Extend the blog website to include user registration, login, and REST APIs for blog posts.

To extend the blog website to include user registration, login, and REST APIs for blog posts, we will follow these steps:

1. **Add User Authentication**
2. **Create REST APIs**
3. **Test the Application**

Step 1: Add User Authentication

1. **Install Django REST framework and Django Allauth:**

```
pip install djangorestframework django-allauth dj-rest-auth
```

2. **Update `settings.py`:**
   - Add the required applications to the `INSTALLED_APPS`:

     python

```
INSTALLED_APPS = [
    ...
    'rest_framework',
    'rest_framework.authtoken',
    'dj_rest_auth',
    'allauth',
    'allauth.account',
    'allauth.socialaccount',
    'dj_rest_auth.registration',
]
```

   - Add the REST framework configuration:

     python

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.SessionAuthentication',
        'rest_framework.authentication.TokenAuthentication',
    ],
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ],
}
```

   - Update the authentication backends and add email settings:

     python

```
AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.ModelBackend',
    'allauth.account.auth_backends.AuthenticationBackend',
```

```
)

EMAIL_BACKEND =
'django.core.mail.backends.console.EmailBackend'

SITE_ID = 1
```

3. **Update `urls.py`:**
   - Include the authentication URLs in the main `urls.py`:

   python

```
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/auth/', include('dj_rest_auth.urls')),
    path('api/auth/registration/',
include('dj_rest_auth.registration.urls')),
    path('', include('blog.urls')),
]
```

4. **Create Registration and Login Templates:**
   - Create `templates/account` folder in your project.
   - Add `login.html` and `signup.html` files for user login and registration templates. Here are basic examples:

   login.html:

```
html
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
    <h1>Login</h1>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Login</button>
    </form>
    <a href="{% url 'account_signup' %}">Sign Up</a>
</body>
</html>
```

   signup.html:

   html

```
<!DOCTYPE html>
<html>
<head>
    <title>Sign Up</title>
</head>
<body>
```

```
    <h1>Sign Up</h1>
    <form method="post">
        {% csrf_token %}
        {{ form.as_p }}
        <button type="submit">Sign Up</button>
    </form>
    <a href="{% url 'account_login' %}">Login</a>
</body>
</html>
```

Step 2: Create REST APIs

1. **Create a Serializer for the Post Model:**
   - Create `blog/serializers.py`:

     python

```
from rest_framework import serializers
from .models import Post

class PostSerializer(serializers.ModelSerializer):
    class Meta:
        model = Post
        fields = '__all__'
```

2. **Create API Views:**
   - Create `blog/api_views.py`:

     python

```
from rest_framework import generics
from rest_framework.permissions import
IsAuthenticatedOrReadOnly
from .models import Post
from .serializers import PostSerializer

class PostListCreateView(generics.ListCreateAPIView):
    queryset = Post.objects.all().order_by('-created_at')
    serializer_class = PostSerializer
    permission_classes = [IsAuthenticatedOrReadOnly]

class
PostRetrieveUpdateDestroyView(generics.RetrieveUpdateDestroyAPI
View):
    queryset = Post.objects.all()
    serializer_class = PostSerializer
    permission_classes = [IsAuthenticatedOrReadOnly]
```

3. **Update URLs to Include API Endpoints:**
   - Create `blog/api_urls.py`:

     python

```
from django.urls import path
from . import api_views

urlpatterns = [
```

```
    path('posts/', api_views.PostListCreateView.as_view(),
name='post_list_create'),
    path('posts/<int:pk>/',
api_views.PostRetrieveUpdateDestroyView.as_view(),
name='post_detail_update_destroy'),
]
```

- Include the API URLs in the main `urls.py`:

```
python
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/auth/', include('dj_rest_auth.urls')),
    path('api/auth/registration/',
include('dj_rest_auth.registration.urls')),
    path('api/', include('blog.api_urls')),
    path('', include('blog.urls')),
]
```

Step 3: Test the Application

1. **Run the Development Server:**
   - Start the development server:

```
python manage.py runserver
```

2. **Test User Registration and Login:**
   - Navigate to `http://127.0.0.1:8000/api/auth/registration/` to register a new user.
   - Navigate to `http://127.0.0.1:8000/api/auth/login/` to log in.
3. **Test API Endpoints:**
   - Use a tool like Postman or your browser to test the API endpoints:
     - `GET /api/posts/` - List all posts.
     - `POST /api/posts/` - Create a new post (requires authentication).
     - `GET /api/posts/<id>/` - Retrieve a specific post.
     - `PUT /api/posts/<id>/` - Update a specific post (requires authentication).
     - `DELETE /api/posts/<id>/` - Delete a specific post (requires authentication).

By following these steps, you will have extended your Django blog website to include user registration, login, and REST APIs for blog posts. You can further enhance this project by adding features like comments, categories, and tags, and by improving the frontend with more advanced templates and styling.