Frontend Plan for index.html and Auth Pages (Login, Register, Forgot Password, Verify Email)

---

1. index.html

Location: frontend/index.html
Related assets:

CSS: css/man.css (core styles), assets/components.css (UI components), assets/themes.css (color themes), css/responsive.css (media queries)

JS: assets/home.js (homepage logic), assets/toast.js (notifications), assets/utils.js (utilities), assets/router.js (client-side routing if SPA)

Design Plan:

Navbar:

Desktop: Logo on left, right-aligned links “Login”, “Register”

Mobile: Logo left, hamburger menu right (toggles slide-out menu with links: Login, Register, About, Contact)

Use CSS grid/flexbox for layout, media queries in responsive.css for breakpoint adjustments

Hero Section:

Full viewport height, engaging shopping-related background image (optimised with modern formats, lazy-loaded)

Centered main headline + subheading + primary CTA button (“Get Started”) → navigates to register page

Text shadows and overlay to ensure readability on image background

Info Section (desktop only):

2-3 horizontally aligned cards or sections highlighting About Us, Contact, Features

Hidden on mobile for clarity and performance

Use CSS grid for card layout, consistent spacing and typography

Footer:

Minimalist with copyright, possibly quick links

Fixed or static depending on content height

JS Functionality:

home.js:

Handle hero CTA button click navigation

Initialize mobile menu toggle (hamburger) with ARIA roles for accessibility

toast.js: Show any user messages or alerts (e.g., “Welcome to [AppName]!”)

utils.js: Helper functions (e.g., debounce for menu toggle, element selectors)

router.js (optional): If SPA, handle navigation without full page reloads

---

2. Auth Pages Group

Pages:

login.html (frontend/login.html)

register.html (frontend/register.html)

forgot-password.html (frontend/forgot-password.html)

verify-email.html (frontend/pages/verfy-email.html)

Shared CSS:

Use core styles man.css for layout and typography

components.css for form inputs, buttons, and UI elements

responsive.css to adjust form widths, font sizes, and stack elements vertically on small screens

Use consistent color scheme from themes.css (primary, secondary, error colors)

Shared JS:

auth.js (in assets/):

Form validation (email format, password strength, required fields)

Handle show/hide password toggles

Client-side input sanitation

Form submission logic (calls to backend APIs via api.js)

Handle feedback UI with toast.js for success/error notifications

Handle redirects after successful auth (e.g., login → dashboard)

forms.js: More generic form-handling utilities reusable across pages (input event handlers, error message rendering)

utils.js: Helpers as needed

2.1. login.html

Design Plan:

Centered vertical form with:

Email/username input

Password input with show/hide toggle

“Forgot password?” link below password input

Login button

Link to “Register” below form for new users

Optional social login buttons (Google, Facebook) aligned horizontally beneath form (future enhancement)

Responsive design:

Desktop: form width ~400px, with some whitespace margin

Mobile: form full width with padding

2.2. register.html

Design Plan:

Multi-step or single-page form with fields:

Username

Email

Password + confirm password

Location selection dropdown (Delta State, Edo State initially)

Optional interests (checkboxes for categories)

Progress indicator if multi-step

Submit button disabled until all validations pass

Link to “Login” for existing users

Responsive: stacked inputs on mobile, horizontal or two-column on desktop if multi-step

---

2.3. forgot-password.html

Design Plan:

Simple form asking for email input

Submit button to send reset email

Confirmation message shown after submission (“If the email exists, you will receive reset instructions”)

Link back to login page

Minimalist clean layout

2.4. verify-email.html

Design Plan:

Inform user to check their email for verification link

Button to resend verification email (disabled temporarily after send to prevent spam)

Link to login page

Clean, focused UI to avoid confusion

Folder Structure and File Usage

Page	CSS Files Used	JS Files Used

index.html	man.css, components.css, themes.css, responsive.css	home.js, toast.js, utils.js, router.js (optional)
login.html	man.css, components.css, themes.css, responsive.css	auth.js, forms.js, toast.js, api.js, utils.js
register.html	man.css, components.css, themes.css, responsive.css	auth.js, forms.js, toast.js, api.js, utils.js
forgot-password.html	man.css, components.css, themes.css, responsive.css	auth.js, forms.js, toast.js, api.js, utils.js
verify-email.html	man.css, components.css, themes.css, responsive.css	auth.js, toast.js, utils.js

---

Additional Notes

Keep man.css focused on typography, spacing, grid/flex layouts.

components.css includes form controls, buttons, cards, modals.

responsive.css handles all media queries—breakpoints for desktop (≥1024px), tablet (768-1023px), mobile (<768px).

JS modularization: auth.js only handles authentication-related logic, forms.js for generic form handlers, api.js for all backend communication (POST login, register, etc).

Use semantic HTML5 (form elements, ARIA attributes) for accessibility compliance.

Use client-side routing with router.js only if SPA is desired; otherwise, traditional multi-page approach works.

Frontend Plan for user/ Folder Pages

Typical pages inside user/ folder:

profile.html — User profile overview and edit

orders.html — User order history and details

wishlist.html — User wishlist management

settings.html (likely in root or under user) — Account and app settings management

Any additional user-specific pages you may add

---

1. profile.html

Purpose: Display and allow users to view and update their personal profile info.

Design Plan:

Layout:

Split into two main sections (responsive stack on mobile):

Left sidebar with user avatar, username, quick stats (orders, wishlist count)

Right main content with editable user details form

Profile Details Form:

Fields: Full name, username, email (read-only or editable with verification), phone number, address, profile picture upload

Edit mode toggle: read-only fields by default with an “Edit” button to enable input fields

Save and Cancel buttons

Validation on fields (email format, phone number format)

UI Components:

Avatar upload with preview and drag-drop support

Tooltips or inline help for sensitive fields (like email changes)

Confirmation modals on saving changes (success/failure notifications via toast.js)

Responsive:

Desktop: sidebar and form side-by-side

Mobile: stacked vertically, avatar on top, form below

---

2. orders.html

Purpose: Allow users to view past and current orders with details and statuses.

Design Plan:

Layout:

List or card view of user orders with pagination (pagination.js) if many

Each order card shows: order ID, date, total amount, status badge (Pending, Shipped, Delivered, Cancelled)

Order Details View:

Expand/collapse or separate page/modal to show line items, quantities, prices, delivery address, payment method, tracking info

Actions: Cancel order (if allowed), reorder, rate/review

Filters & Sorting:

Filter by date range, status

Sort by recent, amount, status

Responsive:

Cards stack on mobile with concise info

Desktop view can use multi-column table or grid layout

---

3. wishlist.html

Purpose: Users manage their saved favorite products for future purchase.

Design Plan:

Layout:

Grid or list view of wishlist products

Product card includes image, name, price, seller/shop, add-to-cart button, remove from wishlist (heart icon toggle)

Interactions:

Click product card to go to product detail page

Add to cart directly from wishlist

Remove product with confirmation

Sync wishlist with backend and keep isolated per user

Responsive:

Grid adjusts number of columns based on screen width

Mobile shows single column with larger touch targets

---

4. settings.html (if in user folder or root)

Purpose: Allow users to configure account and app preferences.

Design Plan:

Sections:

Account Management: Change password, update email, username change request

Notification Preferences: Email alerts, SMS alerts toggles

Privacy Settings: Data sharing preferences, marketing opt-ins

Payment Settings: Placeholder for future payment methods integration

Delete Account: Danger zone with confirmation dialog requiring password

UI Elements:

Tabbed or accordion layout to group settings logically

Inline form validation and feedback

Confirmation modals for sensitive actions

Responsive:

Stacked layout on mobile

Tabs visible or converted to dropdown on small screens

Common Assets and JS Usage Across user/ Pages

CSS:

man.css: Base layout and typography

components.css: Form elements, cards, buttons, badges responsive.css: Media queries for sidebar stacking, grid changes, font sizes

themes.css: Color theming, especially for status badges and alerts

JS:

profile.js: Handles profile form logic, avatar upload preview, validations

orders.js: Pagination, filters, order action handlers

wishlist.js: Manage wishlist interactions (add/remove), sync with API

settings.js: Settings form handling, modal confirmations

Shared: toast.js for notifications, api.js for backend calls, utils.js for helpers

---

UX and Functional Considerations

Security:

Input validation on frontend + backend for all forms

Confirmation dialogs for destructive actions

Session/token verification for user actions

Performance:

Lazy load images in wishlist and orders for faster page loads

Pagination or infinite scroll for orders if large data sets

Accessibility:

Keyboard navigability on forms and lists

ARIA roles on modals and interactive elements

Contrast compliance in color choices

Summary Table

Page	Key Features	CSS Files Used	JS Files Used
profile.html	Editable user info, avatar upload	man.css, components.css, responsive.css, themes.css	profile.js, auth.js, toast.js, api.js, utils.js
orders.html	Orders list, filtering, pagination	man.css, components.css, responsive.css, themes.css	orders.js, pagination.js, toast.js, api.js
wishlist.html	Product cards, add/remove wishlist	man.css, components.css, responsive.css, themes.css	wishlist.js, toast.js, api.js
settings.html	Account & notification prefs	man.css, components.css, responsive.css, themes.css	settings.js, toast.js, api.js

Frontend Plan for admin/ Folder Pages

Typical pages in admin/ folder:

dashboard.html — Admin overview and key metrics

users.html — User account management

sellers.html — Seller/submaster management

orders.html — Order monitoring and management

categories.html — Product category and subcategory management products.html — Product listing management

analytics.html — Sales and user behavior analytics

settings.html — Admin panel settings and permissions control

---

1. dashboard.html

Purpose: Central control panel giving the master/submaster admins a quick overview of the platform's status and health.

Design Plan:

Layout:

Dashboard cards/widgets showing:

Total users (active/inactive)

Total sellers/submasters

Total products

Total orders (pending, shipped, delivered, cancelled)

Revenue summary

Recent activities (user signups, product listings, orders)

Interactive charts (line, bar) for sales trends and user growth (analytics.js reusable)

Notifications area for critical alerts (system errors, pending approvals)

UI Components:

Cards with icons and concise data

Dynamic real-time data updates (polling/websocket optional)

Expandable widgets for detailed drill-down

Responsive:

Multi-column grid on desktop, stacked single column on mobile

Collapsible side navigation

---

2. users.html

Purpose: Manage all user accounts (customers).

Design Plan:

Layout:

Table or list view with pagination

Columns: Username, Email, Status (active/suspended), Date Joined, Last Login, Actions (View/Edit/Delete)

Search bar and filters (status, registration date)

Functionality:

Edit user details (status toggle, reset password)

Delete user accounts (with confirmation)

Bulk actions (activate, deactivate)

Export user data option

Responsive:

Table scroll or switch to card view on mobile

---

3. sellers.html

Purpose: Manage sellers/submaster accounts and permissions.

Design Plan:

Layout:

Similar to users.html, but with additional columns for permissions granted

Columns: Seller Name, Email, Permissions (view, edit, add, delete), Status, Actions

Functionality:

Create new seller/submaster accounts (invite or direct creation)

Assign/revoke permissions granularly

View seller activity logs

Disable/enable seller accounts

Responsive:

Same as users page

---

4. orders.html

Purpose: Full visibility and control over all orders.

Design Plan:

Layout:

Detailed table with columns: Order ID, User, Seller, Date, Status, Total, Payment Status, Actions

Filters by status, date, seller, payment status

Search functionality

Functionality:

View/edit order details

Manually update order statuses

Issue refunds or cancellations

Export order reports

Responsive:

Scrollable tables or card layout on smaller screens

---

5. categories.html

Purpose: Manage product categories and subcategories.

Design Plan:

Layout:

Tree or nested list view of categories and subcategories

Add/Edit/Delete category modal forms

Drag-and-drop to reorder categories (optional)

Functionality:

Create new categories and subcategories

Edit names, descriptions, and visibility

Assign products to categories

Bulk category operations

Responsive:

Tree collapses for mobile; simplified list view

---

6. products.html

Purpose: Manage the product inventory listed by sellers.

Design Plan:

Layout:

Table or grid showing product thumbnails, names, seller, category, price, stock status, and actions

Filters: category, seller, price range, stock status

Search bar

Functionality:

Add new product (redirect to add-product.html or modal)

Edit product details and images

Delete or deactivate products

Bulk update prices or stock status

Responsive:

Grid adjusts columns on different screens

Mobile cards with expandable details

---

7. analytics.html

Purpose: Present data insights on sales, users, and products to inform decision-making.

Design Plan:

Layout:

Multiple interactive charts (bar, pie, line, heatmaps) for:

Sales by category and region

User signups over time

Top-selling products

Revenue by seller

Filters by date range and dimensions

Functionality:

Export charts and reports

Drill-down capability

Real-time data updates optional

Responsive:

Charts resize and stack vertically on smaller screens

---

8. settings.html

Purpose: Admin panel settings including permissions and system configurations.

Design Plan:

Sections:

User roles & permissions management (master, submaster)

System-wide settings (site status, maintenance mode)

Notification preferences (email alerts for admins)

API keys and integrations management

Security settings (password policy, session timeout)

UI Elements:

Form sections with save buttons

Toggle switches for boolean settings

Confirmation dialogs for sensitive changes

Responsive:

Vertical stacking on mobile, tabs or accordion on desktop

Common Assets and JS Usage for admin/ Pages

CSS:

man.css (core styles), components.css (tables, forms, modals), responsive.css, themes.css

JS:

admin.js: Core admin page logic and event handlers

pagination.js: For users, orders, products lists

modal.js: Confirmations and forms in modals

api.js: Backend communication

toast.js: Notifications

analytics.js: Chart rendering and interaction auth.js: Permission enforcement and session handling

---

UX and Security Considerations

Security:

Strict role-based access control

Confirm destructive actions (delete user/product)

Audit logs visible for master accounts

Performance:

Efficient pagination, lazy loading for large datasets

Async data fetching with loading states

Accessibility:

Keyboard navigable tables and forms

ARIA roles for modals and alerts

Color contrast compliance

Summary Table

Page	Key Features	CSS Files Used	JS Files Used
dashboard.html	Overview metrics, recent activity, charts	man.css, components.css, responsive.css, themes.css	admin.js, analytics.js, toast.js, api.js
users.html	User management, search/filter, bulk actions	man.css, components.css, responsive.css	admin.js, pagination.js, modal.js, api.js
sellers.html	Seller accounts, permissions, activity logs	man.css, components.css, responsive.css	admin.js, modal.js, api.js

orders.html	Order tracking, update statuses, export	man.css, components.css, responsive.css	admin.js, pagination.js, modal.js, api.js
categories.html	Category tree, add/edit/delete, reorder	man.css, components.css, responsive.css	admin.js, modal.js, api.js
products.html	Product listings, bulk updates, filters	man.css, components.css, responsive.css	admin.js, pagination.js, modal.js, api.js
analytics.html	Interactive charts and reports	man.css, components.css, responsive.css	analytics.js, admin.js, toast.js
settings.html	Role & permission management, system config	man.css, components.css, responsive.css	admin.js, modal.js, api.js

Frontend Plan for public/ Folder Pages

---

1. about.html

Purpose: Inform visitors about the company, vision, mission, and values.

Design Plan:

Layout:

Header with logo and simple nav (Home, Marketplace, Contact)

Hero section with compelling image/banner and a headline

Sections for:

Company overview (who we are)

Mission & Vision statements

Team or leadership profiles (optional)

Timeline or milestones (growth history)

Testimonials or user reviews (optional)

Footer with contact info, social media links

UI Elements:

Clean typography, consistent with brand colors (themes.css)

Responsive images and text blocks

Possibly subtle animations or parallax for engagement

Responsiveness:

Multi-column layout on desktop, stacked vertically on mobile/tablet

---

2. marketplace.html

Purpose: Public-facing product catalog browsing page.

Design Plan:

Layout:

Top search bar with filters (categories, price range, brands/shops)

Sidebar for categories and subcategories navigation

Main content area with grid of products: thumbnails, names, prices, seller info, quick add to wishlist/cart

Pagination or infinite scroll for product listings

Sorting options (popularity, price, newest)

UI Elements:

Product cards with hover effects showing quick actions

Filters collapsible on mobile (slide-in panel or accordion)

Sticky sidebar or dropdown on smaller screens

Responsiveness:

Grid adjusts product columns based on screen size

Sidebar collapses into a hamburger menu on phones

---

3. shop-profile.html

Purpose: Public profile page for a shop or brand showcasing their products and info.

Design Plan:

Layout:

Header with shop logo and banner image

Shop details section: name, description, location (Delta, Edo), ratings/reviews

Navigation tabs or anchors: Products, About Shop, Reviews, Contact

Product listing grid under Products tab

Follow/Subscribe button to shop (adds to user’s wishlist/follow list)

UI Elements:

Ratings display (stars)

Social media or external shop links

Contact button or form to message the shop

Responsive carousel or grid for product display

Responsiveness:

Tabs stack vertically on mobile

Images and grids scale fluidly

---

4. product-detail.html

Purpose: Detailed product information page for a single item.

Design Plan:

Layout:

Large product image carousel/gallery

Product title, price, available stock

Seller/shop info with link to shop profile

Add to cart and wishlist buttons

Product description, specifications, and features

Reviews section with user ratings and comments

Related products or upsell suggestions at the bottom

UI Elements:

Interactive image zoom or lightbox

Clear call-to-action buttons

Tabs or accordion for product details, specs, and reviews

Review submission form for logged-in users

Responsiveness:

Images stack on top of details on phones

Carousel switches to swipe gestures on touch devices

---

5. contact.html

Purpose: Contact page for general inquiries and support.

Design Plan:

Layout:

Contact form with fields: name, email, subject, message

Company contact info: email, phone, address

Embedded map showing office locations in Delta and Edo States

FAQs or quick help links (optional)

UI Elements:

Form validation and feedback messages

Clear call to action to submit inquiries

Responsive map integration (Google Maps or similar)

Social media links

Responsiveness:

Form and contact info side-by-side on desktop

Stacked vertically on mobile

Common Assets and JS Usage for public/ Pages

CSS:

man.css for core styling, components.css for UI elements, themes.css for branding colors

responsive.css for media queries and device-specific styles

JS:

home.js or marketplace.js for product browsing, filtering, and search functionality

api.js for backend data fetching (products, shops, etc.)

toast.js for notifications (e.g., add to wishlist success)

modal.js for popups or confirmations

forms.js for contact form validation and submission

seller/dashboard.html

Design Plan:

Layout: Clean dashboard with sidebar navigation and main content area. Sidebar includes links: Overview, Products, Orders, Analytics, Shop Profile, Add Product.

Main content: Summary cards showing sales, orders, products count, pending orders, revenue graph.

Responsive: Sidebar collapses to a hamburger menu on phones, top nav with dropdowns.

Visuals: Use your theme.css for consistent branding, charts/graphs with lightweight JS libraries for analytics preview.

Functionality: Dashboard data fetched from backend via API (api.js), state managed by state.js. Toast notifications on updates or errors (toast.js).

---

seller/add-product.html

Design Plan:

Form-heavy page: Multi-section form collecting product details (name, description, price, category, images, stock, variants).

UX: Inline validation (forms.js), progress indicators for image uploads, real-time preview of product card as data is filled.

Responsive: Form fields stacked vertically on phones, multi-column on tablets/laptops.

UI components: Use reusable input components from components.css for consistency.

JS: Handle image uploads, dynamic attribute fields (color, size), form submission with feedback via toast.js and modal.js confirmation dialogs.

---

seller/shop-profile.html

Design Plan:

Profile view and edit mode: Display shop details, branding (logo, banner), description, contact info. Edit mode toggles input fields for updates.

Layout: Two-column design on desktop — left with branding/media, right with text/details. On mobile, stacked layout with collapsible sections.

Visuals: Prominent call-to-action buttons (Edit, Save, Cancel), consistent color scheme from themes.css.

Functionality: Fetch and update shop info via API calls, validations with forms.js.

Extras: Option to upload branding assets with preview.

---

seller/orders.html

Design Plan:

Orders management table: Paginated, sortable list of orders related to the seller. Columns include order number, date, customer, status, total, actions (view, update).

Filtering: Filters for status (pending, shipped, delivered), date range.

Responsive: Table scroll or convert to card layout on mobile with essential info.

JS: Use pagination.js for order pages, modal.js for order detail popup, toast.js for status update feedback.

Security: Ensure seller can only view orders linked to their shop.

---

seller/products.html

Design Plan:

Product list: Grid or table view of seller's products with key info: image, name, price, stock, status (active/inactive).

Actions: Edit, deactivate, delete buttons for each product with confirmation modals.

Bulk actions: Select multiple products for bulk delete or status change.

Responsive: Switch between grid on desktop and vertical card list on mobile.

JS: Interactions via marketplace.js or seller.js, confirmation with modal.js.

---

seller/analytics.html

Design Plan:

Visual analytics dashboard: Sales charts, product performance, traffic sources, conversion rates.

Layout: Use cards with graphs (line, bar, pie charts) arranged in grid.

Responsiveness: Cards stack vertically on mobile, grid on desktop.

JS: Use lightweight chart libraries with data from backend APIs, refresh on date range filter.

Extras: Export data options (CSV, PDF) triggered from buttons.

General Notes for Seller Pages:

CSS: Use core styles (man.css) + component.css for buttons, forms, tables, modals. Use responsive.css for media queries.

JS: Use modular files (seller.js for seller-specific logic, forms.js for validation, api.js for backend calls, toast.js and modal.js for UI feedback).

Routing: Client-side routing handled via router.js for SPA-like transitions where possible.

Accessibility: Ensure keyboard navigability and proper ARIA labels.

Security: Role-based UI elements visibility controlled via auth.js and backend permissions.