HostelRec Project

Complete Source Code Documentation

Total files: 102
Generated: 1765226221.7225232

Table of Contents

39. backend\app\services\admin_service.py
40. backend\app\services\booking_service.py
41. backend\app\services\mfa_service.py
42. backend\app\services\password_reset_service.py
43. backend\app\services\session_service.py
44. backend\app\services\stats_service.py
45. backend\AUTHENTICATION_SYSTEM.md
46. backend\requirements.txt
47. DATABASE_RELATIONSHIPS.md
48. docker-compose.yml
49. frontend\package.json
50. frontend\postcss.config.js
51. frontend\src\App.tsx
52. frontend\src\components\BookingWizard.tsx
53. frontend\src\components\ChatWidget.tsx
54. frontend\src\components\FloorPlan.tsx
55. frontend\src\components\GanttTimeline.tsx
56. frontend\src\components\HeroSection.tsx
57. frontend\src\components\KpiCard.tsx
58. frontend\src\components\LanguageToggle.tsx
59. frontend\src\components\Navbar.tsx
60. frontend\src\components\RevenueChart.tsx
61. frontend\src\components\RoomCard.tsx
62. frontend\src\hooks\useAuthStore.ts
63. frontend\src\i18n.ts
64. frontend\src\index.css
65. frontend\src\layouts\AdminLayout.tsx
66. frontend\src\locales\en.json
67. frontend\src\locales\ru.json
68. frontend\src\main.tsx
69. frontend\src\pages\AdminDashboard.tsx
70. frontend\src\pages\AuditLogPage.tsx
71. frontend\src\pages\CheckInOutPage.tsx
72. frontend\src\pages\CleanerPage.tsx
73. frontend\src\pages\ContentManagementPage.tsx
74. frontend\src\pages\CustomerPage.tsx
75. frontend\src\pages\ForgotPasswordPage.tsx
76. frontend\src\pages\LoginPage.tsx
77. frontend\src\pages\MenuPage.tsx
78. frontend\src\pages\MFAPage.tsx
79. frontend\src\pages\OrdersPage.tsx

80. frontend\src\pages\PasswordChangePage.tsx
81. frontend\src\pages\PublicHome.tsx
82. frontend\src\pages\RoomDetailPage.tsx
83. frontend\src\pages\SecurityPage.tsx
84. frontend\src\pages\SessionsPage.tsx
85. frontend\src\pages\SignupPage.tsx
86. frontend\src\pages\StaffManagementPage.tsx
87. frontend\src\services\api.ts
88. frontend\src\types\babel__core\index.d.ts
89. frontend\src\types\framer-motion.d.ts
90. frontend\src\types\jsx.d.ts
91. frontend\src\types\react-i18next.d.ts
92. frontend\src\types\react-jsx-runtime.d.ts
93. frontend\src\types\react-router-dom.d.ts
94. frontend\src\types\react.d.ts
95. frontend\src\vite-env.d.ts
96. frontend\tailwind.config.js
97. frontend\tsconfig.json
98. frontend\tsconfig.node.json
99. frontend\vite.config.ts
100. generate_code_pdf.py
101. README.md
102. sql\README.md

File: backend\app__init__.py

Full path: C:\Users\itodo\Hostelrec\backend\app__init__.py

File: backend\app\api__init__.py

Full path: C:\Users\itodo\Hostelrec\backend\app\api__init__.py

```
"""API routers package."""
```

File: backend\app\api\routes__init__.py

Full path: C:\Users\itodo\Hostelrec\backend\app\api\routes__init__.py

```
from fastapi import APIRouter

from . import admin, auth, bookings, customer, orders, stats, staff, chat, content, guests, payments

api_router = APIRouter()

api_router.include_router(auth.router, prefix="/auth", tags=["auth"])
api_router.include_router(bookings.router, prefix="/bookings", tags=["bookings"])
api_router.include_router(customer.router, prefix="/customer", tags=["customer"])
api_router.include_router(orders.router, prefix="/orders", tags=["orders"])
api_router.include_router(stats.router, prefix="/stats", tags=["stats"])
api_router.include_router(admin.router, prefix="/admin", tags=["admin"])
api_router.include_router(staff.router, prefix="/staff", tags=["staff"])
api_router.include_router(chat.router, prefix="/chat", tags=["chat"])
api_router.include_router(content.router, prefix="/content", tags=["content"])
api_router.include_router(guests.router, prefix="/guests", tags=["guests"])
api_router.include_router(payments.router, prefix="/payments", tags=["payments"])
```

File: backend\app\api\routes\admin.py

Full path: C:\Users\itodo\Hostelrec\backend\app\api\routes\admin.py

```
from fastapi import APIRouter, Depends, HTTPException
from sqlalchemy.ext.asyncio import AsyncSession

from app.core.config import get_settings
from app.core.security import require_role
from app.db.session import get_db
from app.services import admin_service

router = APIRouter()


@router.get("/logs")
async def audit_logs(
    session: AsyncSession = Depends(get_db),
    _manager=Depends(require_role("manager")),
):
    return await admin_service.fetch_audit_logs(session)


@router.get("/security/api-keys")
async def reveal_api_keys(
    secret_phrase: str,
    session: AsyncSession = Depends(get_db),
    _manager=Depends(require_role("manager")),
):
    settings = get_settings()
    if secret_phrase != settings.super_secret_phrase:
        raise HTTPException(status_code=403, detail="Invalid secret phrase")
    return await admin_service.fetch_api_keys(session, super_secret=secret_phrase)
```

File: backend\app\api\routes\auth.py

Full path: C:\Users\itodo\Hostelrec\backend\app\api\routes\auth.py

```
"""
[illegible].

[illegible]:
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
"""
from datetime import datetime, timezone
from fastapi import APIRouter, Depends, HTTPException, status, Request
from fastapi.security import OAuth2PasswordRequestForm
from sqlalchemy import select
from sqlalchemy.ext.asyncio import AsyncSession

from app.core.security import create_access_token, get_password_hash, verify_password, get_current_user, Tok
from app.db.session import get_db
from app.models import User
from app.schemas.auth import (
    TokenResponse, SignupRequest, PasswordChangeRequest,
    MFAEnableRequest, MFAVerifyRequest, MFADisableRequest,
    ForgotPasswordRequest, ResetPasswordRequest
)
from app.services.session_service import (
    create_session, update_session_activity, end_session, end_all_user_sessions,
    get_active_sessions_count
)
from app.services.mfa_service import create_mfa_code, verify_mfa_code, enable_mfa, disable_mfa
from app.services.password_reset_service import (
    create_password_reset_code, verify_password_reset_code
)
from app.models.session import UserSession

router = APIRouter()


@router.post("/token", response_model=TokenResponse)
async def login(
    request: Request,
    form_data: OAuth2PasswordRequestForm = Depends(),
    db: AsyncSession = Depends(get_db),
) -&gt; TokenResponse:
    """
    [illegible].

    [illegible], [illegible]
    [illegible].
    [illegible] (MFA).

    Args:
        form_data: [illegible] username [illegible] password
        db: [illegible]
        request: HTTP [illegible] ([illegible] IP [illegible])

    Returns:
        TokenResponse: [illegible]

    Raises:
        HTTPException: [illegible]
    """
    # [illegible] username
    result = await db.execute(select(User).where(User.username == form_data.username))
    user: User | None = result.scalar_one_or_none()

    # [illegible]
    if not user or not verify_password(form_data.password, user.password):
        raise HTTPException(
```

```
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Invalid credentials"
        )

    # ██████████, ██████████ ██ MFA
    # ████ MFA ████████ ▪ ███ ██ ██████████████, ███████ ██████████ MFA
    if hasattr(user, 'mfa_enabled') and user.mfa_enabled:
        # ▪ ████████ ██████████ ██████ ███████ ████ ████████ ████ MFA
        # ███ ██████████, ████ MFA ████████, ███████ ███ ██████ ██████████ endpoint
        pass

    # ███████ ██████ ████████████████
    token = create_access_token(subject=user.username, role=user.role)

    # ████████ IP ██████ ▪ User-Agent ██ ████████
    ip_address = None
    user_agent = None
    if request:
        ip_address = request.client.host if request.client else None
        user_agent = request.headers.get("user-agent")

    # ███████ ███████ ██████████████
    await create_session(
        db=db,
        user_id=user.id,
        token=token,
        expires_minutes=60,
        ip_address=ip_address,
        user_agent=user_agent,
    )

    mfa_required = hasattr(user, 'mfa_enabled') and user.mfa_enabled

    return TokenResponse(
        access_token=token,
        role=user.role,
        mfa_required=mfa_required
    )


@router.post("/signup", response_model=TokenResponse, status_code=status.HTTP_201_CREATED)
async def signup(
    request: Request,
    payload: SignupRequest,
    db: AsyncSession = Depends(get_db),
) -&gt; TokenResponse:
    """
    ████████████ ███████ ██████████████.

    ███████ ███████ ██████████████ ▪ ██████ 'customer' ██ ██████████.
    ████████████ email ███ ████████████████ ███ ██████ (username).

    Args:
        payload: ███████ ████████████ (email, full_name, password)
        db: ███████ ████ ███████
        request: HTTP ███████

    Returns:
        TokenResponse: ██████ ████████ ███ ███████ ██████████████

    Raises:
        HTTPException: ████ ██████████████ ▪ ██████ email ███ ████████████
    """
    # ██████████, ████████████ ██ ██████████████ ▪ ██████ email (username)
    result = await db.execute(select(User).where(User.username == payload.email))
    if result.scalar_one_or_none():
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="Username already exists"
        )

    # ███████ ███████ ██████████████ ▪ ██████████████ ████████
    user = User(
```

```
        username=payload.email,
        password=get_password_hash(payload.password),
        role="customer",
        mfa_enabled=False,
    )
    db.add(user)
    await db.commit()
    await db.refresh(user)

    # ████████ ██████ ■ ███████
    token = create_access_token(subject=user.username, role=user.role)

    ip_address = request.client.host if request.client else None
    user_agent = request.headers.get("user-agent")

    await create_session(
        db=db,
        user_id=user.id,
        token=token,
        expires_minutes=60,
        ip_address=ip_address,
        user_agent=user_agent,
    )

    return TokenResponse(access_token=token, role=user.role)


@router.post("/change-password", status_code=status.HTTP_200_OK)
async def change_password(
    payload: PasswordChangeRequest,
    db: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; dict:
    """
    ██████ ███████ ████████████████.

    ██████████ ██████████████ ██████████ █████ ███████ ■ ██████████ ████████ ███████.

    Args:
        payload: ███████ ■ ██████ ███████
        db: ███████ █████ ████████
        current: ████████ ██████████████████████ ██████████████

    Returns:
        dict: ██████████ ██ ██████████ ██████ ███████

    Raises:
        HTTPException: █████ ███████ ███████ ████████
    """
    # ██████ ███████████████
    result = await db.execute(select(User).where(User.username == current.sub))
    user = result.scalar_one_or_none()

    if not user:
        raise HTTPException(
            status_code=status.HTTP_404_NOT_FOUND,
            detail="User not found"
        )

    # ██████████ ███████ ███████
    if not verify_password(payload.old_password, user.password):
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="Invalid old password"
        )

    # █████████ ███████
    user.password = get_password_hash(payload.new_password)
    await db.commit()

    # ██████████ ████ ██████████ ███████ ██████████████ (██████████ ██████████ █████)
    await end_all_user_sessions(db, user.id)
```

```
    return {"message": "Password changed successfully. Please login again."}


@router.post("/logout", status_code=status.HTTP_200_OK)
async def logout(
    db: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; dict:
    """
    ██████ ██ ████████ (██████████ ███████).

    ██████████████ ███ █████████ ███████ ██████████████.
    ■ ██████████ ██████ ██████████ ███████ ████████ ███████ ██ ███████.

    Args:
        db: ███████ █████ ███████
        current: ████████ ██████████████████████ ██████████████

    Returns:
        dict: ██████████ ██ ████████ ███████
    """
    result = await db.execute(select(User).where(User.username == current.sub))
    user = result.scalar_one_or_none()

    if user:
        await end_all_user_sessions(db, user.id)

    return {"message": "Logged out successfully"}


@router.post("/mfa/enable", status_code=status.HTTP_200_OK)
async def enable_mfa_endpoint(
    payload: MFAEnableRequest,
    db: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; dict:
    """
    ████████ ████████████████ ████████████████ ████ ██████████████.

    Args:
        payload: ████████████████ ███████
        db: ███████ █████ ███████
        current: ████████ ██████████████████████ ██████████████

    Returns:
        dict: ██████████ █████ MFA (■ ████████ ██████████ ██████████████ ██ email)
    """
    result = await db.execute(select(User).where(User.username == current.sub))
    user = result.scalar_one_or_none()

    if not user:
        raise HTTPException(
            status_code=status.HTTP_404_NOT_FOUND,
            detail="User not found"
        )

    # ██████████ ███████
    if not verify_password(payload.password, user.password):
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="Invalid password"
        )

    # ████████ MFA
    secret = await enable_mfa(db, user.id)

    # ████████ ██████████████ (█████ ████ ████ ■ ███████)
    if hasattr(user, 'mfa_enabled'):
        user.mfa_enabled = True
        user.mfa_secret = secret
        await db.commit()

    return {
```

```
        "message": "MFA enabled successfully",
        "secret": secret  # ■ ■■■■■■■■■■ ■■ ■■■■■■■■■■ ■■■■■■■
    }


@router.post("/mfa/verify", status_code=status.HTTP_200_OK)
async def verify_mfa_endpoint(
    payload: MFAVerifyRequest,
    db: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; dict:
    """
    ■■■■■■■■■■ ■■■■ MFA.

    Args:
        payload: ■■■■ MFA
        db: ■■■■■■■ ■■■■■ ■■■■■■■
        current: ■■■■■■■■ ■■■■■■■■■■■■■■■■■■■■■ ■■■■■■■■■■■■■

    Returns:
        dict: ■■■■■■■■■■ ■■■■■■■■■
    """
    result = await db.execute(select(User).where(User.username == current.sub))
    user = result.scalar_one_or_none()

    if not user:
        raise HTTPException(
            status_code=status.HTTP_404_NOT_FOUND,
            detail="User not found"
        )

    is_valid = await verify_mfa_code(db, user.id, payload.code)

    if not is_valid:
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="Invalid MFA code"
        )

    return {"message": "MFA code verified successfully"}


@router.post("/mfa/disable", status_code=status.HTTP_200_OK)
async def disable_mfa_endpoint(
    payload: MFADisableRequest,
    db: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; dict:
    """
    ■■■■■■■■■■ ■■■■■■■■■■■■■■■ ■■■■■■■■■■■■■■■ ■■■■ ■■■■■■■■■■■■■■.

    Args:
        payload: ■■■■■■■■■■■■■■■ ■■■■■■■
        db: ■■■■■■■ ■■■■■ ■■■■■■■
        current: ■■■■■■■■ ■■■■■■■■■■■■■■■■■■■■■ ■■■■■■■■■■■■■

    Returns:
        dict: ■■■■■■■■■■■ ■■ ■■■■■■■■■ ■■■■■■■■■■■■ MFA
    """
    result = await db.execute(select(User).where(User.username == current.sub))
    user = result.scalar_one_or_none()

    if not user:
        raise HTTPException(
            status_code=status.HTTP_404_NOT_FOUND,
            detail="User not found"
        )

    # ■■■■■■■■■■ ■■■■■■■
    if not verify_password(payload.password, user.password):
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="Invalid password"
```

```
        )

    # ■■■■■■■■■■ MFA
    await disable_mfa(db, user.id)

    # ■■■■■■■■■ ■■■■■■■■■■■■■
    if hasattr(user, 'mfa_enabled'):
        user.mfa_enabled = False
        user.mfa_secret = None
        await db.commit()

    return {"message": "MFA disabled successfully"}


@router.get("/me", response_model=dict)
async def get_current_user_info(
    db: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; dict:
    """
    ■■■■■■■■■ ■■■■■■■■■■■ ■ ■■■■■■■■ ■■■■■■■■■■■■■■■■■■■■■■ ■■■■■■■■■■■■■.

    Args:
        db: ■■■■■■■ ■■■■■ ■■■■■■■
        current: ■■■■■■■■ ■■■■■■■■■■■■■■■■■■■■■ ■■■■■■■■■■■■■

    Returns:
        dict: ■■■■■■■■■■■ ■ ■■■■■■■■■■■■■
    """
    result = await db.execute(select(User).where(User.username == current.sub))
    user = result.scalar_one_or_none()

    if not user:
        raise HTTPException(
            status_code=status.HTTP_404_NOT_FOUND,
            detail="User not found"
        )

    # ■■■■■■■■■ ■■■■■■■■■■■ ■■■■■■■■■ ■■■■■■■
    active_sessions_count = await get_active_sessions_count(db, user.id)

    return {
        "id": user.id,
        "username": user.username,
        "role": user.role,
        "mfa_enabled": getattr(user, 'mfa_enabled', False),
        "active_sessions_count": active_sessions_count,
    }


@router.get("/sessions", response_model=list[dict])
async def get_user_sessions(
    request: Request,
    db: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; list[dict]:
    """
    ■■■■■■■■■ ■■■■■■■ ■■■■■■■■■ ■■■■■■■ ■■■■■■■■■■■■■.

    Args:
        request: HTTP ■■■■■■■ (■■■■ ■■■■■■■■■■ ■■■■■■■■■ ■■■■■■■)
        db: ■■■■■■■ ■■■■■ ■■■■■■■
        current: ■■■■■■■■ ■■■■■■■■■■■■■■■■■■■■■ ■■■■■■■■■■■■■

    Returns:
        list[dict]: ■■■■■■■ ■■■■■■■■■ ■■■■■■■
    """
    result = await db.execute(select(User).where(User.username == current.sub))
    user = result.scalar_one_or_none()

    if not user:
        raise HTTPException(
            status_code=status.HTTP_404_NOT_FOUND,
```

```
            detail="User not found"
        )

    # ■■■■■■■■■ ■■■■■■■■ ■■■■■■ ■■ ■■■■■■■■■■ Authorization
    auth_header = request.headers.get("authorization", "")
    current_token = auth_header.replace("Bearer ", "") if auth_header.startswith("Bearer ") else ""

    # ■■■■■■■■■ ■■■■ ■■■■■■■■■ ■■■■■■■ ■■■■■■■■■■■■■
    sessions_result = await db.execute(
        select(UserSession).where(
            UserSession.user_id == user.id,
            UserSession.is_active == True,
            UserSession.expires_at &gt; datetime.now(timezone.utc)
        ).order_by(UserSession.last_activity.desc())
    )
    sessions = sessions_result.scalars().all()

    return [
        {
            "id": session.id,
            "created_at": session.created_at.isoformat(),
            "last_activity": session.last_activity.isoformat(),
            "expires_at": session.expires_at.isoformat(),
            "ip_address": session.ip_address,
            "user_agent": session.user_agent,
            "is_current": session.token == current_token,
        }
        for session in sessions
    ]


@router.delete("/sessions/{session_id}", status_code=status.HTTP_200_OK)
async def delete_session(
    session_id: int,
    db: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; dict:
    """
    ■■■■■■■■■■ ■■■■■■■■■■■ ■■■■■■■ ■■■■■■■■■■■■■.

    Args:
        session_id: ID ■■■■■■■ ■■■■ ■■■■■■■■■■■
        db: ■■■■■■■ ■■■■■ ■■■■■■■
        current: ■■■■■■■■ ■■■■■■■■■■■■■■■■■■■■■ ■■■■■■■■■■■■■

    Returns:
        dict: ■■■■■■■■■■ ■■ ■■■■■■■■■ ■■■■■■■■■■■ ■■■■■■■
    """
    result = await db.execute(select(User).where(User.username == current.sub))
    user = result.scalar_one_or_none()

    if not user:
        raise HTTPException(
            status_code=status.HTTP_404_NOT_FOUND,
            detail="User not found"
        )

    # ■■■■■■ ■■■■■■■ ■ ■■■■■■■■■■, ■■■■ ■■■■ ■■■■■■■■■■■■ ■■■■■■■■■ ■■■■■■■■■■■■■
    session_result = await db.execute(
        select(UserSession).where(
            UserSession.id == session_id,
            UserSession.user_id == user.id
        )
    )
    session = session_result.scalar_one_or_none()

    if not session:
        raise HTTPException(
            status_code=status.HTTP_404_NOT_FOUND,
            detail="Session not found"
        )

    # ■■■■■■■■■■ ■■■■■■■
```

```
    session.is_active = False
    await db.commit()

    return {"message": "Session ended successfully"}


@router.post("/forgot-password", status_code=status.HTTP_200_OK)
async def forgot_password(
    payload: ForgotPasswordRequest,
    db: AsyncSession = Depends(get_db),
) -&gt; dict:
    """
    ■■■■■■■ ■■ ■■■■■■■■■■■■■■■ ■■■■■■■.

    ■■■■■■■■■■■ ■■■■ ■■■■■■■■■■■■■■■■ ■■■■■■■ ■■ email ■■■■■■■■■■■■■■.
    ■■■■ ■■■■■■■■■■■■■■ ■■ ■■■■■■■■■■■■, ■■■■■■■■■■■■ ■■ ■■■■■■■■■■■■■■■ ■ ■■■■■■ email.

    Args:
        payload: Email ■■■■■■■■■■■■■■
        db: ■■■■■■■ ■■■■■ ■■■■■■■

    Returns:
        dict: ■■■■■■■■■■ ■■ ■■■■■■■■■ ■■■■■■■■■ ■■■■■ (■■■■■■■ ■■■■■■■■ ■■■■ ■■■■■■■■■■■■■■■)
    """
    # ■■■■■■■■ ■ ■■■■■■■■■■ ■■■■ ■■■■■■■■■■■■■■■■
    # ■■■■■ ■■■■■■■■■■■■■■■ ■■ ■■■■■■■■, ■■■■ ■■■■■■ ■■■■■■■■■■■■ ■■■■■■ (■■■■■■■■■■■■■■■)
    await create_password_reset_code(db, payload.email)

    return {
        "message": "If an account with this email exists, a password reset code has been sent."
    }


@router.post("/reset-password", status_code=status.HTTP_200_OK)
async def reset_password(
    payload: ResetPasswordRequest,
    db: AsyncSession = Depends(get_db),
) -&gt; dict:
    """
    ■■■■■■■■■■■■ ■■■■■■ ■■■■■■■ ■■■■■■ ■■■■■■■■■■■■■■■■■.

    ■■■■■■■■■■ ■■■■ ■■■■■■■■■■■■■■■■ ■ ■■■■■■■■■■■■■■■ ■■■■■■ ■■■■■■■.
    ■■■■■■ ■■■■■■■■■■ ■■■■■■■ ■■■■ ■■■■■■■■■ ■■■■■■■ ■■■■■■■■■■■■■■ ■■■■■■■■■■■■■.

    Args:
        payload: Email, ■■■■ ■■■■■■■■■■■■■■■■ ■ ■■■■■■ ■■■■■■■
        db: ■■■■■■■ ■■■■■ ■■■■■■■

    Returns:
        dict: ■■■■■■■■■■ ■■ ■■■■■■■■■ ■■■■■■■ ■■■■■■■

    Raises:
        HTTPException: ■■■■■ ■■■■ ■■■■■■■■ ■■■■ ■■■■■■
    """
    # ■■■■■■■■■■ ■■■■ ■■■■■■■■■■■■■■■■
    user_id = await verify_password_reset_code(db, payload.email, payload.code)

    if not user_id:
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="Invalid or expired reset code"
        )

    # ■■■■■■ ■■■■■■■■■■■■■■
    result = await db.execute(select(User).where(User.id == user_id))
    user = result.scalar_one_or_none()

    if not user:
        raise HTTPException(
            status_code=status.HTTP_404_NOT_FOUND,
            detail="User not found"
        )
```

```
# ■■■■■■■■■■ ■■■■■■■ ■■■■■■■
if len(payload.new_password) &lt; 6:
    raise HTTPException(
        status_code=status.HTTP_400_BAD_REQUEST,
        detail="Password must be at least 6 characters long"
    )

# ■■■■■■■■■■■ ■■■■■■ ■■■■■■■
user.password = get_password_hash(payload.new_password)
await db.commit()

# ■■■■■■■■■■ ■■■■ ■■■■■■■■■ ■■■■■■■ ■■■■■■■■■■■■■■
await end_all_user_sessions(db, user.id)

return {
    "message": "Password reset successfully. Please login with your new password."
}
```

File: backend\app\api\routes\bookings.py

Full path: C:\Users\itodo\Hostelrec\backend\app\api\routes\bookings.py

```
from fastapi import APIRouter, Depends, HTTPException, status
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy import text
from typing import List, Optional

from app.core.security import get_current_user, require_role, TokenPayload
from app.core.audit import set_audit_user
from app.db.session import get_db
from app.schemas.booking import BookingCreate, BookingResponse
from app.services.booking_service import create_booking

router = APIRouter()


@router.post("/", response_model=BookingResponse, status_code=status.HTTP_201_CREATED)
async def create_booking_endpoint(
    payload: BookingCreate,
    session: AsyncSession = Depends(get_db),
) -&gt; BookingResponse:
    """
    Create a booking. Public endpoint - anyone can book a room without authentication.
    """
    try:
        # Set audit user - use guest name for tracking
        username = f"guest_{payload.guest_name.replace(' ', '_').lower()}"
        if hasattr(payload, 'guest_email') and payload.guest_email:
            username = payload.guest_email

        await set_audit_user(session, username)
        return await create_booking(session, payload, username)
    except ValueError as exc:
        raise HTTPException(status_code=status.HTTP_400_BAD_REQUEST, detail=str(exc)) from exc


@router.get("/my", response_model=List[dict])
async def get_my_bookings(
    passport_number: Optional[str] = None,
    phone_number: Optional[str] = None,
    session: AsyncSession = Depends(get_db),
) -&gt; List[dict]:
    """
    Get bookings for a guest by passport number or phone number.
    Public endpoint - guests can view their own bookings without authentication.
    """
    if not passport_number and not phone_number:
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="Either passport_number or phone_number must be provided"
        )

    query = """
        SELECT
            b.id AS booking_id,
            b.room_id,
            b.check_in,
            b.check_out,
            b.status AS booking_status,
            b.payment_method,
            b.payment_status,
            b.total_amount,
            b.created_at AS booking_created_at,
            b.updated_at AS booking_updated_at,
            g.id AS guest_id,
            g.name AS guest_name,
            g.passport_number AS guest_passport,
            g.phone_number AS guest_phone,
            g.email AS guest_email,
            r.room_number,
```

```
            r.title AS room_title,
            r.price AS room_price,
            r.location AS room_location
        FROM bookings AS b
        JOIN guests AS g ON b.guest_id = g.id
        LEFT JOIN rooms AS r ON b.room_id = r.id
        WHERE 1=1
    """
    params = {}

    if passport_number:
        query += " AND g.passport_number = :passport_number"
        params["passport_number"] = passport_number

    if phone_number:
        query += " AND g.phone_number = :phone_number"
        params["phone_number"] = phone_number

    query += " ORDER BY b.created_at DESC"

    result = await session.execute(text(query), params)
    rows = result.mappings().all()
    return [dict(row) for row in rows]


@router.get("/", response_model=List[dict])
async def list_bookings(
    status_filter: Optional[str] = None,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
) -&gt; List[dict]:
    """
    List all bookings. Receptionist and manager can view all bookings.
    Shows guest information, room details, and booking status.
    """
    query = """
        SELECT
            b.id AS booking_id,
            b.room_id,
            b.check_in,
            b.check_out,
            b.status AS booking_status,
            b.payment_method,
            b.payment_status,
            b.total_amount,
            b.created_at AS booking_created_at,
            b.updated_at AS booking_updated_at,
            g.id AS guest_id,
            g.name AS guest_name,
            g.passport_number AS guest_passport,
            g.phone_number AS guest_phone,
            g.email AS guest_email,
            r.room_number,
            r.title AS room_title,
            r.price AS room_price,
            r.location AS room_location
        FROM bookings AS b
        JOIN guests AS g ON b.guest_id = g.id
        LEFT JOIN rooms AS r ON b.room_id = r.id
    """
    params = {}

    if status_filter:
        query += " WHERE b.status = :status_filter"
        params["status_filter"] = status_filter

    query += " ORDER BY b.created_at DESC"

    result = await session.execute(text(query), params)
    rows = result.mappings().all()
    return [dict(row) for row in rows]
```

File: backend\app\api\routes\chat.py

Full path: C:\Users\itodo\Hostelrec\backend\app\api\routes\chat.py

```
from fastapi import APIRouter, WebSocket, WebSocketDisconnect, Depends, HTTPException, status
from sqlalchemy import text, select
from sqlalchemy.ext.asyncio import AsyncSession
from typing import Set
import json

from app.db.session import AsyncSessionLocal, get_db
from app.core.security import get_current_user, TokenPayload
from app.models import User

router = APIRouter()

# Simple in-memory connection manager (for production, use Redis or similar)
class ConnectionManager:
    def __init__(self):
        self.active_connections: Set[WebSocket] = set()

    async def connect(self, websocket: WebSocket):
        await websocket.accept()
        self.active_connections.add(websocket)

    def disconnect(self, websocket: WebSocket):
        self.active_connections.discard(websocket)

    async def broadcast(self, message: dict):
        disconnected = set()
        for connection in self.active_connections:
            try:
                await connection.send_json(message)
            except:
                disconnected.add(connection)
        for conn in disconnected:
            self.active_connections.discard(conn)

manager = ConnectionManager()


@router.websocket("/ws")
async def websocket_endpoint(websocket: WebSocket):
    await manager.connect(websocket)
    try:
        while True:
            data = await websocket.receive_text()
            message_data = json.loads(data)

            # Get user_id from username if not provided
            user_id = message_data.get("user_id")
            username = message_data.get("username", "Anonymous")

            # Save message to database
            async with AsyncSessionLocal() as session:
                # If user_id not provided, try to get it from username
                if not user_id and username != "Anonymous":
                    result_user = await session.execute(
                        select(User).where(User.username == username)
                    )
                    user = result_user.scalar_one_or_none()
                    if user:
                        user_id = user.id

                result = await session.execute(
                    text(
                        """
                        INSERT INTO chat_messages (user_id, username, message)
                        VALUES (:user_id, :username, :message)
                        RETURNING id, user_id, created_at
                        """
                    ),
```

```
                    {
                        "user_id": user_id,
                        "username": username,
                        "message": message_data.get("message", ""),
                    },
                )
                row = result.mappings().first()
                await session.commit()

            # Broadcast to all connected clients
            await manager.broadcast({
                "type": "message",
                "id": row["id"] if row else None,
                "user_id": row["user_id"] if row else user_id,
                "username": username,
                "message": message_data.get("message", ""),
                "timestamp": message_data.get("timestamp") or (row["created_at"].isoformat() if row else Non
            })
    except WebSocketDisconnect:
        manager.disconnect(websocket)


@router.get("/messages")
async def get_recent_messages(session: AsyncSession = Depends(get_db)):
    """Get last 50 chat messages"""
    result = await session.execute(
        text(
            """
            SELECT id, user_id, username, message, created_at
            FROM chat_messages
            ORDER BY created_at DESC
            LIMIT 50
            """
        )
    )
    rows = result.mappings().all()
    return [dict(row) for row in reversed(rows)]  # Return in chronological order


@router.delete("/messages/{message_id}")
async def delete_message(
    message_id: int,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
):
    """Delete a chat message. Users can delete their own messages, staff can delete any."""
    # Get message info
    result = await session.execute(
        text(
            """
            SELECT user_id FROM chat_messages WHERE id = :message_id
            """
        ),
        {"message_id": message_id},
    )
    message_row = result.mappings().first()
    if not message_row:
        raise HTTPException(status_code=status.HTTP_404_NOT_FOUND, detail="Message not found")

    # Check permissions: user can delete their own messages, staff can delete any
    if current.role not in ("manager", "receptionist"):
        # Get current user's ID
        user_result = await session.execute(
            text("SELECT id FROM hostel_users WHERE username = :username"),
            {"username": current.sub},
        )
        user_row = user_result.mappings().first()
        if not user_row or user_row["id"] != message_row["user_id"]:
            raise HTTPException(
                status_code=status.HTTP_403_FORBIDDEN, detail="You can only delete your own messages"
            )

    # Delete the message
```

```
await session.execute(
    text("DELETE FROM chat_messages WHERE id = :message_id"),
    {"message_id": message_id},
)
await session.commit()
return {"id": message_id, "deleted": True}
```

File: backend\app\api\routes\content.py

Full path: C:\Users\itodo\Hostelrec\backend\app\api\routes\content.py

```
from fastapi import APIRouter, Depends, HTTPException, status, UploadFile, File, Form
from sqlalchemy import text, select
from sqlalchemy.ext.asyncio import AsyncSession
from typing import List, Optional
import os
import uuid
import aiofiles
from pathlib import Path

from app.core.security import require_role, get_current_user, TokenPayload
from app.core.audit import set_audit_user
from app.db.session import get_db

router = APIRouter()

# Create uploads directory if it doesn't exist
UPLOAD_DIR = Path("uploads")
UPLOAD_DIR.mkdir(exist_ok=True)
UPLOAD_DIR.joinpath("rooms").mkdir(exist_ok=True)
UPLOAD_DIR.joinpath("foods").mkdir(exist_ok=True)
UPLOAD_DIR.joinpath("drinks").mkdir(exist_ok=True)
UPLOAD_DIR.joinpath("events").mkdir(exist_ok=True)
UPLOAD_DIR.joinpath("promos").mkdir(exist_ok=True)


async def save_upload_file(file: UploadFile, category: str) -&gt; str:
    """Save uploaded file and return the URL path"""
    file_ext = file.filename.split(".")[-1] if "." in file.filename else "jpg"
    filename = f"{uuid.uuid4()}.{file_ext}"
    filepath = UPLOAD_DIR / category / filename

    async with aiofiles.open(filepath, "wb") as f:
        content = await file.read()
        await f.write(content)

    return f"/uploads/{category}/{filename}"


# Room Images
@router.post("/rooms/images")
async def upload_room_image(
    room_number: str = Form(...),
    description: Optional[str] = Form(None),
    file: UploadFile = File(...),
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Upload a room image (receptionist/manager only)"""
    await set_audit_user(session, current.sub)
    image_url = await save_upload_file(file, "rooms")

    await session.execute(
        text(
            """
            INSERT INTO room_images (room_number, image_url, description)
            VALUES (:room_number, :image_url, :description)
            """
        ),
        {"room_number": room_number, "image_url": image_url, "description": description},
    )
    await session.commit()
    return {"room_number": room_number, "image_url": image_url, "description": description}


@router.post("/rooms/{room_id}/images")
async def upload_multiple_room_images(
    room_id: int,
```

```
    files: List[UploadFile] = File(...),
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Upload multiple room images (up to 5) for a room (receptionist/manager only)"""
    await set_audit_user(session, current.sub)

    # Get room_number from room_id
    result = await session.execute(
        text("SELECT room_number FROM rooms WHERE id = :room_id"),
        {"room_id": room_id},
    )
    room = result.mappings().first()
    if not room:
        raise HTTPException(status_code=404, detail="Room not found")

    room_number = room["room_number"]

    # Limit to 5 images
    if len(files) &gt; 5:
        raise HTTPException(status_code=400, detail="Maximum 5 images allowed per room")

    uploaded_images = []
    for file in files:
        image_url = await save_upload_file(file, "rooms")
        await session.execute(
            text(
                """
                INSERT INTO room_images (room_number, image_url, description)
                VALUES (:room_number, :image_url, :description)
                """
            ),
            {"room_number": room_number, "image_url": image_url, "description": None},
        )
        uploaded_images.append({"room_number": room_number, "image_url": image_url})

    await session.commit()
    return {"room_id": room_id, "room_number": room_number, "images": uploaded_images}


@router.get("/rooms/images")
async def get_room_images(
    room_number: Optional[str] = None,
    session: AsyncSession = Depends(get_db),
):
    """Get room images (public)"""
    if room_number:
        result = await session.execute(
            text("SELECT * FROM room_images WHERE room_number = :room_number ORDER BY created_at DESC"),
            {"room_number": room_number},
        )
    else:
        result = await session.execute(
            text("SELECT * FROM room_images ORDER BY created_at DESC")
        )
    rows = result.mappings().all()
    return [dict(row) for row in rows]


@router.delete("/rooms/images/{image_id}")
async def delete_room_image(
    image_id: int,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Delete a room image (receptionist/manager only)"""
    await set_audit_user(session, current.sub)

    # Get the image record to find the file path
    result = await session.execute(
        text("SELECT image_url FROM room_images WHERE id = :image_id"),
```

```
        {"image_id": image_id},
    )
    image = result.mappings().first()
    if not image:
        raise HTTPException(status_code=404, detail="Image not found")

    # Delete the database record
    await session.execute(
        text("DELETE FROM room_images WHERE id = :image_id"),
        {"image_id": image_id},
    )
    await session.commit()

    # Try to delete the physical file
    try:
        image_path = image["image_url"]
        if image_path and image_path.startswith("/uploads/"):
            file_path = UPLOAD_DIR / image_path.replace("/uploads/", "")
            if file_path.exists():
                file_path.unlink()
    except Exception as e:
        # Log but don't fail if file deletion fails
        print(f"Warning: Failed to delete image file: {e}")

    return {"id": image_id, "deleted": True}


# Rooms Management
@router.post("/rooms")
async def create_room(
    room_number: str = Form(...),
    title: str = Form(...),
    description: Optional[str] = Form(None),
    price: float = Form(...),
    location: Optional[str] = Form(None),
    wifi: bool = Form(True),
    features: Optional[str] = Form(None),  # Comma-separated string
    status: str = Form("available"),
    file: Optional[UploadFile] = File(None),
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Create a room (receptionist/manager only)"""
    # Ensure we start with a clean transaction state
    try:
        await session.rollback()
    except Exception:
        pass  # Ignore if there's nothing to rollback

    # Check if room number already exists
    result = await session.execute(
        text("SELECT id FROM rooms WHERE room_number = :room_number"),
        {"room_number": room_number}
    )
    if result.scalar_one_or_none():
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail=f"Room number '{room_number}' already exists. Please use a different room number."
        )

    await set_audit_user(session, current.sub)
    image_url = None
    if file:
        image_url = await save_upload_file(file, "rooms")

    # Convert features string to array
    features_array = []
    if features:
        features_array = [f.strip() for f in features.split(",") if f.strip()]

    try:
        await session.execute(
```

```
            text(
                """
                INSERT INTO rooms (room_number, title, description, price, location, wifi, features, image_u
                VALUES (:room_number, :title, :description, :price, :location, :wifi, :features, :image_url,
                """
            ),
            {
                "room_number": room_number,
                "title": title,
                "description": description,
                "price": price,
                "location": location,
                "wifi": wifi,
                "features": features_array,
                "image_url": image_url,
                "status": status,
            },
        )
        await session.commit()
    except Exception as e:
        await session.rollback()
        # Check if it's a duplicate key error
        error_str = str(e)
        if "duplicate key" in error_str.lower() or "unique constraint" in error_str.lower() or "already exis
    _str.lower():
            raise HTTPException(
                status_code=status.HTTP_400_BAD_REQUEST,
                detail=f"Room number '{room_number}' already exists. Please use a different room number."
            )
        # Re-raise other errors
        raise HTTPException(
            status_code=status.HTTP_500_INTERNAL_SERVER_ERROR,
            detail=f"Failed to create room: {str(e)}"
        )

    return {"room_number": room_number, "title": title, "price": price, "image_url": image_url, "status": st


@router.get("/rooms")
async def get_rooms(
    available_only: bool = False,
    session: AsyncSession = Depends(get_db),
):
    """Get rooms (public)"""
    query = "SELECT * FROM rooms"
    if available_only:
        query += " WHERE status = 'available'"
    query += " ORDER BY room_number ASC"

    result = await session.execute(text(query))
    rows = result.mappings().all()
    return [dict(row) for row in rows]


@router.get("/rooms/{room_id}")
async def get_room_detail(
    room_id: int,
    session: AsyncSession = Depends(get_db),
):
    """Get room details with all images"""
    # Get room info
    result = await session.execute(
        text("SELECT * FROM rooms WHERE id = :room_id"),
        {"room_id": room_id},
    )
    room = result.mappings().first()
    if not room:
        raise HTTPException(status_code=404, detail="Room not found")

    # Get all images for this room
    images_result = await session.execute(
        text(
            """
```

```
            SELECT id, image_url, description, created_at
            FROM room_images
            WHERE room_number = :room_number
            ORDER BY created_at DESC
            LIMIT 5
            """
        ),
        {"room_number": room["room_number"]},
    )
    images = [dict(row) for row in images_result.mappings().all()]

    room_dict = dict(room)
    room_dict["images"] = images
    return room_dict


@router.put("/rooms/{room_id}")
async def update_room(
    room_id: int,
    room_number: Optional[str] = Form(None),
    title: Optional[str] = Form(None),
    description: Optional[str] = Form(None),
    price: Optional[float] = Form(None),
    location: Optional[str] = Form(None),
    wifi: Optional[bool] = Form(None),
    features: Optional[str] = Form(None),
    status: Optional[str] = Form(None),
    file: Optional[UploadFile] = File(None),
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Update a room (receptionist/manager only) - supports file upload"""
    await set_audit_user(session, current.sub)
    updates = []
    params = {"room_id": room_id}

    # Handle file upload if provided
    if file:
        image_url = await save_upload_file(file, "rooms")
        updates.append("image_url = :image_url")
        params["image_url"] = image_url

    if room_number:
        updates.append("room_number = :room_number")
        params["room_number"] = room_number
    if title:
        updates.append("title = :title")
        params["title"] = title
    if description is not None:
        updates.append("description = :description")
        params["description"] = description
    if price is not None:
        updates.append("price = :price")
        params["price"] = price
    if location is not None:
        updates.append("location = :location")
        params["location"] = location
    if wifi is not None:
        updates.append("wifi = :wifi")
        params["wifi"] = wifi
    if features is not None:
        features_array = [f.strip() for f in features.split(",") if f.strip()]
        updates.append("features = :features")
        params["features"] = features_array
    if status:
        updates.append("status = :status")
        params["status"] = status

    if not updates:
        raise HTTPException(status_code=400, detail="No fields to update")

    updates.append("updated_at = NOW()")
```

```
    await session.execute(
        text(f"UPDATE rooms SET {', '.join(updates)} WHERE id = :room_id"),
        params,
    )
    await session.commit()
    return {"id": room_id, "updated": True}


@router.post("/rooms/{room_id}/mark-cleaned")
async def mark_room_cleaned(
    room_id: int,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("cleaner", "receptionist", "manager")),
):
    """Mark a room as cleaned (cleaner/receptionist/manager only)"""
    await set_audit_user(session, current.sub)

    # Update room's updated_at timestamp to indicate it was cleaned
    await session.execute(
        text("UPDATE rooms SET updated_at = NOW() WHERE id = :room_id"),
        {"room_id": room_id}
    )
    await session.commit()
    return {"room_id": room_id, "cleaned_at": "now", "status": "success"}


@router.delete("/rooms/{room_id}")
async def delete_room(
    room_id: int,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Delete a room (receptionist/manager only)"""
    await set_audit_user(session, current.sub)
    await session.execute(text("DELETE FROM rooms WHERE id = :room_id"), {"room_id": room_id})
    await session.commit()
    return {"id": room_id, "deleted": True}


# Food Items
@router.post("/foods")
async def create_food_item(
    name: str = Form(...),
    description: Optional[str] = Form(None),
    price: float = Form(0.0),
    category: str = Form("food"),
    available: bool = Form(True),
    is_tea: bool = Form(False),
    contains_alcohol: bool = Form(False),
    file: Optional[UploadFile] = File(None),
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Create a food item (receptionist/manager only)"""
    await set_audit_user(session, current.sub)
    image_url = None
    if file:
        image_url = await save_upload_file(file, "foods")

    await session.execute(
        text(
            """
            INSERT INTO food_items (name, description, price, image_url, category, available, is_tea, contai
            VALUES (:name, :description, :price, :image_url, :category, :available, :is_tea, :contains_alcoh
            """
        ),
        {
            "name": name,
            "description": description,
            "price": price,
```

```
            "image_url": image_url,
            "category": category,
            "available": available,
            "is_tea": is_tea,
            "contains_alcohol": contains_alcohol,
        },
    )
    await session.commit()
    return {"name": name, "description": description, "price": price, "image_url": image_url, "category": ca
    ilable": available, "is_tea": is_tea, "contains_alcohol": contains_alcohol}


@router.get("/foods")
async def get_food_items(
    available_only: bool = False,
    session: AsyncSession = Depends(get_db),
):
    """Get food items (public)"""
    query = "SELECT * FROM food_items"
    if available_only:
        query += " WHERE available = true"
    query += " ORDER BY created_at DESC"

    result = await session.execute(text(query))
    rows = result.mappings().all()
    return [dict(row) for row in rows]


@router.put("/foods/{item_id}")
async def update_food_item(
    item_id: int,
    name: Optional[str] = None,
    description: Optional[str] = None,
    price: Optional[float] = None,
    available: Optional[bool] = None,
    is_tea: Optional[bool] = None,
    contains_alcohol: Optional[bool] = None,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Update a food item (receptionist/manager only)"""
    await set_audit_user(session, current.sub)
    updates = []
    params = {"item_id": item_id}

    if name:
        updates.append("name = :name")
        params["name"] = name
    if description is not None:
        updates.append("description = :description")
        params["description"] = description
    if price is not None:
        updates.append("price = :price")
        params["price"] = price
    if available is not None:
        updates.append("available = :available")
        params["available"] = available
    if is_tea is not None:
        updates.append("is_tea = :is_tea")
        params["is_tea"] = is_tea
    if contains_alcohol is not None:
        updates.append("contains_alcohol = :contains_alcohol")
        params["contains_alcohol"] = contains_alcohol

    if not updates:
        raise HTTPException(status_code=400, detail="No fields to update")

    await session.execute(
        text(f"UPDATE food_items SET {', '.join(updates)} WHERE id = :item_id"),
        params,
    )
    await session.commit()
```

```
    return {"id": item_id, "updated": True}


@router.delete("/foods/{item_id}")
async def delete_food_item(
    item_id: int,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Delete a food item (receptionist/manager only)"""
    await set_audit_user(session, current.sub)
    await session.execute(text("DELETE FROM food_items WHERE id = :item_id"), {"item_id": item_id})
    await session.commit()
    return {"id": item_id, "deleted": True}


# Drink Items (similar to foods)
@router.post("/drinks")
async def create_drink_item(
    name: str = Form(...),
    description: Optional[str] = Form(None),
    price: float = Form(0.0),
    category: str = Form("drink"),
    available: bool = Form(True),
    is_tea: bool = Form(False),
    contains_alcohol: bool = Form(False),
    file: Optional[UploadFile] = File(None),
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Create a drink item (receptionist/manager only)"""
    await set_audit_user(session, current.sub)
    image_url = None
    if file:
        image_url = await save_upload_file(file, "drinks")

    await session.execute(
        text(
            """
            INSERT INTO drink_items (name, description, price, image_url, category, available, is_tea, conta
            VALUES (:name, :description, :price, :image_url, :category, :available, :is_tea, :contains_alcoh
            """
        ),
        {
            "name": name,
            "description": description,
            "price": price,
            "image_url": image_url,
            "category": category,
            "available": available,
            "is_tea": is_tea,
            "contains_alcohol": contains_alcohol,
        },
    )
    await session.commit()
    return {"name": name, "description": description, "price": price, "image_url": image_url, "category": ca
    ilable": available, "is_tea": is_tea, "contains_alcohol": contains_alcohol}


@router.get("/drinks")
async def get_drink_items(
    available_only: bool = False,
    session: AsyncSession = Depends(get_db),
):
    """Get drink items (public)"""
    query = "SELECT * FROM drink_items"
    if available_only:
        query += " WHERE available = true"
    query += " ORDER BY created_at DESC"

    result = await session.execute(text(query))
```

```
    rows = result.mappings().all()
    return [dict(row) for row in rows]


@router.put("/drinks/{item_id}")
async def update_drink_item(
    item_id: int,
    name: Optional[str] = None,
    description: Optional[str] = None,
    price: Optional[float] = None,
    available: Optional[bool] = None,
    is_tea: Optional[bool] = None,
    contains_alcohol: Optional[bool] = None,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Update a drink item (receptionist/manager only)"""
    await set_audit_user(session, current.sub)
    updates = []
    params = {"item_id": item_id}

    if name:
        updates.append("name = :name")
        params["name"] = name
    if description is not None:
        updates.append("description = :description")
        params["description"] = description
    if price is not None:
        updates.append("price = :price")
        params["price"] = price
    if available is not None:
        updates.append("available = :available")
        params["available"] = available
    if is_tea is not None:
        updates.append("is_tea = :is_tea")
        params["is_tea"] = is_tea
    if contains_alcohol is not None:
        updates.append("contains_alcohol = :contains_alcohol")
        params["contains_alcohol"] = contains_alcohol

    if not updates:
        raise HTTPException(status_code=400, detail="No fields to update")

    await session.execute(
        text(f"UPDATE drink_items SET {', '.join(updates)} WHERE id = :item_id"),
        params,
    )
    await session.commit()
    return {"id": item_id, "updated": True}


@router.delete("/drinks/{item_id}")
async def delete_drink_item(
    item_id: int,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Delete a drink item (receptionist/manager only)"""
    await set_audit_user(session, current.sub)
    await session.execute(text("DELETE FROM drink_items WHERE id = :item_id"), {"item_id": item_id})
    await session.commit()
    return {"id": item_id, "deleted": True}


# Events
@router.post("/events")
async def create_event(
    title: str = Form(...),
    description: Optional[str] = Form(None),
    event_date: str = Form(...),
    location: Optional[str] = Form(None),
```

```
    active: bool = Form(True),
    file: Optional[UploadFile] = File(None),
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Create an event (receptionist/manager only)"""
    await set_audit_user(session, current.sub)
    image_url = None
    if file:
        image_url = await save_upload_file(file, "events")

    await session.execute(
        text(
            """
            INSERT INTO events (title, description, event_date, location, image_url, active)
            VALUES (:title, :description, :event_date::timestamptz, :location, :image_url, :active)
            """
        ),
        {
            "title": title,
            "description": description,
            "event_date": event_date,
            "location": location,
            "image_url": image_url,
            "active": active,
        },
    )
    await session.commit()
    return {"title": title, "description": description, "event_date": event_date, "location": location, "ima
    ge_url, "active": active}


@router.get("/events")
async def get_events(
    active_only: bool = False,
    session: AsyncSession = Depends(get_db),
):
    """Get events (public)"""
    query = "SELECT * FROM events"
    if active_only:
        query += " WHERE active = true AND event_date &gt;= NOW()"
    query += " ORDER BY event_date ASC"

    result = await session.execute(text(query))
    rows = result.mappings().all()
    return [dict(row) for row in rows]


@router.put("/events/{event_id}")
async def update_event(
    event_id: int,
    title: Optional[str] = None,
    description: Optional[str] = None,
    event_date: Optional[str] = None,
    location: Optional[str] = None,
    active: Optional[bool] = None,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Update an event (receptionist/manager only)"""
    await set_audit_user(session, current.sub)
    updates = []
    params = {"event_id": event_id}

    if title:
        updates.append("title = :title")
        params["title"] = title
    if description is not None:
        updates.append("description = :description")
        params["description"] = description
    if event_date:
```

```
        updates.append("event_date = :event_date::timestamptz")
        params["event_date"] = event_date
    if location is not None:
        updates.append("location = :location")
        params["location"] = location
    if active is not None:
        updates.append("active = :active")
        params["active"] = active

    if not updates:
        raise HTTPException(status_code=400, detail="No fields to update")

    await session.execute(
        text(f"UPDATE events SET {', '.join(updates)} WHERE id = :event_id"),
        params,
    )
    await session.commit()
    return {"id": event_id, "updated": True}


@router.delete("/events/{event_id}")
async def delete_event(
    event_id: int,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Delete an event (receptionist/manager only)"""
    await set_audit_user(session, current.sub)
    await session.execute(text("DELETE FROM events WHERE id = :event_id"), {"event_id": event_id})
    await session.commit()
    return {"id": event_id, "deleted": True}


# Promos
@router.post("/promos")
async def create_promo(
    title: str = Form(...),
    description: Optional[str] = Form(None),
    discount_percent: Optional[int] = Form(None),
    discount_amount: Optional[float] = Form(None),
    code: Optional[str] = Form(None),
    valid_from: str = Form(...),
    valid_until: str = Form(...),
    active: bool = Form(True),
    file: Optional[UploadFile] = File(None),
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Create a promo (receptionist/manager only)"""
    await set_audit_user(session, current.sub)
    image_url = None
    if file:
        image_url = await save_upload_file(file, "promos")

    await session.execute(
        text(
            """
            INSERT INTO promos (title, description, discount_percent, discount_amount, code, valid_from, val
    age_url, active)
            VALUES (:title, :description, :discount_percent, :discount_amount, :code, :valid_from::timestamp
    ntil::timestamptz, :image_url, :active)
            """
        ),
        {
            "title": title,
            "description": description,
            "discount_percent": discount_percent,
            "discount_amount": discount_amount,
            "code": code,
            "valid_from": valid_from,
            "valid_until": valid_until,
```

```
            "image_url": image_url,
            "active": active,
        },
    )
    await session.commit()
    return {"title": title, "description": description, "code": code, "image_url": image_url, "active": acti


@router.get("/promos")
async def get_promos(
    active_only: bool = False,
    session: AsyncSession = Depends(get_db),
):
    """Get promos (public)"""
    query = "SELECT * FROM promos"
    if active_only:
        query += " WHERE active = true AND valid_from &lt;= NOW() AND valid_until &gt;= NOW()"
    query += " ORDER BY created_at DESC"

    result = await session.execute(text(query))
    rows = result.mappings().all()
    return [dict(row) for row in rows]


@router.put("/promos/{promo_id}")
async def update_promo(
    promo_id: int,
    title: Optional[str] = None,
    description: Optional[str] = None,
    discount_percent: Optional[int] = None,
    discount_amount: Optional[float] = None,
    code: Optional[str] = None,
    valid_from: Optional[str] = None,
    valid_until: Optional[str] = None,
    active: Optional[bool] = None,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Update a promo (receptionist/manager only)"""
    await set_audit_user(session, current.sub)
    updates = []
    params = {"promo_id": promo_id}

    if title:
        updates.append("title = :title")
        params["title"] = title
    if description is not None:
        updates.append("description = :description")
        params["description"] = description
    if discount_percent is not None:
        updates.append("discount_percent = :discount_percent")
        params["discount_percent"] = discount_percent
    if discount_amount is not None:
        updates.append("discount_amount = :discount_amount")
        params["discount_amount"] = discount_amount
    if code:
        updates.append("code = :code")
        params["code"] = code
    if valid_from:
        updates.append("valid_from = :valid_from::timestamptz")
        params["valid_from"] = valid_from
    if valid_until:
        updates.append("valid_until = :valid_until::timestamptz")
        params["valid_until"] = valid_until
    if active is not None:
        updates.append("active = :active")
        params["active"] = active

    if not updates:
        raise HTTPException(status_code=400, detail="No fields to update")

    await session.execute(
```

```
        text(f"UPDATE promos SET {', '.join(updates)} WHERE id = :promo_id"),
        params,
    )
    await session.commit()
    return {"id": promo_id, "updated": True}


@router.delete("/promos/{promo_id}")
async def delete_promo(
    promo_id: int,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
):
    """Delete a promo (receptionist/manager only)"""
    await set_audit_user(session, current.sub)
    await session.execute(text("DELETE FROM promos WHERE id = :promo_id"), {"promo_id": promo_id})
    await session.commit()
    return {"id": promo_id, "deleted": True}
```

File: backend\app\api\routes\customer.py

Full path: C:\Users\itodo\Hostelrec\backend\app\api\routes\customer.py

```
from fastapi import APIRouter, Depends, HTTPException, status
from sqlalchemy import select, text
from sqlalchemy.ext.asyncio import AsyncSession

from app.core.security import get_current_user, require_role, TokenPayload
from app.db.session import get_db
from app.models import User
from app.schemas.customer import CustomerProfile

router = APIRouter()


@router.get("/profile")
async def get_customer_profile(
    current: TokenPayload = Depends(get_current_user),
    session: AsyncSession = Depends(get_db),
):
    """
    Return a simple customer-facing profile for the currently logged-in user.
    Compatible with the old endpoint format.
    """
    try:
        # Try to get full profile
        profile = await get_my_profile(current=current, session=session)
        return {
            "email": profile.email,
            "room_number": profile.room_number,
            "floor": profile.floor,
            "preferences": {
                "food": [],
                "drinks": [],
            },
        }
    except HTTPException:
        # If not a customer or profile doesn't exist, return basic info
        return {
            "email": current.sub,
            "room_number": None,
            "floor": None,
            "preferences": {
                "food": [],
                "drinks": [],
            },
        }


async def _ensure_profile_row(user_id: int, session: AsyncSession) -&gt; None:
    """Create an empty customer profile row if it doesn't exist."""
    await session.execute(
        text(
            """
            INSERT INTO customer_profiles (user_id)
            VALUES (:user_id)
            ON CONFLICT (user_id) DO NOTHING
            """
        ),
        {"user_id": user_id},
    )


@router.get("/me", response_model=CustomerProfile)
async def get_my_profile(
    current: TokenPayload = Depends(get_current_user),
    session: AsyncSession = Depends(get_db),
) -&gt; CustomerProfile:
    """
    Return the logged-in customer's profile.
    If the user is not a customer, return 403.
```

```
    """
    result = await session.execute(select(User).where(User.username == current.sub))
    user: User | None = result.scalar_one_or_none()
    if not user:
        raise HTTPException(status_code=status.HTTP_404_NOT_FOUND, detail="User not found")

    if user.role != "customer":
        raise HTTPException(status_code=status.HTTP_403_FORBIDDEN, detail="Not a customer account")

    # Ensure a profile row exists
    await _ensure_profile_row(user.id, session)

    prof_result = await session.execute(
        text(
            """
            SELECT c.full_name,
                   c.room_number,
                   c.floor,
                   c.likes_food,
                   c.likes_water,
                   c.notes
            FROM customer_profiles AS c
            WHERE c.user_id = :user_id
            """
        ),
        {"user_id": user.id},
    )
    row = prof_result.mappings().first()

    return CustomerProfile(
        email=user.username,
        full_name=row["full_name"] if row else None,
        room_number=row["room_number"] if row else None,
        floor=row["floor"] if row else None,
        likes_food=row["likes_food"] if row else None,
        likes_water=row["likes_water"] if row else None,
        notes=row["notes"] if row else None,
    )


@router.put("/me", response_model=CustomerProfile)
async def update_my_profile(
    payload: CustomerProfile,
    current: TokenPayload = Depends(get_current_user),
    session: AsyncSession = Depends(get_db),
) -&gt; CustomerProfile:
    """
    Allow a customer to update their own profile fields like room, floor and preferences.
    Email in the payload is ignored (comes from the authenticated user).
    """
    result = await session.execute(select(User).where(User.username == current.sub))
    user: User | None = result.scalar_one_or_none()
    if not user:
        raise HTTPException(status_code=status.HTTP_404_NOT_FOUND, detail="User not found")
    if user.role != "customer":
        raise HTTPException(status_code=status.HTTP_403_FORBIDDEN, detail="Not a customer account")

    await _ensure_profile_row(user.id, session)

    await session.execute(
        text(
            """
            UPDATE customer_profiles
            SET full_name = COALESCE(:full_name, full_name),
                room_number = COALESCE(:room_number, room_number),
                floor = COALESCE(:floor, floor),
                likes_food = COALESCE(:likes_food, likes_food),
                likes_water = COALESCE(:likes_water, likes_water),
                notes = COALESCE(:notes, notes)
            WHERE user_id = :user_id
            """
        ),
        {
```

```
        "user_id": user.id,
        "full_name": payload.full_name,
        "room_number": payload.room_number,
        "floor": payload.floor,
        "likes_food": payload.likes_food,
        "likes_water": payload.likes_water,
        "notes": payload.notes,
    },
)
await session.commit()

return await get_my_profile(current=current, session=session)
```

File: backend\app\api\routes\guests.py

Full path: C:\Users\itodo\Hostelrec\backend\app\api\routes\guests.py

```
from fastapi import APIRouter, Depends, HTTPException, status, Form
from sqlalchemy import text, select
from sqlalchemy.ext.asyncio import AsyncSession
from typing import Optional
from datetime import datetime

from app.core.security import get_current_user, require_role, TokenPayload
from app.core.audit import set_audit_user
from app.db.session import get_db
from app.models import User

router = APIRouter()


@router.post("/check-in")
async def check_in_guest(
    guest_name: str = Form(...),
    guest_email: Optional[str] = Form(None),
    guest_passport: Optional[str] = Form(None),
    room_number: Optional[str] = Form(None),
    notes: Optional[str] = Form(None),
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; dict:
    """
    Receptionist checks in a guest.
    """
    if current.role not in ("manager", "receptionist"):
        raise HTTPException(
            status_code=status.HTTP_403_FORBIDDEN, detail="Insufficient rights"
        )

    # Get the staff user record
    result = await session.execute(select(User).where(User.username == current.sub))
    staff_user = result.scalar_one_or_none()
    if not staff_user:
        raise HTTPException(status_code=404, detail="Staff user not found")

    # Set audit user for logging
    await set_audit_user(session, current.sub)

    # Insert check-in record
    result = await session.execute(
        text(
            """
            INSERT INTO guest_checkins (guest_name, guest_email, guest_passport, room_number, notes, checked
    us)
            VALUES (:guest_name, :guest_email, :guest_passport, :room_number, :notes, :checked_in_by, 'check
            RETURNING id, guest_name, guest_email, room_number, check_in_date, status
            """
        ),
        {
            "guest_name": guest_name,
            "guest_email": guest_email,
            "guest_passport": guest_passport,
            "room_number": room_number,
            "notes": notes,
            "checked_in_by": staff_user.id,
        },
    )
    row = result.mappings().first()
    await session.commit()

    return {
        "id": row["id"],
        "guest_name": row["guest_name"],
        "guest_email": row["guest_email"],
        "room_number": row["room_number"],
```

```
        "check_in_date": row["check_in_date"],
        "status": row["status"],
    }


@router.post("/{checkin_id}/check-out")
async def check_out_guest(
    checkin_id: int,
    notes: Optional[str] = Form(None),
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; dict:
    """
    Receptionist checks out a guest.
    """
    if current.role not in ("manager", "receptionist"):
        raise HTTPException(
            status_code=status.HTTP_403_FORBIDDEN, detail="Insufficient rights"
        )

    # Get the staff user record
    result = await session.execute(select(User).where(User.username == current.sub))
    staff_user = result.scalar_one_or_none()
    if not staff_user:
        raise HTTPException(status_code=404, detail="Staff user not found")

    # Set audit user for logging
    await set_audit_user(session, current.sub)

    # Update check-in record to check-out
    result = await session.execute(
        text(
            """
            UPDATE guest_checkins
            SET check_out_date = NOW(),
                checked_out_by = :checked_out_by,
                status = 'checked_out',
                notes = COALESCE(:notes, notes)
            WHERE id = :checkin_id AND status = 'checked_in'
            RETURNING id, guest_name, guest_email, room_number, check_in_date, check_out_date, status
            """
        ),
        {
            "checkin_id": checkin_id,
            "checked_out_by": staff_user.id,
            "notes": notes,
        },
    )
    row = result.mappings().first()
    if not row:
        raise HTTPException(
            status_code=404, detail="Check-in not found or already checked out"
        )

    await session.commit()
    return {
        "id": row["id"],
        "guest_name": row["guest_name"],
        "guest_email": row["guest_email"],
        "room_number": row["room_number"],
        "check_in_date": row["check_in_date"],
        "check_out_date": row["check_out_date"],
        "status": row["status"],
    }


@router.get("/")
async def list_guests(
    status_filter: Optional[str] = None,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; list[dict]:
    """
```

```
    List all guest check-ins. Receptionist and manager can view all.
    """
    if current.role not in ("manager", "receptionist"):
        raise HTTPException(
            status_code=status.HTTP_403_FORBIDDEN, detail="Insufficient rights"
        )

    query = """
        SELECT g.id,
               g.guest_name,
               g.guest_email,
               g.guest_passport,
               g.room_number,
               g.check_in_date,
               g.check_out_date,
               g.status,
               g.notes,
               u1.username AS checked_in_by_name,
               u2.username AS checked_out_by_name
        FROM guest_checkins AS g
        LEFT JOIN hostel_users AS u1 ON g.checked_in_by = u1.id
        LEFT JOIN hostel_users AS u2 ON g.checked_out_by = u2.id
    """
    params = {}

    if status_filter:
        query += " WHERE g.status = :status_filter"
        params["status_filter"] = status_filter

    query += " ORDER BY g.check_in_date DESC"

    result = await session.execute(text(query), params)
    rows = result.mappings().all()
    return [dict(row) for row in rows]


@router.get("/{checkin_id}")
async def get_guest_details(
    checkin_id: int,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; dict:
    """
    Get details of a specific guest check-in.
    """
    if current.role not in ("manager", "receptionist"):
        raise HTTPException(
            status_code=status.HTTP_403_FORBIDDEN, detail="Insufficient rights"
        )

    result = await session.execute(
        text(
            """
            SELECT g.id,
                   g.guest_name,
                   g.guest_email,
                   g.guest_passport,
                   g.room_number,
                   g.check_in_date,
                   g.check_out_date,
                   g.status,
                   g.notes,
                   u1.username AS checked_in_by_name,
                   u2.username AS checked_out_by_name
            FROM guest_checkins AS g
            LEFT JOIN hostel_users AS u1 ON g.checked_in_by = u1.id
            LEFT JOIN hostel_users AS u2 ON g.checked_out_by = u2.id
            WHERE g.id = :checkin_id
            """
        ),
        {"checkin_id": checkin_id},
    )
    row = result.mappings().first()
```

```
if not row:
    raise HTTPException(status_code=404, detail="Check-in not found")

return dict(row)
```

File: backend\app\api\routes\orders.py

Full path: C:\Users\itodo\Hostelrec\backend\app\api\routes\orders.py

```
from fastapi import APIRouter, Depends, HTTPException, status
from sqlalchemy import text, select
from sqlalchemy.ext.asyncio import AsyncSession

from app.core.security import get_current_user, TokenPayload
from app.core.audit import set_audit_user
from app.db.session import get_db
from app.models import User
from app.schemas.order import OrderCreate, OrderResponse

router = APIRouter()


async def _get_current_user_record(
    current: TokenPayload, session: AsyncSession
) -&gt; User:
    result = await session.execute(select(User).where(User.username == current.sub))
    user: User | None = result.scalar_one_or_none()
    if not user:
        raise HTTPException(status_code=status.HTTP_404_NOT_FOUND, detail="User not found")
    return user


@router.post("/", response_model=OrderResponse, status_code=status.HTTP_201_CREATED)
async def create_order(
    payload: OrderCreate,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; OrderResponse:
    """
    Customer creates a new food/drink/other order.
    """
    user = await _get_current_user_record(current, session)
    await set_audit_user(session, current.sub)

    result = await session.execute(
        text(
            """
            INSERT INTO customer_orders (user_id, item_name, category, quantity)
            VALUES (:user_id, :item_name, :category, :quantity)
            RETURNING id, item_name, category, quantity, status, created_at
            """
        ),
        {
            "user_id": user.id,
            "item_name": payload.item_name,
            "category": payload.category,
            "quantity": payload.quantity,
        },
    )
    row = result.mappings().first()
    await session.commit()

    return OrderResponse(
        id=row["id"],
        item_name=row["item_name"],
        category=row["category"],
        quantity=row["quantity"],
        status=row["status"],
        created_at=row["created_at"],
        customer_email=user.username,
    )


@router.get("/my", response_model=list[OrderResponse])
async def my_orders(
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
```

```
) -&gt; list[OrderResponse]:
    """
    List orders for the logged-in customer.
    """
    user = await _get_current_user_record(current, session)

    result = await session.execute(
        text(
            """
            SELECT o.id,
                   o.item_name,
                   o.category,
                   o.quantity,
                   o.status,
                   o.created_at,
                   u.username AS customer_email
            FROM customer_orders AS o
            JOIN hostel_users AS u ON o.user_id = u.id
            WHERE o.user_id = :user_id
            ORDER BY o.created_at DESC
            """
        ),
        {"user_id": user.id},
    )
    return [OrderResponse(**row) for row in result.mappings().all()]


@router.get("/", response_model=list[OrderResponse])
async def all_orders_for_staff(
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; list[OrderResponse]:
    """
    Staff view of all customer orders.
    Accessible to manager and receptionist roles.
    """
    if current.role not in ("manager", "receptionist"):
        raise HTTPException(
            status_code=status.HTTP_403_FORBIDDEN, detail="Insufficient rights"
        )

    result = await session.execute(
        text(
            """
            SELECT o.id,
                   o.item_name,
                   o.category,
                   o.quantity,
                   o.status,
                   o.created_at,
                   u.username AS customer_email
            FROM customer_orders AS o
            JOIN hostel_users AS u ON o.user_id = u.id
            ORDER BY o.created_at DESC
            """
        )
    )
    return [OrderResponse(**row) for row in result.mappings().all()]


@router.post("/{order_id}/status", response_model=OrderResponse)
async def update_order_status(
    order_id: int,
    new_status: str,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; OrderResponse:
    """
    Staff updates the status of an order (pending, in_progress, served, etc.).
    """
    if current.role not in ("manager", "receptionist"):
        raise HTTPException(
            status_code=status.HTTP_403_FORBIDDEN, detail="Insufficient rights"
```

```
        )

    await set_audit_user(session, current.sub)

    result = await session.execute(
        text(
            """
            UPDATE customer_orders
            SET status = :status
            WHERE id = :order_id
            RETURNING id, item_name, category, quantity, status, created_at,
                      (SELECT username FROM hostel_users WHERE id = user_id) AS customer_email
            """
        ),
        {"status": new_status, "order_id": order_id},
    )
    row = result.mappings().first()
    if not row:
        raise HTTPException(status_code=status.HTTP_404_NOT_FOUND, detail="Order not found")

    await session.commit()
    return OrderResponse(**row)


@router.post("/{order_id}/cancel", response_model=OrderResponse)
async def cancel_order(
    order_id: int,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; OrderResponse:
    """
    Cancel an order. Customers can cancel their own orders, staff can cancel any order.
    """
    user = await _get_current_user_record(current, session)

    # Check if order exists and belongs to user (if customer) or allow staff
    result = await session.execute(
        text(
            """
            SELECT user_id FROM customer_orders WHERE id = :order_id
            """
        ),
        {"order_id": order_id},
    )
    order_row = result.mappings().first()
    if not order_row:
        raise HTTPException(status_code=status.HTTP_404_NOT_FOUND, detail="Order not found")

    # Check permissions: customer can only cancel their own orders, staff can cancel any
    if current.role not in ("manager", "receptionist") and order_row["user_id"] != user.id:
        raise HTTPException(
            status_code=status.HTTP_403_FORBIDDEN, detail="You can only cancel your own orders"
        )

    await set_audit_user(session, current.sub)

    # Update order status to cancelled
    result = await session.execute(
        text(
            """
            UPDATE customer_orders
            SET status = 'cancelled'
            WHERE id = :order_id
            RETURNING id, item_name, category, quantity, status, created_at,
                      (SELECT username FROM hostel_users WHERE id = user_id) AS customer_email
            """
        ),
        {"order_id": order_id},
    )
    row = result.mappings().first()
    await session.commit()
    return OrderResponse(**row)
```

File: backend\app\api\routes\payments.py

Full path: C:\Users\itodo\Hostelrec\backend\app\api\routes\payments.py

```
from fastapi import APIRouter, Depends, HTTPException, status
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy import text
from typing import List, Optional
from pydantic import BaseModel

from app.core.security import get_current_user, require_role, TokenPayload
from app.core.audit import set_audit_user
from app.db.session import get_db

router = APIRouter()


class PaymentCreate(BaseModel):
    booking_id: Optional[int] = None
    order_id: Optional[int] = None
    amount: float
    payment_method: str
    transaction_id: Optional[str] = None
    notes: Optional[str] = None


class PaymentResponse(BaseModel):
    id: int
    booking_id: Optional[int]
    order_id: Optional[int]
    amount: float
    payment_method: str
    payment_status: str
    transaction_id: Optional[str]
    created_at: str


@router.post("/", response_model=PaymentResponse, status_code=status.HTTP_201_CREATED)
async def create_payment(
    payload: PaymentCreate,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
) -&gt; PaymentResponse:
    """
    Create a payment record for a booking or order.
    """
    await set_audit_user(session, current.sub)

    if not payload.booking_id and not payload.order_id:
        raise HTTPException(
            status_code=status.HTTP_400_BAD_REQUEST,
            detail="Either booking_id or order_id must be provided"
        )

    result = await session.execute(
        text(
            """
            INSERT INTO payments (booking_id, order_id, amount, payment_method, payment_status, transaction_
            VALUES (:booking_id, :order_id, :amount, :payment_method, 'completed', :transaction_id, :notes)
            RETURNING id, booking_id, order_id, amount, payment_method, payment_status, transaction_id, crea
            """
        ),
        {
            "booking_id": payload.booking_id,
            "order_id": payload.order_id,
            "amount": payload.amount,
            "payment_method": payload.payment_method,
            "transaction_id": payload.transaction_id,
            "notes": payload.notes,
        },
    )
    row = result.mappings().first()
```

```
    if not row:
        raise HTTPException(status_code=status.HTTP_500_INTERNAL_SERVER_ERROR, detail="Failed to create paym

    # Update booking payment status if booking_id provided
    if payload.booking_id:
        await session.execute(
            text(
                """
                UPDATE bookings
                SET payment_status = 'completed'
                WHERE id = :booking_id
                """
            ),
            {"booking_id": payload.booking_id},
        )

    await session.commit()
    return PaymentResponse(**row)


@router.get("/", response_model=List[dict])
async def list_payments(
    booking_id: Optional[int] = None,
    order_id: Optional[int] = None,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _staff=Depends(require_role("receptionist", "manager")),
) -&gt; List[dict]:
    """
    List payments. Receptionist and manager can view all payments.
    """
    query = "SELECT * FROM payments WHERE 1=1"
    params = {}

    if booking_id:
        query += " AND booking_id = :booking_id"
        params["booking_id"] = booking_id

    if order_id:
        query += " AND order_id = :order_id"
        params["order_id"] = order_id

    query += " ORDER BY created_at DESC"

    result = await session.execute(text(query), params)
    rows = result.mappings().all()
    return [dict(row) for row in rows]


@router.get("/methods", response_model=List[str])
async def get_payment_methods() -&gt; List[str]:
    """
    Get available payment methods.
    """
    return ["cash", "card", "bank_transfer", "online"]
```

File: backend\app\api\routes\staff.py

Full path: C:\Users\itodo\Hostelrec\backend\app\api\routes\staff.py

```
from fastapi import APIRouter, Depends, HTTPException, status
from sqlalchemy.ext.asyncio import AsyncSession
from sqlalchemy import select, text

from app.core.security import require_role, get_current_user, TokenPayload
from app.core.audit import set_audit_user
from app.db.session import get_db
from app.models import User
from app.schemas.auth import UserCreate, UserResponse, UserUpdate

router = APIRouter()


@router.get("/", response_model=list[UserResponse])
async def list_staff(
    session: AsyncSession = Depends(get_db),
    _manager=Depends(require_role("manager")),
):
    """List all staff members (manager only)"""
    result = await session.execute(
        select(User).where(User.role.in_(["receptionist", "manager", "cleaner"]))
    )
    users = result.scalars().all()
    return [UserResponse(id=u.id, username=u.username, role=u.role) for u in users]


@router.post("/", response_model=UserResponse, status_code=status.HTTP_201_CREATED)
async def create_staff(
    payload: UserCreate,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _manager=Depends(require_role("manager")),
):
    """Create a new staff member (manager only)"""
    from app.core.security import get_password_hash

    # Set audit user for logging
    await set_audit_user(session, current.sub)

    # Check if username exists
    result = await session.execute(select(User).where(User.username == payload.username))
    if result.scalar_one_or_none():
        raise HTTPException(status_code=400, detail="Username already exists")

    # For PostgreSQL crypt(), we need to use the crypt function directly
    # But for now, we'll store the bcrypt hash and update auth to handle both
    hashed = get_password_hash(payload.password)
    new_user = User(
        username=payload.username,
        password=hashed,  # Store bcrypt hash
        role=payload.role or "receptionist",
    )
    session.add(new_user)
    await session.commit()
    await session.refresh(new_user)
    return UserResponse(id=new_user.id, username=new_user.username, role=new_user.role)


@router.put("/{user_id}", response_model=UserResponse)
async def update_staff(
    user_id: int,
    payload: UserUpdate,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _manager=Depends(require_role("manager")),
):
    """Update staff member role (manager only)"""
    await set_audit_user(session, current.sub)
```

```
    result = await session.execute(select(User).where(User.id == user_id))
    user = result.scalar_one_or_none()
    if not user:
        raise HTTPException(status_code=404, detail="User not found")

    if payload.role:
        user.role = payload.role
    if payload.password:
        from app.core.security import get_password_hash
        user.password = get_password_hash(payload.password)

    await session.commit()
    await session.refresh(user)
    return UserResponse(id=user.id, username=user.username, role=user.role)


@router.delete("/{user_id}", status_code=status.HTTP_204_NO_CONTENT)
async def delete_staff(
    user_id: int,
    session: AsyncSession = Depends(get_db),
    current: TokenPayload = Depends(get_current_user),
    _manager=Depends(require_role("manager")),
):
    """Delete a staff member (manager only)"""
    await set_audit_user(session, current.sub)

    result = await session.execute(select(User).where(User.id == user_id))
    user = result.scalar_one_or_none()
    if not user:
        raise HTTPException(status_code=404, detail="User not found")

    await session.delete(user)
    await session.commit()
    return None


@router.post("/attendance/check-in")
async def check_in_for_today(
    current: TokenPayload = Depends(get_current_user),
    session: AsyncSession = Depends(get_db),
):
    """
    Allow the currently logged-in staff member to check in for today.
    Creates one attendance record per day per user.
    """
    # Find the User record for this token subject
    result = await session.execute(select(User).where(User.username == current.sub))
    user = result.scalar_one_or_none()
    if not user:
        raise HTTPException(status_code=404, detail="User not found")

    # Insert attendance row if not already present today
    await session.execute(
        text(
            """
            INSERT INTO staff_attendance (user_id, work_date)
            VALUES (:user_id, CURRENT_DATE)
            ON CONFLICT (user_id, work_date) DO NOTHING
            """
        ),
        {"user_id": user.id},
    )

    # Return total shifts worked
    total_result = await session.execute(
        text(
            "SELECT COUNT(*) AS total_shifts FROM staff_attendance WHERE user_id = :user_id"
        ),
        {"user_id": user.id},
    )
    total_shifts = int(total_result.scalar() or 0)

    await session.commit()
```

```
    return {"username": user.username, "total_shifts": total_shifts}


@router.get("/attendance/summary")
async def attendance_summary(
    session: AsyncSession = Depends(get_db),
    _manager=Depends(require_role("manager")),
):
    """
    Manager view: show how many days each staff member has worked.
    """
    result = await session.execute(
        text(
            """
            SELECT u.id, u.username, u.role, COALESCE(COUNT(a.id), 0) AS total_shifts
            FROM hostel_users AS u
            LEFT JOIN staff_attendance AS a ON u.id = a.user_id
            WHERE u.role IN ('receptionist', 'manager', 'cleaner')
            GROUP BY u.id, u.username, u.role
            ORDER BY u.username
            """
        )
    )
    rows = [
        {
            "id": r.id,
            "username": r.username,
            "role": r.role,
            "total_shifts": int(r.total_shifts or 0),
        }
        for r in result
    ]
    return rows
```

File: backend\app\api\routes\stats.py

Full path: C:\Users\itodo\Hostelrec\backend\app\api\routes\stats.py

```
from fastapi import APIRouter, Depends
from sqlalchemy.ext.asyncio import AsyncSession

from app.core.security import get_current_user, require_role
from app.db.session import get_db
from app.schemas.stats import OccupancyStats, RevenueSlice
from app.services.stats_service import get_occupancy, get_revenue

router = APIRouter()


@router.get("/occupancy", response_model=OccupancyStats)
async def occupancy(session: AsyncSession = Depends(get_db), _user=Depends(get_current_user)):
    return await get_occupancy(session)


@router.get("/revenue", response_model=list[RevenueSlice])
async def revenue(
    session: AsyncSession = Depends(get_db),
    _manager=Depends(require_role("manager")),
):
    return await get_revenue(session)
```

File: backend\app\core\audit.py

Full path: C:\Users\itodo\Hostelrec\backend\app\core\audit.py

```
"""Helper functions for audit logging"""
from sqlalchemy import text
from sqlalchemy.ext.asyncio import AsyncSession


async def set_audit_user(session: AsyncSession, username: str) -&gt; None:
    """
    Set the current user for audit logging in the database session.
    Uses a temporary table to store the username for the trigger to access.
    This approach works reliably across all PostgreSQL versions.
    """
    try:
        # Create a temporary table to store the current user for this transaction
        # ON COMMIT DROP ensures it's cleaned up automatically
        await session.execute(
            text("""
                CREATE TEMP TABLE IF NOT EXISTS _audit_current_user (
                    username VARCHAR(255) PRIMARY KEY
                ) ON COMMIT DROP
            """)
        )
        # Clear any existing value and insert the new one
        await session.execute(text("DELETE FROM _audit_current_user"))
        await session.execute(
            text("INSERT INTO _audit_current_user (username) VALUES (:username)"),
            {"username": username}
        )
    except Exception:
        # If setting fails, the trigger will default to 'system'
        # Don't block the request - audit logging is important but not critical
        pass
```

File: backend\app\core\config.py

Full path: C:\Users\itodo\Hostelrec\backend\app\core\config.py

```
from functools import lru_cache
from pydantic_settings import BaseSettings, SettingsConfigDict


class Settings(BaseSettings):
    """Application configuration sourced from environment variables."""

    project_name: str = "HostelRec API"
    environment: str = "development"

    database_url: str = "postgresql+asyncpg://postgres:postgres@db:5432/hostelrec"

    auth_token_expire_minutes: int = 60
    auth_algorithm: str = "HS256"
    auth_secret_key: str = "change-me"

    super_secret_phrase: str = "lab3-secret"

    model_config = SettingsConfigDict(env_file=".env", env_file_encoding="utf-8", extra="ignore")


@lru_cache
def get_settings() -&gt; Settings:
    """Provide cached settings instance."""
    return Settings()
```

File: backend\app\core\security.py

Full path: C:\Users\itodo\Hostelrec\backend\app\core\security.py

```
"""
[redacted].

[redacted], [redacted] [redacted]
JWT [redacted], [redacted] [redacted].

[redacted]:
- pbkdf2_sha256 [redacted] ([redacted])
- JWT (HS256) [redacted]
"""
from datetime import datetime, timedelta, timezone
from typing import Annotated

from fastapi import Depends, HTTPException, status
from fastapi.security import OAuth2PasswordBearer
from jose import jwt
from passlib.context import CryptContext
from pydantic import BaseModel

from app.core.config import get_settings

# [redacted] pbkdf2_sha256 - [redacted] [redacted]
# passlib [redacted]
pwd_context = CryptContext(schemes=["pbkdf2_sha256"], deprecated="auto")


def get_password_hash(password: str) -&gt; str:
    """
    [redacted].

    [redacted] pbkdf2_sha256 [redacted].
    [redacted].

    Args:
        password: [redacted]

    Returns:
        str: [redacted] ([redacted])

    Example:
        &gt;&gt;&gt; hash = get_password_hash("my_password")
        &gt;&gt;&gt; len(hash) &gt; 50  # [redacted]
        True
    """
    return pwd_context.hash(password)


def verify_password(plain_password: str, hashed_password: str) -&gt; bool:
    """
    [redacted].

    [redacted], [redacted].

    Args:
        plain_password: [redacted]
        hashed_password: [redacted]

    Returns:
        bool: True, [redacted], False [redacted]

    Example:
        &gt;&gt;&gt; hash = get_password_hash("my_password")
        &gt;&gt;&gt; verify_password("my_password", hash)
        True
        &gt;&gt;&gt; verify_password("wrong_password", hash)
        False
    """
```

```
    return pwd_context.verify(plain_password, hashed_password)

# OAuth2 ██████ ███ ██████████ ███████ ██ ██████████ Authorization
oauth2_scheme = OAuth2PasswordBearer(tokenUrl="/auth/token")


class TokenPayload(BaseModel):
    """
    ██████████ ██████████ JWT ███████.

    ██████████ ████████████ ███ ███████████████, ████████████████ ███ ███████.
    """
    sub: str  # Subject (username ███████████████)
    role: str  # ██████ ███████████████
    exp: int  # ███████ ██████████ ███████ (Unix timestamp)


def create_access_token(*, subject: str, role: str, expires_minutes: int | None = None) -&gt; str:
    """
    ████████ JWT ███████ ██████████████████.

    ████████████ ████████████████████ ████████████ ███████ ███ ██████████████████ JWT (HS256).
    ███████ ██████████ ████████████ ███ ███████████████ ███ ███████ ███████████.

    Args:
        subject: ████ ███████████████ (username)
        role: ██████ ███████████████
        expires_minutes: ███████ ███████ ███████ ███ ████████ (██ ██████████ ██ ██████████)

    Returns:
        str: JWT ███████ ███ ██████ ███████

    Example:
        &gt;&gt;&gt; token = create_access_token(subject="user@example.com", role="customer")
        &gt;&gt;&gt; len(token) &gt; 100
        True
    """
    settings = get_settings()
    expire = datetime.now(timezone.utc) + timedelta(
        minutes=expires_minutes or settings.auth_token_expire_minutes
    )
    payload = {"sub": subject, "role": role, "exp": int(expire.timestamp())}
    return jwt.encode(payload, settings.auth_secret_key, algorithm=settings.auth_algorithm)


def get_current_user(token: Annotated[str, Depends(oauth2_scheme)]) -&gt; TokenPayload:
    """
    ██████████ ██████████ ████████████████████████ ██████████████ ██ ███████.

    ████████████ ███ ██████████ JWT ███████ ██ ██████████ Authorization.
    ██████████████ ████ ████████████ FastAPI ████ ███████ ████████████.

    Args:
        token: JWT ███████ ██ ██████████ Authorization

    Returns:
        TokenPayload: ███████ ██████████████ ██ ███████

    Raises:
        HTTPException: ██████ ███████ ██████████ ████ ███████

    Example:
        ███████████████ ███ ██████████:
        &gt;&gt;&gt; @router.get("/protected")
        &gt;&gt;&gt; async def protected_route(current: TokenPayload = Depends(get_current_user)):
        &gt;&gt;&gt;     return {"username": current.sub, "role": current.role}
    """
    settings = get_settings()
    try:
        payload = jwt.decode(token, settings.auth_secret_key, algorithms=[settings.auth_algorithm])
        token_data = TokenPayload(**payload)
        return token_data
    except Exception as exc:
```

```
        raise HTTPException(
            status_code=status.HTTP_401_UNAUTHORIZED,
            detail="Could not validate credentials",
            headers={"WWW-Authenticate": "Bearer"},
        ) from exc


def require_role(*allowed_roles: str):
    """
    ██████████ ███ █████████ ████ ████████ ██ ███████ ██████.

    ████████ ████████████ FastAPI, ████████ ██████████, ██████ ██ ██████████████
    ████ ██ ████████████ ██████. ██████████████ ███ ███████ ███████████.

    Args:
        *allowed_roles: ███████ ████████████ ██████

    Returns:
        ████████-████████████ ███ FastAPI

    Example:
        ███████████████ ■ ██████████:
        &gt;&gt;&gt; @router.get("/admin-only")
        &gt;&gt;&gt; async def admin_route(
        &gt;&gt;&gt;     _admin=Depends(require_role("manager", "admin"))
        &gt;&gt;&gt; ):
        &gt;&gt;&gt;     return {"message": "Admin access granted"}
    """
    def role_dependency(user: Annotated[TokenPayload, Depends(get_current_user)]) -&gt; TokenPayload:
        if user.role not in allowed_roles:
            raise HTTPException(
                status_code=status.HTTP_403_FORBIDDEN,
                detail="Insufficient rights"
            )
        return user

    return role_dependency
```

File: backend\app\db\session.py

Full path: C:\Users\itodo\Hostelrec\backend\app\db\session.py

```
from sqlalchemy.ext.asyncio import AsyncSession, async_sessionmaker, create_async_engine

from app.core.config import get_settings

settings = get_settings()

engine = create_async_engine(settings.database_url, echo=False, future=True)

AsyncSessionLocal = async_sessionmaker(engine, expire_on_commit=False)


async def get_db() -&gt; AsyncSession:
    async with AsyncSessionLocal() as session:
        yield session
```

File: backend\app\main.py

Full path: C:\Users\itodo\Hostelrec\backend\app\main.py

```
from fastapi import FastAPI
from fastapi.middleware.cors import CORSMiddleware
from fastapi.staticfiles import StaticFiles
from sqlalchemy import select, text
from pathlib import Path

from app.api.routes import api_router
from app.core.config import get_settings
from app.core.security import get_password_hash
from app.db.session import AsyncSessionLocal
from app.models import User

settings = get_settings()

app = FastAPI(title=settings.project_name)

app.add_middleware(
    CORSMiddleware,
    allow_origins=["*"],
    allow_credentials=True,
    allow_methods=["*"],
    allow_headers=["*"],
)


@app.on_event("startup")
async def create_default_admin() -&gt; None:
    """Ensure a default manager account exists for initial login."""
    async with AsyncSessionLocal() as session:
        try:
            # Commit any pending transactions first
            await session.commit()
        except Exception:
            await session.rollback()
        # Ensure core tables exist (simple bootstrap, not a full migration system).
        # asyncpg does not allow multiple SQL commands in a single prepared statement,
        # so we execute each CREATE TABLE separately.
        await session.execute(
            text(
                """
                CREATE TABLE IF NOT EXISTS hostel_users (
                    id SERIAL PRIMARY KEY,
                    username VARCHAR(255) UNIQUE NOT NULL,
                    password VARCHAR(255) NOT NULL,
                    role VARCHAR(50) NOT NULL
                )
                """
            )
        )
        # ■■■■■■■■■ ■■■■■■■■ MFA ■ ■■■■■■■■■■■■■ ■■■■■■■■ (■■■■■ ■■ ■■■■ ■■■■)
        # ■■■■■■■■■■■ DO ■■■■■ ■■■■ ■■■■■■■■■ ■■■■■■■■■■■■■■■ ■■■■■■■■
        await session.execute(
            text("""
                DO $$
                BEGIN
                    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                                   WHERE table_name='hostel_users' AND column_name='mfa_enabled') THEN
                        ALTER TABLE hostel_users ADD COLUMN mfa_enabled BOOLEAN DEFAULT false;
                    END IF;
                    IF NOT EXISTS (SELECT 1 FROM information_schema.columns
                                   WHERE table_name='hostel_users' AND column_name='mfa_secret') THEN
                        ALTER TABLE hostel_users ADD COLUMN mfa_secret VARCHAR(255);
                    END IF;
                END $$;
            """)
        )
        await session.execute(
            text(
```

```
        """
        CREATE TABLE IF NOT EXISTS user_sessions (
            id SERIAL PRIMARY KEY,
            user_id INTEGER NOT NULL REFERENCES hostel_users(id) ON DELETE CASCADE,
            token VARCHAR(500) UNIQUE NOT NULL,
            created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
            last_activity TIMESTAMPTZ NOT NULL DEFAULT now(),
            expires_at TIMESTAMPTZ NOT NULL,
            is_active BOOLEAN DEFAULT true,
            ip_address VARCHAR(45),
            user_agent TEXT
        )
        """
    )
)
# ■■■■■■■■ ■■■■■■■■ ■■■■■■■■■ (asyncpg ■■ ■■■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■ ■ ■■■■■■ ■■■■■■■
await session.execute(
    text("CREATE INDEX IF NOT EXISTS idx_user_sessions_user_id ON user_sessions(user_id)")
)
await session.execute(
    text("CREATE INDEX IF NOT EXISTS idx_user_sessions_token ON user_sessions(token)")
)
await session.execute(
    text("CREATE INDEX IF NOT EXISTS idx_user_sessions_expires_at ON user_sessions(expires_at)")
)
await session.execute(
    text(
        """
        CREATE TABLE IF NOT EXISTS staff_attendance (
            id SERIAL PRIMARY KEY,
            user_id INTEGER NOT NULL REFERENCES hostel_users(id) ON DELETE CASCADE,
            work_date DATE NOT NULL,
            created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
            CONSTRAINT uq_staff_attendance_user_date UNIQUE (user_id, work_date)
        )
        """
    )
)
await session.execute(
    text(
        """
        CREATE TABLE IF NOT EXISTS customer_profiles (
            user_id INTEGER PRIMARY KEY REFERENCES hostel_users(id) ON DELETE CASCADE,
            full_name VARCHAR(255),
            room_number VARCHAR(50),
            floor INTEGER,
            likes_food BOOLEAN,
            likes_water BOOLEAN,
            notes TEXT,
            profile_picture_url TEXT
        )
        """
    )
)
await session.execute(
    text(
        """
        CREATE TABLE IF NOT EXISTS customer_orders (
            id SERIAL PRIMARY KEY,
            user_id INTEGER NOT NULL REFERENCES hostel_users(id) ON DELETE CASCADE,
            item_name VARCHAR(255) NOT NULL,
            category VARCHAR(50) NOT NULL,
            quantity INTEGER NOT NULL DEFAULT 1,
            status VARCHAR(50) NOT NULL DEFAULT 'pending',
            created_at TIMESTAMPTZ NOT NULL DEFAULT now()
        )
        """
    )
)
await session.execute(
    text(
        """
        CREATE TABLE IF NOT EXISTS chat_messages (
```

```
            id SERIAL PRIMARY KEY,
            user_id INTEGER NOT NULL REFERENCES hostel_users(id) ON DELETE CASCADE,
            username VARCHAR(255) NOT NULL,
            message TEXT NOT NULL,
            created_at TIMESTAMPTZ NOT NULL DEFAULT now()
        )
        """
    )
)
await session.execute(
    text(
        """
        CREATE TABLE IF NOT EXISTS room_images (
            id SERIAL PRIMARY KEY,
            room_number VARCHAR(50) NOT NULL,
            image_url TEXT NOT NULL,
            description TEXT,
            created_at TIMESTAMPTZ NOT NULL DEFAULT now()
        )
        """
    )
)
await session.execute(
    text(
        """
        CREATE TABLE IF NOT EXISTS rooms (
            id SERIAL PRIMARY KEY,
            room_number VARCHAR(50) UNIQUE NOT NULL,
            title VARCHAR(255) NOT NULL,
            description TEXT,
            price DECIMAL(10, 2) NOT NULL,
            location VARCHAR(255),
            wifi BOOLEAN DEFAULT true,
            features TEXT[],  -- Array of feature strings
            image_url TEXT,
            status VARCHAR(50) DEFAULT 'available',
            created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
            updated_at TIMESTAMPTZ NOT NULL DEFAULT now()
        )
        """
    )
)
await session.execute(
    text(
        """
        CREATE TABLE IF NOT EXISTS food_items (
            id SERIAL PRIMARY KEY,
            name VARCHAR(255) NOT NULL,
            description TEXT,
            price DECIMAL(10, 2) NOT NULL,
            image_url TEXT,
            category VARCHAR(50) DEFAULT 'food',
            available BOOLEAN DEFAULT true,
            is_tea BOOLEAN DEFAULT false,
            contains_alcohol BOOLEAN DEFAULT false,
            created_at TIMESTAMPTZ NOT NULL DEFAULT now()
        )
        """
    )
)
await session.execute(
    text(
        """
        CREATE TABLE IF NOT EXISTS drink_items (
            id SERIAL PRIMARY KEY,
            name VARCHAR(255) NOT NULL,
            description TEXT,
            price DECIMAL(10, 2) NOT NULL,
            image_url TEXT,
            category VARCHAR(50) DEFAULT 'drink',
            available BOOLEAN DEFAULT true,
            is_tea BOOLEAN DEFAULT false,
            contains_alcohol BOOLEAN DEFAULT false,
```

```
            created_at TIMESTAMPTZ NOT NULL DEFAULT now()
        )
        """
    )
)
await session.execute(
    text(
        """
        CREATE TABLE IF NOT EXISTS events (
            id SERIAL PRIMARY KEY,
            title VARCHAR(255) NOT NULL,
            description TEXT,
            event_date TIMESTAMPTZ NOT NULL,
            image_url TEXT,
            location VARCHAR(255),
            active BOOLEAN DEFAULT true,
            created_at TIMESTAMPTZ NOT NULL DEFAULT now()
        )
        """
    )
)
await session.execute(
    text(
        """
        CREATE TABLE IF NOT EXISTS promos (
            id SERIAL PRIMARY KEY,
            title VARCHAR(255) NOT NULL,
            description TEXT,
            discount_percent INTEGER,
            discount_amount DECIMAL(10, 2),
            code VARCHAR(50) UNIQUE,
            valid_from TIMESTAMPTZ NOT NULL,
            valid_until TIMESTAMPTZ NOT NULL,
            image_url TEXT,
            active BOOLEAN DEFAULT true,
            created_at TIMESTAMPTZ NOT NULL DEFAULT now()
        )
        """
    )
)
await session.execute(
    text(
        """
        CREATE TABLE IF NOT EXISTS guest_checkins (
            id SERIAL PRIMARY KEY,
            guest_name VARCHAR(255) NOT NULL,
            guest_email VARCHAR(255),
            guest_passport VARCHAR(255),
            room_number VARCHAR(50) NOT NULL,
            check_in_date TIMESTAMPTZ NOT NULL DEFAULT now(),
            check_out_date TIMESTAMPTZ,
            status VARCHAR(50) NOT NULL DEFAULT 'checked_in',
            notes TEXT,
            checked_in_by INTEGER REFERENCES hostel_users(id),
            checked_out_by INTEGER REFERENCES hostel_users(id),
            created_at TIMESTAMPTZ NOT NULL DEFAULT now()
        )
        """
    )
)
# Create guests table for bookings
await session.execute(
    text(
        """
        CREATE TABLE IF NOT EXISTS guests (
            id SERIAL PRIMARY KEY,
            name VARCHAR(255) NOT NULL,
            passport_number VARCHAR(255) NOT NULL,
            phone_number VARCHAR(50) NOT NULL,
            email VARCHAR(255),
            created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
            updated_at TIMESTAMPTZ NOT NULL DEFAULT now(),
            CONSTRAINT guests_passport_unique UNIQUE (passport_number)
```

```
        )
        """
    )
)
# Ensure unique constraint exists (in case table exists without constraint)
await session.execute(
    text(
        """
        DO $$
        BEGIN
            IF NOT EXISTS (
                SELECT 1 FROM pg_constraint
                WHERE conname = 'guests_passport_unique'
            ) THEN
                ALTER TABLE guests
                ADD CONSTRAINT guests_passport_unique UNIQUE (passport_number);
            END IF;
        END $$;
        """
    )
)
await session.commit()

# Create bookings table (must be after guests table)
await session.execute(
    text(
        """
        CREATE TABLE IF NOT EXISTS bookings (
            id SERIAL PRIMARY KEY,
            guest_id INTEGER NOT NULL REFERENCES guests(id) ON DELETE CASCADE,
            room_id INTEGER NOT NULL,
            check_in DATE NOT NULL,
            check_out DATE NOT NULL,
            status VARCHAR(50) NOT NULL DEFAULT 'pending',
            payment_method VARCHAR(50),
            payment_status VARCHAR(50) NOT NULL DEFAULT 'pending',
            total_amount DECIMAL(10, 2),
            created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
            updated_at TIMESTAMPTZ NOT NULL DEFAULT now(),
            CONSTRAINT valid_dates CHECK (check_out &gt; check_in)
        )
        """
    )
)
# Ensure constraint exists
await session.execute(
    text(
        """
        DO $$
        BEGIN
            IF NOT EXISTS (
                SELECT 1 FROM pg_constraint
                WHERE conname = 'valid_dates'
            ) THEN
                ALTER TABLE bookings
                ADD CONSTRAINT valid_dates CHECK (check_out &gt; check_in);
            END IF;
        END $$;
        """
    )
)
await session.commit()
# Create payments table
await session.execute(
    text(
        """
        CREATE TABLE IF NOT EXISTS payments (
            id SERIAL PRIMARY KEY,
            booking_id INTEGER REFERENCES bookings(id) ON DELETE CASCADE,
            order_id INTEGER REFERENCES customer_orders(id) ON DELETE CASCADE,
            amount DECIMAL(10, 2) NOT NULL,
            payment_method VARCHAR(50) NOT NULL,
            payment_status VARCHAR(50) NOT NULL DEFAULT 'pending',
```

```
            transaction_id VARCHAR(255),
            notes TEXT,
            created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
            updated_at TIMESTAMPTZ NOT NULL DEFAULT now()
        )
        """
    )
)
# Create audit log table
await session.execute(
    text(
        """
        CREATE TABLE IF NOT EXISTS hostel_audit_log (
            id SERIAL PRIMARY KEY,
            event_time TIMESTAMPTZ NOT NULL DEFAULT now(),
            username VARCHAR(255) NOT NULL DEFAULT 'system',
            action VARCHAR(50) NOT NULL,
            table_name VARCHAR(255) NOT NULL,
            record_id INTEGER,
            details TEXT
        )
        """
    )
)
# Alter existing table to add default if it doesn't have one
await session.execute(
    text(
        """
        DO $$
        BEGIN
            -- Check if default doesn't exist and add it
            IF NOT EXISTS (
                SELECT 1 FROM pg_attrdef
                WHERE adrelid = 'hostel_audit_log'::regclass
                AND adnum = (SELECT attnum FROM pg_attribute
                             WHERE attrelid = 'hostel_audit_log'::regclass
                             AND attname = 'username')
            ) THEN
                ALTER TABLE hostel_audit_log
                ALTER COLUMN username SET DEFAULT 'system';
            END IF;
        END $$;
        """
    )
)
await session.commit()

# Verify that guests and bookings tables exist
try:
    await session.execute(text("SELECT 1 FROM guests LIMIT 1"))
    await session.execute(text("SELECT 1 FROM bookings LIMIT 1"))
    print("✓ Guests and bookings tables verified successfully")
except Exception as e:
    print(f"■ Warning: Could not verify guests/bookings tables: {e}")
    # Try to create them again
    try:
        await session.execute(
            text(
                """
                CREATE TABLE IF NOT EXISTS guests (
                    id SERIAL PRIMARY KEY,
                    name VARCHAR(255) NOT NULL,
                    passport_number VARCHAR(255) NOT NULL,
                    phone_number VARCHAR(50) NOT NULL,
                    email VARCHAR(255),
                    created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
                    updated_at TIMESTAMPTZ NOT NULL DEFAULT now()
                )
                """
            )
        )
        await session.execute(
            text(
```

```
                """
                DO $$
                BEGIN
                    IF NOT EXISTS (
                        SELECT 1 FROM pg_constraint
                        WHERE conname = 'guests_passport_unique'
                    ) THEN
                        ALTER TABLE guests
                        ADD CONSTRAINT guests_passport_unique UNIQUE (passport_number);
                    END IF;
                END $$;
                """
            )
        )
        await session.execute(
            text(
                """
                CREATE TABLE IF NOT EXISTS bookings (
                    id SERIAL PRIMARY KEY,
                    guest_id INTEGER NOT NULL REFERENCES guests(id) ON DELETE CASCADE,
                    room_id INTEGER NOT NULL,
                    check_in DATE NOT NULL,
                    check_out DATE NOT NULL,
                    status VARCHAR(50) NOT NULL DEFAULT 'pending',
                    payment_method VARCHAR(50),
                    payment_status VARCHAR(50) NOT NULL DEFAULT 'pending',
                    total_amount DECIMAL(10, 2),
                    created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
                    updated_at TIMESTAMPTZ NOT NULL DEFAULT now()
                )
                """
            )
        )
        await session.commit()
        print("✓ Guests and bookings tables created successfully on retry")
    except Exception as retry_error:
        print(f"✗ Error creating guests/bookings tables on retry: {retry_error}")
        await session.rollback()

# Create audit trigger function
await session.execute(
    text(
        """
        CREATE OR REPLACE FUNCTION log_audit_event()
        RETURNS TRIGGER AS $$
        DECLARE
            current_user_name VARCHAR(255);
            action_type VARCHAR(50);
            record_details TEXT;
        BEGIN
            -- Get current username from temp table or use 'system'
            -- Initialize with 'system' as default
            current_user_name := 'system';
            -- Try to get from temp table (set by set_audit_user function)
            BEGIN
                SELECT username INTO current_user_name
                FROM _audit_current_user
                LIMIT 1;
                -- If empty string or null, use 'system'
                IF current_user_name IS NULL OR current_user_name = '' THEN
                    current_user_name := 'system';
                END IF;
            EXCEPTION WHEN OTHERS THEN
                -- If temp table doesn't exist or any error, use 'system'
                current_user_name := 'system';
            END;

            IF TG_OP = 'INSERT' THEN
                action_type := 'INSERT';
                record_details := row_to_json(NEW)::TEXT;
            ELSIF TG_OP = 'UPDATE' THEN
                action_type := 'UPDATE';
                record_details := 'OLD: ' || row_to_json(OLD)::TEXT || ' | NEW: ' || row_to_json(NEW
```

```
            ELSIF TG_OP = 'DELETE' THEN
                action_type := 'DELETE';
                record_details := row_to_json(OLD)::TEXT;
            END IF;

            -- Try to insert into audit log, but don't abort transaction if it fails
            BEGIN
                INSERT INTO hostel_audit_log (username, action, table_name, record_id, details)
                VALUES (
                    current_user_name,
                    action_type,
                    TG_TABLE_NAME,
                    COALESCE((NEW.id), (OLD.id)),
                    record_details
                );
            EXCEPTION WHEN OTHERS THEN
                -- If audit log insert fails, log the error but don't abort the transaction
                -- This ensures the main operation can complete even if audit logging fails
                RAISE WARNING 'Audit log insert failed: %', SQLERRM;
                -- Continue execution - don't re-raise the exception
            END;

            IF TG_OP = 'DELETE' THEN
                RETURN OLD;
            ELSE
                RETURN NEW;
            END IF;
        END;
        $$ LANGUAGE plpgsql;
        """
    )
)
await session.commit()

# Create triggers for key tables
tables_to_audit = [
    'rooms',
    'food_items',
    'drink_items',
    'events',
    'promos',
    'customer_orders',
    'guest_checkins',
    'guests',
    'bookings',
    'payments',
    'hostel_users'
]

for table_name in tables_to_audit:
    # Drop existing trigger if it exists
    await session.execute(
        text(f"DROP TRIGGER IF EXISTS audit_trigger_{table_name} ON {table_name}")
    )
    # Create new trigger
    await session.execute(
        text(
            f"""
            CREATE TRIGGER audit_trigger_{table_name}
            AFTER INSERT OR UPDATE OR DELETE ON {table_name}
            FOR EACH ROW
            EXECUTE FUNCTION log_audit_event()
            """
        )
    )
await session.commit()

result = await session.execute(
    select(User).where(User.username == "itodomichael00@gmail.com")
)
user = result.scalar_one_or_none()
if not user:
    admin = User(
```

```
                username="itodomichael00@gmail.com",
                password=get_password_hash("Mike07it"),
                role="manager",
            )
            session.add(admin)
            await session.commit()


app.include_router(api_router)

# Serve uploaded files
uploads_dir = Path("uploads")
uploads_dir.mkdir(exist_ok=True)
app.mount("/uploads", StaticFiles(directory="uploads"), name="uploads")


@app.get("/healthz")
async def health_check():
    return {"status": "ok", "environment": settings.environment}
```

File: backend\app\models__init__.py

Full path: C:\Users\itodo\Hostelrec\backend\app\models__init__.py

```
"""
SQLAlchemy ORM models live here. The real tables come from the existing labs,
so ORM classes can map to them without owning migrations.
"""

from .user import Base, User
from .staff import StaffAttendance
from .customer import CustomerProfile, CustomerOrder
from .chat import ChatMessage
from .room import RoomImage, Room
from .menu import FoodItem, DrinkItem
from .content import Event, Promo
from .booking import Guest, Booking, GuestCheckin
from .payment import Payment
from .audit import HostelAuditLog
from .session import UserSession

__all__ = [
    "Base",
    "User",
    "StaffAttendance",
    "CustomerProfile",
    "CustomerOrder",
    "ChatMessage",
    "RoomImage",
    "Room",
    "FoodItem",
    "DrinkItem",
    "Event",
    "Promo",
    "Guest",
    "Booking",
    "GuestCheckin",
    "Payment",
    "HostelAuditLog",
    "UserSession",
]
```

File: backend\app\models\audit.py

Full path: C:\Users\itodo\Hostelrec\backend\app\models\audit.py

```
from sqlalchemy import Column, Integer, String, Text, DateTime
from sqlalchemy.sql import func
from .user import Base


class HostelAuditLog(Base):
    __tablename__ = "hostel_audit_log"

    id = Column(Integer, primary_key=True, index=True)
    event_time = Column(DateTime(timezone=True), server_default=func.now(), nullable=False)
    username = Column(String(255), nullable=False, default="system")
    action = Column(String(50), nullable=False)
    table_name = Column(String(255), nullable=False)
    record_id = Column(Integer)
    details = Column(Text)
```

File: backend\app\models\booking.py

Full path: C:\Users\itodo\Hostelrec\backend\app\models\booking.py

```
from sqlalchemy import Column, Integer, String, Date, Numeric, ForeignKey, DateTime, Text
from sqlalchemy.orm import relationship
from sqlalchemy.sql import func
from .user import Base


class Guest(Base):
    __tablename__ = "guests"

    id = Column(Integer, primary_key=True, index=True)
    name = Column(String(255), nullable=False)
    passport_number = Column(String(255), unique=True, nullable=False)
    phone_number = Column(String(50), nullable=False)
    email = Column(String(255))
    created_at = Column(DateTime(timezone=True), server_default=func.now(), nullable=False)
    updated_at = Column(DateTime(timezone=True), server_default=func.now(), onupdate=func.now(), nullable=Fa

    bookings = relationship("Booking", back_populates="guest", cascade="all, delete-orphan")


class Booking(Base):
    __tablename__ = "bookings"

    id = Column(Integer, primary_key=True, index=True)
    guest_id = Column(Integer, ForeignKey("guests.id", ondelete="CASCADE"), nullable=False)
    room_id = Column(Integer, nullable=False)
    check_in = Column(Date, nullable=False)
    check_out = Column(Date, nullable=False)
    status = Column(String(50), nullable=False, default="pending")
    payment_method = Column(String(50))
    payment_status = Column(String(50), nullable=False, default="pending")
    total_amount = Column(Numeric(10, 2))
    created_at = Column(DateTime(timezone=True), server_default=func.now(), nullable=False)
    updated_at = Column(DateTime(timezone=True), server_default=func.now(), onupdate=func.now(), nullable=Fa

    guest = relationship("Guest", back_populates="bookings")
    payments = relationship("Payment", back_populates="booking", cascade="all, delete-orphan")


class GuestCheckin(Base):
    __tablename__ = "guest_checkins"

    id = Column(Integer, primary_key=True, index=True)
    guest_name = Column(String(255), nullable=False)
    guest_email = Column(String(255))
    guest_passport = Column(String(255))
    room_number = Column(String(50), nullable=False)
    check_in_date = Column(DateTime(timezone=True), server_default=func.now(), nullable=False)
    check_out_date = Column(DateTime(timezone=True))
    status = Column(String(50), nullable=False, default="checked_in")
    notes = Column(Text)
    checked_in_by = Column(Integer, ForeignKey("hostel_users.id"))
    checked_out_by = Column(Integer, ForeignKey("hostel_users.id"))
    created_at = Column(DateTime(timezone=True), server_default=func.now(), nullable=False)

    checked_in_by_user = relationship("User", foreign_keys=[checked_in_by], backref="checkins_performed")
    checked_out_by_user = relationship("User", foreign_keys=[checked_out_by], backref="checkouts_performed")
```

File: backend\app\models\chat.py

Full path: C:\Users\itodo\Hostelrec\backend\app\models\chat.py

```
from sqlalchemy import Column, Integer, String, Text, ForeignKey, DateTime
from sqlalchemy.orm import relationship
from sqlalchemy.sql import func
from .user import Base


class ChatMessage(Base):
    __tablename__ = "chat_messages"

    id = Column(Integer, primary_key=True, index=True)
    user_id = Column(Integer, ForeignKey("hostel_users.id", ondelete="CASCADE"), nullable=False)
    username = Column(String(255), nullable=False)
    message = Column(Text, nullable=False)
    created_at = Column(DateTime(timezone=True), server_default=func.now(), nullable=False)

    user = relationship("User", backref="chat_messages")
```

File: backend\app\models\content.py

Full path: C:\Users\itodo\Hostelrec\backend\app\models\content.py

```
from sqlalchemy import Column, Integer, String, Text, Numeric, Boolean, DateTime
from sqlalchemy.sql import func
from .user import Base


class Event(Base):
    __tablename__ = "events"

    id = Column(Integer, primary_key=True, index=True)
    title = Column(String(255), nullable=False)
    description = Column(Text)
    event_date = Column(DateTime(timezone=True), nullable=False)
    image_url = Column(Text)
    location = Column(String(255))
    active = Column(Boolean, default=True)
    created_at = Column(DateTime(timezone=True), server_default=func.now(), nullable=False)


class Promo(Base):
    __tablename__ = "promos"

    id = Column(Integer, primary_key=True, index=True)
    title = Column(String(255), nullable=False)
    description = Column(Text)
    discount_percent = Column(Integer)
    discount_amount = Column(Numeric(10, 2))
    code = Column(String(50), unique=True)
    valid_from = Column(DateTime(timezone=True), nullable=False)
    valid_until = Column(DateTime(timezone=True), nullable=False)
    image_url = Column(Text)
    active = Column(Boolean, default=True)
    created_at = Column(DateTime(timezone=True), server_default=func.now(), nullable=False)
```

File: backend\app\models\customer.py

Full path: C:\Users\itodo\Hostelrec\backend\app\models\customer.py

```
from sqlalchemy import Column, Integer, String, Boolean, Text, ForeignKey, DateTime
from sqlalchemy.orm import relationship
from sqlalchemy.sql import func
from .user import Base


class CustomerProfile(Base):
    __tablename__ = "customer_profiles"

    user_id = Column(Integer, ForeignKey("hostel_users.id", ondelete="CASCADE"), primary_key=True)
    full_name = Column(String(255))
    room_number = Column(String(50))
    floor = Column(Integer)
    likes_food = Column(Boolean)
    likes_water = Column(Boolean)
    notes = Column(Text)
    profile_picture_url = Column(Text)

    user = relationship("User", backref="customer_profile")


class CustomerOrder(Base):
    __tablename__ = "customer_orders"

    id = Column(Integer, primary_key=True, index=True)
    user_id = Column(Integer, ForeignKey("hostel_users.id", ondelete="CASCADE"), nullable=False)
    item_name = Column(String(255), nullable=False)
    category = Column(String(50), nullable=False)
    quantity = Column(Integer, nullable=False, default=1)
    status = Column(String(50), nullable=False, default="pending")
    created_at = Column(DateTime(timezone=True), server_default=func.now(), nullable=False)

    user = relationship("User", backref="orders")
```

File: backend\app\models\menu.py

Full path: C:\Users\itodo\Hostelrec\backend\app\models\menu.py

```
from sqlalchemy import Column, Integer, String, Text, Numeric, Boolean, DateTime
from sqlalchemy.sql import func
from .user import Base


class FoodItem(Base):
    __tablename__ = "food_items"

    id = Column(Integer, primary_key=True, index=True)
    name = Column(String(255), nullable=False)
    description = Column(Text)
    price = Column(Numeric(10, 2), nullable=False)
    image_url = Column(Text)
    category = Column(String(50), default="food")
    available = Column(Boolean, default=True)
    is_tea = Column(Boolean, default=False)
    contains_alcohol = Column(Boolean, default=False)
    created_at = Column(DateTime(timezone=True), server_default=func.now(), nullable=False)


class DrinkItem(Base):
    __tablename__ = "drink_items"

    id = Column(Integer, primary_key=True, index=True)
    name = Column(String(255), nullable=False)
    description = Column(Text)
    price = Column(Numeric(10, 2), nullable=False)
    image_url = Column(Text)
    category = Column(String(50), default="drink")
    available = Column(Boolean, default=True)
    is_tea = Column(Boolean, default=False)
    contains_alcohol = Column(Boolean, default=False)
    created_at = Column(DateTime(timezone=True), server_default=func.now(), nullable=False)
```

File: backend\app\models\payment.py

Full path: C:\Users\itodo\Hostelrec\backend\app\models\payment.py

```
from sqlalchemy import Column, Integer, String, Numeric, Text, ForeignKey, DateTime
from sqlalchemy.orm import relationship
from sqlalchemy.sql import func
from .user import Base


class Payment(Base):
    __tablename__ = "payments"

    id = Column(Integer, primary_key=True, index=True)
    booking_id = Column(Integer, ForeignKey("bookings.id", ondelete="CASCADE"))
    order_id = Column(Integer, ForeignKey("customer_orders.id", ondelete="CASCADE"))
    amount = Column(Numeric(10, 2), nullable=False)
    payment_method = Column(String(50), nullable=False)
    payment_status = Column(String(50), nullable=False, default="pending")
    transaction_id = Column(String(255))
    notes = Column(Text)
    created_at = Column(DateTime(timezone=True), server_default=func.now(), nullable=False)
    updated_at = Column(DateTime(timezone=True), server_default=func.now(), onupdate=func.now(), nullable=Fa

    booking = relationship("Booking", back_populates="payments")
    order = relationship("CustomerOrder", backref="payments")
```

File: backend\app\models\room.py

Full path: C:\Users\itodo\Hostelrec\backend\app\models\room.py

```
from sqlalchemy import Column, Integer, String, Text, Numeric, Boolean, ARRAY, DateTime
from sqlalchemy.sql import func
from .user import Base


class RoomImage(Base):
    __tablename__ = "room_images"

    id = Column(Integer, primary_key=True, index=True)
    room_number = Column(String(50), nullable=False)
    image_url = Column(Text, nullable=False)
    description = Column(Text)
    created_at = Column(DateTime(timezone=True), server_default=func.now(), nullable=False)


class Room(Base):
    __tablename__ = "rooms"

    id = Column(Integer, primary_key=True, index=True)
    room_number = Column(String(50), unique=True, nullable=False)
    title = Column(String(255), nullable=False)
    description = Column(Text)
    price = Column(Numeric(10, 2), nullable=False)
    location = Column(String(255))
    wifi = Column(Boolean, default=True)
    features = Column(ARRAY(String))
    image_url = Column(Text)
    status = Column(String(50), default="available")
    created_at = Column(DateTime(timezone=True), server_default=func.now(), nullable=False)
    updated_at = Column(DateTime(timezone=True), server_default=func.now(), onupdate=func.now(), nullable=Fa
```

File: backend\app\models\session.py

Full path: C:\Users\itodo\Hostelrec\backend\app\models\session.py

```
"""
[illegible].

[illegible]
[illegible].
"""
from datetime import datetime
from sqlalchemy import Column, Integer, String, DateTime, ForeignKey, Boolean
from sqlalchemy.orm import relationship
from sqlalchemy.sql import func

from .user import Base


class UserSession(Base):
    """
    [illegible].

    [illegible], [illegible].
    """
    __tablename__ = "user_sessions"

    id = Column(Integer, primary_key=True, index=True)
    user_id = Column(Integer, ForeignKey("hostel_users.id", ondelete="CASCADE"), nullable=False, index=True)
    token = Column(String, unique=True, nullable=False, index=True)  # JWT [illegible] UUID
    created_at = Column(DateTime(timezone=True), server_default=func.now(), nullable=False)
    last_activity = Column(DateTime(timezone=True), server_default=func.now(), onupdate=func.now(), nullable
    expires_at = Column(DateTime(timezone=True), nullable=False)
    is_active = Column(Boolean, default=True, nullable=False)
    ip_address = Column(String, nullable=True)  # IP [illegible]
    user_agent = Column(String, nullable=True)  # User-Agent [illegible]

    user = relationship("User", backref="sessions")
```

File: backend\app\models\staff.py

Full path: C:\Users\itodo\Hostelrec\backend\app\models\staff.py

```
from sqlalchemy import Column, Integer, Date, ForeignKey, DateTime, UniqueConstraint
from sqlalchemy.orm import relationship
from sqlalchemy.sql import func
from .user import Base


class StaffAttendance(Base):
    __tablename__ = "staff_attendance"

    id = Column(Integer, primary_key=True, index=True)
    user_id = Column(Integer, ForeignKey("hostel_users.id", ondelete="CASCADE"), nullable=False)
    work_date = Column(Date, nullable=False)
    created_at = Column(DateTime(timezone=True), server_default=func.now(), nullable=False)

    user = relationship("User", backref="attendance_records")

    __table_args__ = (
        UniqueConstraint("user_id", "work_date", name="uq_staff_attendance_user_date"),
    )
```

File: backend\app\models\user.py

Full path: C:\Users\itodo\Hostelrec\backend\app\models\user.py

```
"""
[redacted].

[redacted], [redacted], [redacted],
[redacted].
"""
from sqlalchemy import Column, Integer, String, Boolean
from sqlalchemy.orm import declarative_base

Base = declarative_base()


class User(Base):
    """
    [redacted].

    [redacted]:
    - [redacted] (username, password)
    - [redacted] (customer, receptionist, manager, cleaner)
    - [redacted]
    """
    __tablename__ = "hostel_users"  # Match the table name from Lab 3

    id = Column(Integer, primary_key=True, index=True)
    username = Column(String, unique=True, index=True, nullable=False)
    password = Column(String, nullable=False)  # [redacted] pbkdf2_sha256
    role = Column(String, nullable=False)  # receptionist, manager, cleaner, customer
    mfa_enabled = Column(Boolean, default=False, nullable=False)  # [redacted] MFA
    mfa_secret = Column(String, nullable=True)  # [redacted] MFA
```

File: backend\app\schemas__init__.py

Full path: C:\Users\itodo\Hostelrec\backend\app\schemas__init__.py

```
"""Pydantic schemas for request/response payloads."""
```

File: backend\app\schemas\auth.py

Full path: C:\Users\itodo\Hostelrec\backend\app\schemas\auth.py

```
"""
[illegible].

[illegible] Pydantic [illegible] [illegible] [illegible]
[illegible] API [illegible].
"""
from pydantic import BaseModel, EmailStr


class TokenResponse(BaseModel):
    """[illegible] [illegible] [illegible]."""
    access_token: str
    token_type: str = "bearer"
    role: str
    mfa_required: bool = False  # [illegible] [illegible] MFA


class LoginRequest(BaseModel):
    """[illegible] [illegible] [illegible] [illegible]."""
    username: str
    password: str
    mfa_code: str | None = None  # [illegible] MFA ([illegible] [illegible])


class SignupRequest(BaseModel):
    """[illegible] [illegible] [illegible] [illegible] [illegible]."""
    email: str
    full_name: str
    password: str


class PasswordChangeRequest(BaseModel):
    """[illegible] [illegible] [illegible] [illegible] [illegible]."""
    old_password: str
    new_password: str


class MFAEnableRequest(BaseModel):
    """[illegible] [illegible] [illegible] MFA."""
    password: str  # [illegible] [illegible] [illegible] [illegible]


class MFAVerifyRequest(BaseModel):
    """[illegible] [illegible] [illegible] [illegible] MFA."""
    code: str


class MFADisableRequest(BaseModel):
    """[illegible] [illegible] [illegible] MFA."""
    password: str  # [illegible] [illegible] [illegible] [illegible]


class UserCreate(BaseModel):
    """[illegible] [illegible] [illegible] [illegible] ([illegible])."""
    username: str
    password: str
    role: str | None = None


class UserUpdate(BaseModel):
    """[illegible] [illegible] [illegible] [illegible]."""
    role: str | None = None
    password: str | None = None


class UserResponse(BaseModel):
    """[illegible] [illegible] [illegible] [illegible] [illegible]."""
    id: int
```

```
    username: str
    role: str
    mfa_enabled: bool = False


class ForgotPasswordRequest(BaseModel):
    """[illegible]."""
    email: EmailStr


class ResetPasswordRequest(BaseModel):
    """[illegible]."""
    email: EmailStr
    code: str
    new_password: str
```

File: backend\app\schemas\booking.py

Full path: C:\Users\itodo\Hostelrec\backend\app\schemas\booking.py

```
from datetime import date
from typing import Optional
from pydantic import BaseModel, Field


class BookingCreate(BaseModel):
    check_in: date
    check_out: date
    room_id: int
    guest_name: str
    guest_passport: str = Field(min_length=6)
    phone_number: str = Field(..., min_length=10, description="Guest phone number")
    payment_method: str = Field(..., description="Payment method: cash, card, bank_transfer, online")


class BookingResponse(BaseModel):
    booking_id: int
    room_id: int
    guest_id: int
    check_in: date
    check_out: date
    payment_method: Optional[str] = None
    payment_status: Optional[str] = None
    total_amount: Optional[float] = None
```

File: backend\app\schemas\customer.py

Full path: C:\Users\itodo\Hostelrec\backend\app\schemas\customer.py

```
from pydantic import BaseModel


class CustomerProfile(BaseModel):
  email: str
  full_name: str | None = None
  room_number: str | None = None
  floor: int | None = None
  likes_food: bool | None = None
  likes_water: bool | None = None
  notes: str | None = None
```

File: backend\app\schemas\order.py

Full path: C:\Users\itodo\Hostelrec\backend\app\schemas\order.py

```
from datetime import datetime
from pydantic import BaseModel


class OrderCreate(BaseModel):
    item_name: str
    category: str
    quantity: int = 1


class OrderResponse(BaseModel):
    id: int
    item_name: str
    category: str
    quantity: int
    status: str
    created_at: datetime
    customer_email: str
```

File: backend\app\schemas\stats.py

Full path: C:\Users\itodo\Hostelrec\backend\app\schemas\stats.py

```
from pydantic import BaseModel


class OccupancyStats(BaseModel):
    occupancy_rate: float
    available_rooms: int
    occupied_rooms: int


class RevenueSlice(BaseModel):
    room_type: str
    revenue: float
```

File: backend\app\services__init__.py

Full path: C:\Users\itodo\Hostelrec\backend\app\services__init__.py

```
"""Service-layer helpers for orchestrating DB logic."""
```

File: backend\app\services\admin_service.py

Full path: C:\Users\itodo\Hostelrec\backend\app\services\admin_service.py

```
from sqlalchemy import text
from sqlalchemy.exc import ProgrammingError
from sqlalchemy.ext.asyncio import AsyncSession


async def fetch_audit_logs(session: AsyncSession) -&gt; list[dict]:
    """
    Fetch recent audit logs.
    If the underlying table doesn't exist yet (lab DB not initialized),
    return an empty list instead of crashing the API.
    """
    try:
        result = await session.execute(
            text("select * from hostel_audit_log order by event_time desc limit 200")
        )
        return [dict(row) for row in result.mappings().all()]
    except ProgrammingError as exc:
        # Gracefully handle missing audit table in lab environments
        if "hostel_audit_log" in str(exc.orig):
            return []
        raise


async def fetch_api_keys(session: AsyncSession, *, super_secret: str) -&gt; list[dict]:
    result = await session.execute(
        text(
            """
            select integration_name,
                   pgp_sym_decrypt(api_key::bytea, :super_secret) as api_key
            from api_integration_keys
            order by integration_name
            """
        ),
        {"super_secret": super_secret},
    )
    return [dict(row) for row in result.mappings().all()]
```

File: backend\app\services\booking_service.py

Full path:
C:\Users\itodo\Hostelrec\backend\app\services\booking_service.py

```
from sqlalchemy import text
from sqlalchemy.ext.asyncio import AsyncSession

from app.schemas.booking import BookingCreate, BookingResponse
from app.core.audit import set_audit_user


async def create_booking(session: AsyncSession, payload: BookingCreate, username: str = "system") -&gt; Book
    """
    Create guest + booking atomically.
    Tries stored procedure first, falls back to direct SQL if it doesn't exist.
    """
    # Set audit user for logging
    await set_audit_user(session, username)
    # Try stored procedure first
    try:
        result = await session.execute(
            text(
                """
                select * from create_guest_and_booking(
                    :guest_name,
                    :guest_passport,
                    :room_id,
                    :check_in,
                    :check_out
                )
                """
            ),
            payload.model_dump(),
        )
        row = result.mappings().first()
        if row:
            await session.commit()
            return BookingResponse(**row)
    except Exception:
        # Stored procedure doesn't exist, rollback and use fallback
        await session.rollback()

    # Fallback: Create booking directly (simplified version)
    # Note: This assumes you have a bookings table
    try:
        # First, get or create guest
        guest_result = await session.execute(
            text(
                """
                INSERT INTO guests (name, passport_number, phone_number, email)
                VALUES (:guest_name, :guest_passport, :phone_number, :guest_email)
                ON CONFLICT (passport_number) DO UPDATE
                    SET name = EXCLUDED.name,
                        phone_number = EXCLUDED.phone_number,
                        email = COALESCE(EXCLUDED.email, guests.email),
                        updated_at = now()
                RETURNING id
                """
            ),
            {
                "guest_name": payload.guest_name,
                "guest_passport": payload.guest_passport,
                "phone_number": payload.phone_number,
                "guest_email": getattr(payload, 'guest_email', None)
            },
        )
        guest_row = guest_result.mappings().first()
        if not guest_row:
            raise ValueError("Failed to create guest")
        guest_id = guest_row["id"]
```

```
        # Get room price to calculate total
        room_result = await session.execute(
            text("SELECT price FROM rooms WHERE id = :room_id"),
            {"room_id": payload.room_id}
        )
        room = room_result.mappings().first()
        room_price = room["price"] if room else 0.0

        # Calculate total amount (price per night * number of nights)
        check_in_date = payload.check_in
        check_out_date = payload.check_out
        nights = (check_out_date - check_in_date).days
        total_amount = float(room_price) * nights if nights &gt; 0 else float(room_price)

        # Create booking
        booking_result = await session.execute(
            text(
                """
                INSERT INTO bookings (guest_id, room_id, check_in, check_out, payment_method, payment_status
nt)
                VALUES (:guest_id, :room_id, :check_in, :check_out, :payment_method, 'pending', :total_amoun
                RETURNING id, room_id, guest_id, check_in, check_out
                """
            ),
            {
                "guest_id": guest_id,
                "room_id": payload.room_id,
                "check_in": payload.check_in,
                "check_out": payload.check_out,
                "payment_method": payload.payment_method,
                "total_amount": total_amount,
            },
        )
        booking_row = booking_result.mappings().first()
        if not booking_row:
            raise ValueError("Failed to create booking")

        await session.commit()
        return BookingResponse(
            booking_id=booking_row["id"],
            room_id=booking_row["room_id"],
            guest_id=booking_row["guest_id"],
            check_in=booking_row["check_in"],
            check_out=booking_row["check_out"],
        )
    except Exception as e:
        await session.rollback()
        error_msg = str(e).lower()
        # If tables don't exist, provide a helpful error
        if "does not exist" in error_msg or "relation" in error_msg or "table" in error_msg:
            # Try to verify tables exist
            try:
                await session.execute(text("SELECT 1 FROM guests LIMIT 1"))
                await session.execute(text("SELECT 1 FROM bookings LIMIT 1"))
                # Tables exist, so the error is something else
                raise ValueError(f"Failed to create booking: {str(e)}")
            except Exception as table_check_error:
                table_error_msg = str(table_check_error).lower()
                if "does not exist" in table_error_msg or "relation" in table_error_msg:
                    raise ValueError(
                        "Booking tables not set up. The guests and bookings tables do not exist. "
                        "Please restart the backend server to create them automatically. "
                        f"Error details: {str(e)}"
                    )
                # Tables exist but there's another error
                raise ValueError(f"Failed to create booking: {str(e)}")
        raise ValueError(f"Failed to create booking: {str(e)}")
```

File: backend\app\services\mfa_service.py

Full path: C:\Users\itodo\Hostelrec\backend\app\services\mfa_service.py

```
"""
[illegible] [illegible] [illegible] (MFA).

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
[illegible] [illegible] [illegible] [illegible], [illegible] [illegible] email.
"""
import secrets
import string
from datetime import datetime, timedelta, timezone
from typing import Optional
from sqlalchemy import select
from sqlalchemy.ext.asyncio import AsyncSession

from app.models.user import User


def generate_mfa_code(length: int = 6) -&gt; str:
    """
    [illegible] [illegible] [illegible] [illegible] MFA.

    Args:
        length: [illegible] [illegible] ([illegible] [illegible] 6 [illegible])

    Returns:
        str: [illegible] [illegible]
    """
    return ''.join(secrets.choice(string.digits) for _ in range(length))


def generate_mfa_secret() -&gt; str:
    """
    [illegible] [illegible] [illegible] [illegible] MFA.

    Returns:
        str: [illegible] [illegible]
    """
    return secrets.token_urlsafe(32)


# [illegible] [illegible] [illegible] ([illegible] [illegible] [illegible] [illegible] Redis)
_mfa_codes: dict[str, dict] = {}


async def send_mfa_code_email(user_email: str, code: str) -&gt; bool:
    """
    [illegible] [illegible] MFA [illegible] email [illegible].

    [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] email-[illegible].
    [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

    Args:
        user_email: Email [illegible]
        code: [illegible] [illegible]

    Returns:
        bool: True, [illegible] [illegible] [illegible] [illegible]
    """
    # [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] email
    # [illegible], [illegible] SendGrid, AWS SES, [illegible] SMTP
    print(f"[MFA] [illegible] [illegible] {user_email}: {code}")
    return True


async def create_mfa_code(
    db: AsyncSession,
    user_id: int,
    user_email: str,
) -&gt; str:
```

```
    """
    ████████ ■ ██████████ ███ MFA ██████████████.

    Args:
        db: ███████ █████ ███████
        user_id: ID ██████████████
        user_email: Email ██████████████

    Returns:
        str: ████████████████ ███
    """
    code = generate_mfa_code()
    expires_at = datetime.now(timezone.utc) + timedelta(minutes=10)

    # ██████████ ███ ██ ██████████ ██████████
    _mfa_codes[str(user_id)] = {
        "code": code,
        "expires_at": expires_at,
        "used": False,
    }

    # ██████████ ███ ██ email
    await send_mfa_code_email(user_email, code)

    return code


async def verify_mfa_code(
    db: AsyncSession,
    user_id: int,
    code: str,
) -&gt; bool:
    """
    ██████████ ███ MFA.

    Args:
        db: ███████ █████ ███████
        user_id: ID ██████████████
        code: ██████████ ███

    Returns:
        bool: True, ████ ███ ███████ ■ ██ ██████
    """
    user_id_str = str(user_id)

    if user_id_str not in _mfa_codes:
        return False

    mfa_data = _mfa_codes[user_id_str]

    # ██████████, ██ ████████████ ██ ███
    if mfa_data["used"]:
        return False

    # ██████████, ██ ██████ ██ ███
    if datetime.now(timezone.utc) &gt; mfa_data["expires_at"]:
        del _mfa_codes[user_id_str]
        return False

    # ██████████ ███
    if mfa_data["code"] != code:
        return False

    # █████████ ███ ███ ████████████████
    mfa_data["used"] = True

    return True


async def enable_mfa(
    db: AsyncSession,
    user_id: int,
) -&gt; str:
```

```
    """
    ████████ MFA ███ ████████████.

    Args:
        db: ██████ ████ ██████
        user_id: ID ████████████

    Returns:
        str: ██████████ ████ MFA
    """
    result = await db.execute(select(User).where(User.id == user_id))
    user = result.scalar_one_or_none()

    if not user:
        raise ValueError("User not found")

    secret = generate_mfa_secret()

    # ████████ ████████████ (████ ████ ████ mfa_secret ■ ██████)
    # ■ ████████ ██████████ ██████ ████████ ███ ████ ■ ██████ User
    # user.mfa_enabled = True
    # user.mfa_secret = secret

    await db.commit()

    return secret


async def disable_mfa(
    db: AsyncSession,
    user_id: int,
) -&gt; bool:
    """
    ██████████ MFA ███ ████████████.

    Args:
        db: ██████ ████ ██████
        user_id: ID ████████████

    Returns:
        bool: True, ████ MFA ███████ ████████
    """
    result = await db.execute(select(User).where(User.id == user_id))
    user = result.scalar_one_or_none()

    if not user:
        return False

    # ████████ ████████████
    # user.mfa_enabled = False
    # user.mfa_secret = None

    await db.commit()

    # ███████ ███ ████ ███ ██████ ████████████
    user_id_str = str(user_id)
    if user_id_str in _mfa_codes:
        del _mfa_codes[user_id_str]

    return True
```

File: backend\app\services\password_reset_service.py

Full path:
C:\Users\itodo\Hostelrec\backend\app\services\password_reset_service.py

```
"""
[illegible].

[illegible] [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
[illegible] [illegible] [illegible], [illegible] [illegible] email.
"""
import secrets
import string
from datetime import datetime, timedelta, timezone
from typing import Optional
from sqlalchemy import select
from sqlalchemy.ext.asyncio import AsyncSession

from app.models.user import User


def generate_reset_code(length: int = 8) -&gt; str:
    """
    [illegible] [illegible] [illegible] [illegible] [illegible].

    Args:
        length: [illegible] [illegible] ([illegible] [illegible] 8 [illegible])

    Returns:
        str: [illegible] [illegible]
    """
    # [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
    chars = string.ascii_uppercase + string.digits
    return ''.join(secrets.choice(chars) for _ in range(length))


# [illegible] [illegible] [illegible] [illegible] ([illegible] [illegible] [illegible] [illegible] Redis)
_password_reset_codes: dict[str, dict] = {}


async def send_password_reset_email(user_email: str, code: str) -&gt; bool:
    """
    [illegible] [illegible] [illegible] [illegible] email [illegible].

    [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] email-[illegible].
    [illegible] [illegible] [illegible] [illegible] [illegible].

    Args:
        user_email: Email [illegible]
        code: [illegible] [illegible]

    Returns:
        bool: True, [illegible] [illegible]
    """
    # [illegible] [illegible] [illegible] [illegible] [illegible] [illegible] email
    # [illegible], [illegible] SendGrid, AWS SES, [illegible] SMTP
    print(f"[Password Reset] [illegible] [illegible] [illegible] {user_email}: {code}")
    print(f"[Password Reset] [illegible] [illegible] [illegible]: /reset-password?code={code}&amp;email={user_em
    return True


async def create_password_reset_code(
    db: AsyncSession,
    user_email: str,
) -&gt; Optional[str]:
    """
    [illegible] [illegible] [illegible] [illegible] [illegible] [illegible].

    Args:
        db: [illegible] [illegible]
        user_email: Email [illegible] (username)
```

```
    Returns:
        str: ████████████████ ███ ███ None, ████ ████████████ ██ ███████
    """
    # ██████ ████████████ ██ email (username)
    result = await db.execute(select(User).where(User.username == user_email))
    user = result.scalar_one_or_none()

    if not user:
        # ██ ██████████, ██████████ ██ ████████████ (████████████)
        return None

    code = generate_reset_code()
    expires_at = datetime.now(timezone.utc) + timedelta(minutes=30)  # ███ ████████████ 30 ██████

    # ██████████ ████ ██ ██████████ ██████████
    _password_reset_codes[user_email] = {
        "code": code,
        "expires_at": expires_at,
        "used": False,
        "user_id": user.id,
    }

    # ██████████ ████ ██ email
    await send_password_reset_email(user_email, code)

    return code


async def verify_password_reset_code(
    db: AsyncSession,
    user_email: str,
    code: str,
) -&gt; Optional[int]:
    """
    ██████████ ████ ████████████████ ███████.

    Args:
        db: ███████ ████ ███████
        user_email: Email ██████████████
        code: ██████████ ████

    Returns:
        int: ID ██████████████, ████ ███ ███████, ██████ None
    """
    if user_email not in _password_reset_codes:
        return None

    reset_data = _password_reset_codes[user_email]

    # ██████████, ██ ████████████ ██ ███
    if reset_data["used"]:
        return None

    # ██████████, ██ ██████ ██ ███
    if datetime.now(timezone.utc) &gt; reset_data["expires_at"]:
        del _password_reset_codes[user_email]
        return None

    # ██████████ ████
    if reset_data["code"] != code:
        return None

    # █████████ ███ ███ ████████████████
    reset_data["used"] = True

    return reset_data["user_id"]


async def cleanup_expired_reset_codes() -&gt; int:
    """
    ████████ ████ ██████████ ████ ████████████████ ███████.

    Returns:
```

```
    int: ■■■■■■■■■■ ■■■■■■■■■ ■■■■■
"""
now = datetime.now(timezone.utc)
expired_emails = [
    email for email, data in _password_reset_codes.items()
    if now &gt; data["expires_at"]
]

for email in expired_emails:
    del _password_reset_codes[email]

return len(expired_emails)
```

File: backend\app\services\session_service.py

Full path:
C:\Users\itodo\Hostelrec\backend\app\services\session_service.py

```
"""
[redacted].

[redacted], [redacted]
[redacted]
"""
from datetime import datetime, timedelta, timezone
from typing import Optional
from sqlalchemy import select, delete
from sqlalchemy.ext.asyncio import AsyncSession

from app.models.session import UserSession
from app.models.user import User


async def create_session(
    db: AsyncSession,
    user_id: int,
    token: str,
    expires_minutes: int = 60,
    ip_address: Optional[str] = None,
    user_agent: Optional[str] = None,
) -&gt; UserSession:
    """
    [redacted].

    Args:
        db: [redacted]
        user_id: ID [redacted]
        token: [redacted] (JWT [redacted] UUID)
        expires_minutes: [redacted]
        ip_address: IP [redacted] ([redacted])
        user_agent: User-Agent [redacted] ([redacted])

    Returns:
        UserSession: [redacted]
    """
    expires_at = datetime.now(timezone.utc) + timedelta(minutes=expires_minutes)

    session = UserSession(
        user_id=user_id,
        token=token,
        expires_at=expires_at,
        ip_address=ip_address,
        user_agent=user_agent,
        is_active=True,
    )

    db.add(session)
    await db.commit()
    await db.refresh(session)

    return session


async def update_session_activity(
    db: AsyncSession,
    token: str,
) -&gt; Optional[UserSession]:
    """
    [redacted].

    Args:
        db: [redacted]
        token: [redacted]

    Returns:
```

```
        UserSession: ████████████ ███████ ███ None, ████ ███████ ██ ████████
    """
    result = await db.execute(
        select(UserSession).where(
            UserSession.token == token,
            UserSession.is_active == True,
            UserSession.expires_at &gt; datetime.now(timezone.utc)
        )
    )
    session = result.scalar_one_or_none()

    if session:
        session.last_activity = datetime.now(timezone.utc)
        await db.commit()
        await db.refresh(session)

    return session


async def end_session(
    db: AsyncSession,
    token: str,
) -&gt; bool:
    """
    ██████████ ███████ ████████████ (███████████████ ██████).

    Args:
        db: ███████ ████ ███████
        token: ██████ ████████████████

    Returns:
        bool: True, ████ ███████ ████ ████████ ██████████
    """
    result = await db.execute(
        select(UserSession).where(UserSession.token == token)
    )
    session = result.scalar_one_or_none()

    if session:
        session.is_active = False
        await db.commit()
        return True

    return False


async def end_all_user_sessions(
    db: AsyncSession,
    user_id: int,
) -&gt; int:
    """
    ██████████ ███ ████████ ███████ ████████████.

    Args:
        db: ███████ ████ ███████
        user_id: ID ████████████

    Returns:
        int: ██████████ ██████████ ███████
    """
    result = await db.execute(
        select(UserSession).where(
            UserSession.user_id == user_id,
            UserSession.is_active == True
        )
    )
    sessions = result.scalars().all()

    count = 0
    for session in sessions:
        session.is_active = False
        count += 1
```

```
    await db.commit()
    return count


async def cleanup_expired_sessions(db: AsyncSession) -&gt; int:
    """
    ████████ ████ ██████████ ███████ ██ ██████ ███████.

    Args:
        db: ███████ ██████ ███████

    Returns:
        int: ████████████ ██████████ ███████
    """
    result = await db.execute(
        delete(UserSession).where(
            UserSession.expires_at &lt; datetime.now(timezone.utc)
        )
    )
    await db.commit()
    return result.rowcount


async def get_active_sessions_count(
    db: AsyncSession,
    user_id: int,
) -&gt; int:
    """
    ██████████ ████████████ ██████████ ███████ ███████████████.

    Args:
        db: ███████ ██████ ███████
        user_id: ID ███████████████

    Returns:
        int: ████████████ ██████████ ███████
    """
    result = await db.execute(
        select(UserSession).where(
            UserSession.user_id == user_id,
            UserSession.is_active == True,
            UserSession.expires_at &gt; datetime.now(timezone.utc)
        )
    )
    sessions = result.scalars().all()
    return len(list(sessions))
```

File: backend\app\services\stats_service.py

Full path: C:\Users\itodo\Hostelrec\backend\app\services\stats_service.py

```
from collections.abc import Sequence
from sqlalchemy import text
from sqlalchemy.ext.asyncio import AsyncSession

from app.schemas.stats import OccupancyStats, RevenueSlice


async def get_occupancy(session: AsyncSession) -&gt; OccupancyStats:
    # Try to use the view if it exists, otherwise calculate from rooms table
    try:
        result = await session.execute(text("select * from v_room_occupancy_today limit 1"))
        row = result.mappings().first()
        if row:
            return OccupancyStats(**row)
    except Exception:
        # View doesn't exist, rollback and calculate from rooms table
        await session.rollback()

    # Fallback: Calculate occupancy from rooms table
    try:
        result = await session.execute(
            text("""
                SELECT
                    COUNT(*) FILTER (WHERE status = 'available') as available_rooms,
                    COUNT(*) FILTER (WHERE status = 'occupied') as occupied_rooms,
                    COUNT(*) as total_rooms
                FROM rooms
            """)
        )
        row = result.mappings().first()
        if not row:
            # Return default values if no rooms exist
            return OccupancyStats(
                occupancy_rate=0.0,
                available_rooms=0,
                occupied_rooms=0
            )

        total = row["total_rooms"] or 0
        occupied = row["occupied_rooms"] or 0
        available = row["available_rooms"] or 0

        occupancy_rate = (occupied / total) if total &gt; 0 else 0.0

        return OccupancyStats(
            occupancy_rate=occupancy_rate,
            available_rooms=available,
            occupied_rooms=occupied
        )
    except Exception:
        # If fallback also fails, rollback and return defaults
        await session.rollback()
        return OccupancyStats(
            occupancy_rate=0.0,
            available_rooms=0,
            occupied_rooms=0
        )


async def get_revenue(session: AsyncSession) -&gt; Sequence[RevenueSlice]:
    # Try to use the view if it exists, otherwise return empty list
    try:
        result = await session.execute(text("select * from v_revenue_by_room_type"))
        rows = result.mappings().all()
        if rows:
            return [RevenueSlice(**row) for row in rows]
    except Exception:
        # View doesn't exist, rollback and return empty list
```

```
    await session.rollback()

# Fallback: Return empty list if view doesn't exist
return []
```

File: backend\AUTHENTICATION_SYSTEM.md

Full path: C:\Users\itodo\Hostelrec\backend\AUTHENTICATION_SYSTEM.md

```
# ████████ ████████████████ ■ ████████████

## █████████

█████ █████████ ██████████ ████████████ ████████ ████████████████ ■ ████████████ ████ ████████████

## ███████████████ ████████████

### 1. ██████ ███████ ████ ██████████ ████████████████ ■

**████████ `hostel_users`:**
- `id` - ████████████ ████████████████ ████████████████
- `username` - ████ ████████████████ (████████████)
- `password` - ████████████████ ████████
- `role` - ██████ ████████████████ (customer, receptionist, manager, cleaner)
- `mfa_enabled` - ██████ ████████████ ████████████████████ ████████████████████
- `mfa_secret` - ████████████ ██████ ████ MFA

**████████ `user_sessions`:**
- `id` - ████████████ ████████████████ ████████
- `user_id` - ████████ ██ ████████████████
- `token` - ███████ ████████████████████ (JWT)
- `created_at` - ███████ ██████████ ████████
- `last_activity` - ███████ ████████████ ████████████
- `expires_at` - ███████ ████████████ ████████
- `is_active` - ██████ ████████████ ████████
- `ip_address` - IP ███████ ████████████████
- `user_agent` - User-Agent ██████████

### 2. ████████ ████████████ ████████ ■

**██████:** `backend/app/core/security.py`

**████████:**
- `get_password_hash(password: str) -&gt; str` - ██████████ ████████ ■ ████████████████ pbkdf2_sha256
- `verify_password(plain_password: str, hashed_password: str) -&gt; bool` - ████████████ ████████ ████████ ■■

**██████████████:**
- ████████████ ██████████ pbkdf2_sha256 (████████████ ██████████ ■ ████████████████ ███████)
- ████████ ████████ ████████████ ■ ████████████ ███████
- ██████ ████████████████ ████████████████ ████████████ passlib

### 3. ████████ ████████████████ ■

**██████:** `backend/app/api/routes/auth.py`

**████████████:**
- `POST /auth/token` - ██████ ■ ████████
- `POST /auth/signup` - ████████████ ████████ ████████████████

**████████████████████:**
- ██████████ ███████ ████████████████ ■ ████████
- ████████████ JWT ████████ ████ ██████████ ████████████████████
- ██████████ ████████ ████████████████
- ████████████ ████████████████ ████████████████

**████████ ████████:**
```bash
curl -X POST "http://localhost:8000/auth/token" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "username=user@example.com&password=password123"
```

████████ ████████:
```json
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "bearer",
```
```

```
  "role": "customer",
  "mfa_required": false
}
```

4. ■■■■■■■ ■■■■■■■■■■■■ ■

■■■■■: `backend/app/core/security.py`

■■■■■■■■:
- `get_current_user(token)` - ■■■■■■■■■■ ■■■■■■■■■ ■■■■■■■■■■■■■ ■■ ■■■■■■■
- `require_role(*allowed_roles)` - ■■■■■■■■■■ ■■■■ ■■■■■■■■■ ■■■■■ ■■■■■■■■

■■■■■■■■■■■■■■:
```python
@router.get("/admin-only")
async def admin_route(
    _admin=Depends(require_role("manager", "admin"))
):
    return {"message": "Admin access granted"}
```

■■■■■ ■■■■■■■■■■■■■■■:
- `customer` - ■■■■■■■■ ■■■■■■■
- `receptionist` - ■■■■■■■■■■■■■■■ ■■■■■■■■■■
- `manager` - ■■■■■■■■■■
- `cleaner` - ■■■■■■■■

5. ■■■■■■■ ■■■■■■■■■■■ ■■■■■■■■■ ■

■■■■■: `backend/app/services/session_service.py`

■■■■■■■■:
- `create_session()` - ■■■■■■■■ ■■■■■■ ■■■■■■■ ■■■■■■■■■■■■■■■
- `update_session_activity()` - ■■■■■■■■■ ■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■■■
- `end_session()` - ■■■■■■■■■■■ ■■■■■■■■ (■■■■■■■■■■■■■■■■ ■■■■■■■)
- `end_all_user_sessions()` - ■■■■■■■■■■■ ■■■■ ■■■■■■■■ ■■■■■■■■■■■■■■■
- `cleanup_expired_sessions()` - ■■■■■■■■■ ■■■■■■■■■ ■■■■■■■■
- `get_active_sessions_count()` - ■■■■■■■■■■ ■■■■■■■■■■■ ■■■■■■■■■ ■■■■■■■■

■■■■■■■■: `backend/app/models/session.py`

■■■■■■■■■■■:
- `POST /auth/logout` - ■■■■■■ ■■ ■■■■■■■■ (■■■■■■■■■■■■ ■■■■■ ■■■■■■■■)

6. ■■■■■■■■■■■■■■■■ ■■■■■■■■ ■

■■■■■■ ■■■■■■■ ■■■■■■■■■■■■■■■■

■■■■■■■■■■: `POST /auth/change-password`

■■■■■■■■:
```json
{
  "old_password": "old_password123",
  "new_password": "new_password456"
}
```

■■■■■■■■■■■■■:
- ■■■■■■■■ ■■■■■■■■■■■■■■■■■■
- ■■■■■■■■■■■ ■■■■■■■■ ■■■■■■■■ ■■■■■■ ■■■■■■■■
- ■■■■■■ ■■■■■■ ■■■■■■■■ ■■■■ ■■■■■■■■■■ ■■■■■■■■ ■■■■■■■■■■■■■

■■■■■■■■■■■■■■■■ ■■■■■■■■■■■■■■■■ (MFA)

■■■■■: `backend/app/services/mfa_service.py`

■■■■■■■■■■■:
- `POST /auth/mfa/enable` - ■■■■■■■■■■ MFA
- `POST /auth/mfa/verify` - ■■■■■■■■■■■ ■■■■ MFA
- `POST /auth/mfa/disable` - ■■■■■■■■■■■ MFA

[redacted]:
- [redacted] [redacted] [redacted] (6 [redacted])
- [redacted] [redacted] [redacted] email ([redacted]-[redacted] [redacted] [redacted])
- [redacted] [redacted] [redacted]: 10 [redacted]
- [redacted] [redacted] [redacted] [redacted]

[redacted]:
```bash
# [redacted] MFA
curl -X POST "http://localhost:8000/auth/mfa/enable" \
  -H "Authorization: Bearer &lt;token&gt;" \
  -H "Content-Type: application/json" \
  -d '{"password": "user_password"}'

# [redacted] [redacted] MFA
curl -X POST "http://localhost:8000/auth/mfa/verify" \
  -H "Authorization: Bearer &lt;token&gt;" \
  -H "Content-Type: application/json" \
  -d '{"code": "123456"}'
```

[redacted]

[redacted] [redacted]
- [redacted] [redacted] pbkdf2_sha256
- [redacted] [redacted] [redacted] [redacted] [redacted] [redacted]
- [redacted] [redacted] [redacted] [redacted] [redacted]

[redacted] [redacted]
- [redacted] JWT (JSON Web Tokens)
- [redacted] [redacted]: HS256
- [redacted] [redacted] [redacted] [redacted]
- [redacted] [redacted] [redacted] [redacted]

[redacted] [redacted]
- [redacted] [redacted] [redacted] [redacted]
- [redacted] [redacted] [redacted] [redacted]
- [redacted] [redacted] [redacted] [redacted] [redacted]

[redacted] [redacted]
- [redacted] [redacted] [redacted] (rate limiting [redacted])
- [redacted] [redacted] [redacted]
- [redacted] [redacted] [redacted] [redacted]

[redacted]

[redacted] [redacted]

[redacted] [redacted] [redacted] [redacted] [redacted] [redacted]:

1. **[redacted] [redacted] [redacted] signup:**
```bash
curl -X POST "http://localhost:8000/auth/signup" \
  -H "Content-Type: application/json" \
  -d '{
    "email": "test@example.com",
    "full_name": "Test User",
    "password": "testpassword123"
  }'
```

2. **[redacted] [redacted]:**
```bash
curl -X POST "http://localhost:8000/auth/token" \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "username=test@example.com&amp;password=testpassword123"
```

3. **[redacted] [redacted] ([redacted] [redacted]):**
```bash
curl -X POST "http://localhost:8000/auth/change-password" \
  -H "Authorization: Bearer &lt;token&gt;" \
  -H "Content-Type: application/json" \
```

```
  -d '{
    "old_password": "testpassword123",
    "new_password": "newpassword456"
  }'
```

■■■■■■■■■■ ■■■■■■■■

```
backend/
■■■■ app/
■   ■■■■ api/
■   ■   ■■■■ routes/
■   ■       ■■■■ auth.py          # ■■■■■■■■■■ ■■■■■■■■■■■■■■■■
■   ■■■■ core/
■   ■   ■■■■ security.py          # ■■■■■■■ ■■■■■■■■■■■■■■
■   ■■■■ models/
■   ■   ■■■■ user.py              # ■■■■■■■ ■■■■■■■■■■■■■■
■   ■   ■■■■ session.py           # ■■■■■■■ ■■■■■■■
■   ■■■■ schemas/
■   ■   ■■■■ auth.py              # ■■■■■■ ■■■■■■■
■   ■■■■ services/
■       ■■■■ session_service.py   # ■■■■■■■ ■■■■■■■■■■■■ ■■■■■■■■■
■       ■■■■ mfa_service.py       # ■■■■■■■ MFA
■■■■ AUTHENTICATION_SYSTEM.md     # ■■■■ ■■■■■■■■■■■■■■
```

■■■■■■■■■■■■■■ ■■■■ ■■■■■■■■■■■■

1. **■■■■■■■■■■ MFA ■■■■■■:** ■■■■■■■■■■■■■■■ Redis ■■■■■■■ ■■■■■■■■ ■ ■■■■■■■
2. **■■■■■■■■■ email:** ■■■■■■■■■■■■■■■■ ■■■■■■■■■ email-■■■■■■■ (SendGrid, AWS SES)
3. **Rate limiting:** ■■■■■■■■■ ■■■■■■■■■■■■■ ■■■■■■■■■■■■ ■■■■■■■■ ■■■■■■
4. **■■■■■■■■■■■■■:** ■■■■■■■■■ ■■■■■■■■■■■■■ ■■■■■ ■■■■■■■■ ■■■■■■■■■■■■■■■■
5. **■■■■■■■■■■■■:** ■■■■■■■■■■■■■■■ ■■■■■■■■■■■■■■■■ ■■■■■■■■■■■■
6. **HTTPS:** ■■■■■■■■■■■■■■■ ■■■■■■■ HTTPS ■ ■■■■■■■■■■■■
7. **■■■■■■■■■■■ ■■■■■■:** ■■■■■■■■ ■■■■■■■■■■■ ■■■■■■ ■ ■■■■■■■■■■■■ ■■■■■■■■■■■
8. **■■■■■■■■ ■■■■■■■■:** ■■■■■■■■■■■■■■ ■■■■■■■■■■ ■■■■■■■■■■■■ ■■■■■■■■■

■■■■■■■■■■■■

■■■■■■■■ ■■■■■■■■■■■■■■■■ ■ ■■■■■■■■■■■■ ■■■■■■■■■■■ ■■■■ ■■■■■■■■■■■■ ■■■■■■■■■■■■■■■ ■■■■■■■ ■6:

■ ■■■■■ ■■■■■■■ ■■■■ ■■■■■■■■■■■■■■■■
■ ■■■■■■■ ■■■■■■■■■■■■■■ ■■■■■■■■ ■ ■■■■■■
■ ■■■■■■■ ■■■■■■■■■■■■■■■■■ ■ JWT ■■■■■■■■■■
■ ■■■■■■■ ■■■■■■■■■■■■■ ■■ ■■■■■■■ ■■■■■■
■ ■■■■■■■ ■■■■■■■■■■■■ ■■■■■■■■■■
■ ■■■■■■ ■■■■■■■ ■■■■■■■■■■■■■■■■
■ ■■■■■■■■■■■■■■■■■ ■■■■■■■■■■■■■■■■■
■ ■■■■■■■■■■■■■■ ■■■■■

■■■■■■■■ ■■■■■■■ ■ ■■■■■■■■■■■■■■■ ■ ■■■■■■ ■■■■■ ■■■■■■■■■■ ■■■■ ■■■■■■■■■■■■.

File: backend\requirements.txt

Full path: C:\Users\itodo\Hostelrec\backend\requirements.txt

```
fastapi==0.111.0
uvicorn[standard]==0.30.1
sqlalchemy[asyncio]==2.0.30
asyncpg==0.29.0
psycopg[binary]==3.1.19
pydantic==2.7.3
pydantic-settings==2.3.1
python-jose==3.3.0
passlib[bcrypt]==1.7.4
python-multipart==0.0.9
alembic==1.13.2
loguru==0.7.2
websockets==12.0
Pillow==10.4.0
aiofiles==24.1.0
reportlab==4.0.9
```

File: DATABASE_RELATIONSHIPS.md

Full path: C:\Users\itodo\Hostelrec\DATABASE_RELATIONSHIPS.md

```
# Database Relationships - HostelRec System

This document outlines all the relationships between database tables in the HostelRec system.

## Core Entity: User (hostel_users)

The `User` table is the central entity that connects to most other tables.

### User Relationships

1. **User → CustomerProfile** (One-to-One)
   - `customer_profiles.user_id` → `hostel_users.id` (CASCADE on delete)
   - Each user can have one customer profile
   - Relationship: `User.customer_profile` (backref)

2. **User → CustomerOrder** (One-to-Many)
   - `customer_orders.user_id` → `hostel_users.id` (CASCADE on delete)
   - A user can have multiple orders
   - Relationship: `User.orders` (backref)

3. **User → ChatMessage** (One-to-Many)
   - `chat_messages.user_id` → `hostel_users.id` (CASCADE on delete)
   - A user can send multiple chat messages
   - Relationship: `User.chat_messages` (backref)

4. **User → StaffAttendance** (One-to-Many)
   - `staff_attendance.user_id` → `hostel_users.id` (CASCADE on delete)
   - A staff member can have multiple attendance records
   - Relationship: `User.attendance_records` (backref)

5. **User → GuestCheckin** (One-to-Many, Two Foreign Keys)
   - `guest_checkins.checked_in_by` → `hostel_users.id`
   - `guest_checkins.checked_out_by` → `hostel_users.id`
   - A user (receptionist/manager) can check in/out multiple guests
   - Relationships:
     - `User.checkins_performed` (backref via `checked_in_by`)
     - `User.checkouts_performed` (backref via `checked_out_by`)

## Booking System Relationships

### Guest → Booking (One-to-Many)
- `bookings.guest_id` → `guests.id` (CASCADE on delete)
- A guest can have multiple bookings
- Relationship: `Guest.bookings` (back_populates)
- Relationship: `Booking.guest` (back_populates)

### Booking → Payment (One-to-Many)
- `payments.booking_id` → `bookings.id` (CASCADE on delete)
- A booking can have multiple payments
- Relationship: `Booking.payments` (back_populates)
- Relationship: `Payment.booking` (back_populates)

### CustomerOrder → Payment (One-to-Many)
- `payments.order_id` → `customer_orders.id` (CASCADE on delete)
- An order can have multiple payments
- Relationship: `CustomerOrder.payments` (backref)
- Relationship: `Payment.order` (back_populates)

## Room System

### Room (Standalone)
- `rooms` table has no foreign keys
- Referenced by:
  - `bookings.room_id` (Integer, no FK constraint - references room by ID)
  - `room_images.room_number` (String, no FK constraint - references by room number)
  - `customer_profiles.room_number` (String, no FK constraint - references by room number)
  - `guest_checkins.room_number` (String, no FK constraint - references by room number)
```

RoomImage (Related to Room by room_number)
- `room_images.room_number` → `rooms.room_number` (logical relationship, no FK)
- Multiple images can belong to one room

Content Management (Standalone Tables)

These tables have no foreign key relationships:

1. **FoodItem** (`food_items`)
   - Standalone table for food menu items
   - Referenced by `customer_orders.item_name` (logical relationship)

2. **DrinkItem** (`drink_items`)
   - Standalone table for drink menu items
   - Referenced by `customer_orders.item_name` (logical relationship)

3. **Event** (`events`)
   - Standalone table for hostel events
   - No relationships

4. **Promo** (`promos`)
   - Standalone table for promotions
   - No relationships

Audit System

HostelAuditLog (Standalone)
- `hostel_audit_log` table tracks changes to other tables
- No foreign keys, but references other tables via:
  - `table_name` (string reference)
  - `record_id` (integer reference)
- Created via database triggers on:
  - `rooms`
  - `food_items`
  - `drink_items`
  - `events`
  - `promos`
  - `customer_orders`
  - `guest_checkins`
  - `guests`
  - `bookings`
  - `payments`
  - `hostel_users`

Relationship Summary Diagram

```
User (hostel_users)
■■■ 1:1 → CustomerProfile
■■■ 1:N → CustomerOrder
■         ■■■ 1:N → Payment
■■■ 1:N → ChatMessage
■■■ 1:N → StaffAttendance
■■■ 1:N → GuestCheckin (as checked_in_by)
    ■■■ 1:N → GuestCheckin (as checked_out_by)

Guest (guests)
■■■ 1:N → Booking
    ■■■ 1:N → Payment
    ■■■ N:1 → Room (via room_id, no FK)

Room (rooms)
■■■ 1:N → RoomImage (via room_number, logical)
■■■ 1:N → Booking (via room_id, logical)
■■■ 1:N → CustomerProfile (via room_number, logical)
■■■ 1:N → GuestCheckin (via room_number, logical)

Standalone Tables:
- FoodItem (food_items)
- DrinkItem (drink_items)
- Event (events)
- Promo (promos)
- HostelAuditLog (hostel_audit_log)
```

```

## Key Constraints

1. **Unique Constraints:**
   - `hostel_users.username` (unique)
   - `guests.passport_number` (unique)
   - `rooms.room_number` (unique)
   - `promos.code` (unique)
   - `staff_attendance(user_id, work_date)` (unique composite)

2. **Cascade Deletes:**
   - Deleting a User cascades to:
     - CustomerProfile
     - CustomerOrder
     - ChatMessage
     - StaffAttendance
   - Deleting a Guest cascades to:
     - Booking
   - Deleting a Booking cascades to:
     - Payment
   - Deleting a CustomerOrder cascades to:
     - Payment

3. **Check Constraints:**
   - `bookings.check_out &gt; bookings.check_in` (valid date range)

## Notes

- Some relationships are **logical** (no foreign key constraint):
  - `room_id` in bookings references rooms by ID
  - `room_number` fields reference rooms by room number string
  - `item_name` in orders references food/drink items by name

- The system uses **soft references** for some relationships to allow flexibility:
  - Room references by both ID and room number
  - Order items referenced by name rather than ID

- **Audit logging** is handled via database triggers, not application-level relationships.
```

File: docker-compose.yml

Full path: C:\Users\itodo\Hostelrec\docker-compose.yml

```
services:
  db:
    image: postgres:15
    restart: unless-stopped
    environment:
      POSTGRES_DB: hostelrec
      POSTGRES_USER: postgres
      POSTGRES_PASSWORD: postgres
    ports:
      - "5432:5432"
    volumes:
      - pgdata:/var/lib/postgresql/data
      - ./sql:/docker-entrypoint-initdb.d

  backend:
    build:
      context: ./backend
    environment:
      DATABASE_URL: postgresql+asyncpg://postgres:postgres@db:5432/hostelrec
      AUTH_SECRET_KEY: super-secret-key-change
      AUTH_TOKEN_EXPIRE_MINUTES: 60
      SUPER_SECRET_PHRASE: lab3-secret
    depends_on:
      - db
    ports:
      - "8000:8000"

  frontend:
    build:
      context: ./frontend
    environment:
      VITE_API_URL: http://localhost:8000
    depends_on:
      - backend
    ports:
      - "5173:4173"

volumes:
  pgdata:
```

File: frontend\package.json

Full path: C:\Users\itodo\Hostelrec\frontend\package.json

```
{
  "name": "hostelrec-frontend",
  "version": "0.0.1",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "axios": "^1.7.2",
    "clsx": "^2.1.1",
    "framer-motion": "^11.2.9",
    "i18next": "^23.11.5",
    "react": "^18.2.0",
    "react-dom": "^18.2.0",
    "react-i18next": "^14.1.3",
    "react-router-dom": "^6.23.1",
    "recharts": "^2.9.0",
    "zustand": "^4.5.5"
  },
  "devDependencies": {
    "@types/react": "^18.3.3",
    "@types/react-dom": "^18.3.0",
    "@vitejs/plugin-react": "^4.3.0",
    "autoprefixer": "^10.4.19",
    "postcss": "^8.4.39",
    "tailwindcss": "^3.4.4",
    "typescript": "^5.4.5",
    "vite": "^5.2.12"
  }
}
```

File: frontend\postcss.config.js

Full path: C:\Users\itodo\Hostelrec\frontend\postcss.config.js

```
export default {
  plugins: {
    tailwindcss: {},
    autoprefixer: {}
  }
};
```

File: frontend\src\App.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\App.tsx

```
import { Navigate, Route, Routes } from "react-router-dom";
import { useTranslation } from "react-i18next";

import AdminDashboard from "./pages/AdminDashboard";
import AuditLogPage from "./pages/AuditLogPage";
import CleanerPage from "./pages/CleanerPage";
import LoginPage from "./pages/LoginPage";
import SignupPage from "./pages/SignupPage";
import CustomerPage from "./pages/CustomerPage";
import OrdersPage from "./pages/OrdersPage";
import PublicHome from "./pages/PublicHome";
import SecurityPage from "./pages/SecurityPage";
import StaffManagementPage from "./pages/StaffManagementPage";
import ContentManagementPage from "./pages/ContentManagementPage";
import MenuPage from "./pages/MenuPage";
import CheckInOutPage from "./pages/CheckInOutPage";
import RoomDetailPage from "./pages/RoomDetailPage";
import PasswordChangePage from "./pages/PasswordChangePage";
import MFAPage from "./pages/MFAPage";
import SessionsPage from "./pages/SessionsPage";
import ForgotPasswordPage from "./pages/ForgotPasswordPage";
import AdminLayout from "./layouts/AdminLayout";
import ChatWidget from "./components/ChatWidget";
import { useAuthStore } from "./hooks/useAuthStore";

const App = () =&gt; {
  const { t } = useTranslation();
  const role = useAuthStore((state) =&gt; state.role);

  return (
    &lt;&gt;
      &lt;Routes&gt;
      &lt;Route path="/" element={&lt;PublicHome /&gt;} /&gt;
      &lt;Route path="/login" element={&lt;LoginPage /&gt;} /&gt;
      &lt;Route path="/signup" element={&lt;SignupPage /&gt;} /&gt;
      &lt;Route path="/forgot-password" element={&lt;ForgotPasswordPage /&gt;} /&gt;
      &lt;Route path="/customer" element={&lt;CustomerPage /&gt;} /&gt;
      &lt;Route path="/menu" element={&lt;MenuPage /&gt;} /&gt;
      &lt;Route path="/room/:roomId" element={&lt;RoomDetailPage /&gt;} /&gt;
      &lt;Route
        path="/password-change"
        element={
          role ? (
            &lt;PasswordChangePage /&gt;
          ) : (
            &lt;Navigate to="/login" replace state={{ message: t("login_required") }} /&gt;
          )
        }
      /&gt;
      &lt;Route
        path="/mfa"
        element={
          role ? (
            &lt;MFAPage /&gt;
          ) : (
            &lt;Navigate to="/login" replace state={{ message: t("login_required") }} /&gt;
          )
        }
      /&gt;
      &lt;Route
        path="/sessions"
        element={
          role ? (
            &lt;SessionsPage /&gt;
          ) : (
            &lt;Navigate to="/login" replace state={{ message: t("login_required") }} /&gt;
          )
        }
```

```
      /&gt;
      &lt;Route
        path="/admin/*"
        element={
          role &amp;&amp; (role === "manager" || role === "receptionist") ? (
            &lt;AdminLayout /&gt;
          ) : (
            &lt;Navigate to="/login" replace state={{ message: t("login_required") }} /&gt;
          )
        }
      &gt;
        &lt;Route index element={&lt;AdminDashboard /&gt;} /&gt;
        &lt;Route path="staff" element={&lt;StaffManagementPage /&gt;} /&gt;
        &lt;Route path="logs" element={&lt;AuditLogPage /&gt;} /&gt;
        &lt;Route path="orders" element={&lt;OrdersPage /&gt;} /&gt;
        &lt;Route path="guests" element={&lt;CheckInOutPage /&gt;} /&gt;
        &lt;Route path="security" element={&lt;SecurityPage /&gt;} /&gt;
        &lt;Route path="content" element={&lt;ContentManagementPage /&gt;} /&gt;
      &lt;/Route&gt;
      &lt;Route
        path="/cleaner"
        element={
          role === "cleaner" ? (
            &lt;CleanerPage /&gt;
          ) : (
            &lt;Navigate to="/login" replace state={{ message: t("login_required") }} /&gt;
          )
        }
      /&gt;
    &lt;/Routes&gt;
    &lt;ChatWidget /&gt;
    &lt;/&gt;
  );
};

export default App;
```

File: frontend\src\components\BookingWizard.tsx

Full path:
C:\Users\itodo\Hostelrec\frontend\src\components\BookingWizard.tsx

```
import { useState } from "react";
import { useTranslation } from "react-i18next";

import { api } from "../services/api";
import { useAuthStore, type AuthState } from "../hooks/useAuthStore";

type Step = 1 | 2 | 3;
type InputEvent = { target: { value: string } };

const BookingWizard = () =&gt; {
  const { t } = useTranslation();
  const [step, setStep] = useState&lt;Step&gt;(1);
  const [form, setForm] = useState({
    check_in: "",
    check_out: "",
    room_id: "",
    guest_name: "",
    guest_passport: "",
    phone_number: "",
    payment_method: "cash"
  });
  const [status, setStatus] = useState&lt;string | null&gt;(null);
  const token = useAuthStore((state: AuthState) =&gt; state.token);

  const next = () =&gt; setStep((prev: Step) =&gt; (prev === 3 ? 3 : ((prev + 1) as Step)));
  const prev = () =&gt; setStep((prev: Step) =&gt; (prev === 1 ? 1 : ((prev - 1) as Step)));

  const submit = async () =&gt; {
    try {
      if (!token) {
        setStatus("unauthorized");
        return;
      }
      await api.post("/bookings", {
        ...form,
        room_id: Number(form.room_id),
        phone_number: form.phone_number,
        payment_method: form.payment_method
      });
      setStatus("success");
    } catch (error) {
      console.error(error);
      setStatus("error");
    }
  };

  return (
    &lt;div className="rounded-3xl bg-white p-6 shadow-xl"&gt;
      &lt;div className="flex items-center justify-between"&gt;
        {[1, 2, 3].map((num) =&gt; (
          &lt;div key={num} className="flex flex-col items-center text-sm font-semibold"&gt;
            &lt;div
              className={`flex h-10 w-10 items-center justify-center rounded-full ${
                num === step ? "bg-accent text-primary" : "bg-slate-100"
              }`}
            &gt;
              {num}
            &lt;/div&gt;
            &lt;span className="mt-2 text-slate-500"&gt;
              {num === 1 ? t("dates") : num === 2 ? t("guest_details") : t("confirm_pay")}
            &lt;/span&gt;
          &lt;/div&gt;
        ))}
      &lt;/div&gt;

      &lt;div className="mt-8 space-y-5"&gt;
        {step === 1 &amp;amp;&amp;amp; (
```

```
    &lt;div className="grid gap-4 md:grid-cols-3"&gt;
      &lt;input
        type="date"
        value={form.check_in}
        onChange={(e: InputEvent) =&gt; setForm({ ...form, check_in: e.target.value })}
        className="rounded-xl border border-slate-200 px-3 py-3"
      /&gt;
      &lt;input
        type="date"
        value={form.check_out}
        onChange={(e: InputEvent) =&gt; setForm({ ...form, check_out: e.target.value })}
        className="rounded-xl border border-slate-200 px-3 py-3"
      /&gt;
      &lt;input
        type="number"
        placeholder="Room ID"
        value={form.room_id}
        onChange={(e: InputEvent) =&gt; setForm({ ...form, room_id: e.target.value })}
        className="rounded-xl border border-slate-200 px-3 py-3"
      /&gt;
    &lt;/div&gt;
  )}

  {step === 2 &amp;&amp; (
    &lt;div className="grid gap-4 md:grid-cols-2"&gt;
      &lt;input
        type="text"
        placeholder="Guest name"
        value={form.guest_name}
        onChange={(e: InputEvent) =&gt; setForm({ ...form, guest_name: e.target.value })}
        className="rounded-xl border border-slate-200 px-3 py-3"
      /&gt;
      &lt;input
        type="text"
        placeholder="Passport"
        value={form.guest_passport}
        onChange={(e: InputEvent) =&gt; setForm({ ...form, guest_passport: e.target.value })}
        className="rounded-xl border border-slate-200 px-3 py-3"
      /&gt;
      &lt;input
        type="tel"
        placeholder="Phone number"
        value={form.phone_number}
        onChange={(e: InputEvent) =&gt; setForm({ ...form, phone_number: e.target.value })}
        className="rounded-xl border border-slate-200 px-3 py-3"
      /&gt;
      &lt;select
        value={form.payment_method}
        onChange={(e: InputEvent) =&gt; setForm({ ...form, payment_method: e.target.value })}
        className="rounded-xl border border-slate-200 px-3 py-3"
      &gt;
        &lt;option value="cash"&gt;■ Cash&lt;/option&gt;
        &lt;option value="card"&gt;■ Card&lt;/option&gt;
        &lt;option value="bank_transfer"&gt;■ Bank Transfer&lt;/option&gt;
        &lt;option value="online"&gt;■ Online&lt;/option&gt;
      &lt;/select&gt;
    &lt;/div&gt;
  )}

  {step === 3 &amp;&amp; (
    &lt;div className="rounded-2xl border border-slate-200 p-4"&gt;
      &lt;h4 className="text-lg font-semibold text-primary"&gt;Summary&lt;/h4&gt;
      &lt;p className="text-sm text-slate-500"&gt;Room #{form.room_id}&lt;/p&gt;
      &lt;p className="text-sm text-slate-500"&gt;
        {form.check_in} → {form.check_out}
      &lt;/p&gt;
    &lt;/div&gt;
  )}
&lt;/div&gt;

&lt;div className="mt-6 flex items-center justify-between"&gt;
  &lt;button
    onClick={prev}
```

```
          disabled={step === 1}
          className="rounded-xl border border-slate-300 px-4 py-2 disabled:opacity-40"
        &gt;
          Back
        &lt;/button&gt;
        {step &lt; 3 ? (
          &lt;button onClick={next} className="rounded-xl bg-accent px-4 py-2 font-semibold text-primary"&gt
            Next
          &lt;/button&gt;
        ) : (
          &lt;button
            onClick={submit}
            className={`rounded-xl px-4 py-2 font-semibold text-white ${
              token ? "bg-success" : "bg-slate-400 cursor-not-allowed"
            }`}
            disabled={!token}
          &gt;
            Pay
          &lt;/button&gt;
        )}
      &lt;/div&gt;

      {status === "success" &amp;&amp; (
        &lt;p className="mt-4 rounded-xl bg-success/10 px-4 py-3 text-success"&gt;{t("booking_confirmed")}&l
      )}
      {status === "error" &amp;&amp; (
        &lt;p className="mt-4 rounded-xl bg-danger/10 px-4 py-3 text-danger"&gt;{t("booking_failed")}&lt;/p&
      )}
      {status === "unauthorized" &amp;&amp; (
        &lt;p className="mt-4 rounded-xl bg-accent/10 px-4 py-3 text-primary"&gt;{t("login_required")}&lt;/p
      )}
    &lt;/div&gt;
  );
};

export default BookingWizard;
```

File: frontend\src\components\ChatWidget.tsx

Full path:
C:\Users\itodo\Hostelrec\frontend\src\components\ChatWidget.tsx

```
// @ts-nocheck
import { useState, useEffect, useRef } from "react";
import { motion, AnimatePresence } from "framer-motion";
import { useTranslation } from "react-i18next";
import { useAuthStore } from "../hooks/useAuthStore";
import { api } from "../services/api";

type ChatMessage = {
  id?: number;
  user_id?: number;
  username: string;
  message: string;
  timestamp?: string;
  created_at?: string;
};

type Customer = {
  id: number;
  username: string;
  full_name?: string;
  room_number?: string;
  profile_picture_url?: string;
};

const ChatWidget = () =&gt; {
  const { t } = useTranslation();
  const [isOpen, setIsOpen] = useState(false);
  const [messages, setMessages] = useState&lt;ChatMessage[]&gt;([]);
  const [inputMessage, setInputMessage] = useState("");
  const [ws, setWs] = useState&lt;WebSocket | null&gt;(null);
  const [connected, setConnected] = useState(false);
  const messagesEndRef = useRef&lt;HTMLDivElement&gt;(null);
  const token = useAuthStore((state) =&gt; state.token);
  const role = useAuthStore((state) =&gt; state.role);
  const [currentUserId, setCurrentUserId] = useState&lt;number | null&gt;(null);
  const [customers, setCustomers] = useState&lt;Customer[]&gt;([]);
  const [showCustomerSelect, setShowCustomerSelect] = useState(false);
  const [selectedCustomer, setSelectedCustomer] = useState&lt;Customer | null&gt;(null);

  // Extract username from token (JWT payload) or use default
  const getUsername = () =&gt; {
    if (!token) return "Guest";
    try {
      const payload = JSON.parse(atob(token.split('.')[1]));
      return payload.sub || "Guest";
    } catch {
      return "Guest";
    }
  };
  const username = getUsername();

  // Get current user ID
  useEffect(() =&gt; {
    if (token) {
      try {
        const payload = JSON.parse(atob(token.split('.')[1]));
        // You might need to fetch user ID from API if not in token
        setCurrentUserId(payload.user_id || null);
      } catch {
        setCurrentUserId(null);
      }
    }
  }, [token]);

  const scrollToBottom = () =&gt; {
    messagesEndRef.current?.scrollIntoView({ behavior: "smooth" });
  };
```

```
useEffect(() =&gt; {
  scrollToBottom();
}, [messages]);

// Load customers list for staff
useEffect(() =&gt; {
  if ((role === "receptionist" || role === "manager") &amp;&amp; isOpen) {
    api.get&lt;Customer[]&gt;("/customer/list")
      .then((res) =&gt; setCustomers(res.data))
      .catch((err) =&gt; console.error("Failed to load customers", err));
  }
}, [isOpen, role]);

useEffect(() =&gt; {
  if (!isOpen) return;

  // Load recent messages
  const apiUrl = import.meta.env.VITE_API_URL ?? "http://localhost:8000";
  fetch(`${apiUrl}/chat/messages`, {
    headers: token ? { Authorization: `Bearer ${token}` } : {}
  })
    .then((res) =&gt; res.json())
    .then((data) =&gt; {
      setMessages(data || []);
    })
    .catch((err) =&gt; console.error("Failed to load messages", err));

  // Connect WebSocket
  const wsUrl = apiUrl.replace("http", "ws");
  const websocket = new WebSocket(`${wsUrl}/chat/ws`);

  websocket.onopen = () =&gt; {
    setConnected(true);
    setWs(websocket);
  };

  websocket.onmessage = (event) =&gt; {
    const data = JSON.parse(event.data);
    if (data.type === "message") {
      setMessages((prev) =&gt; [...prev, data]);
    } else if (data.type === "message_deleted") {
      setMessages((prev) =&gt; prev.filter((msg) =&gt; msg.id !== data.message_id));
    }
  };

  websocket.onerror = (error) =&gt; {
    console.error("WebSocket error:", error);
    setConnected(false);
  };

  websocket.onclose = () =&gt; {
    setConnected(false);
  };

  return () =&gt; {
    websocket.close();
  };
}, [isOpen]);

const sendMessage = (e: any) =&gt; {
  e.preventDefault();
  if (!inputMessage.trim() || !ws || !connected) return;

  const messageData = {
    user_id: currentUserId || 0,
    username: username,
    message: inputMessage,
    timestamp: new Date().toISOString(),
  };

  ws.send(JSON.stringify(messageData));
  setInputMessage("");
};
```

```
const deleteMessage = async (messageId: number) =&gt; {
  if (!confirm("Are you sure you want to delete this message?")) return;
  try {
    await api.delete(`/chat/messages/${messageId}`);
    setMessages((prev) =&gt; prev.filter((msg) =&gt; msg.id !== messageId));
  } catch (err) {
    console.error("Failed to delete message", err);
    alert("Failed to delete message");
  }
};

const canDeleteMessage = (msg: ChatMessage) =&gt; {
  // Staff can delete any message, users can delete their own
  return role === "manager" || role === "receptionist" || (msg.user_id &amp;&amp; msg.user_id === currentU
};

return (
  &lt;&gt;
    {/* Chat Button */}
    &lt;motion.button
      whileHover={{ scale: 1.05 }}
      whileTap={{ scale: 0.95 }}
      onClick={() =&gt; setIsOpen(!isOpen)}
      className="fixed bottom-6 right-6 z-50 flex h-14 w-14 items-center justify-center rounded-full bg-gr
  from-primary to-accent text-white shadow-2xl transition-all hover:shadow-3xl"
    &gt;
      &lt;AnimatePresence mode="wait"&gt;
        {isOpen ? (
          &lt;motion.svg
            key="close"
            initial={{ rotate: -90, opacity: 0 }}
            animate={{ rotate: 0, opacity: 1 }}
            exit={{ rotate: 90, opacity: 0 }}
            className="h-6 w-6"
            fill="none"
            stroke="currentColor"
            viewBox="0 0 24 24"
          &gt;
            &lt;path
              strokeLinecap="round"
              strokeLinejoin="round"
              strokeWidth={2}
              d="M6 18L18 6M6 6l12 12"
            /&gt;
          &lt;/motion.svg&gt;
        ) : (
          &lt;motion.svg
            key="chat"
            initial={{ rotate: 90, opacity: 0 }}
            animate={{ rotate: 0, opacity: 1 }}
            exit={{ rotate: -90, opacity: 0 }}
            className="h-6 w-6"
            fill="none"
            stroke="currentColor"
            viewBox="0 0 24 24"
          &gt;
            &lt;path
              strokeLinecap="round"
              strokeLinejoin="round"
              strokeWidth={2}
              d="M8 12h.01M12 12h.01M16 12h.01M21 12c0 4.418-4.03 8-9 8a9.863 9.863 0 01-4.255-.949L3 20l1
  512 15.042 3 13.574 3 12c0-4.418 4.03-8 9-8s9 3.582 9 8z"
            /&gt;
          &lt;/motion.svg&gt;
        )}
      &lt;/AnimatePresence&gt;
      {!connected &amp;&amp; isOpen &amp;&amp; (
        &lt;motion.span
          className="absolute -top-1 -right-1 h-3 w-3 rounded-full bg-red-500"
          animate={{ scale: [1, 1.2, 1] }}
          transition={{ repeat: Infinity, duration: 1 }}
        /&gt;
      )}
```

```
  &lt;/motion.button&gt;

  {/* Chat Window */}
  &lt;AnimatePresence&gt;
    {isOpen &amp;&amp; (
      &lt;motion.div
        initial={{ opacity: 0, y: 20, scale: 0.9 }}
        animate={{ opacity: 1, y: 0, scale: 1 }}
        exit={{ opacity: 0, y: 20, scale: 0.9 }}
        className="fixed bottom-24 right-6 z-40 flex h-[500px] w-96 flex-col rounded-2xl bg-white shadow
      &gt;
        {/* Header */}
        &lt;div className="flex items-center justify-between rounded-t-2xl bg-gradient-to-r from-primary
-4 text-white relative"&gt;
          &lt;div className="flex items-center gap-2"&gt;
            &lt;div className="h-2 w-2 rounded-full bg-white"&gt;&lt;/div&gt;
            &lt;h3 className="font-semibold text-sm"&gt;
              {selectedCustomer ? `Chat with ${selectedCustomer.full_name || selectedCustomer.username}`
title") || "Live Chat")}
            &lt;/h3&gt;
          &lt;/div&gt;
          &lt;div className="flex items-center gap-2"&gt;
            {(role === "receptionist" || role === "manager") &amp;&amp; (
              &lt;button
                onClick={() =&gt; setShowCustomerSelect(!showCustomerSelect)}
                className="p-1 hover:bg-white/20 rounded"
                title="Select customer"
              &gt;
                ■
              &lt;/button&gt;
            )}
            &lt;span
              className={`h-2 w-2 rounded-full ${
                connected ? "bg-green-300" : "bg-red-300"
              }`}
            /&gt;
          &lt;/div&gt;

          {/* Customer Selection Dropdown (Staff Only) */}
          {showCustomerSelect &amp;&amp; (role === "receptionist" || role === "manager") &amp;&amp; (
            &lt;div className="absolute top-full left-0 right-0 bg-white border border-slate-200 rounded
g z-50 max-h-64 overflow-y-auto mt-2"&gt;
              {customers.length === 0 ? (
                &lt;div className="p-4 text-center text-slate-500"&gt;No customers found&lt;/div&gt;
              ) : (
                customers.map((customer) =&gt; (
                  &lt;button
                    key={customer.id}
                    onClick={() =&gt; {
                      setSelectedCustomer(customer);
                      setShowCustomerSelect(false);
                    }}
                    className="w-full flex items-center gap-3 p-3 hover:bg-slate-50 text-left"
                  &gt;
                    &lt;div className="w-10 h-10 rounded-full bg-gradient-to-r from-primary to-accent fl
ter justify-center text-white font-semibold"&gt;
                      {customer.profile_picture_url ? (
                        &lt;img src={customer.profile_picture_url} alt={customer.full_name || customer.u
ssName="w-full h-full rounded-full object-cover" /&gt;
                      ) : (
                        (customer.full_name || customer.username).charAt(0).toUpperCase()
                      )}
                    &lt;/div&gt;
                    &lt;div className="flex-1"&gt;
                      &lt;p className="font-semibold text-slate-800"&gt;{customer.full_name || customer.
;/p&gt;
                      {customer.room_number &amp;&amp; (
                        &lt;p className="text-xs text-slate-500"&gt;Room {customer.room_number}&lt;/p&gt
                      )}
                    &lt;/div&gt;
                  &lt;/button&gt;
                ))
              )}
```

```
            &lt;/div&gt;
          )}
        &lt;/div&gt;

        {/* Messages */}
        &lt;div className="flex-1 overflow-y-auto p-4 space-y-3"&gt;
          &lt;AnimatePresence&gt;
            {messages.map((msg, idx) =&gt; {
              const isOwn = msg.username === username;
              const canDelete = canDeleteMessage(msg);
              return (
                &lt;motion.div
                  key={msg.id || idx}
                  initial={{ opacity: 0, x: isOwn ? 20 : -20 }}
                  animate={{ opacity: 1, x: 0 }}
                  exit={{ opacity: 0, x: isOwn ? 20 : -20 }}
                  className={`flex ${isOwn ? "justify-end" : "justify-start"} group`}
                &gt;
                  &lt;div className="relative"&gt;
                    &lt;div
                      className={`max-w-[75%] rounded-2xl px-4 py-2 ${
                        isOwn
                          ? "bg-gradient-to-r from-primary to-accent text-white"
                          : "bg-slate-100 text-slate-800"
                      }`}
                    &gt;
                      {!isOwn &amp;&amp; (
                        &lt;p className="text-xs font-semibold mb-1 opacity-80"&gt;
                          {msg.username}
                        &lt;/p&gt;
                      )}
                      &lt;p className="text-sm"&gt;{msg.message}&lt;/p&gt;
                      &lt;p className="text-xs mt-1 opacity-60"&gt;
                        {new Date(msg.timestamp || msg.created_at).toLocaleTimeString()}
                      &lt;/p&gt;
                    &lt;/div&gt;
                    {canDelete &amp;&amp; msg.id &amp;&amp; (
                      &lt;motion.button
                        initial={{ opacity: 0 }}
                        whileHover={{ opacity: 1 }}
                        className="absolute -top-2 -right-2 h-5 w-5 rounded-full bg-red-500 text-white t
items-center justify-center opacity-0 group-hover:opacity-100 transition-opacity"
                        onClick={() =&gt; deleteMessage(msg.id!)}
                        title="Delete message"
                      &gt;
                        ×
                      &lt;/motion.button&gt;
                    )}
                  &lt;/div&gt;
                &lt;/motion.div&gt;
              );
            })}
          &lt;/AnimatePresence&gt;
          &lt;div ref={messagesEndRef} /&gt;
        &lt;/div&gt;

        {/* Input */}
        &lt;form onSubmit={sendMessage} className="border-t border-slate-200 p-4"&gt;
          &lt;div className="flex gap-2"&gt;
            &lt;input
              type="text"
              value={inputMessage}
              onChange={(e) =&gt; setInputMessage(e.target.value)}
              placeholder={t("chat_placeholder") || "Type a message..."}
              className="flex-1 rounded-xl border border-slate-200 px-4 py-2 focus:outline-none focus:ri
ing-primary"
              disabled={!connected}
            /&gt;
            &lt;motion.button
              whileHover={{ scale: 1.05 }}
              whileTap={{ scale: 0.95 }}
              type="submit"
              disabled={!connected || !inputMessage.trim()}
```

```
                  className="rounded-xl bg-primary px-4 py-2 text-white disabled:opacity-50"
                &gt;
                  &lt;svg
                    className="h-5 w-5"
                    fill="none"
                    stroke="currentColor"
                    viewBox="0 0 24 24"
                  &gt;
                    &lt;path
                      strokeLinecap="round"
                      strokeLinejoin="round"
                      strokeWidth={2}
                      d="M12 19l9 2-9-18-9 18 9-2zm0 0v-8"
                    /&gt;
                  &lt;/svg&gt;
                &lt;/motion.button&gt;
              &lt;/div&gt;
            &lt;/form&gt;
          &lt;/motion.div&gt;
        )}
      &lt;/AnimatePresence&gt;
    &lt;/&gt;
  );
};

export default ChatWidget;
```

File: frontend\src\components\FloorPlan.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\components\FloorPlan.tsx

```
type Room = {
  id: number;
  room_number: string;
  status: string;
};

type FloorPlanProps = {
  rooms: Room[];
};

const FloorPlan = ({ rooms }: FloorPlanProps) =&gt; {
  // Use real room data from database
  const displayRooms = rooms.slice(0, 8); // Show up to 8 rooms

  if (displayRooms.length === 0) {
    return (
      &lt;div className="w-full rounded-2xl bg-white p-8 shadow-md text-center text-slate-600"&gt;
        &lt;p&gt;No rooms to display&lt;/p&gt;
      &lt;/div&gt;
    );
  }

  const cols = Math.ceil(Math.sqrt(displayRooms.length));
  const rows = Math.ceil(displayRooms.length / cols);
  const viewBoxWidth = cols * 100 + 40;
  const viewBoxHeight = rows * 120 + 80;

  return (
    &lt;svg viewBox={`0 0 ${viewBoxWidth} ${viewBoxHeight}`} className="w-full rounded-2xl bg-white p-4 shad
      {displayRooms.map((room, idx) =&gt; {
        const col = idx % cols;
        const row = Math.floor(idx / cols);
        const x = 20 + col * 100;
        const y = 40 + row * 120;
        const color = room.status === "occupied" ? "#EF4444" : "#10B981";
        return (
          &lt;g key={room.id}&gt;
            &lt;rect x={x} y={y} width="80" height="100" fill={color} rx="12" opacity={0.7} /&gt;
            &lt;text x={x + 40} y={y + 50} textAnchor="middle" fontSize="16" fill="#0F172A" fontWeight="bold
              {room.room_number}
            &lt;/text&gt;
            &lt;text x={x + 40} y={y + 70} textAnchor="middle" fontSize="12" fill="#0F172A" opacity={0.7}&gt
              {room.status}
            &lt;/text&gt;
          &lt;/g&gt;
        );
      })}
    &lt;/svg&gt;
  );
};

export default FloorPlan;
```

File: frontend\src\components\GanttTimeline.tsx

Full path:
C:\Users\itodo\Hostelrec\frontend\src\components\GanttTimeline.tsx

```
import { useTranslation } from "react-i18next";

type BookingBar = {
  room: string;
  start: number;
  end: number;
  color: string;
};

const GanttTimeline = ({ data }: { data: BookingBar[] }) =&gt; {
  const { t } = useTranslation();

  return (
    &lt;div className="rounded-2xl bg-white p-6 shadow-md"&gt;
      &lt;h3 className="text-lg font-semibold text-primary"&gt;{t("gantt_schedule")}&lt;/h3&gt;
      &lt;div className="mt-4 space-y-4"&gt;
        {data.map((bar) =&gt; (
          &lt;div key={bar.room}&gt;
            &lt;p className="text-sm font-semibold text-slate-500"&gt;{bar.room}&lt;/p&gt;
            &lt;div className="relative h-8 rounded-full bg-slate-100"&gt;
              &lt;span
                className="absolute h-8 rounded-full text-xs font-bold text-white"
                style={{
                  left: `${bar.start}%`,
                  width: `${bar.end - bar.start}%`,
                  backgroundColor: bar.color,
                  display: "flex",
                  alignItems: "center",
                  justifyContent: "center"
                }}
              &gt;
                {t("stay")}
              &lt;/span&gt;
            &lt;/div&gt;
          &lt;/div&gt;
        ))}
      &lt;/div&gt;
    &lt;/div&gt;
  );
};

export default GanttTimeline;
```

File: frontend\src\components\HeroSection.tsx

Full path:
C:\Users\itodo\Hostelrec\frontend\src\components\HeroSection.tsx

```
// @ts-nocheck
import { motion } from "framer-motion";
import { useTranslation } from "react-i18next";

const HeroSection = ({ occupancyHigh }: { occupancyHigh: boolean }) =&gt; {
  const { t } = useTranslation();

  return (
    &lt;section className="relative overflow-hidden rounded-3xl bg-primary text-white"&gt;
      &lt;div className="absolute inset-0"&gt;
        &lt;video
          className="h-full w-full object-cover opacity-40"
          autoPlay
          loop
          muted
          playsInline
          src="https://storage.googleapis.com/coverr-main/mp4/Mt_Baker.mp4"
        /&gt;
      &lt;/div&gt;
      &lt;div className="relative z-10 p-12"&gt;
        &lt;motion.p
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ duration: 0.6 }}
          className="text-accent font-semibold uppercase tracking-wide"
        &gt;
          {t("welcome")}
        &lt;/motion.p&gt;
        &lt;motion.h1
          initial={{ opacity: 0 }}
          animate={{ opacity: 1 }}
          transition={{ duration: 1 }}
          className="mt-4 text-4xl font-extrabold md:text-6xl"
        &gt;
          {t("best_hostel")}
        &lt;/motion.h1&gt;
        &lt;motion.p
          className="mt-6 max-w-2xl text-lg text-white/80"
          initial={{ opacity: 0 }}
          animate={{ opacity: 1 }}
          transition={{ delay: 0.3, duration: 0.8 }}
        &gt;
          {t("hero_subtitle")}
        &lt;/motion.p&gt;

        &lt;motion.div
          initial={{ scale: 0.9, opacity: 0 }}
          animate={{ scale: 1, opacity: 1 }}
          transition={{ delay: 0.5 }}
          className="mt-8 flex items-center gap-4"
        &gt;
          &lt;button className="rounded-full bg-accent px-6 py-3 text-lg font-semibold text-primary shadow-l
    ow-xl transition"&gt;
            {t("book_btn")}
          &lt;/button&gt;
          {occupancyHigh ? (
            &lt;span className="flex items-center gap-2 rounded-full bg-danger/90 px-4 py-2 text-sm font-sem
    hite animate-pulse"&gt;
              {t("availability_high")}
            &lt;/span&gt;
          ) : (
            &lt;span className="rounded-full bg-white/10 px-4 py-2 text-sm font-semibold"&gt;
              {t("availability_normal")}
            &lt;/span&gt;
          )}
        &lt;/motion.div&gt;
      &lt;/div&gt;
```

```
    &lt;/section&gt;
  );
};

export default HeroSection;
```

File: frontend\src\components\KpiCard.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\components\KpiCard.tsx

```
type Props = {
  label: string;
  value: string;
  accent?: string;
};

const KpiCard = ({ label, value, accent = "text-primary" }: Props) =&gt; (
  &lt;div className="rounded-2xl bg-white p-6 shadow-md"&gt;
    &lt;p className="text-sm font-semibold text-slate-500"&gt;{label}&lt;/p&gt;
    &lt;p className={`mt-3 text-3xl font-bold ${accent}`}&gt;{value}&lt;/p&gt;
  &lt;/div&gt;
);

export default KpiCard;
```

File: frontend\src\components\LanguageToggle.tsx

Full path:
C:\Users\itodo\Hostelrec\frontend\src\components\LanguageToggle.tsx

```
import { useTranslation } from "react-i18next";

const LanguageToggle = () =&gt; {
  const { i18n, t } = useTranslation();
  const toggle = () =&gt; {
    i18n.changeLanguage(i18n.language === "en" ? "ru" : "en");
  };

  return (
    &lt;button
      aria-label={t("toggle_language")}
      onClick={toggle}
      className="rounded-full border border-primary/20 px-4 py-1 text-sm font-semibold"
    &gt;
      {i18n.language === "en" ? t("lang_ru") : t("lang_en")}
    &lt;/button&gt;
  );
};

export default LanguageToggle;
```

File: frontend\src\components\Navbar.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\components\Navbar.tsx

```
// @ts-nocheck
import { Link } from "react-router-dom";
import { motion } from "framer-motion";
import { useTranslation } from "react-i18next";

import { useAuthStore } from "../hooks/useAuthStore";
import LanguageToggle from "./LanguageToggle";

const Navbar = () =&gt; {
  const { t } = useTranslation();
  const role = useAuthStore((state) =&gt; state.role);
  const logout = useAuthStore((state) =&gt; state.logout);

  return (
    &lt;motion.header
      initial={{ y: -100, opacity: 0 }}
      animate={{ y: 0, opacity: 1 }}
      transition={{ duration: 0.5 }}
      className="sticky top-0 z-30 flex items-center justify-between bg-white/90 backdrop-blur-md px-6 py-4
    rder-b border-slate-200/50"
    &gt;
      &lt;motion.div whileHover={{ scale: 1.05 }} whileTap={{ scale: 0.95 }}&gt;
        &lt;Link to="/" className="text-2xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-text t
    ent"&gt;
          HostelRec
        &lt;/Link&gt;
      &lt;/motion.div&gt;

      &lt;nav className="flex items-center gap-4 text-sm font-semibold"&gt;
        &lt;motion.div whileHover={{ scale: 1.05 }} whileTap={{ scale: 0.95 }}&gt;
          &lt;Link to="/" className="hover:text-accent transition-colors relative group"&gt;
            {t("nav_public")}
            &lt;span className="absolute bottom-0 left-0 w-0 h-0.5 bg-accent group-hover:w-full transition-a
    300"&gt;&lt;/span&gt;
          &lt;/Link&gt;
        &lt;/motion.div&gt;
        &lt;motion.div whileHover={{ scale: 1.05 }} whileTap={{ scale: 0.95 }}&gt;
          &lt;Link to="/customer" className="hover:text-accent transition-colors relative group"&gt;
            {t("nav_customer")}
            &lt;span className="absolute bottom-0 left-0 w-0 h-0.5 bg-accent group-hover:w-full transition-a
    300"&gt;&lt;/span&gt;
          &lt;/Link&gt;
        &lt;/motion.div&gt;
        &lt;motion.div whileHover={{ scale: 1.05 }} whileTap={{ scale: 0.95 }}&gt;
          &lt;Link to="/menu" className="hover:text-accent transition-colors relative group"&gt;
            {t("nav_menu") || "Menu"}
            &lt;span className="absolute bottom-0 left-0 w-0 h-0.5 bg-accent group-hover:w-full transition-a
    300"&gt;&lt;/span&gt;
          &lt;/Link&gt;
        &lt;/motion.div&gt;
        {(role === "manager" || role === "receptionist") &amp;&amp; (
          &lt;motion.div whileHover={{ scale: 1.05 }} whileTap={{ scale: 0.95 }}&gt;
            &lt;Link to="/admin" className="hover:text-accent transition-colors relative group"&gt;
              {t("nav_admin")}
              &lt;span className="absolute bottom-0 left-0 w-0 h-0.5 bg-accent group-hover:w-full transition
    n-300"&gt;&lt;/span&gt;
            &lt;/Link&gt;
          &lt;/motion.div&gt;
        )}
        {role === "cleaner" &amp;&amp; (
          &lt;motion.div whileHover={{ scale: 1.05 }} whileTap={{ scale: 0.95 }}&gt;
            &lt;Link to="/cleaner" className="hover:text-accent transition-colors relative group"&gt;
              {t("cleaner")}
              &lt;span className="absolute bottom-0 left-0 w-0 h-0.5 bg-accent group-hover:w-full transition
    n-300"&gt;&lt;/span&gt;
            &lt;/Link&gt;
          &lt;/motion.div&gt;
        )}
```

```
      &lt;/nav&gt;

      &lt;div className="flex items-center gap-3"&gt;
        &lt;LanguageToggle /&gt;
        {role ? (
          &lt;motion.button
            whileHover={{ scale: 1.05 }}
            whileTap={{ scale: 0.95 }}
            onClick={logout}
            className="rounded-lg bg-gradient-to-r from-primary to-accent px-4 py-2 text-white shadow-lg hov
     transition-all"
          &gt;
            {t("Sign out")}
          &lt;/motion.button&gt;
        ) : (
          &lt;motion.div whileHover={{ scale: 1.05 }} whileTap={{ scale: 0.95 }}&gt;
            &lt;Link to="/login" className="rounded-lg border-2 border-primary px-4 py-2 text-primary hover:
    over:text-white transition-all"&gt;
              {t("Sign in")}
            &lt;/Link&gt;
          &lt;/motion.div&gt;
        )}
      &lt;/div&gt;
    &lt;/motion.header&gt;
  );
};

export default Navbar;
```

File: frontend\src\components\RevenueChart.tsx

Full path:
C:\Users\itodo\Hostelrec\frontend\src\components\RevenueChart.tsx

```
// @ts-nocheck
import { Pie, PieChart, ResponsiveContainer, Tooltip, Cell } from "recharts";
import { useTranslation } from "react-i18next";

const colors = ["#F59E0B", "#6366F1", "#10B981", "#0EA5E9"];

type Props = {
  data: { room_type: string; revenue: number }[];
};

const RevenueChart = ({ data }: Props) =&gt; {
  const { t } = useTranslation();

  return (
    &lt;div className="rounded-2xl bg-white p-6 shadow-md"&gt;
      &lt;h3 className="text-lg font-semibold text-primary"&gt;{t("revenue_by_room_type")}&lt;/h3&gt;
      &lt;div className="h-64"&gt;
        &lt;ResponsiveContainer&gt;
          &lt;PieChart&gt;
            &lt;Pie data={data} dataKey="revenue" nameKey="room_type" outerRadius={90} label&gt;
              {data.map((_, idx) =&gt; (
                &lt;Cell key={idx} fill={colors[idx % colors.length]} /&gt;
              ))}
            &lt;/Pie&gt;
            &lt;Tooltip /&gt;
          &lt;/PieChart&gt;
        &lt;/ResponsiveContainer&gt;
      &lt;/div&gt;
    &lt;/div&gt;
  );
};

export default RevenueChart;
```

File: frontend\src\components\RoomCard.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\components\RoomCard.tsx

```
import { motion } from "framer-motion";
import { useTranslation } from "react-i18next";
import { useNavigate } from "react-router-dom";

type Props = {
  id?: number;
  title: string;
  price: number | string;
  amenities: string[];
  status: "available" | "occupied";
  image_url?: string | null;
  location?: string | null;
  wifi?: boolean;
  key?: string;
};

const RoomCard = ({ id, title, price, amenities, status, image_url, location, wifi }: Props) =&gt; {
  const { t } = useTranslation();
  const navigate = useNavigate();
  const apiUrl = import.meta.env.VITE_API_URL ?? "http://localhost:8000";
  const defaultImage = "https://images.unsplash.com/photo-1512918728675-ed5a9ecdebfd?auto=format&amp;fit=cro
    amp;q=60";
  const imageSrc = image_url ? `${apiUrl}${image_url}` : defaultImage;
  const priceDisplay = typeof price === "number" ? `■${price.toFixed(2)}` : price;

  const handleClick = () =&gt; {
    if (id) {
      navigate(`/room/${id}`);
    }
  };

  return (
    &lt;div onClick={id ? handleClick : undefined} className={id ? "cursor-pointer" : ""}&gt;
    &lt;motion.div
      whileHover={{ scale: 1.05 }}
      className="flex flex-col rounded-2xl bg-white shadow-lg"
    &gt;
        &lt;div className="h-40 rounded-t-2xl bg-cover bg-center" style={{ backgroundImage: `url('${imageSrc
    ;
      &lt;div className="flex flex-1 flex-col gap-3 p-5"&gt;
        &lt;div className="flex items-center justify-between"&gt;
          &lt;h3 className="text-lg font-semibold text-primary"&gt;{title}&lt;/h3&gt;
          &lt;span
            className={`rounded-full px-3 py-1 text-xs font-bold ${
              status === "available" ? "bg-success/10 text-success" : "bg-danger/10 text-danger"
            }`}
          &gt;
            {t(status)}
          &lt;/span&gt;
        &lt;/div&gt;
          &lt;p className="text-2xl font-bold text-accent"&gt;{priceDisplay}&lt;/p&gt;
          {location &amp;&amp; &lt;p className="text-sm text-slate-500"&gt;■ {location}&lt;/p&gt;}
          {wifi !== undefined &amp;&amp; (
            &lt;p className="text-sm text-slate-500"&gt;{wifi ? "■ WiFi Available" : "■ No WiFi"}&lt;/p&gt;
          )}
        &lt;ul className="text-sm text-slate-600"&gt;
          {amenities.map((item) =&gt; (
            &lt;li key={item}&gt;• {item}&lt;/li&gt;
          ))}
        &lt;/ul&gt;
      &lt;/div&gt;
    &lt;/motion.div&gt;
    &lt;/div&gt;
  );
};

export default RoomCard;
```

File: frontend\src\hooks\useAuthStore.ts

Full path: C:\Users\itodo\Hostelrec\frontend\src\hooks\useAuthStore.ts

```
import { create } from "zustand";
import { persist } from "zustand/middleware";

import { setAuthToken } from "../services/api";
import { api } from "../services/api";

export type Role = "receptionist" | "manager" | "cleaner" | "customer" | null;

export interface AuthState {
  token: string | null;
  role: Role;
  setCredentials: (token: string, role: Role) =&gt; void;
  logout: () =&gt; Promise&lt;void&gt;;
}

export const useAuthStore = create&lt;AuthState&gt;()(
  persist(
    (set) =&gt; ({
      token: null,
      role: null,
      setCredentials: (token, role) =&gt; {
        setAuthToken(token);
        set({ token, role });
      },
      logout: async () =&gt; {
        try {
          // ████████ API ███ ████████████ ███████ ██ ████████
          await api.post("/auth/logout");
        } catch (error) {
          // ████████████ ███████ ███ ███████ (██████ ██████ █████ ███ ████████████)
          console.error("Logout error:", error);
        } finally {
          setAuthToken(undefined);
          set({ token: null, role: null });
        }
      }
    }),
    {
      name: "hostelrec-auth",
      onRehydrateStorage: () =&gt; (state) =&gt; {
        if (state?.token) {
          setAuthToken(state.token);
        }
      }
    }
  )
);
```

File: frontend\src\i18n.ts

Full path: C:\Users\itodo\Hostelrec\frontend\src\i18n.ts

```
// @ts-nocheck
import i18n from "i18next";
import { initReactI18next } from "react-i18next";

import en from "./locales/en.json";
import ru from "./locales/ru.json";

i18n.use(initReactI18next).init({
  resources: {
    en: { translation: en },
    ru: { translation: ru }
  },
  lng: "en",
  fallbackLng: "en",
  interpolation: {
    escapeValue: false
  }
});

export default i18n;
```

File: frontend\src\index.css

Full path: C:\Users\itodo\Hostelrec\frontend\src\index.css

```
@tailwind base;
@tailwind components;
@tailwind utilities;

@layer base {
  body {
    font-family: "Inter", system-ui, -apple-system, BlinkMacSystemFont, "Segoe UI", sans-serif;
    background-color: #f8fafc;
    color: #0f172a;
    -webkit-font-smoothing: antialiased;
    -moz-osx-font-smoothing: grayscale;
  }

  * {
    @apply border-slate-200;
  }
}

@layer utilities {
  .shadow-3xl {
    box-shadow: 0 35px 60px -12px rgba(0, 0, 0, 0.25);
  }

  .backdrop-blur-md {
    backdrop-filter: blur(12px);
    -webkit-backdrop-filter: blur(12px);
  }
}
```

File: frontend\src\layouts\AdminLayout.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\layouts\AdminLayout.tsx

```
import { NavLink, Outlet } from "react-router-dom";
import { useTranslation } from "react-i18next";
import { useAuthStore, type AuthState, type Role } from "../hooks/useAuthStore";

const AdminLayout = () =&gt; {
  const role = useAuthStore((state: AuthState) =&gt; state.role);

  const { t } = useTranslation();

  type AdminLink = { to: string; label: string; restricted?: Role };

  const links: AdminLink[] = [
    { to: "/admin", label: t("dashboard") },
    { to: "/admin/guests", label: t("guest_checkin_checkout") || "Check-In/Out" },
    { to: "/admin/orders", label: t("orders_title") || "Orders" },
    { to: "/admin/content", label: t("content_management") || "Content Management" },
    { to: "/admin/staff", label: t("staff_management"), restricted: "manager" },
    { to: "/admin/logs", label: t("audit_logs"), restricted: "manager" },
    { to: "/admin/security", label: t("security"), restricted: "manager" }
  ];

  return (
    &lt;div className="flex min-h-screen bg-background"&gt;
      &lt;aside className="w-64 bg-primary text-white"&gt;
        &lt;div className="p-6 text-2xl font-bold"&gt;HostelRec&lt;/div&gt;
        &lt;nav className="flex flex-col gap-2 p-4"&gt;
          {links.map((link) =&gt; {
            const restricted = link.restricted &amp;&amp; link.restricted !== role;
            return (
              &lt;NavLink
                end
                key={link.to}
                to={link.to}
                className={({ isActive }: { isActive: boolean }) =&gt;
                  `rounded-lg px-3 py-2 text-sm font-semibold ${
                    isActive ? "bg-white/20" : "hover:bg-white/10"
                  } ${restricted ? "pointer-events-none opacity-40" : ""}`
                }
              &gt;
                {link.label}
              &lt;/NavLink&gt;
            );
          })}
        &lt;/nav&gt;
      &lt;/aside&gt;
      &lt;main className="flex-1 p-8"&gt;
        &lt;Outlet /&gt;
      &lt;/main&gt;
    &lt;/div&gt;
  );
};

export default AdminLayout;
```

File: frontend\src\locales\en.json

Full path: C:\Users\itodo\Hostelrec\frontend\src\locales\en.json


```
{
  "welcome": "Welcome to HostelRec",
  "book_btn": "Book Now",
  "best_hostel": "Best Hostel in St. Petersburg",
  "hero_subtitle": "Experience modern comfort, curated social spaces, and smart pricing.",
  "nav_public": "Explore",
  "nav_customer": "For Guests",
  "nav_admin": "Admin",
  "availability_high": "High Demand",
  "availability_normal": "Plenty of rooms",
  "rooms_available": "Rooms Available",
  "rooms_occupied": "Rooms Occupied",
  "login": "Login",
  "logout": "Log out",
  "language": "Language",
  "dates": "Select Dates",
  "guest_details": "Guest Details",
  "confirm_pay": "Confirm &amp;amp; Pay",
  "booking_confirmed": "Booking Confirmed!",
  "booking_failed": "Booking failed",
  "manager_only": "Manager only",
  "login_required": "Please sign in to complete your booking.",
  "signin": "Sign In",
  "signup": "Sign Up",
  "username": "Username",
  "password": "Password",
  "full_name": "Full Name",
  "email": "Email",
  "phone": "Phone",
  "passport": "Passport Number",
  "check_in": "Check-in Date",
  "check_out": "Check-out Date",
  "room_type": "Room Type",
  "price": "Price",
  "amenities": "Amenities",
  "status": "Status",
  "available": "Available",
  "occupied": "Occupied",
  "dashboard": "Dashboard",
  "audit_logs": "Audit Logs",
  "security": "Security",
  "staff_management": "Staff Management",
  "cleaner": "Cleaner",
  "occupancy": "Occupancy",
  "revenue_today": "Revenue Today",
  "total_guests": "Total Guests Today",
  "operations": "Operations",
  "insert": "INSERT",
  "update": "UPDATE",
  "delete": "DELETE",
  "timestamp": "Timestamp",
  "table_name": "Table Name",
  "user_name": "User Name",
  "reveal_keys": "Reveal API Keys",
  "secret_phrase": "Secret Phrase",
  "api_keys": "API Keys",
  "add_staff": "Add Staff",
  "edit_staff": "Edit Staff",
  "delete_staff": "Delete Staff",
  "staff_name": "Staff Name",
  "staff_role": "Role",
  "receptionist": "Receptionist",
  "manager": "Manager",
  "cleaner_role": "Cleaner",
  "assign_role": "Assign Role",
  "save": "Save",
  "cancel": "Cancel",
  "delete_confirm": "Are you sure you want to delete this staff member?",
```

```
"rooms_to_clean": "Rooms to Clean",
"cleaning_status": "Cleaning Status",
"pending": "Pending",
"in_progress": "In Progress",
"completed": "Completed",
"mark_cleaned": "Mark as Cleaned",
"room_number": "Room Number",
"last_cleaned": "Last Cleaned",
"next_cleaning": "Next Cleaning",
"today": "Today",
"yesterday": "Yesterday",
"no_rooms": "No rooms need cleaning",
"all_rooms_clean": "All rooms are clean!",
"select_room": "Select Room",
"guest_name": "Guest Name",
"total_price": "Total Price",
"payment_method": "Payment Method",
"card": "Card",
"cash": "Cash",
"online": "Online",
"back": "Back",
"next": "Next",
"previous": "Previous",
"step": "Step",
"of": "of",
"search_rooms": "Search Rooms",
"filter": "Filter",
"reset": "Reset",
"floor_plan": "Floor Plan",
"todays_occupancy": "Today's Occupancy",
"loading": "Loading...",
"all_rooms": "All Rooms",
"total_shifts": "Days worked",
"toggle_language": "Toggle language",
"lang_en": "EN",
"lang_ru": "RU",
"revenue_by_room_type": "Revenue by Room Type",
"gantt_schedule": "Gantt Schedule",
"stay": "Stay",
"room_premium_ensuite_title": "Premium Ensuite",
"room_premium_ensuite_price": "4200■",
"room_premium_ensuite_bathroom": "Private bathroom",
"room_premium_ensuite_desk": "Desk",
"room_premium_ensuite_wifi": "Fiber Wi-Fi",
"room_shared_6_title": "Shared 6-bed",
"room_shared_6_price": "1500■",
"room_shared_6_lockers": "Lockers",
"room_shared_6_curtains": "Curtains",
"room_shared_6_events": "Community events",
"room_family_suite_title": "Family Suite",
"room_family_suite_price": "5200■",
"room_family_suite_kitchenette": "Kitchenette",
"room_family_suite_tv": "Smart TV",
"invalid_credentials": "Invalid credentials",
"signup_failed": "Signup failed",
"no_account": "No account yet?",
"admin_dashboard_title": "Admin Dashboard",
"revenue_hidden_for_role": "Revenue chart hidden for your role.",
"action": "Action",
"details": "Details",
"security_center": "Security Center",
"invalid_secret": "Invalid secret phrase",
"room_type_deluxe": "Deluxe",
"customer_portal_title": "Welcome, dear guest",
"customer_portal_subtitle": "Browse our most popular rooms, learn about your stay, and reach out if you ha
  ions before arrival.",
"customer_portal_rooms": "Popular room types",
"customer_portal_rooms_help": "Browse our available rooms and find the perfect accommodation for your stay
"customer_portal_stay_info": "Good to know before you arrive",
"customer_portal_checkin": "Check-in from 14:00, check-out until 11:00.",
"customer_portal_night_policy": "Quiet hours after 23:00 to keep the hostel calm for everyone.",
"customer_portal_breakfast": "Optional breakfast is available every morning in the common area.",
"customer_portal_wifi": "High-speed Wi■Fi is available in all rooms and common spaces.",
```

```
"customer_portal_contact": "Contact the reception",
"customer_portal_contact_help": "Have a special request about your booking, arrival time, or room? Send us
  sage.",
"customer_portal_message_placeholder": "Tell us about your arrival time, special requests, or questions...
"customer_portal_send": "Send message",
"customer_floor_label": "Floor",
"customer_order_title": "Order food &amp;amp; drinks",
"customer_order_placeholder": "e.g. Pasta, bottle of water, extra towels",
"customer_order_submit": "Place order",
"customer_order_history": "Your orders",
"customer_order_empty": "You have no orders yet.",
"order_category_food": "Food",
"order_category_drink": "Drink",
"order_category_other": "Other",
"orders_title": "Customer Orders",
"orders_refresh": "Refresh",
"orders_customer": "Customer",
"orders_item": "Item",
"orders_category": "Category",
"orders_quantity": "Qty",
"orders_status": "Status",
"orders_created": "Created",
"orders_mark_in_progress": "In progress",
"orders_mark_served": "Served",
"orders_empty": "No customer orders yet.",
"chat_title": "Live Chat",
"chat_placeholder": "Type a message...",
"chat_connected": "Connected",
"chat_disconnected": "Disconnected",
"content_management": "Content Management",
"content_tab_rooms": "Room Images",
"content_tab_foods": "Food Items",
"content_tab_drinks": "Drink Items",
"content_tab_events": "Events",
"content_tab_promos": "Promotions",
"add": "Add New",
"edit": "Edit",
"name": "Name",
"title": "Title",
"description": "Description",
"location": "Location",
"promo_code": "Promo Code",
"discount_percent": "Discount %",
"nav_menu": "Menu",
"menu_title": "Food &amp;amp; Drinks Menu",
"menu_subtitle": "Delicious meals and refreshing beverages",
"food": "Food",
"drinks": "Drinks",
"menu_empty": "No items available at the moment.",
"upcoming_events": "Upcoming Events",
"current_promos": "Current Promotions",
"guest_checkin_checkout": "Guest Check-In / Check-Out",
"check_in_guest": "Check-In Guest",
"check_out_guest": "Check-Out Guest",
"check_in": "Check-In",
"check_out": "Check-Out",
"checked_in": "Checked In",
"checked_out": "Checked Out",
"guest_name": "Guest Name",
"no_guests": "No guests found.",
"filter_all": "All",
"filter_checked_in": "Checked In",
"filter_checked_out": "Checked Out",
"checkout_notes": "Check-out notes (optional)",
"notes": "Notes",
"change_password": "Change Password",
"change_password_description": "Enter your old password and new password to change",
"old_password": "Old Password",
"new_password": "New Password",
"confirm_password": "Confirm Password",
"password_changed_successfully": "Password changed successfully. You will be redirected to the login page.
"passwords_do_not_match": "Passwords do not match",
"password_too_short": "Password must be at least 6 characters long",
```

```
  "invalid_old_password": "Invalid old password",
  "password_change_failed": "Failed to change password",
  "changing": "Changing...",
  "mfa_settings": "Multi-Factor Authentication Settings",
  "mfa_description": "Multi-factor authentication adds an extra layer of security to your account",
  "mfa_status": "MFA Status",
  "enabled": "Enabled",
  "disabled": "Disabled",
  "enable_mfa": "Enable MFA",
  "disable_mfa": "Disable MFA",
  "verify_mfa_code": "Verify MFA Code",
  "mfa_code": "MFA Code",
  "mfa_code_required": "Multi-factor authentication code required",
  "mfa_enable_description": "To enable MFA, enter your password",
  "mfa_verify_description": "Enter the code sent to your email",
  "mfa_disable_description": "To disable MFA, enter your password",
  "mfa_enabled_successfully": "MFA enabled successfully",
  "mfa_disabled_successfully": "MFA disabled successfully",
  "mfa_code_verified": "MFA code verified successfully",
  "mfa_enable_failed": "Failed to enable MFA",
  "mfa_disable_failed": "Failed to disable MFA",
  "mfa_verify_failed": "Failed to verify MFA code",
  "invalid_mfa_code": "Invalid MFA code",
  "enabling": "Enabling...",
  "disabling": "Disabling...",
  "verifying": "Verifying...",
  "enable": "Enable",
  "disable": "Disable",
  "verify": "Verify",
  "signing_in": "Signing in...",
  "login_failed": "Failed to sign in",
  "Sign In": "Sign In",
  "Sign in": "Sign In",
  "Sign out": "Sign Out",
  "active_sessions": "Active Sessions",
  "manage_active_sessions": "Manage your active sessions",
  "current_session": "Current Session",
  "session": "Session",
  "current": "Current",
  "ip_address": "IP Address",
  "device": "Device",
  "created_at": "Created At",
  "last_activity": "Last Activity",
  "expires_at": "Expires At",
  "end_session": "End Session",
  "end_session_confirm": "Are you sure you want to end this session?",
  "failed_to_end_session": "Failed to end session",
  "failed_to_load_sessions": "Failed to load sessions",
  "no_active_sessions": "No active sessions",
  "unknown": "Unknown",
  "forgot_password": "Forgot Password?",
  "forgot_password_description": "Enter your email and we'll send you a password reset code",
  "reset_password": "Reset Password",
  "reset_password_description": "Enter the code sent to your email and your new password",
  "reset_code": "Reset Code",
  "send_reset_code": "Send Code",
  "reset_code_sent": "Password reset code has been sent to your email",
  "reset_request_failed": "Failed to send reset code",
  "invalid_reset_code": "Invalid or expired reset code",
  "password_reset_successful": "Password reset successfully. You will be redirected to the login page.",
  "password_reset_failed": "Failed to reset password",
  "remember_password": "Remember your password?",
  "sending": "Sending...",
  "resetting": "Resetting..."
}
```

File: frontend\src\locales\ru.json

Full path: C:\Users\itodo\Hostelrec\frontend\src\locales\ru.json


```
{
  "welcome": "██████ ████████████ ■ HostelRec",
  "book_btn": "████████████████",
  "best_hostel": "███████ ███████ ■ ██████-████████████",
  "hero_subtitle": "████████████ ████████, ██████████ ██████████ ■ ██████ ██████████████████.",
  "nav_public": "█████████",
  "nav_customer": "████ ███████",
  "nav_admin": "███████",
  "availability_high": "████████ ███████",
  "availability_normal": "███████ █████████",
  "rooms_available": "████████████ ███████",
  "rooms_occupied": "█████████ ███████",
  "login": "███████",
  "logout": "███████",
  "language": "██████",
  "dates": "███████ ████",
  "guest_details": "███████ ███████",
  "confirm_pay": "████████████████ ■ ███████",
  "booking_confirmed": "███████ ████████████████!",
  "booking_failed": "███████ ████████████████",
  "manager_only": "████████ ██████████",
  "login_required": "████████████, ████████ ■ ████████, ██████ ██████████ ███████.",
  "signin": "███████",
  "signup": "██████████████",
  "username": "████ ████████████████",
  "password": "████████",
  "full_name": "███████ ████",
  "email": "████████████ ███████",
  "phone": "█████████",
  "passport": "███████ ██████████",
  "check_in": "██████ ████████",
  "check_out": "██████ ████████",
  "room_type": "████ ███████",
  "price": "██████",
  "amenities": "██████████",
  "status": "████████",
  "available": "██████████",
  "occupied": "███████",
  "dashboard": "████████ ████████████",
  "audit_logs": "███████ ███████",
  "security": "████████████████",
  "staff_management": "████████████ ████████████",
  "cleaner": "█████████",
  "occupancy": "██████████████████",
  "revenue_today": "███████ █████████",
  "total_guests": "███████ ███████ ████████",
  "operations": "██████████",
  "insert": "█████████",
  "update": "████████████",
  "delete": "██████████",
  "timestamp": "███████████ ███████",
  "table_name": "████ █████████",
  "user_name": "████ ████████████████",
  "reveal_keys": "██████████ API ███████",
  "secret_phrase": "███████████ ███████",
  "api_keys": "API ███████",
  "add_staff": "██████████ ████████████",
  "edit_staff": "████████████████ ████████████",
  "delete_staff": "█████████ ████████████",
  "staff_name": "████ ████████████",
  "staff_role": "██████",
  "receptionist": "████████████████",
  "manager": "██████████",
  "cleaner_role": "█████████",
  "assign_role": "████████████ ██████",
  "save": "████████████",
  "cancel": "████████",
  "delete_confirm": "██ ████████, ███ ███████ ████████ ██████ ████████████?",
```

```
"rooms_to_clean": "[redacted]",
"cleaning_status": "[redacted]",
"pending": "[redacted]",
"in_progress": "[redacted]",
"completed": "[redacted]",
"mark_cleaned": "[redacted]",
"room_number": "[redacted]",
"last_cleaned": "[redacted]",
"next_cleaning": "[redacted]",
"today": "[redacted]",
"yesterday": "[redacted]",
"no_rooms": "[redacted]",
"all_rooms_clean": "[redacted]!",
"select_room": "[redacted]",
"guest_name": "[redacted]",
"total_price": "[redacted]",
"payment_method": "[redacted]",
"card": "[redacted]",
"cash": "[redacted]",
"online": "[redacted]",
"back": "[redacted]",
"next": "[redacted]",
"previous": "[redacted]",
"step": "[redacted]",
"of": "[redacted]",
"search_rooms": "[redacted]",
"filter": "[redacted]",
"reset": "[redacted]",
"floor_plan": "[redacted]",
"todays_occupancy": "[redacted]",
"loading": "[redacted]...",
"all_rooms": "[redacted]",
"total_shifts": "[redacted]",
"toggle_language": "[redacted]",
"lang_en": "EN",
"lang_ru": "RU",
"revenue_by_room_type": "[redacted]",
"gantt_schedule": "[redacted] (Gantt)",
"stay": "[redacted]",
"room_premium_ensuite_title": "[redacted]",
"room_premium_ensuite_price": "4200[redacted]",
"room_premium_ensuite_bathroom": "[redacted]",
"room_premium_ensuite_desk": "[redacted]",
"room_premium_ensuite_wifi": "[redacted] Wi[redacted]Fi",
"room_shared_6_title": "[redacted] 6 [redacted]",
"room_shared_6_price": "1500[redacted]",
"room_shared_6_lockers": "[redacted]",
"room_shared_6_curtains": "[redacted]",
"room_shared_6_events": "[redacted]",
"room_family_suite_title": "[redacted]",
"room_family_suite_price": "5200[redacted]",
"room_family_suite_kitchenette": "[redacted]",
"room_family_suite_tv": "[redacted]TV",
"invalid_credentials": "[redacted]",
"signup_failed": "[redacted]",
"no_account": "[redacted]?",
"admin_dashboard_title": "[redacted]",
"revenue_hidden_for_role": "[redacted].",
"action": "[redacted]",
"details": "[redacted]",
"security_center": "[redacted]",
"invalid_secret": "[redacted]",
"room_type_deluxe": "[redacted]",
"customer_portal_title": "[redacted]",
"customer_portal_subtitle": "[redacted].",
"customer_portal_rooms": "[redacted]",
"customer_portal_rooms_help": "[redacted]
  lRec.",
"customer_portal_stay_info": "[redacted]",
"customer_portal_checkin": "[redacted] 14:00, [redacted] 11:00.",
"customer_portal_night_policy": "[redacted] 23:00, [redacted]",
"customer_portal_breakfast": "[redacted].",
```

```
"customer_portal_wifi": "[redacted] Wi[redacted]Fi [redacted] [redacted] [redacted] [redacted] [redacted] [redacted].",
"customer_portal_contact": "[redacted] [redacted] [redacted]",
"customer_portal_contact_help": "[redacted] [redacted] [redacted] [redacted] [redacted], [redacted] [redacted] [redacted] [redacted]
   [redacted].",
"customer_portal_message_placeholder": "[redacted] [redacted] [redacted], [redacted] [redacted] [redacted] [redacted]
"customer_portal_send": "[redacted] [redacted]",
"customer_floor_label": "[redacted]",
"customer_order_title": "[redacted] [redacted] [redacted]",
"customer_order_placeholder": "[redacted], [redacted], [redacted] [redacted], [redacted] [redacted]",
"customer_order_submit": "[redacted] [redacted]",
"customer_order_history": "[redacted] [redacted]",
"customer_order_empty": "[redacted] [redacted] [redacted] [redacted] [redacted].",
"order_category_food": "[redacted]",
"order_category_drink": "[redacted]",
"order_category_other": "[redacted]",
"orders_title": "[redacted] [redacted]",
"orders_refresh": "[redacted]",
"orders_customer": "[redacted]",
"orders_item": "[redacted]",
"orders_category": "[redacted]",
"orders_quantity": "[redacted]-[redacted]",
"orders_status": "[redacted]",
"orders_created": "[redacted]",
"orders_mark_in_progress": "[redacted] [redacted]",
"orders_mark_served": "[redacted]",
"orders_empty": "[redacted] [redacted] [redacted] [redacted].",
"chat_title": "[redacted] [redacted]",
"chat_placeholder": "[redacted] [redacted]...",
"chat_connected": "[redacted]",
"chat_disconnected": "[redacted]",
"content_management": "[redacted] [redacted]",
"content_tab_rooms": "[redacted] [redacted]",
"content_tab_foods": "[redacted]",
"content_tab_drinks": "[redacted]",
"content_tab_events": "[redacted]",
"content_tab_promos": "[redacted]",
"add": "[redacted]",
"edit": "[redacted]",
"name": "[redacted]",
"title": "[redacted]",
"description": "[redacted]",
"location": "[redacted]",
"promo_code": "[redacted]",
"discount_percent": "[redacted] %",
"nav_menu": "[redacted]",
"menu_title": "[redacted] [redacted] [redacted] [redacted]",
"menu_subtitle": "[redacted] [redacted] [redacted] [redacted] [redacted]",
"food": "[redacted]",
"drinks": "[redacted]",
"menu_empty": "[redacted] [redacted] [redacted] [redacted] [redacted] [redacted].",
"upcoming_events": "[redacted] [redacted]",
"current_promos": "[redacted] [redacted]",
"guest_checkin_checkout": "[redacted] / [redacted] [redacted]",
"check_in_guest": "[redacted] [redacted]",
"check_out_guest": "[redacted] [redacted]",
"check_in": "[redacted]",
"check_out": "[redacted]",
"checked_in": "[redacted]",
"checked_out": "[redacted]",
"guest_name": "[redacted] [redacted]",
"no_guests": "[redacted] [redacted] [redacted].",
"filter_all": "[redacted]",
"filter_checked_in": "[redacted]",
"filter_checked_out": "[redacted]",
"checkout_notes": "[redacted] [redacted] [redacted] ([redacted])",
"notes": "[redacted]",
"change_password": "[redacted] [redacted]",
"change_password_description": "[redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted] [redacted]",
"old_password": "[redacted] [redacted]",
"new_password": "[redacted] [redacted]",
"confirm_password": "[redacted] [redacted]",
"password_changed_successfully": "[redacted] [redacted] [redacted]. [redacted] [redacted] [redacted] [redacted] [redacted]
"passwords_do_not_match": "[redacted] [redacted] [redacted]",
```

```
  "password_too_short": "[redacted] 6 [redacted]",
  "invalid_old_password": "[redacted]",
  "password_change_failed": "[redacted]",
  "changing": "[redacted]...",
  "mfa_settings": "[redacted]",
  "mfa_description": "[redacted]
  "mfa_status": "[redacted] MFA",
  "enabled": "[redacted]",
  "disabled": "[redacted]",
  "enable_mfa": "[redacted] MFA",
  "disable_mfa": "[redacted] MFA",
  "verify_mfa_code": "[redacted] MFA",
  "mfa_code": "[redacted] MFA",
  "mfa_code_required": "[redacted]",
  "mfa_enable_description": "[redacted] MFA [redacted]",
  "mfa_verify_description": "[redacted], [redacted]",
  "mfa_disable_description": "[redacted] MFA [redacted]",
  "mfa_enabled_successfully": "MFA [redacted]",
  "mfa_disabled_successfully": "MFA [redacted]",
  "mfa_code_verified": "[redacted] MFA [redacted]",
  "mfa_enable_failed": "[redacted] MFA",
  "mfa_disable_failed": "[redacted] MFA",
  "mfa_verify_failed": "[redacted] MFA",
  "invalid_mfa_code": "[redacted] MFA",
  "enabling": "[redacted]...",
  "disabling": "[redacted]...",
  "verifying": "[redacted]...",
  "enable": "[redacted]",
  "disable": "[redacted]",
  "verify": "[redacted]",
  "signing_in": "[redacted]...",
  "login_failed": "[redacted]",
  "Sign In": "[redacted]",
  "Sign in": "[redacted]",
  "Sign out": "[redacted]",
  "active_sessions": "[redacted]",
  "manage_active_sessions": "[redacted]",
  "current_session": "[redacted]",
  "session": "[redacted]",
  "current": "[redacted]",
  "ip_address": "IP [redacted]",
  "device": "[redacted]",
  "created_at": "[redacted]",
  "last_activity": "[redacted]",
  "expires_at": "[redacted]",
  "end_session": "[redacted]",
  "end_session_confirm": "[redacted], [redacted]?",
  "failed_to_end_session": "[redacted]",
  "failed_to_load_sessions": "[redacted]",
  "no_active_sessions": "[redacted]",
  "unknown": "[redacted]",
  "forgot_password": "[redacted]?",
  "forgot_password_description": "[redacted] email, [redacted]
  "reset_password": "[redacted]",
  "reset_password_description": "[redacted], [redacted] email, [redacted]",
  "reset_code": "[redacted]",
  "send_reset_code": "[redacted]",
  "reset_code_sent": "[redacted] email",
  "reset_request_failed": "[redacted]",
  "invalid_reset_code": "[redacted]",
  "password_reset_successful": "[redacted]. [redacted]
  "password_reset_failed": "[redacted]",
  "remember_password": "[redacted]?",
  "sending": "[redacted]...",
  "resetting": "[redacted]..."
}
```

File: frontend\src\main.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\main.tsx

```
import React from "react";
import ReactDOM from "react-dom/client";
import { BrowserRouter } from "react-router-dom";
import { I18nextProvider } from "react-i18next";

import App from "./App";
import "./index.css";
import i18n from "./i18n";

ReactDOM.createRoot(document.getElementById("root")!).render(
  &lt;React.StrictMode&gt;
    &lt;I18nextProvider i18n={i18n}&gt;
      &lt;BrowserRouter&gt;
        &lt;App /&gt;
      &lt;/BrowserRouter&gt;
    &lt;/I18nextProvider&gt;
  &lt;/React.StrictMode&gt;
);
```

File: frontend\src\pages\AdminDashboard.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\pages\AdminDashboard.tsx

```
import { useEffect, useState } from "react";
import { useTranslation } from "react-i18next";

import KpiCard from "../components/KpiCard";
import RevenueChart from "../components/RevenueChart";
import GanttTimeline from "../components/GanttTimeline";
import { api } from "../services/api";
import { useAuthStore, type AuthState } from "../hooks/useAuthStore";

type Occupancy = {
  occupancy_rate: number;
  available_rooms: number;
  occupied_rooms: number;
};

type Revenue = { room_type: string; revenue: number };

const AdminDashboard = () =&gt; {
  const { t } = useTranslation();
  const role = useAuthStore((state: AuthState) =&gt; state.role);
  const [occupancy, setOccupancy] = useState&lt;Occupancy | null&gt;(null);
  const [revenue, setRevenue] = useState&lt;Revenue[]&gt;([]);

  useEffect(() =&gt; {
    api.get("/stats/occupancy").then((res: { data: Occupancy }) =&gt; setOccupancy(res.data));
    if (role === "manager") {
      api.get("/stats/revenue").then((res: { data: Revenue[] }) =&gt; setRevenue(res.data));
    }
  }, [role]);

  return (
    &lt;div className="space-y-6"&gt;
      &lt;h1 className="text-3xl font-bold text-primary"&gt;{t("admin_dashboard_title")}&lt;/h1&gt;
      &lt;div className="grid gap-4 md:grid-cols-3"&gt;
        &lt;KpiCard label={t("occupancy")} value={`${Math.round((occupancy?.occupancy_rate ?? 0) * 100)}%`}
        &lt;KpiCard label={t("rooms_available")} value={`${occupancy?.available_rooms ?? "--"}`} /&gt;
        {role === "manager" &amp;&amp; (
          &lt;KpiCard label={t("revenue_today")} value="■ 180,000" accent="text-accent" /&gt;
        )}
      &lt;/div&gt;

      &lt;div className="grid gap-6 md:grid-cols-2"&gt;
        {role === "manager" ? (
          &lt;RevenueChart data={revenue.length ? revenue : [{ room_type: t("room_type_deluxe"), revenue: 75
        ) : (
          &lt;div className="rounded-2xl bg-slate-100 p-6 text-slate-500"&gt;
            {t("revenue_hidden_for_role")}
          &lt;/div&gt;
        )}
        &lt;GanttTimeline
          data={[
            { room: "101", start: 10, end: 55, color: "#10B981" },
            { room: "102", start: 30, end: 80, color: "#F59E0B" },
            { room: "201", start: 0, end: 40, color: "#0EA5E9" }
          ]}
        /&gt;
      &lt;/div&gt;
    &lt;/div&gt;
  );
};

export default AdminDashboard;
```

File: frontend\src\pages\AuditLogPage.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\pages\AuditLogPage.tsx

```
import { useEffect, useState } from "react";
import { useTranslation } from "react-i18next";

import { api } from "../services/api";
import { useAuthStore } from "../hooks/useAuthStore";

type AuditLog = {
  event_time: string;
  username: string;
  action: "INSERT" | "UPDATE" | "DELETE";
  details: string;
};

const badgeMap: Record&lt;AuditLog["action"], string&gt; = {
  INSERT: "bg-success/10 text-success",
  UPDATE: "bg-amber-100 text-amber-700",
  DELETE: "bg-danger/10 text-danger"
};

const AuditLogPage = () =&gt; {
  const { t } = useTranslation();
  const role = useAuthStore((s: { role: "manager" | "receptionist" | "cleaner" | null }) =&gt; s.role);
  const [logs, setLogs] = useState&lt;AuditLog[]&gt;([]);

  useEffect(() =&gt; {
    if (role === "manager") {
      api.get("/admin/logs").then((res: { data: AuditLog[] }) =&gt; setLogs(res.data));
    }
  }, [role]);

  if (role !== "manager") {
    return &lt;div className="rounded-2xl bg-white p-6 shadow"&gt;{t("manager_only")}&lt;/div&gt;;
  }

  return (
    &lt;div className="space-y-4"&gt;
      &lt;h2 className="text-2xl font-bold text-primary"&gt;{t("audit_logs")}&lt;/h2&gt;
      &lt;div className="overflow-hidden rounded-2xl bg-white shadow"&gt;
        &lt;table className="min-w-full text-left text-sm"&gt;
          &lt;thead className="bg-slate-50 text-xs uppercase tracking-wide text-slate-500"&gt;
            &lt;tr&gt;
              &lt;th className="px-4 py-3"&gt;{t("timestamp")}&lt;/th&gt;
              &lt;th className="px-4 py-3"&gt;{t("user_name")}&lt;/th&gt;
              &lt;th className="px-4 py-3"&gt;{t("action")}&lt;/th&gt;
              &lt;th className="px-4 py-3"&gt;{t("details")}&lt;/th&gt;
            &lt;/tr&gt;
          &lt;/thead&gt;
          &lt;tbody&gt;
            {logs.map((log) =&gt; (
              &lt;tr key={`${log.event_time}-${log.username}`} className="border-b border-slate-100"&gt;
                &lt;td className="px-4 py-3"&gt;{new Date(log.event_time).toLocaleString()}&lt;/td&gt;
                &lt;td className="px-4 py-3 font-semibold"&gt;{log.username}&lt;/td&gt;
                &lt;td className="px-4 py-3"&gt;
                  &lt;span className={`rounded-full px-3 py-1 text-xs font-bold ${badgeMap[log.action]}`}&gt
                    {log.action}
                  &lt;/span&gt;
                &lt;/td&gt;
                &lt;td className="px-4 py-3 text-slate-500"&gt;{log.details}&lt;/td&gt;
              &lt;/tr&gt;
            ))}
          &lt;/tbody&gt;
        &lt;/table&gt;
      &lt;/div&gt;
    &lt;/div&gt;
  );
};

export default AuditLogPage;
```

File: frontend\src\pages\CheckInOutPage.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\pages\CheckInOutPage.tsx

```
// @ts-nocheck
import { useState, useEffect } from "react";
import { motion } from "framer-motion";
import { useTranslation } from "react-i18next";
import { api } from "../services/api";

type GuestCheckIn = {
  id: number;
  guest_name: string;
  guest_email: string;
  guest_passport: string;
  room_number: string;
  check_in_date: string;
  check_out_date: string | null;
  status: string;
  notes: string;
  checked_in_by_name: string;
  checked_out_by_name: string | null;
};

const CheckInOutPage = () =&gt; {
  const { t } = useTranslation();
  const [guests, setGuests] = useState&lt;GuestCheckIn[]&gt;([]);
  const [filter, setFilter] = useState&lt;"all" | "checked_in" | "checked_out"&gt;("all");
  const [showCheckInModal, setShowCheckInModal] = useState(false);
  const [showCheckOutModal, setShowCheckOutModal] = useState(false);
  const [selectedGuest, setSelectedGuest] = useState&lt;GuestCheckIn | null&gt;(null);
  const [checkInForm, setCheckInForm] = useState({
    guest_name: "",
    guest_email: "",
    guest_passport: "",
    room_number: "",
    notes: "",
  });
  const [checkOutNotes, setCheckOutNotes] = useState("");

  useEffect(() =&gt; {
    loadGuests();
  }, [filter]);

  const loadGuests = async () =&gt; {
    try {
      const params = filter === "all" ? {} : { status_filter: filter };
      const res = await api.get("/guests/", { params });
      setGuests(res.data);
    } catch (err) {
      console.error("Failed to load guests", err);
    }
  };

  const handleCheckIn = async (e: any) =&gt; {
    e.preventDefault();
    try {
      const formData = new FormData();
      formData.append("guest_name", checkInForm.guest_name);
      if (checkInForm.guest_email) formData.append("guest_email", checkInForm.guest_email);
      if (checkInForm.guest_passport) formData.append("guest_passport", checkInForm.guest_passport);
      if (checkInForm.room_number) formData.append("room_number", checkInForm.room_number);
      if (checkInForm.notes) formData.append("notes", checkInForm.notes);

      await api.post("/guests/check-in", formData, {
        headers: { "Content-Type": "multipart/form-data" },
      });
      setShowCheckInModal(false);
      setCheckInForm({
        guest_name: "",
        guest_email: "",
        guest_passport: "",
```

```
      room_number: "",
      notes: "",
    });
    loadGuests();
  } catch (err) {
    console.error("Failed to check in guest", err);
    alert("Failed to check in guest");
  }
};

const handleCheckOut = async (e: any) =&gt; {
  e.preventDefault();
  if (!selectedGuest) return;
  try {
    const formData = new FormData();
    if (checkOutNotes) {
      formData.append("notes", checkOutNotes);
    }
    await api.post(`/guests/${selectedGuest.id}/check-out`, formData, {
      headers: { "Content-Type": "multipart/form-data" },
    });
    setShowCheckOutModal(false);
    setSelectedGuest(null);
    setCheckOutNotes("");
    loadGuests();
  } catch (err) {
    console.error("Failed to check out guest", err);
    alert("Failed to check out guest");
  }
};

const openCheckOut = (guest: GuestCheckIn) =&gt; {
  setSelectedGuest(guest);
  setShowCheckOutModal(true);
};

return (
  &lt;div className="space-y-6"&gt;
    &lt;div className="flex items-center justify-between"&gt;
      &lt;h1 className="text-3xl font-bold text-primary"&gt;
        {t("guest_checkin_checkout") || "Guest Check-In / Check-Out"}
      &lt;/h1&gt;
      &lt;motion.button
        whileHover={{ scale: 1.05 }}
        whileTap={{ scale: 0.95 }}
        onClick={() =&gt; setShowCheckInModal(true)}
        className="rounded-lg bg-primary px-4 py-2 text-white font-semibold"
      &gt;
        {t("check_in_guest") || "Check-In Guest"}
      &lt;/motion.button&gt;
    &lt;/div&gt;

    {/* Filter Tabs */}
    &lt;div className="flex gap-2 border-b border-slate-200"&gt;
      {(["all", "checked_in", "checked_out"] as const).map((f) =&gt; (
        &lt;button
          key={f}
          onClick={() =&gt; setFilter(f)}
          className={`px-4 py-2 font-semibold border-b-2 transition-colors ${
            filter === f
              ? "border-primary text-primary"
              : "border-transparent text-slate-600 hover:text-primary"
          }`}
        &gt;
          {t(`filter_${f}`) || f.replace("_", " ").replace(/\b\w/g, (l) =&gt; l.toUpperCase())}
        &lt;/button&gt;
      ))}
    &lt;/div&gt;

    {/* Guests Table */}
    &lt;div className="bg-white rounded-lg shadow-md overflow-hidden"&gt;
      &lt;table className="w-full text-sm"&gt;
        &lt;thead className="bg-primary text-white"&gt;
```

```
      &lt;tr&gt;
        &lt;th className="px-4 py-3 text-left"&gt;{t("guest_name") || "Guest Name"}&lt;/th&gt;
        &lt;th className="px-4 py-3 text-left"&gt;{t("email")}&lt;/th&gt;
        &lt;th className="px-4 py-3 text-left"&gt;{t("room_number")}&lt;/th&gt;
        &lt;th className="px-4 py-3 text-left"&gt;{t("check_in")}&lt;/th&gt;
        &lt;th className="px-4 py-3 text-left"&gt;{t("check_out")}&lt;/th&gt;
        &lt;th className="px-4 py-3 text-left"&gt;{t("status")}&lt;/th&gt;
        &lt;th className="px-4 py-3 text-right"&gt;{t("operations")}&lt;/th&gt;
      &lt;/tr&gt;
    &lt;/thead&gt;
    &lt;tbody&gt;
      {guests.map((guest) =&gt; (
        &lt;tr key={guest.id} className="border-b hover:bg-slate-50"&gt;
          &lt;td className="px-4 py-3 font-semibold"&gt;{guest.guest_name}&lt;/td&gt;
          &lt;td className="px-4 py-3"&gt;{guest.guest_email || "-"}&lt;/td&gt;
          &lt;td className="px-4 py-3"&gt;{guest.room_number}&lt;/td&gt;
          &lt;td className="px-4 py-3"&gt;
            {new Date(guest.check_in_date).toLocaleString()}
          &lt;/td&gt;
          &lt;td className="px-4 py-3"&gt;
            {guest.check_out_date
              ? new Date(guest.check_out_date).toLocaleString()
              : "-"}
          &lt;/td&gt;
          &lt;td className="px-4 py-3"&gt;
            &lt;span
              className={`px-2 py-1 rounded text-xs font-semibold ${
                guest.status === "checked_in"
                  ? "bg-green-100 text-green-700"
                  : "bg-slate-100 text-slate-700"
              }`}
            &gt;
              {guest.status === "checked_in"
                ? t("checked_in") || "Checked In"
                : t("checked_out") || "Checked Out"}
            &lt;/span&gt;
          &lt;/td&gt;
          &lt;td className="px-4 py-3 text-right"&gt;
            {guest.status === "checked_in" &amp;&amp; (
              &lt;motion.button
                whileHover={{ scale: 1.05 }}
                whileTap={{ scale: 0.95 }}
                onClick={() =&gt; openCheckOut(guest)}
                className="text-accent hover:underline font-semibold"
              &gt;
                {t("check_out") || "Check-Out"}
              &lt;/motion.button&gt;
            )}
          &lt;/td&gt;
        &lt;/tr&gt;
      ))}
      {guests.length === 0 &amp;&amp; (
        &lt;tr&gt;
          &lt;td
            colSpan={7}
            className="px-4 py-6 text-center text-slate-500 text-sm"
          &gt;
            {t("no_guests") || "No guests found."}
          &lt;/td&gt;
        &lt;/tr&gt;
      )}
    &lt;/tbody&gt;
  &lt;/table&gt;
&lt;/div&gt;

{/* Check-In Modal */}
{showCheckInModal &amp;&amp; (
  &lt;div className="fixed inset-0 bg-black/50 flex items-center justify-center z-50"&gt;
    &lt;motion.div
      initial={{ opacity: 0, scale: 0.9 }}
      animate={{ opacity: 1, scale: 1 }}
      className="bg-white rounded-2xl p-6 max-w-md w-full"
    &gt;
```

```
    &lt;h3 className="text-xl font-bold mb-4"&gt;
      {t("check_in_guest") || "Check-In Guest"}
    &lt;/h3&gt;
    &lt;form onSubmit={handleCheckIn} className="space-y-4"&gt;
      &lt;input
        type="text"
        placeholder={t("guest_name") || "Guest Name"}
        value={checkInForm.guest_name}
        onChange={(e) =&gt;
          setCheckInForm({ ...checkInForm, guest_name: e.target.value })
        }
        required
        className="w-full rounded-xl border border-slate-200 px-4 py-2"
      /&gt;
      &lt;input
        type="email"
        placeholder={t("email")}
        value={checkInForm.guest_email}
        onChange={(e) =&gt;
          setCheckInForm({ ...checkInForm, guest_email: e.target.value })
        }
        className="w-full rounded-xl border border-slate-200 px-4 py-2"
      /&gt;
      &lt;input
        type="text"
        placeholder={t("passport") || "Passport Number"}
        value={checkInForm.guest_passport}
        onChange={(e) =&gt;
          setCheckInForm({ ...checkInForm, guest_passport: e.target.value })
        }
        className="w-full rounded-xl border border-slate-200 px-4 py-2"
      /&gt;
      &lt;input
        type="text"
        placeholder={t("room_number")}
        value={checkInForm.room_number}
        onChange={(e) =&gt;
          setCheckInForm({ ...checkInForm, room_number: e.target.value })
        }
        required
        className="w-full rounded-xl border border-slate-200 px-4 py-2"
      /&gt;
      &lt;textarea
        placeholder={t("notes") || "Notes (optional)"}
        value={checkInForm.notes}
        onChange={(e) =&gt;
          setCheckInForm({ ...checkInForm, notes: e.target.value })
        }
        className="w-full rounded-xl border border-slate-200 px-4 py-2"
        rows={3}
      /&gt;
      &lt;div className="flex gap-2"&gt;
        &lt;motion.button
          whileHover={{ scale: 1.05 }}
          whileTap={{ scale: 0.95 }}
          type="submit"
          className="flex-1 rounded-xl bg-primary px-4 py-2 text-white font-semibold"
        &gt;
          {t("check_in") || "Check-In"}
        &lt;/motion.button&gt;
        &lt;motion.button
          whileHover={{ scale: 1.05 }}
          whileTap={{ scale: 0.95 }}
          type="button"
          onClick={() =&gt; setShowCheckInModal(false)}
          className="flex-1 rounded-xl border border-slate-200 px-4 py-2 font-semibold"
        &gt;
          {t("cancel")}
        &lt;/motion.button&gt;
      &lt;/div&gt;
    &lt;/form&gt;
  &lt;/motion.div&gt;
&lt;/div&gt;
```

```
      )}

      {/* Check-Out Modal */}
      {showCheckOutModal &amp;&amp; selectedGuest &amp;&amp; (
        &lt;div className="fixed inset-0 bg-black/50 flex items-center justify-center z-50"&gt;
          &lt;motion.div
            initial={{ opacity: 0, scale: 0.9 }}
            animate={{ opacity: 1, scale: 1 }}
            className="bg-white rounded-2xl p-6 max-w-md w-full"
          &gt;
            &lt;h3 className="text-xl font-bold mb-4"&gt;
              {t("check_out_guest") || "Check-Out Guest"}
            &lt;/h3&gt;
            &lt;div className="mb-4 space-y-2"&gt;
              &lt;p&gt;
                &lt;span className="font-semibold"&gt;{t("guest_name") || "Guest"}:&lt;/span&gt;{" "}
                {selectedGuest.guest_name}
              &lt;/p&gt;
              &lt;p&gt;
                &lt;span className="font-semibold"&gt;{t("room_number")}:&lt;/span&gt;{" "}
                {selectedGuest.room_number}
              &lt;/p&gt;
            &lt;/div&gt;
            &lt;form onSubmit={handleCheckOut} className="space-y-4"&gt;
              &lt;textarea
                placeholder={t("checkout_notes") || "Check-out notes (optional)"}
                value={checkOutNotes}
                onChange={(e) =&gt; setCheckOutNotes(e.target.value)}
                className="w-full rounded-xl border border-slate-200 px-4 py-2"
                rows={3}
              /&gt;
              &lt;div className="flex gap-2"&gt;
                &lt;motion.button
                  whileHover={{ scale: 1.05 }}
                  whileTap={{ scale: 0.95 }}
                  type="submit"
                  className="flex-1 rounded-xl bg-accent px-4 py-2 text-white font-semibold"
                &gt;
                  {t("check_out") || "Check-Out"}
                &lt;/motion.button&gt;
                &lt;motion.button
                  whileHover={{ scale: 1.05 }}
                  whileTap={{ scale: 0.95 }}
                  type="button"
                  onClick={() =&gt; {
                    setShowCheckOutModal(false);
                    setSelectedGuest(null);
                  }}
                  className="flex-1 rounded-xl border border-slate-200 px-4 py-2 font-semibold"
                &gt;
                  {t("cancel")}
                &lt;/motion.button&gt;
              &lt;/div&gt;
            &lt;/form&gt;
          &lt;/motion.div&gt;
        &lt;/div&gt;
      )}
    &lt;/div&gt;
  );
};

export default CheckInOutPage;
```

File: frontend\src\pages\CleanerPage.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\pages\CleanerPage.tsx

```
import { useEffect, useState } from "react";
import { useTranslation } from "react-i18next";
import { api } from "../services/api";

type RoomCleaning = {
  room_number: string;
  status: "pending" | "in_progress" | "completed";
  last_cleaned: string | null;
  next_cleaning: string | null;
  guest_name: string | null;
};

const CleanerPage = () =&gt; {
  const { t } = useTranslation();
  const [rooms, setRooms] = useState&lt;RoomCleaning[]&gt;([]);
  const [loading, setLoading] = useState(true);

  useEffect(() =&gt; {
    loadRooms();
  }, []);

  const loadRooms = async () =&gt; {
    try {
      // Fetch real rooms from API
      const roomsRes = await api.get("/content/rooms");
      const rooms: RoomCleaning[] = roomsRes.data.map((room: any) =&gt; ({
        room_number: room.room_number,
        status: room.status === "occupied" ? "pending" : "completed",
        last_cleaned: room.updated_at ? new Date(room.updated_at).toISOString().split('T')[0] : null,
        next_cleaning: null, // Can be calculated based on check-out dates
        guest_name: room.status === "occupied" ? "Guest" : null
      }));

      // Try to get guest check-ins to populate guest names
      try {
        const checkinsRes = await api.get("/guests/", { params: { status_filter: "checked_in" } });
        const checkins = checkinsRes.data;
        const checkinsMap = new Map(checkins.map((c: any) =&gt; [c.room_number, c.guest_name]));

        rooms.forEach(room =&gt; {
          if (room.status === "pending" &amp;&amp; checkinsMap.has(room.room_number)) {
            const guestName = checkinsMap.get(room.room_number);
            room.guest_name = guestName ? String(guestName) : null;
          }
        });
      } catch (err) {
        console.error("Failed to load guest check-ins", err);
      }

      setRooms(rooms);
    } catch (err) {
      console.error("Failed to load rooms", err);
      setRooms([]);
    } finally {
      setLoading(false);
    }
  };

  const handleMarkCleaned = async (roomNumber: string) =&gt; {
    try {
      // Find room ID from room number
      const room = rooms.find(r =&gt; r.room_number === roomNumber);
      if (!room) {
        alert("Room not found");
        return;
      }

      // Get room ID from API
```

```
    const roomsRes = await api.get("/content/rooms");
    const allRooms = roomsRes.data;
    const roomData = allRooms.find((r: any) =&gt; r.room_number === roomNumber);

    if (roomData &amp;&amp; roomData.id) {
      await api.post(`/content/rooms/${roomData.id}/mark-cleaned`);
      // Reload rooms to get updated data
      loadRooms();
      alert("Room marked as cleaned successfully!");
    } else {
      // Fallback: just update local state if API doesn't return ID
      setRooms(rooms.map(r =&gt;
        r.room_number === roomNumber
          ? { ...r, status: "completed", last_cleaned: new Date().toISOString().split('T')[0] }
          : r
      ));
    }
  } catch (err) {
    console.error("Failed to mark room as cleaned", err);
    alert("Failed to mark room as cleaned");
  }
};

const getStatusColor = (status: string) =&gt; {
  switch (status) {
    case "pending": return "bg-danger/20 text-danger";
    case "in_progress": return "bg-accent/20 text-accent";
    case "completed": return "bg-success/20 text-success";
    default: return "bg-slate-200 text-slate-700";
  }
};

const pendingRooms = rooms.filter(r =&gt; r.status === "pending" || r.status === "in_progress");

if (loading) {
  return &lt;div className="text-center py-10"&gt;{t("loading") || "Loading..."}&lt;/div&gt;;
}

return (
  &lt;div className="min-h-screen bg-background p-8"&gt;
    &lt;div className="max-w-6xl mx-auto space-y-6"&gt;
      &lt;div className="flex items-center justify-between"&gt;
        &lt;h1 className="text-3xl font-bold text-primary"&gt;{t("cleaner")}&lt;/h1&gt;
        &lt;div className="text-sm text-slate-600"&gt;
          {pendingRooms.length} {t("rooms_to_clean")}
        &lt;/div&gt;
      &lt;/div&gt;

      {pendingRooms.length === 0 ? (
        &lt;div className="bg-white rounded-lg shadow-md p-12 text-center"&gt;
          &lt;p className="text-xl text-slate-600"&gt;{t("all_rooms_clean")}&lt;/p&gt;
        &lt;/div&gt;
      ) : (
        &lt;div className="grid gap-4 md:grid-cols-2 lg:grid-cols-3"&gt;
          {pendingRooms.map((room) =&gt; (
            &lt;div key={room.room_number} className="bg-white rounded-lg shadow-md p-6"&gt;
              &lt;div className="flex items-center justify-between mb-4"&gt;
                &lt;h3 className="text-2xl font-bold text-primary"&gt;
                  {t("room_number")} {room.room_number}
                &lt;/h3&gt;
                &lt;span className={`px-3 py-1 rounded-full text-sm font-semibold ${getStatusColor(room.st
                  {t(room.status)}
                &lt;/span&gt;
              &lt;/div&gt;

              {room.guest_name &amp;&amp; (
                &lt;p className="text-sm text-slate-600 mb-2"&gt;
                  {t("guest_name")}: {room.guest_name}
                &lt;/p&gt;
              )}

              &lt;div className="space-y-2 text-sm mb-4"&gt;
                {room.last_cleaned &amp;&amp; (
```

```
                  &lt;p className="text-slate-600"&gt;
                    {t("last_cleaned")}: {room.last_cleaned}
                  &lt;/p&gt;
                )}
                {room.next_cleaning &amp;&amp; (
                  &lt;p className="text-slate-600"&gt;
                    {t("next_cleaning")}: {room.next_cleaning}
                  &lt;/p&gt;
                )}
              &lt;/div&gt;

              {room.status !== "completed" &amp;&amp; (
                &lt;button
                  onClick={() =&gt; handleMarkCleaned(room.room_number)}
                  className="w-full rounded-lg bg-success px-4 py-2 text-white font-semibold hover:bg-emer
    sition-colors"
                &gt;
                  {t("mark_cleaned")}
                &lt;/button&gt;
              )}
            &lt;/div&gt;
          ))}
        &lt;/div&gt;
      )}

      &lt;div className="bg-white rounded-lg shadow-md p-6"&gt;
        &lt;h2 className="text-xl font-bold mb-4"&gt;{t("all_rooms") || "All Rooms"}&lt;/h2&gt;
        &lt;div className="overflow-x-auto"&gt;
          &lt;table className="w-full"&gt;
            &lt;thead className="bg-primary text-white"&gt;
              &lt;tr&gt;
                &lt;th className="px-4 py-2 text-left"&gt;{t("room_number")}&lt;/th&gt;
                &lt;th className="px-4 py-2 text-left"&gt;{t("cleaning_status")}&lt;/th&gt;
                &lt;th className="px-4 py-2 text-left"&gt;{t("last_cleaned")}&lt;/th&gt;
                &lt;th className="px-4 py-2 text-left"&gt;{t("next_cleaning")}&lt;/th&gt;
              &lt;/tr&gt;
            &lt;/thead&gt;
            &lt;tbody&gt;
              {rooms.map((room) =&gt; (
                &lt;tr key={room.room_number} className="border-b hover:bg-slate-50"&gt;
                  &lt;td className="px-4 py-2 font-semibold"&gt;{room.room_number}&lt;/td&gt;
                  &lt;td className="px-4 py-2"&gt;
                    &lt;span className={`px-2 py-1 rounded text-xs ${getStatusColor(room.status)}`}&gt;
                      {t(room.status)}
                    &lt;/span&gt;
                  &lt;/td&gt;
                  &lt;td className="px-4 py-2 text-slate-600"&gt;{room.last_cleaned || "-"}&lt;/td&gt;
                  &lt;td className="px-4 py-2 text-slate-600"&gt;{room.next_cleaning || "-"}&lt;/td&gt;
                &lt;/tr&gt;
              ))}
            &lt;/tbody&gt;
          &lt;/table&gt;
        &lt;/div&gt;
      &lt;/div&gt;
    &lt;/div&gt;
  );
};

export default CleanerPage;
```

File: frontend\src\pages\ContentManagementPage.tsx

Full path:
C:\Users\itodo\Hostelrec\frontend\src\pages\ContentManagementPage.tsx

```
// @ts-nocheck
import { useState, useEffect } from "react";
import { motion } from "framer-motion";
import { useTranslation } from "react-i18next";
import { api } from "../services/api";

type FoodItem = {
  id: number;
  name: string;
  description: string;
  price: number;
  image_url: string;
  category: string;
  available: boolean;
  is_tea: boolean;
  contains_alcohol: boolean;
};

type DrinkItem = {
  id: number;
  name: string;
  description: string;
  price: number;
  image_url: string;
  category: string;
  available: boolean;
  is_tea: boolean;
  contains_alcohol: boolean;
};

type Event = {
  id: number;
  title: string;
  description: string;
  event_date: string;
  location: string;
  image_url: string;
  active: boolean;
};

type Promo = {
  id: number;
  title: string;
  description: string;
  discount_percent: number;
  discount_amount: number;
  code: string;
  valid_from: string;
  valid_until: string;
  image_url: string;
  active: boolean;
};

type RoomImage = {
  id: number;
  room_number: string;
  image_url: string;
  description: string;
};

type Room = {
  id: number;
  room_number: string;
  title: string;
  description: string | null;
  price: number;
  location: string | null;
```

```
  wifi: boolean;
  features: string[] | null;
  image_url: string | null;
  status: string;
};

const ContentManagementPage = () =&gt; {
  const { t } = useTranslation();
  const [activeTab, setActiveTab] = useState&lt;"rooms" | "foods" | "drinks" | "events" | "promos"&gt;("room
  const [foods, setFoods] = useState&lt;FoodItem[]&gt;([]);
  const [drinks, setDrinks] = useState&lt;DrinkItem[]&gt;([]);
  const [events, setEvents] = useState&lt;Event[]&gt;([]);
  const [promos, setPromos] = useState&lt;Promo[]&gt;([]);
  const [roomImages, setRoomImages] = useState&lt;RoomImage[]&gt;([]);
  const [rooms, setRooms] = useState&lt;Room[]&gt;([]);
  const [showModal, setShowModal] = useState(false);
  const [editing, setEditing] = useState&lt;any&gt;(null);

  useEffect(() =&gt; {
    loadData();
  }, [activeTab]);

  const loadData = async () =&gt; {
    try {
      if (activeTab === "foods") {
        const res = await api.get("/content/foods");
        setFoods(res.data);
      } else if (activeTab === "drinks") {
        const res = await api.get("/content/drinks");
        setDrinks(res.data);
      } else if (activeTab === "events") {
        const res = await api.get("/content/events");
        setEvents(res.data);
      } else if (activeTab === "promos") {
        const res = await api.get("/content/promos");
        setPromos(res.data);
      } else if (activeTab === "rooms") {
        const res = await api.get("/content/rooms");
        setRooms(res.data);
        const imgRes = await api.get("/content/rooms/images");
        setRoomImages(imgRes.data);
      }
    } catch (err) {
      console.error("Failed to load data", err);
    }
  };

  const handleSubmit = async (file: File | null, category: string, data: any, isEditing: boolean) =&gt; {
    try {
      if (isEditing &amp;&amp; data.id) {
        // For updates
        if (category === "rooms" &amp;&amp; file) {
          // Rooms can handle file uploads in PUT
          const formData = new FormData();
          formData.append("file", file);
          Object.keys(data).forEach((key) =&gt; {
            if (key !== "id" &amp;&amp; key !== "file" &amp;&amp; key !== "image_url" &amp;&amp; data[key] !
  ;&amp;&amp; data[key] !== undefined) {
              // Convert boolean to string properly for FastAPI Form fields
              if (typeof data[key] === "boolean") {
                formData.append(key, data[key] ? "true" : "false");
              } else {
                formData.append(key, data[key].toString());
              }
            }
          });
          await api.put(`/content/rooms/${data.id}`, formData, {
            headers: { "Content-Type": "multipart/form-data" },
          });
        } else {
          // Other categories or rooms without file - use regular PUT
          const updateData: any = {};
          Object.keys(data).forEach((key) =&gt; {
```

```
        if (key !== "id" &amp;&amp; key !== "file" &amp;&amp; key !== "image_url" &amp;&amp; data[key] !
;&amp; data[key] !== undefined) {
          updateData[key] = data[key];
        }
      });

      if (category === "foods") {
        await api.put(`/content/foods/${data.id}`, updateData);
      } else if (category === "drinks") {
        await api.put(`/content/drinks/${data.id}`, updateData);
      } else if (category === "events") {
        await api.put(`/content/events/${data.id}`, updateData);
      } else if (category === "promos") {
        await api.put(`/content/promos/${data.id}`, updateData);
      } else if (category === "rooms") {
        await api.put(`/content/rooms/${data.id}`, updateData);
      }

      // Note: For foods/drinks/events/promos, image updates would require recreation
      if (file &amp;&amp; category !== "rooms") {
        alert("Note: Image updates for this item type require deleting and recreating the item.");
      }
    }
  } else {
    // For new items, use POST with file upload
    const formData = new FormData();
    if (file) {
      formData.append("file", file);
    }
    Object.keys(data).forEach((key) =&gt; {
      if (key !== "id" &amp;&amp; key !== "file" &amp;&amp; key !== "image_url" &amp;&amp; data[key] !==
amp; data[key] !== undefined) {
        // Convert boolean to string properly for FastAPI Form fields
        if (typeof data[key] === "boolean") {
          formData.append(key, data[key] ? "true" : "false");
        } else if (key === "price" &amp;&amp; typeof data[key] === "number") {
          // Ensure price is sent as a proper number string
          formData.append(key, data[key].toString());
        } else {
          formData.append(key, data[key].toString());
        }
      }
    });

    if (category === "foods") {
      await api.post("/content/foods", formData, {
        headers: { "Content-Type": "multipart/form-data" },
      });
    } else if (category === "drinks") {
      await api.post("/content/drinks", formData, {
        headers: { "Content-Type": "multipart/form-data" },
      });
    } else if (category === "events") {
      await api.post("/content/events", formData, {
        headers: { "Content-Type": "multipart/form-data" },
      });
    } else if (category === "promos") {
      await api.post("/content/promos", formData, {
        headers: { "Content-Type": "multipart/form-data" },
      });
    } else if (category === "rooms") {
      if (data.room_number &amp;&amp; data.title &amp;&amp; data.price) {
        await api.post("/content/rooms", formData, {
          headers: { "Content-Type": "multipart/form-data" },
        });
      } else {
        await api.post("/content/rooms/images", formData, {
          headers: { "Content-Type": "multipart/form-data" },
        });
      }
    }
  }
  loadData();
```

```
    setShowModal(false);
    setEditing(null);
  } catch (err: any) {
    console.error("Failed to save", err);
    const errorMessage = err?.response?.data?.detail || err?.message || "Unknown error";
    // Handle array of validation errors
    if (Array.isArray(errorMessage)) {
      const validationErrors = errorMessage.map((e: any) =&gt; `${e.loc?.join('.')}: ${e.msg}`).join(', ')
      alert("Save failed: " + validationErrors);
    } else {
      alert("Save failed: " + errorMessage);
    }
  }
};

const apiUrl = import.meta.env.VITE_API_URL ?? "http://localhost:8000";

return (
  &lt;div className="space-y-6"&gt;
    &lt;h1 className="text-3xl font-bold text-primary"&gt;{t("content_management") || "Content Management"

    {/* Tabs */}
    &lt;div className="flex gap-2 border-b border-slate-200 overflow-x-auto"&gt;
      {(["rooms", "foods", "drinks", "events", "promos"] as const).map((tab) =&gt; (
        &lt;button
          key={tab}
          onClick={() =&gt; setActiveTab(tab)}
          className={`px-3 sm:px-4 py-2 font-semibold border-b-2 transition-colors whitespace-nowrap ${
            activeTab === tab
              ? "border-primary text-primary"
              : "border-transparent text-slate-600 hover:text-primary"
          }`}
        &gt;
          {t(`content_tab_${tab}`) || tab.charAt(0).toUpperCase() + tab.slice(1)}
        &lt;/button&gt;
      ))}
    &lt;/div&gt;

    {/* Content */}
    &lt;div className="space-y-4"&gt;
      &lt;div className="flex flex-col sm:flex-row justify-between items-start sm:items-center gap-3"&gt;
        &lt;h2 className="text-xl font-semibold"&gt;
          {t(`content_tab_${activeTab}`) || activeTab.charAt(0).toUpperCase() + activeTab.slice(1)}
        &lt;/h2&gt;
        &lt;motion.button
          whileHover={{ scale: 1.05 }}
          whileTap={{ scale: 0.95 }}
          onClick={() =&gt; {
            setEditing(null);
            setShowModal(true);
          }}
          className="rounded-lg bg-primary px-4 py-2 text-white font-semibold w-full sm:w-auto"
        &gt;
          {t("add") || "Add New"}
        &lt;/motion.button&gt;
      &lt;/div&gt;

      {/* Rooms */}
      {activeTab === "rooms" &amp;&amp; (
        &lt;div className="space-y-6"&gt;
          &lt;div&gt;
            &lt;h3 className="text-lg font-semibold mb-4"&gt;Rooms&lt;/h3&gt;
            &lt;div className="grid gap-4 grid-cols-1 sm:grid-cols-2 lg:grid-cols-3"&gt;
              {rooms.map((room) =&gt; (
                &lt;motion.div
                  key={room.id}
                  whileHover={{ scale: 1.02, y: -5 }}
                  className="rounded-xl bg-white p-4 shadow-lg border border-slate-200"
                &gt;
                  {room.image_url &amp;&amp; (
                    &lt;img
                      src={`${apiUrl}${room.image_url}`}
                      alt={room.title}
```

```
              className="w-full h-48 object-cover rounded-lg mb-2"
            /&gt;
          )}
          &lt;p className="font-semibold text-lg"&gt;{room.title}&lt;/p&gt;
          &lt;p className="text-sm text-slate-600"&gt;Room {room.room_number}&lt;/p&gt;
          {room.location &amp;&amp; &lt;p className="text-sm text-slate-500"&gt;■ {room.location}&
          &lt;p className="text-sm text-slate-500"&gt;{room.wifi ? "■ WiFi" : "■ No WiFi"}&lt;/p&
          &lt;p className="text-lg font-bold text-primary mt-2"&gt;■{room.price.toFixed(2)}&lt;/p&
          {room.features &amp;&amp; room.features.length &gt; 0 &amp;&amp; (
            &lt;div className="mt-2"&gt;
              &lt;p className="text-xs text-slate-500"&gt;Features:&lt;/p&gt;
              &lt;div className="flex flex-wrap gap-1 mt-1"&gt;
                {room.features.map((feature, idx) =&gt; (
                  &lt;span key={idx} className="text-xs bg-slate-100 px-2 py-1 rounded"&gt;
                    {feature}
                  &lt;/span&gt;
                ))}
              &lt;/div&gt;
            &lt;/div&gt;
          )}
          &lt;span className={`text-xs px-2 py-1 rounded mt-2 inline-block ${
            room.status === "available" ? "bg-green-100 text-green-700" : "bg-red-100 text-red-700
          }`}&gt;
            {room.status}
          &lt;/span&gt;
          &lt;div className="flex gap-2 mt-3"&gt;
            &lt;button
              onClick={() =&gt; {
                setEditing(room);
                setShowModal(true);
              }}
              className="flex-1 rounded-lg bg-accent px-3 py-2 text-white text-sm font-semibold"
            &gt;
              Edit
            &lt;/button&gt;
            &lt;button
              onClick={async () =&gt; {
                if (confirm("Are you sure you want to delete this room?")) {
                  try {
                    await api.delete(`/content/rooms/${room.id}`);
                    loadData();
                  } catch (err) {
                    console.error("Failed to delete room", err);
                    alert("Failed to delete room");
                  }
                }
              }}
              className="flex-1 rounded-lg bg-danger px-3 py-2 text-white text-sm font-semibold"
            &gt;
              Delete
            &lt;/button&gt;
          &lt;/div&gt;
        &lt;/motion.div&gt;
      ))}
    &lt;/div&gt;
  &lt;/div&gt;
  &lt;div&gt;
    &lt;h3 className="text-lg font-semibold mb-4"&gt;Room Images&lt;/h3&gt;
    &lt;div className="grid gap-4 grid-cols-1 sm:grid-cols-2 lg:grid-cols-3"&gt;
      {roomImages.map((img) =&gt; (
        &lt;motion.div
          key={img.id}
          whileHover={{ scale: 1.02, y: -5 }}
          className="rounded-xl bg-white p-4 shadow-lg border border-slate-200 relative"
        &gt;
          &lt;img
            src={`${apiUrl}${img.image_url}`}
            alt={img.description}
            className="w-full h-48 object-cover rounded-lg mb-2"
          /&gt;
          &lt;p className="font-semibold"&gt;Room {img.room_number}&lt;/p&gt;
          &lt;p className="text-sm text-slate-600"&gt;{img.description}&lt;/p&gt;
          &lt;button
```

```
                  onClick={async () =&gt; {
                    if (confirm("Are you sure you want to delete this image?")) {
                      try {
                        await api.delete(`/content/rooms/images/${img.id}`);
                        loadData();
                      } catch (err) {
                        console.error("Failed to delete image", err);
                        alert("Failed to delete image");
                      }
                    }}
                    className="mt-2 w-full rounded-lg bg-danger px-3 py-2 text-white text-sm font-semibold
                  &gt;
                    Delete
                  &lt;/button&gt;
                &lt;/motion.div&gt;
              ))}
            &lt;/div&gt;
          &lt;/div&gt;
        &lt;/div&gt;
      )}

      {/* Foods */}
      {activeTab === "foods" &amp;&amp; (
        &lt;div className="grid gap-4 grid-cols-1 sm:grid-cols-2 lg:grid-cols-3"&gt;
          {foods.map((food) =&gt; (
            &lt;motion.div
              key={food.id}
              whileHover={{ scale: 1.02, y: -5 }}
              className="rounded-xl bg-white p-4 shadow-lg border border-slate-200"
            &gt;
              {food.image_url &amp;&amp; (
                &lt;img
                  src={`${apiUrl}${food.image_url}`}
                  alt={food.name}
                  className="w-full h-48 object-cover rounded-lg mb-2"
                /&gt;
              )}
              &lt;p className="font-semibold"&gt;{food.name}&lt;/p&gt;
              &lt;p className="text-sm text-slate-600"&gt;{food.description}&lt;/p&gt;
              &lt;p className="text-lg font-bold text-primary mt-2"&gt;■{food.price}&lt;/p&gt;
              &lt;div className="flex flex-wrap gap-2 mt-2"&gt;
                {food.is_tea &amp;&amp; &lt;span className="text-xs bg-amber-100 text-amber-700 px-2 py-1
■ Tea&lt;/span&gt;}
                {food.contains_alcohol &amp;&amp; &lt;span className="text-xs bg-red-100 text-red-700 px-2
d"&gt;■ Alcohol&lt;/span&gt;}
                &lt;span className={`text-xs px-2 py-1 rounded ${food.available ? "bg-green-100 text-green
red-100 text-red-700"}`}&gt;
                  {food.available ? "Available" : "Unavailable"}
                &lt;/span&gt;
              &lt;/div&gt;
              &lt;div className="flex gap-2 mt-3"&gt;
                &lt;button
                  onClick={() =&gt; {
                    setEditing(food);
                    setShowModal(true);
                  }}
                  className="flex-1 rounded-lg bg-accent px-3 py-2 text-white text-sm font-semibold"
                &gt;
                  Edit
                &lt;/button&gt;
                &lt;button
                  onClick={async () =&gt; {
                    if (confirm("Are you sure you want to delete this item?")) {
                      try {
                        await api.delete(`/content/foods/${food.id}`);
                        loadData();
                      } catch (err) {
                        console.error("Failed to delete food", err);
                        alert("Failed to delete food");
                      }
                    }}
```

```
                className="flex-1 rounded-lg bg-danger px-3 py-2 text-white text-sm font-semibold"
              &gt;
                Delete
              &lt;/button&gt;
            &lt;/div&gt;
          &lt;/motion.div&gt;
        ))}
      &lt;/div&gt;
    )}

    {/* Drinks */}
    {activeTab === "drinks" &amp;&amp; (
      &lt;div className="grid gap-4 grid-cols-1 sm:grid-cols-2 lg:grid-cols-3"&gt;
        {drinks.map((drink) =&gt; (
          &lt;motion.div
            key={drink.id}
            whileHover={{ scale: 1.02, y: -5 }}
            className="rounded-xl bg-white p-4 shadow-lg border border-slate-200"
          &gt;
            {drink.image_url &amp;&amp; (
              &lt;img
                src={`${apiUrl}${drink.image_url}`}
                alt={drink.name}
                className="w-full h-48 object-cover rounded-lg mb-2"
              /&gt;
            )}
            &lt;p className="font-semibold"&gt;{drink.name}&lt;/p&gt;
            &lt;p className="text-sm text-slate-600"&gt;{drink.description}&lt;/p&gt;
            &lt;p className="text-lg font-bold text-primary mt-2"&gt;■{drink.price}&lt;/p&gt;
            &lt;div className="flex flex-wrap gap-2 mt-2"&gt;
              {drink.is_tea &amp;&amp; &lt;span className="text-xs bg-amber-100 text-amber-700 px-2 py-1
;■ Tea&lt;/span&gt;}
              {drink.contains_alcohol &amp;&amp; &lt;span className="text-xs bg-red-100 text-red-700 px-
ed"&gt;■ Alcohol&lt;/span&gt;}
              &lt;span className={`text-xs px-2 py-1 rounded ${drink.available ? "bg-green-100 text-gree
-red-100 text-red-700"}`}&gt;
                {drink.available ? "Available" : "Unavailable"}
              &lt;/span&gt;
            &lt;/div&gt;
            &lt;div className="flex gap-2 mt-3"&gt;
              &lt;button
                onClick={() =&gt; {
                  setEditing(drink);
                  setShowModal(true);
                }}
                className="flex-1 rounded-lg bg-accent px-3 py-2 text-white text-sm font-semibold"
              &gt;
                Edit
              &lt;/button&gt;
              &lt;button
                onClick={async () =&gt; {
                  if (confirm("Are you sure you want to delete this item?")) {
                    try {
                      await api.delete(`/content/drinks/${drink.id}`);
                      loadData();
                    } catch (err) {
                      console.error("Failed to delete drink", err);
                      alert("Failed to delete drink");
                    }
                  }
                }}
                className="flex-1 rounded-lg bg-danger px-3 py-2 text-white text-sm font-semibold"
              &gt;
                Delete
              &lt;/button&gt;
            &lt;/div&gt;
          &lt;/motion.div&gt;
        ))}
      &lt;/div&gt;
    )}

    {/* Events */}
    {activeTab === "events" &amp;&amp; (
```

```
      &lt;div className="grid gap-4 grid-cols-1 sm:grid-cols-2"&gt;
        {events.map((event) =&gt; (
          &lt;motion.div
            key={event.id}
            whileHover={{ scale: 1.02, y: -5 }}
            className="rounded-xl bg-white p-4 shadow-lg border border-slate-200"
          &gt;
            {event.image_url &amp;&amp; (
              &lt;img
                src={`${apiUrl}${event.image_url}`}
                alt={event.title}
                className="w-full h-48 object-cover rounded-lg mb-2"
              /&gt;
            )}
            &lt;p className="font-semibold text-lg"&gt;{event.title}&lt;/p&gt;
            &lt;p className="text-sm text-slate-600"&gt;{event.description}&lt;/p&gt;
            &lt;p className="text-xs text-slate-500 mt-2"&gt;
              {new Date(event.event_date).toLocaleString()} · {event.location}
            &lt;/p&gt;
          &lt;/motion.div&gt;
        ))}
      &lt;/div&gt;
    )}

    {/* Promos */}
    {activeTab === "promos" &amp;&amp; (
      &lt;div className="grid gap-4 grid-cols-1 sm:grid-cols-2"&gt;
        {promos.map((promo) =&gt; (
          &lt;motion.div
            key={promo.id}
            whileHover={{ scale: 1.02, y: -5 }}
            className="rounded-xl bg-white p-4 shadow-lg border border-slate-200"
          &gt;
            {promo.image_url &amp;&amp; (
              &lt;img
                src={`${apiUrl}${promo.image_url}`}
                alt={promo.title}
                className="w-full h-48 object-cover rounded-lg mb-2"
              /&gt;
            )}
            &lt;p className="font-semibold text-lg"&gt;{promo.title}&lt;/p&gt;
            &lt;p className="text-sm text-slate-600"&gt;{promo.description}&lt;/p&gt;
            {promo.code &amp;&amp; &lt;p className="text-sm font-semibold text-accent mt-2"&gt;Code: {pr
;/p&gt;}
            {(promo.discount_percent || promo.discount_amount) &amp;&amp; (
              &lt;p className="text-lg font-bold text-primary mt-2"&gt;
                {promo.discount_percent ? `${promo.discount_percent}% OFF` : `■${promo.discount_amount}
              &lt;/p&gt;
            )}
          &lt;/motion.div&gt;
        ))}
      &lt;/div&gt;
    )}
  &lt;/div&gt;

  {/* Modal for adding/editing */}
  {showModal &amp;&amp; (
    &lt;div className="fixed inset-0 bg-black/50 flex items-center justify-center z-50 p-4"&gt;
      &lt;motion.div
        initial={{ opacity: 0, scale: 0.9 }}
        animate={{ opacity: 1, scale: 1 }}
        className="bg-white rounded-2xl p-4 sm:p-6 max-w-md w-full max-h-[90vh] overflow-y-auto"
      &gt;
        &lt;h3 className="text-xl font-bold mb-4"&gt;
          {editing ? t("edit") || "Edit" : t("add") || "Add New"}
        &lt;/h3&gt;
        &lt;form
          onSubmit={(e) =&gt; {
            e.preventDefault();
            const form = e.target as HTMLFormElement;
            const formData = new FormData(form);
            const fileInput = form.querySelector('input[type="file"]') as HTMLInputElement;
            const file = fileInput?.files?.[0];
```

```
    const data: any = {};

    // Get all form values including checkboxes
    const wifiCheckbox = form.querySelector&lt;HTMLInputElement&gt;('[name="wifi"]');
    const isTeaCheckbox = form.querySelector&lt;HTMLInputElement&gt;('[name="is_tea"]');
    const containsAlcoholCheckbox = form.querySelector&lt;HTMLInputElement&gt;('[name="contains_

    formData.forEach((value, key) =&gt; {
      if (key !== "file") {
        data[key] = value;
      }
    });

    // Set checkbox values explicitly
    if (wifiCheckbox) data.wifi = wifiCheckbox.checked;
    if (isTeaCheckbox) data.is_tea = isTeaCheckbox.checked;
    if (containsAlcoholCheckbox) data.contains_alcohol = containsAlcoholCheckbox.checked;

    // Handle submission
    const isEditing = !!editing;
    handleSubmit(file || null, activeTab, data, isEditing);
  }}
  className="space-y-4"
&gt;
  {activeTab === "rooms" &amp;&amp; (
    &lt;&gt;
      &lt;input
        type="text"
        name="room_number"
        placeholder={t("room_number") || "Room Number"}
        required
        defaultValue={editing?.room_number || ""}
        className="w-full rounded-xl border border-slate-200 px-4 py-2"
      /&gt;
      &lt;input
        type="text"
        name="title"
        placeholder={t("title") || "Room Title"}
        required
        defaultValue={editing?.title || ""}
        className="w-full rounded-xl border border-slate-200 px-4 py-2"
      /&gt;
      &lt;textarea
        name="description"
        placeholder={t("description") || "Description"}
        defaultValue={editing?.description || ""}
        className="w-full rounded-xl border border-slate-200 px-4 py-2"
      /&gt;
      &lt;input
        type="number"
        name="price"
        placeholder={t("price") || "Price"}
        step="0.01"
        required
        defaultValue={editing?.price || ""}
        className="w-full rounded-xl border border-slate-200 px-4 py-2"
      /&gt;
      &lt;input
        type="text"
        name="location"
        placeholder={t("location") || "Location (e.g., Floor 2, Building A)"}
        defaultValue={editing?.location || ""}
        className="w-full rounded-xl border border-slate-200 px-4 py-2"
      /&gt;
      &lt;div className="flex items-center gap-2"&gt;
        &lt;input
          type="checkbox"
          name="wifi"
          id="wifi"
          defaultChecked={editing?.wifi !== false}
          className="w-4 h-4"
        /&gt;
        &lt;label htmlFor="wifi" className="text-sm"&gt;WiFi Available&lt;/label&gt;
```

```
              &lt;/div&gt;
              &lt;input
                type="text"
                name="features"
                placeholder={t("features") || "Features (comma-separated, e.g., TV, AC, Balcony)"}
                defaultValue={editing?.features ? editing.features.join(", ") : ""}
                className="w-full rounded-xl border border-slate-200 px-4 py-2"
              /&gt;
              &lt;select
                name="status"
                className="w-full rounded-xl border border-slate-200 px-4 py-2"
                defaultValue={editing?.status || "available"}
              &gt;
                &lt;option value="available"&gt;Available&lt;/option&gt;
                &lt;option value="occupied"&gt;Occupied&lt;/option&gt;
                &lt;option value="maintenance"&gt;Maintenance&lt;/option&gt;
              &lt;/select&gt;
              {editing?.image_url &amp;&amp; (
                &lt;div className="text-sm text-slate-600"&gt;
                  Current image: &lt;img src={`${apiUrl}${editing.image_url}`} alt="Current" className="
ject-cover rounded mt-1" /&gt;
                &lt;/div&gt;
              )}
              &lt;input
                type="file"
                name="file"
                accept="image/*"
                className="w-full rounded-xl border border-slate-200 px-4 py-2"
              /&gt;
              {editing &amp;&amp; (
                &lt;&gt;
                  &lt;input type="hidden" name="id" value={editing.id} /&gt;
                  &lt;div className="mt-4 p-4 bg-slate-50 rounded-xl"&gt;
                    &lt;p className="text-sm font-semibold mb-2"&gt;Upload Additional Images (up to 5 to
t;
                    &lt;input
                      type="file"
                      id="additional-images"
                      accept="image/*"
                      multiple
                      className="w-full rounded-xl border border-slate-200 px-4 py-2"
                      onChange={async (e) =&gt; {
                        const files = Array.from(e.target.files || []);
                        if (files.length &gt; 5) {
                          alert("Maximum 5 images allowed");
                          return;
                        }
                        if (editing?.id) {
                          try {
                            const formData = new FormData();
                            files.forEach((file) =&gt; {
                              formData.append("files", file);
                            });
                            await api.post(`/content/rooms/${editing.id}/images`, formData, {
                              headers: { "Content-Type": "multipart/form-data" },
                            });
                            alert("Images uploaded successfully!");
                            loadData();
                            e.target.value = ""; // Reset input
                          } catch (err) {
                            console.error("Failed to upload images", err);
                            alert("Failed to upload images");
                          }
                        }
                      }}
                    /&gt;
                    &lt;p className="text-xs text-slate-500 mt-1"&gt;You can upload up to 5 images per r
;
                  &lt;/div&gt;
                &lt;/&gt;
              )}
            &lt;/&gt;
          )}
```

```
            {(activeTab === "foods" || activeTab === "drinks") &amp;&amp; (
              &lt;&gt;
                &lt;input
                  type="text"
                  name="name"
                  placeholder={t("name") || "Name"}
                  required
                  defaultValue={editing?.name || ""}
                  className="w-full rounded-xl border border-slate-200 px-4 py-2"
                /&gt;
                &lt;textarea
                  name="description"
                  placeholder={t("description") || "Description"}
                  defaultValue={editing?.description || ""}
                  className="w-full rounded-xl border border-slate-200 px-4 py-2"
                /&gt;
                &lt;input
                  type="number"
                  name="price"
                  placeholder={t("price")}
                  step="0.01"
                  required
                  defaultValue={editing?.price || ""}
                  className="w-full rounded-xl border border-slate-200 px-4 py-2"
                /&gt;
                &lt;div className="flex items-center gap-2"&gt;
                  &lt;input
                    type="checkbox"
                    name="is_tea"
                    id="is_tea"
                    defaultChecked={editing?.is_tea || false}
                    className="w-4 h-4"
                  /&gt;
                  &lt;label htmlFor="is_tea" className="text-sm"&gt;Is Tea&lt;/label&gt;
                &lt;/div&gt;
                &lt;div className="flex items-center gap-2"&gt;
                  &lt;input
                    type="checkbox"
                    name="contains_alcohol"
                    id="contains_alcohol"
                    defaultChecked={editing?.contains_alcohol || false}
                    className="w-4 h-4"
                  /&gt;
                  &lt;label htmlFor="contains_alcohol" className="text-sm"&gt;Contains Alcohol&lt;/label&g
                &lt;/div&gt;
                {editing?.image_url &amp;&amp; (
                  &lt;div className="text-sm text-slate-600"&gt;
                    Current image: &lt;img src={`${apiUrl}${editing.image_url}`} alt="Current" className="
ject-cover rounded mt-1" /&gt;
                  &lt;/div&gt;
                )}
                &lt;input
                  type="file"
                  name="file"
                  accept="image/*"
                  className="w-full rounded-xl border border-slate-200 px-4 py-2"
                /&gt;
                {editing &amp;&amp; &lt;input type="hidden" name="id" value={editing.id} /&gt;}
              &lt;/&gt;
            )}
            {activeTab === "events" &amp;&amp; (
              &lt;&gt;
                &lt;input
                  type="text"
                  name="title"
                  placeholder={t("title") || "Title"}
                  required
                  className="w-full rounded-xl border border-slate-200 px-4 py-2"
                /&gt;
                &lt;textarea
                  name="description"
                  placeholder={t("description") || "Description"}
                  className="w-full rounded-xl border border-slate-200 px-4 py-2"
```

```
    /&gt;
    &lt;input
      type="datetime-local"
      name="event_date"
      required
      className="w-full rounded-xl border border-slate-200 px-4 py-2"
    /&gt;
    &lt;input
      type="text"
      name="location"
      placeholder={t("location") || "Location"}
      className="w-full rounded-xl border border-slate-200 px-4 py-2"
    /&gt;
    &lt;input
      type="file"
      name="file"
      accept="image/*"
      className="w-full rounded-xl border border-slate-200 px-4 py-2"
    /&gt;
  &lt;/&gt;
)}
{activeTab === "promos" &amp;&amp; (
  &lt;&gt;
    &lt;input
      type="text"
      name="title"
      placeholder={t("title") || "Title"}
      required
      className="w-full rounded-xl border border-slate-200 px-4 py-2"
    /&gt;
    &lt;textarea
      name="description"
      placeholder={t("description") || "Description"}
      className="w-full rounded-xl border border-slate-200 px-4 py-2"
    /&gt;
    &lt;input
      type="text"
      name="code"
      placeholder={t("promo_code") || "Promo Code"}
      className="w-full rounded-xl border border-slate-200 px-4 py-2"
    /&gt;
    &lt;input
      type="number"
      name="discount_percent"
      placeholder={t("discount_percent") || "Discount %"}
      className="w-full rounded-xl border border-slate-200 px-4 py-2"
    /&gt;
    &lt;input
      type="datetime-local"
      name="valid_from"
      required
      className="w-full rounded-xl border border-slate-200 px-4 py-2"
    /&gt;
    &lt;input
      type="datetime-local"
      name="valid_until"
      required
      className="w-full rounded-xl border border-slate-200 px-4 py-2"
    /&gt;
    &lt;input
      type="file"
      name="file"
      accept="image/*"
      className="w-full rounded-xl border border-slate-200 px-4 py-2"
    /&gt;
  &lt;/&gt;
)}
&lt;div className="flex flex-col sm:flex-row gap-2"&gt;
  &lt;motion.button
    whileHover={{ scale: 1.05 }}
    whileTap={{ scale: 0.95 }}
    type="submit"
    className="flex-1 rounded-xl bg-primary px-4 py-2 text-white font-semibold"
```

```
                &gt;
                  {t("save") || "Save"}
                &lt;/motion.button&gt;
                &lt;motion.button
                  whileHover={{ scale: 1.05 }}
                  whileTap={{ scale: 0.95 }}
                  type="button"
                  onClick={() =&gt; {
                    setShowModal(false);
                    setEditing(null);
                  }}
                  className="flex-1 rounded-xl border border-slate-200 px-4 py-2 font-semibold"
                &gt;
                  {t("cancel") || "Cancel"}
                &lt;/motion.button&gt;
              &lt;/div&gt;
            &lt;/form&gt;
          &lt;/motion.div&gt;
        &lt;/div&gt;
      )}
    &lt;/div&gt;
  );
};

export default ContentManagementPage;
```

File: frontend\src\pages\CustomerPage.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\pages\CustomerPage.tsx

```
// @ts-nocheck
import { useEffect, useState } from "react";
import { motion, AnimatePresence } from "framer-motion";
import { useTranslation } from "react-i18next";
import { Link } from "react-router-dom";

import Navbar from "../components/Navbar";
import RoomCard from "../components/RoomCard";
import { api } from "../services/api";

type CustomerProfile = {
  email: string;
  room_number: string;
  floor: number;
  preferences: {
    food: string[];
    drinks: string[];
  };
};

type CustomerOrder = {
  id: number;
  item_name: string;
  category: string;
  quantity: number;
  status: string;
  created_at: string;
  customer_email: string;
};

type Booking = {
  booking_id: number;
  room_id: number;
  check_in: string;
  check_out: string;
  booking_status: string;
  payment_method: string;
  payment_status: string;
  total_amount: number;
  booking_created_at: string;
  guest_name: string;
  guest_passport: string;
  guest_phone: string;
  room_number: string;
  room_title: string;
  room_price: number;
  room_location: string | null;
};

type Event = {
  id: number;
  title: string;
  description: string;
  event_date: string;
  location: string;
  image_url: string;
  active: boolean;
};

type Promo = {
  id: number;
  title: string;
  description: string;
  discount_percent: number;
  discount_amount: number;
  code: string;
  valid_from: string;
  valid_until: string;
```

```
  image_url: string;
  active: boolean;
};

const CustomerPage = () =&gt; {
  const { t } = useTranslation();
  const [profile, setProfile] = useState&lt;CustomerProfile | null&gt;(null);
  const [orders, setOrders] = useState&lt;CustomerOrder[]&gt;([]);
  const [bookings, setBookings] = useState&lt;Booking[]&gt;([]);
  const [events, setEvents] = useState&lt;Event[]&gt;([]);
  const [promos, setPromos] = useState&lt;Promo[]&gt;([]);
  const [loading, setLoading] = useState(true);
  const [bookingLookup, setBookingLookup] = useState({
    passport_number: "",
    phone_number: "",
  });
  const [showBookingLookup, setShowBookingLookup] = useState(false);

  useEffect(() =&gt; {
    loadData();
  }, []);

  const loadData = async () =&gt; {
    setLoading(true);
    try {
      const [profileRes, ordersRes, eventsRes, promosRes] = await Promise.all([
        api.get&lt;CustomerProfile&gt;("/customer/profile").catch(() =&gt; null),
        api.get&lt;CustomerOrder[]&gt;("/orders/my").catch(() =&gt; []),
        api.get&lt;Event[]&gt;("/content/events?active_only=true").catch(() =&gt; []),
        api.get&lt;Promo[]&gt;("/content/promos?active_only=true").catch(() =&gt; []),
      ]);

      if (profileRes) setProfile(profileRes.data);
      if (ordersRes) setOrders(ordersRes.data);
      if (eventsRes) setEvents(eventsRes.data);
      if (promosRes) setPromos(promosRes.data);
    } catch (err) {
      console.error("Failed to load data", err);
    } finally {
      setLoading(false);
    }
  };

  const loadBookings = async () =&gt; {
    if (!bookingLookup.passport_number &amp;amp;&amp;amp; !bookingLookup.phone_number) {
      alert("Please enter either passport number or phone number");
      return;
    }
    try {
      const params: any = {};
      if (bookingLookup.passport_number) {
        params.passport_number = bookingLookup.passport_number;
      }
      if (bookingLookup.phone_number) {
        params.phone_number = bookingLookup.phone_number;
      }
      const res = await api.get&lt;Booking[]&gt;("/bookings/my", { params });
      setBookings(res.data);
      setShowBookingLookup(false);
    } catch (err: any) {
      console.error("Failed to load bookings", err);
      const errorMsg = err.response?.data?.detail || "Failed to load bookings";
      alert(errorMsg);
    }
  };


  const cancelOrder = async (orderId: number) =&gt; {
    if (!confirm("Are you sure you want to cancel this order?")) return;
    try {
      await api.post(`/orders/${orderId}/cancel`);
      setOrders((prev) =&gt; prev.map(o =&gt; o.id === orderId ? { ...o, status: "cancelled" } : o));
    } catch (err) {
```

```
    console.error("Failed to cancel order", err);
    alert("Failed to cancel order");
  }
};

const getStatusColor = (status: string) =&gt; {
  switch (status.toLowerCase()) {
    case "pending": return "bg-yellow-100 text-yellow-700 border-yellow-300";
    case "in_progress": return "bg-blue-100 text-blue-700 border-blue-300";
    case "served": return "bg-green-100 text-green-700 border-green-300";
    case "cancelled": return "bg-red-100 text-red-700 border-red-300";
    default: return "bg-slate-100 text-slate-700 border-slate-300";
  }
};

const apiUrl = import.meta.env.VITE_API_URL ?? "http://localhost:8000";

if (loading) {
  return (
    &lt;div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-purple-50 flex items-ce
  -center"&gt;
      &lt;motion.div
        animate={{ rotate: 360 }}
        transition={{ duration: 1, repeat: Infinity, ease: "linear" }}
        className="w-16 h-16 border-4 border-primary border-t-transparent rounded-full"
      /&gt;
    &lt;/div&gt;
  );
}

return (
  &lt;div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-purple-50"&gt;
    &lt;Navbar /&gt;
    &lt;main className="mx-auto flex max-w-7xl flex-col gap-6 sm:gap-10 px-4 sm:px-6 pb-10 sm:pb-20 pt-6 s
      {/* Hero Section */}
      &lt;motion.section
        initial={{ opacity: 0, y: -20 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ duration: 0.6 }}
        className="relative overflow-hidden rounded-3xl bg-gradient-to-r from-primary via-blue-700 to-acce
  e p-6 sm:p-10 shadow-2xl"
      &gt;
        &lt;div
          className="absolute inset-0 opacity-20"
          style={{
            backgroundImage: `url("data:image/svg+xml,%3Csvg width='60' height='60' viewBox='0 0 60 60' xm
  www.w3.org/2000/svg'%3E%3Cg fill='none' fill-rule='evenodd'%3E%3Cg fill='%23ffffff' fill-opacity='0.05'%
  'M36 34v-4h-2v4h-4v2h4v4h2v-4h4v-2h-4zm0-30V0h-2v4h-4v2h4v4h2V6h4V4h-4zM6 34v-4H4v4H0v2h4v4h2v-4h4v-2H6z
  v2h4v4h2V6h4V4H6z'/%3E%3C/g%3E%3C/g%3E%3C/svg%3E")`
          }}
        &gt;&lt;/div&gt;
        &lt;div className="relative z-10"&gt;
          &lt;motion.h1
            initial={{ opacity: 0, x: -20 }}
            animate={{ opacity: 1, x: 0 }}
            transition={{ delay: 0.2 }}
            className="text-3xl sm:text-4xl lg:text-5xl font-bold mb-3"
          &gt;
            {t("customer_portal_title") || "Welcome to Your Portal"}
          &lt;/motion.h1&gt;
          &lt;motion.p
            initial={{ opacity: 0, x: -20 }}
            animate={{ opacity: 1, x: 0 }}
            transition={{ delay: 0.3 }}
            className="text-white/90 text-sm sm:text-base max-w-2xl mb-6"
          &gt;
            {t("customer_portal_subtitle") || "Manage your stay, orders, and more"}
          &lt;/motion.p&gt;
          {profile &amp;&amp; (
            &lt;motion.div
              initial={{ opacity: 0, y: 10 }}
              animate={{ opacity: 1, y: 0 }}
              transition={{ delay: 0.4 }}
```

```
            className="flex flex-wrap gap-4 text-sm sm:text-base"
          &gt;
            &lt;div className="bg-white/10 backdrop-blur-sm rounded-xl px-4 py-2"&gt;
              &lt;span className="font-semibold"&gt;■ {t("email")}:&lt;/span&gt; {profile.email}
            &lt;/div&gt;
            &lt;div className="bg-white/10 backdrop-blur-sm rounded-xl px-4 py-2"&gt;
              &lt;span className="font-semibold"&gt;■ {t("room_number")}:&lt;/span&gt; {profile.room_num
            &lt;/div&gt;
            &lt;div className="bg-white/10 backdrop-blur-sm rounded-xl px-4 py-2"&gt;
              &lt;span className="font-semibold"&gt;■ {t("customer_floor_label") || "Floor"}:&lt;/span&g
floor}
            &lt;/div&gt;
          &lt;/motion.div&gt;
        )}
      &lt;/div&gt;
    &lt;/motion.section&gt;

    {/* Quick Actions Grid */}
    &lt;motion.section
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ delay: 0.2, duration: 0.5 }}
      className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-4 gap-4 sm:gap-6"
    &gt;
      &lt;motion.div
        whileHover={{ scale: 1.05, y: -5 }}
        whileTap={{ scale: 0.95 }}
        onClick={() =&gt; window.location.href = "/"}
        className="rounded-2xl bg-white/90 backdrop-blur-md p-6 shadow-xl border border-white/20 cursor-
      &gt;
        &lt;div className="text-4xl mb-3"&gt;■&lt;/div&gt;
        &lt;h3 className="text-lg font-semibold text-primary mb-2"&gt;Book a Room&lt;/h3&gt;
        &lt;p className="text-sm text-slate-600"&gt;Browse available rooms&lt;/p&gt;
      &lt;/motion.div&gt;
      &lt;motion.div
        whileHover={{ scale: 1.05, y: -5 }}
        whileTap={{ scale: 0.95 }}
        onClick={() =&gt; setShowBookingLookup(true)}
        className="rounded-2xl bg-white/90 backdrop-blur-md p-6 shadow-xl border border-white/20 cursor-
      &gt;
        &lt;div className="text-4xl mb-3"&gt;■&lt;/div&gt;
        &lt;h3 className="text-lg font-semibold text-primary mb-2"&gt;My Bookings&lt;/h3&gt;
        &lt;p className="text-sm text-slate-600"&gt;View your reservations&lt;/p&gt;
      &lt;/motion.div&gt;
      &lt;motion.div
        whileHover={{ scale: 1.05, y: -5 }}
        whileTap={{ scale: 0.95 }}
        className="rounded-2xl bg-white/90 backdrop-blur-md p-6 shadow-xl border border-white/20"
      &gt;
        &lt;div className="text-4xl mb-3"&gt;■■&lt;/div&gt;
        &lt;h3 className="text-lg font-semibold text-primary mb-2"&gt;Order Food &amp;amp; Drinks&lt;/h3&gt;
        &lt;p className="text-sm text-slate-600"&gt;Place your order quickly&lt;/p&gt;
      &lt;/motion.div&gt;
      &lt;motion.div
        whileHover={{ scale: 1.05, y: -5 }}
        whileTap={{ scale: 0.95 }}
        className="rounded-2xl bg-white/90 backdrop-blur-md p-6 shadow-xl border border-white/20"
      &gt;
        &lt;div className="text-4xl mb-3"&gt;■&lt;/div&gt;
        &lt;h3 className="text-lg font-semibold text-primary mb-2"&gt;Promotions&lt;/h3&gt;
        &lt;p className="text-sm text-slate-600"&gt;{promos.length} active promos&lt;/p&gt;
      &lt;/motion.div&gt;
    &lt;/motion.section&gt;

    {/* Security Settings Section */}
    &lt;motion.section
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ delay: 0.3, duration: 0.5 }}
      className="mt-8"
    &gt;
      &lt;h2 className="text-2xl font-bold text-primary mb-4"&gt;{t("security")}&lt;/h2&gt;
      &lt;div className="grid grid-cols-1 sm:grid-cols-2 gap-4"&gt;
```

```
    &lt;Link to="/password-change"&gt;
      &lt;motion.div
        whileHover={{ scale: 1.02, y: -2 }}
        whileTap={{ scale: 0.98 }}
        className="rounded-2xl bg-white/90 backdrop-blur-md p-6 shadow-xl border border-white/20 cur
      &gt;
        &lt;div className="text-3xl mb-3"&gt;■&lt;/div&gt;
        &lt;h3 className="text-lg font-semibold text-primary mb-2"&gt;{t("change_password")}&lt;/h3&
        &lt;p className="text-sm text-slate-600"&gt;{t("change_password_description")}&lt;/p&gt;
      &lt;/motion.div&gt;
    &lt;/Link&gt;
    &lt;Link to="/mfa"&gt;
      &lt;motion.div
        whileHover={{ scale: 1.02, y: -2 }}
        whileTap={{ scale: 0.98 }}
        className="rounded-2xl bg-white/90 backdrop-blur-md p-6 shadow-xl border border-white/20 cur
      &gt;
        &lt;div className="text-3xl mb-3"&gt;■■&lt;/div&gt;
        &lt;h3 className="text-lg font-semibold text-primary mb-2"&gt;{t("mfa_settings")}&lt;/h3&gt;
        &lt;p className="text-sm text-slate-600"&gt;{t("mfa_description")}&lt;/p&gt;
      &lt;/motion.div&gt;
    &lt;/Link&gt;
    &lt;Link to="/sessions"&gt;
      &lt;motion.div
        whileHover={{ scale: 1.02, y: -2 }}
        whileTap={{ scale: 0.98 }}
        className="rounded-2xl bg-white/90 backdrop-blur-md p-6 shadow-xl border border-white/20 cur
      &gt;
        &lt;div className="text-3xl mb-3"&gt;■&lt;/div&gt;
        &lt;h3 className="text-lg font-semibold text-primary mb-2"&gt;{t("active_sessions")}&lt;/h3&
        &lt;p className="text-sm text-slate-600"&gt;{t("manage_active_sessions")}&lt;/p&gt;
      &lt;/motion.div&gt;
    &lt;/Link&gt;
  &lt;/div&gt;
&lt;/motion.section&gt;

{/* Booking Lookup Modal */}
{showBookingLookup &amp;&amp; (
  &lt;motion.div
    initial={{ opacity: 0 }}
    animate={{ opacity: 1 }}
    className="fixed inset-0 bg-black/50 flex items-center justify-center z-50 p-4"
    onClick={() =&gt; setShowBookingLookup(false)}
  &gt;
    &lt;motion.div
      initial={{ scale: 0.9, opacity: 0 }}
      animate={{ scale: 1, opacity: 1 }}
      onClick={(e) =&gt; e.stopPropagation()}
      className="bg-white rounded-2xl p-6 max-w-md w-full"
    &gt;
      &lt;h3 className="text-xl font-bold text-primary mb-4"&gt;View My Bookings&lt;/h3&gt;
      &lt;p className="text-sm text-slate-600 mb-4"&gt;
        Enter your passport number or phone number to view your bookings
      &lt;/p&gt;
      &lt;div className="space-y-4"&gt;
        &lt;div&gt;
          &lt;label className="block text-sm font-semibold text-slate-700 mb-2"&gt;
            Passport Number
          &lt;/label&gt;
          &lt;input
            type="text"
            value={bookingLookup.passport_number}
            onChange={(e) =&gt; setBookingLookup({ ...bookingLookup, passport_number: e.target.value
            className="w-full rounded-xl border border-slate-200 px-4 py-2"
            placeholder="Enter passport number"
          /&gt;
        &lt;/div&gt;
        &lt;div&gt;
          &lt;label className="block text-sm font-semibold text-slate-700 mb-2"&gt;
            Phone Number
          &lt;/label&gt;
          &lt;input
            type="tel"
```

```
            value={bookingLookup.phone_number}
            onChange={(e) =&gt; setBookingLookup({ ...bookingLookup, phone_number: e.target.value })}
            className="w-full rounded-xl border border-slate-200 px-4 py-2"
            placeholder="Enter phone number"
          /&gt;
        &lt;/div&gt;
        &lt;div className="flex gap-2"&gt;
          &lt;button
            onClick={loadBookings}
            className="flex-1 rounded-xl bg-primary px-4 py-2 text-white font-semibold"
          &gt;
            View Bookings
          &lt;/button&gt;
          &lt;button
            onClick={() =&gt; setShowBookingLookup(false)}
            className="flex-1 rounded-xl border border-slate-200 px-4 py-2 font-semibold"
          &gt;
            Cancel
          &lt;/button&gt;
        &lt;/div&gt;
      &lt;/div&gt;
    &lt;/motion.div&gt;
  &lt;/motion.div&gt;
)}

{/* Bookings Section */}
{bookings.length &gt; 0 &amp;&amp; (
  &lt;motion.section
    initial={{ opacity: 0, y: 20 }}
    animate={{ opacity: 1, y: 0 }}
    transition={{ delay: 0.25, duration: 0.5 }}
    className="rounded-3xl bg-white/90 backdrop-blur-md p-6 sm:p-8 shadow-xl border border-white/20"
  &gt;
    &lt;div className="flex items-center justify-between mb-6"&gt;
      &lt;h2 className="text-2xl font-bold text-primary flex items-center gap-2"&gt;
        &lt;span className="text-3xl"&gt;■&lt;/span&gt;
        {t("my_bookings") || "My Bookings"}
      &lt;/h2&gt;
      &lt;button
        onClick={() =&gt; {
          setBookingLookup({ passport_number: "", phone_number: "" });
          setShowBookingLookup(true);
        }}
        className="text-sm text-primary hover:underline"
      &gt;
        Lookup Another
      &lt;/button&gt;
    &lt;/div&gt;
    &lt;div className="space-y-4"&gt;
      {bookings.map((booking) =&gt; (
        &lt;motion.div
          key={booking.booking_id}
          initial={{ opacity: 0, x: -20 }}
          animate={{ opacity: 1, x: 0 }}
          className="rounded-xl border-2 p-4 bg-gradient-to-r from-white to-slate-50"
        &gt;
          &lt;div className="flex items-start justify-between gap-3"&gt;
            &lt;div className="flex-1"&gt;
              &lt;div className="flex items-center gap-2 mb-2"&gt;
                &lt;h3 className="font-bold text-lg text-slate-800"&gt;{booking.room_title}&lt;/h3&g
                &lt;span className="text-sm text-slate-500"&gt;Room {booking.room_number}&lt;/span&g
              &lt;/div&gt;
              &lt;div className="space-y-1 text-sm text-slate-600 mb-2"&gt;
                &lt;p&gt;■ Check-in: {new Date(booking.check_in).toLocaleDateString()}&lt;/p&gt;
                &lt;p&gt;■ Check-out: {new Date(booking.check_out).toLocaleDateString()}&lt;/p&gt;
                &lt;p&gt;■ Payment: {booking.payment_method} ({booking.payment_status})&lt;/p&gt;
                &lt;p&gt;■ Total: ■{booking.total_amount.toFixed(2)}&lt;/p&gt;
                {booking.room_location &amp;&amp; &lt;p&gt;■ {booking.room_location}&lt;/p&gt;}
              &lt;/div&gt;
              &lt;span className={`inline-block px-3 py-1 rounded-full text-xs font-semibold border
                booking.booking_status === "confirmed" ? "bg-green-100 text-green-700 border-green-3
                booking.booking_status === "pending" ? "bg-yellow-100 text-yellow-700 border-yellow-
                "bg-slate-100 text-slate-700 border-slate-300"
```

```
                  }`}&gt;
                    {booking.booking_status}
                  &lt;/span&gt;
                &lt;/div&gt;
              &lt;/div&gt;
            &lt;/motion.div&gt;
          ))}
        &lt;/div&gt;
      &lt;/motion.section&gt;
    )}

    {/* Order History Section */}
    &lt;motion.section
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ delay: 0.3, duration: 0.5 }}
      className="rounded-3xl bg-white/90 backdrop-blur-md p-6 sm:p-8 shadow-xl border border-white/20"
    &gt;
      &lt;h2 className="text-2xl font-bold text-primary mb-6 flex items-center gap-2"&gt;
        &lt;span className="text-3xl"&gt;■&lt;/span&gt;
        {t("customer_order_history") || "Your Orders"}
      &lt;/h2&gt;
        {orders.length === 0 ? (
          &lt;div className="text-center py-12"&gt;
            &lt;div className="text-6xl mb-4"&gt;■&lt;/div&gt;
            &lt;p className="text-slate-500"&gt;{t("customer_order_empty") || "You have no orders yet."}
          &lt;/div&gt;
        ) : (
          &lt;div className="space-y-3 max-h-[500px] overflow-y-auto"&gt;
            &lt;AnimatePresence&gt;
              {orders.map((order, index) =&gt; (
                &lt;motion.div
                  key={order.id}
                  initial={{ opacity: 0, x: -20 }}
                  animate={{ opacity: 1, x: 0 }}
                  exit={{ opacity: 0, x: 20 }}
                  transition={{ delay: index * 0.05 }}
                  className="rounded-xl border-2 p-4 bg-gradient-to-r from-white to-slate-50"
                &gt;
                  &lt;div className="flex items-start justify-between gap-3"&gt;
                    &lt;div className="flex-1"&gt;
                      &lt;div className="flex items-center gap-2 mb-2"&gt;
                        &lt;h3 className="font-bold text-lg text-slate-800"&gt;
                          {order.item_name}
                        &lt;/h3&gt;
                        &lt;span className="text-sm text-slate-500"&gt;× {order.quantity}&lt;/span&gt;
                      &lt;/div&gt;
                      &lt;p className="text-xs text-slate-500 mb-2"&gt;
                        {order.category} · {new Date(order.created_at).toLocaleString()}
                      &lt;/p&gt;
                      &lt;span className={`inline-block px-3 py-1 rounded-full text-xs font-semibold bor
tusColor(order.status)}`}&gt;
                        {order.status}
                      &lt;/span&gt;
                    &lt;/div&gt;
                    {order.status !== "cancelled" &amp;&amp; order.status !== "served" &amp;&amp; (
                      &lt;motion.button
                        whileHover={{ scale: 1.1 }}
                        whileTap={{ scale: 0.9 }}
                        onClick={() =&gt; cancelOrder(order.id)}
                        className="text-red-500 hover:text-red-700 text-xl font-bold px-2"
                        title="Cancel order"
                      &gt;
                        ×
                      &lt;/motion.button&gt;
                    )}
                  &lt;/div&gt;
                &lt;/motion.div&gt;
              ))}
            &lt;/AnimatePresence&gt;
          &lt;/div&gt;
        )}
    &lt;/motion.section&gt;
```

```
{/* Events Section */}
{events.length &gt; 0 &amp;&amp; (
  &lt;motion.section
    initial={{ opacity: 0, y: 20 }}
    animate={{ opacity: 1, y: 0 }}
    transition={{ delay: 0.4, duration: 0.5 }}
    className="space-y-6"
  &gt;
    &lt;h2 className="text-3xl font-bold text-primary flex items-center gap-3"&gt;
      &lt;span className="text-4xl"&gt;■&lt;/span&gt;
      {t("upcoming_events") || "Upcoming Events"}
    &lt;/h2&gt;
    &lt;div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-6"&gt;
      {events.map((event, index) =&gt; (
        &lt;motion.div
          key={event.id}
          initial={{ opacity: 0, scale: 0.9 }}
          animate={{ opacity: 1, scale: 1 }}
          transition={{ delay: 0.5 + index * 0.1 }}
          whileHover={{ scale: 1.05, y: -10 }}
          className="rounded-3xl bg-white/90 backdrop-blur-md overflow-hidden shadow-xl border borde
        &gt;
          {event.image_url &amp;&amp; (
            &lt;div className="h-48 overflow-hidden"&gt;
              &lt;img
                src={`${apiUrl}${event.image_url}`}
                alt={event.title}
                className="w-full h-full object-cover hover:scale-110 transition-transform duration-
              /&gt;
            &lt;/div&gt;
          )}
          &lt;div className="p-6"&gt;
            &lt;h3 className="text-xl font-bold text-primary mb-2"&gt;{event.title}&lt;/h3&gt;
            &lt;p className="text-sm text-slate-600 mb-4 line-clamp-2"&gt;{event.description}&lt;/p&
            &lt;div className="space-y-2 text-xs text-slate-500"&gt;
              &lt;p className="flex items-center gap-2"&gt;
                &lt;span&gt;■&lt;/span&gt;
                {new Date(event.event_date).toLocaleString()}
              &lt;/p&gt;
              {event.location &amp;&amp; (
                &lt;p className="flex items-center gap-2"&gt;
                  &lt;span&gt;■&lt;/span&gt;
                  {event.location}
                &lt;/p&gt;
              )}
            &lt;/div&gt;
          &lt;/div&gt;
        &lt;/motion.div&gt;
      ))}
    &lt;/div&gt;
  &lt;/motion.section&gt;
)}

{/* Promos Section */}
{promos.length &gt; 0 &amp;&amp; (
  &lt;motion.section
    initial={{ opacity: 0, y: 20 }}
    animate={{ opacity: 1, y: 0 }}
    transition={{ delay: 0.6, duration: 0.5 }}
    className="space-y-6"
  &gt;
    &lt;h2 className="text-3xl font-bold text-primary flex items-center gap-3"&gt;
      &lt;span className="text-4xl"&gt;■&lt;/span&gt;
      {t("current_promos") || "Current Promotions"}
    &lt;/h2&gt;
    &lt;div className="grid grid-cols-1 sm:grid-cols-2 gap-6"&gt;
      {promos.map((promo, index) =&gt; (
        &lt;motion.div
          key={promo.id}
          initial={{ opacity: 0, scale: 0.9 }}
          animate={{ opacity: 1, scale: 1 }}
          transition={{ delay: 0.7 + index * 0.1 }}
          whileHover={{ scale: 1.03, y: -5 }}
```

```
                  className="rounded-3xl bg-gradient-to-br from-accent/20 via-primary/20 to-blue-500/20 p-6
    w-xl border-2 border-accent/30 relative overflow-hidden"
                &gt;
                  &lt;div className="absolute top-0 right-0 w-32 h-32 bg-accent/10 rounded-full -mr-16 -mt-1
    iv&gt;
                  {promo.image_url &amp;&amp; (
                    &lt;div className="h-48 overflow-hidden rounded-xl mb-4"&gt;
                      &lt;img
                        src={`${apiUrl}${promo.image_url}`}
                        alt={promo.title}
                        className="w-full h-full object-cover hover:scale-110 transition-transform duration-
                      /&gt;
                    &lt;/div&gt;
                  )}
                  &lt;h3 className="text-2xl font-bold text-primary mb-2 relative z-10"&gt;{promo.title}&lt;
                  &lt;p className="text-sm text-slate-700 mb-4 relative z-10"&gt;{promo.description}&lt;/p&g
                  {(promo.discount_percent || promo.discount_amount) &amp;&amp; (
                    &lt;p className="text-3xl font-bold text-accent mb-4 relative z-10"&gt;
                      {promo.discount_percent ? `${promo.discount_percent}% OFF` : `■${promo.discount_amount
                    &lt;/p&gt;
                  )}
                  {promo.code &amp;&amp; (
                    &lt;div className="bg-white/80 backdrop-blur-sm rounded-xl px-4 py-2 mb-4 relative z-10"
                      &lt;p className="text-sm font-semibold text-primary"&gt;
                        {t("promo_code") || "Code"}: &lt;span className="text-accent text-lg"&gt;{promo.code
    t;
                      &lt;/p&gt;
                    &lt;/div&gt;
                  )}
                  &lt;p className="text-xs text-slate-600 relative z-10"&gt;
                    Valid until: {new Date(promo.valid_until).toLocaleDateString()}
                  &lt;/p&gt;
                &lt;/motion.div&gt;
              ))}
            &lt;/div&gt;
          &lt;/motion.section&gt;
        )}
      &lt;/main&gt;
    &lt;/div&gt;
  );
};

export default CustomerPage;
```

File: frontend\src\pages\ForgotPasswordPage.tsx

Full path:
C:\Users\itodo\Hostelrec\frontend\src\pages\ForgotPasswordPage.tsx

```
// @ts-nocheck
import { useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import { motion } from "framer-motion";
import { useTranslation } from "react-i18next";

import Navbar from "../components/Navbar";
import { api } from "../services/api";

const ForgotPasswordPage = () =&gt; {
  const { t } = useTranslation();
  const navigate = useNavigate();
  const [email, setEmail] = useState("");
  const [code, setCode] = useState("");
  const [newPassword, setNewPassword] = useState("");
  const [confirmPassword, setConfirmPassword] = useState("");
  const [step, setStep] = useState&lt;"request" | "reset"&gt;("request");
  const [error, setError] = useState&lt;string | null&gt;(null);
  const [success, setSuccess] = useState&lt;string | null&gt;(null);
  const [loading, setLoading] = useState(false);

  const handleRequestReset = async (e: { preventDefault: () =&gt; void }) =&gt; {
    e.preventDefault();
    setError(null);
    setSuccess(null);
    setLoading(true);

    try {
      await api.post("/auth/forgot-password", {
        email: email,
      });

      setSuccess("reset_code_sent");
      setStep("reset");
    } catch (err: any) {
      console.error(err);
      setError("reset_request_failed");
    } finally {
      setLoading(false);
    }
  };

  const handleResetPassword = async (e: { preventDefault: () =&gt; void }) =&gt; {
    e.preventDefault();
    setError(null);
    setSuccess(null);

    // ■■■■■■■■■
    if (newPassword !== confirmPassword) {
      setError("passwords_do_not_match");
      return;
    }

    if (newPassword.length &lt; 6) {
      setError("password_too_short");
      return;
    }

    setLoading(true);

    try {
      await api.post("/auth/reset-password", {
        email: email,
        code: code,
        new_password: newPassword,
      });
```

```
    setSuccess("password_reset_successful");
    // ■■■■■■■■■■■■■■ ■■ ■■■■■■■■ ■■■■■■ ■■■■■■ 2 ■■■■■■■■
    setTimeout(() =&gt; {
      navigate("/login");
    }, 2000);
  } catch (err: any) {
    console.error(err);
    if (err.response?.status === 400) {
      setError(err.response?.data?.detail || "invalid_reset_code");
    } else {
      setError("password_reset_failed");
    }
  } finally {
    setLoading(false);
  }
};

return (
  &lt;div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-purple-50"&gt;
    &lt;Navbar /&gt;
    &lt;main className="mx-auto flex max-w-md flex-col gap-6 px-4 py-20"&gt;
      {step === "request" ? (
        &lt;motion.form
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ duration: 0.5 }}
          onSubmit={handleRequestReset}
          className="rounded-3xl bg-white/90 backdrop-blur-md p-8 shadow-2xl border border-white/20"
        &gt;
          &lt;motion.h2
            initial={{ opacity: 0, x: -20 }}
            animate={{ opacity: 1, x: 0 }}
            transition={{ delay: 0.2 }}
            className="text-3xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-text text-transp
          &gt;
            {t("forgot_password")}
          &lt;/motion.h2&gt;

          &lt;motion.p
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            transition={{ delay: 0.3 }}
            className="mt-2 text-sm text-slate-600"
          &gt;
            {t("forgot_password_description")}
          &lt;/motion.p&gt;

          &lt;motion.input
            initial={{ opacity: 0, x: -20 }}
            animate={{ opacity: 1, x: 0 }}
            transition={{ delay: 0.4 }}
            type="email"
            placeholder={t("email")}
            value={email}
            onChange={(e: { target: { value: string } }) =&gt;
              setEmail(e.target.value)
            }
            className="mt-6 w-full rounded-xl border-2 border-slate-200 px-4 py-3 focus:border-primary foc
  one focus:ring-2 focus:ring-primary/20 transition-all"
            required
          /&gt;

          {error &amp;&amp; (
            &lt;motion.p
              initial={{ opacity: 0, scale: 0.9 }}
              animate={{ opacity: 1, scale: 1 }}
              className="mt-4 rounded-xl bg-danger/10 px-4 py-2 text-danger text-sm"
            &gt;
              {t(error)}
            &lt;/motion.p&gt;
          )}

          {success &amp;&amp; (
```

```
          &lt;motion.p
            initial={{ opacity: 0, scale: 0.9 }}
            animate={{ opacity: 1, scale: 1 }}
            className="mt-4 rounded-xl bg-green-500/10 px-4 py-2 text-green-600 text-sm"
          &gt;
            {t(success)}
          &lt;/motion.p&gt;
        )}

        &lt;motion.button
          whileHover={{ scale: 1.02 }}
          whileTap={{ scale: 0.98 }}
          type="submit"
          disabled={loading}
          className="mt-6 w-full rounded-xl bg-gradient-to-r from-primary to-accent py-3 text-white font
adow-lg hover:shadow-xl transition-all disabled:opacity-50 disabled:cursor-not-allowed"
        &gt;
          {loading ? t("sending") : t("send_reset_code")}
        &lt;/motion.button&gt;

        &lt;motion.p
          initial={{ opacity: 0 }}
          animate={{ opacity: 1 }}
          transition={{ delay: 0.5 }}
          className="mt-4 text-center text-sm text-slate-600"
        &gt;
          {t("remember_password")}{" "}
          &lt;Link to="/login" className="text-accent hover:underline font-semibold"&gt;
            {t("Sign In")}
          &lt;/Link&gt;
        &lt;/motion.p&gt;
      &lt;/motion.form&gt;
    ) : (
      &lt;motion.form
        initial={{ opacity: 0, y: 20 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ duration: 0.5 }}
        onSubmit={handleResetPassword}
        className="rounded-3xl bg-white/90 backdrop-blur-md p-8 shadow-2xl border border-white/20"
      &gt;
        &lt;motion.h2
          initial={{ opacity: 0, x: -20 }}
          animate={{ opacity: 1, x: 0 }}
          transition={{ delay: 0.2 }}
          className="text-3xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-text text-transp
        &gt;
          {t("reset_password")}
        &lt;/motion.h2&gt;

        &lt;motion.p
          initial={{ opacity: 0 }}
          animate={{ opacity: 1 }}
          transition={{ delay: 0.3 }}
          className="mt-2 text-sm text-slate-600"
        &gt;
          {t("reset_password_description")}
        &lt;/motion.p&gt;

        &lt;motion.input
          initial={{ opacity: 0, x: -20 }}
          animate={{ opacity: 1, x: 0 }}
          transition={{ delay: 0.4 }}
          type="text"
          placeholder={t("reset_code")}
          value={code}
          onChange={(e: { target: { value: string } }) =&gt;
            setCode(e.target.value.toUpperCase().replace(/[^A-Z0-9]/g, "").slice(0, 8))
          }
          className="mt-6 w-full rounded-xl border-2 border-slate-200 px-4 py-3 text-center text-xl trac
focus:border-primary focus:outline-none focus:ring-2 focus:ring-primary/20 transition-all"
          maxLength={8}
          required
        /&gt;
```

```
        &lt;motion.input
          initial={{ opacity: 0, x: -20 }}
          animate={{ opacity: 1, x: 0 }}
          transition={{ delay: 0.5 }}
          type="password"
          placeholder={t("new_password")}
          value={newPassword}
          onChange={(e: { target: { value: string } }) =&gt;
            setNewPassword(e.target.value)
          }
          className="mt-3 w-full rounded-xl border-2 border-slate-200 px-4 py-3 focus:border-primary foc
one focus:ring-2 focus:ring-primary/20 transition-all"
          required
          minLength={6}
        /&gt;

        &lt;motion.input
          initial={{ opacity: 0, x: -20 }}
          animate={{ opacity: 1, x: 0 }}
          transition={{ delay: 0.6 }}
          type="password"
          placeholder={t("confirm_password")}
          value={confirmPassword}
          onChange={(e: { target: { value: string } }) =&gt;
            setConfirmPassword(e.target.value)
          }
          className="mt-3 w-full rounded-xl border-2 border-slate-200 px-4 py-3 focus:border-primary foc
one focus:ring-2 focus:ring-primary/20 transition-all"
          required
          minLength={6}
        /&gt;

        {error &amp;&amp; (
          &lt;motion.p
            initial={{ opacity: 0, scale: 0.9 }}
            animate={{ opacity: 1, scale: 1 }}
            className="mt-4 rounded-xl bg-danger/10 px-4 py-2 text-danger text-sm"
          &gt;
            {t(error)}
          &lt;/motion.p&gt;
        )}

        {success &amp;&amp; (
          &lt;motion.p
            initial={{ opacity: 0, scale: 0.9 }}
            animate={{ opacity: 1, scale: 1 }}
            className="mt-4 rounded-xl bg-green-500/10 px-4 py-2 text-green-600 text-sm"
          &gt;
            {t(success)}
          &lt;/motion.p&gt;
        )}

        &lt;div className="flex gap-3 mt-6"&gt;
          &lt;motion.button
            whileHover={{ scale: 1.02 }}
            whileTap={{ scale: 0.98 }}
            type="button"
            onClick={() =&gt; {
              setStep("request");
              setCode("");
              setNewPassword("");
              setConfirmPassword("");
              setError(null);
              setSuccess(null);
            }}
            className="flex-1 rounded-xl border-2 border-slate-200 px-4 py-3 text-slate-700 font-semibol
late-50 transition-all"
          &gt;
            {t("back")}
          &lt;/motion.button&gt;
          &lt;motion.button
            whileHover={{ scale: 1.02 }}
            whileTap={{ scale: 0.98 }}
```

```
                type="submit"
                disabled={loading || code.length !== 8}
                className="flex-1 rounded-xl bg-gradient-to-r from-primary to-accent py-3 text-white font-se
    w-lg hover:shadow-xl transition-all disabled:opacity-50 disabled:cursor-not-allowed"
              &gt;
                {loading ? t("resetting") : t("reset_password")}
              &lt;/motion.button&gt;
            &lt;/div&gt;
          &lt;/motion.form&gt;
        )}
      &lt;/main&gt;
    &lt;/div&gt;
  );
};

export default ForgotPasswordPage;
```

File: frontend\src\pages\LoginPage.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\pages\LoginPage.tsx

```
// @ts-nocheck
import { useState } from "react";
import { Link, useNavigate } from "react-router-dom";
import { motion } from "framer-motion";
import { useTranslation } from "react-i18next";

import Navbar from "../components/Navbar";
import { api } from "../services/api";
import { useAuthStore } from "../hooks/useAuthStore";

const LoginPage = () =&gt; {
  const { t } = useTranslation();
  const navigate = useNavigate();
  const setCredentials = useAuthStore((state: any) =&gt; state.setCredentials);
  const [form, setForm] = useState({ username: "", password: "", mfa_code: "" });
  const [error, setError] = useState&lt;string | null&gt;(null);
  const [mfaRequired, setMfaRequired] = useState(false);
  const [loading, setLoading] = useState(false);

  const submit = async (e: { preventDefault: () =&gt; void }) =&gt; {
    e.preventDefault();
    setError(null);
    setLoading(true);

    try {
      const params = new URLSearchParams();
      params.append("username", form.username);
      params.append("password", form.password);
      if (form.mfa_code) {
        params.append("mfa_code", form.mfa_code);
      }

      const res = await api.post("/auth/token", params, {
        headers: { "Content-Type": "application/x-www-form-urlencoded" }
      });

      // ■■■■■■■■■■, ■■■■■■■■■■ ■■ MFA
      if (res.data.mfa_required &amp;amp;&amp;amp; !form.mfa_code) {
        setMfaRequired(true);
        setLoading(false);
        return;
      }

      setCredentials(res.data.access_token, res.data.role);
      if (res.data.role === "manager" || res.data.role === "receptionist") {
        navigate("/admin");
      } else if (res.data.role === "cleaner") {
        navigate("/cleaner");
      } else {
        navigate("/customer");
      }
    } catch (err: any) {
      console.error(err);
      if (err.response?.status === 401) {
        setError("invalid_credentials");
      } else {
        setError("login_failed");
      }
    } finally {
      setLoading(false);
    }
  };

  return (
    &lt;div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-purple-50"&gt;
      &lt;Navbar /&gt;
      &lt;main className="mx-auto flex max-w-md flex-col gap-6 px-4 py-20"&gt;
        &lt;motion.form
```

```
      initial={{ opacity: 0, y: 20 }}
      animate={{ opacity: 1, y: 0 }}
      transition={{ duration: 0.5 }}
      onSubmit={submit}
      className="rounded-3xl bg-white/90 backdrop-blur-md p-8 shadow-2xl border border-white/20"
    &gt;
      &lt;motion.h2
        initial={{ opacity: 0, x: -20 }}
        animate={{ opacity: 1, x: 0 }}
        transition={{ delay: 0.2 }}
        className="text-3xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-text text-transpar
      &gt;
        {t("Sign In")}
      &lt;/motion.h2&gt;
      &lt;motion.input
        initial={{ opacity: 0, x: -20 }}
        animate={{ opacity: 1, x: 0 }}
        transition={{ delay: 0.3 }}
        type="email"
        placeholder={t("email")}
        value={form.username}
        onChange={(e: { target: { value: string } }) =&gt;
          setForm({ ...form, username: e.target.value })
        }
        className="mt-6 w-full rounded-xl border-2 border-slate-200 px-4 py-3 focus:border-primary focus
e focus:ring-2 focus:ring-primary/20 transition-all"
      /&gt;
      &lt;motion.input
        initial={{ opacity: 0, x: -20 }}
        animate={{ opacity: 1, x: 0 }}
        transition={{ delay: 0.4 }}
        type="password"
        placeholder={t("password")}
        value={form.password}
        onChange={(e: { target: { value: string } }) =&gt;
          setForm({ ...form, password: e.target.value })
        }
        className="mt-3 w-full rounded-xl border-2 border-slate-200 px-4 py-3 focus:border-primary focus
e focus:ring-2 focus:ring-primary/20 transition-all"
      /&gt;
      {mfaRequired &amp;&amp; (
        &lt;motion.div
          initial={{ opacity: 0, height: 0 }}
          animate={{ opacity: 1, height: "auto" }}
          transition={{ delay: 0.1 }}
          className="mt-3"
        &gt;
          &lt;motion.p
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            className="text-sm text-slate-600 mb-2"
          &gt;
            {t("mfa_code_required")}
          &lt;/motion.p&gt;
          &lt;motion.input
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            type="text"
            placeholder={t("mfa_code")}
            value={form.mfa_code}
            onChange={(e: { target: { value: string } }) =&gt;
              setForm({ ...form, mfa_code: e.target.value.replace(/\D/g, "").slice(0, 6) })
            }
            className="w-full rounded-xl border-2 border-slate-200 px-4 py-3 text-center text-xl trackin
us:border-primary focus:outline-none focus:ring-2 focus:ring-primary/20 transition-all"
            maxLength={6}
          /&gt;
        &lt;/motion.div&gt;
      )}
      &lt;motion.button
        whileHover={{ scale: 1.02 }}
        whileTap={{ scale: 0.98 }}
        type="submit"
```

```
            disabled={loading}
            className="mt-6 w-full rounded-xl bg-gradient-to-r from-primary to-accent py-3 text-white font-s
    ow-lg hover:shadow-xl transition-all disabled:opacity-50 disabled:cursor-not-allowed"
          &gt;
            {loading ? t("signing_in") : t("Sign In")}
          &lt;/motion.button&gt;
          {error &amp;&amp; (
            &lt;motion.p
              initial={{ opacity: 0, scale: 0.9 }}
              animate={{ opacity: 1, scale: 1 }}
              className="mt-4 rounded-xl bg-danger/10 px-4 py-2 text-danger"
            &gt;
              {t(error)}
            &lt;/motion.p&gt;
          )}
        &lt;/motion.form&gt;
        &lt;motion.p
          initial={{ opacity: 0 }}
          animate={{ opacity: 1 }}
          transition={{ delay: 0.5 }}
          className="text-center text-sm text-slate-600"
        &gt;
          {t("no_account") || "No account yet?"}{" "}
          &lt;Link to="/signup" className="text-accent hover:underline font-semibold"&gt;
            {t("sign Up")}
          &lt;/Link&gt;
        &lt;/motion.p&gt;
        &lt;motion.p
          initial={{ opacity: 0 }}
          animate={{ opacity: 1 }}
          transition={{ delay: 0.6 }}
          className="text-center text-sm text-slate-600"
        &gt;
          &lt;Link to="/forgot-password" className="text-accent hover:underline font-semibold"&gt;
            {t("forgot_password")}
          &lt;/Link&gt;
        &lt;/motion.p&gt;
      &lt;/main&gt;
    &lt;/div&gt;
  );
};

export default LoginPage;
```

File: frontend\src\pages\MenuPage.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\pages\MenuPage.tsx

```
// @ts-nocheck
import { useEffect, useState } from "react";
import { motion } from "framer-motion";
import { useTranslation } from "react-i18next";
import Navbar from "../components/Navbar";
import { api } from "../services/api";

type FoodItem = {
  id: number;
  name: string;
  description: string;
  price: number;
  image_url: string;
  category: string;
  available: boolean;
};

type DrinkItem = {
  id: number;
  name: string;
  description: string;
  price: number;
  image_url: string;
  category: string;
  available: boolean;
};

const MenuPage = () =&gt; {
  const { t } = useTranslation();
  const [foods, setFoods] = useState&lt;FoodItem[]&gt;([]);
  const [drinks, setDrinks] = useState&lt;DrinkItem[]&gt;([]);
  const [activeTab, setActiveTab] = useState&lt;"food" | "drinks"&gt;("food");

  useEffect(() =&gt; {
    loadMenu();
  }, []);

  const loadMenu = async () =&gt; {
    try {
      const [foodsRes, drinksRes] = await Promise.all([
        api.get("/content/foods?available_only=true"),
        api.get("/content/drinks?available_only=true"),
      ]);
      setFoods(foodsRes.data);
      setDrinks(drinksRes.data);
    } catch (err) {
      console.error("Failed to load menu", err);
    }
  };

  const apiUrl = import.meta.env.VITE_API_URL ?? "http://localhost:8000";

  return (
    &lt;div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-purple-50"&gt;
      &lt;Navbar /&gt;
      &lt;main className="mx-auto flex max-w-6xl flex-col gap-10 px-4 pb-20 pt-10"&gt;
        &lt;motion.section
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          className="text-center"
        &gt;
          &lt;h1 className="text-4xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-text text-tra
    ;
            {t("menu_title") || "Food &amp; Drinks Menu"}
          &lt;/h1&gt;
          &lt;p className="mt-3 text-slate-600"&gt;
            {t("menu_subtitle") || "Delicious meals and refreshing beverages"}
          &lt;/p&gt;
```

```
&lt;/motion.section&gt;

{/* Tabs */}
&lt;div className="flex gap-2 justify-center border-b border-slate-200"&gt;
  &lt;button
    onClick={() =&gt; setActiveTab("food")}
    className={`px-6 py-2 font-semibold border-b-2 transition-colors ${
      activeTab === "food"
        ? "border-primary text-primary"
        : "border-transparent text-slate-600 hover:text-primary"
    }`}
  &gt;
    {t("food") || "Food"}
  &lt;/button&gt;
  &lt;button
    onClick={() =&gt; setActiveTab("drinks")}
    className={`px-6 py-2 font-semibold border-b-2 transition-colors ${
      activeTab === "drinks"
        ? "border-primary text-primary"
        : "border-transparent text-slate-600 hover:text-primary"
    }`}
  &gt;
    {t("drinks") || "Drinks"}
  &lt;/button&gt;
&lt;/div&gt;

{/* Food Items */}
{activeTab === "food" &amp;&amp; (
  &lt;motion.div
    initial={{ opacity: 0, y: 20 }}
    animate={{ opacity: 1, y: 0 }}
    className="grid gap-6 md:grid-cols-3"
  &gt;
    {foods.map((food, idx) =&gt; (
      &lt;motion.div
        key={food.id}
        initial={{ opacity: 0, y: 20 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ delay: idx * 0.1 }}
        whileHover={{ scale: 1.05, y: -5 }}
        className="rounded-2xl bg-white/90 backdrop-blur-md p-6 shadow-xl border border-white/20"
      &gt;
        {food.image_url &amp;&amp; (
          &lt;img
            src={`${apiUrl}${food.image_url}`}
            alt={food.name}
            className="w-full h-48 object-cover rounded-xl mb-4"
          /&gt;
        )}
        &lt;h3 className="text-xl font-bold text-primary"&gt;{food.name}&lt;/h3&gt;
        &lt;p className="text-sm text-slate-600 mt-2"&gt;{food.description}&lt;/p&gt;
        &lt;p className="text-2xl font-bold text-accent mt-4"&gt;■{food.price}&lt;/p&gt;
      &lt;/motion.div&gt;
    ))}
  &lt;/motion.div&gt;
)}

{/* Drink Items */}
{activeTab === "drinks" &amp;&amp; (
  &lt;motion.div
    initial={{ opacity: 0, y: 20 }}
    animate={{ opacity: 1, y: 0 }}
    className="grid gap-6 md:grid-cols-3"
  &gt;
    {drinks.map((drink, idx) =&gt; (
      &lt;motion.div
        key={drink.id}
        initial={{ opacity: 0, y: 20 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ delay: idx * 0.1 }}
        whileHover={{ scale: 1.05, y: -5 }}
        className="rounded-2xl bg-white/90 backdrop-blur-md p-6 shadow-xl border border-white/20"
      &gt;
```

```
                {drink.image_url &amp;&amp; (
                  &lt;img
                    src={`${apiUrl}${drink.image_url}`}
                    alt={drink.name}
                    className="w-full h-48 object-cover rounded-xl mb-4"
                  /&gt;
                )}
                &lt;h3 className="text-xl font-bold text-primary"&gt;{drink.name}&lt;/h3&gt;
                &lt;p className="text-sm text-slate-600 mt-2"&gt;{drink.description}&lt;/p&gt;
                &lt;p className="text-2xl font-bold text-accent mt-4"&gt;■{drink.price}&lt;/p&gt;
              &lt;/motion.div&gt;
            ))}
          &lt;/motion.div&gt;
        )}

        {((activeTab === "food" &amp;&amp; foods.length === 0) || (activeTab === "drinks" &amp;&amp; drinks.
    )) &amp;&amp; (
          &lt;p className="text-center text-slate-500 py-10"&gt;
            {t("menu_empty") || "No items available at the moment."}
          &lt;/p&gt;
        )}
      &lt;/main&gt;
    &lt;/div&gt;
  );
};

export default MenuPage;
```

File: frontend\src\pages\MFAPage.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\pages\MFAPage.tsx

```
// @ts-nocheck
import { useState, useEffect } from "react";
import { motion } from "framer-motion";
import { useTranslation } from "react-i18next";

import Navbar from "../components/Navbar";
import { api } from "../services/api";

const MFAPage = () =&gt; {
  const { t } = useTranslation();
  const [mfaEnabled, setMfaEnabled] = useState(false);
  const [loading, setLoading] = useState(false);
  const [error, setError] = useState&lt;string | null&gt;(null);
  const [success, setSuccess] = useState&lt;string | null&gt;(null);
  const [password, setPassword] = useState("");
  const [mfaCode, setMfaCode] = useState("");
  const [step, setStep] = useState&lt;"main" | "enable" | "verify" | "disable"&gt;("main");

  useEffect(() =&gt; {
    loadMFAStatus();
  }, []);

  const loadMFAStatus = async () =&gt; {
    try {
      const res = await api.get("/auth/me");
      setMfaEnabled(res.data.mfa_enabled || false);
    } catch (err) {
      console.error("Failed to load MFA status:", err);
    }
  };

  const handleEnableMFA = async (e: { preventDefault: () =&gt; void }) =&gt; {
    e.preventDefault();
    setError(null);
    setSuccess(null);
    setLoading(true);

    try {
      const res = await api.post("/auth/mfa/enable", {
        password: password,
      });

      setSuccess("mfa_enabled_successfully");
      setMfaEnabled(true);
      setStep("main");
      setPassword("");
      await loadMFAStatus();

      // ■ ■■■■■■■■ ■■■■■■■■■■ ■■■■■■ ■■ ■■■■ ■■■■■■■■■■ ■■■■■ ■■■■ QR-■■■■■
      console.log("MFA Secret:", res.data.secret);
    } catch (err: any) {
      console.error(err);
      if (err.response?.status === 400) {
        setError(err.response?.data?.detail || "invalid_password");
      } else {
        setError("mfa_enable_failed");
      }
    } finally {
      setLoading(false);
    }
  };

  const handleVerifyMFA = async (e: { preventDefault: () =&gt; void }) =&gt; {
    e.preventDefault();
    setError(null);
    setSuccess(null);
    setLoading(true);
```

```
  try {
    await api.post("/auth/mfa/verify", {
      code: mfaCode,
    });

    setSuccess("mfa_code_verified");
    setMfaCode("");
    setStep("main");
  } catch (err: any) {
    console.error(err);
    if (err.response?.status === 400) {
      setError(err.response?.data?.detail || "invalid_mfa_code");
    } else {
      setError("mfa_verify_failed");
    }
  } finally {
    setLoading(false);
  }
};

const handleDisableMFA = async (e: { preventDefault: () =&gt; void }) =&gt; {
  e.preventDefault();
  setError(null);
  setSuccess(null);
  setLoading(true);

  try {
    await api.post("/auth/mfa/disable", {
      password: password,
    });

    setSuccess("mfa_disabled_successfully");
    setMfaEnabled(false);
    setStep("main");
    setPassword("");
    await loadMFAStatus();
  } catch (err: any) {
    console.error(err);
    if (err.response?.status === 400) {
      setError(err.response?.data?.detail || "invalid_password");
    } else {
      setError("mfa_disable_failed");
    }
  } finally {
    setLoading(false);
  }
};

return (
  &lt;div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-purple-50"&gt;
    &lt;Navbar /&gt;
    &lt;main className="mx-auto flex max-w-md flex-col gap-6 px-4 py-20"&gt;
      &lt;motion.div
        initial={{ opacity: 0, y: 20 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ duration: 0.5 }}
        className="rounded-3xl bg-white/90 backdrop-blur-md p-8 shadow-2xl border border-white/20"
      &gt;
        &lt;motion.h2
          initial={{ opacity: 0, x: -20 }}
          animate={{ opacity: 1, x: 0 }}
          transition={{ delay: 0.2 }}
          className="text-3xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-text text-transpar
        &gt;
          {t("mfa_settings")}
        &lt;/motion.h2&gt;

        &lt;motion.p
          initial={{ opacity: 0 }}
          animate={{ opacity: 1 }}
          transition={{ delay: 0.3 }}
          className="mt-2 text-sm text-slate-600"
        &gt;
```

```
        {t("mfa_description")}
      &lt;/motion.p&gt;

      {step === "main" &amp;&amp; (
        &lt;motion.div
          initial={{ opacity: 0 }}
          animate={{ opacity: 1 }}
          transition={{ delay: 0.4 }}
          className="mt-6 space-y-4"
        &gt;
          &lt;div className="flex items-center justify-between p-4 rounded-xl bg-slate-50"&gt;
            &lt;div&gt;
              &lt;p className="font-semibold"&gt;{t("mfa_status")}&lt;/p&gt;
              &lt;p className="text-sm text-slate-600"&gt;
                {mfaEnabled ? t("enabled") : t("disabled")}
              &lt;/p&gt;
            &lt;/div&gt;
            &lt;div className={`px-3 py-1 rounded-full text-xs font-semibold ${
              mfaEnabled ? "bg-green-100 text-green-700" : "bg-red-100 text-red-700"
            }`}&gt;
              {mfaEnabled ? t("enabled") : t("disabled")}
            &lt;/div&gt;
          &lt;/div&gt;

          {!mfaEnabled ? (
            &lt;motion.button
              whileHover={{ scale: 1.02 }}
              whileTap={{ scale: 0.98 }}
              onClick={() =&gt; setStep("enable")}
              className="w-full rounded-xl bg-gradient-to-r from-primary to-accent py-3 text-white font-
dow-lg hover:shadow-xl transition-all"
            &gt;
              {t("enable_mfa")}
            &lt;/motion.button&gt;
          ) : (
            &lt;div className="space-y-3"&gt;
              &lt;motion.button
                whileHover={{ scale: 1.02 }}
                whileTap={{ scale: 0.98 }}
                onClick={() =&gt; setStep("verify")}
                className="w-full rounded-xl bg-blue-500 py-3 text-white font-semibold shadow-lg hover:s
nsition-all"
              &gt;
                {t("verify_mfa_code")}
              &lt;/motion.button&gt;
              &lt;motion.button
                whileHover={{ scale: 1.02 }}
                whileTap={{ scale: 0.98 }}
                onClick={() =&gt; setStep("disable")}
                className="w-full rounded-xl bg-red-500 py-3 text-white font-semibold shadow-lg hover:sh
sition-all"
              &gt;
                {t("disable_mfa")}
              &lt;/motion.button&gt;
            &lt;/div&gt;
          )}
        &lt;/motion.div&gt;
      )}

      {step === "enable" &amp;&amp; (
        &lt;motion.form
          initial={{ opacity: 0 }}
          animate={{ opacity: 1 }}
          onSubmit={handleEnableMFA}
          className="mt-6 space-y-4"
        &gt;
          &lt;p className="text-sm text-slate-600"&gt;{t("mfa_enable_description")}&lt;/p&gt;
          &lt;input
            type="password"
            placeholder={t("password")}
            value={password}
            onChange={(e) =&gt; setPassword(e.target.value)}
            className="w-full rounded-xl border-2 border-slate-200 px-4 py-3 focus:border-primary focus:
```

```
 focus:ring-2 focus:ring-primary/20 transition-all"
            required
          /&gt;
          &lt;div className="flex gap-3"&gt;
            &lt;button
              type="button"
              onClick={() =&gt; {
                setStep("main");
                setPassword("");
                setError(null);
              }}
              className="flex-1 rounded-xl border-2 border-slate-200 px-4 py-3 text-slate-700 font-semib
-slate-50 transition-all"
            &gt;
              {t("cancel")}
            &lt;/button&gt;
            &lt;button
              type="submit"
              disabled={loading}
              className="flex-1 rounded-xl bg-gradient-to-r from-primary to-accent py-3 text-white font-
dow-lg hover:shadow-xl transition-all disabled:opacity-50"
            &gt;
              {loading ? t("enabling") : t("enable")}
            &lt;/button&gt;
          &lt;/div&gt;
        &lt;/motion.form&gt;
      )}

      {step === "verify" &amp;&amp; (
        &lt;motion.form
          initial={{ opacity: 0 }}
          animate={{ opacity: 1 }}
          onSubmit={handleVerifyMFA}
          className="mt-6 space-y-4"
        &gt;
          &lt;p className="text-sm text-slate-600"&gt;{t("mfa_verify_description")}&lt;/p&gt;
          &lt;input
            type="text"
            placeholder={t("mfa_code")}
            value={mfaCode}
            onChange={(e) =&gt; setMfaCode(e.target.value.replace(/\D/g, "").slice(0, 6))}
            className="w-full rounded-xl border-2 border-slate-200 px-4 py-3 text-center text-2xl tracki
cus:border-primary focus:outline-none focus:ring-2 focus:ring-primary/20 transition-all"
            maxLength={6}
            required
          /&gt;
          &lt;div className="flex gap-3"&gt;
            &lt;button
              type="button"
              onClick={() =&gt; {
                setStep("main");
                setMfaCode("");
                setError(null);
              }}
              className="flex-1 rounded-xl border-2 border-slate-200 px-4 py-3 text-slate-700 font-semib
-slate-50 transition-all"
            &gt;
              {t("cancel")}
            &lt;/button&gt;
            &lt;button
              type="submit"
              disabled={loading || mfaCode.length !== 6}
              className="flex-1 rounded-xl bg-blue-500 py-3 text-white font-semibold shadow-lg hover:sha
ition-all disabled:opacity-50"
            &gt;
              {loading ? t("verifying") : t("verify")}
            &lt;/button&gt;
          &lt;/div&gt;
        &lt;/motion.form&gt;
      )}

      {step === "disable" &amp;&amp; (
        &lt;motion.form
```

```
            initial={{ opacity: 0 }}
            animate={{ opacity: 1 }}
            onSubmit={handleDisableMFA}
            className="mt-6 space-y-4"
          &gt;
            &lt;p className="text-sm text-slate-600"&gt;{t("mfa_disable_description")}&lt;/p&gt;
            &lt;input
              type="password"
              placeholder={t("password")}
              value={password}
              onChange={(e) =&gt; setPassword(e.target.value)}
              className="w-full rounded-xl border-2 border-slate-200 px-4 py-3 focus:border-primary focus:
     focus:ring-2 focus:ring-primary/20 transition-all"
              required
            /&gt;
            &lt;div className="flex gap-3"&gt;
              &lt;button
                type="button"
                onClick={() =&gt; {
                  setStep("main");
                  setPassword("");
                  setError(null);
                }}
                className="flex-1 rounded-xl border-2 border-slate-200 px-4 py-3 text-slate-700 font-semib
    -slate-50 transition-all"
              &gt;
                {t("cancel")}
              &lt;/button&gt;
              &lt;button
                type="submit"
                disabled={loading}
                className="flex-1 rounded-xl bg-red-500 py-3 text-white font-semibold shadow-lg hover:shad
    tion-all disabled:opacity-50"
              &gt;
                {loading ? t("disabling") : t("disable")}
              &lt;/button&gt;
            &lt;/div&gt;
          &lt;/motion.form&gt;
        )}

        {error &amp;&amp; (
          &lt;motion.p
            initial={{ opacity: 0, scale: 0.9 }}
            animate={{ opacity: 1, scale: 1 }}
            className="mt-4 rounded-xl bg-danger/10 px-4 py-2 text-danger text-sm"
          &gt;
            {t(error)}
          &lt;/motion.p&gt;
        )}

        {success &amp;&amp; (
          &lt;motion.p
            initial={{ opacity: 0, scale: 0.9 }}
            animate={{ opacity: 1, scale: 1 }}
            className="mt-4 rounded-xl bg-green-500/10 px-4 py-2 text-green-600 text-sm"
          &gt;
            {t(success)}
          &lt;/motion.p&gt;
        )}
      &lt;/motion.div&gt;
    &lt;/main&gt;
  &lt;/div&gt;
 );
};

export default MFAPage;
```

File: frontend\src\pages\OrdersPage.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\pages\OrdersPage.tsx

```
import { useEffect, useState } from "react";
import { useTranslation } from "react-i18next";

import { api } from "../services/api";

type StaffOrder = {
  id: number;
  item_name: string;
  category: string;
  quantity: number;
  status: string;
  created_at: string;
  customer_email: string;
};

const OrdersPage = () =&gt; {
  const { t } = useTranslation();
  const [orders, setOrders] = useState&lt;StaffOrder[]&gt;([]);

  const loadOrders = async () =&gt; {
    try {
      const res = await api.get&lt;StaffOrder[]&gt;("/orders");
      setOrders(res.data);
    } catch (err) {
      console.error("Failed to load orders", err);
    }
  };

  useEffect(() =&gt; {
    loadOrders();
  }, []);

  const updateStatus = async (id: number, status: string) =&gt; {
    try {
      await api.post(`/orders/${id}/status?new_status=${status}`);
      loadOrders();
    } catch (err) {
      console.error("Failed to update order status", err);
      alert("Failed to update order status");
    }
  };

  const cancelOrder = async (id: number) =&gt; {
    if (!confirm("Are you sure you want to cancel this order?")) return;
    try {
      await api.post(`/orders/${id}/cancel`);
      loadOrders();
    } catch (err) {
      console.error("Failed to cancel order", err);
      alert("Failed to cancel order");
    }
  };

  return (
    &lt;div className="space-y-6"&gt;
      &lt;div className="flex items-center justify-between"&gt;
        &lt;h1 className="text-3xl font-bold text-primary"&gt;
          {t("orders_title") || "Customer Orders"}
        &lt;/h1&gt;
        &lt;button
          onClick={loadOrders}
          className="rounded-lg border border-slate-300 px-4 py-2 text-sm"
        &gt;
          {t("orders_refresh") || "Refresh"}
        &lt;/button&gt;
      &lt;/div&gt;

      &lt;div className="bg-white rounded-lg shadow-md overflow-hidden"&gt;
```

```
        &lt;table className="w-full text-sm"&gt;
          &lt;thead className="bg-primary text-white"&gt;
            &lt;tr&gt;
              &lt;th className="px-4 py-2 text-left"&gt;{t("orders_customer") || "Customer"}&lt;/th&gt;
              &lt;th className="px-4 py-2 text-left"&gt;{t("orders_item") || "Item"}&lt;/th&gt;
              &lt;th className="px-4 py-2 text-left"&gt;{t("orders_category") || "Category"}&lt;/th&gt;
              &lt;th className="px-4 py-2 text-left"&gt;{t("orders_quantity") || "Qty"}&lt;/th&gt;
              &lt;th className="px-4 py-2 text-left"&gt;{t("orders_status") || "Status"}&lt;/th&gt;
              &lt;th className="px-4 py-2 text-left"&gt;{t("orders_created") || "Created"}&lt;/th&gt;
              &lt;th className="px-4 py-2 text-right"&gt;{t("operations")}&lt;/th&gt;
            &lt;/tr&gt;
          &lt;/thead&gt;
          &lt;tbody&gt;
            {orders.map((o) =&gt; (
              &lt;tr key={o.id} className="border-b hover:bg-slate-50"&gt;
                &lt;td className="px-4 py-2"&gt;{o.customer_email}&lt;/td&gt;
                &lt;td className="px-4 py-2"&gt;{o.item_name}&lt;/td&gt;
                &lt;td className="px-4 py-2 capitalize"&gt;{o.category}&lt;/td&gt;
                &lt;td className="px-4 py-2"&gt;{o.quantity}&lt;/td&gt;
                &lt;td className="px-4 py-2 capitalize"&gt;{o.status}&lt;/td&gt;
                &lt;td className="px-4 py-2"&gt;
                  {new Date(o.created_at).toLocaleString()}
                &lt;/td&gt;
                &lt;td className="px-4 py-2 text-right space-x-2"&gt;
                  {o.status !== "cancelled" &amp;&amp; o.status !== "served" &amp;&amp; (
                    &lt;&gt;
                      &lt;button
                        onClick={() =&gt; updateStatus(o.id, "in_progress")}
                        className="text-accent text-xs hover:underline px-2 py-1 rounded hover:bg-accent/10"
                      &gt;
                        {t("orders_mark_in_progress") || "In progress"}
                      &lt;/button&gt;
                      &lt;button
                        onClick={() =&gt; updateStatus(o.id, "served")}
                        className="text-success text-xs hover:underline px-2 py-1 rounded hover:bg-success/1
                      &gt;
                        {t("orders_mark_served") || "Served"}
                      &lt;/button&gt;
                      &lt;button
                        onClick={() =&gt; cancelOrder(o.id)}
                        className="text-danger text-xs hover:underline px-2 py-1 rounded hover:bg-danger/10
    d"
                      &gt;
                        {t("orders_cancel") || "Cancel"}
                      &lt;/button&gt;
                    &lt;/&gt;
                  )}
                  {o.status === "cancelled" &amp;&amp; (
                    &lt;span className="text-xs text-red-600 font-semibold"&gt;Cancelled&lt;/span&gt;
                  )}
                  {o.status === "served" &amp;&amp; (
                    &lt;span className="text-xs text-green-600 font-semibold"&gt;Served&lt;/span&gt;
                  )}
                &lt;/td&gt;
              &lt;/tr&gt;
            ))}
            {orders.length === 0 &amp;&amp; (
              &lt;tr&gt;
                &lt;td
                  colSpan={7}
                  className="px-4 py-6 text-center text-slate-500 text-sm"
                &gt;
                  {t("orders_empty") || "No customer orders yet."}
                &lt;/td&gt;
              &lt;/tr&gt;
            )}
          &lt;/tbody&gt;
        &lt;/table&gt;
      &lt;/div&gt;
    &lt;/div&gt;
  );
};
```

```
export default OrdersPage;
```

File: frontend\src\pages\PasswordChangePage.tsx

Full path:
C:\Users\itodo\Hostelrec\frontend\src\pages\PasswordChangePage.tsx

```
// @ts-nocheck
import { useState } from "react";
import { useNavigate } from "react-router-dom";
import { motion } from "framer-motion";
import { useTranslation } from "react-i18next";

import Navbar from "../components/Navbar";
import { api } from "../services/api";
import { useAuthStore } from "../hooks/useAuthStore";

const PasswordChangePage = () =&gt; {
  const { t } = useTranslation();
  const navigate = useNavigate();
  const logout = useAuthStore((state: any) =&gt; state.logout);
  const [form, setForm] = useState({ old_password: "", new_password: "", confirm_password: "" });
  const [error, setError] = useState&lt;string | null&gt;(null);
  const [success, setSuccess] = useState(false);
  const [loading, setLoading] = useState(false);

  const submit = async (e: { preventDefault: () =&gt; void }) =&gt; {
    e.preventDefault();
    setError(null);
    setSuccess(false);

    // ■■■■■■■■■■
    if (form.new_password !== form.confirm_password) {
      setError("passwords_do_not_match");
      return;
    }

    if (form.new_password.length &lt; 6) {
      setError("password_too_short");
      return;
    }

    setLoading(true);
    try {
      await api.post("/auth/change-password", {
        old_password: form.old_password,
        new_password: form.new_password,
      });

      setSuccess(true);
      // ■■■■■■ ■■ ■■■■■■■■ ■■■■■■ 2 ■■■■■■■■
      setTimeout(() =&gt; {
        logout();
        navigate("/login");
      }, 2000);
    } catch (err: any) {
      console.error(err);
      if (err.response?.status === 400) {
        setError(err.response?.data?.detail || "invalid_old_password");
      } else {
        setError("password_change_failed");
      }
    } finally {
      setLoading(false);
    }
  };

  return (
    &lt;div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-purple-50"&gt;
      &lt;Navbar /&gt;
      &lt;main className="mx-auto flex max-w-md flex-col gap-6 px-4 py-20"&gt;
        &lt;motion.form
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
```

```
      transition={{ duration: 0.5 }}
      onSubmit={submit}
      className="rounded-3xl bg-white/90 backdrop-blur-md p-8 shadow-2xl border border-white/20"
    &gt;
      &lt;motion.h2
        initial={{ opacity: 0, x: -20 }}
        animate={{ opacity: 1, x: 0 }}
        transition={{ delay: 0.2 }}
        className="text-3xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-text text-transpar
      &gt;
        {t("change_password")}
      &lt;/motion.h2&gt;

      &lt;motion.p
        initial={{ opacity: 0 }}
        animate={{ opacity: 1 }}
        transition={{ delay: 0.3 }}
        className="mt-2 text-sm text-slate-600"
      &gt;
        {t("change_password_description")}
      &lt;/motion.p&gt;

      &lt;motion.input
        initial={{ opacity: 0, x: -20 }}
        animate={{ opacity: 1, x: 0 }}
        transition={{ delay: 0.4 }}
        type="password"
        placeholder={t("old_password")}
        value={form.old_password}
        onChange={(e: { target: { value: string } }) =&gt;
          setForm({ ...form, old_password: e.target.value })
        }
        className="mt-6 w-full rounded-xl border-2 border-slate-200 px-4 py-3 focus:border-primary focus
e focus:ring-2 focus:ring-primary/20 transition-all"
        required
      /&gt;

      &lt;motion.input
        initial={{ opacity: 0, x: -20 }}
        animate={{ opacity: 1, x: 0 }}
        transition={{ delay: 0.5 }}
        type="password"
        placeholder={t("new_password")}
        value={form.new_password}
        onChange={(e: { target: { value: string } }) =&gt;
          setForm({ ...form, new_password: e.target.value })
        }
        className="mt-3 w-full rounded-xl border-2 border-slate-200 px-4 py-3 focus:border-primary focus
e focus:ring-2 focus:ring-primary/20 transition-all"
        required
        minLength={6}
      /&gt;

      &lt;motion.input
        initial={{ opacity: 0, x: -20 }}
        animate={{ opacity: 1, x: 0 }}
        transition={{ delay: 0.6 }}
        type="password"
        placeholder={t("confirm_password")}
        value={form.confirm_password}
        onChange={(e: { target: { value: string } }) =&gt;
          setForm({ ...form, confirm_password: e.target.value })
        }
        className="mt-3 w-full rounded-xl border-2 border-slate-200 px-4 py-3 focus:border-primary focus
e focus:ring-2 focus:ring-primary/20 transition-all"
        required
        minLength={6}
      /&gt;

      {error &amp;&amp; (
        &lt;motion.p
          initial={{ opacity: 0, scale: 0.9 }}
          animate={{ opacity: 1, scale: 1 }}
```

```
              className="mt-4 rounded-xl bg-danger/10 px-4 py-2 text-danger text-sm"
            &gt;
              {t(error)}
            &lt;/motion.p&gt;
          )}

          {success &amp;&amp; (
            &lt;motion.p
              initial={{ opacity: 0, scale: 0.9 }}
              animate={{ opacity: 1, scale: 1 }}
              className="mt-4 rounded-xl bg-green-500/10 px-4 py-2 text-green-600 text-sm"
            &gt;
              {t("password_changed_successfully")}
            &lt;/motion.p&gt;
          )}

          &lt;motion.button
            whileHover={{ scale: 1.02 }}
            whileTap={{ scale: 0.98 }}
            type="submit"
            disabled={loading}
            className="mt-6 w-full rounded-xl bg-gradient-to-r from-primary to-accent py-3 text-white font-s
    ow-lg hover:shadow-xl transition-all disabled:opacity-50 disabled:cursor-not-allowed"
          &gt;
            {loading ? t("changing") : t("change_password")}
          &lt;/motion.button&gt;
        &lt;/motion.form&gt;
      &lt;/main&gt;
    &lt;/div&gt;
  );
};

export default PasswordChangePage;
```

File: frontend\src\pages\PublicHome.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\pages\PublicHome.tsx

```
// @ts-nocheck
import { useEffect, useState } from "react";
import { motion } from "framer-motion";
import { useTranslation } from "react-i18next";

import Navbar from "../components/Navbar";
import HeroSection from "../components/HeroSection";
import RoomCard from "../components/RoomCard";
import BookingWizard from "../components/BookingWizard";
import FloorPlan from "../components/FloorPlan";
import { api } from "../services/api";

type Occupancy = {
  occupancy_rate: number;
  available_rooms: number;
  occupied_rooms: number;
};

type Room = {
  id: number;
  room_number: string;
  title: string;
  description: string | null;
  price: number;
  location: string | null;
  wifi: boolean;
  features: string[] | null;
  image_url: string | null;
  status: string;
};

const PublicHome = () =&gt; {
  const { t } = useTranslation();
  const [occupancy, setOccupancy] = useState&lt;Occupancy | null&gt;(null);
  const [rooms, setRooms] = useState&lt;Room[]&gt;([]);

  useEffect(() =&gt; {
    // Fetch real occupancy data from API
    api
      .get("/stats/occupancy")
      .then((res: { data: Occupancy }) =&gt; setOccupancy(res.data))
      .catch((err) =&gt; {
        console.error("Failed to load occupancy", err);
        setOccupancy(null);
      });

    // Fetch real rooms from API
    api
      .get("/content/rooms", { params: { available_only: true } })
      .then((res: { data: Room[] }) =&gt; {
        setRooms(res.data);
      })
      .catch((err) =&gt; {
        console.error("Failed to load rooms", err);
        setRooms([]);
      });
  }, []);

  return (
    &lt;div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-purple-50"&gt;
      &lt;Navbar /&gt;
      &lt;main className="mx-auto flex max-w-6xl flex-col gap-6 sm:gap-10 px-4 sm:px-6 pb-10 sm:pb-20 pt-6 s
        &lt;HeroSection occupancyHigh={(occupancy?.occupancy_rate ?? 0) &gt; 0.8} /&gt;

        &lt;motion.section
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ delay: 0.2, duration: 0.5 }}
```

```
          className="grid gap-6 grid-cols-1 sm:grid-cols-2 lg:grid-cols-3"
        &gt;
          {rooms.length &gt; 0 ? (
            rooms.map((room) =&gt; (
              &lt;RoomCard
                key={room.id}
                id={room.id}
                title={room.title}
                price={room.price}
                amenities={room.features || []}
                status={room.status === "available" ? "available" : "occupied"}
                image_url={room.image_url}
                location={room.location}
                wifi={room.wifi}
              /&gt;
            ))
          ) : (
            &lt;div className="col-span-full text-center py-12 text-slate-600"&gt;
              &lt;p className="text-lg"&gt;{t("no_rooms_available") || "No rooms available at the moment."}&
              &lt;p className="text-sm mt-2"&gt;{t("check_back_later") || "Please check back later."}&lt;/p&
            &lt;/div&gt;
          )}
        &lt;/motion.section&gt;

        &lt;motion.section
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ delay: 0.3, duration: 0.5 }}
          className="grid gap-8 grid-cols-1 lg:grid-cols-2"
        &gt;
          &lt;BookingWizard /&gt;
          &lt;div className="space-y-4"&gt;
            &lt;motion.div
              whileHover={{ scale: 1.02, y: -5 }}
              className="rounded-3xl bg-white/90 backdrop-blur-md p-6 shadow-xl border border-white/20"
            &gt;
              &lt;h3 className="text-xl font-semibold text-primary"&gt;{t("todays occupancy")}&lt;/h3&gt;
              &lt;p className="text-5xl font-bold text-primary"&gt;
                {occupancy ? Math.round(occupancy.occupancy_rate * 100) : "--"}%
              &lt;/p&gt;
              &lt;p className="text-sm text-slate-500"&gt;
                {occupancy?.available_rooms} {t("rooms available")} · {occupancy?.occupied_rooms} {t("rooms
              &lt;/p&gt;
            &lt;/motion.div&gt;
            &lt;motion.div
              initial={{ opacity: 0, scale: 0.9 }}
              animate={{ opacity: 1, scale: 1 }}
              transition={{ delay: 0.4 }}
            &gt;
              &lt;FloorPlan rooms={rooms} /&gt;
            &lt;/motion.div&gt;
          &lt;/div&gt;
        &lt;/motion.section&gt;
      &lt;/main&gt;
    &lt;/div&gt;
  );
};

export default PublicHome;
```

File: frontend\src\pages\RoomDetailPage.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\pages\RoomDetailPage.tsx

```
// @ts-nocheck
import { useEffect, useState } from "react";
import { useParams, useNavigate, Link } from "react-router-dom";
import { motion, AnimatePresence } from "framer-motion";
import { useTranslation } from "react-i18next";

import Navbar from "../components/Navbar";
import { api } from "../services/api";

type RoomDetail = {
  id: number;
  room_number: string;
  title: string;
  description: string | null;
  price: number;
  location: string | null;
  wifi: boolean;
  features: string[] | null;
  image_url: string | null;
  status: string;
  images: Array&lt;{
    id: number;
    image_url: string;
    description: string | null;
    created_at: string;
  }&gt;;
};

const RoomDetailPage = () =&gt; {
  const { t } = useTranslation();
  const { roomId } = useParams&lt;{ roomId: string }&gt;();
  const navigate = useNavigate();
  const [room, setRoom] = useState&lt;RoomDetail | null&gt;(null);
  const [loading, setLoading] = useState(true);
  const [selectedImageIndex, setSelectedImageIndex] = useState(0);
  const [bookingForm, setBookingForm] = useState({
    guest_name: "",
    guest_passport: "",
    phone_number: "",
    check_in: "",
    check_out: "",
    payment_method: "cash",
  });
  const [bookingError, setBookingError] = useState&lt;string | null&gt;(null);
  const [bookingSuccess, setBookingSuccess] = useState(false);

  const apiUrl = import.meta.env.VITE_API_URL ?? "http://localhost:8000";

  useEffect(() =&gt; {
    if (roomId) {
      loadRoomDetail();
    }
  }, [roomId]);

  const loadRoomDetail = async () =&gt; {
    try {
      const res = await api.get&lt;RoomDetail&gt;(`/content/rooms/${roomId}`);
      setRoom(res.data);
      // Set selected image to main image or first gallery image
      if (res.data.image_url) {
        setSelectedImageIndex(-1); // -1 means main image
      } else if (res.data.images &amp;&amp; res.data.images.length &gt; 0) {
        setSelectedImageIndex(0);
      }
    } catch (err) {
      console.error("Failed to load room details", err);
      navigate("/");
    } finally {
```

```
    setLoading(false);
  }
};

const handleBooking = async (e: { preventDefault: () =&gt; void }) =&gt; {
  e.preventDefault();
  setBookingError(null);
  setBookingSuccess(false);

  if (!room) return;

  // Validate form
  if (!bookingForm.guest_name || !bookingForm.guest_passport || !bookingForm.phone_number) {
    setBookingError("Please fill in all required fields (name, passport, and phone number).");
    return;
  }

  if (!bookingForm.check_in || !bookingForm.check_out) {
    setBookingError("Please select both check-in and check-out dates.");
    return;
  }

  if (new Date(bookingForm.check_out) &lt;= new Date(bookingForm.check_in)) {
    setBookingError("Check-out date must be after check-in date.");
    return;
  }

  if (!bookingForm.payment_method) {
    setBookingError("Please select a payment method.");
    return;
  }

  try {
    const response = await api.post("/bookings", {
      guest_name: bookingForm.guest_name,
      guest_passport: bookingForm.guest_passport,
      phone_number: bookingForm.phone_number,
      room_id: room.id,
      check_in: bookingForm.check_in, // HTML date input returns YYYY-MM-DD format
      check_out: bookingForm.check_out,
      payment_method: bookingForm.payment_method,
    });
    setBookingSuccess(true);
    setBookingForm({
      guest_name: "",
      guest_passport: "",
      phone_number: "",
      check_in: "",
      check_out: "",
      payment_method: "cash",
    });
    setTimeout(() =&gt; setBookingSuccess(false), 5000);
  } catch (err: any) {
    console.error("Failed to book room", err);
    const errorDetail = err.response?.data?.detail;
    if (Array.isArray(errorDetail)) {
      // Pydantic validation errors
      setBookingError(errorDetail.map((e: any) =&gt; `${e.loc?.join('.')}: ${e.msg}`).join(', '));
    } else {
      setBookingError(errorDetail || "Failed to book room. Please try again.");
    }
  }
};

if (loading) {
  return (
    &lt;div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-purple-50 flex items-ce
  -center"&gt;
      &lt;motion.div
        animate={{ rotate: 360 }}
        transition={{ duration: 1, repeat: Infinity, ease: "linear" }}
        className="w-16 h-16 border-4 border-primary border-t-transparent rounded-full"
      /&gt;
```

```
    &lt;/div&gt;
  );
}

if (!room) {
  return (
    &lt;div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-purple-50"&gt;
      &lt;Navbar /&gt;
      &lt;div className="flex items-center justify-center min-h-[60vh]"&gt;
        &lt;div className="text-center"&gt;
          &lt;h2 className="text-2xl font-bold text-primary mb-4"&gt;Room not found&lt;/h2&gt;
          &lt;Link to="/" className="text-accent hover:underline"&gt;
            Return to home
          &lt;/Link&gt;
        &lt;/div&gt;
      &lt;/div&gt;
    &lt;/div&gt;
  );
}

// Combine main image and gallery images
const allImages = [];
if (room.image_url) {
  allImages.push({ url: room.image_url, isMain: true });
}
room.images.forEach((img) =&gt; {
  allImages.push({ url: img.image_url, isMain: false });
});

const currentImage = selectedImageIndex === -1
  ? (room.image_url ? { url: room.image_url, isMain: true } : allImages[0])
  : allImages[selectedImageIndex] || allImages[0];

return (
  &lt;div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-purple-50"&gt;
    &lt;Navbar /&gt;
    &lt;main className="mx-auto max-w-7xl px-4 sm:px-6 pb-10 sm:pb-20 pt-6 sm:pt-10"&gt;
      {/* Back Button */}
      &lt;motion.button
        initial={{ opacity: 0, x: -20 }}
        animate={{ opacity: 1, x: 0 }}
        onClick={() =&gt; navigate(-1)}
        className="mb-6 flex items-center gap-2 text-slate-600 hover:text-primary transition-colors"
      &gt;
        &lt;svg className="w-5 h-5" fill="none" stroke="currentColor" viewBox="0 0 24 24"&gt;
          &lt;path strokeLinecap="round" strokeLinejoin="round" strokeWidth={2} d="M15 19l-7-7 7-7" /&gt;
        &lt;/svg&gt;
        {t("back") || "Back"}
      &lt;/motion.button&gt;

      &lt;div className="grid grid-cols-1 lg:grid-cols-2 gap-8"&gt;
        {/* Image Gallery */}
        &lt;motion.div
          initial={{ opacity: 0, y: 20 }}
          animate={{ opacity: 1, y: 0 }}
          transition={{ duration: 0.5 }}
          className="space-y-4"
        &gt;
          {/* Main Image */}
          &lt;div className="relative rounded-3xl overflow-hidden shadow-2xl bg-white"&gt;
            &lt;AnimatePresence mode="wait"&gt;
              &lt;motion.img
                key={currentImage?.url}
                initial={{ opacity: 0 }}
                animate={{ opacity: 1 }}
                exit={{ opacity: 0 }}
                transition={{ duration: 0.3 }}
                src={`${apiUrl}${currentImage?.url}`}
                alt={room.title}
                className="w-full h-[500px] object-cover"
              /&gt;
            &lt;/AnimatePresence&gt;
            {room.status === "available" &amp;&amp; (
```

```
            &lt;div className="absolute top-4 right-4 bg-green-500 text-white px-4 py-2 rounded-full fon
hadow-lg"&gt;
              {t("available") || "Available"}
            &lt;/div&gt;
          )}
        &lt;/div&gt;

        {/* Thumbnail Gallery */}
        {allImages.length &gt; 1 &amp;&amp; (
          &lt;div className="grid grid-cols-5 gap-2"&gt;
            {room.image_url &amp;&amp; (
              &lt;motion.button
                whileHover={{ scale: 1.05 }}
                whileTap={{ scale: 0.95 }}
                onClick={() =&gt; setSelectedImageIndex(-1)}
                className={`relative rounded-xl overflow-hidden border-2 ${
                  selectedImageIndex === -1 ? "border-primary" : "border-transparent"
                }`}
              &gt;
                &lt;img
                  src={`${apiUrl}${room.image_url}`}
                  alt="Main"
                  className="w-full h-20 object-cover"
                /&gt;
                {selectedImageIndex === -1 &amp;&amp; (
                  &lt;div className="absolute inset-0 bg-primary/20 flex items-center justify-center"&gt
                    &lt;div className="w-3 h-3 bg-primary rounded-full" /&gt;
                  &lt;/div&gt;
                )}
              &lt;/motion.button&gt;
            )}
            {room.images.slice(0, 5 - (room.image_url ? 1 : 0)).map((img, idx) =&gt; {
              const actualIndex = room.image_url ? idx : idx;
              return (
                &lt;motion.button
                  key={img.id}
                  whileHover={{ scale: 1.05 }}
                  whileTap={{ scale: 0.95 }}
                  onClick={() =&gt; setSelectedImageIndex(actualIndex)}
                  className={`relative rounded-xl overflow-hidden border-2 ${
                    selectedImageIndex === actualIndex ? "border-primary" : "border-transparent"
                  }`}
                &gt;
                  &lt;img
                    src={`${apiUrl}${img.image_url}`}
                    alt={img.description || `Image ${idx + 1}`}
                    className="w-full h-20 object-cover"
                  /&gt;
                  {selectedImageIndex === actualIndex &amp;&amp; (
                    &lt;div className="absolute inset-0 bg-primary/20 flex items-center justify-center"&
                      &lt;div className="w-3 h-3 bg-primary rounded-full" /&gt;
                    &lt;/div&gt;
                  )}
                &lt;/motion.button&gt;
              );
            })}
          &lt;/div&gt;
        )}
      &lt;/motion.div&gt;

      {/* Room Details &amp; Booking */}
      &lt;motion.div
        initial={{ opacity: 0, y: 20 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ delay: 0.2, duration: 0.5 }}
        className="space-y-6"
      &gt;
        {/* Room Info */}
        &lt;div className="bg-white/90 backdrop-blur-md rounded-3xl p-6 sm:p-8 shadow-xl border border-w
          &lt;div className="flex items-start justify-between mb-4"&gt;
            &lt;div&gt;
              &lt;h1 className="text-3xl sm:text-4xl font-bold text-primary mb-2"&gt;{room.title}&lt;/h1
              &lt;p className="text-lg text-slate-600"&gt;Room {room.room_number}&lt;/p&gt;
```

```
    &lt;/div&gt;
    &lt;div className="text-right"&gt;
      &lt;p className="text-3xl font-bold text-accent"&gt;■{room.price.toFixed(2)}&lt;/p&gt;
      &lt;p className="text-sm text-slate-500"&gt;per night&lt;/p&gt;
    &lt;/div&gt;
  &lt;/div&gt;

  {room.description &amp;&amp; (
    &lt;p className="text-slate-700 mb-6 leading-relaxed"&gt;{room.description}&lt;/p&gt;
  )}

  &lt;div className="space-y-4"&gt;
    {room.location &amp;&amp; (
      &lt;div className="flex items-center gap-3"&gt;
        &lt;span className="text-2xl"&gt;■&lt;/span&gt;
        &lt;div&gt;
          &lt;p className="font-semibold text-slate-700"&gt;Location&lt;/p&gt;
          &lt;p className="text-slate-600"&gt;{room.location}&lt;/p&gt;
        &lt;/div&gt;
      &lt;/div&gt;
    )}

    &lt;div className="flex items-center gap-3"&gt;
      &lt;span className="text-2xl"&gt;{room.wifi ? "■" : "■"}&lt;/span&gt;
      &lt;div&gt;
        &lt;p className="font-semibold text-slate-700"&gt;WiFi&lt;/p&gt;
        &lt;p className="text-slate-600"&gt;{room.wifi ? "Available" : "Not Available"}&lt;/p&gt
      &lt;/div&gt;
    &lt;/div&gt;

    {room.features &amp;&amp; room.features.length &gt; 0 &amp;&amp; (
      &lt;div&gt;
        &lt;p className="font-semibold text-slate-700 mb-2"&gt;Features&lt;/p&gt;
        &lt;div className="flex flex-wrap gap-2"&gt;
          {room.features.map((feature, idx) =&gt; (
            &lt;span
              key={idx}
              className="bg-primary/10 text-primary px-3 py-1 rounded-full text-sm font-medium"
            &gt;
              {feature}
            &lt;/span&gt;
          ))}
        &lt;/div&gt;
      &lt;/div&gt;
    )}
  &lt;/div&gt;
&lt;/div&gt;

{/* Booking Form */}
{room.status === "available" &amp;&amp; (
  &lt;motion.div
    initial={{ opacity: 0, y: 20 }}
    animate={{ opacity: 1, y: 0 }}
    transition={{ delay: 0.4, duration: 0.5 }}
    className="bg-white/90 backdrop-blur-md rounded-3xl p-6 sm:p-8 shadow-xl border border-white
  &gt;
    &lt;h2 className="text-2xl font-bold text-primary mb-6"&gt;Book This Room&lt;/h2&gt;

    {bookingSuccess &amp;&amp; (
      &lt;motion.div
        initial={{ opacity: 0, y: -10 }}
        animate={{ opacity: 1, y: 0 }}
        className="mb-4 p-4 bg-green-100 border border-green-300 rounded-xl text-green-700"
      &gt;
        ■ Booking successful! We'll contact you soon.
      &lt;/motion.div&gt;
    )}

    {bookingError &amp;&amp; (
      &lt;motion.div
        initial={{ opacity: 0, y: -10 }}
        animate={{ opacity: 1, y: 0 }}
        className="mb-4 p-4 bg-red-100 border border-red-300 rounded-xl text-red-700"
```

```
              &gt;
                {bookingError}
              &lt;/motion.div&gt;
            )}

            &lt;form onSubmit={handleBooking} className="space-y-4"&gt;
              &lt;div&gt;
                &lt;label className="block text-sm font-semibold text-slate-700 mb-2"&gt;
                  Guest Name *
                &lt;/label&gt;
                &lt;input
                  type="text"
                  required
                  value={bookingForm.guest_name}
                  onChange={(e) =&gt; setBookingForm({ ...bookingForm, guest_name: e.target.value })}
                  className="w-full rounded-xl border border-slate-200 px-4 py-3 focus:ring-2 focus:ring
us:border-transparent"
                  placeholder="Enter your full name"
                /&gt;
              &lt;/div&gt;

              &lt;div&gt;
                &lt;label className="block text-sm font-semibold text-slate-700 mb-2"&gt;
                  Passport Number *
                &lt;/label&gt;
                &lt;input
                  type="text"
                  required
                  minLength={6}
                  value={bookingForm.guest_passport}
                  onChange={(e) =&gt; setBookingForm({ ...bookingForm, guest_passport: e.target.value })
                  className="w-full rounded-xl border border-slate-200 px-4 py-3 focus:ring-2 focus:ring
us:border-transparent"
                  placeholder="Enter passport number"
                /&gt;
              &lt;/div&gt;

              &lt;div&gt;
                &lt;label className="block text-sm font-semibold text-slate-700 mb-2"&gt;
                  Phone Number *
                &lt;/label&gt;
                &lt;input
                  type="tel"
                  required
                  minLength={10}
                  value={bookingForm.phone_number}
                  onChange={(e) =&gt; setBookingForm({ ...bookingForm, phone_number: e.target.value })}
                  className="w-full rounded-xl border border-slate-200 px-4 py-3 focus:ring-2 focus:ring
us:border-transparent"
                  placeholder="Enter phone number (e.g., +1234567890)"
                /&gt;
              &lt;/div&gt;

              &lt;div className="grid grid-cols-1 sm:grid-cols-2 gap-4"&gt;
                &lt;div&gt;
                  &lt;label className="block text-sm font-semibold text-slate-700 mb-2"&gt;
                    Check-in Date *
                  &lt;/label&gt;
                  &lt;input
                    type="date"
                    required
                    value={bookingForm.check_in}
                    onChange={(e) =&gt; setBookingForm({ ...bookingForm, check_in: e.target.value })}
                    min={new Date().toISOString().split("T")[0]}
                    className="w-full rounded-xl border border-slate-200 px-4 py-3 focus:ring-2 focus:ri
ocus:border-transparent"
                  /&gt;
                &lt;/div&gt;

                &lt;div&gt;
                  &lt;label className="block text-sm font-semibold text-slate-700 mb-2"&gt;
                    Check-out Date *
                  &lt;/label&gt;
```

```
                  &lt;input
                    type="date"
                    required
                    value={bookingForm.check_out}
                    onChange={(e) =&gt; setBookingForm({ ...bookingForm, check_out: e.target.value })}
                    min={bookingForm.check_in || new Date().toISOString().split("T")[0]}
                    className="w-full rounded-xl border border-slate-200 px-4 py-3 focus:ring-2 focus:ri
ocus:border-transparent"
                  /&gt;
                &lt;/div&gt;
              &lt;/div&gt;

              &lt;div&gt;
                &lt;label className="block text-sm font-semibold text-slate-700 mb-2"&gt;
                  Payment Method *
                &lt;/label&gt;
                &lt;select
                  required
                  value={bookingForm.payment_method}
                  onChange={(e) =&gt; setBookingForm({ ...bookingForm, payment_method: e.target.value })
                  className="w-full rounded-xl border border-slate-200 px-4 py-3 focus:ring-2 focus:ring
us:border-transparent"
                &gt;
                  &lt;option value="cash"&gt;■ Cash (Pay on arrival)&lt;/option&gt;
                  &lt;option value="card"&gt;■ Card (Pay on arrival)&lt;/option&gt;
                  &lt;option value="bank_transfer"&gt;■ Bank Transfer&lt;/option&gt;
                  &lt;option value="online"&gt;■ Online Payment&lt;/option&gt;
                &lt;/select&gt;
              &lt;/div&gt;

              {room &amp;&amp; bookingForm.check_in &amp;&amp; bookingForm.check_out &amp;&amp; (
                &lt;div className="bg-primary/10 rounded-xl p-4"&gt;
                  &lt;div className="flex justify-between items-center"&gt;
                    &lt;span className="font-semibold text-slate-700"&gt;Total Amount:&lt;/span&gt;
                    &lt;span className="text-2xl font-bold text-primary"&gt;
                      ■{(() =&gt; {
                        const checkIn = new Date(bookingForm.check_in);
                        const checkOut = new Date(bookingForm.check_out);
                        const nights = Math.ceil((checkOut.getTime() - checkIn.getTime()) / (1000 * 60 *
                        return (nights &gt; 0 ? nights : 1) * room.price;
                      })().toFixed(2)}
                    &lt;/span&gt;
                  &lt;/div&gt;
                  &lt;p className="text-xs text-slate-600 mt-1"&gt;
                    {(() =&gt; {
                      const checkIn = new Date(bookingForm.check_in);
                      const checkOut = new Date(bookingForm.check_out);
                      const nights = Math.ceil((checkOut.getTime() - checkIn.getTime()) / (1000 * 60 * 6
                      return `${nights &gt; 0 ? nights : 1} night(s) × ■${room.price.toFixed(2)}`;
                    })()}
                  &lt;/p&gt;
                &lt;/div&gt;
              )}

              &lt;motion.button
                whileHover={{ scale: 1.02 }}
                whileTap={{ scale: 0.98 }}
                type="submit"
                className="w-full rounded-xl bg-gradient-to-r from-primary to-accent px-6 py-4 text-whit
text-lg shadow-lg hover:shadow-xl transition-all"
              &gt;
                {t("book_now") || "Book Now"}
              &lt;/motion.button&gt;
            &lt;/form&gt;
          &lt;/motion.div&gt;
        )}

        {room.status !== "available" &amp;&amp; (
          &lt;div className="bg-slate-100 rounded-3xl p-6 sm:p-8 text-center"&gt;
            &lt;p className="text-lg font-semibold text-slate-600"&gt;
              This room is currently {room.status}
            &lt;/p&gt;
            &lt;Link to="/" className="text-primary hover:underline mt-2 inline-block"&gt;
```

```
                View other available rooms
              &lt;/Link&gt;
            &lt;/div&gt;
          )}
        &lt;/motion.div&gt;
      &lt;/div&gt;
    &lt;/main&gt;
  &lt;/div&gt;
 );
};

export default RoomDetailPage;
```

File: frontend\src\pages\SecurityPage.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\pages\SecurityPage.tsx

```
import { useState } from "react";
import { useTranslation } from "react-i18next";

import { api } from "../services/api";
import { useAuthStore } from "../hooks/useAuthStore";

type ApiKey = {
  integration_name: string;
  api_key: string;
};

const SecurityPage = () =&gt; {
  const { t } = useTranslation();
  const role = useAuthStore((state: any) =&gt; state.role);
  const [keys, setKeys] = useState&lt;ApiKey[]&gt;([]);
  const [secret, setSecret] = useState("");
  const [error, setError] = useState&lt;string | null&gt;(null);

  const reveal = async (e: any) =&gt; {
    e.preventDefault();
    try {
      const res = await api.get&lt;ApiKey[]&gt;("/admin/security/api-keys", { params: { secret_phrase: secre
      setKeys(res.data);
      setError(null);
    } catch (err) {
      console.error(err);
      setError("Invalid secret phrase");
    }
  };

  if (role !== "manager") {
    return &lt;div className="rounded-2xl bg-white p-6 shadow"&gt;{t("manager_only")}&lt;/div&gt;;
  }

  return (
    &lt;div className="space-y-6"&gt;
      &lt;h2 className="text-2xl font-bold text-primary"&gt;{t("security_center")}&lt;/h2&gt;
      &lt;form onSubmit={reveal} className="flex gap-3"&gt;
        &lt;input
          type="password"
          placeholder={t("secret_phrase")}
          value={secret}
          onChange={(e: any) =&gt; setSecret(e.target.value)}
          className="flex-1 rounded-xl border border-slate-200 px-4 py-2"
        /&gt;
        &lt;button className="rounded-xl bg-primary px-6 py-2 text-white"&gt;{t("reveal_keys")}&lt;/button&g
      &lt;/form&gt;
      {error &amp;&amp; &lt;p className="rounded-xl bg-danger/10 px-4 py-2 text-danger"&gt;{t("invalid_secre
    t;}
      &lt;div className="space-y-3"&gt;
        {keys.map((entry) =&gt; (
          &lt;div key={entry.integration_name} className="rounded-2xl bg-white p-4 shadow"&gt;
            &lt;p className="text-sm font-semibold uppercase text-slate-500"&gt;{entry.integration_name}&lt;
            &lt;p className="text-xl font-mono"&gt;{entry.api_key}&lt;/p&gt;
          &lt;/div&gt;
        ))}
      &lt;/div&gt;
    &lt;/div&gt;
  );
};

export default SecurityPage;
```

File: frontend\src\pages\SessionsPage.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\pages\SessionsPage.tsx

```
// @ts-nocheck
import { useState, useEffect } from "react";
import { motion } from "framer-motion";
import { useTranslation } from "react-i18next";

import Navbar from "../components/Navbar";
import { api } from "../services/api";

type Session = {
  id: number;
  created_at: string;
  last_activity: string;
  expires_at: string;
  ip_address: string | null;
  user_agent: string | null;
  is_current: boolean;
};

const SessionsPage = () =&gt; {
  const { t } = useTranslation();
  const [sessions, setSessions] = useState&lt;Session[]&gt;([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState&lt;string | null&gt;(null);

  useEffect(() =&gt; {
    loadSessions();
  }, []);

  const loadSessions = async () =&gt; {
    try {
      setLoading(true);
      const res = await api.get("/auth/sessions");
      setSessions(res.data);
    } catch (err: any) {
      console.error(err);
      setError("failed_to_load_sessions");
    } finally {
      setLoading(false);
    }
  };

  const handleEndSession = async (sessionId: number) =&gt; {
    if (!confirm(t("end_session_confirm"))) {
      return;
    }

    try {
      await api.delete(`/auth/sessions/${sessionId}`);
      await loadSessions();
    } catch (err: any) {
      console.error(err);
      alert(t("failed_to_end_session"));
    }
  };

  const formatDate = (dateString: string) =&gt; {
    const date = new Date(dateString);
    return date.toLocaleString();
  };

  const getUserAgentInfo = (userAgent: string | null) =&gt; {
    if (!userAgent) return t("unknown");

    // ■■■■■■■■ ■■■■■■■■ User-Agent
    if (userAgent.includes("Chrome")) return "Chrome";
    if (userAgent.includes("Firefox")) return "Firefox";
    if (userAgent.includes("Safari")) return "Safari";
    if (userAgent.includes("Edge")) return "Edge";
```

```
  return userAgent.substring(0, 50) + "...";
};

return (
  &lt;div className="min-h-screen bg-gradient-to-br from-slate-50 via-blue-50 to-purple-50"&gt;
    &lt;Navbar /&gt;
    &lt;main className="mx-auto max-w-4xl px-4 py-20"&gt;
      &lt;motion.div
        initial={{ opacity: 0, y: 20 }}
        animate={{ opacity: 1, y: 0 }}
        transition={{ duration: 0.5 }}
        className="rounded-3xl bg-white/90 backdrop-blur-md p-8 shadow-2xl border border-white/20"
      &gt;
        &lt;motion.h2
          initial={{ opacity: 0, x: -20 }}
          animate={{ opacity: 1, x: 0 }}
          transition={{ delay: 0.2 }}
          className="text-3xl font-bold bg-gradient-to-r from-primary to-accent bg-clip-text text-transpar
        &gt;
          {t("active_sessions")}
        &lt;/motion.h2&gt;

        {loading ? (
          &lt;div className="text-center py-8"&gt;
            &lt;p className="text-slate-600"&gt;{t("loading")}&lt;/p&gt;
          &lt;/div&gt;
        ) : error ? (
          &lt;div className="rounded-xl bg-danger/10 px-4 py-2 text-danger"&gt;
            {t(error)}
          &lt;/div&gt;
        ) : sessions.length === 0 ? (
          &lt;div className="text-center py-8"&gt;
            &lt;p className="text-slate-600"&gt;{t("no_active_sessions")}&lt;/p&gt;
          &lt;/div&gt;
        ) : (
          &lt;div className="space-y-4"&gt;
            {sessions.map((session, index) =&gt; (
              &lt;motion.div
                key={session.id}
                initial={{ opacity: 0, y: 10 }}
                animate={{ opacity: 1, y: 0 }}
                transition={{ delay: 0.1 * index }}
                className={`rounded-xl p-4 border-2 ${
                  session.is_current
                    ? "border-primary bg-primary/5"
                    : "border-slate-200 bg-white"
                }`}
              &gt;
                &lt;div className="flex items-start justify-between"&gt;
                  &lt;div className="flex-1"&gt;
                    &lt;div className="flex items-center gap-2 mb-2"&gt;
                      &lt;h3 className="font-semibold text-primary"&gt;
                        {session.is_current ? t("current_session") : t("session")} #{session.id}
                      &lt;/h3&gt;
                      {session.is_current &amp;&amp; (
                        &lt;span className="px-2 py-1 rounded-full bg-primary text-white text-xs font-semi
                          {t("current")}
                        &lt;/span&gt;
                      )}
                    &lt;/div&gt;
                    &lt;div className="grid grid-cols-1 md:grid-cols-2 gap-2 text-sm text-slate-600"&gt;
                      &lt;div&gt;
                        &lt;span className="font-semibold"&gt;{t("ip_address")}:&lt;/span&gt;{" "}
                        {session.ip_address || t("unknown")}
                      &lt;/div&gt;
                      &lt;div&gt;
                        &lt;span className="font-semibold"&gt;{t("device")}:&lt;/span&gt;{" "}
                        {getUserAgentInfo(session.user_agent)}
                      &lt;/div&gt;
                      &lt;div&gt;
                        &lt;span className="font-semibold"&gt;{t("created_at")}:&lt;/span&gt;{" "}
                        {formatDate(session.created_at)}
                      &lt;/div&gt;
```

```
                        &lt;div&gt;
                          &lt;span className="font-semibold"&gt;{t("last_activity")}:&lt;/span&gt;{" "}
                          {formatDate(session.last_activity)}
                        &lt;/div&gt;
                        &lt;div&gt;
                          &lt;span className="font-semibold"&gt;{t("expires_at")}:&lt;/span&gt;{" "}
                          {formatDate(session.expires_at)}
                        &lt;/div&gt;
                      &lt;/div&gt;
                    &lt;/div&gt;
                    {!session.is_current &amp;&amp; (
                      &lt;button
                        onClick={() =&gt; handleEndSession(session.id)}
                        className="ml-4 px-4 py-2 rounded-xl bg-red-500 text-white text-sm font-semibold hov
    0 transition-all"
                      &gt;
                        {t("end_session")}
                      &lt;/button&gt;
                    )}
                  &lt;/div&gt;
                &lt;/motion.div&gt;
              ))}
            &lt;/div&gt;
          )}
        &lt;/motion.div&gt;
      &lt;/main&gt;
    &lt;/div&gt;
  );
};

export default SessionsPage;
```

File: frontend\src\pages\SignupPage.tsx

Full path: C:\Users\itodo\Hostelrec\frontend\src\pages\SignupPage.tsx

```
import { useState } from "react";
import { useNavigate } from "react-router-dom";
import { useTranslation } from "react-i18next";

import Navbar from "../components/Navbar";
import { api } from "../services/api";
import { useAuthStore } from "../hooks/useAuthStore";

const SignupPage = () =&gt; {
  const { t } = useTranslation();
  const navigate = useNavigate();
  const setCredentials = useAuthStore((state: any) =&gt; state.setCredentials);
  const [form, setForm] = useState({ email: "", full_name: "", password: "" });
  const [error, setError] = useState&lt;string | null&gt;(null);

  const submit = async (e: { preventDefault: () =&gt; void }) =&gt; {
    e.preventDefault();
    try {
      const res = await api.post("/auth/signup", {
        email: form.email,
        full_name: form.full_name,
        password: form.password
      });
      setCredentials(res.data.access_token, res.data.role);
      if (res.data.role === "manager" || res.data.role === "receptionist") {
        navigate("/admin");
      } else if (res.data.role === "cleaner") {
        navigate("/cleaner");
      } else {
        navigate("/customer");
      }
    } catch (err) {
      console.error(err);
      setError("signup_failed");
    }
  };

  return (
    &lt;div className="space-y-10"&gt;
      &lt;Navbar /&gt;
      &lt;main className="mx-auto flex max-w-md flex-col gap-6 px-4"&gt;
        &lt;form onSubmit={submit} className="rounded-3xl bg-white p-8 shadow-xl"&gt;
          &lt;h2 className="text-2xl font-bold text-primary"&gt;{t("signup")}&lt;/h2&gt;
          &lt;input
            type="email"
            placeholder={t("email")}
            value={form.email}
            onChange={(e: { target: { value: string } }) =&gt;
              setForm({ ...form, email: e.target.value })
            }
            className="mt-6 w-full rounded-xl border border-slate-200 px-4 py-3"
          /&gt;
          &lt;input
            type="text"
            placeholder={t("full_name")}
            value={form.full_name}
            onChange={(e: { target: { value: string } }) =&gt;
              setForm({ ...form, full_name: e.target.value })
            }
            className="mt-3 w-full rounded-xl border border-slate-200 px-4 py-3"
          /&gt;
          &lt;input
            type="password"
            placeholder={t("password")}
            value={form.password}
            onChange={(e: { target: { value: string } }) =&gt;
              setForm({ ...form, password: e.target.value })
            }
```

```
            className="mt-3 w-full rounded-xl border border-slate-200 px-4 py-3"
          /&gt;
          &lt;button className="mt-6 w-full rounded-xl bg-primary py-3 text-white"&gt;
            {t("signup")}
          &lt;/button&gt;
          {error &amp;&amp; (
            &lt;p className="mt-4 rounded-xl bg-danger/10 px-4 py-2 text-danger"&gt;
              {t(error)}
            &lt;/p&gt;
          )}
        &lt;/form&gt;
      &lt;/main&gt;
    &lt;/div&gt;
  );
};

export default SignupPage;
```

File: frontend\src\pages\StaffManagementPage.tsx

Full path:
C:\Users\itodo\Hostelrec\frontend\src\pages\StaffManagementPage.tsx

```
import { useEffect, useState } from "react";
import { useTranslation } from "react-i18next";
import { api } from "../services/api";

type StaffMember = {
  id: number;
  username: string;
  role: string;
  total_shifts?: number;
};

const StaffManagementPage = () =&gt; {
  const { t } = useTranslation();
  const [staff, setStaff] = useState&lt;StaffMember[]&gt;([]);
  const [loading, setLoading] = useState(true);
  const [showModal, setShowModal] = useState(false);
  const [editing, setEditing] = useState&lt;StaffMember | null&gt;(null);
  const [formData, setFormData] = useState({ username: "", password: "", role: "receptionist" });

  useEffect(() =&gt; {
    loadStaff();
  }, []);

  const loadStaff = async () =&gt; {
    try {
      // Load staff plus attendance summary (how many days they've worked)
      const [staffRes, attendanceRes] = await Promise.all([
        api.get&lt;StaffMember[]&gt;("/staff/"),
        api.get&lt;{ id: number; total_shifts: number }[]&gt;("/staff/attendance/summary")
      ]);
      const attendanceMap = new Map(
        attendanceRes.data.map((a) =&gt; [a.id, a.total_shifts])
      );
      setStaff(
        staffRes.data.map((member) =&gt; ({
          ...member,
          total_shifts: attendanceMap.get(member.id) ?? 0
        }))
      );
    } catch (err) {
      console.error("Failed to load staff", err);
    } finally {
      setLoading(false);
    }
  };

  const handleSubmit = async (e: React.FormEvent) =&gt; {
    e.preventDefault();
    try {
      if (editing) {
        await api.put(`/staff/${editing.id}`, { role: formData.role, password: formData.password || undefine
      } else {
        await api.post("/staff/", formData);
      }
      setShowModal(false);
      setEditing(null);
      setFormData({ username: "", password: "", role: "receptionist" });
      loadStaff();
    } catch (err) {
      console.error("Failed to save staff", err);
      alert("Failed to save staff member");
    }
  };

  const handleDelete = async (id: number) =&gt; {
    if (!confirm(t("delete_confirm"))) return;
    try {
```

```
    await api.delete(`/staff/${id}`);
    loadStaff();
  } catch (err) {
    console.error("Failed to delete staff", err);
    alert("Failed to delete staff member");
  }
};

const openEdit = (member: StaffMember) =&gt; {
  setEditing(member);
  setFormData({ username: member.username, password: "", role: member.role });
  setShowModal(true);
};

const openAdd = () =&gt; {
  setEditing(null);
  setFormData({ username: "", password: "", role: "receptionist" });
  setShowModal(true);
};

if (loading) {
  return &lt;div className="text-center py-10"&gt;{t("loading") || "Loading..."}&lt;/div&gt;;
}

return (
  &lt;div className="space-y-6"&gt;
    &lt;div className="flex items-center justify-between"&gt;
      &lt;h1 className="text-3xl font-bold text-primary"&gt;{t("staff_management")}&lt;/h1&gt;
      &lt;button
        onClick={openAdd}
        className="rounded-lg bg-accent px-4 py-2 text-white font-semibold hover:bg-amber-600 transition-c
      &gt;
        {t("add_staff")}
      &lt;/button&gt;
    &lt;/div&gt;

    &lt;div className="bg-white rounded-lg shadow-md overflow-hidden"&gt;
      &lt;table className="w-full"&gt;
        &lt;thead className="bg-primary text-white"&gt;
          &lt;tr&gt;
            &lt;th className="px-6 py-3 text-left"&gt;{t("staff_name")}&lt;/th&gt;
            &lt;th className="px-6 py-3 text-left"&gt;{t("staff_role")}&lt;/th&gt;
            &lt;th className="px-6 py-3 text-left"&gt;{t("total_shifts") || "Days worked"}&lt;/th&gt;
            &lt;th className="px-6 py-3 text-right"&gt;{t("operations")}&lt;/th&gt;
          &lt;/tr&gt;
        &lt;/thead&gt;
        &lt;tbody&gt;
          {staff.map((member) =&gt; (
            &lt;tr key={member.id} className="border-b hover:bg-slate-50"&gt;
              &lt;td className="px-6 py-4"&gt;{member.username}&lt;/td&gt;
              &lt;td className="px-6 py-4"&gt;
                &lt;span className="px-3 py-1 rounded-full text-sm bg-slate-200 text-slate-700"&gt;
                  {t(member.role) || member.role}
                &lt;/span&gt;
              &lt;/td&gt;
              &lt;td className="px-6 py-4"&gt;
                &lt;span className="text-sm font-semibold"&gt;
                  {member.total_shifts ?? 0}
                &lt;/span&gt;
              &lt;/td&gt;
              &lt;td className="px-6 py-4 text-right space-x-2"&gt;
                &lt;button
                  onClick={() =&gt; openEdit(member)}
                  className="text-accent hover:underline"
                &gt;
                  {t("edit_staff")}
                &lt;/button&gt;
                &lt;button
                  onClick={() =&gt; handleDelete(member.id)}
                  className="text-danger hover:underline"
                &gt;
                  {t("delete_staff")}
                &lt;/button&gt;
```

```
            &lt;/td&gt;
          &lt;/tr&gt;
        ))}
      &lt;/tbody&gt;
    &lt;/table&gt;
  &lt;/div&gt;

  {showModal &amp;&amp; (
    &lt;div className="fixed inset-0 bg-black/50 flex items-center justify-center z-50"&gt;
      &lt;div className="bg-white rounded-lg p-6 w-full max-w-md"&gt;
        &lt;h2 className="text-2xl font-bold mb-4"&gt;
          {editing ? t("edit_staff") : t("add_staff")}
        &lt;/h2&gt;
        &lt;form onSubmit={handleSubmit} className="space-y-4"&gt;
          &lt;div&gt;
            &lt;label className="block text-sm font-semibold mb-1"&gt;{t("username")}&lt;/label&gt;
            &lt;input
              type="text"
              value={formData.username}
              onChange={(e) =&gt; setFormData({ ...formData, username: e.target.value })}
              disabled={!!editing}
              required={!editing}
              className="w-full rounded-lg border px-3 py-2"
            /&gt;
          &lt;/div&gt;
          &lt;div&gt;
            &lt;label className="block text-sm font-semibold mb-1"&gt;
              {t("password")} {editing &amp;&amp; "(leave blank to keep current)"}
            &lt;/label&gt;
            &lt;input
              type="password"
              value={formData.password}
              onChange={(e) =&gt; setFormData({ ...formData, password: e.target.value })}
              required={!editing}
              className="w-full rounded-lg border px-3 py-2"
            /&gt;
          &lt;/div&gt;
          &lt;div&gt;
            &lt;label className="block text-sm font-semibold mb-1"&gt;{t("staff_role")}&lt;/label&gt;
            &lt;select
              value={formData.role}
              onChange={(e) =&gt; setFormData({ ...formData, role: e.target.value })}
              className="w-full rounded-lg border px-3 py-2"
            &gt;
              &lt;option value="receptionist"&gt;{t("receptionist")}&lt;/option&gt;
              &lt;option value="manager"&gt;{t("manager")}&lt;/option&gt;
              &lt;option value="cleaner"&gt;{t("cleaner_role")}&lt;/option&gt;
            &lt;/select&gt;
          &lt;/div&gt;
          &lt;div className="flex gap-3 justify-end"&gt;
            &lt;button
              type="button"
              onClick={() =&gt; {
                setShowModal(false);
                setEditing(null);
              }}
              className="px-4 py-2 rounded-lg border hover:bg-slate-50"
            &gt;
              {t("cancel")}
            &lt;/button&gt;
            &lt;button
              type="submit"
              className="px-4 py-2 rounded-lg bg-accent text-white hover:bg-amber-600"
            &gt;
              {t("save")}
            &lt;/button&gt;
          &lt;/div&gt;
        &lt;/form&gt;
      &lt;/div&gt;
    &lt;/div&gt;
  )}
&lt;/div&gt;
);
```

```
};

export default StaffManagementPage;
```

File: frontend\src\services\api.ts

Full path: C:\Users\itodo\Hostelrec\frontend\src\services\api.ts

```
import axios from "axios";

export const api = axios.create({
  baseURL: import.meta.env.VITE_API_URL ?? "http://localhost:8000",
  withCredentials: false
});

export const setAuthToken = (token?: string) =&gt; {
  if (token) {
    api.defaults.headers.common.Authorization = `Bearer ${token}`;
  } else {
    delete api.defaults.headers.common.Authorization;
  }
};
```

File: frontend\src\types\babel__core\index.d.ts

Full path:
C:\Users\itodo\Hostelrec\frontend\src\types\babel__core\index.d.ts

```
export {};

declare namespace babel__core {
  // Minimal placeholder so TypeScript stops complaining about missing 'babel__core' lib types.
  const _placeholder: unknown;
}
```

File: frontend\src\types\framer-motion.d.ts

Full path: C:\Users\itodo\Hostelrec\frontend\src\types\framer-motion.d.ts

```
declare module "framer-motion" {
  import * as React from "react";

  export interface MotionProps extends React.HTMLAttributes&lt;HTMLElement&gt; {
    className?: string;
    initial?: any;
    animate?: any;
    transition?: any;
    whileHover?: any;
    whileTap?: any;
  }

  export const motion: {
    div: React.FC&lt;MotionProps&gt;;
    span: React.FC&lt;MotionProps&gt;;
    section: React.FC&lt;MotionProps&gt;;
  };
}
```

File: frontend\src\types\jsx.d.ts

Full path: C:\Users\itodo\Hostelrec\frontend\src\types\jsx.d.ts

```
export {};

declare global {
  namespace JSX {
    interface IntrinsicElements {
      [elemName: string]: any;
    }
  }
}
```

File: frontend\src\types\react-i18next.d.ts

Full path: C:\Users\itodo\Hostelrec\frontend\src\types\react-i18next.d.ts

```
declare module "react-i18next" {
  import * as React from "react";

  export interface UseTranslationResponse {
    t: (key: string, options?: Record&amp;lt;string, unknown&amp;gt;) =&amp;gt; string;
    i18n: {
      language: string;
      changeLanguage: (lng: string) =&amp;gt; Promise&amp;lt;void&amp;gt;;
    };
  }

  export function useTranslation(): UseTranslationResponse;

  export const I18nextProvider: React.FC&amp;lt;{ i18n: any; children?: React.ReactNode }&amp;gt;;
}
```

File: frontend\src\types\react-jsx-runtime.d.ts

Full path:
C:\Users\itodo\Hostelrec\frontend\src\types\react-jsx-runtime.d.ts

```
declare module "react/jsx-runtime" {
  export const Fragment: unique symbol;
  export function jsx(type: any, props: any, key?: any): any;
  export function jsxs(type: any, props: any, key?: any): any;
  export function jsxDEV(type: any, props: any, key?: any, isStaticChildren?: boolean, source?: any, self?:
}
```

File: frontend\src\types\react-router-dom.d.ts

Full path:
C:\Users\itodo\Hostelrec\frontend\src\types\react-router-dom.d.ts

```
declare module "react-router-dom" {
  import * as React from "react";

  export const BrowserRouter: React.FC&lt;{ children?: React.ReactNode }&gt;;
  export const Routes: React.FC&lt;{ children?: React.ReactNode }&gt;;
  export const Route: React.FC&lt;{ path?: string; element?: React.ReactNode }&gt;;
  export const Navigate: React.FC&lt;{ to: string; replace?: boolean; state?: unknown }&gt;;
  export const NavLink: React.FC&lt;{
    to: string;
    end?: boolean;
    className?: string | ((props: { isActive: boolean }) =&gt; string);
    children?: React.ReactNode | ((props: { isActive: boolean }) =&gt; React.ReactNode);
    key?: React.Key;
  }&gt;;
  export const Outlet: React.FC;
  export const Link: React.FC&lt;{ to: string; className?: string; children?: React.ReactNode }&gt;;
  export const useNavigate: () =&gt; (to: string, opts?: { replace?: boolean; state?: unknown }) =&gt; void;
  export const useLocation: () =&gt; { pathname: string };
  export const useParams: &lt;T extends Record&lt;string, string&gt;&gt;() =&gt; T;
}
```

File: frontend\src\types\react.d.ts

Full path: C:\Users\itodo\Hostelrec\frontend\src\types\react.d.ts

```
declare module "react" {
  export type ReactNode = any;
  export type FC&lt;P = {}&gt; = (props: P &amp; { children?: ReactNode }) =&gt; ReactNode;

  export function createElement(type: any, props?: any, ...children: any[]): ReactNode;
  export function useState&lt;T&gt;(initial: T): [T, (value: T | ((prev: T) =&gt; T)) =&gt; void];
  export function useEffect(effect: () =&gt; void | (() =&gt; void), deps?: any[]): void;
  export const Fragment: unique symbol;
  export const StrictMode: FC&lt;{ children?: ReactNode }&gt;;

  const React: {
    createElement: typeof createElement;
    Fragment: typeof Fragment;
    StrictMode: typeof StrictMode;
    useState: typeof useState;
    useEffect: typeof useEffect;
  };

  export default React;
}

declare module "react-dom/client" {
  export function createRoot(container: Element | DocumentFragment): {
    render(children: React.ReactNode): void;
  };
}
```

File: frontend\src\vite-env.d.ts

Full path: C:\Users\itodo\Hostelrec\frontend\src\vite-env.d.ts

```
interface ImportMetaEnv {
  readonly VITE_API_URL?: string;
  // add more env vars here as needed
}

interface ImportMeta {
  readonly env: ImportMetaEnv;
}
```

File: frontend\tailwind.config.js

Full path: C:\Users\itodo\Hostelrec\frontend\tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
export default {
  content: ["./index.html", "./src/**/*.{ts,tsx}"],
  theme: {
    extend: {
      colors: {
        primary: "#0F172A",
        accent: "#F59E0B",
        success: "#10B981",
        danger: "#EF4444",
        background: "#F8FAFC"
      }
    }
  },
  plugins: []
};
```

File: frontend\tsconfig.json

Full path: C:\Users\itodo\Hostelrec\frontend\tsconfig.json

```
{
  "extends": "./tsconfig.node.json",
  "compilerOptions": {
    "jsx": "react-jsx"
  },
  "include": ["src"]
}
```

File: frontend\tsconfig.node.json

Full path: C:\Users\itodo\Hostelrec\frontend\tsconfig.node.json

```
{
  "compilerOptions": {
    "target": "ES2020",
    "lib": ["ES2020", "DOM", "DOM.Iterable"],
    "module": "ESNext",
    "moduleResolution": "Node",
    "resolveJsonModule": true,
    "isolatedModules": true,
    "esModuleInterop": true,
    "skipLibCheck": true,
    "strict": true,
    "typeRoots": ["./src/types", "./node_modules/@types"]
  },
  "include": ["vite.config.ts"]
}
```

File: frontend\vite.config.ts

Full path: C:\Users\itodo\Hostelrec\frontend\vite.config.ts

```
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react";

export default defineConfig({
  plugins: [react()],
  server: {
    port: 4173,
    host: true
  },
  preview: {
    port: 4173
  }
});
```

File: generate_code_pdf.py

Full path: C:\Users\itodo\Hostelrec\generate_code_pdf.py

```
"""
Script to generate a PDF containing all source code files from the project.
Requires: pip install reportlab markdown
"""
import os
from pathlib import Path
from reportlab.lib.pagesizes import A4
from reportlab.lib.styles import getSampleStyleSheet, ParagraphStyle
from reportlab.lib.units import inch
from reportlab.platypus import SimpleDocTemplate, Paragraph, Spacer, PageBreak, Preformatted
from reportlab.lib.enums import TA_LEFT, TA_CENTER
from reportlab.pdfbase import pdfmetrics
from reportlab.pdfbase.ttfonts import TTFont
from reportlab.lib.colors import HexColor

# File extensions to include
CODE_EXTENSIONS = {
    '.py', '.tsx', '.ts', '.js', '.jsx', '.json',
    '.sql', '.md', '.yml', '.yaml', '.txt', '.css'
}

# Directories to exclude
EXCLUDE_DIRS = {
    'node_modules', '__pycache__', '.git', 'dist', 'build',
    '.next', 'venv', 'env', '.venv', 'target'
}

# Files to exclude
EXCLUDE_FILES = {
    'package-lock.json', 'yarn.lock', '.DS_Store'
}

def should_include_file(file_path: Path) -&gt; bool:
    """Check if file should be included in PDF."""
    # Check if file extension is in allowed list
    if file_path.suffix not in CODE_EXTENSIONS:
        return False

    # Check if file is in exclude list
    if file_path.name in EXCLUDE_FILES:
        return False

    # Check if any parent directory is excluded
    parts = file_path.parts
    for part in parts:
        if part in EXCLUDE_DIRS:
            return False

    return True

def get_all_code_files(root_dir: Path) -&gt; list[Path]:
    """Get all code files from the project."""
    code_files = []

    for root, dirs, files in os.walk(root_dir):
        # Filter out excluded directories
        dirs[:] = [d for d in dirs if d not in EXCLUDE_DIRS]

        for file in files:
            file_path = Path(root) / file
            if should_include_file(file_path):
                code_files.append(file_path)

    # Sort files for consistent ordering
    code_files.sort()
    return code_files

def read_file_content(file_path: Path) -&gt; str:
```

```
    """Read file content with error handling."""
    try:
        with open(file_path, 'r', encoding='utf-8') as f:
            return f.read()
    except UnicodeDecodeError:
        try:
            with open(file_path, 'r', encoding='latin-1') as f:
                return f.read()
        except Exception as e:
            return f"[Error reading file: {e}]"
    except Exception as e:
        return f"[Error reading file: {e}]"

def create_pdf(output_path: str, root_dir: Path):
    """Create PDF with all code files."""
    # Get all code files
    code_files = get_all_code_files(root_dir)

    print(f"Found {len(code_files)} code files to include in PDF...")

    # Create PDF document
    doc = SimpleDocTemplate(
        output_path,
        pagesize=A4,
        rightMargin=72,
        leftMargin=72,
        topMargin=72,
        bottomMargin=18
    )

    # Container for PDF elements
    story = []

    # Define styles
    styles = getSampleStyleSheet()

    # Title style
    title_style = ParagraphStyle(
        'CustomTitle',
        parent=styles['Heading1'],
        fontSize=24,
        textColor=HexColor('#1e40af'),
        spaceAfter=30,
        alignment=TA_CENTER
    )

    # Heading style
    heading_style = ParagraphStyle(
        'CustomHeading',
        parent=styles['Heading2'],
        fontSize=16,
        textColor=HexColor('#3b82f6'),
        spaceAfter=12,
        spaceBefore=20
    )

    # File path style
    path_style = ParagraphStyle(
        'FilePath',
        parent=styles['Normal'],
        fontSize=10,
        textColor=HexColor('#6b7280'),
        fontName='Courier',
        spaceAfter=6
    )

    # Code style
    code_style = ParagraphStyle(
        'Code',
        parent=styles['Code'],
        fontSize=8,
        fontName='Courier',
        leading=10,
```

```
        leftIndent=0,
        spaceAfter=12
    )

    # Add title page
    story.append(Paragraph("HostelRec Project", title_style))
    story.append(Paragraph("Complete Source Code Documentation", styles['Heading2']))
    story.append(Spacer(1, 0.5*inch))
    story.append(Paragraph(f"Total files: {len(code_files)}", styles['Normal']))
    story.append(Paragraph(f"Generated: {Path(output_path).stat().st_mtime if Path(output_path).exists() else 'Now'}", s
    tyles['Normal']))
    story.append(PageBreak())

    # Add table of contents
    story.append(Paragraph("Table of Contents", heading_style))
    story.append(Spacer(1, 0.2*inch))

    for i, file_path in enumerate(code_files, 1):
        rel_path = file_path.relative_to(root_dir)
        story.append(Paragraph(f"{i}. {rel_path}", path_style))

    story.append(PageBreak())

    # Add each file
    for file_path in code_files:
        rel_path = file_path.relative_to(root_dir)

        # Add file header
        story.append(Paragraph(f"File: {rel_path}", heading_style))
        story.append(Paragraph(f"Full path: {file_path}", path_style))
        story.append(Spacer(1, 0.1*inch))

        # Read and add file content
        content = read_file_content(file_path)

        # Escape special characters for PDF
        content = content.replace('&amp;', '&amp;amp;')
        content = content.replace('&lt;', '&amp;lt;')
        content = content.replace('&gt;', '&amp;gt;')

        # Split content into lines and create paragraphs
        lines = content.split('\n')

        # Use Preformatted for code blocks
        # Limit line length to prevent overflow
        code_text = '\n'.join(lines)
        # Split long lines if needed
        max_line_length = 120
        wrapped_lines = []
        for line in lines:
            if len(line) &gt; max_line_length:
                # Wrap long lines
                while len(line) &gt; max_line_length:
                    wrapped_lines.append(line[:max_line_length])
                    line = '    ' + line[max_line_length:]
                wrapped_lines.append(line)
            else:
                wrapped_lines.append(line)

        code_text = '\n'.join(wrapped_lines)
        story.append(Preformatted(code_text, code_style, maxLineLength=max_line_length))

        story.append(Spacer(1, 0.2*inch))
        story.append(PageBreak())

    # Build PDF
    print("Building PDF...")
    doc.build(story)
    print(f"PDF created successfully: {output_path}")

def main():
    """Main function."""
    # Get project root directory
```

```
    project_root = Path(__file__).parent

    # Output PDF path
    output_pdf = project_root / "HostelRec_Complete_Code.pdf"

    print(f"Project root: {project_root}")
    print(f"Output PDF: {output_pdf}")

    # Create PDF
    create_pdf(str(output_pdf), project_root)

    print(f"\nPDF generation complete!")
    print(f"Output file: {output_pdf}")
    print(f"File size: {output_pdf.stat().st_size / 1024 / 1024:.2f} MB")

if __name__ == "__main__":
    main()
```

File: README.md

Full path: C:\Users\itodo\Hostelrec\README.md

```
# HostelRec - Hostel Management System

A full-stack hostel management system built with FastAPI backend and React frontend, featuring a comprehensi
    ation and authorization system.

## ■ Table of Contents

- [Overview](#overview)
- [Features](#features)
- [Project Structure](#project-structure)
- [Getting Started](#getting-started)
- [Authentication System](#authentication-system)
- [API Documentation](#api-documentation)
- [Security Features](#security-features)
- [Technology Stack](#technology-stack)
- [Development](#development)

## ■ Overview

HostelRec is a modern hostel management system that provides:
- Room booking and management
- Guest check-in/check-out
- Staff management
- Order management (food &amp;amp; drinks)
- Customer portal
- Comprehensive authentication and authorization system

## ■ Features

### Core Features
- **Room Management**: Browse, book, and manage hostel rooms
- **Guest Management**: Check-in/check-out functionality
- **Order System**: Food and drink ordering for guests
- **Staff Dashboard**: Administrative tools for staff members
- **Customer Portal**: Self-service portal for guests
- **Multi-language Support**: English and Russian

### Authentication &amp;amp; Security Features
- **User Authentication**: Secure login with JWT tokens
- **Password Hashing**: pbkdf2_sha256 with automatic salt generation
- **Session Management**: Track and manage user sessions
- **Multi-Factor Authentication (MFA)**: Optional two-factor authentication via email
- **Password Recovery**: Secure password reset functionality
- **Role-Based Access Control**: Different access levels (customer, receptionist, manager, cleaner)
- **Session Tracking**: Monitor active sessions with IP and device information

## ■ Project Structure

```
HostelRec/
■■■ backend/
■   ■■■ app/
■   ■   ■■■ api/
■   ■   ■   ■■■ routes/
■   ■   ■   ■■■ auth.py   # Authentication endpoints
■   ■   ■■■ core/
■   ■   ■   ■■■ security.py   # Security utilities (hashing, JWT)
■   ■   ■■■ models/
■   ■   ■   ■■■ user.py   # User model
■   ■   ■   ■■■ session.py   # Session model
■   ■   ■■■ schemas/
■   ■   ■   ■■■ auth.py   # Pydantic schemas
■   ■   ■■■ services/
■   ■   ■   ■■■ session_service.py   # Session management
■   ■   ■   ■■■ mfa_service.py   # MFA functionality
■   ■   ■   ■■■ password_reset_service.py # Password recovery
■   ■   ■■■ main.py   # FastAPI application
■   ■■■ requirements.txt
```
```

```
■   ■■■ Dockerfile
■■■ frontend/
■   ■■■ src/
■   ■   ■■■ pages/
■   ■   ■   ■■■ LoginPage.tsx
■   ■   ■   ■■■ SignupPage.tsx
■   ■   ■   ■■■ PasswordChangePage.tsx
■   ■   ■   ■■■ MFAPage.tsx
■   ■   ■   ■■■ SessionsPage.tsx
■   ■   ■   ■■■ ForgotPasswordPage.tsx
■   ■   ■■■ components/
■   ■   ■■■ hooks/
■   ■   ■   ■■■ useAuthStore.ts
■   ■   ■■■ services/
■   ■       ■■■ api.ts
■   ■■■ package.json
■   ■■■ Dockerfile
■■■ sql/                              # Database scripts
■■■ docker-compose.yml
■■■ README.md
```

■ Getting Started

Prerequisites

- Docker and Docker Compose
- Git

Installation

1. **Clone the repository**
   ```bash
   git clone &lt;repository-url&gt;
   cd Hostelrec
   ```

2. **Start the application**
   ```bash
   docker-compose up --build
   ```

3. **Access the application**
   - Frontend: http://localhost:5173
   - Backend API: http://localhost:8000
   - API Documentation: http://localhost:8000/docs
   - Database: localhost:5432

Default Credentials

The system automatically creates a default manager account on first startup. Check the backend logs for cred

■ Authentication System

Overview

The authentication system implements all requirements from Lab Work #6:
- User database with roles
- Password hashing with salt
- JWT token-based authentication
- Role-based authorization
- Session management
- Multi-factor authentication (MFA)
- Password recovery

User Roles

- **customer**: Regular guest users
- **receptionist**: Front desk staff
- **manager**: Administrative staff
- **cleaner**: Cleaning staff

Password Security

- **Algorithm**: pbkdf2_sha256
- **Salt**: Automatically generated for each password
- **Storage**: Only hashed passwords are stored in the database

Session Management

- Sessions are tracked in the `user_sessions` table
- Each session includes:
  - IP address
  - User agent (device/browser info)
  - Creation time
  - Last activity time
  - Expiration time
- Users can view and manage their active sessions

■ API Documentation

Authentication Endpoints

Public Endpoints

POST `/auth/token` - Login
```json
// Request (form-data)
username: "user@example.com"
password: "password123"

// Response
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
  "token_type": "bearer",
  "role": "customer",
  "mfa_required": false
}
```

POST `/auth/signup` - Register new user
```json
// Request
{
  "email": "user@example.com",
  "full_name": "John Doe",
  "password": "password123"
}

// Response
{
  "access_token": "...",
  "token_type": "bearer",
  "role": "customer"
}
```

POST `/auth/forgot-password` - Request password reset
```json
// Request
{
  "email": "user@example.com"
}

// Response
{
  "message": "If an account with this email exists, a password reset code has been sent."
}
```

POST `/auth/reset-password` - Reset password with code
```json
// Request
{
  "email": "user@example.com",
  "code": "ABC12345",
  "new_password": "newpassword123"
```

```
}

// Response
{
  "message": "Password reset successfully. Please login with your new password."
}
```

Protected Endpoints (Require Authentication)

POST `/auth/change-password` - Change password
```json
// Request
{
  "old_password": "oldpassword",
  "new_password": "newpassword123"
}

// Response
{
  "message": "Password changed successfully. Please login again."
}
```

POST `/auth/logout` - Logout
```json
// Response
{
  "message": "Logged out successfully"
}
```

GET `/auth/me` - Get current user info
```json
// Response
{
  "id": 1,
  "username": "user@example.com",
  "role": "customer",
  "mfa_enabled": false,
  "active_sessions_count": 2
}
```

GET `/auth/sessions` - Get active sessions
```json
// Response
[
  {
    "id": 1,
    "created_at": "2025-12-08T17:00:00Z",
    "last_activity": "2025-12-08T18:00:00Z",
    "expires_at": "2025-12-08T19:00:00Z",
    "ip_address": "192.168.1.1",
    "user_agent": "Mozilla/5.0...",
    "is_current": true
  }
]
```

DELETE `/auth/sessions/{session_id}` - End specific session
```json
// Response
{
  "message": "Session ended successfully"
}
```

MFA Endpoints

POST `/auth/mfa/enable` - Enable MFA
```json
// Request
```

```
{
  "password": "userpassword"
}

// Response
{
  "message": "MFA enabled successfully",
  "secret": "secret_key_here"
}
```

POST `/auth/mfa/verify` - Verify MFA code
```json
// Request
{
  "code": "123456"
}

// Response
{
  "message": "MFA code verified successfully"
}
```

POST `/auth/mfa/disable` - Disable MFA
```json
// Request
{
  "password": "userpassword"
}

// Response
{
  "message": "MFA disabled successfully"
}
```

Using the API

All protected endpoints require a Bearer token in the Authorization header:

```bash
curl -H "Authorization: Bearer YOUR_JWT_TOKEN" \
     http://localhost:8000/auth/me
```

■ Security Features

Password Security
- Passwords are hashed using pbkdf2_sha256
- Each password gets a unique salt automatically
- Passwords are never stored in plain text
- Minimum password length: 6 characters

Token Security
- JWT tokens with expiration (default: 60 minutes)
- Tokens are signed with a secret key
- Tokens include user role for authorization

Session Security
- Sessions are tracked with IP and device information
- Users can view all active sessions
- Users can terminate specific sessions
- All sessions are terminated when password is changed

Multi-Factor Authentication
- Optional MFA via email codes
- 6-digit codes with 10-minute expiration
- Codes are single-use only

Password Recovery
- Secure password reset via email codes
- 8-character alphanumeric codes

- 30-minute expiration
- Codes are single-use only
- All sessions terminated after password reset

Security Best Practices
- Rate limiting recommended for production
- HTTPS required in production
- Secret keys should be stored in environment variables
- Email service integration needed for production MFA

■ Technology Stack

Backend
- **FastAPI** 0.111.0 - Modern Python web framework
- **SQLAlchemy** 2.0.30 - ORM with async support
- **PostgreSQL** 15 - Database
- **asyncpg** 0.29.0 - Async PostgreSQL driver
- **python-jose** 3.3.0 - JWT handling
- **passlib** 1.7.4 - Password hashing

Frontend
- **React** 18.2.0 - UI library
- **TypeScript** 5.4.5 - Type safety
- **Vite** 5.2.12 - Build tool
- **Tailwind CSS** 3.4.4 - Styling
- **Framer Motion** 11.2.9 - Animations
- **React Router** 6.23.1 - Routing
- **Zustand** 4.5.5 - State management
- **i18next** 23.11.5 - Internationalization
- **Axios** 1.7.2 - HTTP client

■ Database Schema

Users Table (`hostel_users`)
```sql
CREATE TABLE hostel_users (
    id SERIAL PRIMARY KEY,
    username VARCHAR(255) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    role VARCHAR(50) NOT NULL,
    mfa_enabled BOOLEAN DEFAULT false,
    mfa_secret VARCHAR(255)
);
```

Sessions Table (`user_sessions`)
```sql
CREATE TABLE user_sessions (
    id SERIAL PRIMARY KEY,
    user_id INTEGER NOT NULL REFERENCES hostel_users(id) ON DELETE CASCADE,
    token VARCHAR(500) UNIQUE NOT NULL,
    created_at TIMESTAMPTZ NOT NULL DEFAULT now(),
    last_activity TIMESTAMPTZ NOT NULL DEFAULT now(),
    expires_at TIMESTAMPTZ NOT NULL,
    is_active BOOLEAN DEFAULT true,
    ip_address VARCHAR(45),
    user_agent TEXT
);
```

■ Testing

Manual Testing

1. **Test Authentication Flow**
   ```bash
   # Register a new user
   curl -X POST http://localhost:8000/auth/signup \
     -H "Content-Type: application/json" \
     -d '{"email":"test@example.com","full_name":"Test User","password":"test123"}'

   # Login
   curl -X POST http://localhost:8000/auth/token \
   ```

```
     -H "Content-Type: application/x-www-form-urlencoded" \
     -d "username=test@example.com&amp;password=test123"
   ```

2. **Test Password Change**
   ```bash
   curl -X POST http://localhost:8000/auth/change-password \
     -H "Authorization: Bearer YOUR_TOKEN" \
     -H "Content-Type: application/json" \
     -d '{"old_password":"test123","new_password":"newpass123"}'
   ```

3. **Test Password Recovery**
   ```bash
   # Request reset code
   curl -X POST http://localhost:8000/auth/forgot-password \
     -H "Content-Type: application/json" \
     -d '{"email":"test@example.com"}'

   # Reset password (check console for code)
   curl -X POST http://localhost:8000/auth/reset-password \
     -H "Content-Type: application/json" \
     -d '{"email":"test@example.com","code":"ABC12345","new_password":"newpass123"}'
   ```

■ Environment Variables

Backend
```env
DATABASE_URL=postgresql+asyncpg://postgres:postgres@db:5432/hostelrec
AUTH_SECRET_KEY=your-secret-key-here
AUTH_TOKEN_EXPIRE_MINUTES=60
AUTH_ALGORITHM=HS256
SUPER_SECRET_PHRASE=your-secret-phrase
```

Frontend
```env
VITE_API_URL=http://localhost:8000
```

■ Development

Running in Development Mode

Backend:
```bash
cd backend
pip install -r requirements.txt
uvicorn app.main:app --reload --host 0.0.0.0 --port 8000
```

Frontend:
```bash
cd frontend
npm install
npm run dev
```

Database Migrations

The application automatically creates necessary tables on startup. For production, consider using Alembic fo
    rations.

■ Additional Documentation

- [Authentication System Documentation](./backend/AUTHENTICATION_SYSTEM.md) - Detailed authentication system
    on (in Russian)
- [Database Relationships](./DATABASE_RELATIONSHIPS.md) - Database schema relationships

■ Security Recommendations for Production

1. **Use HTTPS**: Always use HTTPS in production

2. **Change Secret Keys**: Update `AUTH_SECRET_KEY` and `SUPER_SECRET_PHRASE`
3. **Email Service**: Integrate real email service (SendGrid, AWS SES) for MFA and password recovery
4. **Rate Limiting**: Implement rate limiting for login and password reset endpoints
5. **Redis**: Use Redis for session storage instead of in-memory storage
6. **Monitoring**: Add logging and monitoring for security events
7. **CORS**: Configure CORS properly for production domains
8. **Input Validation**: Ensure all inputs are properly validated
9. **SQL Injection**: Use parameterized queries (already implemented via SQLAlchemy)
10. **XSS Protection**: Frontend should sanitize user inputs

■ License

This project is part of a coursework assignment.

■ Authors

Developed as part of Lab Work #6 - Authentication and Authorization Systems.

■ Acknowledgments

- FastAPI documentation
- React documentation
- PostgreSQL documentation

File: sql\README.md

Full path: C:\Users\itodo\Hostelrec\sql\README.md

```
Place your Lab 1-3 SQL scripts here (schema, views, triggers, seed data).
These files are mounted into the Postgres container via docker-compose.
```