SSS BAGS

Full Stack E-Commerce Platform

Project Report for College Showcase

Production-ready • Rails 7.2 API • React 18 • Purple Theme

---

1. Project Overview

SSS BAGS is a production-ready, full-stack e-commerce platform for selling bags, handbags, backpacks, wallets, and luggage. It features a modern purple-themed UI, complete authentication with email and phone verification, role-based access control (Admin & Customer), shopping cart, order management, and an admin dashboard.

> **Key Highlights:** Full-stack architecture (Rails API + React SPA) • Production deployment on Render + Vercel • JWT authentication with OTP verification • Complete e-commerce flow (Browse → Cart → Checkout → Order tracking) • Admin dashboard for products, orders, users, banners

2. Technology Stack

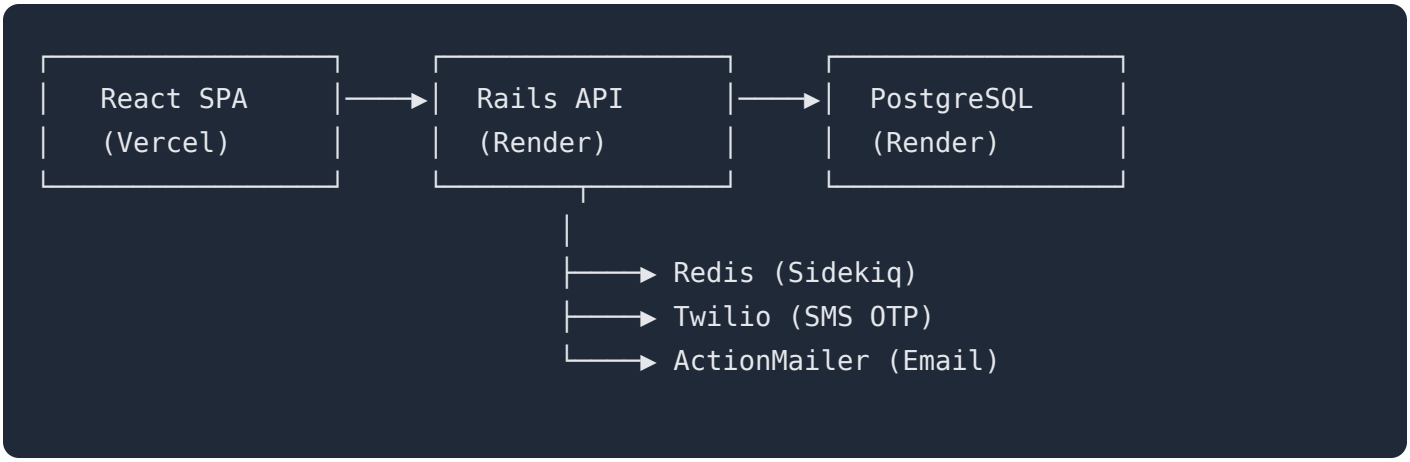
Backend

Technology	Purpose
Ruby 3.3+	Programming language
Rails 7.2	API framework
PostgreSQL	Primary database
Redis	Caching, Sidekiq background jobs
JWT	Access + refresh token authentication
bcrypt	Password hashing
Pundit	Role-based authorization
ActiveStorage	Image uploads
Sidekiq	Background job processing
Twilio	SMS OTP for phone verification
ActionMailer	Email verification
RSwag	API documentation (Swagger)

Frontend

Technology	Purpose
React 18	UI framework
Vite 5	Build tool, dev server
React Router 6	Client-side routing
Redux Toolkit	State management
Axios	HTTP client
Tailwind CSS	Styling (purple theme)
Framer Motion	Animations

3. System Architecture



All API routes are versioned under `/api/v1/`.

4. Database Schema & Models

Table	Description
users	Customers and admins (name, email, phone, role, email_verified, phone_verified)
categories	Product categories (Handbags, Backpacks, Wallets, Luggage)
products	Items (name, price, discount_price, stock, slug, status, featured)
carts / cart_items	Shopping cart
orders / order_items	Placed orders with line items
addresses	User shipping addresses
payments	Payment records linked to orders
banners	Homepage promo banners
announcements	Top strip announcements
otp_rate_limits	Rate limiting for OTP sends

5. Authentication & Authorization

Authentication Flow

1. **Signup** → Account created → Email verification sent (background job)
2. **Email verification** → User clicks link → email_verified
3. **Phone verification** → OTP via Twilio SMS → User enters code → phone_verified
4. **Login** → JWT access + refresh tokens
5. **Token refresh** → Frontend uses refresh token when access expires

Roles

customer Profile, Cart, Addresses, Orders, Product browse

admin Dashboard, Products CRUD, Orders, Users, Banners, Announcements

Security

bcrypt passwords • JWT access (short-lived) + refresh tokens • OTP rate limiting • Pundit authorization • CORS • Strong params

6. API Endpoints

Public

POST /auth/signup, /auth/login, /auth/refresh, /auth/verify_email • GET /products, /products/:slug, /products/search • GET /categories, /banners, /announcements

Customer (Auth)

GET/PATCH /customer/profile • CRUD /customer/cart, /customer/addresses • GET/POST /customer/orders • GET /customer/orders/:id/track • POST /auth/send_otp, /auth/verify_otp

Admin (Auth)

GET /admin/dashboard • CRUD /admin/products, /admin/orders, /admin/users, /admin/categories, /admin/banners, /admin/announcements • GET /admin/payments • POST /admin/uploads

Swagger docs: `/api-docs`

7. Frontend Pages

Route	Page
/	Landing (Hero, featured products, categories)
/products	Product listing with filters
/products/:slug	Product detail, add to cart
/login, /signup	Auth
/verify-email, /verify-otp	Verification
/profile, /cart, /orders	Customer area
/admin, /admin/products, /admin/orders	Admin dashboard

8. Order Flow

1. Validate cart, check stock
2. Calculate total (effective_price: discount or regular)
3. Create Order with shipping_address
4. Create OrderItems, decrement stock
5. Clear cart, create Payment record

Order statuses: pending → confirmed → shipped → delivered | cancelled

9. Services & Background Jobs

JwtService – encode/decode access & refresh tokens

OtpService – Send OTP via Twilio, verify (6-digit, 10 min expiry)

EmailVerificationService – Validate email token

OrderCreationService – Create order from cart

EmailVerificationJob – Sidekiq job for verification emails

10. Deployment

Live URLs:

- Frontend: https://sss-bags-nine.vercel.app/products
- Backend: https://sss-bags.onrender.com
- API Docs: https://sss-bags.onrender.com/api-docs

Default admin: admin@sssbags.com / admin123

Hosting: Render (backend + PostgreSQL) • Vercel (frontend) • GitHub (source)

11. Local Development

```
docker-compose up -d db redis
cd backend && bundle install && bin/rails db:create db:migrate db:seed
bundle exec puma -C config/puma.rb
cd frontend && npm install && npm run dev
```

Backend: http://localhost:3000 • Frontend: http://localhost:5173

12. Summary for College Presentation

SSS BAGS demonstrates:

- Full-stack development (Rails API + React SPA)
- Authentication & security (JWT, OTP, email verification, role-based access)
- Complete e-commerce flow (products, cart, checkout, orders, tracking)
- Admin capabilities (dashboard, CRUD for products, orders, users, content)
- Production deployment (Render + Vercel, live URLs)
- Modern tooling (Docker, Vite, Tailwind, Redux)
- RESTful API design (versioned, documented via Swagger)