1. UserPreference Schema

```
import { Prop, Schema, SchemaFactory } from '@nestjs/mongoose';
import { Document } from 'mongoose';

export type UserPreferenceDocument = UserPreference & Document;

@Schema({ timestamps: true })
export class UserPreference {
  @Prop({ required: true })
  userId: string;

  @Prop({ required: true, match: /^[^\s@]+@[^\s@]+\.[^\s@]+$/ })
  email: string;

  @Prop({ required: true })
  preferences: {
    marketing: boolean;
    newsletter: boolean;
    updates: boolean;
    frequency: 'daily' | 'weekly' | 'monthly' | 'never';
    channels: { email: boolean; sms: boolean; push: boolean };
  };

  @Prop({ required: true })
  timezone: string;
}

export const UserPreferenceSchema = SchemaFactory.createForClass(UserPreference);
```

2. NotificationLog Schema

```
@Schema({ timestamps: true })
export class NotificationLog {
  @Prop({ required: true })
  userId: string;

  @Prop({ required: true })
  type: 'marketing' | 'newsletter' | 'updates';

  @Prop({ required: true })
  channel: 'email' | 'sms' | 'push';

  @Prop({ required: true, enum: ['pending', 'sent', 'failed'] })
  status: 'pending' | 'sent' | 'failed';

  @Prop()
  sentAt?: Date;

  @Prop()
  failureReason?: string;

  @Prop({ type: Object, default: {} })
  metadata: Record<string, any>;
}

export const NotificationLogSchema = SchemaFactory.createForClass(NotificationLog);
```

3. User Preferences

```
import { Body, Controller, Get, Post, Patch, Delete, Param } from '@nestjs/common';
import { UserPreferencesService } from './user-preferences.service';
import { CreateUserPreferenceDto } from './dto/create-user-preference.dto';

@Controller('preferences')
export class UserPreferencesController {
  constructor(private readonly service: UserPreferencesService) {}

  @Post()
  async create(@Body() dto: CreateUserPreferenceDto) {
    return this.service.create(dto);
  }

  @Get(':userId')
  async findOne(@Param('userId') userId: string) {
    return this.service.findOne(userId);
  }

  @Patch(':userId')
  async update(@Param('userId') userId: string, @Body() dto:
Partial<CreateUserPreferenceDto>) {
    return this.service.update(userId, dto);
  }

  @Delete(':userId')
  async remove(@Param('userId') userId: string) {
    return this.service.remove(userId);
  }
}
```