ClockWork Backend

Complete backend implementation for the ClockWork Universal Business Platform.

🚀 Quick Start

Prerequisites

- Node.js 18+
- Docker & Docker Compose
- PostgreSQL (via Docker)
- Redis (via Docker)

Setup

1. Run the quick start script:

bash

```
chmod +x quickstart.sh
./quickstart.sh
```

2. Start the databases:

bash

```
docker-compose up -d
```

3. Run migrations:

bash

```
npm run migrate
```

4. Start the development server:

bash

```
npm run dev
```

Your backend is now running at `http://localhost:3001`!

📁 Project Structure

```
clockwork-backend/
├── src/
│   ├── config/         # Database, Redis, Stripe configurations
│   ├── controllers/    # Business logic for each route
│   ├── middleware/     # Auth, validation, error handling
│   ├── models/         # Database models (if using ORM)
│   ├── routes/        # API endpoint definitions
│   ├── services/      # Email, SMS, file upload services
│   ├── socket/        # Real-time WebSocket handlers
│   ├── utils/         # Helper functions and validators
│   └── server.js      # Main application entry point
├── migrations/        # Database migration files
├── seeds/             # Database seed files
├── tests/             # Test files
├── .env               # Environment variables
├── docker-compose.yml # Docker services configuration
├── knexfile.js        # Database migration config
└── package.json       # Dependencies and scripts
```

🔧 Configuration

Environment Variables

Create a `.env` file with the following variables:

env

```
# Server
NODE_ENV=development
PORT=3001

# Database
DATABASE_URL=postgresql://clockwork:password@localhost:5432/clockwork
REDIS_URL=redis://localhost:6379

# Authentication
JWT_SECRET=your-secret-key
JWT_REFRESH_SECRET=your-refresh-secret
JWT_EXPIRE=15m
JWT_REFRESH_EXPIRE=7d

# Stripe
STRIPE_SECRET_KEY=sk_test_...
STRIPE_WEBHOOK_SECRET=whsec_...

# Email (SendGrid)
SENDGRID_API_KEY=SG...
EMAIL_FROM=noreply@clockwork.platform

# Frontend
FRONTEND_URL=http://localhost:3000
```

📡 API Endpoints

Authentication

- `POST /api/auth/signup` - Create new account
- `POST /api/auth/login` - Login
- `POST /api/auth/logout` - Logout
- `POST /api/auth/refresh` - Refresh tokens
- `POST /api/auth/verify-2fa` - Verify 2FA code
- `POST /api/auth/reset-password/request` - Request password reset
- `POST /api/auth/reset-password/confirm` - Confirm password reset

Users

- `GET /api/users` - Get all users (admin only)
- `GET /api/users/:id` - Get user by ID
- `PUT /api/users/:id` - Update user
- `DELETE /api/users/:id` - Delete user

Measurements

- `GET /api/measurements` - Get measurements
- `POST /api/measurements` - Create measurement
- `PUT /api/measurements/:id` - Update measurement
- `DELETE /api/measurements/:id` - Delete measurement

Workouts

- `GET /api/workouts` - Get workouts
- `POST /api/workouts` - Create workout
- `PUT /api/workouts/:id` - Update workout
- `DELETE /api/workouts/:id` - Delete workout
- `POST /api/workouts/:id/complete` - Mark workout complete

Nutrition

- `GET /api/nutrition` - Get nutrition plan
- `PUT /api/nutrition` - Update nutrition plan

Goals

- `GET /api/goals` - Get goals
- `POST /api/goals` - Create goal
- `PUT /api/goals/:id` - Update goal

- `DELETE /api/goals/:id` - Delete goal

Billing

- `GET /api/billing/invoices` - Get invoices
- `GET /api/billing/subscriptions` - Get subscriptions
- `POST /api/billing/checkout` - Create checkout session
- `POST /api/billing/webhook` - Stripe webhook handler
- `DELETE /api/billing/subscriptions/:id` - Cancel subscription

Chat (WebSocket)

- `connection` - Authenticate and connect
- `join-conversation` - Join a conversation room
- `send-message` - Send a message
- `typing` - Send typing indicator
- `mark-read` - Mark messages as read

🗄️ Database Schema

Core Tables

- `users` - User accounts and profiles
- `measurements` - Body measurements and health data
- `workouts` - Workout plans and exercises
- `nutrition` - Nutrition plans and tracking
- `goals` - Client goals and milestones
- `messages` - Chat messages
- `invoices` - Billing invoices
- `subscriptions` - Recurring subscriptions
- `audit_logs` - Activity tracking

🧪 Testing

Run the test suite:
bash

npm test

Run tests in watch mode:
bash

npm test -- --watch

🚀 Deployment

Using Docker

1. Build the image:

bash

```
docker build -t clockwork-backend .
```

2. Run with Docker Compose:

bash

```
docker-compose -f docker-compose.prod.yml up -d
```

Manual Deployment

1. Install dependencies:

bash

```
npm ci --only=production
```

2. Run migrations:

bash

```
NODE_ENV=production npm run migrate
```

3. Start with PM2:

bash

```
pm2 start src/server.js --name clockwork-backend
```

🔒 Security Features

- JWT authentication with refresh tokens
- Two-factor authentication (2FA)
- Password hashing with bcrypt
- Rate limiting on sensitive endpoints
- Input validation with Joi
- SQL injection protection
- XSS protection with helmet
- CORS configuration

📊 Monitoring

- Health check endpoint: `GET /health`
- Structured logging with Winston
- Error tracking ready for Sentry integration

- Performance monitoring ready for New Relic/DataDog

🛠️ Development

Adding a New Feature

1. Create migration:

bash

```
npm run migrate:make add_feature_table
```

2. Create controller:

javascript

```
// src/controllers/featureController.js
const { db } = require('../config/database');

const getFeatures = async (req, res) => {
  // Implementation
};


module.exports = { getFeatures };
```

3. Create routes:

javascript

```
// src/routes/features.js
const router = require('express').Router();
const { authenticate } = require('../middleware/auth');
const featureController = require('../controllers/featureController');

router.get('/', authenticate, featureController.getFeatures);


module.exports = router;
```

4. Add to server.js:

javascript

```
const featureRoutes = require('./routes/features');
app.use('/api/features', featureRoutes);
```

Common Commands

bash

```
# Start development server
npm run dev
```

```
# Run migrations
npm run migrate

# Create new migration
npm run migrate:make migration_name

# Run seeds
npm run seed

# Create new seed
npm run seed:make seed_name

# Run tests
npm test

# Check code style
npm run lint

# Format code

npm run format
```

🤝 Contributing

1. Fork the repository
2. Create your feature branch (`git checkout -b feature/amazing-feature`)
3. Commit your changes (`git commit -m 'Add some amazing feature'`)
4. Push to the branch (`git push origin feature/amazing-feature`)
5. Open a Pull Request

📝 License

This project is licensed under the MIT License.

🆘 Support

For support, email support@clockwork.platform or join our Slack channel.

🎯 Next Steps

1. Set up Stripe: Get your API keys from Stripe Dashboard
2. Set up SendGrid: Get your API key from SendGrid
3. Set up AWS S3: Configure bucket for file uploads
4. Enable 2FA: Set up TOTP for enhanced security

5. Configure monitoring: Set up Sentry, New Relic, or DataDog

📚 Resources

- API Documentation (when running)
- Database Schema
- WebSocket Events
- Deployment Guide