# Sprint 2 – Completed User Story (Technical Development)

## 1. User Story Description

**User Story ID:** US 13

**Title:** Madison login

**Priority:** P1 / 1

**Story Points:** 2

**User Story:**

As Madison (site owner), I want to log in securely so that I can manage my site content without unauthorized access.

## 2. Acceptance Criteria

• Given I am logged out, When I enter valid credentials, I should be logged in securely.

## 3. Technical Implementation Details

This branch implements a **database-based authentication system** with login/logout functionality for the Real Estate Portal Django application. The implementation replaces the previous settings-based authentication with a proper SQLite database-backed user system that matches the ERD schema design.

**Key Features:**

- Custom User model extending Django's `AbstractBaseUser`
- Email-based authentication (no username required)

- Password hashing using Django's PBKDF2 algorithm
- Secure logout using POST requests (CSRF protected)
- Admin-managed user accounts (no public registration)
- Session-based authentication
- Custom login form with email validation and normalization

**Authentication Flow:**

1. User navigates to `/login/`
2. Enters email address and password
3. Django validates credentials against database
4. On success: User is authenticated and redirected to homepage
5. On failure: Error message displayed
6. Logout: Secure POST request clears session and redirects to login page

# 4. Code Breakdown

## 4.1 Models Updated/Created:

**Created: `listings/models.py` - Custom User Model**

```python
class UserManager(BaseUserManager):
    """Manager for custom User model."""
    def create_user(self, email, password=None, firstname='', lastname=''):
        """Create and return a regular user."""
        # Validates email, normalizes it, creates user with hashed password

    def create_superuser(self, email, password=None, firstname='', lastname=''):
        """Create and return a superuser."""
        # Creates user with staff and superuser privileges

class User(AbstractBaseUser):
    """Custom User model matching the database schema."""
    user_id = models.AutoField(primary_key=True, db_column='User_ID')
    email = models.EmailField(unique=True, db_column='Email')
    password = models.CharField(max_length=128, db_column='Password_Hash')
    firstname = models.CharField(max_length=100, db_column='Firstname')
    lastname = models.CharField(max_length=100, db_column='Lastname')

    # Django auth fields
    is_active = models.BooleanField(default=True)
    is_staff = models.BooleanField(default=False)
    is_superuser = models.BooleanField(default=False)
    date_joined = models.DateTimeField(auto_now_add=True)

    USERNAME_FIELD = 'email'
    REQUIRED_FIELDS = ['firstname', 'lastname']

    def get_full_name(self):
        """Return full name for display."""
        return f"{self.firstname} {self.lastname}".strip()
```

### Key Model Features:

- Maps to database schema columns ( `User_ID` , `Email` , `Password_Hash` , `Firstname` , `Lastname` )
- Uses email as the username field (no separate username)
- Includes Django authentication fields for permissions and session management
- Custom manager for creating users and superusers

## 4.2 Views Implemented:

**Updated:** `listings/views.py`

```python
def home(request):
    """Public homepage showing published listings."""
    # Use is_visible for filtering (backward compatible with is_published property)
    listings = Listing.objects.filter(is_visible=True).order_by('-listed_date')
    return render(request, 'listings/home.html', {'listings': listings})
```

**Authentication Views:**

- **Login:** Uses Django's built-in `LoginView` with custom form ( `CustomLoginForm` )
- **Logout:** Uses Django's built-in `LogoutView` (POST-only for security)
- Both views are configured via URL patterns, not custom view functions

**View Configuration:**

- Login view configured in `listings/urls.py` with custom template and form
- Logout view configured with redirect to login page
- Home view updated to work with authenticated users (displays user name in navigation)

## 4.3 URLs Added:

**Updated:** `listings/urls.py`

```python
from django.contrib.auth import views as auth_views
from .forms import CustomLoginForm

urlpatterns = [
    path('', views.home, name='home'),
    path('home/', views.home, name='home_alias'),
    path(
        'login/',
        auth_views.LoginView.as_view(
            template_name='accounts/login.html',
            authentication_form=CustomLoginForm
        ),
        name='login'
    ),
    path('logout/', auth_views.LogoutView.as_view(), name='logout'),
]
```

### URL Routes:

- `/login/` - Login page (GET: display form, POST: authenticate)
- `/logout/` - Logout endpoint (POST only, redirects to login)
- `/` and `/home/` - Homepage (shows listings, displays user info if logged in)

**Root URL Configuration:** `realestate_portal/urls.py`

```python
urlpatterns = [
    path('admin/', admin.site.urls),
    path('accounts/', include('django.contrib.auth.urls')),
    path('', include('listings.urls')),
]
```

## 4.4 Templates Created:

**Created:** `templates/accounts/login.html`

```
{% extends 'base.html' %}

{% block content %}
<div class="login-page">
    <div class="login-container">
        <div class="login-left">
            <div class="login-form-card">
                <div class="message">Welcome to AdVance Real Estate</div>
                <form method="post" action="{% url 'login' %}" class="login-form">
                    {% csrf_token %}
                    <!-- Email field with icon -->
                    <!-- Password field with icon -->
                    <!-- Submit button -->
                    <div class="form-footer">
                        <p>Need an account? Contact your administrator.</p>
                    </div>
                </form>
            </div>
        </div>
        <aside class="login-right">
            <!-- Promotional content -->
        </aside>
    </div>
</div>
{% endblock %}
```

### Updated: `templates/base.html`

```
<nav>
    {% if user.is_authenticated %}
    <span>Welcome, {{ user.get_full_name|default:user.email }}!</span>
    <form method="post" action="{% url 'logout' %}" style="display: inline;">
        {% csrf_token %}
        <button type="submit">Logout</button>
    </form>
    {% else %}
    <a href="{% url 'login' %}">Login</a>
    {% endif %}
</nav>
```

### Template Features:

- Two-column layout (login form on left, promotional content on right)
- Email and password fields with icons
- CSRF token protection
- Error message display
- Responsive design
- Secure logout form (POST request)

## 4.5 Any Supporting Utility Functions:

**Created: `listings/forms.py` - CustomLoginForm**

```python
class CustomLoginForm(AuthenticationForm):
    """Custom login form using email instead of username."""
    username = forms.EmailField(
        label='Email',
        widget=forms.EmailInput(attrs={
            'class': 'form-control',
            'autofocus': True,
            'placeholder': 'Enter your email'
        })
    )
    password = forms.CharField(
        widget=forms.PasswordInput(attrs={
            'class': 'form-control',
            'placeholder': 'Enter your password'
        })
    )

    def clean_username(self):
        """Normalize email to lowercase."""
        username = self.cleaned_data.get('username')
        if username:
            username = username.lower().strip()
        return username
```

**Form Features:**

- Extends Django's `AuthenticationForm`
- Uses `EmailField` instead of `CharField` for username

- Email normalization (lowercase, trimmed)
- Custom styling classes for form controls
- Autofocus on email field

**Settings Configuration:** **`realestate_portal/settings.py`**

```
# Custom User Model
AUTH_USER_MODEL = 'listings.User'

# Authentication Backend
AUTHENTICATION_BACKENDS = [
    'django.contrib.auth.backends.ModelBackend',
]

# Authentication URLs
LOGIN_URL = 'login'
LOGIN_REDIRECT_URL = '/'
LOGOUT_REDIRECT_URL = 'login'
```

**Admin Configuration:** **`listings/admin.py`**

```
@admin.register(User)
class UserAdmin(BaseUserAdmin):
    """Admin interface for custom User model."""
    list_display = ['email', 'firstname', 'lastname', 'is_staff', 'is_active', 'date_joined']
    list_filter = ['is_staff', 'is_superuser', 'is_active', 'date_joined']
    search_fields = ['email', 'firstname', 'lastname']
    ordering = ['email']
    readonly_fields = ['date_joined']
```

**Admin Features:**

- Custom UserAdmin for managing user accounts
- List display with key user information
- Filtering by staff status, superuser status, active status
- Search by email, first name, last name
- Read-only date_joined field

## 5. Testing Performed

### Manual Testing:

**Login Functionality:**

- ✅ Valid credentials: User successfully logs in and is redirected to homepage
- ✅ Invalid email: Error message displayed ("Please enter a valid email address")
- ✅ Invalid password: Error message displayed ("Please enter a correct email and password")
- ✅ Empty fields: Form validation prevents submission
- ✅ Email normalization: Email addresses are converted to lowercase automatically
- ✅ Session persistence: User remains logged in across page navigation
- ✅ Login redirect: After login, user is redirected to homepage ( `/` )

**Logout Functionality:**

- ✅ Secure logout: Logout requires POST request (CSRF protected)
- ✅ Session cleared: User session is properly terminated
- ✅ Redirect: After logout, user is redirected to login page
- ✅ GET request blocked: Attempting logout via GET returns 405 Method Not Allowed

**User Interface:**

- ✅ Navigation display: Shows "Welcome, [Full Name]!" when logged in
- ✅ Login link: Shows "Login" link when logged out
- ✅ Logout button: Displays as button in navigation when logged in
- ✅ Responsive design: Login page displays correctly on different screen sizes

**Admin Interface:**

- ✅ User creation: Admin can create new users via Django admin
- ✅ Password hashing: Passwords are automatically hashed (PBKDF2)
- ✅ User management: Admin can edit, activate/deactivate, and delete users
- ✅ Date joined: Automatically set and displayed as read-only

**Security Testing:**

- ✅ CSRF protection: All forms include CSRF tokens
- ✅ Password hashing: Passwords stored as hashes, not plain text
- ✅ Session security: Django session framework properly configured
- ✅ Secure logout: POST-only logout prevents CSRF attacks

**Database Testing:**

- ✅ User model: Custom User model correctly mapped to database schema
- ✅ Migrations: All migrations applied successfully
- ✅ Foreign keys: Listing model correctly references User model
- ✅ Data integrity: User accounts persist correctly in database

## System Checks:

- ✅ `python manage.py check` - No errors
- ✅ `python manage.py migrate` - All migrations applied successfully
- ✅ Database schema matches ERD design (~99% match rate)

## 6. Screenshots / Evidence

*(Screenshots to be added:)*

- Login page with form
- Successful login redirect to homepage
- Navigation showing logged-in user
- Logout functionality
- Admin interface for user management
- Error messages for invalid credentials

## 7. GitHub Repository Link

**Repository:** https://github.com/tbrzezowsky/ISQA8210-Team3

**Branch:** `database-login`

**Key Commits:**

- Initial database-based login implementation
- Photo table typo fix (Phot_Display_Order → Photo_Display_Order)
- Listing NOT NULL constraints enforcement
- Schema comparison documentation

**Related Documentation:**

- `PROJECT_GUIDE.md` - Complete project documentation
- `SCHEMA_COMPARISON.md` - ERD vs Implementation comparison
- `DATABASE_LOGIN_IMPLEMENTATION.md` - Technical implementation details

# 8. Live Application Link

*(To be updated when deployed:)*

- Development: http://127.0.0.1:8000/
- Login: http://127.0.0.1:8000/login/
- Admin: http://127.0.0.1:8000/admin/

# Additional Notes

## Database Schema Compliance:

- Implementation matches ERD schema design (~99% match rate)
- Only difference: Photo table typo correction (improvement)
- All NOT NULL constraints enforced
- Foreign key relationships properly configured

## Security Features:

- Password hashing: PBKDF2 with SHA256
- CSRF protection: Enabled on all forms
- Session-based authentication: Secure session management
- Secure logout: POST-only requests
- Email validation: Built-in Django email field validation

## User Management:

- **Admin-only user creation:** No public registration form
- **User accounts:** Created and managed through Django admin interface

- **Credentials:** Shared securely by site administrator
- **Password management:** Admin can change user passwords

## Migration Files:

- `0001_initial.py` - Initial schema creation (User, Listing, lookup tables, etc.)
- `0002_fix_photo_display_order_typo.py` - Photo table column name correction
- `0003_enforce_listing_not_null_constraints.py` - Listing NOT NULL constraints

**Document Version:** 1.0
**Last Updated:** November 2025
**Branch:** `database-login`
**Status:** ✅ Complete and Ready for Review