**🚀 Mivton Project - Phase 1.3 Context Prompt**

**Copy and paste this entire prompt in any new chat with Claude:**

**I am working on MIVTON - a futuristic multilingual chat platform. You are my technical developer. We have completed Phases 1.1 and 1.2, and now need to implement Phase 1.3 - Authentication System.**

**🎯 Project Overview**

* **Name**: Mivton
* **Purpose**: Friends-only multilingual text chat with real-time OpenAI translation
* **Target**: Gen Z users with futuristic design
* **Status**: Phase 1.1 ✅ COMPLETE, Phase 1.2 ✅ COMPLETE, Phase 1.3 🔄 IN PROGRESS

**✅ Phase 1.1 - COMPLETED**

* **Landing Page**: Live at https://mivton-production.up.railway.app
* **Domain**: mivton.com (connected, DNS propagating)
* **Infrastructure**: Express.js + Socket.IO + Railway deployment
* **Design**: Futuristic Gen Z aesthetic with glassmorphism effects

**✅ Phase 1.2 - COMPLETED**

* **Database**: PostgreSQL fully set up on Railway
* **Tables Created**: users, session, waitlist, user\_stats
* **Features**: Waitlist system storing emails, database health checks
* **API Endpoints**: /health, /api/status, /api/waitlist, /api/waitlist/stats
* **Infrastructure**: Connection pooling, session management, error handling
* **Files**: database/, middleware/, utils/ directories complete

**🔄 CURRENT PHASE 1.3 - Authentication System (Day 2-3)**

**Goal**: Complete user registration and login system with futuristic UI **Priority**: IMPLEMENT THIS PHASE

**Phase 1.3 Tasks:**

* [ ] Build user registration with validation
* [ ] Implement secure login with bcrypt
* [ ] Create session management
* [ ] Design futuristic login/register pages
* [ ] Add basic email verification setup
* [ ] Test: Complete user registration and login flow

**Required Authentication Features:**

1. **User Registration**
   * Email validation and uniqueness check
   * Password strength requirements
   * Username availability validation
   * Language selection (50+ languages)
   * Gender selection with modern options
2. **Secure Login**
   * bcrypt password hashing (10 rounds)
   * Session creation and management
   * Remember me functionality
   * Login attempt rate limiting
3. **Futuristic UI Pages**
   * Modern login page with glassmorphism
   * Registration form with smooth animations
   * Form validation with real-time feedback
   * Loading states and micro-interactions
4. **Email Integration**
   * Welcome emails via Hostinger SMTP
   * Email verification system (basic setup)
   * Password reset preparation

**🗄️ Database Schema (Already Created in Phase 1.2)**

-- Users table ready for authentication

users (

id SERIAL PRIMARY KEY,

username VARCHAR(20) UNIQUE NOT NULL,

email VARCHAR(255) UNIQUE NOT NULL,

password\_hash VARCHAR(255) NOT NULL,

full\_name VARCHAR(100) NOT NULL,

gender VARCHAR(10) NOT NULL,

native\_language VARCHAR(10) NOT NULL,

is\_verified BOOLEAN DEFAULT TRUE,

is\_admin BOOLEAN DEFAULT FALSE,

admin\_level INTEGER DEFAULT 0,

is\_blocked BOOLEAN DEFAULT FALSE,

status VARCHAR(20) DEFAULT 'offline',

last\_login TIMESTAMP DEFAULT NULL,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

updated\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

);

-- Session table ready for authentication

session (

sid VARCHAR NOT NULL PRIMARY KEY,

sess JSON NOT NULL,

expire TIMESTAMP(6) NOT NULL

);

**🔧 Technical Infrastructure (Ready)**

**Railway Environment Variables ✅**

NODE\_ENV=production

JWT\_SECRET=mivton-super-secret-jwt-key-2025-production

OPENAI\_API\_KEY=sk-proj-ssWG4RYWzRSkC6N5wSrwR-ajRzmcGMLG2agNfBO1IRpplc7a8LzwmHrKNeShj4J2gm8ynWDu\_2T3BlbkFJob8\_1Ny3bs5vVdEKNo48BEsXZOB4kGzgtQ-O0-JAssePAFHU7CFAi-cLPc-xYop2z362vHtd4A

SMTP\_HOST=smtp.hostinger.com

SMTP\_PORT=465

SMTP\_SECURE=true

SMTP\_USER=info@mivton.com

SMTP\_PASS=Bacau@2012

APP\_URL=https://mivton.com

FRONTEND\_URL=https://mivton.com

DATABASE\_URL=[Railway PostgreSQL URL]

**Current File Structure ✅**

mivton/

├── server.js # Main Express server ✅

├── package.json # Dependencies ✅

├── .env # Environment variables ✅

├── public/

│ ├── index.html # Landing page ✅

│ ├── css/style.css # Futuristic styles ✅

│ └── js/app.js # Frontend logic ✅

├── database/ # Database layer ✅

│ ├── connection.js # PostgreSQL connection ✅

│ ├── schema.sql # Core schema ✅

│ └── init.js # DB initialization ✅

├── middleware/ # Middleware layer ✅

│ └── database.js # DB middleware ✅

├── utils/ # Utility layer ✅

│ ├── database.js # DB utilities ✅

│ └── waitlist.js # Waitlist functions ✅

├── routes/ # 🔄 NEEDS AUTH ROUTES

└── socket/ # 🔄 READY FOR FUTURE

**🎨 Design System (Established)**

Primary: #6366f1 (Electric Blue)

Secondary: #8b5cf6 (Vibrant Purple)

Accent: #06b6d4 (Cyan)

Success: #10b981 (Green)

Warning: #f59e0b (Amber)

Error: #ef4444 (Red)

Background: #0f172a (Dark Navy)

Surface: #1e293b (Slate)

Text: #f1f5f9 (Light)

**💻 Technology Stack**

* **Backend**: Node.js + Express.js ✅
* **Database**: PostgreSQL (Railway) ✅
* **Real-time**: Socket.IO ✅
* **Sessions**: PostgreSQL-backed sessions ✅
* **Translation**: OpenAI GPT-4 (ready)
* **Email**: Hostinger SMTP ✅
* **Frontend**: HTML5 + CSS3 + Vanilla JS ✅
* **Deployment**: Railway CLI (no GitHub) ✅

**📋 Phase 1.3 Specific Requirements**

**New Dependencies Needed:**

* bcrypt - Password hashing
* express-validator - Input validation
* nodemailer - Email functionality (Hostinger SMTP)

**Files to Create in Phase 1.3:**

routes/

├── auth.js # Authentication routes

└── users.js # User management routes

middleware/

├── auth.js # Auth middleware

└── validation.js # Input validation

utils/

├── auth.js # Auth utilities

├── email.js # Email system (Hostinger)

└── validation.js # Validation helpers

public/

├── login.html # Futuristic login page

├── register.html # Futuristic register page

├── dashboard.html # User dashboard (basic)

├── css/

│ └── auth.css # Authentication page styles

└── js/

├── auth.js # Auth page interactions

└── validation.js # Client-side validation

**API Endpoints to Create:**

* POST /api/auth/register - User registration
* POST /api/auth/login - User login
* POST /api/auth/logout - User logout
* GET /api/auth/me - Current user data
* GET /api/auth/check-username - Username availability
* GET /api/auth/check-email - Email availability

**Authentication Features:**

1. **Password Security**: bcrypt hashing with 10 rounds
2. **Session Management**: PostgreSQL-backed sessions
3. **Input Validation**: Server-side and client-side validation
4. **Rate Limiting**: Login attempt protection
5. **Email Integration**: Welcome emails via Hostinger
6. **Language Support**: 50+ language selection during signup

**🔮 Phase 2.1 Preview (Dashboard Framework)**

**After Phase 1.3, we'll implement:**

* Main dashboard HTML structure
* Sidebar navigation with glassmorphism
* Responsive mobile navigation
* User profile section
* Navigation between auth pages and dashboard

This context helps prepare authentication to integrate with dashboard.

**👥 Our Roles**

* **You (Claude)**: Create authentication files, routes, middleware, futuristic UI pages
* **Me**: Copy files to directories, run railway up to deploy, test functionality

**🚀 What I Need Help With**

I need you to help me implement Phase 1.3 - Authentication System. Specifically:

1. **Authentication routes** - Register, login, logout endpoints
2. **Password security** - bcrypt implementation
3. **Futuristic UI pages** - Login/register with Gen Z design
4. **Input validation** - Server and client-side validation
5. **Email integration** - Welcome emails via Hostinger SMTP
6. **Session management** - User login state management
7. **Integration instructions** - How to add auth to existing server.js

Please provide the exact files, code, and deployment steps I need for Phase 1.3!

**🎯 Success Criteria**

* Users can register with email/username/password
* Users can login securely with session management
* Futuristic login/register pages match Mivton design
* Email verification system foundation in place
* All authentication flows working on production

**END OF PROMPT - Ready for Phase 1.3 Authentication System Implementation**