ScholarHomes

By Ryan Shannon & Jomi Kafi

Our Ambition for The Project - Jomi

SchlorHomes project is to revolutionize student accommodation by providing a user-friendly platform that not only simplifies the search process but ensures students have access to safe, affordable, and well-located housing.

Our goal is to alleviate one of the major stresses faced by students today by offering a reliable and efficient tool that aids them in quickly finding their ideal home close to their educational institutions.

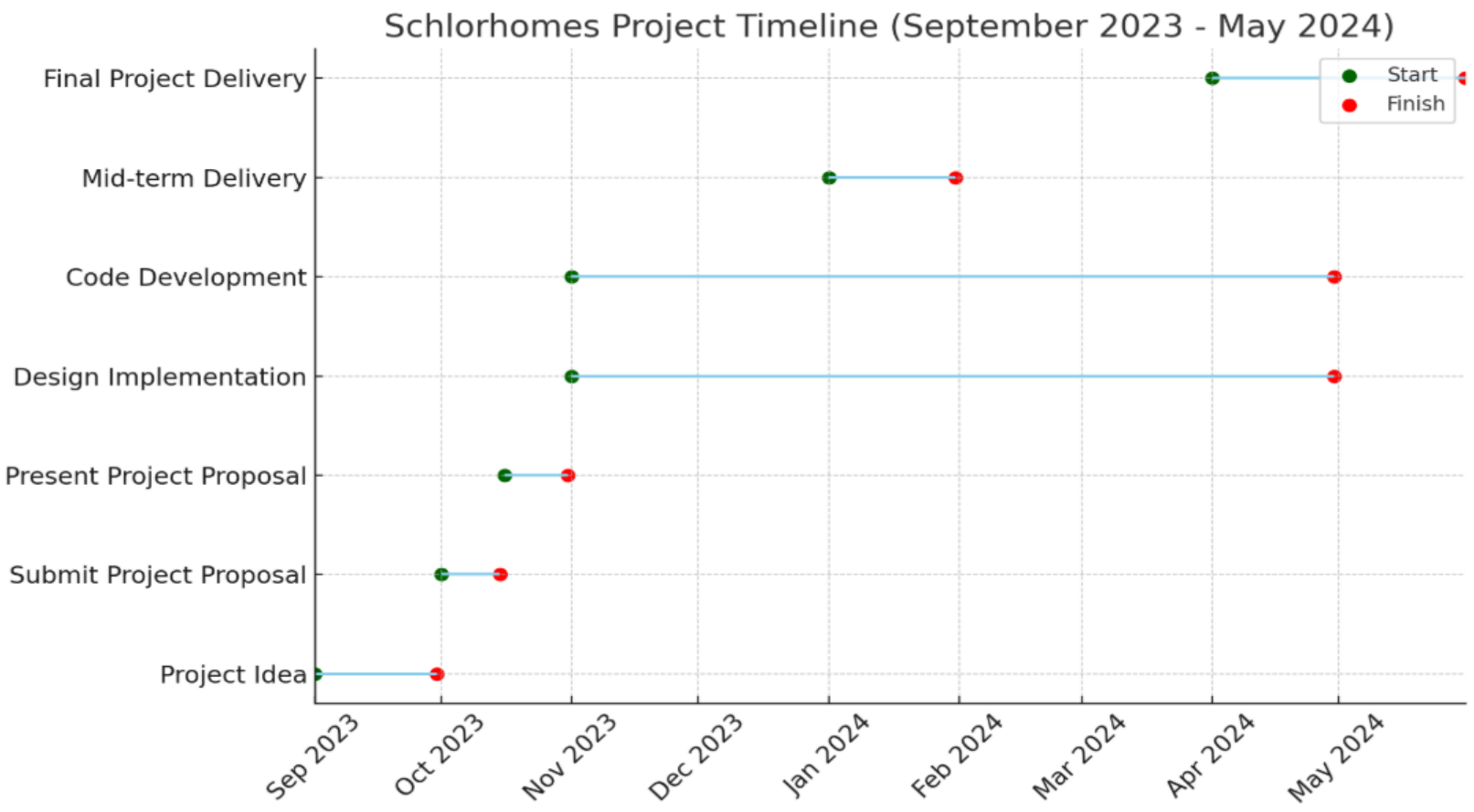
Our Solution - jomi

- **Student-Centric Housing Solutions:** SchlorHomes is dedicated to helping students find the best and most affordable accommodation options near their educational institutions.

- **Optimized Search Features:** The app provides tailored search functionalities that enable students to filter and select housing based on their specific needs and budget.

- **Accessibility and Affordability:** Focused on affordability, SchlorHomes ensures that the housing options available are within the financial reach of students, making it easier to secure convenient and cost-effective living spaces.

Project Timeline (e.g. Gantt) - Jomi



Primary Research - Ryan

We conducted primary research by asking 9 university students who have previously lived in student accommodation about their experiences

- **6 out of 9** faced difficulty securing accommodation
- We interviewed both **Irish** and **International students such Gemma**
- Gemma was one of several students who stated they had a **fear of living with a stranger**

Secondary Research - Ryan

There is a clear lack of student accommodation in Ireland:

- **Protests** about the lack of student accommodation (4th of October 2023, 300 students participants)

- The average rent in Ireland surpasses the peak of the Celtic Tiger era by more than 50%

Market - Ryan

Information gotten from **HEA.ie**

- **Main Demographic**
  Students aged 18 – 30
- **Customer Channel**
  Social Media, Student organizations, Online advertisements
- **Target Market**
  Made up of 202,100
  (demand – current accommodation available)

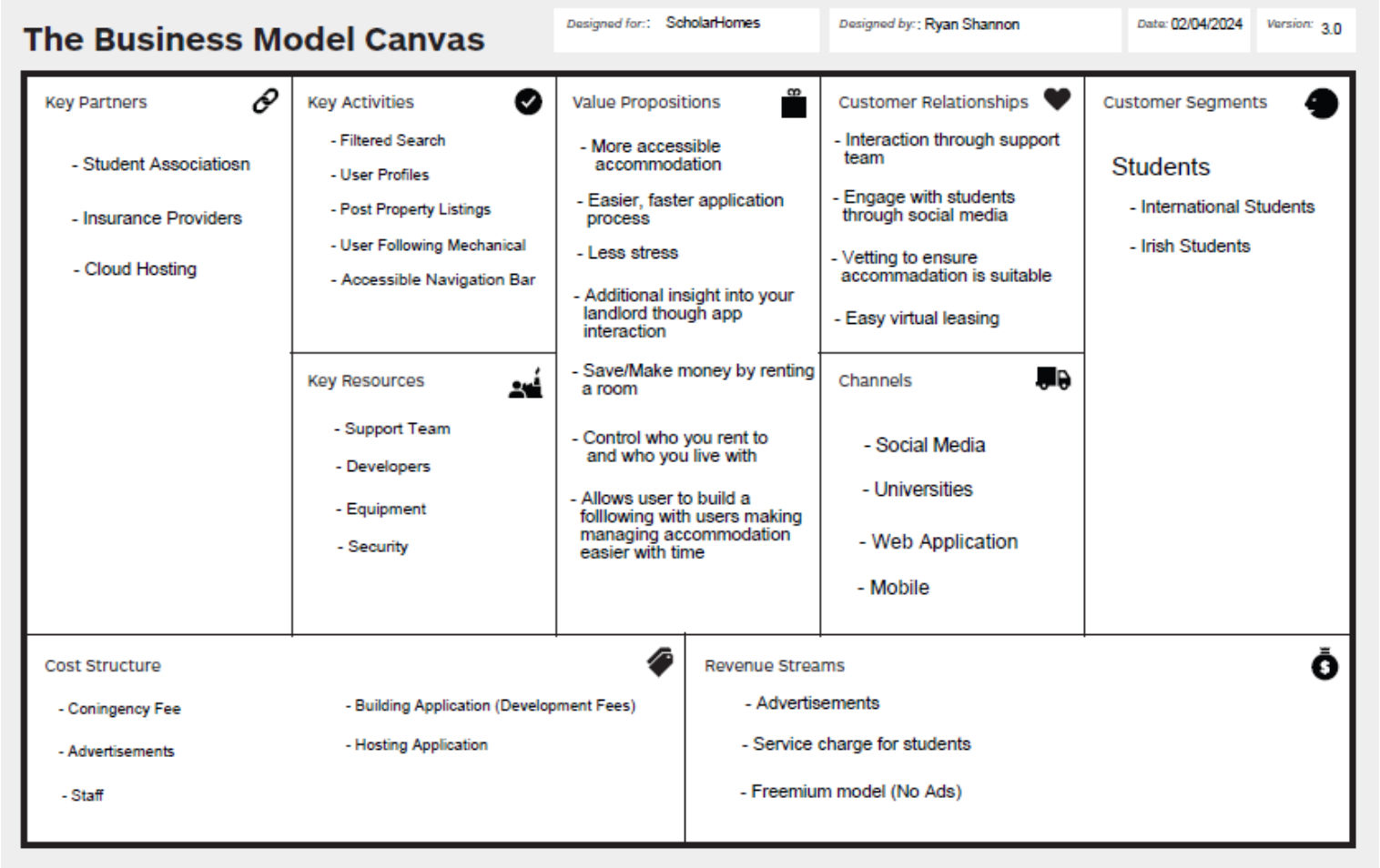
BMC - Ryan

Potential Revenue

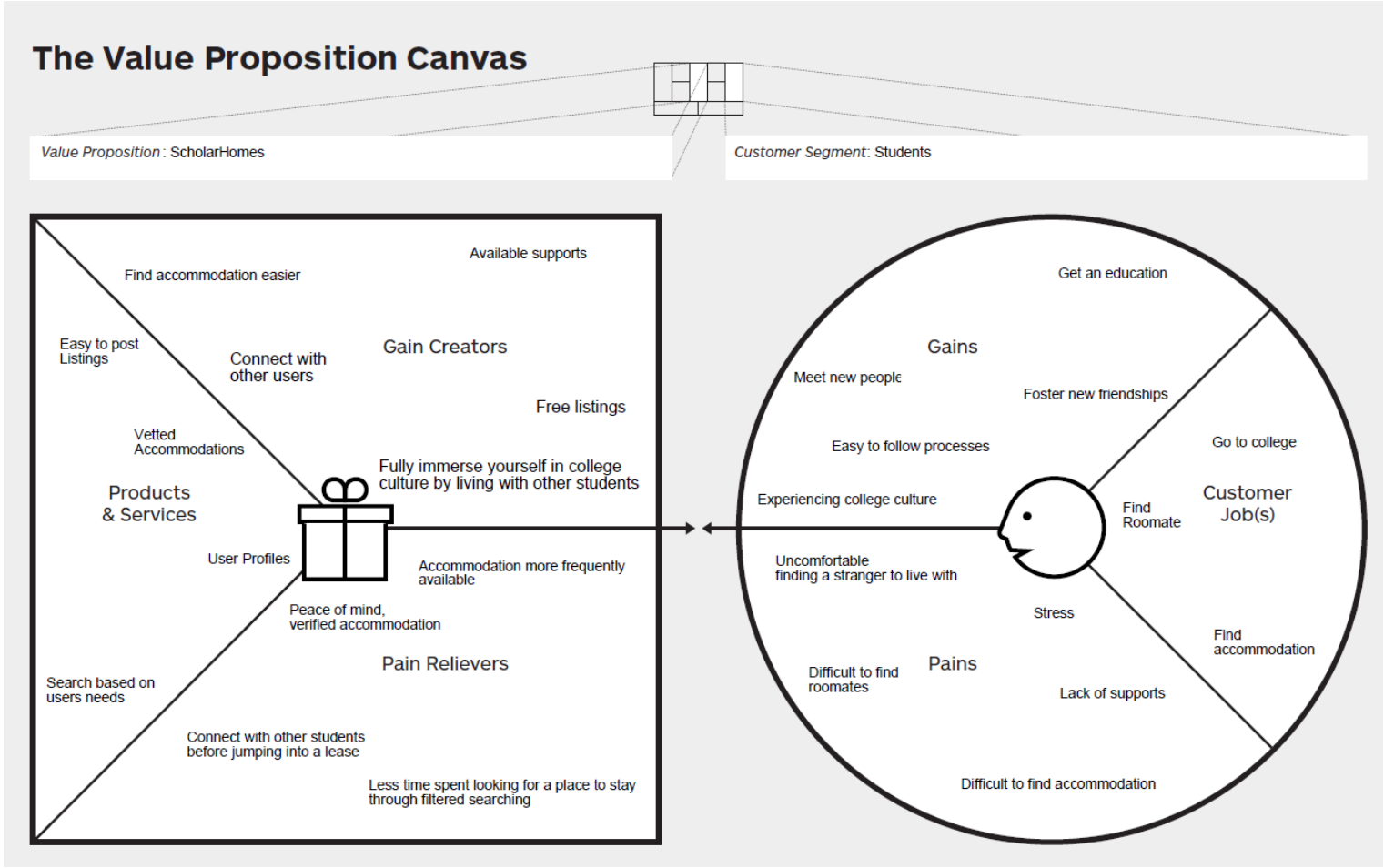
- Advertisements
- Service Charge
- Freemium model

Potential Cost

- Marketing
- Staff
- Development
- Contingency
- Cloud hosting fee



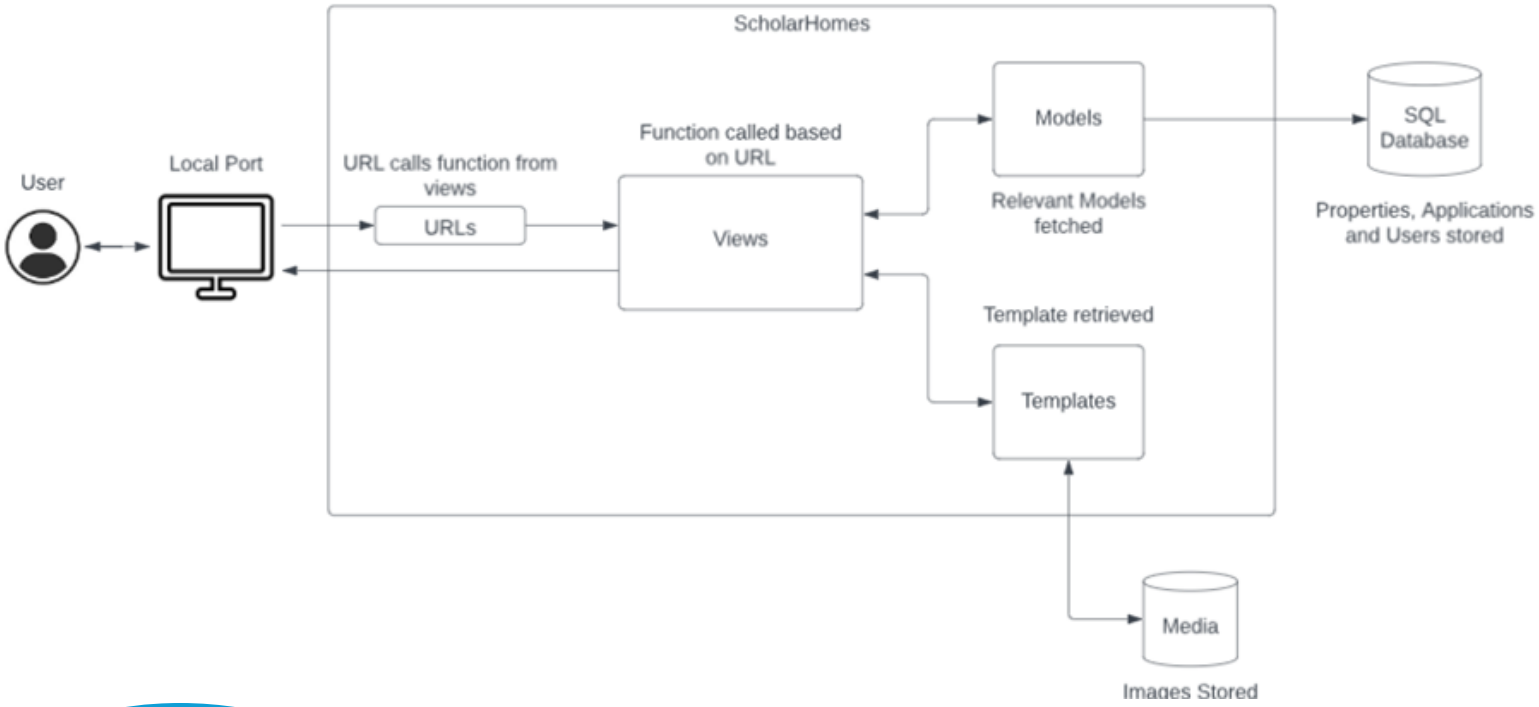
VPC – jomi


The Value Proposition Canvas
Value Proposition: ScholarHomes
Customer Segment: Students
Available supports
Find accommodation easier
Easy to post Listings
Connect with other users
Gain Creators
Free listings
Vetted Accommodations
Fully immerse yourself in college culture by living with other students
Products & Services
User Profiles
Accommodation more frequently available
Peace of mind, verified accommodation
Pain Relievers
Search based on users needs
Connect with other students before jumping into a lease
Less time spent looking for a place to stay through filtered searching
Get an education
Gains
Meet new people
Foster new friendships
Easy to follow processes
Go to college
Experiencing college culture
Find Roomate
Customer Job(s)
Uncomfortable finding a stranger to live with
Stress
Find accommodation
Pains
Difficult to find roomates
Lack of supports
Difficult to find accommodation

Competitor Analysis - Jomi

	DCUaccommodation	Daft.ie	Rent.ie	ScholarHomes
Accessibility				X
Long-Term viability	X	X		X
Chat features		X		X
Personal user profiles			X	X
Free Listings	X			X
Mobile compatibility			X	X

Django Framework - Ryan



We utilized the Django framework to create the application

Frontend

- Html
- CSS

Backend

- Python
- Java

External Interfaces - Ryan

Summary

The google map API allows ScholarHomes to display an embedded google map into our web page from an address

Requirements

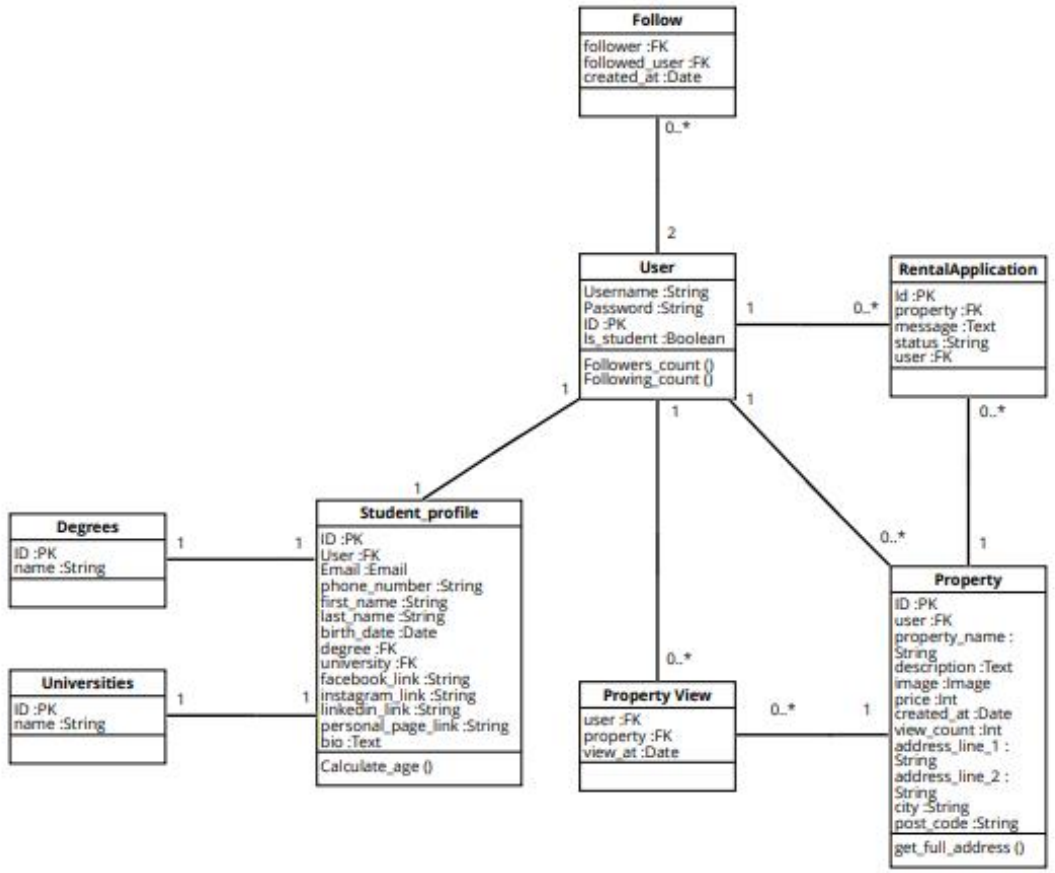
In order to use this API, we must first enable the APIs we wish to use on Google cloud and then create an API key

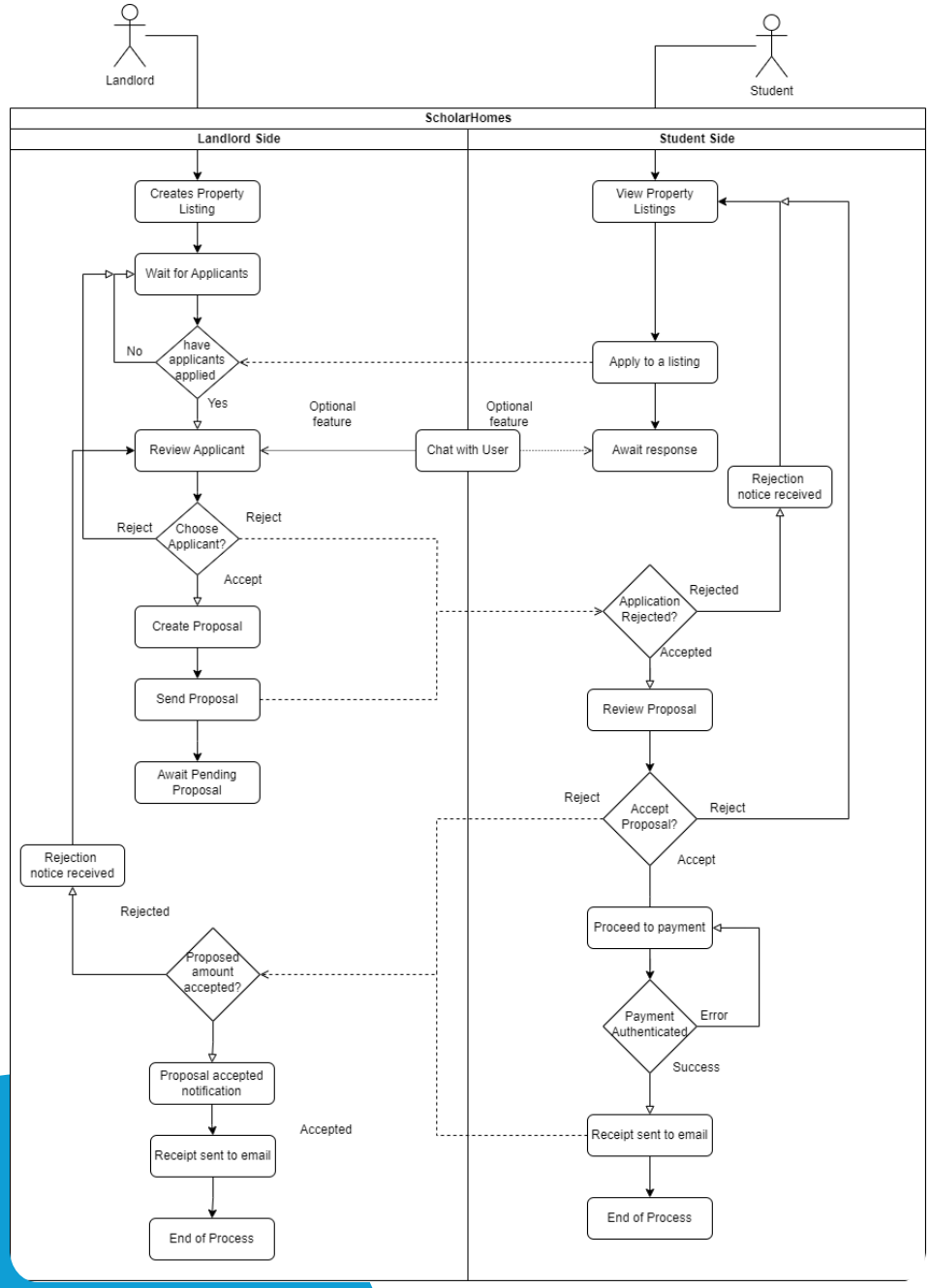
```
<!-- If there's an address, display google map -->
{% if property.address_line_1 %}
<h2>Find Us On Google Maps:</h2>
<iframe
    width="600"
    height="450"
    frameborder="0"
    style="border:0"
    src="https://www.google.com/maps/embed/v1/place?q={{ property.
    allowfullscreen>
</iframe>
{% endif %}
```

Name
Directions API
Geocoding API
Maps Embed API
Maps JavaScript API
Places API

Database Design - Ryan

ScholarHomes Database UML





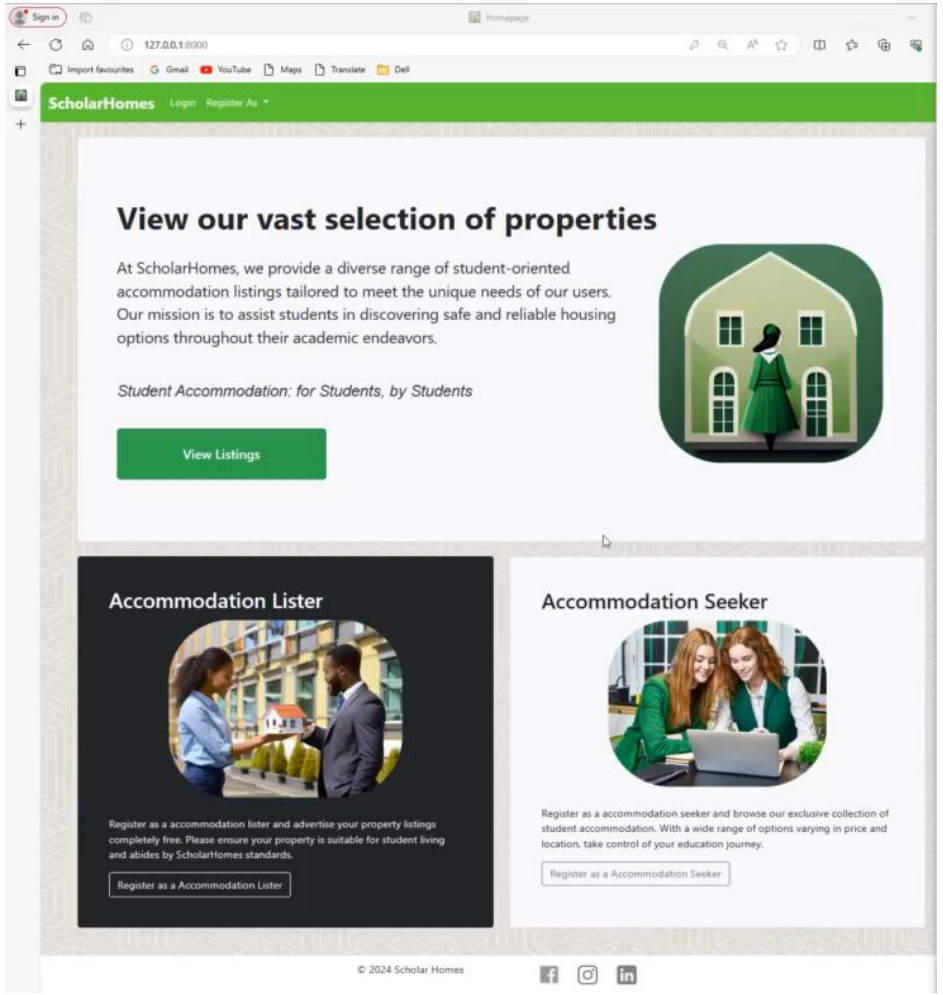
BPMN

Process we're trying to create - Ryan

1. **Lister**: Create a listing
2. **Seeker**: View property listings
3. **Seeker**: Apply to listing
4. **Lister**: Look through potential applicants
5. **Lister**: Accept and notify client

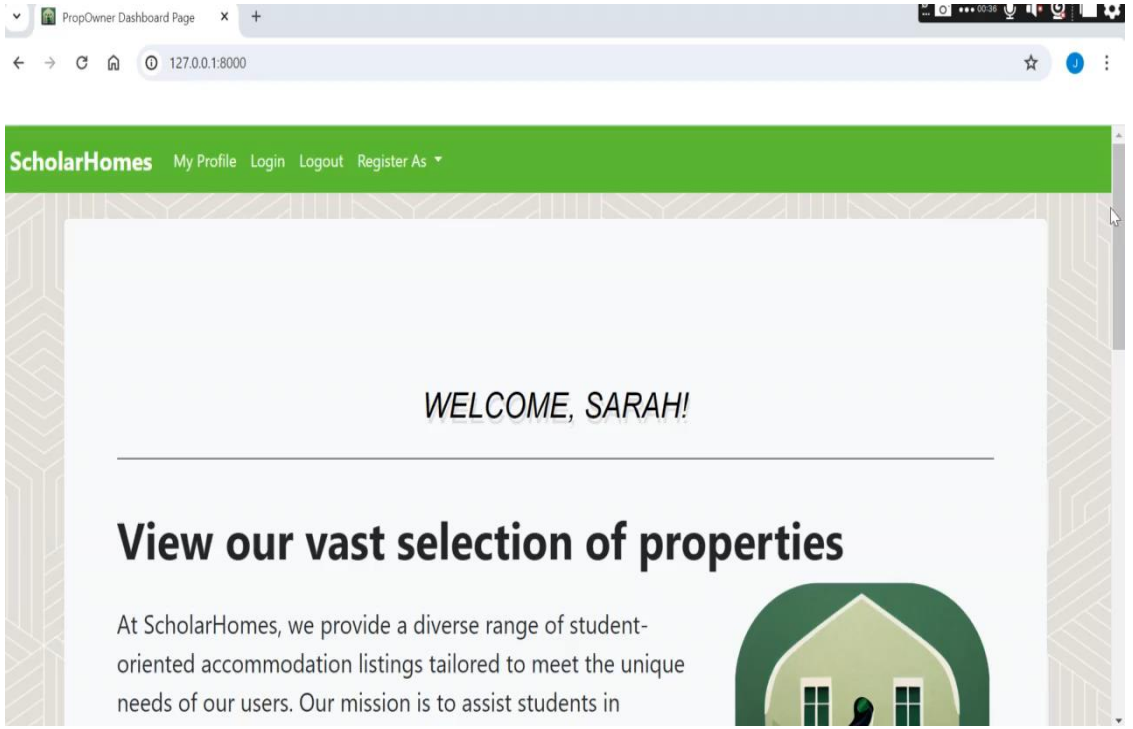
Registering Student - Ryan

```python
def PropListerSignupView(request):
    #if user submits
    if request.method == 'POST':
        user_form = StudentSignupform(request.POST)
        profile_form = StudentProfileform(request.POST, request.FILES)#REQUE

        #if both user and profile forms are valid
        if user_form.is_valid() and profile_form.is_valid():
            user = user_form.save()#save form details as user variable
            user.is_student = False #set student to Lister
            user.save()  #save user variable to db

            profile = profile_form.save(commit=False) #save form details as
            profile.user = user # set user as profile owner
            profile.save() #save user variable to db

            return redirect('registration_successful')  # Redirect to a succ
    else: #else return to form
        user_form = StudentSignupform()
        profile_form = StudentProfileform()
    return render(request, 'Lister_signup.html', {'user_form': user_form, 'p
```



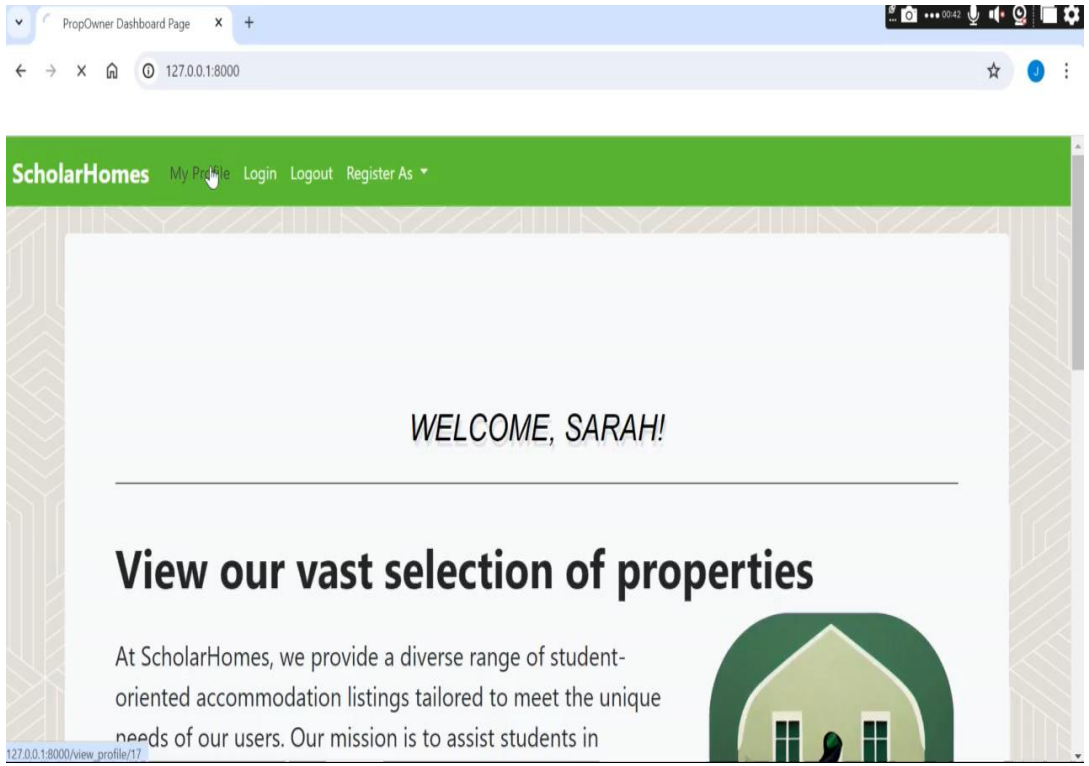
View Properties – Jomi

```python
# retrieves all properties from database and displays them on the property list page
def property_list(request):
    properties = Property.objects.exclude(ongoing=False).order_by('-created_at')  # Order by creation

    # Filter based on property_name
    property_name_query = request.GET.get('property_name')
    if property_name_query:
        properties = properties.filter(property_name__icontains=property_name_query)

    # Filter based on price
    price_query = request.GET.get('price')
    if price_query:
        properties = properties.filter(price__lte=price_query)

    # Filter based on property lister's Degree
    selected_degree = request.GET.get('degree')
    if selected_degree:
        properties = properties.filter(user__student_profile__degree__name=selected_degree)

    # Filter based on property lister's university
    selected_university = request.GET.get('university')
    if selected_university:
        properties = properties.filter(user__student_profile__university__name=selected_university)

    # Retrieve all degrees and universities
    all_degrees = Degrees.objects.all()
    all_universities = Universities.objects.all()
```



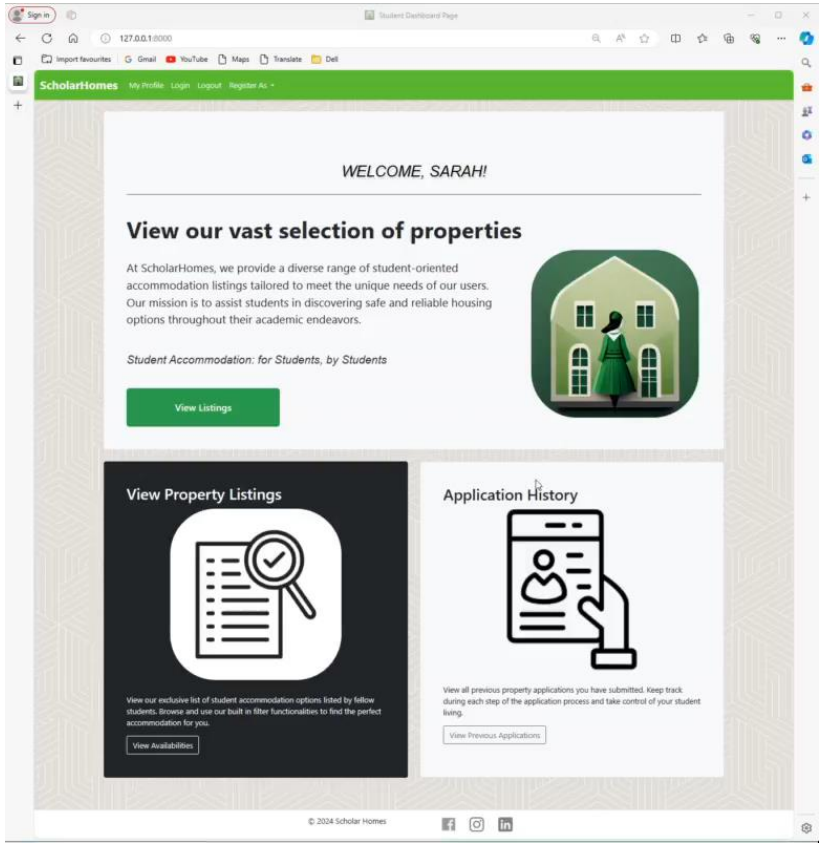
View Profile - jomi

```python
#View user profile who's ID has been passed in the url
def view_profile(request, user_id):
    user = User.objects.get(id=user_id)
    is_following = False  # Initialize the variable indicating the user is not following (used in template)

    #if following
    if Follow.objects.filter(follower=request.user, followed_user=user).exists():
        is_following = True # Changes the variable to following

    #if the user is a lister, pass properties
    if user.is_student == False:
        properties = Property.objects.filter(user=user)
        return render(request, 'view_profile.html', {'user': user, 'properties': properties, 'is_following': is_following})
    else:
        return render(request, 'view_profile.html', {'user': user, 'is_following': is_following})
```



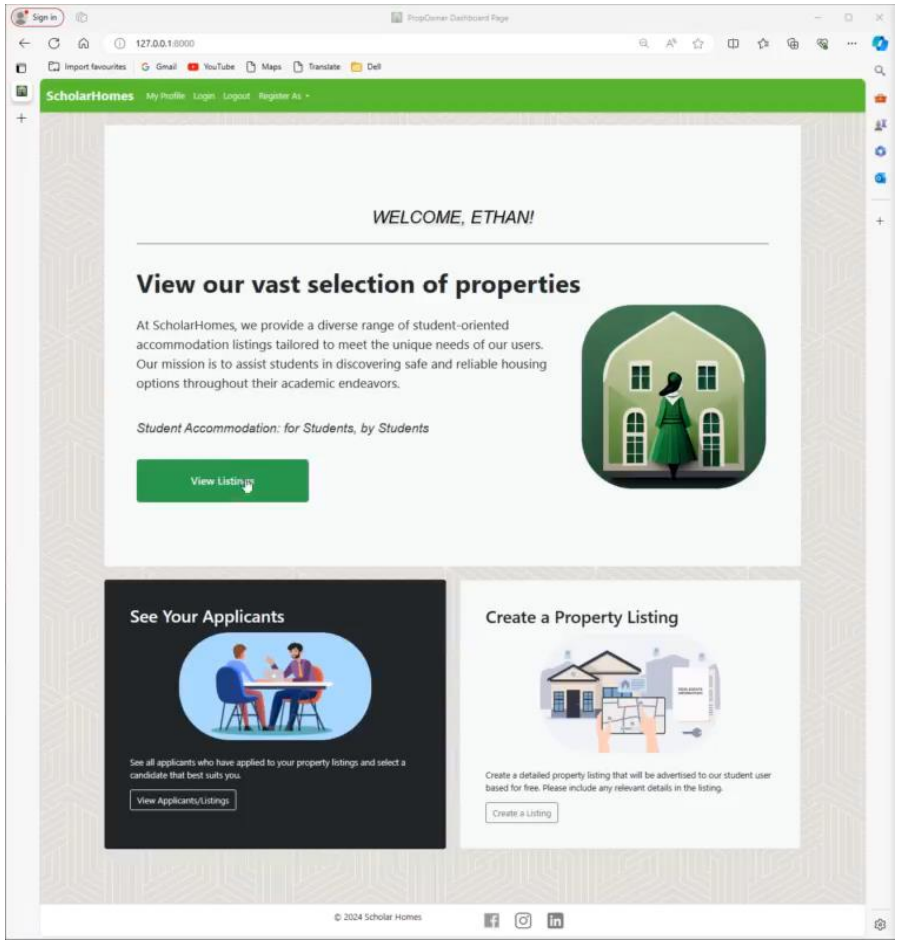
Following/Follower - Ryan

```python
#Function passed to intiate following a user
def follow_user(request, followed_user_id):
    #if user is logged in
    if request.user.is_authenticated:
        is_following = True  # Initialize the variable indicating whe
        followed_user = User.objects.get(pk=followed_user_id)  #set f
        properties = Property.objects.filter(user=followed_user) #get
        if followed_user != request.user:  # Ensure user can't follow
            Follow.objects.get_or_create(follower=request.user, follo
    #go to followed users page
    return render(request,'view_profile.html', {'user':followed_user,


#Function passed to intiate unfollowing a user
def unfollow_user(request, followed_user_id):
    is_following = False  # Initialize the variable indicating whethe
    #if user is logged in
    if request.user.is_authenticated:
        followed_user = User.objects.get(pk=followed_user_id) #set fo
        properties = Property.objects.filter(user=followed_user) #get
        Follow.objects.filter(follower=request.user, followed_user=fo
    #go to unfollowed users page
    return render(request,'view_profile.html', {'user':followed_user,
```



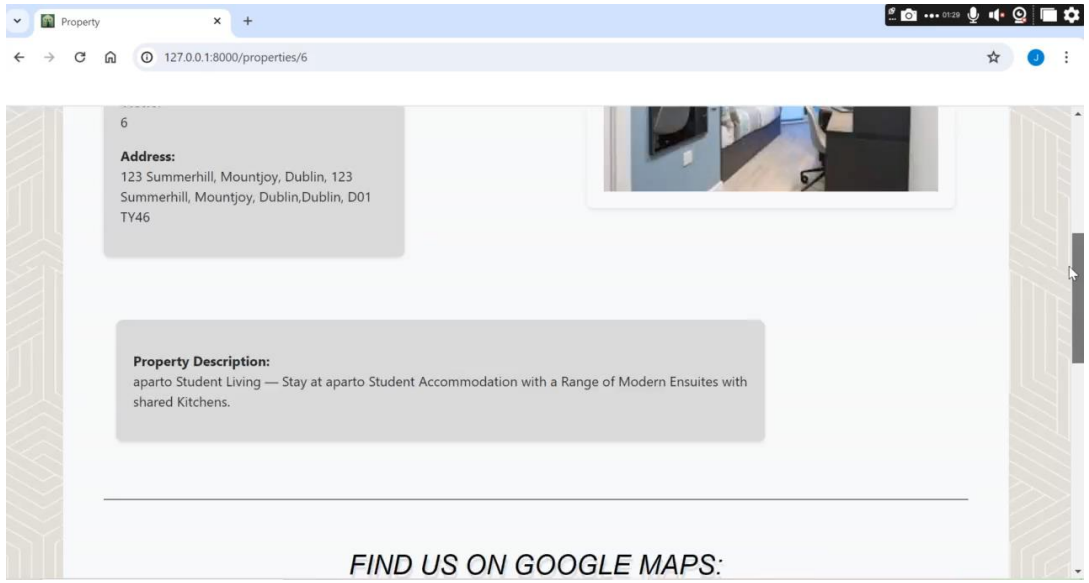
Creating a Listing - Ryan

```python
#directs user to create listing form, must be logged in to access
@login_required
def create_listing(request):

    #if user submits
    if request.method == 'POST':
        form = PropertyForm(request.POST, request.FILES) #REQUEST.FILE
        #if form is valid
        if form.is_valid():
            application = form.save(commit=False) #set form infomation
            application.user = request.user #set applications user to
            application.save() #save to db
            return redirect('application_successful')
    else: #else resubmit form
        form = PropertyForm()

    return render(request, 'create_listing.html', {'form': form})
```



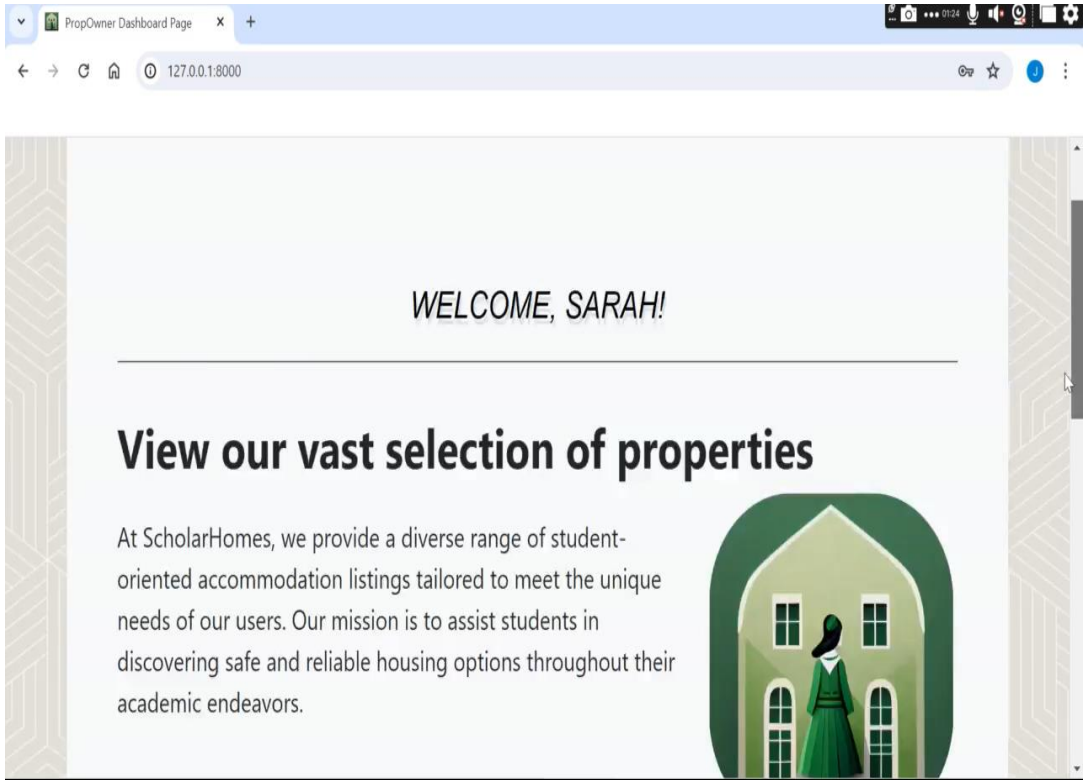
Apply to Listing (View/Delete listings) - Jomi

```python
#Allows users to delete an application they have made
@login_required
def delete_application(request, application_id):
    if request.method == 'POST':
        application = get_object_or_404(RentalApplication,
        # Check if the logged-in user owns this application
        if application.user == request.user:
            application.delete()
            applications = RentalApplication.objects.filter
            return render(request, 'my_applications.html',
        else:
            error_msg = 'Unable to delete application due t
            applications = RentalApplication.objects.filter
            return render(request, 'my_applications.html',
    else:
        # Handle GET request if needed
        pass
```



View my Listings and Applications - Jomi

```python
#directs user to owner dashboard, must be logged in to access, displays all properties user created and ap
@login_required
def Listing_hub(request):
    properties = Property.objects.filter(user=request.user) #filter to be user's properties
    applications = RentalApplication.objects.filter(property__in=properties).exclude(status='declined') #F

    return render(request, 'Listings_hub.html', {'properties': properties, 'applications': applications})


#directs my applications page, must be logged in to access, displays all applications user created
@login_required
def my_applications(request):
    # Retrieve applications for the current user
    applications = RentalApplication.objects.filter(user=request.user) #retrieve user's applications

    return render(request, 'my_applications.html', {'applications': applications})
```



Future Features - Jomi

- Roommate Matching
- Virtual Tours
- Rating and Review System
- Real-time Notifications

The Future Goals and Objectives - Jomi

- Partnerships with Universities: Form partnerships with institutions to become their go-to resource for housing searches. This will make it easy for students to find safe housing.
- Sustainability Initiatives: Support ways of living that are good for the earth and work with companies that provide eco-friendly housing to provide sustainable lodging.
- Make a Community Platform: Create a community within the app so that students can talk about housing, share their experiences, and give each other tips, creating a helpful network.
- Regular changes and New Features: Commit to regular changes and adding new features that meet changing user needs and take advantage of new technology.