Narula Immigration – Signup API (Node.js + Express + Joi + Sequelize + MySQL)

This document defines a **robust Signup API** with **strong Joi validation**, **Agent/Student logic**, and **Sequelize models**.

1️⃣Business Rules

User Types

- **Agent**
- **Student**

Agent-specific Rules

- Agent `user_id` format: **AGT10001, AGT10002, ...**
- `is_verified = 0` → Not verified (default)
- `is_verified = 1` → Verified

Student-specific Rules

- Normal auto-increment `id`
- `is_verified = 1` by default

2️⃣Database Model (Sequelize)

☢️models/User.ts

```
import { DataTypes, Model, Optional } from "sequelize";
import sequelize from "../config/database";

interface UserAttributes {
  id: number;
  user_id: string | null;
  name: string;
  email: string;
  phone_number: string;
  password: string;
  user_type: "agent" | "student";
  is_verified: boolean;
  createdAt?: Date;
  updatedAt?: Date;
}

interface UserCreationAttributes extends Optional<UserAttributes, "id" |
"user_id" | "is_verified"> {}

class User extends Model<UserAttributes, UserCreationAttributes> implements
UserAttributes {
  public id!: number;
  public user_id!: string | null;
  public name!: string;
  public email!: string;
  public phone_number!: string;
  public password!: string;
  public user_type!: "agent" | "student";
  public is_verified!: boolean;
}

User.init(
  {
    id: {
      type: DataTypes.INTEGER,
      autoIncrement: true,
      primaryKey: true,
    },
    user_id: {
      type: DataTypes.STRING,
      allowNull: true,
      unique: true,
    },
    name: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    email: {
      type: DataTypes.STRING,
      allowNull: false,
      unique: true,
    },
    phone_number: {
      type: DataTypes.STRING,
      allowNull: false,
      unique: true,
    },
    password: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    user_type: {
      type: DataTypes.ENUM("agent", "student"),
      allowNull: false,
    },
    is_verified: {
      type: DataTypes.BOOLEAN,
      defaultValue: false,
    },
  },
  {
    sequelize,
    tableName: "users",
  }
);

export default User;
```

3️⃣Joi Validation Schema

☢️validations/signup.validation.ts

```
import Joi from "joi";

export const signupSchema = Joi.object({
  name: Joi.string().min(3).max(50).required(),

  email: Joi.string().email().required(),

  phone_number: Joi.string()
    .pattern(/^[6-9][0-9]{9}$/)
    .required()
    .messages({
      "string.pattern.base": "Invalid Indian phone number",
    }),

  password: Joi.string()
    .min(8)
    .pattern(/^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])/)
    .required()
    .messages({
      "string.pattern.base": "Password must contain upper, lower, number &
special character",
    }),
  user_type: Joi.string().valid("agent", "student").required(),
});
```

4️⃣Utility: Generate Agent ID

☢️utils/generateAgentId.ts

```
import User from "../models/User";

export const generateAgentId = async (): Promise<string> => {
  const lastAgent = await User.findOne({
    where: { user_type: "agent" },
    order: [["createdAt", "DESC"]],
  });

  if (!lastAgent || !lastAgent.user_id) {
    return "AGT10001";
  }

  const lastNumber = parseInt(lastAgent.user_id.replace("AGT", ""));
  return `AGT${lastNumber + 1}`;
};
```

5️⃣Signup Controller (Strong Logic)

☢️controllers/auth.controller.ts

```
import { Request, Response } from "express";
import bcrypt from "bcrypt";
import User from "../models/User";
import { signupSchema } from "../validations/signup.validation";
import { generateAgentId } from "../utils/generateAgentId";

export const signup = async (req: Request, res: Response) => {
  try {
    // Joi Validation
    const { error, value } = signupSchema.validate(req.body);
    if (error) {
      return res.status(400).json({
        success: false,
        message: error.details[0].message,
    });
  }

  const { name, email, phone_number, password, user_type } = value;

  // Check existing user
  const existingUser = await User.findOne({
    where: { email },
  });

  if (existingUser) {
    return res.status(409).json({
      success: false,
      message: "Email already registered",
    });
  }

  // Hash password
  const hashedPassword = await bcrypt.hash(password, 10);

  let userId: string | null = null;
  let is_verified = true;

  // Agent Logic
  if (user_type === "agent") {
    userId = await generateAgentId();
    is_verified = false;
  }

  // Create User
  const user = await User.create({
    user_id: userId,
    name,
    email,
    phone_number,
    password: hashedPassword,
    user_type,
    is_verified,
  });

  return res.status(201).json({
    success: true,
    message: "Signup successful",
    data: {
      id: user.id,
      user_id: user.user_id,
      name: user.name,
      email: user.email,
          phone_number: user.phone_number,
          user_type: user.user_type,
          is_verified: user.is_verified,
        },
      });
    } catch (err) {
      console.error(err);
      return res.status(500).json({
        success: false,
        message: "Internal server error",
      });
    }
  };
```

6️⃣Route Definition

☢️routes/auth.routes.ts

```
import { Router } from "express";
import { signup } from "../controllers/auth.controller";

const router = Router();

router.post("/signup", signup);

export default router;
```

7️⃣API Endpoint

```
POST /api/auth/signup
```

Sample Request

```
{
  "name": "Rahul Sharma",
  "email": "rahul@gmail.com",
  "phone_number": "9876543210",
  "password": "Strong@123",
  "user_type": "agent"
}
```

Sample Response

```json
{
  "success": true,
  "message": "Signup successful",
  "data": {
    "id": 1,
    "user_id": "AGT10001",
    "name": "Rahul Sharma",
    "email": "rahul@gmail.com",
    "phone_number": "9876543210",
    "user_type": "agent",
    "is_verified": false
  }
}
```

✅Features Covered

- Strong Joi validation
- Secure password hashing
- Unique Agent ID generation
- Clean folder structure
- Production-ready logic

If you want: - OTP verification flow - Admin agent verification API - Login + JWT - Swagger docs

Tell me 👍