Yus Montessori School Management System - Project Setup

Project Structure

```
yus-montessori-school/
├── server.js                # Main server entry point
├── package.json             # Backend dependencies
├── .env.example             # Environment variables template
├── .gitignore               # Git ignore file
├── README.md                # Project documentation
├── config/
│   ├── database.js          # Database configuration
│   ├── email.js             # Email service configuration
│   └── ai.js                # AI service configuration
├── models/
│   ├── User.js              # User model (parents, teachers, admin)
│   ├── Student.js           # Student model
│   ├── Email.js             # Email tracking model
│   ├── Payment.js           # Payment tracking model
│   ├── Waitlist.js          # Waitlist model
│   ├── Expense.js           # Expense tracking model
│   └── Newsletter.js        # Newsletter model
├── routes/
│   ├── auth.js              # Authentication routes
│   ├── students.js          # Student management routes
│   ├── emails.js            # Email management routes
│   ├── payments.js          # Payment processing routes
│   ├── waitlist.js          # Waitlist management routes
│   ├── expenses.js          # Expense tracking routes
│   └── newsletters.js       # Newsletter routes
├── middleware/
│   ├── auth.js              # Authentication middleware
│   ├── validation.js        # Input validation middleware
│   └── errorHandler.js      # Error handling middleware
├── services/
│   ├── emailService.js      # Gmail/Email integration
│   ├── aiService.js         # OpenAI integration
│   ├── paymentService.js    # Stripe integration
│   ├── pdfService.js        # PDF generation service
│   └── notificationService.js # Notification service
├── utils/
│   ├── logger.js            # Winston logger configuration
```

```
│   ├── helpers.js          # Utility functions
│   └── constants.js        # Application constants
├── tests/
│   ├── unit/               # Unit tests
│   └── integration/        # Integration tests
└── client/                 # React frontend
    ├── package.json        # Frontend dependencies
    ├── public/
    ├── src/
    │   ├── components/     # Reusable React components
    │   ├── pages/          # Page components
    │   ├── services/       # API service calls
    │   ├── hooks/          # Custom React hooks
    │   ├── context/        # React context providers
    │   ├── utils/          # Frontend utilities
    │   └── styles/         # CSS/styling files
    └── build/              # Production build files
```

Quick Start Instructions

1. Clone and Setup

bash

```bash
# Clone the repository
git clone <your-repo-url>
cd yus-montessori-school

# Install all dependencies (backend and frontend)
npm run install:all

# Copy environment variables template
cp .env.example .env
```

2. Environment Configuration

Edit `.env` file with your credentials:

```
# Database
MONGODB_URI=your_mongodb_connection_string
DB_NAME=yus_montessori

# JWT
JWT_SECRET=your_jwt_secret_key
JWT_EXPIRES_IN=7d

# Email Configuration
GMAIL_CLIENT_ID=your_gmail_client_id
GMAIL_CLIENT_SECRET=your_gmail_client_secret
GMAIL_REFRESH_TOKEN=your_gmail_refresh_token
GMAIL_USER=school@yusmontessori.com

# AI Service
OPENAI_API_KEY=your_openai_api_key

# Payment Processing
STRIPE_SECRET_KEY=your_stripe_secret_key
STRIPE_WEBHOOK_SECRET=your_stripe_webhook_secret

# Application Settings
NODE_ENV=development
PORT=5000
FRONTEND_URL=http://localhost:3000
```

3. Development Setup

```bash
# Start both backend and frontend in development mode
npm run dev

# Or start them separately:
# Backend only:
npm run server:dev

# Frontend only (in another terminal):
npm run client:dev
```

4. Production Deployment

```bash
```

```
# Build the frontend
npm run build

# Start production server
npm start
```

Next Steps

1. **Database Setup**: Ensure MongoDB is running and accessible
2. **Gmail API Setup**: Configure Gmail API credentials in Google Cloud Console
3. **Stripe Setup**: Configure Stripe account for payment processing
4. **OpenAI Setup**: Set up OpenAI API key for AI features
5. **Domain Configuration**: Set up custom domain and SSL certificate
6. **Testing**: Run tests before deployment

Key Features to Implement

Phase 1 (MVP)

- [ ] User authentication (parents, teachers, admin)
- [ ] Student management dashboard
- [ ] Basic email integration and categorization
- [ ] Payment tracking system

Phase 2 (Enhanced Features)

- [ ] AI-powered email responses
- [ ] Automated newsletter generation
- [ ] Expense tracking and tax preparation
- [ ] Waitlist management with AI insights

Phase 3 (Advanced Automation)

- [ ] Calendar integration
- [ ] Predictive analytics
- [ ] Advanced reporting
- [ ] Mobile app companion

Development Guidelines

- Follow RESTful API design principles

- Use async/await for all asynchronous operations
- Implement comprehensive error handling
- Write tests for all new features
- Use ESLint for code consistency
- Implement proper logging for debugging
- Follow security best practices (OWASP guidelines)
- Ensure COPPA and FERPA compliance for student data