

Tamiti Studio Backend Setup Documentation

Stack

- **Framework:** Django 5.1, Django REST Framework
 - **Auth:** SimpleJWT (with rotation + blacklist)
 - **Database:** SQLite (dev), PostgreSQL (prod ready)
 - **Email:** Console backend (dev), pluggable SMTP (prod)
 - **Test Suite:** Pytest + FactoryBoy
 - **Permissions:** Role-based (custom DRF permissions)
-

Initial Setup

Apps Created

- `core`: Abstract base models (`BaseModel`, `SingletonBaseModel`)
- `users`: Custom user model with roles, proxy classes
- `accounts`: `StaffProfile`, `CustomerProfile`, `Department`, `Designation`, `Branch`

Core Model Structure

```
# users.models.User
- username (login field)
- email (unique, used for verification)
- phone (optional)
- is_verified
- role: Admin | Staff | Customer

# accounts.models
- StaffProfile: links User to Branch, Designation, Department
- CustomerProfile: links User to Referral
```

settings.py Highlights

Installed Apps

```
INSTALLED_APPS = [
    'rest_framework',
    'rest_framework_simplejwt',
    'rest_framework_simplejwt.token_blacklist',
```

```
'core', 'users', 'accounts',  
]
```

JWT Settings

```
SIMPLE_JWT = {  
    "ROTATE_REFRESH_TOKENS": True,  
    "BLACKLIST_AFTER_ROTATION": True,  
    "ACCESS_TOKEN_LIFETIME": timedelta(minutes=15),  
    "REFRESH_TOKEN_LIFETIME": timedelta(days=1),  
    "AUTH_HEADER_TYPES": ("Bearer",),  
}
```


Environment Split

Settings refactored to:

```
tamiti_studio/settings/  
├─ base.py  
├─ dev.py  
└─ prod.py
```

With `.env` support using `django-environ`


Auth Endpoints Implemented

 `/api/auth/register/`

Registers a user with username, email, phone, password. Sends verification email.

 `/api/auth/login/`


Accepts username/password. Returns JWT tokens.

 `/api/auth/verify-email/?uid=...&token=...`

Verifies email using a secure token + encoded UID

 `/api/auth/password-reset-request/`

Accepts email, sends reset link

 `/api/auth/password-reset-confirm/`

Accepts uid, token, new password — resets user password

Email Token Logic

Used Django's `PasswordResetTokenGenerator` + `urlsafe_base64` encoding. Encapsulated in:

- `users/tokens.py`
 - `users/utils.py`
-

Permissions

Custom DRF classes in `core/permissions.py`

```
class IsAdmin(BasePermission)
class IsStaff(BasePermission)
class IsCustomer(BasePermission)
class IsAdminOrStaff(BasePermission)
class IsOwnerOrReadOnly(BasePermission)
```

Use via `permission_classes = [IsAuthenticated, IsAdminOrStaff]`

Management Commands

Seed HR Data:

```
python manage.py seed_accounts
```

Creates:

- Branch: Headoffice
 - Departments: Finance, Projects, Digital, etc.
 - Designations: Manager, Officer, etc.
-

Tests

Using `pytest` + `factory_boy`

Structure:

```
tests/  
├─ factories.py      # UserFactory  
├─ test_auth.py      # Covers register/login/reset/verify
```

Example Test

```
@pytest.mark.django_db  
def test_register_user():  
    response = client.post('/api/auth/register/', {...})  
    assert response.status_code == 201
```

Ready For Next Sprint

- Project & Task Viewsets
- Role-protected dashboards
- Field agent + lead tracker

This concludes the initial phase: user auth, base models, permissions, and test suite.