

YMV Business System - 완전한 프로젝트 구조

📁 D:\ymv-business-system 프로젝트 구조

```
ymv-business-system/
├── .env                    # 환경 변수 (이미 있음)
├── requirements.txt        # Python 의존성 (이미 있음)
├── database/
│   └── init_db.sql        # DB 스키마 (이미 있음)
├── README.md              # 프로젝트 문서
├── .gitignore              # Git 제외 파일
├── app/
│   ├── main.py            # 메인 애플리케이션
│   ├── config/
│   │   ├── __init__.py
│   │   ├── database.py    # 데이터베이스 연결
│   │   ├── settings.py    # 시스템 설정
│   │   └── constants.py   # 상수 정의
│   ├── modules/
│   │   ├── __init__.py
│   │   ├── auth/
│   │   │   ├── __init__.py
│   │   │   ├── auth_service.py    # 인증 서비스
│   │   │   └── user_management.py  # 사용자 관리
│   │   ├── general_affairs/
│   │   │   ├── __init__.py
│   │   │   ├── purchase_management.py  # 구매 관리
│   │   │   └── expense_request.py      # 지출 요청서
│   │   ├── sales/
│   │   │   ├── __init__.py
│   │   │   ├── quotation_management.py  # 견적서 관리
│   │   │   └── customer_management.py    # 고객 관리
│   │   └── system/
│   │       ├── __init__.py
│   │       ├── product_management.py    # 제품 관리
│   │       └── code_management.py        # 코드 관리
│   └── shared/
│       ├── __init__.py
│       ├── database.py    # DB 연결 유틸
│       ├── utils.py       # 공통 유틸리티
│       ├── file_handler.py    # 파일 처리
│       ├── pdf_generator.py   # PDF 생성
│       ├── language_manager.py    # 다국어 관리
│       └── components.py      # UI 컴포넌트
└── pages/
```

```
| | |—— _init_.py
| | |—— dashboard.py      # 대시보드
| | |—— login.py          # 로그인 페이지
| | |—— purchase_list.py  # 구매 목록
| | |—— expense_list.py   # 지출 목록
| | |—— quotation_list.py # 견적서 목록
| | |—— customer_list.py  # 고객 목록
| | |—— product_list.py   # 제품 목록
| |—— static/
| |—— css/
| |—— style.css           # 커스텀 스타일
| |—— js/
| |—— images/
|—— uploads/              # 업로드 파일
|—— exports/              # 내보내기 파일
|—— logs/                 # 로그 파일
|—— fonts/                # 폰트 파일
|—— tests/                # 테스트 파일
|—— _init_.py
|—— test_auth.py
|—— test_purchase.py
|—— test_quotation.py
|—— test_utils.py
```

파일별 상세 내용

1. README.md

markdown

YMV Business Management System

베트남 소재 한국 기업을 위한 통합 비즈니스 관리 시스템

프로젝트 정보

- ****위치****: D:\ymv-business-system
- ****GitHub****: 연결됨
- ****Database****: Supabase 연결됨
- ****개발환경****: Python + Streamlit

주요 기능

- 구매 관리 (사무용품, 판매제품, 핫런너)
- 지출 요청서 관리
- 견적서 작성 및 관리 (PDF 출력)
- 고객 관리
- 제품 코드 관리 (CSV 업로드)
- 다국어 지원 (한국어/영어/베트남어)

기술 스택

- ****Frontend****: Streamlit
- ****Backend****: Python
- ****Database****: PostgreSQL (Supabase)
- ****Authentication****: bcrypt
- ****PDF Generation****: ReportLab
- ****File Processing****: pandas, openpyxl

설치 및 실행

환경 설정

```
```bash
```

#### # 가상환경 생성

```
python -m venv venv
```

```
venv\Scripts\activate # Windows
```

#### # 의존성 설치

```
pip install -r requirements.txt
```

## 환경 변수 확인

`.env` 파일에서 Supabase 연결 정보 확인:

- SUPABASE\_URL
- SUPABASE\_KEY
- DB\_PASSWORD

## 데이터베이스 초기화

Supabase SQL Editor에서 `database/init_db.sql` 실행

## 애플리케이션 실행

```
bash

streamlit run app/main.py
```

## 기본 계정

- Username: Master
- Password: 1023

## 개발 진행 상황

- ☒ 프로젝트 구조 설계
- ☒ 데이터베이스 스키마 완성
- ☒ 환경 설정 완료
- ☐ 인증 시스템 구현
- ☐ 구매 관리 모듈
- ☐ 견적서 관리 모듈
- ☐ PDF 생성 기능
- ☐ CSV 업로드/다운로드

## 연락처

- 개발자: YMV팀
- 이메일: [dev@ymv.com](mailto:dev@ymv.com)

### 2. .gitignore

``gitignore

# Python

\_\_pycache\_\_/

\*.py[cod]

\*\$py.class

\*.so

.Python

build/

develop-eggs/

dist/

downloads/

eggs/

.eggs/

lib/

lib64/

parts/

sdist/

var/

wheels/

\*.egg-info/

.installed.cfg

\*.egg

# Virtual Environment

venv/

env/

ENV/

# Environment Variables

.env

.env.local

.env.production

.env.staging

# IDE

.vscode/

.idea/

\*.swp

\*.swo

\*~

# OS

.DS\_Store

Thumbs.db

# Application Specific

uploads/\*

!uploads/.gitkeep

exports/\*

!exports/.gitkeep

logs/\*

!logs/.gitkeep

# Database

\*.db

\*.sqlite3

# Temporary files

tmp/

temp/

\*.tmp

# Backup files

\*.bak

\*.backup

# PDF output

outputs/\*.pdf

# Node modules (if any)

node\_modules/

### 3. app/config/database.py

python

```

app/config/database.py
import os
import psycopg2
from psycopg2.extras import RealDictCursor
from contextlib import contextmanager
import streamlit as st
from typing import List, Dict, Any, Optional

class DatabaseConnection:
 """Supabase PostgreSQL 데이터베이스 연결 클래스"""

 def __init__(self):
 self.connection_params = {
 'host': os.getenv('DB_HOST'),
 'port': int(os.getenv('DB_PORT', 5432)),
 'database': os.getenv('DB_NAME'),
 'user': os.getenv('DB_USER'),
 'password': os.getenv('DB_PASSWORD'),
 'sslmode': 'require'
 }

 @contextmanager
 def get_connection(self):
 """데이터베이스 연결 컨텍스트 매니저"""
 conn = None
 try:
 conn = psycopg2.connect(**self.connection_params)
 yield conn
 except Exception as e:
 if conn:
 conn.rollback()
 st.error(f"데이터베이스 연결 오류: {str(e)}")
 raise
 finally:
 if conn:
 conn.close()

 def execute_query(self, query: str, params: tuple = None) -> List[Dict[str, Any]]:
 """쿼리 실행 및 결과 반환"""
 try:
 with self.get_connection() as conn:
 with conn.cursor(cursor_factory=RealDictCursor) as cursor:
 cursor.execute(query, params)

 # SELECT 쿼리인 경우 결과 반환
 if cursor.description:

```

```

 return [dict(row) for row in cursor.fetchall()]
 else:
 conn.commit()
 return []

except Exception as e:
 st.error(f"쿼리 실행 오류: {str(e)}")
 return []

def execute_transaction(self, queries: List[tuple]) -> bool:
 """트랜잭션으로 여러 쿼리 실행"""
 try:
 with self.get_connection() as conn:
 with conn.cursor() as cursor:
 for query, params in queries:
 cursor.execute(query, params)
 conn.commit()
 return True

 except Exception as e:
 st.error(f"트랜잭션 실행 오류: {str(e)}")
 return False

def test_connection(self) -> bool:
 """데이터베이스 연결 테스트"""
 try:
 with self.get_connection() as conn:
 with conn.cursor() as cursor:
 cursor.execute("SELECT 1")
 return True

 except Exception as e:
 st.error(f"데이터베이스 연결 테스트 실패: {str(e)}")
 return False

전역 데이터베이스 인스턴스
@st.cache_resource
def get_database():
 """캐시된 데이터베이스 연결 인스턴스 반환"""
 return DatabaseConnection()

```

#### 4. app/config/settings.py

```
python
```



```
app/config/settings.py
```

```
import os
```

```
from typing import Dict, Any
```

```
class Settings:
```

```
 """애플리케이션 설정 클래스"""
```

```
 # 애플리케이션 정보
```

```
 APP_NAME = os.getenv('APP_NAME', 'YMV Business Management System')
```

```
 APP_VERSION = os.getenv('APP_VERSION', '1.0.0')
```

```
 DEBUG = os.getenv('DEBUG', 'False').lower() == 'true'
```

```
 # 보안 설정
```

```
 SECRET_KEY = os.getenv('SECRET_KEY')
```

```
 JWT_SECRET_KEY = os.getenv('JWT_SECRET_KEY')
```

```
 JWT_ALGORITHM = os.getenv('JWT_ALGORITHM', 'HS256')
```

```
 JWT_EXPIRATION_HOURS = int(os.getenv('JWT_EXPIRATION_HOURS', 24))
```

```
 # 파일 업로드 설정
```

```
 MAX_FILE_SIZE = int(os.getenv('MAX_FILE_SIZE', 10485760)) # 10MB
```

```
 ALLOWED_EXTENSIONS = os.getenv('ALLOWED_EXTENSIONS', 'csv,xlsx,xls,pdf,png,jpg,jpeg').split(',')
```

```
 # 디렉토리 설정
```

```
 UPLOAD_DIR = 'uploads'
```

```
 EXPORT_DIR = 'exports'
```

```
 LOG_DIR = 'logs'
```

```
 FONT_DIR = 'fonts'
```

```
 # 페이지네이션
```

```
 DEFAULT_PAGE_SIZE = 20
```

```
 MAX_PAGE_SIZE = 100
```

```
 # 다국어 설정
```

```
 SUPPORTED_LANGUAGES = ['ko', 'en', 'vn']
```

```
 DEFAULT_LANGUAGE = 'ko'
```

```
 # 통화 설정
```

```
 SUPPORTED_CURRENCIES = ['USD', 'VND', 'KRW', 'CNY', 'THB', 'JPY', 'EUR']
```

```
 DEFAULT_CURRENCY = 'USD'
```

```
 # 이메일 설정 (선택사항)
```

```
 EMAIL_HOST = os.getenv('EMAIL_HOST')
```

```
 EMAIL_PORT = int(os.getenv('EMAIL_PORT', 587))
```

```
 EMAIL_USER = os.getenv('EMAIL_USER')
```

```
 EMAIL_PASSWORD = os.getenv('EMAIL_PASSWORD')
```

```

@classmethod
def get_all_settings(cls) -> Dict[str, Any]:
 """모든 설정을 딕셔너리로 반환"""
 return {
 'app_name': cls.APP_NAME,
 'app_version': cls.APP_VERSION,
 'debug': cls.DEBUG,
 'max_file_size': cls.MAX_FILE_SIZE,
 'allowed_extensions': cls.ALLOWED_EXTENSIONS,
 'supported_languages': cls.SUPPORTED_LANGUAGES,
 'default_language': cls.DEFAULT_LANGUAGE,
 'supported_currencies': cls.SUPPORTED_CURRENCIES,
 'default_currency': cls.DEFAULT_CURRENCY,
 }

@classmethod
def validate_settings(cls) -> List[str]:
 """설정 유효성 검사"""
 errors = []

 if not cls.SECRET_KEY:
 errors.append("SECRET_KEY가 설정되지 않았습니다.")

 if not cls.JWT_SECRET_KEY:
 errors.append("JWT_SECRET_KEY가 설정되지 않았습니다.")

 return errors

전역 설정 인스턴스
settings = Settings()

```

## 5. app/config/constants.py

```
python
```

```
app/config/constants.py
```

```
문서 타입 코드
```

```
DOCUMENT_TYPES = {
 'QUOTATION': 'Q',
 'PURCHASE': 'P',
 'EXPENSE': 'E',
 'ORDER': 'O'
}
```

```
상태 코드
```

```
STATUS_CODES = {
 'DRAFT': 'draft',
 'PENDING': 'pending',
 'APPROVED': 'approved',
 'REJECTED': 'rejected',
 'SENT': 'sent',
 'ACCEPTED': 'accepted',
 'EXPIRED': 'expired',
 'CANCELLED': 'cancelled',
 'PAID': 'paid',
 'CONFIRMED': 'confirmed',
 'COMPLETED': 'completed'
}
```

```
사용자 역할
```

```
USER_ROLES = {
 'MASTER': 'master',
 'ADMIN': 'admin',
 'MANAGER': 'manager',
 'USER': 'user'
}
```

```
권한 타입
```

```
PERMISSION_TYPES = {
 'READ': 'read',
 'WRITE': 'write',
 'DELETE': 'delete',
 'ADMIN': 'admin'
}
```

```
모듈명
```

```
MODULE_NAMES = {
 'GENERAL_AFFAIRS': 'general_affairs',
 'SALES': 'sales',
 'SYSTEM': 'system',
}
```

```
'ADMIN': 'admin'
}
```

*# 색상 코드*

```
COLORS = {
 'PRIMARY': '#2c3e50',
 'SECONDARY': '#3498db',
 'SUCCESS': '#27ae60',
 'WARNING': '#f39c12',
 'DANGER': '#e74c3c',
 'INFO': '#17a2b8',
 'LIGHT': '#f8f9fa',
 'DARK': '#343a40'
}
```

*# 파일 타입*

```
FILE_TYPES = {
 'CSV': 'csv',
 'EXCEL': 'xlsx',
 'PDF': 'pdf',
 'IMAGE': 'png,jpg,jpeg'
}
```

*# 구매 카테고리*

```
PURCHASE_CATEGORIES = {
 'OFFICE': 'office',
 'FIELD': 'field',
 'OTHER': 'other'
}
```

*# 지출 유형*

```
EXPENSE_TYPES = {
 'BUSINESS_TRIP': 'business_trip',
 'OFFICE_SUPPLY': 'office_supply',
 'ENTERTAINMENT': 'entertainment',
 'TRAINING': 'training',
 'TRANSPORT': 'transport',
 'MEAL': 'meal',
 'COMMUNICATION': 'communication',
 'EQUIPMENT': 'equipment',
 'MAINTENANCE': 'maintenance',
 'MARKETING': 'marketing',
 'OTHER': 'other'
}
```

*# 결제 방법*

```
PAYMENT_METHODS = {
```

```
'CASH': 'cash',
'BANK_TRANSFER': 'bank_transfer',
'CORPORATE_CARD': 'corporate_card',
'CHECK': 'check',
'OTHER': 'other'
}
```

*# 통화 기호*

```
CURRENCY_SYMBOLS = {
 'USD': '$',
 'VND': '₫',
 'KRW': '₩',
 'CNY': '¥',
 'THB': '฿',
 'JPY': '¥',
 'EUR': '€'
}
```

*# 날짜 형식*

```
DATE_FORMATS = {
 'DEFAULT': '%Y-%m-%d',
 'DISPLAY': '%Y년 %m월 %d일',
 'FILENAME': '%Y%m%d_%H%M%S'
}
```

*# 언어 코드 매핑*

```
LANGUAGE_MAPPING = {
 'ko': '한국어',
 'en': 'English',
 'vn': 'Tiếng Việt'
}
```

## 6. app/main.py

python

```
app/main.py
import streamlit as st
import os
import sys
from pathlib import Path

프로젝트 루트 디렉토리를 Python 경로에 추가
project_root = Path(__file__).parent.parent
sys.path.append(str(project_root))

from dotenv import load_dotenv
from app.config.database import get_database
from app.config.settings import settings
from app.shared.utils import create_directories

환경 변수 로드
load_dotenv()

페이지 설정
st.set_page_config(
 page_title=settings.APP_NAME,
 page_icon="🏠",
 layout="wide",
 initial_sidebar_state="expanded"
)

def initialize_app():
 """애플리케이션 초기화"""
 # 필요한 디렉토리 생성
 create_directories()

 # 데이터베이스 연결 테스트
 db = get_database()
 if not db.test_connection():
 st.error("데이터베이스 연결에 실패했습니다. 환경 설정을 확인해주세요.")
 st.stop()

 # 세션 상태 초기화
 if 'authenticated' not in st.session_state:
 st.session_state.authenticated = False
 if 'user_info' not in st.session_state:
 st.session_state.user_info = None
 if 'current_page' not in st.session_state:
 st.session_state.current_page = 'login'

def main():
```

```

"""메인 애플리케이션"""
애플리케이션 초기화
initialize_app()

인증 확인
if not st.session_state.authenticated:
 show_login_page()
else:
 show_main_app()

def show_login_page():
 """로그인 페이지"""
 st.title("🏠 YMV 관리 시스템")
 st.markdown("---")

 # 로그인 폼
 with st.container():
 col1, col2, col3 = st.columns([1, 2, 1])

 with col2:
 st.subheader("로그인")

 with st.form("login_form"):
 username = st.text_input("사용자명")
 password = st.text_input("비밀번호", type="password")
 login_button = st.form_submit_button("로그인", use_container_width=True)

 if login_button:
 if authenticate_user(username, password):
 st.success("로그인 성공!")
 st.rerun()
 else:
 st.error("사용자명 또는 비밀번호가 잘못되었습니다.")

 # 기본 계정 정보 표시
 with st.expander("기본 계정 정보"):
 st.info("""
 Master 계정
 - 사용자명: Master
 - 비밀번호: 1023
 """)

def authenticate_user(username: str, password: str) -> bool:
 """사용자 인증"""
 # TODO: 실제 인증 로직 구현
 if username == "Master" and password == "1023":
 st.session_state.authenticated = True

```

```

st.session_state.user_info = {
 'user_id': 1,
 'username': 'Master',
 'full_name': 'System Administrator',
 'department': 'IT',
 'position': 'Administrator'
}

return True
return False

def show_main_app():
 """메인 애플리케이션 화면"""
 # 사이드바 메뉴
 with st.sidebar:
 st.title("📄 메뉴")

 # 사용자 정보
 user_info = st.session_state.user_info
 st.write(f"👤 **{user_info['full_name']}**")
 st.write(f"🏢 {user_info['department']}")
 st.divider()

 # 메뉴 항목
 menu_items = {
 "dashboard": "🏠 대시보드",
 "purchase": "📦 구매 관리",
 "expense": "💰 지출 요청서",
 "quotation": "📄 견적서 관리",
 "customer": "👥 고객 관리",
 "product": "📦 제품 관리",
 "system": "⚙️ 시스템 설정"
 }

 selected_page = None
 for key, label in menu_items.items():
 if st.button(label, use_container_width=True, key=f"menu_{key}"):
 selected_page = key

 # 로그아웃
 st.divider()
 if st.button("🚪 로그아웃", use_container_width=True):
 st.session_state.authenticated = False
 st.session_state.user_info = None
 st.rerun()

 # 메인 콘텐츠 영역
 if selected_page:

```



```
st.session_state.current_page = selected_page
```

```
current_page = st.session_state.get('current_page', 'dashboard')
```

```
페이지 라우팅
```

```
if current_page == 'dashboard':
```

```
 show_dashboard()
```

```
elif current_page == 'purchase':
```

```
 show_purchase_page()
```

```
elif current_page == 'expense':
```

```
 show_expense_page()
```

```
elif current_page == 'quotation':
```

```
 show_quotation_page()
```

```
elif current_page == 'customer':
```

```
 show_customer_page()
```

```
elif current_page == 'product':
```

```
 show_product_page()
```

```
elif current_page == 'system':
```

```
 show_system_page()
```

```
else:
```

```
 show_dashboard()
```

```
def show_dashboard():
```

```
 """대시보드 페이지"""
```

```
 st.title("📊 대시보드")
```

```
 st.markdown("----")
```

```
통계 카드
```

```
col1, col2, col3, col4 = st.columns(4)
```

```
with col1:
```

```
 st.metric("총 견적서", "24", "3")
```

```
with col2:
```

```
 st.metric("대기중인 구매", "7", "2")
```

```
with col3:
```

```
 st.metric("이번달 지출", "$12,450", "5.2%")
```

```
with col4:
```

```
 st.metric("활성 고객", "156", "8")
```

```
최근 활동
```

```
st.subheader("📝 최근 활동")
```

```
임시 데이터로 차트 표시
```

```
import pandas as pd
```

```
import plotly.express as px
```

```
월별 매출 차트
```

```

chart_data = pd.DataFrame({
 '월': ['1월', '2월', '3월', '4월', '5월', '6월'],
 '매출': [15000, 18000, 22000, 19000, 25000, 28000],
 '지출': [8000, 9500, 11000, 10200, 12500, 13200]
})

fig = px.line(chart_data, x='월', y=['매출', '지출'],
 title="월별 매출/지출 현황")
st.plotly_chart(fig, use_container_width=True)

def show_purchase_page():
 """구매 관리 페이지"""
 st.title("📦 구매 관리")
 st.info("구매 관리 기능을 개발 중입니다.")

def show_expense_page():
 """지출 요청서 페이지"""
 st.title("💰 지출 요청서")
 st.info("지출 요청서 기능을 개발 중입니다.")

def show_quotation_page():
 """견적서 관리 페이지"""
 st.title("📄 견적서 관리")
 st.info("견적서 관리 기능을 개발 중입니다.")

def show_customer_page():
 """고객 관리 페이지"""
 st.title("👤 고객 관리")
 st.info("고객 관리 기능을 개발 중입니다.")

def show_product_page():
 """제품 관리 페이지"""
 st.title("📦 제품 관리")
 st.info("제품 관리 기능을 개발 중입니다.")

def show_system_page():
 """시스템 설정 페이지"""
 st.title("⚙️ 시스템 설정")
 st.info("시스템 설정 기능을 개발 중입니다.")

if __name__ == "__main__":
 main()

```

## 7. app/shared/utils.py

```
python
```

```

app/shared/utils.py
import os
from pathlib import Path
from app.config.settings import settings

def create_directories():
 """필요한 디렉토리 생성"""
 directories = [
 settings.UPLOAD_DIR,
 settings.EXPORT_DIR,
 settings.LOG_DIR,
 settings.FONT_DIR
]

 for directory in directories:
 Path(directory).mkdir(exist_ok=True)
 # .gitkeep 파일 생성 (빈 디렉토리도 Git에서 추적)
 gitkeep_file = Path(directory) / '.gitkeep'
 if not gitkeep_file.exists():
 gitkeep_file.touch()

def format_currency(amount: float, currency: str = 'USD') -> str:
 """통화 포매팅"""
 from app.config.constants import CURRENCY_SYMBOLS

 symbol = CURRENCY_SYMBOLS.get(currency, currency)

 if currency in ['VND', 'KRW']:
 return f"{symbol} {int(amount):,}"
 else:
 return f"{symbol} {amount:,.2f}"

def generate_document_number(doc_type: str, date=None) -> str:
 """문서 번호 생성"""
 from datetime import datetime
 from app.config.constants import DOCUMENT_TYPES

 if date is None:
 date = datetime.now()

 type_code = DOCUMENT_TYPES.get(doc_type, doc_type)
 date_str = date.strftime("%y%m%d")

 # TODO: 데이터베이스에서 실제 카운트 조회
 count = 1

```

```
return f"Y{type_code}{date_str}-{count:03d}"
```

## 8. app/static/css/style.css

CSS

```
/* app/static/css/style.css */
```

```
/* 기본 색상 변수 */
```

```
:root {
 --primary-color: #2c3e50;
 --secondary-color: #3498db;
 --success-color: #27ae60;
 --warning-color: #f39c12;
 --danger-color: #e74c3c;
 --light-gray: #ecf0f1;
 --dark-gray: #95a5a6;
}
```

```
/* 전체 앱 스타일 */
```

```
.main > div {
 padding: 1rem;
}
```

```
/* 카드 스타일 */
```

```
.metric-card {
 background: white;
 padding: 1rem;
 border-radius: 0.5rem;
 box-shadow: 0 2px 4px rgba(0,0,0,0.1);
 border-left: 4px solid var(--primary-color);
}
```

```
/* 버튼 스타일 */
```

```
.stButton > button {
 width: 100%;
 border-radius: 0.5rem;
 font-weight: 500;
}
```

```
/* 사이드바 스타일 */
```

```
.css-1d391kg {
 background-color: var(--light-gray);
}
```

```
/* 데이터 테이블 스타일 */
```

```
.dataframe {
 font-size: 0.9rem;
}
```

```
.dataframe th {
 background-color: var(--primary-color) !important;
```

```

 color: white !important;
}

/* 상태 배지 스타일 */
.status-badge {
 display: inline-block;
 padding: 0.25rem 0.5rem;
 border-radius: 0.25rem;
 font-size: 0.8rem;
 font-weight: 500;
}

.status-pending {
 background-color: var(--warning-color);
 color: white;
}

.status-approved {
 background-color: var(--success-color);
 color: white;
}

.status-rejected {
 background-color: var(--danger-color);
 color: white;
}

/* 반응형 디자인 */
@media (max-width: 768px) {
 .main > div {
 padding: 0.5rem;
 }

 .metric-card {
 margin-bottom: 1rem;
 }
}

```

## 프로젝트 시작 가이드

### 1. 현재 상태 확인

D:\ymv-business-system 폴더에서:

```
bash
```

```
현재 파일들 확인
```

```
dir
```

```
Git 상태 확인
```

```
git status
```

```
Supabase 연결 확인
```

```
.env 파일에서 연결 정보 확인
```

## 2. 누락된 폴더 및 파일 생성

```
bash
```

```
폴더 생성
```

```
mkdir app\config app\modules app\shared app\pages app\static\css uploads exports logs fonts tests
```

```
__init__.py 파일 생성
```

```
echo. > app__init__.py
```

```
echo. > app\config__init__.py
```

```
echo. > app\modules__init__.py
```

```
echo. > app\shared__init__.py
```

```
echo. > app\pages__init__.py
```

## 3. 파일 생성

위의 내용들을 각각 해당 경로에 파일로 생성

## 4. 애플리케이션 실행

```
bash
```

```
가상환경 활성화 (있다면)
```

```
venv\Scripts\activate
```

```
의존성 설치 확인
```

```
pip install -r requirements.txt
```

```
애플리케이션 실행
```

```
streamlit run app\main.py
```

이제 완전한 프로젝트 구조가 준비되었습니다! 각 파일을 생성하고 단계별로 기능을 구현해 나가시면 됩니다.