Rent Near TU

Abstract

Dormitories are essential for quality of life, yet students, workers, and low-to-middle-income families near Thammasat University's Rangsit Campus face barriers in finding affordable, suitable housing. Fragmented systems and inefficient methods, like physical visits or word-of-mouth, often waste time and money. **Rent Near TU**, a Django-based web application, addresses this gap by providing a centralized platform for tenants (students, workers) and landlords (property managers). With budget-based search, advanced filtering by amenities and location, multimedia-rich listings, and real-time management tools, it simplifies dormitory searches, reduces access disparities, and supports educational and professional opportunities. Its scalable design offers potential for broader community applications.

Demo Video

Watch our YouTube video to see Rent Near TU in action, showcasing how students, workers, and landlords use the platform: https://youtu.be/3xOHowoUYAo?si=SKnFD9qQQDiwgi8u Watch the Demo

---

User Stories

The following user stories reflect key usage scenarios:

1. **As a Thammasat University student**, I want to find a budget-friendly dormitory near campus with amenities like air conditioning and Wi-Fi, so I can focus on my studies without financial strain.
2. **As a worker near Rangsit**, I want to search for a dormitory close to my workplace with complete furniture and flexible lease terms, so I can save commuting time and live comfortably.
3. **As a landlord**, I want to list my dormitory rooms with photos and detailed descriptions, so I can attract reliable tenants and manage bookings efficiently.

Usage Scenarios

Rent Near TU supports these user stories through tailored workflows:

- **Thammasat University Student**:
    - **Search**: Filter dorms by proximity to campus, price, and amenities (e.g., Wi-Fi, air conditioning) on the room browsing page.
    - **Book**: Select a room, enter check-in/check-out dates, and provide details (name, phone, email) via the booking form.
    - **Review**: Post-stay, submit a 1-5 star rating and feedback on cleanliness and comfort.
- **Worker near Rangsit**:
    - **Filter**: Browse dorms by workplace proximity, furniture availability (e.g., beds, tables), and lease flexibility.

  - **Book**: Confirm a booking with real-time availability checks and status updates (pending, confirmed).
  - **Preferences**: Update tenant profile with budget and furniture needs for tailored suggestions.

- **Landlord**:

  - **List Rooms**: Create/edit listings via the landlord dashboard, adding `dorm_name`, `price`, `location`, and images.
  - **Manage Bookings**: Approve or cancel bookings, with automatic availability updates.
  - **View Reviews**: Monitor tenant feedback to enhance listings.

UX/UI Design

A user-centered design process ensures usability:

- **User Research**: Interviewed students and workers to identify challenges, like fragmented dormitory information and high search costs.
- **Personas**: Developed personas, including a student prioritizing affordability and a worker needing proximity to Rangsit.
- **Wireframes**: Created layouts for intuitive navigation, with filters for `location`, `price`, and amenities.
- **Prototypes**: Tested booking and listing flows to ensure ease of use for tenants and landlords.
- **UI Design**: Designed a responsive Django template interface, emphasizing room images (via `Images` or `logo.svg`) and clear call-to-action buttons.

Features

- **User Roles**:
  - **Tenants**: Browse, book, review, and set preferences.
  - **Landlords**: Manage listings (price, amenities, images).
- **Room Management**: Create/edit listings with `dorm_name`, `room_name`, `price`, `location`, amenities (`bed_count`, `aircon_count`), and `Images`.
- **Booking System**: Book with check-in/check-out dates, track `status`, and manage availability.
- **Reviews**: Rate rooms (1-5 stars) with comments. Custom Authentication: Extends `AbstractUser` with `phone` and `role`.

Technologies Used

- **Backend**: Django 4.x, Python 3.13
- **Database**: SQLite (configurable to PostgreSQL)
- **Frontend**: Django templates (extendable with CSS, JavaScript, images)
- **Static Files**: Supports images (e.g., `logo.svg`, room photos)
- **Image Processing**: Pillow for `ImageField`

Installation

Prerequisites

- Python 3.13+
- pip
- Virtualenv (recommended)
- Git
- Pillow (for `ImageField`)

Setup Instructions

1. **Clone the Repository**:

```
git clone https://github.com/lynylany/dsi202_2025.git
cd myproject
```

2. **Create a Virtual Environment**:

```
python -m venv venv
source venv/bin/activate  # Windows: venv\Scripts\activate
```

3. **Install Dependencies**:

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

Note: Ensure `requirements.txt` includes:

```
django>=4.0
python-dateutil>=2.8
pillow>=9.0
```

4. **Apply Migrations**:

```
python manage.py makemigrations
python manage.py migrate
```

5. **Create a Superuser**:

```
python manage.py createsuperuser
```

6. **Collect Static Files**:

```
python manage.py collectstatic
```

7. **Run the Server**:

```
python manage.py runserver
```

Access at `http://localhost:8000`.

Project Structure

```
myproject/
├── myapp/
│   ├── migrations/
│   ├── static/
│   │   └── images/
│   │       └── logo.svg
│   ├── templates/
│   ├── models.py
│   ├── views.py
│   ├── urls.py
│   └── allauth_forms.py
├── myproject/
│   ├── settings.py
│   ├── urls.py
├── manage.py
└── README.md
```

Usage

- **Admin Panel**: Access `/admin` to manage users, rooms, bookings, reviews.
- **Tenants**:
    - Filter rooms by `location`, `price`, amenities.
    - Book with check-in/check-out dates and guest details.
    - Submit reviews with ratings/comments.
- **Landlords**:
    - Add/edit listings with `price`, amenities, images.
    - Manage bookings and view reviews.

- **Static Files**: Store images in `myapp/static/images/` and use `{% load static %}` with {% s`tatic 'images/logo.svg' %}`.

Database Models

- **CustomUser**: Extends `AbstractUser` with `phone`, `role`.
- **Tenant**: Stores `budget`, `preferences`.
- **Landlord**: Includes `dorm_name`, `address`, bank details.
- **Room**: Has `dorm_name`, `room_name`, `price`, `location`, `bed_count`, `aircon_count`, `Images`.
- **Booking**: Manages `check_in`, `check_out`, `status`.
- **Review**: Stores ratings, comments.

Troubleshooting

- **ImportError for CustomSignupForm**:
  - Verify `myapp/allauth_forms.py`:

```
from allauth.account.forms import SignupForm

class CustomSignupForm(SignupForm):
    def save(self, request):
        user = super().save(request)
        return user
```

- Or remove from `myapp/views.py`:

```
# Remove: from myapp.allauth_forms import CustomSignupForm
```

- **Migration Issues**:

  - For renamed fields (e.g., `room.name` to `room.dorm_name`), answer `y` or `N`.
  - Add `default` in `models.py` (e.g., `default='Unknown'`).

- **Static Files**:

  - Configure `settings.py`:

```
STATIC_URL = '/static/'
STATICFILES_DIRS = [BASE_DIR / "static"]
```

- Run `collectstatic`.

- **Image Uploads: Install Pillow** (`pip install pillow`).

Contributing

1. Fork the repository.
2. Create a feature branch (`git checkout -b feature/YourFeature`).
3. Commit changes (`git commit -m 'Add YourFeature'`).
4. Push to the branch (`git push origin feature/YourFeature`).
5. Open a pull request.

Contact

For issues, open a ticket at `https://github.com/lynylany/dsi202_2025` or contact the maintainer.