-md ${currentPage === i + 1 ? 'text-white bg-blue-700' : ''}`}
                    >
                      {i + 1}
                    </button>
                  ))}
                  <button
                    onClick={() => paginate(currentPage + 1)}
                    disabled={currentPage === totalPages}
                    className={`[illegible] ${currentPage === totalPages ? 'text-gray-400' : ''}`}
                  >
                    →
                  </button>
                </div>
                <span id="results" className="text-sm">{filteredArticles.length} Results</span>
              </div>
            )}
          </div>
        </div>
      </div>

      {showDialog && (
        <div className="fixed inset-0 flex items-center justify-center bg-black bg-opacity-50">
          <div className="bg-white p-6 rounded-lg shadow-lg">
            <h2 className="text-lg font-semibold mb-4">Confirm Deletion</h2>
            <p className="mb-4">Are you sure you want to delete this article?</p>
            <div className="flex justify-end gap-4">
              <button
                className="px-4 py-2 bg-gray-300 rounded-md"
                onClick={() => setShowDialog(false)}
              >
                Cancel
              </button>
              <button
                className="px-4 py-2 bg-red-500 text-white rounded-md"
                onClick={handleDelete}
              >
                Delete
              </button>
            </div>
          </div>
        </div>
      )}

      <ToastContainer closeOnClick hideProgressBar autoClose={3000} />
    </div>
  );
}
```

Website Customer

Add Article page

Source Code



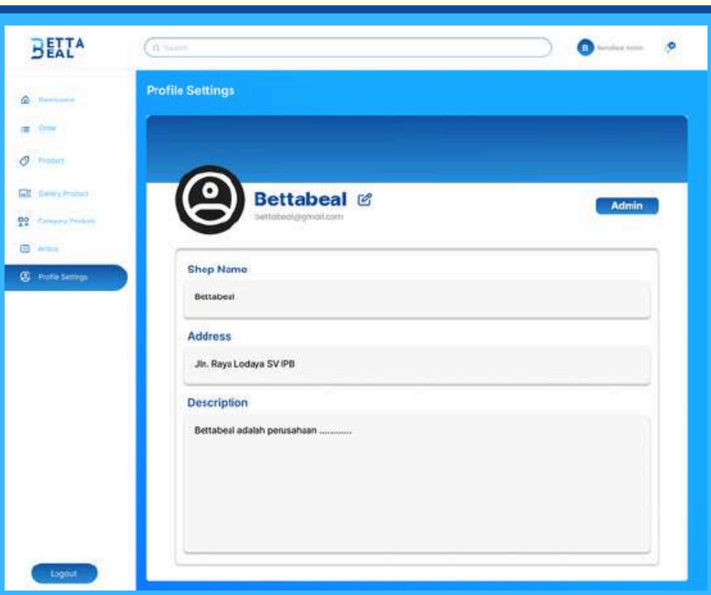
```
[illegible]
```

Website Admin

7. Profile Admin Page

Halaman pengaturan profil admin memungkinkan admin untuk melihat dan mengedit informasi profil, termasuk foto profil, nama toko, alamat toko, dan deskripsi toko. Fitur ini memberikan admin fleksibilitas untuk memperbarui informasi toko sesuai kebutuhan dan menjaga data profil tetap akurat dan terkini.

Pageview
Profile admin page



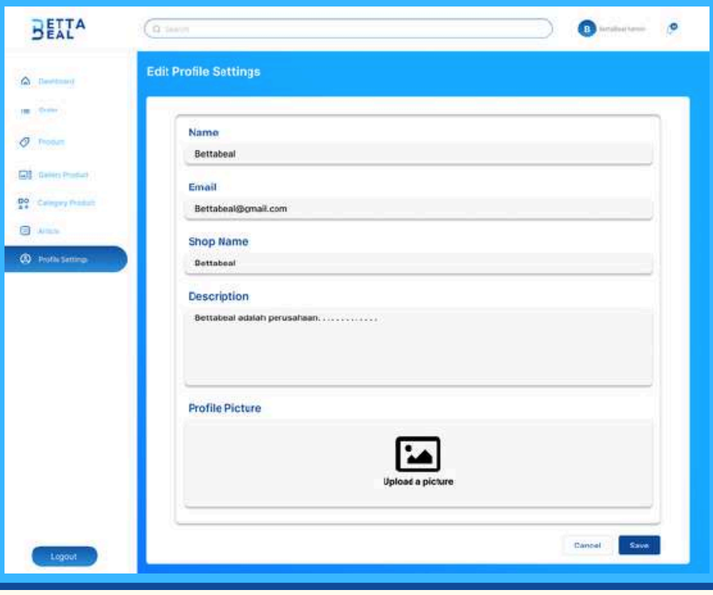
Profile Settings

Bettabeal
bettabeal@gmail.com

Shop Name: Bettabeal

Address: Jln. Raya Lodaya SV IPB

Description: Bettabeal adalah perusahaan ....



Edit Profile Settings

Name: Bettabeal

Email: Bettabeal@gmail.com

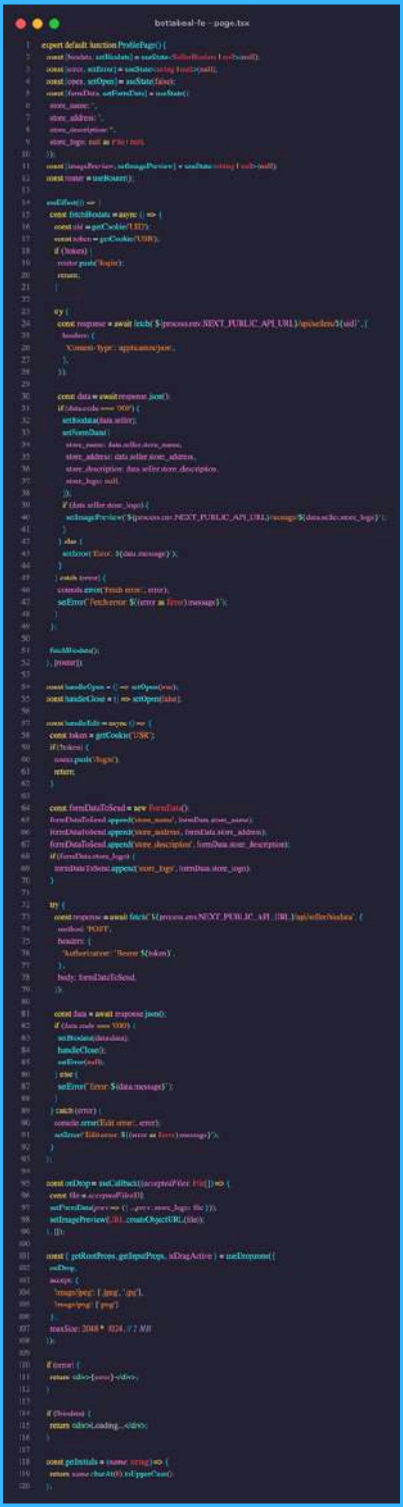
Shop Name: Bettabeal

Description: Bettabeal adalah perusahaan. . . . .

Profile Picture

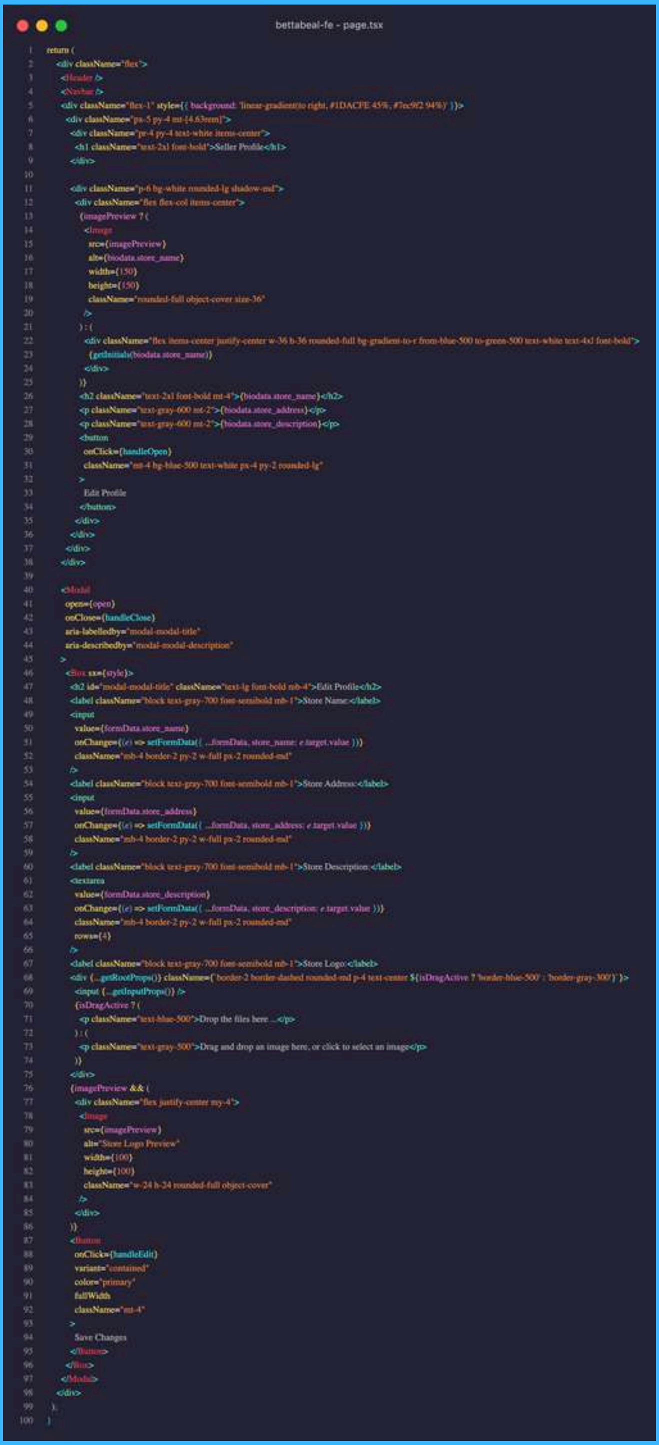
Website Customer

Source Code



bettabeal-fe - page.tsx

```tsx
export default function ProfilePage() {
  const [biodata, setBiodata] = useState<SellerBiodata | null>(null);
  const [error, setError] = useState<string | null>(null);
  const [open, setOpen] = useState(false);
  const [formData, setFormData] = useState({
    store_name: '',
    store_address: '',
    store_description: '',
    store_logo: null as File | null,
  });
  const [imagePreview, setImagePreview] = useState<string | null>(null);
  const router = useRouter();

  useEffect(() => {
    const fetchBiodata = async () => {
      const uid = getCookie('UID');
      const token = getCookie('USR');
      if (!token) {
        router.push('/login');
        return;
      }

      try {
        const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/sellers/${uid}`, {
          headers: {
            'Content-Type': 'application/json',
          },
        });

        const data = await response.json();
        if (data.code === '000') {
          setBiodata(data.seller);
          setFormData({
            store_name: data.seller.store_name,
            store_address: data.seller.store_address,
            store_description: data.seller.store_description,
            store_logo: null,
          });
          if (data.seller.store_logo) {
            setImagePreview(`${process.env.NEXT_PUBLIC_API_URL}/storage/${data.seller.store_logo}`);
          }
        } else {
          setError(`Error: ${data.message}`);
        }
      } catch (error) {
        console.error('Fetch error:', error);
        setError(`Fetch error: ${(error as Error).message}`);
      }
    };

    fetchBiodata();
  }, [router]);

  const handleOpen = () => setOpen(true);
  const handleClose = () => setOpen(false);

  const handleEdit = async () => {
    const token = getCookie('USR');
    if (!token) {
      router.push('/login');
      return;
    }

    const formDataToSend = new FormData();
    formDataToSend.append('store_name', formData.store_name);
    formDataToSend.append('store_address', formData.store_address);
    formDataToSend.append('store_description', formData.store_description);
    if (formData.store_logo) {
      formDataToSend.append('store_logo', formData.store_logo);
    }

    try {
      const response = await fetch(`${process.env.NEXT_PUBLIC_API_URL}/api/seller/biodata`, {
        method: 'POST',
        headers: {
          'Authorization': `Bearer ${token}`,
        },
        body: formDataToSend,
      });

      const data = await response.json();
      if (data.code === '000') {
        setBiodata(data.data);
        handleClose();
        setError(null);
      } else {
        setError(`Error: ${data.message}`);
      }
    } catch (error) {
      console.error('Edit error:', error);
      setError(`Edit error: ${(error as Error).message}`);
    }
  };

  const onDrop = useCallback((acceptedFiles: File[]) => {
    const file = acceptedFiles[0];
    setFormData(prev => ({ ...prev, store_logo: file }));
    setImagePreview(URL.createObjectURL(file));
  }, []);

  const { getRootProps, getInputProps, isDragActive } = useDropzone({
    onDrop,
    accept: {
      'image/jpeg': ['.jpeg', '.jpg'],
      'image/png': ['.png']
    },
    maxSize: 2048 * 1024, // 2 MB
  });

  if (error) {
    return <div>{error}</div>;
  }

  if (!biodata) {
    return <div>Loading...</div>;
  }

  const getInitials = (name: string) => {
    return name.charAt(0).toUpperCase();
  };
```



bettabeal-fe - page.tsx

```tsx
  return (
    <div className="flex">
      <Header />
      <Navbar />
      <div className="flex-1" style={{ background: 'linear-gradient(to right, #1DACFE 45%, #7ec9f2 94%)' }}>
        <div className="px-5 py-4 mt-[4.63rem]">
          <div className="pr-4 py-4 text-white items-center">
            <h1 className="text-2xl font-bold">Seller Profile</h1>
          </div>

          <div className="p-6 bg-white rounded-lg shadow-md">
            <div className="flex flex-col items-center">
              {imagePreview ? (
                <Image
                  src={imagePreview}
                  alt={biodata.store_name}
                  width={150}
                  height={150}
                  className="rounded-full object-cover size-36"
                />
              ) : (
                <div className="flex items-center justify-center w-36 h-36 rounded-full bg-gradient-to-r from-blue-500 to-green-500 text-white text-4xl font-bold">
                  {getInitials(biodata.store_name)}
                </div>
              )}
              <h2 className="text-2xl font-bold mt-4">{biodata.store_name}</h2>
              <p className="text-gray-600 mt-2">{biodata.store_address}</p>
              <p className="text-gray-600 mt-2">{biodata.store_description}</p>
              <button
                onClick={handleOpen}
                className="mt-4 bg-blue-500 text-white px-4 py-2 rounded-lg"
              >
                Edit Profile
              </button>
            </div>
          </div>
        </div>
      </div>

      <Modal
        open={open}
        onClose={handleClose}
        aria-labelledby="modal-modal-title"
        aria-describedby="modal-modal-description"
      >
        <Box sx={style}>
          <h2 id="modal-modal-title" className="text-lg font-bold mb-4">Edit Profile</h2>
          <label className="block text-gray-700 font-semibold mb-1">Store Name:</label>
          <input
            value={formData.store_name}
            onChange={(e) => setFormData({ ...formData, store_name: e.target.value })}
            className="mb-4 border-2 py-2 w-full px-2 rounded-md"
          />
          <label className="block text-gray-700 font-semibold mb-1">Store Address:</label>
          <input
            value={formData.store_address}
            onChange={(e) => setFormData({ ...formData, store_address: e.target.value })}
            className="mb-4 border-2 py-2 w-full px-2 rounded-md"
          />
          <label className="block text-gray-700 font-semibold mb-1">Store Description:</label>
          <textarea
            value={formData.store_description}
            onChange={(e) => setFormData({ ...formData, store_description: e.target.value })}
            className="mb-4 border-2 py-2 w-full px-2 rounded-md"
            rows={4}
          />
          <label className="block text-gray-700 font-semibold mb-1">Store Logo:</label>
          <div {...getRootProps()} className={`border-2 border-dashed rounded-md p-4 text-center ${isDragActive ? 'border-blue-500' : 'border-gray-300'}`}>
            <input {...getInputProps()} />
            {isDragActive ? (
              <p className="text-blue-500">Drop the files here ...</p>
            ) : (
              <p className="text-gray-500">Drag and drop an image here, or click to select an image</p>
            )}
          </div>
          {imagePreview && (
            <div className="flex justify-center my-4">
              <Image
                src={imagePreview}
                alt="Store Logo Preview"
                width={100}
                height={100}
                className="w-24 h-24 rounded-full object-cover"
              />
            </div>
          )}
          <Button
            onClick={handleEdit}
            variant="contained"
            color="primary"
            fullWidth
            className="mt-4"
          >
            Save Changes
          </Button>
        </Box>
      </Modal>
    </div>
  );
}
```

API *Endpoint*

Bettabeal menggunakan API sebagai penghubung antara aplikasi mobile, website customer, dan website dashboard admin dengan backend sistem. API ini dirancang untuk memastikan komunikasi data yang cepat, aman, dan efisien. Fitur API mencakup pengelolaan produk, artikel, dan pesanan, seperti menampilkan daftar produk, menambah atau mengedit artikel, serta memproses data order dari customer. Selain itu, API juga mendukung integrasi pembayaran di aplikasi mobile, memungkinkan transaksi dilakukan secara langsung. API ini dibangun menggunakan standar RESTful dengan autentikasi yang aman, sehingga memastikan data antara server dan klien terlindungi dengan baik. Dengan API ini, semua komponen Bettabeal terhubung secara seamless, memberikan pengalaman pengguna yang optimal.

Daftar API

Orders	METHOD
Create Order	POST
Get Order	GET
Edit Status	PUT
Edit Shipping Status	PUT
Seller order mark as comp	POST

Wishlist	METHOD
Menampilkan wishlist	GET
Menampilkan wishlist diuser	GET
Menambahkan wishlist	POST
Menambahkan wishlist	DEL

Ulasan	METHOD
Get review product	GET
Get setelah direview	GET
Get sebelum direview	GET
Ulasan post	POST

Seller	METHOD
Seller list	GET
Seller detail	GET
Seller / biodata	POST

Daftar API

Categories	METHOD
Categories list	GET
Categories	POST
Categories detail	GET
Categories update	PUT
Non aktif Categories	DEL
Restore Categories	PUT

Article	METHOD
Article/list	GET
Article	POST
Article update	PUT
Article delete	DEL

Komentar	METHOD
Komentar list	GET
Komentar list copy	GET
Komentar	POST
Komentar update	PUT
Komentar delete	DEL

Daftar API

Address	METHOD
Addresses	POST
Get Addresses	GET
Get Addresses ID	GET
Get Districk	GET
Get Poscode by districk	GET
Edit address	PUT
Edit main address	PUT
Delete address	DEL

Cart	METHOD
Menampilkan cart	GET
Menambahkan cart	POST
Kosongkan cart	POST
Pindah ke cart	POST
Pindah ke cart multiple	POST
Mengurangi item	PUT
Menambahkan item	PUT
Hapus item dan cart	DEL

Daftar API

Customer	METHOD
Customer Login	POST
Customer Authentication	GET
Register Customer	POST
Register Admin	POST
Customer Biodata	POST
Customer Biodata	PUT
Customer List	GET
Customer Details	GET

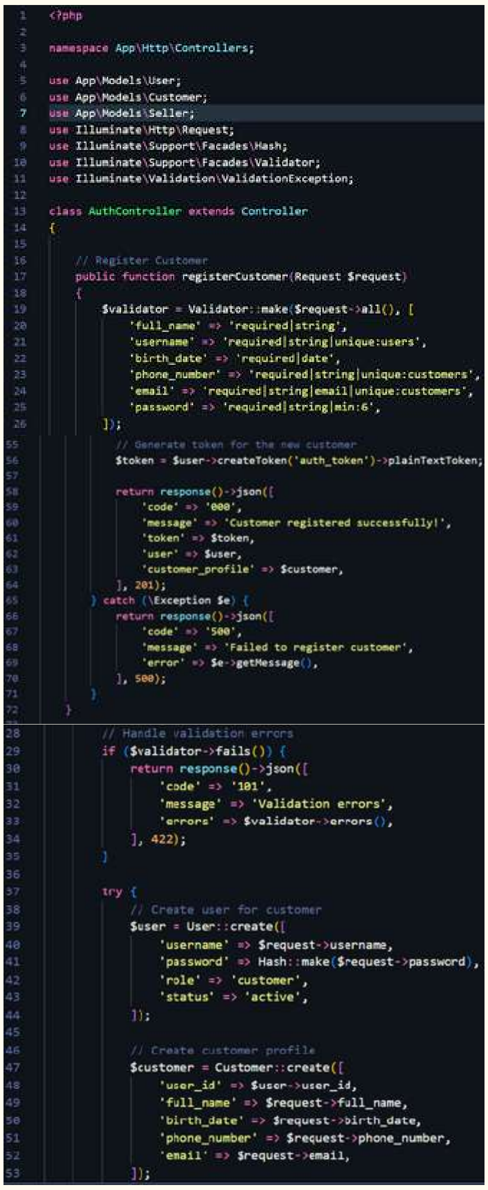
Product	METHOD
Product store	POST
Product list	GET
Product list copy	GET
Product seller	GET
Update Product	PUT
Restore Product	PUT
Update Product	DEL

Daftar API

Gallery Product	METHOD
Gallery product	POST
Gallery product list	GET
Update Gallery product	PUT
Delete Gallery product	DEL

API ENDPOINT BETTABEAL

1. Auth Login dan Register



```php
<?php

namespace App\Http\Controllers;

use App\Models\User;
use App\Models\Customer;
use App\Models\Seller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use Illuminate\Validation\ValidationException;

class AuthController extends Controller
{

    // Register Customer
    public function registerCustomer(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'full_name' => 'required|string',
            'username' => 'required|string|unique:users',
            'birth_date' => 'required|date',
            'phone_number' => 'required|string|unique:customers',
            'email' => 'required|string|email|unique:customers',
            'password' => 'required|string|min:6',
        ]);
```

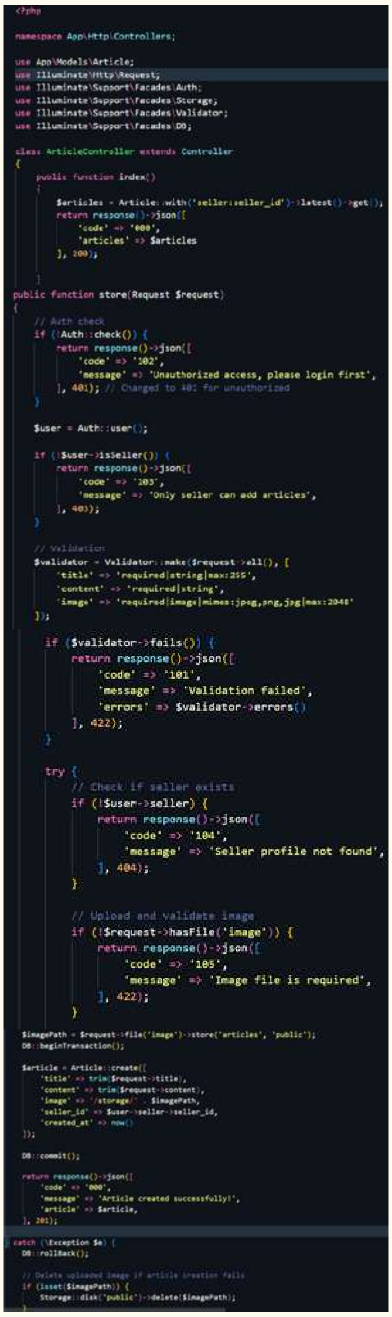
```php
            // Generate token for the new customer
            $token = $user->createToken('auth_token')->plainTextToken;

            return response()->json([
                'code' => '000',
                'message' => 'Customer registered successfully!',
                'token' => $token,
                'user' => $user,
                'customer_profile' => $customer,
            ], 201);
        } catch (\Exception $e) {
            return response()->json([
                'code' => '500',
                'message' => 'Failed to register customer',
                'error' => $e->getMessage(),
            ], 500);
        }
    }
```

```php
        // Handle validation errors
        if ($validator->fails()) {
            return response()->json([
                'code' => '101',
                'message' => 'Validation errors',
                'errors' => $validator->errors(),
            ], 422);
        }


        try {
            // Create user for customer
            $user = User::create([
                'username' => $request->username,
                'password' => Hash::make($request->password),
                'role' => 'customer',
                'status' => 'active',
            ]);

            // Create customer profile
            $customer = Customer::create([
                'user_id' => $user->user_id,
                'full_name' => $request->full_name,
                'birth_date' => $request->birth_date,
                'phone_number' => $request->phone_number,
                'email' => $request->email,
            ]);
```

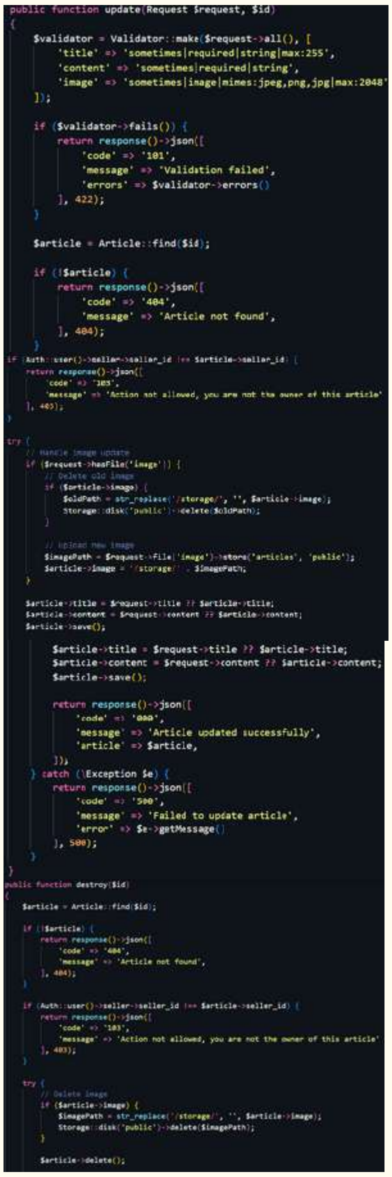
API ini digunakan untuk mengelola aktivitas login, registrasi, dan manajemen token untuk pengguna, baik melalui web maupun aplikasi.

API ENDPOINT BETTABEAL

ARTICLE CONTROLLER



```php
<?php

namespace App\Http\Controllers;

use App\Models\Article;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\DB;

class ArticleController extends Controller
{
    public function index()
    {
        $articles = Article::with('seller:seller_id')->latest()->get();
        return response()->json([
            'code' => '000',
            'articles' => $articles
        ], 200);

    }
public function store(Request $request)
{
    // Auth check
    if (!Auth::check()) {
        return response()->json([
            'code' => '102',
            'message' => 'Unauthorized access, please login first',
        ], 401); // Changed to 401 for unauthorized
    }

    $user = Auth::user();

    if (!$user->isSeller()) {
        return response()->json([
            'code' => '103',
            'message' => 'Only seller can add articles',
        ], 403);
    }

    // Validation
    $validator = Validator::make($request->all(), [
        'title' => 'required|string|max:255',
        'content' => 'required|string',
        'image' => 'required|image|mimes:jpeg,png,jpg|max:2048'
    ]);
      if ($validator->fails()) {
          return response()->json([
              'code' => '101',
              'message' => 'Validation failed',
              'errors' => $validator->errors()
          ], 422);
      }

      try {
          // Check if seller exists
          if (!$user->seller) {
              return response()->json([
                  'code' => '104',
                  'message' => 'Seller profile not found',
              ], 404);
          }

          // Upload and validate image
          if (!$request->hasFile('image')) {
              return response()->json([
                  'code' => '105',
                  'message' => 'Image file is required',
              ], 422);
          }
  $imagePath = $request->file('image')->store('articles', 'public');
  DB::beginTransaction();

  $article = Article::create([
      'title' => trim($request->title),
      'content' => trim($request->content),
      'image' => '/storage/' . $imagePath,
      'seller_id' => $user->seller->seller_id,
      'created_at' => now()
  ]);

  DB::commit();

  return response()->json([
      'code' => '000',
      'message' => 'Article created successfully!',
      'article' => $article,
  ], 201);

} catch (\Exception $e) {
  DB::rollBack();

  // Delete uploaded image if article creation fails
  if (isset($imagePath)) {
      Storage::disk('public')->delete($imagePath);
  }
```

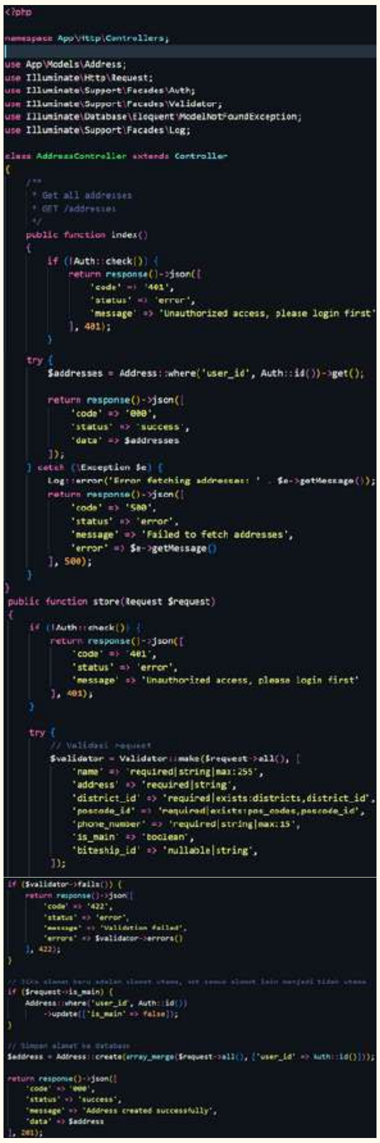


```php
public function update(Request $request, $id)
{
    $validator = Validator::make($request->all(), [
        'title' => 'sometimes|required|string|max:255',
        'content' => 'sometimes|required|string',
        'image' => 'sometimes|image|mimes:jpeg,png,jpg|max:2048'
    ]);

    if ($validator->fails()) {
        return response()->json([
            'code' => '101',
            'message' => 'Validation failed',
            'errors' => $validator->errors()
        ], 422);
    }

    $article = Article::find($id);

    if (!$article) {
        return response()->json([
            'code' => '404',
            'message' => 'Article not found',
        ], 404);
    }
if (Auth::user()->seller->seller_id !== $article->seller_id) {
    return response()->json([
        'code' => '103',
        'message' => 'Action not allowed, you are not the owner of this article'
    ], 403);
}

try {
    // Handle image update
    if ($request->hasFile('image')) {
        // Delete old image
        if ($article->image) {
            $oldPath = str_replace('/storage/', '', $article->image);
            Storage::disk('public')->delete($oldPath);
        }

        // Upload new image
        $imagePath = $request->file('image')->store('articles', 'public');
        $article->image = '/storage/' . $imagePath;
    }

    $article->title = $request->title ?? $article->title;
    $article->content = $request->content ?? $article->content;
    $article->save();
        $article->title = $request->title ?? $article->title;
        $article->content = $request->content ?? $article->content;
        $article->save();

        return response()->json([
            'code' => '000',
            'message' => 'Article updated successfully',
            'article' => $article,
        ]);
    } catch (\Exception $e) {
        return response()->json([
            'code' => '500',
            'message' => 'Failed to update article',
            'error' => $e->getMessage()
        ], 500);
    }
}
public function destroy($id)
{
    $article = Article::find($id);

    if (!$article) {
        return response()->json([
            'code' => '404',
            'message' => 'Article not found',
        ], 404);
    }

    if (Auth::user()->seller->seller_id !== $article->seller_id) {
        return response()->json([
            'code' => '103',
            'message' => 'Action not allowed, you are not the owner of this article'
        ], 403);
    }

    try {
        // Delete image
        if ($article->image) {
            $imagePath = str_replace('/storage/', '', $article->image);
            Storage::disk('public')->delete($imagePath);
        }

        $article->delete();
```

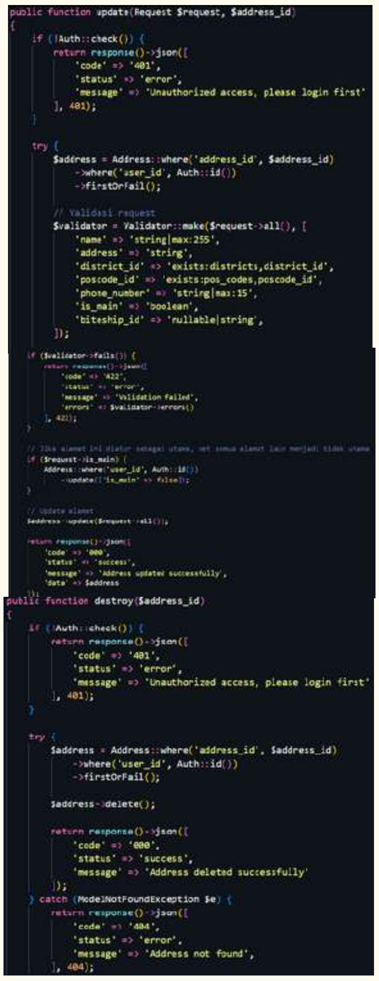
API ini digunakan untuk mengelola artikel, termasuk operasi untuk menampilkan, menambah, memperbarui, dan menghapus artikel oleh penjual.

API ENDPOINT BETTABEAL

ADDRESS CONTROLLER



```php
<?php

namespace App\Http\Controllers;

use App\Models\Address;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Validator;
use Illuminate\Database\Eloquent\ModelNotFoundException;
use Illuminate\Support\Facades\Log;

class AddressController extends Controller
{
    /**
     * Get all addresses
     * GET /addresses
     */
    public function index()
    {
        if (!Auth::check()) {
            return response()->json([
                'code' => '401',
                'status' => 'error',
                'message' => 'Unauthorized access, please login first'
            ], 401);
        }

    try {
        $addresses = Address::where('user_id', Auth::id())->get();

        return response()->json([
            'code' => '000',
            'status' => 'success',
            'data' => $addresses
        ]);
    } catch (\Exception $e) {
        Log::error('Error fetching addresses: ' . $e->getMessage());
        return response()->json([
            'code' => '500',
            'status' => 'error',
            'message' => 'Failed to fetch addresses',
            'error' => $e->getMessage()
        ], 500);
    }
}
public function store(Request $request)
{
    if (!Auth::check()) {
        return response()->json([
            'code' => '401',
            'status' => 'error',
            'message' => 'Unauthorized access, please login first'
        ], 401);
    }

    try {
        // Validasi request
        $validator = Validator::make($request->all(), [
            'name' => 'required|string|max:255',
            'address' => 'required|string',
            'district_id' => 'required|exists:districts,district_id',
            'poscode_id' => 'required|exists:pos_codes,poscode_id',
            'phone_number' => 'required|string|max:15',
            'is_main' => 'boolean',
            'biteship_id' => 'nullable|string',
        ]);
if ($validator->fails()) {
    return response()->json([
        'code' => '422',
        'status' => 'error',
        'message' => 'Validation failed',
        'errors' => $validator->errors()
    ], 422);
}

// Jika alamat baru adalah alamat utama, set semua alamat lain menjadi tidak utama
if ($request->is_main) {
    Address::where('user_id', Auth::id())
        ->update(['is_main' => false]);
}

// Simpan alamat ke database
$address = Address::create(array_merge($request->all(), ['user_id' => Auth::id()]));

return response()->json([
    'code' => '000',
    'status' => 'success',
    'message' => 'Address created successfully',
    'data' => $address
], 201);
```

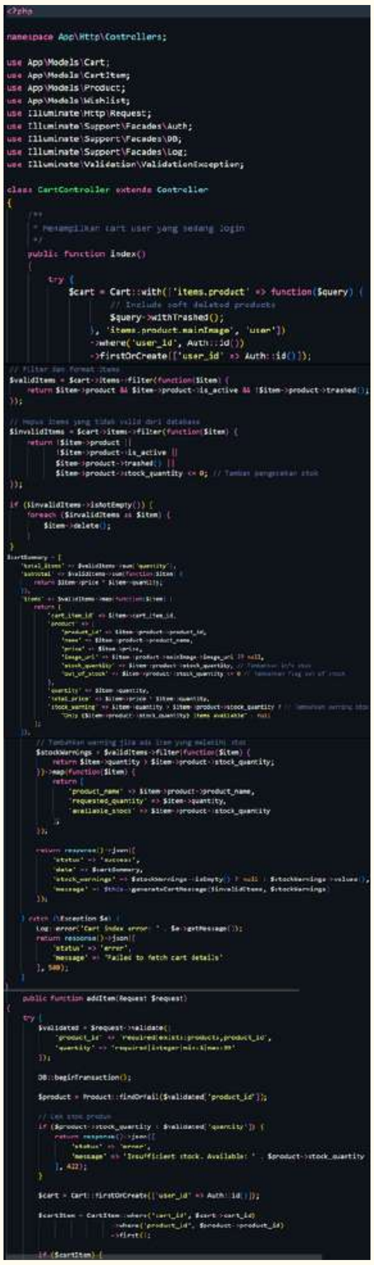


```php
public function update(Request $request, $address_id)
{
    if (!Auth::check()) {
        return response()->json([
            'code' => '401',
            'status' => 'error',
            'message' => 'Unauthorized access, please login first'
        ], 401);
    }

    try {
        $address = Address::where('address_id', $address_id)
            ->where('user_id', Auth::id())
            ->firstOrFail();

        // Validasi request
        $validator = Validator::make($request->all(), [
            'name' => 'string|max:255',
            'address' => 'string',
            'district_id' => 'exists:districts,district_id',
            'poscode_id' => 'exists:pos_codes,poscode_id',
            'phone_number' => 'string|max:15',
            'is_main' => 'boolean',
            'biteship_id' => 'nullable|string',
        ]);
    if ($validator->fails()) {
        return response()->json([
            'code' => '422',
            'status' => 'error',
            'message' => 'Validation failed',
            'errors' => $validator->errors()
        ], 422);
    }

    // Jika alamat ini diatur sebagai utama, set semua alamat lain menjadi tidak utama
    if ($request->is_main) {
        Address::where('user_id', Auth::id())
            ->update(['is_main' => false]);
    }

    // Update alamat
    $address->update($request->all());

    return response()->json([
        'code' => '000',
        'status' => 'success',
        'message' => 'Address updated successfully',
        'data' => $address
    ]);
public function destroy($address_id)
{
    if (!Auth::check()) {
        return response()->json([
            'code' => '401',
            'status' => 'error',
            'message' => 'Unauthorized access, please login first'
        ], 401);
    }

    try {
        $address = Address::where('address_id', $address_id)
            ->where('user_id', Auth::id())
            ->firstOrFail();

        $address->delete();

        return response()->json([
            'code' => '000',
            'status' => 'success',
            'message' => 'Address deleted successfully'
        ]);
    } catch (ModelNotFoundException $e) {
        return response()->json([
            'code' => '404',
            'status' => 'error',
            'message' => 'Address not found',
        ], 404);
```

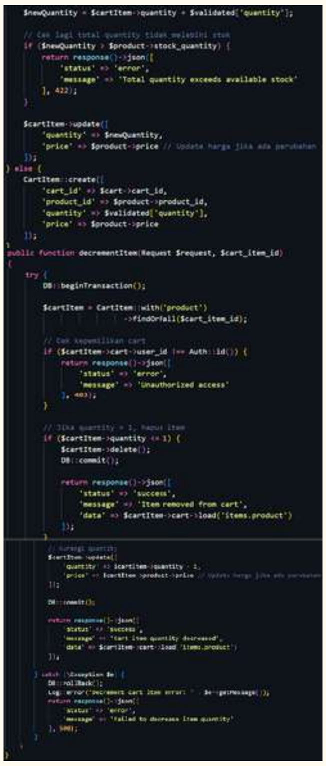
API ini digunakan untuk mengelola alamat pengguna, termasuk operasi untuk menampilkan, menambah, memperbarui, dan menghapus alamat

API ENDPOINT BETTABEAL

CART CONTROLLER



```php
<?php

namespace App\Http\Controllers;

use App\Models\Cart;
use App\Models\CartItem;
use App\Models\Product;
use App\Models\Wishlist;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Log;
use Illuminate\Validation\ValidationException;

class CartController extends Controller
{
    /**
     * Menampilkan cart user yang sedang login
     */
    public function index()
    {
        try {
            $cart = Cart::with(['items.product' => function($query) {
                    // Include soft deleted products
                    $query->withTrashed();
                }, 'items.product.mainImage', 'user'])
                ->where('user_id', Auth::id())
                ->firstOrCreate(['user_id' => Auth::id()]);
// Filter dan format items
$validItems = $cart->items->filter(function($item) {
    return $item->product && $item->product->is_active && !$item->product->trashed();
});

// Hapus items yang tidak valid dari database
$invalidItems = $cart->items->filter(function($item) {
    return !$item->product ||
           !$item->product->is_active ||
           $item->product->trashed() ||
           $item->product->stock_quantity <= 0; // Tambah pengecekan stok
});

if ($invalidItems->isNotEmpty()) {
    foreach ($invalidItems as $item) {
        $item->delete();
    }
}
$cartSummary = [
    'total_items' => $validItems->sum('quantity'),
    'subtotal' => $validItems->sum(function($item) {
        return $item->price * $item->quantity;
    }),
    'items' => $validItems->map(function($item) {
        return [
            'cart_item_id' => $item->cart_item_id,
            'product' => [
                'product_id' => $item->product->product_id,
                'name' => $item->product->product_name,
                'price' => $item->price,
                'image_url' => $item->product->mainImage->image_url ?? null,
                'stock_quantity' => $item->product->stock_quantity, // Tambahkan info stok
                'out_of_stock' => $item->product->stock_quantity <= 0 // Tambahkan flag out of stock
            ],
            'quantity' => $item->quantity,
            'total_price' => $item->price * $item->quantity,
            'stock_warning' => $item->quantity > $item->product->stock_quantity ? // Tambahkan warning stok
                "Only {$item->product->stock_quantity} items available" : null
        ];
    }),
    // Tambahkan warning jika ada item yang melebihi stok
    $stockWarnings = $validItems->filter(function($item) {
        return $item->quantity > $item->product->stock_quantity;
    })->map(function($item) {
        return [
            'product_name' => $item->product->product_name,
            'requested_quantity' => $item->quantity,
            'available_stock' => $item->product->stock_quantity
        ];
    });

    return response()->json([
        'status' => 'success',
        'data' => $cartSummary,
        'stock_warnings' => $stockWarnings->isEmpty() ? null : $stockWarnings->values(),
        'message' => $this->generateCartMessage($invalidItems, $stockWarnings)
    ]);

  } catch (\Exception $e) {
    Log::error('Cart index error: ' . $e->getMessage());
    return response()->json([
        'status' => 'error',
        'message' => 'Failed to fetch cart details'
    ], 500);
  }
}
  public function addItem(Request $request)
{
  try {
    $validated = $request->validate([
        'product_id' => 'required|exists:products,product_id',
        'quantity' => 'required|integer|min:1|max:99'
    ]);

    DB::beginTransaction();

    $product = Product::findOrFail($validated['product_id']);

    // Cek stok produk
    if ($product->stock_quantity < $validated['quantity']) {
        return response()->json([
            'status' => 'error',
            'message' => 'Insufficient stock. Available: ' . $product->stock_quantity
        ], 422);
    }

    $cart = Cart::firstOrCreate(['user_id' => Auth::id()]);

    $cartItem = CartItem::where('cart_id', $cart->cart_id)
                        ->where('product_id', $product->product_id)
                        ->first();

    if ($cartItem) {
```

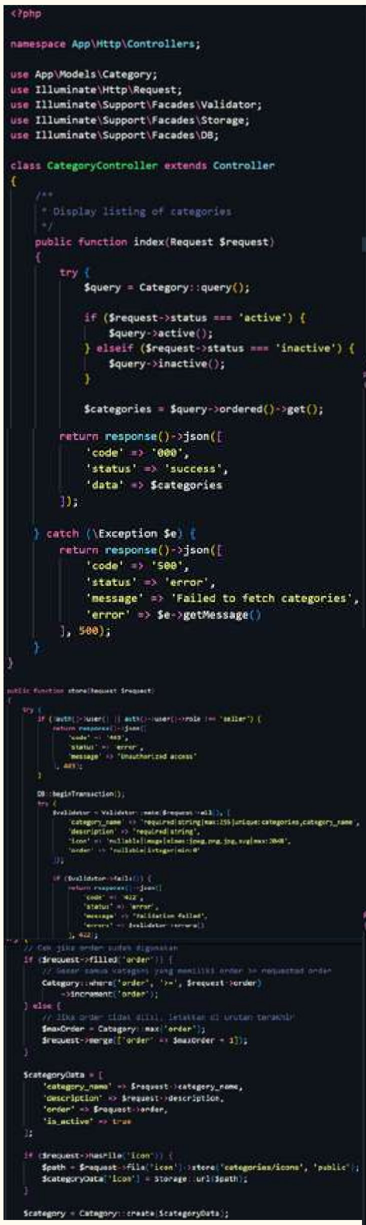


```php
    $newQuantity = $cartItem->quantity + $validated['quantity'];

    // Cek lagi total quantity tidak melebihi stok
    if ($newQuantity > $product->stock_quantity) {
        return response()->json([
            'status' => 'error',
            'message' => 'Total quantity exceeds available stock'
        ], 422);
    }

    $cartItem->update([
        'quantity' => $newQuantity,
        'price' => $product->price // Update harga jika ada perubahan
    ]);
} else {
    CartItem::create([
        'cart_id' => $cart->cart_id,
        'product_id' => $product->product_id,
        'quantity' => $validated['quantity'],
        'price' => $product->price
    ]);
}
public function decrementItem(Request $request, $cart_item_id)
{
    try {
        DB::beginTransaction();

        $cartItem = CartItem::with('product')
                            ->findOrFail($cart_item_id);

        // Cek kepemilikan cart
        if ($cartItem->cart->user_id !== Auth::id()) {
            return response()->json([
                'status' => 'error',
                'message' => 'Unauthorized access'
            ], 403);
        }

        // Jika quantity = 1, hapus item
        if ($cartItem->quantity <= 1) {
            $cartItem->delete();
            DB::commit();

            return response()->json([
                'status' => 'success',
                'message' => 'Item removed from cart',
                'data' => $cartItem->cart->load('items.product')
            ]);
        }
        // Kurangi quantity
        $cartItem->update([
            'quantity' => $cartItem->quantity - 1,
            'price' => $cartItem->product->price // Update harga jika ada perubahan
        ]);

        DB::commit();

        return response()->json([
            'status' => 'success',
            'message' => 'Cart item quantity decreased',
            'data' => $cartItem->cart->load('items.product')
        ]);

    } catch (\Exception $e) {
        DB::rollBack();
        Log::error('Decrement cart item error: ' . $e->getMessage());
        return response()->json([
            'status' => 'error',
            'message' => 'Failed to decrease item quantity'
        ], 500);
    }
}
}
```

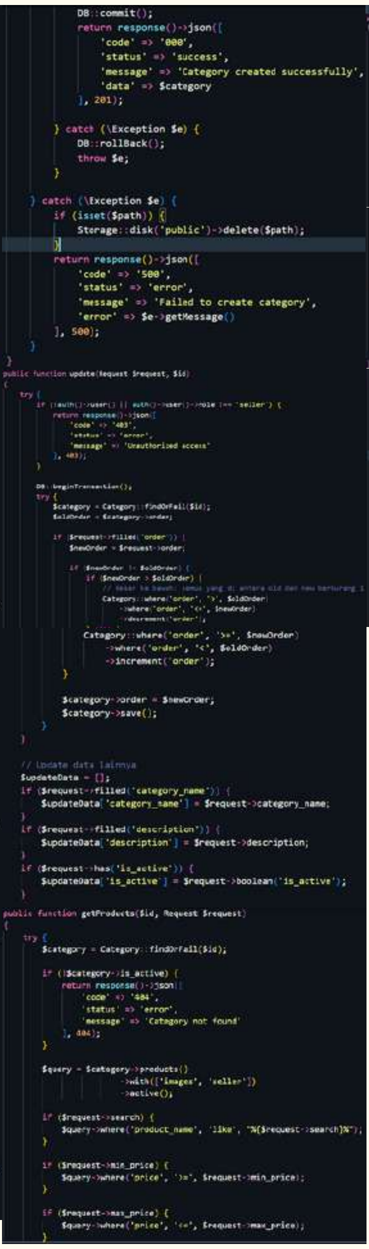
API ini digunakan untuk mengelola keranjang belanja pengguna, termasuk operasi untuk menampilkan, menambahkan, memperbarui, menghapus item

API ENDPOINT BETTABEAL

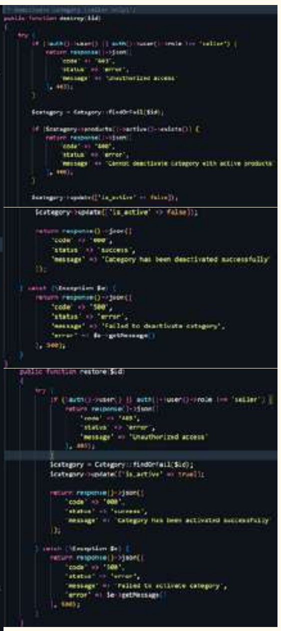
CATEGORY CONTROLLER



```php
<?php

namespace App\Http\Controllers;

use App\Models\Category;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\DB;

class CategoryController extends Controller
{
    /**
     * Display listing of categories
     */
    public function index(Request $request)
    {
        try {
            $query = Category::query();

            if ($request->status === 'active') {
                $query->active();
            } elseif ($request->status === 'inactive') {
                $query->inactive();
            }

            $categories = $query->ordered()->get();

        return response()->json([
            'code' => '000',
            'status' => 'success',
            'data' => $categories
        ]);

    } catch (\Exception $e) {
        return response()->json([
            'code' => '500',
            'status' => 'error',
            'message' => 'Failed to fetch categories',
            'error' => $e->getMessage()
        ], 500);
    }
}

public function store(Request $request)
{
    try {
        if (!auth()->user() || auth()->user()->role !== 'seller') {
            return response()->json([
                'code' => '403',
                'status' => 'error',
                'message' => 'Unauthorized access'
            ], 403);
        }

        DB::beginTransaction();
        try {
            $validator = Validator::make($request->all(), [
                'category_name' => 'required|string|max:255|unique:categories,category_name',
                'description' => 'required|string',
                'icon' => 'nullable|image|mimes:jpeg,png,jpg,svg|max:2048',
                'order' => 'nullable|integer|min:0'
            ]);

            if ($validator->fails()) {
                return response()->json([
                    'code' => '422',
                    'status' => 'error',
                    'message' => 'Validation failed',
                    'errors' => $validator->errors()
                ], 422);
    // Cek jika order sudah digunakan
    if ($request->filled('order')) {
        // Geser semua kategori yang memiliki order >= requested order
        Category::where('order', '>=', $request->order)
            ->increment('order');
    } else {
        // Jika order tidak diisi, letakkan di urutan terakhir
        $maxOrder = Category::max('order');
        $request->merge(['order' => $maxOrder + 1]);
    }

    $categoryData = [
        'category_name' => $request->category_name,
        'description' => $request->description,
        'order' => $request->order,
        'is_active' => true
    ];

    if ($request->hasFile('icon')) {
        $path = $request->file('icon')->store('categories/icons', 'public');
        $categoryData['icon'] = Storage::url($path);
    }

    $category = Category::create($categoryData);
```



```php
            DB::commit();
            return response()->json([
                'code' => '000',
                'status' => 'success',
                'message' => 'Category created successfully',
                'data' => $category
            ], 201);

        } catch (\Exception $e) {
            DB::rollBack();
            throw $e;
        }

    } catch (\Exception $e) {
        if (isset($path)) {
            Storage::disk('public')->delete($path);
        }
        return response()->json([
            'code' => '500',
            'status' => 'error',
            'message' => 'Failed to create category',
            'error' => $e->getMessage()
        ], 500);
    }
}
public function update(Request $request, $id)
{
    try {
        if (!auth()->user() || auth()->user()->role !== 'seller') {
            return response()->json([
                'code' => '403',
                'status' => 'error',
                'message' => 'Unauthorized access'
            ], 403);
        }

        DB::beginTransaction();
        try {
            $category = Category::findOrFail($id);
            $oldOrder = $category->order;

            if ($request->filled('order')) {
                $newOrder = $request->order;

                if ($newOrder != $oldOrder) {
                    if ($newOrder > $oldOrder) {
                        // Geser ke bawah: semua yang di antara old dan new berkurang 1
                        Category::where('order', '>', $oldOrder)
                            ->where('order', '<=', $newOrder)
                            ->decrement('order');
                    Category::where('order', '>=', $newOrder)
                        ->where('order', '<', $oldOrder)
                        ->increment('order');
                }

                $category->order = $newOrder;
                $category->save();
            }
        }

        // Update data lainnya
        $updateData = [];
        if ($request->filled('category_name')) {
            $updateData['category_name'] = $request->category_name;
        }
        if ($request->filled('description')) {
            $updateData['description'] = $request->description;
        }
        if ($request->has('is_active')) {
            $updateData['is_active'] = $request->boolean('is_active');
        }
public function getProducts($id, Request $request)
{
    try {
        $category = Category::findOrFail($id);

        if (!$category->is_active) {
            return response()->json([
                'code' => '404',
                'status' => 'error',
                'message' => 'Category not found'
            ], 404);
        }

        $query = $category->products()
                         ->with(['images', 'seller'])
                         ->active();

        if ($request->search) {
            $query->where('product_name', 'like', "%{$request->search}%");
        }

        if ($request->min_price) {
            $query->where('price', '>=', $request->min_price);
        }

        if ($request->max_price) {
            $query->where('price', '<=', $request->max_price);
        }
```

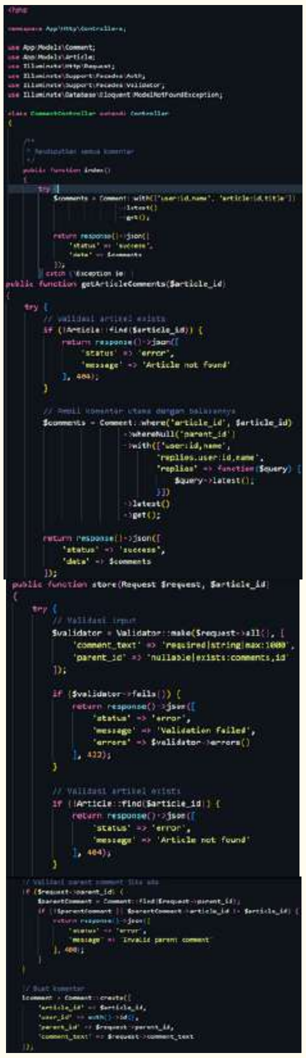


```php
// Deactivate category (seller only)
public function destroy($id)
{
    try {
        if (!auth()->user() || auth()->user()->role !== 'seller') {
            return response()->json([
                'code' => '403',
                'status' => 'error',
                'message' => 'Unauthorized access'
            ], 403);
        }

        $category = Category::findOrFail($id);

        if ($category->products()->active()->exists()) {
            return response()->json([
                'code' => '400',
                'status' => 'error',
                'message' => 'Cannot deactivate category with active products'
            ], 400);
        }

        $category->update(['is_active' => false]);
        $category->update(['is_active' => false]);

        return response()->json([
            'code' => '000',
            'status' => 'success',
            'message' => 'Category has been deactivated successfully'
        ]);

    } catch (\Exception $e) {
        return response()->json([
            'code' => '500',
            'status' => 'error',
            'message' => 'Failed to deactivate category',
            'error' => $e->getMessage()
        ], 500);
    }
}
    public function restore($id)
    {
        try {
            if (!auth()->user() || auth()->user()->role !== 'seller') {
                return response()->json([
                    'code' => '403',
                    'status' => 'error',
                    'message' => 'Unauthorized access'
                ], 403);
            }
            $category = Category::findOrFail($id);
            $category->update(['is_active' => true]);

            return response()->json([
                'code' => '000',
                'status' => 'success',
                'message' => 'Category has been activated successfully'
            ]);

        } catch (\Exception $e) {
            return response()->json([
                'code' => '500',
                'status' => 'error',
                'message' => 'Failed to activate category',
                'error' => $e->getMessage()
            ], 500);
        }
    }
```

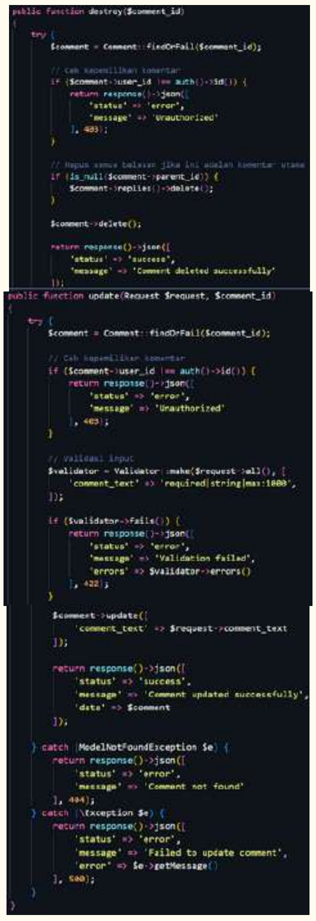
API ini digunakan untuk mengelola kategori, termasuk operasi untuk menampilkan, menambah, memperbarui, menghapus, dan mengaktifkan/deaktivasi kategori. Hanya penjual yang dapat menambah, memperbarui, atau menghapus kategori. Selain itu, API ini juga menyediakan fungsi untuk mendapatkan produk berdasarkan kategori dan mengurutkan kategori untuk memperbaiki urutan.

API ENDPOINT BETTABEAL

COMMENT CONTROLLER



```php
<?php

namespace App\Http\Controllers;

use App\Models\Comment;
use App\Models\Article;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Validator;
use Illuminate\Database\Eloquent\ModelNotFoundException;

class CommentController extends Controller
{

    /**
     * Mendapatkan semua komentar
     */
    public function index()
    {
        try {
            $comments = Comment::with(['user:id,name', 'article:id,title'])
                            ->latest()
                            ->get();

            return response()->json([
                'status' => 'success',
                'data' => $comments
            ]);
        } catch (\Exception $e) {
public function getArticleComments($article_id)
{
    try {
        // Validasi artikel exists
        if (!Article::find($article_id)) {
            return response()->json([
                'status' => 'error',
                'message' => 'Article not found'
            ], 404);
        }

        // Ambil komentar utama dengan balasannya
        $comments = Comment::where('article_id', $article_id)
                        ->whereNull('parent_id')
                        ->with(['user:id,name',
                            'replies.user:id,name',
                            'replies' => function($query) {
                                $query->latest();
                            }])
                        ->latest()
                        ->get();

        return response()->json([
            'status' => 'success',
            'data' => $comments
        ]);
public function store(Request $request, $article_id)
{
    try {
        // Validasi input
        $validator = Validator::make($request->all(), [
            'comment_text' => 'required|string|max:1000',
            'parent_id' => 'nullable|exists:comments,id'
        ]);

        if ($validator->fails()) {
            return response()->json([
                'status' => 'error',
                'message' => 'Validation failed',
                'errors' => $validator->errors()
            ], 422);
        }

        // Validasi artikel exists
        if (!Article::find($article_id)) {
            return response()->json([
                'status' => 'error',
                'message' => 'Article not found'
            ], 404);
        }
// Validasi parent comment jika ada
if ($request->parent_id) {
    $parentComment = Comment::find($request->parent_id);
    if (!$parentComment || $parentComment->article_id != $article_id) {
        return response()->json([
            'status' => 'error',
            'message' => 'Invalid parent comment'
        ], 400);
    }
}

// Buat komentar
$comment = Comment::create([
    'article_id' => $article_id,
    'user_id' => auth()->id(),
    'parent_id' => $request->parent_id,
    'comment_text' => $request->comment_text
]);
```

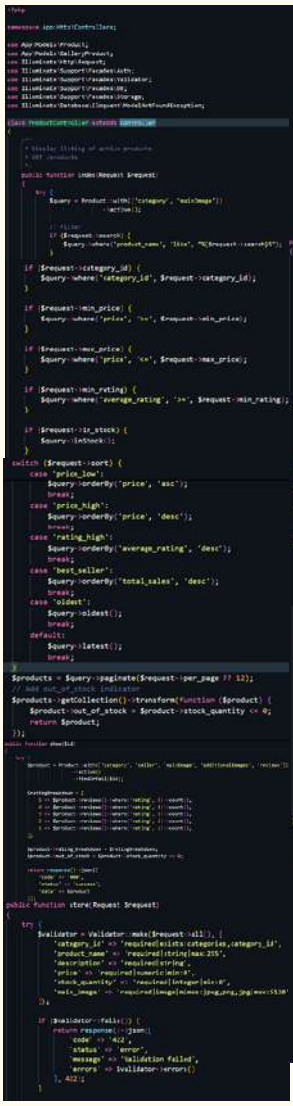


```php
public function destroy($comment_id)
{
    try {
        $comment = Comment::findOrFail($comment_id);

        // Cek kepemilikan komentar
        if ($comment->user_id !== auth()->id()) {
            return response()->json([
                'status' => 'error',
                'message' => 'Unauthorized'
            ], 403);
        }

        // Hapus semua balasan jika ini adalah komentar utama
        if (is_null($comment->parent_id)) {
            $comment->replies()->delete();
        }

        $comment->delete();

        return response()->json([
            'status' => 'success',
            'message' => 'Comment deleted successfully'
        ]);
public function update(Request $request, $comment_id)
{
    try {
        $comment = Comment::findOrFail($comment_id);

        // Cek kepemilikan komentar
        if ($comment->user_id !== auth()->id()) {
            return response()->json([
                'status' => 'error',
                'message' => 'Unauthorized'
            ], 403);
        }

        // Validasi input
        $validator = Validator::make($request->all(), [
            'comment_text' => 'required|string|max:1000',
        ]);

        if ($validator->fails()) {
            return response()->json([
                'status' => 'error',
                'message' => 'Validation failed',
                'errors' => $validator->errors()
            ], 422);
        }

        $comment->update([
            'comment_text' => $request->comment_text
        ]);

        return response()->json([
            'status' => 'success',
            'message' => 'Comment updated successfully',
            'data' => $comment
        ]);

    } catch (ModelNotFoundException $e) {
        return response()->json([
            'status' => 'error',
            'message' => 'Comment not found'
        ], 404);
    } catch (\Exception $e) {
        return response()->json([
            'status' => 'error',
            'message' => 'Failed to update comment',
            'error' => $e->getMessage()
        ], 500);
    }
}
```

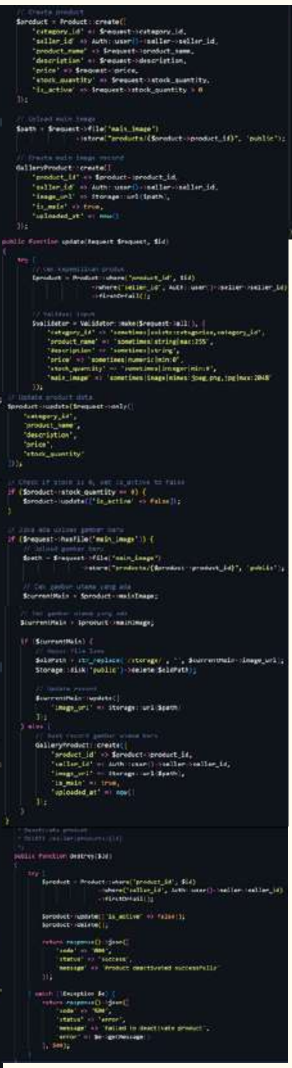
API ini digunakan untuk mengelola komentar, termasuk operasi untuk menampilkan semua komentar, mendapatkan komentar untuk artikel tertentu, menyimpan komentar baru, memperbarui, dan menghapus komentar.

API ENDPOINT BETTABEAL

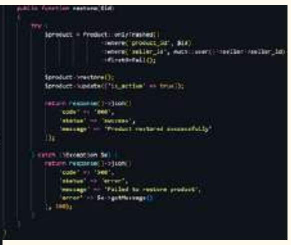
PRODUCT CONTROLLER



```php
<?php

namespace App\Http\Controllers;

use App\Models\Product;
use App\Models\GalleryProduct;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Storage;
use Illuminate\Database\Eloquent\ModelNotFoundException;

class ProductController extends Controller
{
    /**
     * Display listing of active products
     * GET /products
     */
    public function index(Request $request)
    {
        try {
            $query = Product::with(['category', 'mainImage'])
                        ->active();

            // Filter
            if ($request->search) {
                $query->where('product_name', 'like', "%{$request->search}%");
            }
    if ($request->category_id) {
        $query->where('category_id', $request->category_id);
    }

    if ($request->min_price) {
        $query->where('price', '>=', $request->min_price);
    }

    if ($request->max_price) {
        $query->where('price', '<=', $request->max_price);
    }

    if ($request->min_rating) {
        $query->where('average_rating', '>=', $request->min_rating);
    }

    if ($request->in_stock) {
        $query->inStock();
    }
switch ($request->sort) {
    case 'price_low':
        $query->orderBy('price', 'asc');
        break;
    case 'price_high':
        $query->orderBy('price', 'desc');
        break;
    case 'rating_high':
        $query->orderBy('average_rating', 'desc');
        break;
    case 'best_seller':
        $query->orderBy('total_sales', 'desc');
        break;
    case 'oldest':
        $query->oldest();
        break;
    default:
        $query->latest();
        break;
}
$products = $query->paginate($request->per_page ?? 12);
// Add out_of_stock indicator
$products->getCollection()->transform(function ($product) {
    $product->out_of_stock = $product->stock_quantity <= 0;
    return $product;
});
public function show($id)
{
    try {
        $product = Product::with(['category', 'seller', 'mainImage', 'additionalImages', 'reviews'])
                    ->active()
                    ->findOrFail($id);

        $ratingBreakdown = [
            5 => $product->reviews()->where('rating', 5)->count(),
            4 => $product->reviews()->where('rating', 4)->count(),
            3 => $product->reviews()->where('rating', 3)->count(),
            2 => $product->reviews()->where('rating', 2)->count(),
            1 => $product->reviews()->where('rating', 1)->count(),
        ];

        $product->rating_breakdown = $ratingBreakdown;
        $product->out_of_stock = $product->stock_quantity <= 0;

        return response()->json([
            'code' => '200',
            'status' => 'success',
            'data' => $product
        ]);
public function store(Request $request)
{
    try {
        $validator = Validator::make($request->all(), [
            'category_id' => 'required|exists:categories,category_id',
            'product_name' => 'required|string|max:255',
            'description' => 'required|string',
            'price' => 'required|numeric|min:0',
            'stock_quantity' => 'required|integer|min:0',
            'main_image' => 'required|image|mimes:jpeg,png,jpg|max:5530'
        ]);

        if ($validator->fails()) {
            return response()->json([
                'code' => '422',
                'status' => 'error',
                'message' => 'Validation failed',
                'errors' => $validator->errors()
            ], 422);
        }
```



```php
// Create product
$product = Product::create([
    'category_id' => $request->category_id,
    'seller_id' => Auth::user()->seller->seller_id,
    'product_name' => $request->product_name,
    'description' => $request->description,
    'price' => $request->price,
    'stock_quantity' => $request->stock_quantity,
    'is_active' => $request->stock_quantity > 0
]);

// Upload main image
$path = $request->file('main_image')
            ->store("products/{$product->product_id}", 'public');

// Create main image record
GalleryProduct::create([
    'product_id' => $product->product_id,
    'seller_id' => Auth::user()->seller->seller_id,
    'image_url' => Storage::url($path),
    'is_main' => true,
    'uploaded_at' => now()
]);
public function update(Request $request, $id)
{
    try {
        // Cek kepemilikan produk
        $product = Product::where('product_id', $id)
                    ->where('seller_id', Auth::user()->seller->seller_id)
                    ->firstOrFail();

        // Validasi input
        $validator = Validator::make($request->all(), [
            'category_id' => 'sometimes|exists:categories,category_id',
            'product_name' => 'sometimes|string|max:255',
            'description' => 'sometimes|string',
            'price' => 'sometimes|numeric|min:0',
            'stock_quantity' => 'sometimes|integer|min:0',
            'main_image' => 'sometimes|image|mimes:jpeg,png,jpg|max:2048'
        ]);
// Update product data
$product->update($request->only([
    'category_id',
    'product_name',
    'description',
    'price',
    'stock_quantity'
]));

// Check if stock is 0, set is_active to false
if ($product->stock_quantity == 0) {
    $product->update(['is_active' => false]);
}

// Jika ada upload gambar baru
if ($request->hasFile('main_image')) {
    // Upload gambar baru
    $path = $request->file('main_image')
                ->store("products/{$product->product_id}", 'public');

    // Cek gambar utama yang ada
    $currentMain = $product->mainImage;
    // Cek gambar utama yang ada
    $currentMain = $product->mainImage;

    if ($currentMain) {
        // Hapus file lama
        $oldPath = str_replace('/storage/', '', $currentMain->image_url);
        Storage::disk('public')->delete($oldPath);

        // Update record
        $currentMain->update([
            'image_url' => Storage::url($path)
        ]);
    } else {
        // Buat record gambar utama baru
        GalleryProduct::create([
            'product_id' => $product->product_id,
            'seller_id' => Auth::user()->seller->seller_id,
            'image_url' => Storage::url($path),
            'is_main' => true,
            'uploaded_at' => now()
        ]);
    }
}
    * Deactivate product
    * DELETE /seller/products/{id}
    */
public function destroy($id)
{
    try {
        $product = Product::where('product_id', $id)
                    ->where('seller_id', Auth::user()->seller->seller_id)
                    ->firstOrFail();

        $product->update(['is_active' => false]);
        $product->delete();

        return response()->json([
            'code' => '200',
            'status' => 'success',
            'message' => 'Product deactivated successfully'
        ]);

    } catch (\Exception $e) {
        return response()->json([
            'code' => '500',
            'status' => 'error',
            'message' => 'Failed to deactivate product',
            'error' => $e->getMessage()
        ], 500);
    }
```

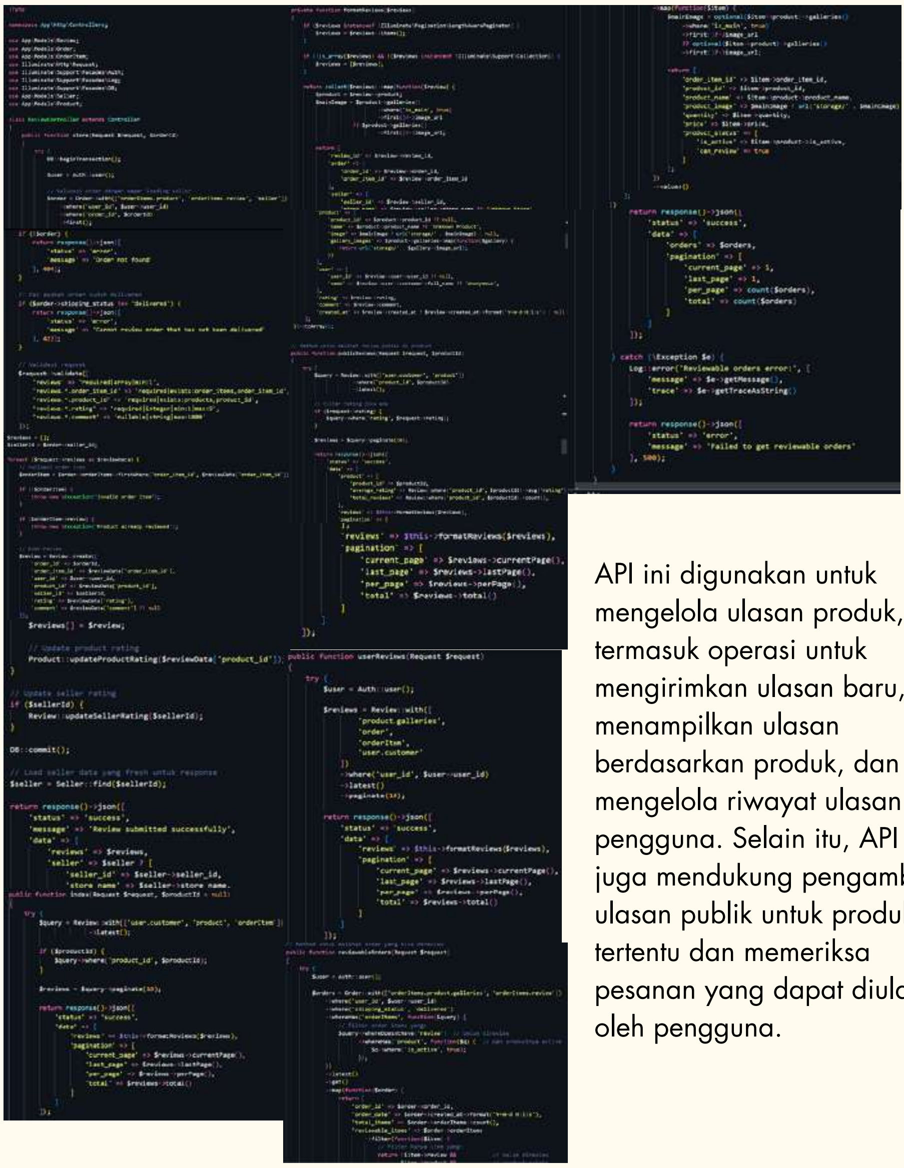


```php
public function restore($id)
{
    try {
        $product = Product::onlyTrashed()
                    ->where('product_id', $id)
                    ->where('seller_id', Auth::user()->seller->seller_id)
                    ->firstOrFail();

        $product->restore();
        $product->update(['is_active' => true]);

        return response()->json([
            'code' => '200',
            'status' => 'success',
            'message' => 'Product restored successfully'
        ]);

    } catch (\Exception $e) {
        return response()->json([
            'code' => '500',
            'status' => 'error',
            'message' => 'Failed to restore product',
            'error' => $e->getMessage()
        ], 500);
    }
}
```

API ini digunakan untuk mengelola produk, termasuk operasi untuk menampilkan daftar produk aktif, menampilkan detail produk, menambahkan produk baru, memperbarui produk yang ada, menonaktifkan produk, dan mengembalikan produk yang dinonaktifkan. Selain itu, API ini juga mendukung pengelolaan produk oleh penjual, termasuk validasi dan pengelolaan gambar utama untuk produk.

API ENDPOINT BETTABEAL

REVIEW CONTROLLER



```php
<?php

namespace App\Http\Controllers;

use App\Models\Review;
use App\Models\Order;
use App\Models\OrderItem;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\DB;
use App\Models\Seller;
use App\Models\Product;

class ReviewController extends Controller
{
    public function store(Request $request, $orderId)
    {
        try {
            DB::beginTransaction();

            $user = Auth::user();

            // [illegible]
            $order = Order::with(['orderItems.product', 'orderItems.review', 'seller'])
                ->where('user_id', $user->user_id)
                ->where('order_id', $orderId)
                ->first();
  if (!$order) {
      return response()->json([
          'status' => 'error',
          'message' => 'Order not found'
      ], 404);
  }

  // [illegible]
  if ($order->shipping_status !== 'delivered') {
      return response()->json([
          'status' => 'error',
          'message' => 'Cannot review order that has not been delivered'
      ], 422);
  }

  // Validasi request
  $request->validate([
      'reviews' => 'required|array|min:1',
      'reviews.*.order_item_id' => 'required|exists:order_items,order_item_id',
      'reviews.*.product_id' => 'required|exists:products,product_id',
      'reviews.*.rating' => 'required|integer|min:1|max:5',
      'reviews.*.comment' => 'nullable|string|max:1000'
  ]);
$reviews = [];
$sellerId = $order->seller_id;

foreach ($request->reviews as $reviewData) {
  // [illegible]
  $orderItem = $order->orderItems->firstWhere('order_item_id', $reviewData['order_item_id']);

  if (!$orderItem) {
      throw new \Exception('Invalid order item');
  }

  if ($orderItem->review) {
      throw new \Exception('Product already reviewed');
  }

  // [illegible]
  $review = Review::create([
      'order_id' => $orderId,
      'order_item_id' => $reviewData['order_item_id'],
      'user_id' => $user->user_id,
      'product_id' => $reviewData['product_id'],
      'seller_id' => $sellerId,
      'rating' => $reviewData['rating'],
      'comment' => $reviewData['comment'] ?? null
  ]);
    $reviews[] = $review;

    // Update product rating
    Product::updateProductRating($reviewData['product_id']);
}

// Update seller rating
if ($sellerId) {
    Review::updateSellerRating($sellerId);
}

DB::commit();

// Load seller data yang fresh untuk response
$seller = Seller::find($sellerId);

return response()->json([
    'status' => 'success',
    'message' => 'Review submitted successfully',
    'data' => [
        'reviews' => $reviews,
        'seller' => $seller ? [
            'seller_id' => $seller->seller_id,
            'store_name' => $seller->store_name,
public function index(Request $request, $productId = null)
{
    try {
        $query = Review::with(['user.customer', 'product', 'orderItem'])
                  ->latest();

        if ($productId) {
            $query->where('product_id', $productId);
        }

        $reviews = $query->paginate(10);

        return response()->json([
            'status' => 'success',
            'data' => [
                'reviews' => $this->formatReviews($reviews),
                'pagination' => [
                    'current_page' => $reviews->currentPage(),
                    'last_page' => $reviews->lastPage(),
                    'per_page' => $reviews->perPage(),
                    'total' => $reviews->total()
                ]
            ]
        ]);
```

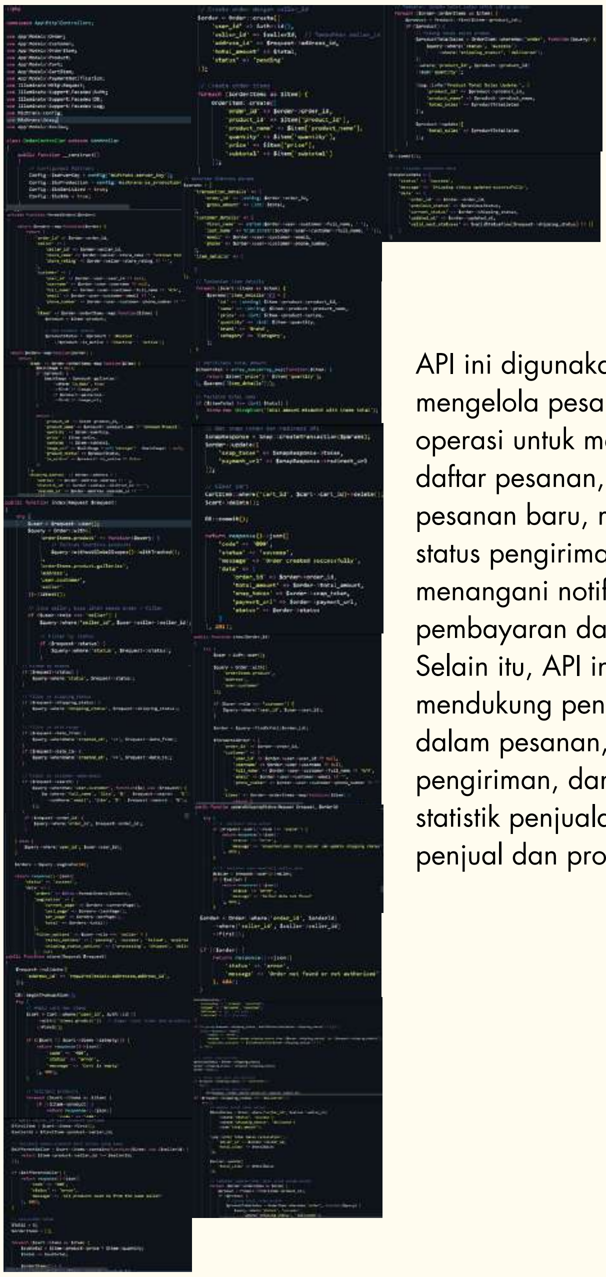
```php
private function formatReviews($reviews)
{
    if ($reviews instanceof \Illuminate\Pagination\LengthAwarePaginator) {
        $reviews = $reviews->items();
    }

    if (!is_array($reviews) && !($reviews instanceof \Illuminate\Support\Collection)) {
        $reviews = [$reviews];
    }

    return collect($reviews)->map(function($review) {
        $product = $review->product;
        $mainImage = $product->galleries
                        ->where('is_main', true)
                        ->first()?->image_url
                    ?? $product->galleries
                        ->first()?->image_url;

        return [
            'review_id' => $review->review_id,
            'order' => [
                'order_id' => $review->order_id,
                'order_item_id' => $review->order_item_id
            ],
            'seller' => [
                'seller_id' => $review->seller_id,
                [illegible]
            ],
            'product' => [
                'product_id' => $review->product->product_id ?? null,
                'name' => $review->product->product_name ?? 'Unknown Product',
                'image' => $mainImage ? url('storage/' . $mainImage) : null,
                'gallery_images' => $product->galleries->map(function($gallery) {
                    return url('storage/' . $gallery->image_url);
                })
            ],
            'user' => [
                'user_id' => $review->user->user_id ?? null,
                'name' => $review->user->customer->full_name ?? 'Anonymous',
            ],
            'rating' => $review->rating,
            'comment' => $review->comment,
            'created_at' => $review->created_at ? $review->created_at->format('Y-m-d H:i:s') : null
        ];
    })->toArray();
}

// [illegible]
public function publicReviews(Request $request, $productId)
{
    try {
        $query = Review::with(['user.customer', 'product'])
                ->where('product_id', $productId)
                ->latest();

        // [illegible]
        if ($request->rating) {
            $query->where('rating', $request->rating);
        }

        $reviews = $query->paginate(10);

        return response()->json([
            'status' => 'success',
            'data' => [
                'product' => [
                    'product_id' => $productId,
                    'average_rating' => Review::where('product_id', $productId)->avg('rating'),
                    'total_reviews' => Review::where('product_id', $productId)->count(),
                ],
                'reviews' => $this->formatReviews($reviews),
                'pagination' => [
                ],
              'reviews' => $this->formatReviews($reviews),
              'pagination' => [
                  'current_page' => $reviews->currentPage(),
                  'last_page' => $reviews->lastPage(),
                  'per_page' => $reviews->perPage(),
                  'total' => $reviews->total()
              ]
          ]
      ]);
public function userReviews(Request $request)
{
    try {
        $user = Auth::user();

        $reviews = Review::with([
                'product.galleries',
                'order',
                'orderItem',
                'user.customer'
            ])
            ->where('user_id', $user->user_id)
            ->latest()
            ->paginate(10);

        return response()->json([
            'status' => 'success',
            'data' => [
                'reviews' => $this->formatReviews($reviews),
                'pagination' => [
                    'current_page' => $reviews->currentPage(),
                    'last_page' => $reviews->lastPage(),
                    'per_page' => $reviews->perPage(),
                    'total' => $reviews->total()
                ]
            ]
        ]);
// [illegible]
public function reviewableOrders(Request $request)
{
    try {
        $user = Auth::user();

        $orders = Order::with(['orderItems.product.galleries', 'orderItems.review'])
            ->where('user_id', $user->user_id)
            ->where('shipping_status', 'delivered')
            ->whereHas('orderItems', function($query) {
                // [illegible]
                $query->whereDoesntHave('review') // [illegible]
                    ->whereHas('product', function($q) { // [illegible]
                        $q->where('is_active', true);
                    });
            })
            ->latest()
            ->get()
            ->map(function($order) {
                return [
                    'order_id' => $order->order_id,
                    'order_date' => $order->created_at->format('Y-m-d H:i:s'),
                    'total_items' => $order->orderItems->count(),
                    'reviewable_items' => $order->orderItems
                        ->filter(function($item) {
                            // [illegible]
                            return !$item->review && [illegible]
```

```php
                ->map(function($item) {
                    $mainImage = optional($item->product->galleries()
                        ->where('is_main', true)
                        ->first())->image_url
                        ?? optional($item->product)->galleries()
                        ->first()?->image_url;

                    return [
                        'order_item_id' => $item->order_item_id,
                        'product_id' => $item->product_id,
                        'product_name' => $item->product->product_name,
                        'product_image' => $mainImage ? url('storage/' . $mainImage) : null,
                        'quantity' => $item->quantity,
                        'price' => $item->price,
                        'product_status' => [
                            'is_active' => $item->product->is_active,
                            'can_review' => true
                        ]
                    ];
                })
                ->values()
            ];
        });
            return response()->json([
                'status' => 'success',
                'data' => [
                    'orders' => $orders,
                    'pagination' => [
                        'current_page' => 1,
                        'last_page' => 1,
                        'per_page' => count($orders),
                        'total' => count($orders)
                    ]
                ]
            ]);

        } catch (\Exception $e) {
            Log::error('Reviewable orders error:', [
                'message' => $e->getMessage(),
                'trace' => $e->getTraceAsString()
            ]);

            return response()->json([
                'status' => 'error',
                'message' => 'Failed to get reviewable orders'
            ], 500);
        }
    }
```

API ini digunakan untuk mengelola ulasan produk, termasuk operasi untuk mengirimkan ulasan baru, menampilkan ulasan berdasarkan produk, dan mengelola riwayat ulasan pengguna. Selain itu, API ini juga mendukung pengambilan ulasan publik untuk produk tertentu dan memeriksa pesanan yang dapat diulas oleh pengguna.

API ENDPOINT BETTABEAL

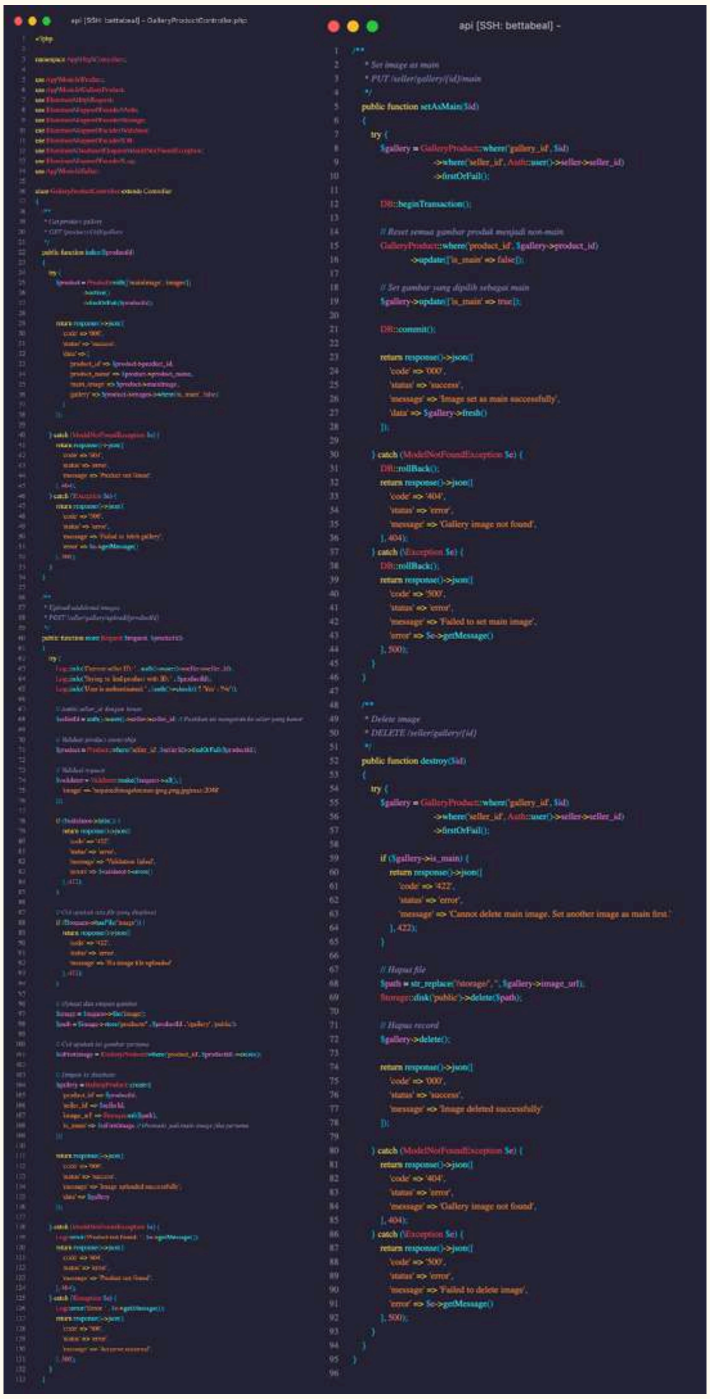
ORDER CONTROLLER


[illegible]

API ini digunakan untuk mengelola pesanan, termasuk operasi untuk menampilkan daftar pesanan, membuat pesanan baru, memperbarui status pengiriman, dan menangani notifikasi pembayaran dari Midtrans. Selain itu, API ini juga mendukung pengelolaan item dalam pesanan, validasi status pengiriman, dan pembaruan statistik penjualan untuk penjual dan produk terkait.

API ENDPOINT BETTABEAL

GALLERY PRODUCT CONTROLLER

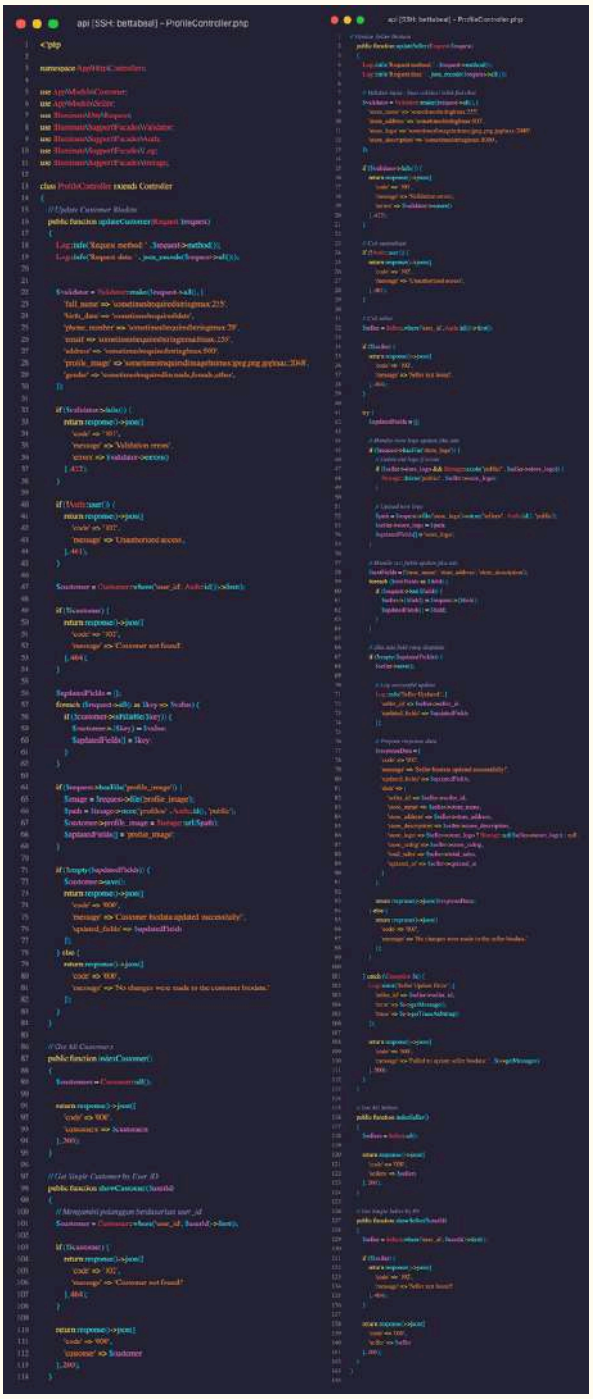


```
<?php

namespace App\Http\Controllers;

use App\Models\Product;
use App\Models\GalleryProduct;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Storage;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\DB;
use Illuminate\Database\Eloquent\ModelNotFoundException;
use Illuminate\Support\Facades\Log;
use App\Models\Seller;

class GalleryProductController extends Controller
{
    /**
     * Get product gallery
     * GET /products/{id}/gallery
     */
    public function index($productId)
    {
        try {
            $product = Product::with(['mainImage', 'images'])
                        ->active()
                        ->findOrFail($productId);

            return response()->json([
                'code' => '000',
                'status' => 'success',
                'data' => [
                    'product_id' => $product->product_id,
                    'product_name' => $product->product_name,
                    'main_image' => $product->mainImage,
                    'gallery' => $product->images->where('is_main', false)
                ]
            ]);

        } catch (ModelNotFoundException $e) {
            return response()->json([
                'code' => '404',
                'status' => 'error',
                'message' => 'Product not found'
            ], 404);
        } catch (\Exception $e) {
            return response()->json([
                'code' => '500',
                'status' => 'error',
                'message' => 'Failed to fetch gallery',
                'error' => $e->getMessage()
            ], 500);
        }
    }

    /**
     * Upload additional images
     * POST /seller/gallery/upload/{productId}
     */
    public function store(Request $request, $productId)
    {
        try {
            Log::info('Current seller ID: ' . auth()->user()->seller->seller_id);
            Log::info('Trying to find product with ID: ' . $productId);
            Log::info('User is authenticated: ' . (auth()->check() ? 'Yes' : 'No'));

            // Ambil seller_id dengan benar
            $sellerId = auth()->user()->seller->seller_id; // Pastikan ini mengambil dari seller yang benar

            // Validasi produk ownership
            $product = Product::where('seller_id', $sellerId)->findOrFail($productId);

            // Validasi request
            $validator = Validator::make($request->all(), [
                'image' => 'required|image|mimes:jpeg,png,jpg|max:2048'
            ]);

            if ($validator->fails()) {
                return response()->json([
                    'code' => '422',
                    'status' => 'error',
                    'message' => 'Validation failed',
                    'errors' => $validator->errors()
                ], 422);
            }

            // Cek apakah ada file yang dikirim
            if (!$request->hasFile('image')) {
                return response()->json([
                    'code' => '422',
                    'status' => 'error',
                    'message' => 'No image file uploaded'
                ], 422);
            }

            // Upload dan simpan gambar
            $image = $request->file('image');
            $path = $image->store('products/' . $productId . '/gallery', 'public');

            // Cek apakah ini gambar pertama
            $isFirstImage = GalleryProduct::where('product_id', $productId)->doesntExist();

            // Simpan ke database
            $gallery = GalleryProduct::create([
                'product_id' => $productId,
                'seller_id' => $sellerId,
                'image_url' => Storage::url($path),
                'is_main' => $isFirstImage // Otomatis jadi main image jika pertama
            ]);

            return response()->json([
                'code' => '000',
                'status' => 'success',
                'message' => 'Image uploaded successfully',
                'data' => $gallery
            ]);

        } catch (ModelNotFoundException $e) {
            Log::error('Product not found: ' . $e->getMessage());
            return response()->json([
                'code' => '404',
                'status' => 'error',
                'message' => 'Product not found'
            ], 404);
        } catch (\Exception $e) {
            Log::error('Error: ' . $e->getMessage());
            return response()->json([
                'code' => '500',
                'status' => 'error',
                'message' => 'An error occurred'
            ], 500);
        }
    }
```

```
/**
 * Set image as main
 * PUT /seller/gallery/{id}/main
 */
public function setAsMain($id)
{
    try {
        $gallery = GalleryProduct::where('gallery_id', $id)
                    ->where('seller_id', Auth::user()->seller->seller_id)
                    ->firstOrFail();

        DB::beginTransaction();

        // Reset semua gambar produk menjadi non-main
        GalleryProduct::where('product_id', $gallery->product_id)
            ->update(['is_main' => false]);

        // Set gambar yang dipilih sebagai main
        $gallery->update(['is_main' => true]);

        DB::commit();

        return response()->json([
            'code' => '000',
            'status' => 'success',
            'message' => 'Image set as main successfully',
            'data' => $gallery->fresh()
        ]);

    } catch (ModelNotFoundException $e) {
        DB::rollBack();
        return response()->json([
            'code' => '404',
            'status' => 'error',
            'message' => 'Gallery image not found',
        ], 404);
    } catch (\Exception $e) {
        DB::rollBack();
        return response()->json([
            'code' => '500',
            'status' => 'error',
            'message' => 'Failed to set main image',
            'error' => $e->getMessage()
        ], 500);
    }
}

/**
 * Delete image
 * DELETE /seller/gallery/{id}
 */
public function destroy($id)
{
    try {
        $gallery = GalleryProduct::where('gallery_id', $id)
                    ->where('seller_id', Auth::user()->seller->seller_id)
                    ->firstOrFail();

        if ($gallery->is_main) {
            return response()->json([
                'code' => '422',
                'status' => 'error',
                'message' => 'Cannot delete main image. Set another image as main first.'
            ], 422);
        }

        // Hapus file
        $path = str_replace('/storage/', '', $gallery->image_url);
        Storage::disk('public')->delete($path);

        // Hapus record
        $gallery->delete();

        return response()->json([
            'code' => '000',
            'status' => 'success',
            'message' => 'Image deleted successfully'
        ]);

    } catch (ModelNotFoundException $e) {
        return response()->json([
            'code' => '404',
            'status' => 'error',
            'message' => 'Gallery image not found',
        ], 404);
    } catch (\Exception $e) {
        return response()->json([
            'code' => '500',
            'status' => 'error',
            'message' => 'Failed to delete image',
            'error' => $e->getMessage()
        ], 500);
    }
}
}
```

API ENDPOINT BETTABEAL

PROFILE CONTROLLER



api [SSH: bettabeal] – ProfileController.php

```php
<?php

namespace App\Http\Controllers;

use App\Models\Customer;
use App\Models\Seller;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Log;
use Illuminate\Support\Facades\Storage;

class ProfileController extends Controller
{
  // Update Customer Biodata
  public function updateCustomer(Request $request)
  {
    Log::info('Request method: ' . $request->method());
    Log::info('Request data: ' . json_encode($request->all()));


    $validator = Validator::make($request->all(), [
      'full_name' => 'sometimes|required|string|max:225',
      'birth_date' => 'sometimes|required|date',
      'phone_number' => 'sometimes|required|string|max:20',
      'email' => 'sometimes|required|string|email|max:255',
      'address' => 'sometimes|required|string|max:500',
      'profile_image' => 'sometimes|required|image|mimes:jpeg,png,jpg|max:2048',
      'gender' => 'sometimes|required|in:male,female,other',
    ]);

    if ($validator->fails()) {
      return response()->json([
        'code' => '001',
        'message' => 'Validation errors',
        'errors' => $validator->errors()
      ], 422);
    }

    if (!Auth::user()) {
      return response()->json([
        'code' => '401',
        'message' => 'Unauthorized access',
      ], 401);
    }

    $customer = Customer::where('user_id', Auth::id())->first();

    if (!$customer) {
      return response()->json([
        'code' => '002',
        'message' => 'Customer not found'
      ], 404);
    }

    $updatedFields = [];
    foreach ($request->all() as $key => $value) {
      if ($customer->isFillable($key)) {
        $customer->{$key} = $value;
        $updatedFields[] = $key;
      }
    }

    if ($request->hasFile('profile_image')) {
      $image = $request->file('profile_image');
      $path = $image->store('profiles/' . Auth::id(), 'public');
      $customer->profile_image = Storage::url($path);
      $updatedFields[] = 'profile_image';
    }

    if (!empty($updatedFields)) {
      $customer->save();
      return response()->json([
        'code' => '000',
        'message' => 'Customer biodata updated successfully',
        'updated_fields' => $updatedFields
      ]);
    } else {
      return response()->json([
        'code' => '000',
        'message' => 'No changes were made to the customer biodata.'
      ]);
    }
  }

  // Get All Customers
  public function indexCustomer()
  {
    $customers = Customer::all();

    return response()->json([
      'code' => '000',
      'customers' => $customers
    ], 200);
  }

  // Get Single Customer by User ID
  public function showCustomer($userId)
  {
    // Mengambil pelanggan berdasarkan user_id
    $customer = Customer::where('user_id', $userId)->first();

    if (!$customer) {
      return response()->json([
        'code' => '002',
        'message' => 'Customer not found!'
      ], 404);
    }

    return response()->json([
      'code' => '000',
      'customer' => $customer
    ], 200);
  }
```

api [SSH: bettabeal] – ProfileController.php

```php
  // Update Seller Biodata
  public function updateSeller(Request $request)
  {
    Log::info('Request method: ' . $request->method());
    Log::info('Request data: ' . json_encode($request->all()));

    // Validasi input - [illegible]
    $validator = Validator::make($request->all(), [
      'store_name' => 'sometimes|string|max:255',
      'store_address' => 'sometimes|string|max:500',
      'store_logo' => 'sometimes|file|image|mimes:jpeg,png,jpg|max:2048',
      'store_description' => 'sometimes|string|max:1000',
    ]);

    if ($validator->fails()) {
      return response()->json([
        'code' => '001',
        'message' => 'Validation errors',
        'errors' => $validator->errors()
      ], 422);
    }

    // Cek autentikasi
    if (!Auth::user()) {
      return response()->json([
        'code' => '401',
        'message' => 'Unauthorized access',
      ], 401);
    }

    // Cek seller
    $seller = Seller::where('user_id', Auth::id())->first();

    if (!$seller) {
      return response()->json([
        'code' => '002',
        'message' => 'Seller not found'
      ], 404);
    }

    try {
      $updatedFields = [];

      // Handle store logo update jika ada
      if ($request->hasFile('store_logo')) {
        // Hapus old logo jika ada
        if ($seller->store_logo && Storage::exists('public/' . $seller->store_logo)) {
          Storage::delete('public/' . $seller->store_logo);
        }

        // Upload new logo
        $path = $request->file('store_logo')->store('sellers/' . Auth::id(), 'public');
        $seller->store_logo = $path;
        $updatedFields[] = 'store_logo';
      }

      // Handle text fields update jika ada
      $textFields = ['store_name', 'store_address', 'store_description'];
      foreach ($textFields as $field) {
        if ($request->has($field)) {
          $seller->{$field} = $request->{$field};
          $updatedFields[] = $field;
        }
      }

      // Jika ada field yang diupdate
      if (!empty($updatedFields)) {
        $seller->save();

        // Log successful update
        Log::info('Seller Updated', [
          'seller_id' => $seller->seller_id,
          'updated_fields' => $updatedFields
        ]);

        // Prepare response data
        $responseData = [
          'code' => '000',
          'message' => 'Seller biodata updated successfully',
          'updated_fields' => $updatedFields,
          'data' => [
            'seller_id' => $seller->seller_id,
            'store_name' => $seller->store_name,
            'store_address' => $seller->store_address,
            'store_description' => $seller->store_description,
            'store_logo' => $seller->store_logo ? Storage::url($seller->store_logo) : null,
            'store_rating' => $seller->store_rating,
            'total_sales' => $seller->total_sales,
            'updated_at' => $seller->updated_at
          ]
        ];

        return response()->json($responseData);
      } else {
        return response()->json([
          'code' => '000',
          'message' => 'No changes were made to the seller biodata.'
        ]);
      }

    } catch (\Exception $e) {
      Log::error('Seller Update Error', [
        'seller_id' => $seller->seller_id,
        'error' => $e->getMessage(),
        'trace' => $e->getTraceAsString()
      ]);

      return response()->json([
        'code' => '500',
        'message' => 'Failed to update seller biodata: ' . $e->getMessage()
      ], 500);
    }
  }

  // Get All Sellers
  public function indexSeller()
  {
    $sellers = Seller::all();

    return response()->json([
      'code' => '000',
      'sellers' => $sellers
    ], 200);
  }

  // Get Single Seller by ID
  public function showSeller($userId)
  {
    $seller = Seller::where('user_id', $userId)->first();

    if (!$seller) {
      return response()->json([
        'code' => '002',
        'message' => 'Seller not found!'
      ], 404);
    }

    return response()->json([
      'code' => '000',
      'seller' => $seller
    ], 200);
  }
}
```

THANK *You*

MASTEAL