Φορολογικός Υπολογιστής - Installation & Deployment Guide

📋 Table of Contents

1. Prerequisites

Required Software

Node.js and Yarn

- **Node.js**: Version 18.x or higher
- **Yarn**: Version 1.22.x or higher

Installation:

```
# Check if Node.js is installed
node --version

# Check if Yarn is installed
yarn --version

# Install Node.js (if not installed)
# Ubuntu/Debian:
sudo apt update
sudo apt install nodejs npm

# macOS (with Homebrew):
brew install node

# Windows:
# Download from https://nodejs.org/

# Install Yarn globally
npm install -g yarn
```

PostgreSQL Database

- **Version**: 12.x or higher

Installation:

```
# Ubuntu/Debian:
sudo apt update
sudo apt install postgresql postgresql-contrib

# macOS (with Homebrew):
brew install postgresql
brew services start postgresql

# Windows:
# Download from https://www.postgresql.org/download/windows/
```

Git (Optional but Recommended)

```
# Ubuntu/Debian:
sudo apt install git

# macOS:
brew install git

# Windows:
# Download from https://git-scm.com/
```

---

2. Local Development Setup

Step 1: Get the Project Files

If you have the project directory:

```
# Navigate to the project
cd /path/to/greek_tax_calculator
```

If you have it in a Git repository:

```
# Clone the repository
git clone <repository-url>
cd greek_tax_calculator
```

Step 2: Install Dependencies

```
# Navigate to the Next.js workspace
cd nextjs_space

# Install all dependencies
yarn install
```

This will install all required packages listed in `package.json` (~100 dependencies).

Expected output:

```
✨ Done in XX.XXs
```

Step 3: Create Environment File

```
# Create .env file in the nextjs_space directory
touch .env
```

See Environment Variables section for configuration.

---

3. Environment Variables

Required Variables

Create a `.env` file in the `nextjs_space` directory with the following variables:

```
# ========================================
# DATABASE CONNECTION
# ========================================
DATABASE_URL="postgresql://username:password@localhost:5432/database_name"

# ========================================
# NEXTAUTH CONFIGURATION
# ========================================
NEXTAUTH_SECRET="your-secret-key-here-min-32-chars"
NEXTAUTH_URL="http://localhost:3000"

# ========================================
# OPTIONAL: NODE ENVIRONMENT
# ========================================
NODE_ENV="development"
```

Detailed Configuration

1. DATABASE_URL

Format: `postgresql://USER:PASSWORD@HOST:PORT/DATABASE`

Example for local PostgreSQL:

```
DATABASE_URL="postgresql://postgres:mypassword@localhost:5432/greek_tax_db"
```

Components:
- `USER` : PostgreSQL username (default: `postgres`)
- `PASSWORD` : Your PostgreSQL password
- `HOST` : Database host (default: `localhost`)
- `PORT` : PostgreSQL port (default: `5432`)
- `DATABASE` : Database name (e.g., `greek_tax_db`)

2. NEXTAUTH_SECRET

Purpose: Encrypts JWT tokens for session management

Generation:

```
# Generate a random secret (32+ characters)
openssl rand -base64 32
```

Example:

```
NEXTAUTH_SECRET="A1b2C3d4E5f6G7h8I9j0K1l2M3n4O5p6Q7r8S9t0U1v2"
```

⚠️ **IMPORTANT**: Use a different secret for production!

3. NEXTAUTH_URL

Development: `http://localhost:3000`
Production: Your deployed URL (e.g., `https://yourdomain.com`)

Sample .env File

```
# Greek Tax Calculator - Environment Variables
# Created: 2025-12-19

# Database
DATABASE_URL="postgresql://postgres:admin123@localhost:5432/greek_tax_calculator"

# Authentication
NEXTAUTH_SECRET="K9xY3mQ8vT2pL5nH7jW1bF4sR6dG0zC8vB2xM5qA3tE9"
NEXTAUTH_URL="http://localhost:3000"

# Environment
NODE_ENV="development"
```

---

4. Database Setup

Step 1: Create PostgreSQL Database

```
# Connect to PostgreSQL
sudo -u postgres psql

# Or on macOS/Windows:
psql -U postgres
```

In PostgreSQL prompt:

```
-- Create database
CREATE DATABASE greek_tax_calculator;

-- Create user (optional, if not using default postgres user)
CREATE USER taxcalc_user WITH ENCRYPTED PASSWORD 'secure_password_here';

-- Grant privileges
GRANT ALL PRIVILEGES ON DATABASE greek_tax_calculator TO taxcalc_user;

-- Exit
\q
```

Step 2: Run Prisma Migrations

```
# Make sure you're in nextjs_space directory
cd nextjs_space

# Generate Prisma Client
yarn prisma generate

# Push schema to database (for development)
yarn prisma db push

# OR for production, use migrations:
yarn prisma migrate deploy
```

Expected output:

```
✔ Generated Prisma Client
🚀 Your database is now in sync with your Prisma schema
```

Step 3: Seed the Database

```
# Run the seed script
yarn prisma db seed
```

This creates:
- Test user: `john@doe.com` / `johndoe123`
- Sample business: ΛΙΧΑΣ ΠΑΝΑΓΙΩΤΗΣ
- Sample calculations for 2024 and 2025

Expected output:

```
✅ Seed user created: john@doe.com
✅ Sample business created: ΛΙΧΑΣ ΠΑΝΑΓΙΩΤΗΣ
✅ Sample calculations created for 2024 and 2025
🌱 Database seeded successfully!
```

Verify Database Setup

```
# Open Prisma Studio (visual database browser)
yarn prisma studio
```

This opens `http://localhost:5555` where you can browse your database tables.

---

5. Running the Application

Development Server

```
# Make sure you're in nextjs_space directory
cd nextjs_space

# Start the development server
yarn dev
```

Output:

```
▲ Next.js 14.2.28
- Local:        http://localhost:3000
- Network:      http://192.168.1.100:3000

✓ Ready in 2.5s
```

Open your browser: http://localhost:3000

Production Build (Local)

```
# Build the application
yarn build

# Start production server
yarn start
```

Available Scripts

```
# Development mode with hot reload
yarn dev

# Production build
yarn build

# Start production server
yarn start

# Lint code
yarn lint

# Prisma commands
yarn prisma generate      # Generate Prisma Client
yarn prisma db push       # Push schema to DB (dev)
yarn prisma db seed       # Seed database
yarn prisma studio        # Open Prisma Studio
yarn prisma migrate dev   # Create migration
```

6. Cloud Deployment

Option A: Vercel (Recommended for Next.js)

Prerequisites

- Vercel account (free tier available)
- Git repository (GitHub, GitLab, or Bitbucket)
- PostgreSQL database (e.g., Neon, Supabase, or AWS RDS)

Step 1: Prepare Database

Option 1: Neon (PostgreSQL Cloud - Free Tier)
1. Go to https://neon.tech
2. Create account and new project
3. Copy connection string

Option 2: Supabase (PostgreSQL + More - Free Tier)
1. Go to https://supabase.com
2. Create new project
3. Go to Settings > Database
4. Copy connection string

Step 2: Deploy to Vercel

Via Vercel Dashboard:
1. Go to https://vercel.com
2. Click “Add New” → “Project”
3. Import your Git repository
4. Configure:
- **Framework Preset**: Next.js
- **Root Directory**: `nextjs_space`
- **Build Command**: `yarn build`
- **Output Directory**: `.next`

1. Add Environment Variables:
```
DATABASE_URL=<your-cloud-database-url>
   NEXTAUTH_SECRET=<generate-new-secret>
   NEXTAUTH_URL=https://your-app-name.vercel.app
```
2. Click “Deploy”

Via Vercel CLI:

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

# Navigate to nextjs_space
cd nextjs_space

# Login
vercel login

# Deploy
vercel

# Follow prompts and set environment variables
```

Step 3: Run Migrations on Cloud Database

```
# Set DATABASE_URL to your cloud database
export DATABASE_URL="postgresql://user:pass@host:5432/dbname"

# Run migrations
yarn prisma migrate deploy

# Seed database (optional)
yarn prisma db seed
```

Option B: DigitalOcean App Platform

Step 1: Create PostgreSQL Database

1. Go to DigitalOcean Dashboard
2. Create → Databases → PostgreSQL
3. Note connection details

Step 2: Create App

1. Create → Apps
2. Connect Git repository
3. Configure:
   - **Source Directory**: `nextjs_space`
   - **Build Command**: `yarn install && yarn build`
   - **Run Command**: `yarn start`
   - **Port**: 3000
4. Add Environment Variables
5. Deploy

Option C: AWS (EC2 + RDS)

Infrastructure Setup

1. Create RDS PostgreSQL Instance

```
- Go to AWS RDS Console
- Create Database
- Engine: PostgreSQL 14+
- Instance: db.t3.micro (free tier)
- Set master password
- Note endpoint and port
```

2. Launch EC2 Instance

```
- AMI: Ubuntu 22.04 LTS
- Instance: t2.micro (free tier)
- Security Group: Allow SSH (22), HTTP (80), HTTPS (443)
- Create or use existing key pair
```

3. Connect to EC2

```
ssh -i your-key.pem ubuntu@your-ec2-ip
```

4. Install Dependencies on EC2

```
# Update system
sudo apt update && sudo apt upgrade -y

# Install Node.js 18
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install -y nodejs

# Install Yarn
npm install -g yarn

# Install PM2 (process manager)
npm install -g pm2

# Install Git
sudo apt install git
```

5. Deploy Application

```
# Clone repository
git clone <your-repo-url>
cd greek_tax_calculator/nextjs_space

# Create .env file
nano .env
# Add your environment variables (use RDS endpoint for DATABASE_URL)

# Install dependencies
yarn install

# Run Prisma migrations
yarn prisma generate
yarn prisma migrate deploy
yarn prisma db seed

# Build application
yarn build

# Start with PM2
pm2 start yarn --name "tax-calculator" -- start
pm2 save
pm2 startup
```

6. Setup Nginx (Optional but Recommended)

```
# Install Nginx
sudo apt install nginx

# Create Nginx config
sudo nano /etc/nginx/sites-available/tax-calculator
```

Nginx Configuration:

```
server {
    listen 80;
    server_name your-domain.com;

    location / {
        proxy_pass http://localhost:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

```
# Enable site
sudo ln -s /etc/nginx/sites-available/tax-calculator /etc/nginx/sites-enabled/
sudo nginx -t
sudo systemctl restart nginx
```

7. Setup SSL with Let’s Encrypt

```
sudo apt install certbot python3-certbot-nginx
sudo certbot --nginx -d your-domain.com
```

Option D: Docker Deployment

Create Dockerfile

Create `Dockerfile` in `nextjs_space` directory:

```
# Dockerfile
FROM node:18-alpine AS base

# Install dependencies only when needed
FROM base AS deps
RUN apk add --no-cache libc6-compat
WORKDIR /app

COPY package.json yarn.lock ./
RUN yarn install --frozen-lockfile

# Rebuild the source code only when needed
FROM base AS builder
WORKDIR /app
COPY --from=deps /app/node_modules ./node_modules
COPY . .

# Generate Prisma Client
RUN yarn prisma generate

# Build Next.js
RUN yarn build

# Production image
FROM base AS runner
WORKDIR /app

ENV NODE_ENV production

RUN addgroup --system --gid 1001 nodejs
RUN adduser --system --uid 1001 nextjs

COPY --from=builder /app/public ./public
COPY --from=builder --chown=nextjs:nodejs /app/.next/standalone ./
COPY --from=builder --chown=nextjs:nodejs /app/.next/static ./.next/static
COPY --from=builder /app/node_modules/.prisma ./node_modules/.prisma
COPY --from=builder /app/prisma ./prisma

USER nextjs

EXPOSE 3000

ENV PORT 3000

CMD ["node", "server.js"]
```

Create docker-compose.yml

```yaml
# docker-compose.yml
version: '3.8'

services:
  db:
    image: postgres:14-alpine
    restart: always
    environment:
      POSTGRES_DB: greek_tax_calculator
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: your_password_here
    volumes:
      - postgres_data:/var/lib/postgresql/data
    ports:
      - "5432:5432"

  app:
    build:
      context: ./nextjs_space
      dockerfile: Dockerfile
    restart: always
    ports:
      - "3000:3000"
    environment:
      DATABASE_URL: postgresql://postgres:your_password_here@db:5432/
greek_tax_calculator
      NEXTAUTH_SECRET: your_nextauth_secret_here
      NEXTAUTH_URL: http://localhost:3000
      NODE_ENV: production
    depends_on:
      - db
    command: sh -c "npx prisma migrate deploy && node server.js"

volumes:
  postgres_data:
```

Deploy with Docker

```bash
# Build and start containers
docker-compose up -d

# View logs
docker-compose logs -f app

# Stop containers
docker-compose down

# Stop and remove volumes (deletes database!)
docker-compose down -v
```

7. Troubleshooting

Common Issues

1. “Cannot connect to database”

Symptoms:

```
Error: P1001: Can't reach database server
```

Solutions:
- Verify PostgreSQL is running: `sudo systemctl status postgresql`
- Check DATABASE_URL in .env file
- Verify database exists: `psql -U postgres -l`
- Check PostgreSQL logs: `sudo tail -f /var/log/postgresql/postgresql-14-main.log`

2. “Prisma Client not generated”

Symptoms:

```
Error: @prisma/client did not initialize yet
```

Solution:

```
yarn prisma generate
```

3. “Port 3000 already in use”

Symptoms:

```
Error: listen EADDRINUSE: address already in use :::3000
```

Solutions:

```
# Find process using port 3000
lsof -ti:3000

# Kill the process
kill -9 $(lsof -ti:3000)

# Or use a different port
PORT=3001 yarn dev
```

4. “NextAuth secret required”

Symptoms:

```
Error: NEXTAUTH_SECRET is required
```

Solution:

```
# Generate secret
openssl rand -base64 32

# Add to .env
NEXTAUTH_SECRET="<generated-secret>"
```

5. Build Errors

Symptoms:

```
Type error: Property 'id' does not exist on type 'User'
```

Solution:
- Ensure `types/next-auth.d.ts` exists
- Restart TypeScript server in your editor
- Delete `.next` folder and rebuild:

```
rm -rf .next
yarn build
```

6. Authentication Not Working

Symptoms:
- Login redirects to error page
- Session not persisting

Solutions:
- Check NEXTAUTH_URL matches your domain
- Clear browser cookies
- Verify NEXTAUTH_SECRET is set
- Check API route: `curl http://localhost:3000/api/auth/session`

---

8. Maintenance

Backup Database

```
# Full database backup
pg_dump -U postgres greek_tax_calculator > backup_$(date +%Y%m%d).sql

# Restore from backup
psql -U postgres greek_tax_calculator < backup_20251219.sql
```

Update Dependencies

```
# Check for outdated packages
yarn outdated

# Update all dependencies
yarn upgrade

# Update specific package
yarn upgrade package-name
```

Database Migrations

```
# Create new migration
yarn prisma migrate dev --name add_new_feature

# Apply migrations (production)
yarn prisma migrate deploy

# Reset database (WARNING: deletes all data)
yarn prisma migrate reset
```

Monitor Logs

Development:
- Next.js logs in terminal
- Browser console for client errors

Production (PM2):

```
pm2 logs tax-calculator
pm2 monit
```

Production (Docker):

```
docker-compose logs -f app
```

Performance Optimization

1. **Enable caching** (add to `next.config.js`):

```
module.exports = {
  swcMinify: true,
  compress: true,
  poweredByHeader: false,
};
```

1. **Database indexes** (already in schema):
   - userId, businessId, year are indexed

2. **Monitor database queries**:

```
# Enable Prisma query logging
DEBUG=prisma:query yarn dev
```

Quick Start Checklist

For Local Development

- [ ] Install Node.js 18+
- [ ] Install Yarn
- [ ] Install PostgreSQL
- [ ] Clone/download project
- [ ] `cd nextjs_space`
- [ ] `yarn install`
- [ ] Create `.env` file with DATABASE_URL and NEXTAUTH_SECRET
- [ ] Create PostgreSQL database
- [ ] `yarn prisma generate`
- [ ] `yarn prisma db push`
- [ ] `yarn prisma db seed`
- [ ] `yarn dev`
- [ ] Open http://localhost:3000
- [ ] Login with `john@doe.com` / `johndoe123`

For Cloud Deployment (Vercel)

- [ ] Create Vercel account
- [ ] Create cloud PostgreSQL (Neon/Supabase)
- [ ] Push code to Git repository
- [ ] Import project to Vercel
- [ ] Set root directory to `nextjs_space`
- [ ] Add environment variables
- [ ] Deploy
- [ ] Run `prisma migrate deploy` with cloud DATABASE_URL
- [ ] Visit deployed URL

Support Resources

- **Next.js Documentation**: https://nextjs.org/docs
- **Prisma Documentation**: https://www.prisma.io/docs
- **NextAuth.js Documentation**: https://next-auth.js.org
- **PostgreSQL Documentation**: https://www.postgresql.org/docs
- **Vercel Documentation**: https://vercel.com/docs

Document Version: 1.0
Last Updated: December 19, 2025
Tested On: Ubuntu 22.04, macOS 13, Windows 11