TISCO Market - Product Requirements Document (PRD)

1. Platform Overview

Vision Statement

TISCO Market is a comprehensive online platform combining e-commerce functionality with service booking capabilities, providing users with a seamless experience to both purchase products and book services from a single, integrated platform.

Platform Goals

- **Primary Goal**: Create a unified marketplace for products and services
- **Secondary Goals**:
  - Streamline customer acquisition for service providers
  - Provide intuitive product discovery and purchasing
  - Enable efficient service booking and management
  - Build a scalable platform for future expansion

2. Success Metrics (KPIs)

Business Metrics

- Monthly Active Users (MAU)
- Gross Merchandise Volume (GMV)
- Service Booking Conversion Rate
- Customer Acquisition Cost (CAC)
- Customer Lifetime Value (CLV)
- Average Order Value (AOV)

Technical Metrics

- Page Load Time (< 2 seconds)
- Platform Uptime (99.9% target)
- Search Response Time (< 500ms)
- Mobile Responsiveness Score

User Experience Metrics

- Task Completion Rate
- User Retention Rate (Day 7, Day 30)

- Support Ticket Volume
- Net Promoter Score (NPS)

3. User Personas

Primary Personas

1. Product Shopper (Sarah)

- **Age**: 25-45
- **Tech Comfort**: Medium to High
- **Goals**: Find quality products, compare prices, quick checkout
- **Pain Points**: Complex navigation, slow loading, unclear product info
- **Behavior**: Mobile-first, price-conscious, reviews-driven

2. Service Seeker (Michael)

- **Age**: 30-55
- **Tech Comfort**: Medium
- **Goals**: Book reliable services, clear pricing, easy scheduling
- **Pain Points**: Unclear service descriptions, complex booking process
- **Behavior**: Research-heavy, availability-focused, quality-oriented

3. Hybrid User (Jessica)

- **Age**: 28-50
- **Tech Comfort**: High
- **Goals**: One-stop shop for products and services
- **Pain Points**: Context switching between platforms
- **Behavior**: Efficiency-focused, loyalty-driven, multi-device usage

4. Prioritized User Stories

Epic 1: Core Navigation & Discovery

- **US-001**: As a user, I want to easily navigate between Shop and Services sections
- **US-002**: As a user, I want to search across products and services with filters
- **US-003**: As a user, I want to see featured/promoted items on the homepage

Epic 2: Product Management

- **US-004**: As a shopper, I want to browse products by categories

- **US-005**: As a shopper, I want to view detailed product information and images
- **US-006**: As a shopper, I want to add products to cart and manage quantities
- **US-007**: As a shopper, I want to complete secure checkout with multiple payment options

Epic 3: Service Booking

- **US-008**: As a service seeker, I want to browse available services with clear descriptions
- **US-009**: As a service seeker, I want to select service type, date, and time preferences
- **US-010**: As a service seeker, I want to provide service-specific details in booking form
- **US-011**: As a service seeker, I want to receive booking confirmation and reminders

Epic 4: User Account Management

- **US-012**: As a user, I want to register/login using email or social accounts
- **US-013**: As a user, I want to view my order history and service bookings
- **US-014**: As a user, I want to manage my profile and preferences

5. MVP Feature List

5.1 Core Features (Must-Have)

Navigation & Layout

- ✅ Responsive navbar with logo, search, cart, and user icons
- ✅ Main navigation: Home, Services, Shop, About, Contact
- ✅ Footer with social media links and company information
- ✅ Mobile-responsive design

Homepage (Section 01)

- ✅ Hero carousel with promotional content and aesthetic images
- ✅ "What's New" section highlighting latest products/services
- ✅ Categories banner with circular tags for easy navigation
- ✅ Featured products grid (3 columns, 2 rows)
- ✅ Service booking CTA banner with promotion and navigation to service page

Shop Page (Section 02)

- ✅ Hero section with promotional imagery and background
- ✅ Filter and search functionality (top section)
- ✅ Left sidebar filters for advanced product filtering
- ✅ Product grid display (3 columns, 5 rows per page)

- ✅ Pagination for product browsing

Services Section

- ✅ Hero with headline about services and supporting text
- ✅ Service booking CTA and "How it Works" explanation
- ✅ Service cards offering (PC Building, Desktop/Office Setup, Computer Repair/Software Installation)
- ✅ General support form for non-specific service requests
- ✅ Service booking form with: service type selection, description field, preferred date/time, email, phone number (auto-fill for registered users)

User Authentication

- ✅ Registration and login pages using Clerk integration
- ✅ Email/password authentication
- ✅ Social login options (Google, GitHub, etc.)
- ✅ User profile management

Product Management

- ✅ Individual product pages with detailed information
- ✅ Shopping cart functionality
- ✅ Basic checkout process

5.2 Enhanced Features (Should-Have)

Content Management

- ✅ AI-generated content for About page
- ✅ AI-generated content for Contact page
- ✅ Dynamic content management system integration

Advanced User Features

- ✅ Order history and tracking
- ✅ Service booking history and status
- ✅ User preference management

Admin Features

- ✅ Basic admin dashboard for content management
- ✅ Order and booking management interface

5.3 Future Roadmap (Could-Have)

Phase 2 Features

- Advanced analytics and reporting
- Multi-vendor marketplace capabilities
- Advanced service provider management
- Customer review and rating system
- Real-time chat support
- Push notifications for bookings/orders

Phase 3 Features

- Mobile app development
- Advanced AI recommendations
- Subscription services
- Multi-language support
- Advanced inventory management

6. Technical Architecture

6.1 Technology Stack

Frontend

- **Framework**: Next.js 14+ (App Router)
- **Styling**: Tailwind CSS
- **UI Components**: shadcn/ui or Headless UI
- **Icons**: Lucide React or Heroicons
- **State Management**: React Context + useReducer (for complex state)

Backend & Services

- **Database**: Supabase (PostgreSQL)
- **Authentication**: Clerk
- **File Storage**: Supabase Storage
- **Email**: Resend or SendGrid
- **Payment Processing**: Stripe (future implementation)

Deployment & DevOps

- **Hosting**: Vercel
- **Version Control**: GitHub
- **CI/CD**: GitHub Actions + Vercel integration
- **Monitoring**: Vercel Analytics + Sentry (error tracking)

6.2 Proposed Database Schema

```
sql
```

```
-- Users (managed by Clerk, reference only)
-- Clerk handles user authentication and basic profile data

-- Categories
CREATE TABLE categories (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name VARCHAR NOT NULL,
  description TEXT,
  image_url TEXT,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Products
CREATE TABLE products (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name VARCHAR NOT NULL,
  description TEXT,
  price DECIMAL(10,2) NOT NULL,
  category_id UUID REFERENCES categories(id),
  image_urls TEXT[],
  stock_quantity INTEGER DEFAULT 0,
  is_featured BOOLEAN DEFAULT FALSE,
  status VARCHAR DEFAULT 'active', -- active, inactive, out_of_stock
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Service Types
CREATE TABLE service_types (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  name VARCHAR NOT NULL,
  description TEXT,
  base_price DECIMAL(10,2),
  estimated_duration_hours INTEGER,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Service Bookings
CREATE TABLE service_bookings (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id VARCHAR NOT NULL, -- Clerk user ID
  service_type_id UUID REFERENCES service_types(id),
  description TEXT,
  preferred_date DATE,
  preferred_time TIME,
  status VARCHAR DEFAULT 'pending', -- pending, confirmed, in_progress, completed, cancelled
  email VARCHAR NOT NULL,
  phone VARCHAR,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Orders
CREATE TABLE orders (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id VARCHAR NOT NULL, -- Clerk user ID
  total_amount DECIMAL(10,2) NOT NULL,
  status VARCHAR DEFAULT 'pending', -- pending, confirmed, shipped, delivered, cancelled
  shipping_address JSONB,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Order Items
CREATE TABLE order_items (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  order_id UUID REFERENCES orders(id) ON DELETE CASCADE,
  product_id UUID REFERENCES products(id),
  quantity INTEGER NOT NULL,
  price_at_time DECIMAL(10,2) NOT NULL,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW()
);

-- Shopping Cart (for persistent cart across sessions)
CREATE TABLE cart_items (
  id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
  user_id VARCHAR NOT NULL, -- Clerk user ID
  product_id UUID REFERENCES products(id),
  quantity INTEGER NOT NULL,
  created_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  updated_at TIMESTAMP WITH TIME ZONE DEFAULT NOW(),
  UNIQUE(user_id, product_id)
);

-- Create indexes for performance
CREATE INDEX idx_products_category ON products(category_id);
CREATE INDEX idx_products_status ON products(status);
CREATE INDEX idx_service_bookings_user ON service_bookings(user_id);
CREATE INDEX idx_orders_user ON orders(user_id);
CREATE INDEX idx_cart_items_user ON cart_items(user_id);
```

6.3 Authentication Flow (Clerk Integration)

Registration/Login Flow

1. User visits `/sign-up` or `/sign-in` pages
2. Clerk handles authentication UI and process
3. Upon successful authentication, redirect to intended page or dashboard
4. User profile data synced with Clerk (name, email, profile image)
5. Shopping cart merged from session storage to database (if applicable)

Protected Routes

- User dashboard: `/dashboard`
- Order history: `/orders`
- Service bookings: `/bookings`
- Admin routes: `/admin/*`

Middleware Configuration

```javascript
// middleware.js
import { authMiddleware } from "@clerk/nextjs";

export default authMiddleware({
  publicRoutes: ["/", "/shop", "/services", "/about", "/contact", "/product/:id"],
  ignoredRoutes: ["/api/webhooks/(.*)"]
});
```

7. Non-Functional Requirements

7.1 Performance Requirements

- **Page Load Time**: < 2 seconds for initial page load
- **Time to Interactive**: < 3 seconds
- **Search Response Time**: < 500ms
- **API Response Time**: < 200ms for most endpoints
- **Image Optimization**: WebP format, responsive images, lazy loading

7.2 Scalability Requirements

- **Concurrent Users**: Support 1,000+ concurrent users
- **Database**: Handle 100,000+ products and 10,000+ users initially

- **CDN**: Global content delivery for static assets
- **Caching**: Implement Redis for session and frequent data caching

7.3 Security Requirements

- **Authentication**: Secure JWT tokens via Clerk
- **Data Encryption**: HTTPS everywhere, encrypted sensitive data
- **Input Validation**: Server-side validation for all inputs
- **SQL Injection Prevention**: Parameterized queries via Supabase
- **Rate Limiting**: API rate limiting to prevent abuse
- **CORS**: Proper CORS configuration

7.4 Accessibility Requirements

- **WCAG 2.1 AA Compliance**: Meet accessibility standards
- **Keyboard Navigation**: Full keyboard accessibility
- **Screen Reader Support**: Proper ARIA labels and semantic HTML
- **Color Contrast**: Minimum 4.5:1 contrast ratio
- **Mobile Accessibility**: Touch-friendly interfaces

8. Deployment Workflow

8.1 Environment Setup

Development Environment

bash

```bash
# Local development setup
1. Clone repository
2. Install dependencies: npm install
3. Set up environment variables (.env.local)
4. Run development server: npm run dev
5. Connect to Supabase local instance or staging
```

Environment Variables

bash

```bash
```

```
# .env.local (development)
NEXT_PUBLIC_SUPABASE_URL=your_supabase_url
NEXT_PUBLIC_SUPABASE_ANON_KEY=your_supabase_anon_key
SUPABASE_SERVICE_ROLE_KEY=your_service_role_key
NEXT_PUBLIC_CLERK_PUBLISHABLE_KEY=your_clerk_publishable_key
CLERK_SECRET_KEY=your_clerk_secret_key
```

8.2 GitHub Integration

- **Repository Structure**: Organized folders for components, pages, utils, etc.
- **Branch Strategy**:
  - `main` → Production deployment
  - `staging` → Staging environment
  - `develop` → Development integration
  - Feature branches for development

8.3 Vercel Deployment Configuration

Staging Environment

- **Branch**: `staging`
- **Domain**: `staging-tisco-market.vercel.app`
- **Environment**: Staging Supabase instance
- **Purpose**: Testing before production deployment

Production Environment

- **Branch**: `main`
- **Domain**: Custom domain (e.g., `tiscomarket.com`)
- **Environment**: Production Supabase instance
- **Purpose**: Live user-facing application

Vercel Configuration (`vercel.json`)

```
json
```

```
{
  "framework": "nextjs",
  "buildCommand": "npm run build",
  "devCommand": "npm run dev",
  "installCommand": "npm install",
  "functions": {
    "app/api/**/*.js": {
      "maxDuration": 30
    }
  },
  "redirects": [
    {
      "source": "/admin",
      "destination": "/admin/dashboard",
      "permanent": false
    }
  ]
}
```

9. Development Phases

Phase 1: Foundation (Weeks 1-2)

- □ Project setup and configuration
- □ Database schema implementation
- □ Clerk authentication integration
- □ Basic layout and navigation
- □ Homepage hero and featured sections

Phase 2: Core Features (Weeks 3-4)

- □ Product catalog and individual product pages
- □ Shopping cart functionality
- □ Service booking form and management
- □ User dashboard and profile management

Phase 3: Polish & Launch (Weeks 5-6)

- □ Checkout process and order management
- □ Admin dashboard basics
- □ Performance optimization
- □ Testing and bug fixes
- □ Production deployment

10. Missing Details & Action Items

🚨 Critical Gaps to Address

Business Requirements

- □ **Payment Processing**: Define payment methods, currency, tax handling
- □ **Shipping/Fulfillment**: Physical product shipping strategy and costs
- □ **Service Provider Network**: How services are fulfilled (in-house vs partners)
- □ **Pricing Strategy**: Product margins, service pricing models
- □ **Legal Requirements**: Terms of service, privacy policy, refund policy

Content & Data

- □ **Product Catalog**: Source of initial products, product data structure
- □ **Service Details**: Detailed service descriptions, pricing, availability
- □ **Brand Assets**: Logo variations, brand colors, fonts, imagery style guide
- □ **Content Strategy**: About page content, company story, team information

Technical Specifications

- □ **Third-party Integrations**: Email service provider, analytics tools
- □ **SEO Requirements**: Meta tags, structured data, sitemap strategy
- □ **Mobile App**: Future mobile app considerations
- □ **API Design**: RESTful API structure for future integrations

Operational Considerations

- □ **Customer Support**: Support channel strategy (email, chat, phone)
- □ **Order Fulfillment Process**: Inventory management, order processing workflow
- □ **Service Booking Management**: Staff scheduling, booking confirmation process
- □ **Quality Assurance**: Testing strategy, user acceptance criteria

📋 Next Steps

1. **Review and Approve PRD**: Stakeholder review of this document
2. **Define Missing Business Logic**: Address critical gaps listed above
3. **Create Design System**: Establish UI components based on reference images
4. **Set Up Development Environment**: Repository, database, authentication
5. **Create Project Timeline**: Detailed development schedule with milestones
6. **Begin Phase 1 Development**: Start with foundation and core architecture

🎯 Success Criteria for MVP Launch

- □ Users can browse products and services seamlessly
- □ Shopping cart and checkout process works reliably

- □ Service booking form captures all necessary information
- □ User authentication and profile management functions properly
- □ Admin can manage basic content and view bookings/orders
- □ Platform performs well on mobile and desktop
- □ All forms validate properly and provide good UX
- □ Error handling provides meaningful feedback to users

Document Version: 1.0

Last Updated: [Current Date]

Next Review: [Date + 2 weeks]