LMS Project Documentation

# LMS Project Documentation

## 1. Frontend (package.json)

### Project Overview

- Name: frontend

- Version: 0.0.0

- Type: module

### Scripts

- `dev`: Run development server using Vite

- `build`: Build the project using Vite

- `lint`: Run ESLint for JavaScript and JSX files

- `preview`: Preview the built project

### Key Dependencies

- React (v18.2.0)

- React Router (v6.10.0)

- Axios (v1.6.5)

- Chart.js (v4.4.0)

- CKEditor 5 (v40.2.0)

- PayPal React Components (v8.1.3)

- Zustand (v4.4.4) for state management

- Various UI libraries (Bootstrap, React Icons)

### Dev Dependencies

- Vite (v4.4.5)

- ESLint (v8.45.0)

- Prettier (v3.0.3)

## 2. Backend (requirements.txt)

### Key Dependencies

- Django (v4.2.7)

- Django REST Framework (v3.14.0)

- djangorestframework-simplejwt (v5.2.2)

- django-cors-headers (v3.14.0)

- django-storages (v1.12.3)

- django-ckeditor-5 (v0.4.4)

- drf-yasg (v1.21.7) for API documentation

- psycopg2 (v2.9.9) for PostgreSQL support

- gunicorn (v21.2.0) for production server

- boto3 (v1.20.26) for AWS services integration

## 3. Django Configuration Snippets (Snippets.txt)

### Installed Apps

```python

INSTALLED\_APPS = [

# ... other apps ...

'api',

'core',

'student',

'teacher',

'userauths',

'rest\_framework',

'rest\_framework\_simplejwt.token\_blacklist',

'corsheaders',

'anymail',

'storages',

'django\_ckeditor\_5',

]

```

### Middleware

```python

MIDDLEWARE = [

# ... other middleware ...

'corsheaders.middleware.CorsMiddleware',

]

```

### JWT Configuration

```python

SIMPLE\_JWT = {

'ACCESS\_TOKEN\_LIFETIME': timedelta(minutes=15),

'REFRESH\_TOKEN\_LIFETIME': timedelta(days=50),

'ROTATE\_REFRESH\_TOKENS': True,

'BLACKLIST\_AFTER\_ROTATION': True,

# ... other configurations ...

}

```

### CORS Configuration

```python

CORS\_ALLOW\_ALL\_ORIGINS = True

```

### API Documentation (drf-yasg)

```python

from drf\_yasg.views import get\_schema\_view

from drf\_yasg import openapi

schema\_view = get\_schema\_view(

openapi.Info(

title="E-commerce Backend APIs",

default\_version='v1',

description="This is the API documentation for Desphixs LMS project APIs",

# ... other configurations ...

),

public=True,

permission\_classes=(permissions.AllowAny,),

)

urlpatterns = [

path('swagger<format>/', schema\_view.without\_ui(cache\_timeout=0), name='schema-json'),

path('', schema\_view.with\_ui('swagger', cache\_timeout=0), name='schema-swagger-ui'),

path('redoc/', schema\_view.with\_ui('redoc', cache\_timeout=0), name='schema-redoc'),

# ... other URLs ...

]

```

### External API Integration

- User location API: `https://nominatim.openstreetmap.org/reverse?format=json&lat=<lat-here>&lon=<lon-here>`

### PayPal Integration (React.js)

```jsx

const initialOptions = {

clientId: PAYPAL\_CLIENT\_ID,

currency: "USD",

intent: "capture",

};

<PayPalScriptProvider options={initialOptions}>

<PayPalButtons

createOrder={(data, actions) => {

// ... order creation logic ...

}}

onApprove={(data, actions) => {

// ... payment approval logic ...

}}

/>

</PayPalScriptProvider>

```

## Project Structure

- Frontend: React-based SPA using Vite

- Backend: Django with Django REST Framework

- Authentication: JWT-based using djangorestframework-simplejwt

- Database: PostgreSQL (inferred from psycopg2 dependency)

- File Storage: Likely using AWS S3 (inferred from boto3 and django-storages)

- API Documentation: Swagger UI and ReDoc using drf-yasg

## Development Workflow

1. Set up virtual environment and install backend dependencies

2. Configure Django settings (database, CORS, JWT, etc.)

3. Develop Django apps (api, core, student, teacher, userauths)

4. Set up frontend project with Vite

5. Develop React components and integrate with backend APIs

6. Use PayPal integration for payment processing

7. Implement user location features using the provided API

## Deployment Considerations

- Use Gunicorn as the WSGI HTTP Server for Django

- Configure CORS settings appropriately for production

- Set up AWS S3 for file storage in production

- Ensure proper environment variable management for sensitive data

- Implement proper error handling and logging

## Security Notes

- JWT tokens are configured with a 15-minute lifetime for access tokens and 50 days for refresh tokens

- CORS is currently set to allow all origins, which should be restricted in production

- Ensure proper authentication and authorization checks in Django views

This documentation provides an overview of the project structure, key dependencies, and important configuration snippets. For detailed implementation, refer to the specific files and Django/React documentation.