🚀 Roster Management SaaS - Complete MVP Development Plan

📋 Project Overview

Project Name: RosterPro - Employee Roster Management System
Type: SaaS (Software as a Service)
Target: Small to medium businesses managing shift-based employees
Timeline: 8-12 weeks for MVP
Team Size: 1-3 developers

🛠️ Tech Stack

Frontend

- **Framework:** Next.js 14 (App Router)
- **Language:** TypeScript
- **UI Library:** React 18
- **Styling:** Tailwind CSS
- **Component Library:** shadcn/ui + Radix UI
- **Animations:** Framer Motion
- **State Management:** Zustand
- **Forms:** React Hook Form + Zod
- **HTTP Client:** Axios
- **Real-time:** Socket.IO Client
- **Date Handling:** date-fns
- **Payments:** Stripe.js (@stripe/stripe-js)

Backend

- **Runtime:** Node.js 20+
- **Framework:** Express.js
- **Language:** TypeScript
- **Database:** PostgreSQL 15
- **ORM:** Prisma
- **Authentication:** JWT (jsonwebtoken) + bcrypt
- **Cache/Session:** Redis 7
- **Real-time:** Socket.IO
- **Payments:** Stripe API
- **File Upload:** Multer (if needed)
- **Email:** Nodemailer or SendGrid
- **Validation:** Zod

DevOps & Deployment

- **Containerization:** Docker + Docker Compose
- **CI/CD:** GitHub Actions
- **Hosting (Backend):** Railway / Render / AWS EC2

- **Hosting (Frontend):** Vercel / Netlify
- **Database:** Railway PostgreSQL / Supabase / AWS RDS
- **Redis:** Upstash / Redis Cloud
- **Domain:** Namecheap / Google Domains
- **SSL:** Let's Encrypt (Auto via hosting)
- **Monitoring:** Sentry (Errors) + Uptime Robot

Development Tools

- **Version Control:** Git + GitHub
- **Code Editor:** VS Code
- **API Testing:** Postman / Insomnia
- **Database GUI:** Prisma Studio / pgAdmin
- **Design:** Figma (optional for mockups)

📁 Project Structure

```
roster-management-saas/
├── backend/
│   ├── src/
│   │   ├── config/
│   │   │   ├── database.ts
│   │   │   ├── redis.ts
│   │   │   └── stripe.ts
│   │   ├── middleware/
│   │   │   ├── auth.ts
│   │   │   ├── error.ts
│   │   │   ├── rateLimiter.ts
│   │   │   └── validation.ts
│   │   ├── routes/
│   │   │   ├── auth.routes.ts
│   │   │   ├── roster.routes.ts
│   │   │   ├── shift.routes.ts
│   │   │   ├── user.routes.ts
│   │   │   ├── payment.routes.ts
│   │   │   └── admin.routes.ts
│   │   ├── controllers/
│   │   │   ├── auth.controller.ts
│   │   │   ├── roster.controller.ts
│   │   │   ├── shift.controller.ts
│   │   │   ├── user.controller.ts
│   │   │   ├── payment.controller.ts
│   │   │   └── admin.controller.ts
│   │   ├── services/
│   │   │   ├── jwt.service.ts
│   │   │   ├── email.service.ts
│   │   │   ├── stripe.service.ts
│   │   │   └── notification.service.ts
│   │   ├── socket/
│   │   │   └── chat.socket.ts
│   │   ├── types/
│   │   │   └── index.ts
│   │   ├── utils/
│   │   │   └── helpers.ts
│   │   └── server.ts
│   ├── prisma/
│   │   ├── schema.prisma
│   │   └── seed.ts
```

```
│   ├── .env.example
│   ├── .gitignore
│   ├── Dockerfile
│   ├── package.json
│   └── tsconfig.json
├── frontend/
│   ├── app/
│   │   ├── (auth)/
│   │   │   ├── login/
│   │   │   │   └── page.tsx
│   │   │   └── register/
│   │   │       └── page.tsx
│   │   ├── (dashboard)/
│   │   │   ├── layout.tsx
│   │   │   ├── page.tsx (Overview)
│   │   │   ├── roster/
│   │   │   │   └── page.tsx
│   │   │   ├── team/
│   │   │   │   └── page.tsx
│   │   │   ├── chat/
│   │   │   │   └── page.tsx
│   │   │   ├── payments/
│   │   │   │   └── page.tsx
│   │   │   └── settings/
│   │   │       └── page.tsx
│   │   ├── admin/
│   │   │   ├── layout.tsx
│   │   │   ├── dashboard/
│   │   │   ├── users/
│   │   │   └── analytics/
│   │   ├── layout.tsx
│   │   └── globals.css
│   ├── components/
│   │   ├── ui/ (shadcn components)
│   │   ├── roster/
│   │   │   ├── Calendar.tsx
│   │   │   ├── ShiftCard.tsx
│   │   │   └── CreateRosterModal.tsx
│   │   ├── chat/
│   │   │   ├── ChatList.tsx
│   │   │   ├── MessageBubble.tsx
```

```
│   │   │   └── ChatInput.tsx
│   │   ├── team/
│   │   │   ├── MemberCard.tsx
│   │   │   └── AddMemberModal.tsx
│   │   ├── layout/
│   │   │   ├── Sidebar.tsx
│   │   │   └── Header.tsx
│   │   └── shared/
│   │       ├── StatCard.tsx
│   │       └── Modal.tsx
│   ├── lib/
│   │   ├── api.ts
│   │   ├── socket.ts
│   │   └── utils.ts
│   ├── hooks/
│   │   ├── useAuth.ts
│   │   ├── useSocket.ts
│   │   └── useRoster.ts
│   ├── store/
│   │   ├── authStore.ts
│   │   └── chatStore.ts
│   ├── types/
│   │   └── index.ts
│   ├── .env.local.example
│   ├── .gitignore
│   ├── Dockerfile
│   ├── next.config.js
│   ├── tailwind.config.ts
│   ├── package.json
│   └── tsconfig.json
├── docker-compose.yml
└── README.md
```

📅 Development Timeline (8-12 Weeks)

Week 1-2: Project Setup & Backend Foundation

Day 1-2: Project Initialization

bash

```
# Create project structure
mkdir roster-management-saas
cd roster-management-saas

# Backend setup
mkdir backend && cd backend
npm init -y
npm install express typescript @types/express @types/node
npm install prisma @prisma/client
npm install jsonwebtoken bcrypt @types/jsonwebtoken @types/bcrypt
npm install redis socket.io stripe
npm install cors helmet dotenv
npm install -D tsx nodemon ts-node

# Initialize TypeScript
npx tsc --init

# Initialize Prisma
npx prisma init

# Frontend setup
cd ..
npx create-next-app@latest frontend --typescript --tailwind --app
cd frontend
npm install axios socket.io-client
npm install framer-motion zustand
npm install @stripe/stripe-js
npm install date-fns
npm install lucide-react
npx shadcn-ui@latest init
```

Day 3-5: Database Schema & Setup

- Design database schema in Prisma
- Create migrations
- Set up PostgreSQL (local/Docker)
- Set up Redis (local/Docker)
- Create seed data for testing

prisma

```
// Example: prisma/schema.prisma
// Copy the schema from the provided code
```

bash

```
# Run migrations
npx prisma migrate dev --name init
npx prisma generate

# Seed database
npx prisma db seed
```

Day 6-10: Authentication System

- [ ] JWT service implementation
- [ ] User registration endpoint
- [ ] User login endpoint
- [ ] Refresh token mechanism
- [ ] Password hashing
- [ ] Auth middleware
- [ ] Role-based access control
- [ ] Redis session management
- [ ] Logout functionality

Files to create:

- `src/services/jwt.service.ts`
- `src/controllers/auth.controller.ts`
- `src/routes/auth.routes.ts`
- `src/middleware/auth.ts`

Day 11-14: Docker Setup

yaml

```yaml
# docker-compose.yml
version: '3.8'
services:
  postgres:
    image: postgres:15-alpine
    environment:
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
      POSTGRES_DB: roster_db
    ports:
      - "5432:5432"
    volumes:
      - postgres_data:/var/lib/postgresql/data

  redis:
    image: redis:7-alpine
    ports:
      - "6379:6379"
    volumes:
      - redis_data:/data

volumes:
  postgres_data:
  redis_data:
```

bash

```
# Start services
docker-compose up -d
```

Week 3-4: Core Backend Features

Day 15-18: Roster Management

- [ ] Create roster endpoint
- [ ] Get rosters endpoint (with filters)
- [ ] Update roster endpoint
- [ ] Delete roster endpoint
- [ ] Publish roster functionality
- [ ] Roster validation

API Endpoints:

```
POST   /api/roster
GET    /api/roster
GET    /api/roster/:id
PATCH  /api/roster/:id
DELETE /api/roster/:id
POST   /api/roster/:id/publish
```

Day 19-22: Shift Management

- [ ] Create shift endpoint
- [ ] Get shifts endpoint
- [ ] Update shift endpoint
- [ ] Delete shift endpoint
- [ ] Assign user to shift
- [ ] Shift conflict detection
- [ ] Shift swap functionality

API Endpoints:

```
POST   /api/shift
GET    /api/shift
PATCH  /api/shift/:id
DELETE /api/shift/:id
POST   /api/shift/:id/assign
POST   /api/shift/:id/swap
```

Day 23-26: User Management

- [ ] Get all users endpoint
- [ ] Get user by ID endpoint
- [ ] Update user endpoint
- [ ] Delete user endpoint
- [ ] User role management
- [ ] User profile updates

Day 27-28: Testing Backend

- Write unit tests for critical functions

- Test all API endpoints with Postman
- Fix bugs and edge cases

Week 5-6: Stripe Integration & Socket.IO

Day 29-33: Stripe Payment Integration

- [ ] Set up Stripe account
- [ ] Create Stripe customer
- [ ] Create subscription plans
- [ ] Checkout session creation
- [ ] Webhook setup for payment events
- [ ] Subscription management
- [ ] Payment history tracking
- [ ] Cancel subscription

Stripe Products to Create:

1. Starter Plan - $29/month
2. Professional Plan - $79/month
3. Enterprise Plan - $199/month

bash

```
# Install Stripe CLI for webhook testing
stripe listen --forward-to localhost:5000/api/payments/webhook
```

Day 34-38: Real-time Chat (Socket.IO)

- [ ] Socket.IO server setup
- [ ] Authentication middleware for sockets
- [ ] Private messaging
- [ ] Room-based chat (company/team)
- [ ] Typing indicators
- [ ] Online status tracking
- [ ] Message read receipts
- [ ] Message persistence to database

Socket Events:

// Client to Server
- private-message
- room-message
- typing
- join-room
- leave-room
- mark-read
- update-status

// Server to Client
- private-message
- room-message
- user-typing
- user-status
- message-read

Day 39-42: Admin Dashboard APIs

- [ ] Company analytics endpoint
- [ ] User statistics endpoint
- [ ] Revenue tracking
- [ ] Export data endpoints
- [ ] System health monitoring

Week 7-8: Frontend Development

Day 43-45: Frontend Setup & Auth

- [ ] Set up Next.js project structure
- [ ] Install and configure Tailwind + shadcn
- [ ] Create layout components (Sidebar, Header)
- [ ] Build login page
- [ ] Build registration page
- [ ] Implement auth context/store
- [ ] Connect to backend auth APIs
- [ ] Protected routes setup

Day 46-49: Dashboard & Overview

- [ ] Dashboard layout
- [ ] Stat cards component
- [ ] Upcoming shifts component
- [ ] Recent activity feed
- [ ] Navigation system

- [ ] Responsive design

Day 50-53: Roster Management UI

- [ ] Calendar component
- [ ] Week/Month view toggle
- [ ] Shift cards
- [ ] Create roster modal
- [ ] Edit roster modal
- [ ] Drag-and-drop shifts (optional)
- [ ] Connect to backend APIs

Day 54-56: Team Management UI

- [ ] Team member list/grid
- [ ] Member detail cards
- [ ] Add member modal
- [ ] Edit member modal
- [ ] Search and filter functionality
- [ ] Role assignment UI
- [ ] Connect to backend APIs

Week 9-10: Advanced Features & Polish

Day 57-60: Chat Interface

- [ ] Chat layout (list + messages)
- [ ] Message bubbles
- [ ] Chat input component
- [ ] Socket.IO client integration
- [ ] Real-time message updates
- [ ] Typing indicators
- [ ] Online status display
- [ ] Unread message badges

Day 61-63: Payment & Subscription UI

- [ ] Current plan display
- [ ] Pricing page with plans
- [ ] Stripe checkout integration
- [ ] Payment history table
- [ ] Subscription management
- [ ] Cancel subscription flow

Day 64-66: Settings Pages

- [ ] Profile settings
- [ ] Company settings
- [ ] Notification preferences

- [ ] Security settings (password change)
- [ ] Account deletion

Day 67-70: Polish & Optimization

- [ ] Add loading states
- [ ] Error handling & user feedback
- [ ] Form validations
- [ ] Animations and transitions
- [ ] Mobile responsiveness
- [ ] Accessibility improvements
- [ ] Performance optimization
- [ ] SEO optimization

Week 11-12: Testing, Deployment & Launch

Day 71-75: Testing

Backend Testing:


bash

```
# Install testing libraries
npm install -D jest @types/jest supertest @types/supertest

# Create test files
backend/src/__tests__/
├── auth.test.ts
├── roster.test.ts
├── shift.test.ts
└── payment.test.ts
```

Frontend Testing:


bash

```
# Install testing libraries
npm install -D @testing-library/react @testing-library/jest-dom
npm install -D @testing-library/user-event
```

Test Checklist:

- [ ] Unit tests for services
- [ ] Integration tests for API endpoints
- [ ] E2E tests for critical flows
- [ ] Manual testing of all features
- [ ] Cross-browser testing
- [ ] Mobile device testing
- [ ] Performance testing
- [ ] Security testing (OWASP top 10)

Day 76-80: Deployment Setup

Environment Variables Setup:


bash

```
# Backend .env (Production)
NODE_ENV=production
PORT=5000
DATABASE_URL=postgresql://user:pass@host:5432/db
REDIS_URL=redis://host:6379
JWT_SECRET=your-super-secret-key
STRIPE_SECRET_KEY=sk_live_xxx
STRIPE_WEBHOOK_SECRET=whsec_xxx
FRONTEND_URL=https://rosterpro.com
```


bash

```
# Frontend .env.local (Production)
NEXT_PUBLIC_API_URL=https://api.rosterpro.com
NEXT_PUBLIC_SOCKET_URL=https://api.rosterpro.com
NEXT_PUBLIC_STRIPE_PUBLIC_KEY=pk_live_xxx
```

Backend Deployment (Railway Example):

1. **Create Railway Account**
   - Go to railway.app
   - Connect GitHub repository
2. **Set up PostgreSQL & Redis**


bash

```
# Railway will provide URLs automatically
```

3. **Deploy Backend**


bash

```
# Railway automatically deploys from GitHub
# Or use Railway CLI
railway login
railway link
railway up
```

4. **Configure Environment Variables**
   - Add all .env variables in Railway dashboard

Frontend Deployment (Vercel Example):

1. **Create Vercel Account**
   - Go to vercel.com
   - Import GitHub repository
2. **Configure Project**


bash

```
# Install Vercel CLI
npm install -g vercel

# Deploy
vercel --prod
```

3. **Set Environment Variables**
   - Add all .env.local variables in Vercel dashboard

Alternative Deployment Options:

Backend:

- Railway (Easiest, recommended)
- Render (Free tier available)
- AWS EC2 + Load Balancer
- DigitalOcean App Platform
- Google Cloud Run
- Heroku

Frontend:

- Vercel (Recommended for Next.js)
- Netlify
- AWS Amplify
- Cloudflare Pages

Database:

- Railway PostgreSQL
- Supabase (Free tier)
- AWS RDS
- ElephantSQL
- Neon (Serverless Postgres)

Redis:

- Upstash (Serverless Redis)
- Redis Cloud
- Railway Redis

Day 81-84: Domain & SSL


bash

```
# 1. Purchase domain
# - Namecheap, Google Domains, etc.

# 2. Configure DNS
# Frontend (Vercel)
A    @      76.76.21.21
CNAME www    cname.vercel-dns.com

# Backend (Railway)
A    api    railway-ip-address
```

3. **SSL Certificates**
    - Automatic with Vercel/Railway
    - Let's Encrypt for custom setups

Day 85-86: Monitoring & Analytics Setup


bash

```
# 1. Error Tracking (Sentry)
npm install @sentry/nextjs
npm install @sentry/node

# 2. Analytics (Google Analytics / Plausible)
# Add tracking code to layout

# 3. Uptime Monitoring
# - UptimeRobot
# - Pingdom
# - Better Uptime
```

🚀 Deployment Checklist

Pre-Deployment

- [ ] All environment variables configured
- [ ] Database migrations run
- [ ] Seed data created (if needed)
- [ ] API endpoints tested
- [ ] Frontend builds successfully
- [ ] No console errors
- [ ] Mobile responsive
- [ ] Cross-browser tested
- [ ] SSL certificates ready
- [ ] Stripe in production mode
- [ ] Backup strategy in place

Backend Deployment

- [ ] Build production Docker image
- [ ] Deploy to hosting platform
- [ ] Configure environment variables
- [ ] Run database migrations
- [ ] Configure CORS for frontend domain
- [ ] Set up health check endpoint
- [ ] Configure rate limiting
- [ ] Set up logging
- [ ] Configure error tracking

Frontend Deployment

- [ ] Build for production
- [ ] Deploy to Vercel/Netlify
- [ ] Configure environment variables

- [ ] Set up custom domain
- [ ] Configure redirects
- [ ] Enable CDN
- [ ] Set up analytics
- [ ] Configure SEO meta tags

Post-Deployment

- [ ] Test all features in production
- [ ] Verify payments work
- [ ] Test real-time chat
- [ ] Check email notifications
- [ ] Monitor error rates
- [ ] Set up alerts
- [ ] Create backup schedule
- [ ] Document API
- [ ] Create user documentation

📊 MVP Feature Checklist

Core Features (Must Have)

- [x] User authentication (register, login, logout)
- [x] Company registration
- [x] User roles (Admin, Manager, Employee)
- [x] Create/Edit/Delete rosters
- [x] Create/Edit/Delete shifts
- [x] Assign employees to shifts
- [x] View roster calendar
- [x] Team member management
- [x] Real-time chat
- [x] Stripe subscription integration
- [x] Payment history
- [x] Basic admin dashboard
- [x] Profile settings
- [x] Company settings

Nice to Have (Post-MVP)

- [ ] Shift swap requests
- [ ] Time-off requests & approval
- [ ] Shift templates
- [ ] Recurring shifts
- [ ] Email notifications
- [ ] SMS notifications
- [ ] Mobile app (React Native)
- [ ] Advanced analytics
- [ ] Export to PDF/Excel

- [ ] Multi-language support
- [ ] Dark mode
- [ ] Calendar integrations (Google, Outlook)
- [ ] Slack integration
- [ ] Custom branding
- [ ] API for third-party integrations

💰 Estimated Costs (Monthly)

Development Phase

- **Domain:** $12/year = $1/month
- **PostgreSQL:** Free (Railway free tier) or $5/month
- **Redis:** Free (Upstash free tier) or $5/month
- **Backend Hosting:** Free (Railway free tier) or $5-20/month
- **Frontend Hosting:** Free (Vercel free tier)
- **Total Development:** ~$0-30/month

Production (After Launch)

- **Domain:** $1/month
- **PostgreSQL:** $10-50/month (depending on usage)
- **Redis:** $10-20/month
- **Backend Hosting:** $20-100/month
- **Frontend Hosting:** $0-20/month
- **Stripe Fees:** 2.9% + $0.30 per transaction
- **Email Service:** $0-10/month (SendGrid free tier: 100 emails/day)
- **Error Tracking:** $0-26/month (Sentry free tier)
- **Monitoring:** Free (UptimeRobot free tier)
- **Total Production:** ~$40-250/month

🔐 Security Best Practices

1. **Environment Variables**
   - Never commit .env files
   - Use different keys for dev/prod
   - Rotate secrets regularly
2. **Authentication**
   - Use strong JWT secrets (32+ characters)
   - Implement refresh token rotation
   - Add rate limiting on auth endpoints
   - Use secure password hashing (bcrypt rounds: 10)
3. **API Security**
   - Implement CORS properly
   - Add rate limiting
   - Validate all inputs
   - Sanitize user data
   - Use HTTPS only in production
   - Add security headers (helmet.js)
4. **Database**

- Use parameterized queries (Prisma handles this)
- Regular backups
- Encrypt sensitive data
- Use connection pooling

5. **Stripe**
   - Verify webhook signatures
   - Use test mode for development
   - Never expose secret keys
   - Handle failed payments gracefully

📚 Documentation to Create

1. **README.md** - Project overview, setup instructions
2. **API Documentation** - All endpoints, request/response formats
3. **User Guide** - How to use the application
4. **Admin Guide** - Admin features and management
5. **Developer Guide** - Architecture, contributing guidelines
6. **Deployment Guide** - Step-by-step deployment instructions

---

🐛 Common Issues & Solutions

Issue: Database connection fails


bash

```
# Check PostgreSQL is running
docker ps

# Check DATABASE_URL format
postgresql://user:password@host:port/database

# Test connection
npx prisma studio
```

Issue: Redis connection fails


bash

```
# Check Redis is running
docker ps

# Test connection
redis-cli ping
```

Issue: JWT token expired

javascript

```
// Implement refresh token logic
// Check token expiration time
// Add token refresh endpoint
```

Issue: CORS errors

javascript

```
// Backend: Configure CORS properly
app.use(cors({
  origin: process.env.FRONTEND_URL,
  credentials: true
}));
```

Issue: Socket.IO not connecting

javascript

```
// Check CORS settings for Socket.IO
// Verify authentication token
// Check firewall rules
```

🎯 Success Metrics

Technical Metrics

- [ ] 99%+ uptime
- [ ] < 2s page load time
- [ ] < 500ms API response time
- [ ] 0 critical bugs
- [ ] 90%+ test coverage

Business Metrics

- [ ] 100+ signups in first month
- [ ] 10+ paying customers
- [ ] < 5% churn rate
- [ ] 4+ star average rating
- [ ] 80%+ user activation rate

📞 Support & Resources

Documentation

- Next.js: https://nextjs.org/docs
- Prisma: https://www.prisma.io/docs
- Stripe: https://stripe.com/docs
- Socket.IO: https://socket.io/docs
- Tailwind CSS: https://tailwindcss.com/docs

Communities

- Next.js Discord
- Prisma Slack
- Stack Overflow
- Reddit: r/webdev, r/reactjs, r/node

Learning Resources

- Next.js Tutorial
- Prisma Getting Started
- Stripe Integration Guide
- TypeScript Handbook

✅ Final Launch Checklist

- [ ] All features working in production
- [ ] Payment processing tested
- [ ] Email notifications working
- [ ] Real-time chat functional

- [ ] Mobile responsive
- [ ] SEO optimized
- [ ] Analytics tracking
- [ ] Error monitoring setup
- [ ] Backup system configured
- [ ] Documentation complete
- [ ] Terms of Service published
- [ ] Privacy Policy published
- [ ] Support email setup
- [ ] Social media accounts created
- [ ] Landing page live
- [ ] Marketing materials ready

Good luck building your MVP! 🚀

Estimated Total Development Time: 400-500 hours
Estimated Cost: $40-250/month operational costs
MVP Timeline: 8-12 weeks with dedicated development