Programación web de CS50 con Python y JavaScript

OpenCourseWare

Donar (https://cs50.harvard.edu/donate)

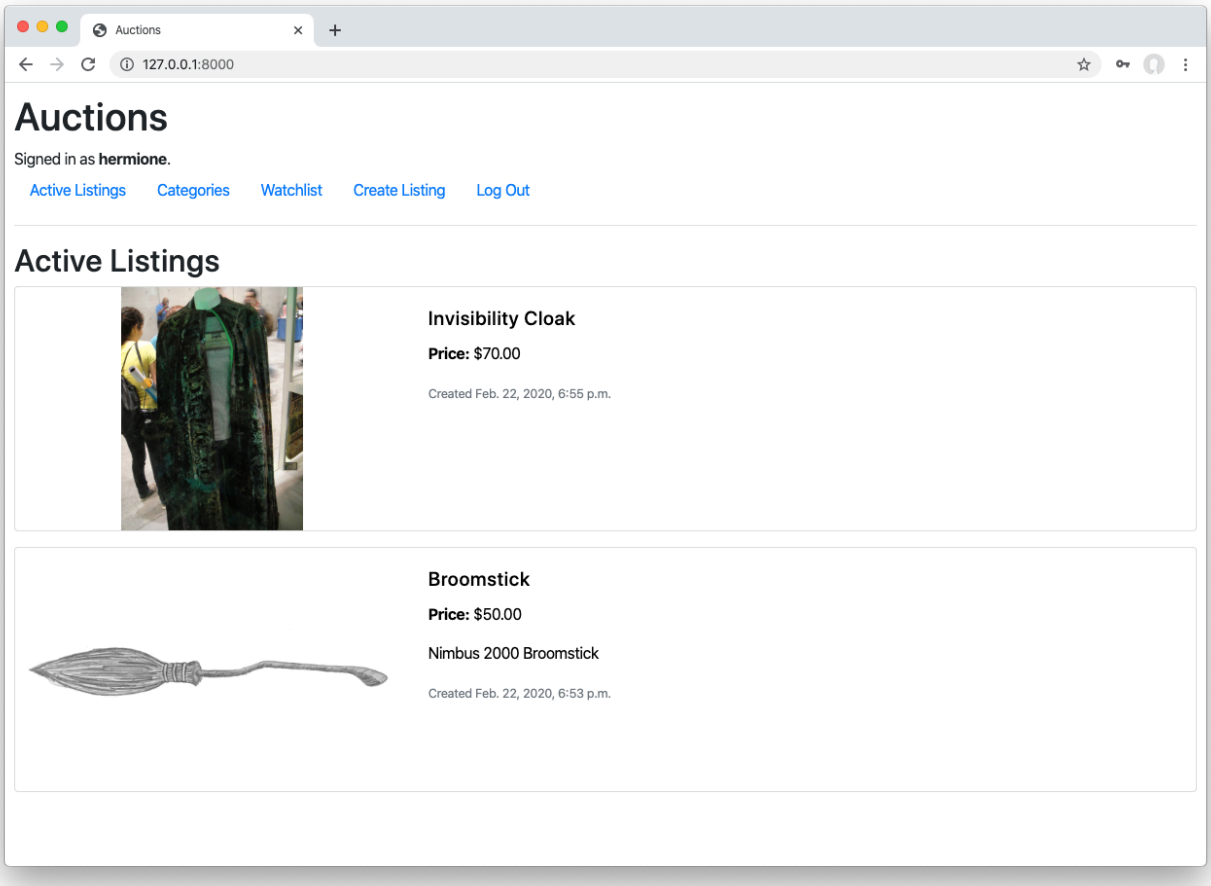
Brian Yu (https://brianyu.me)
brian@cs.harvard.edu

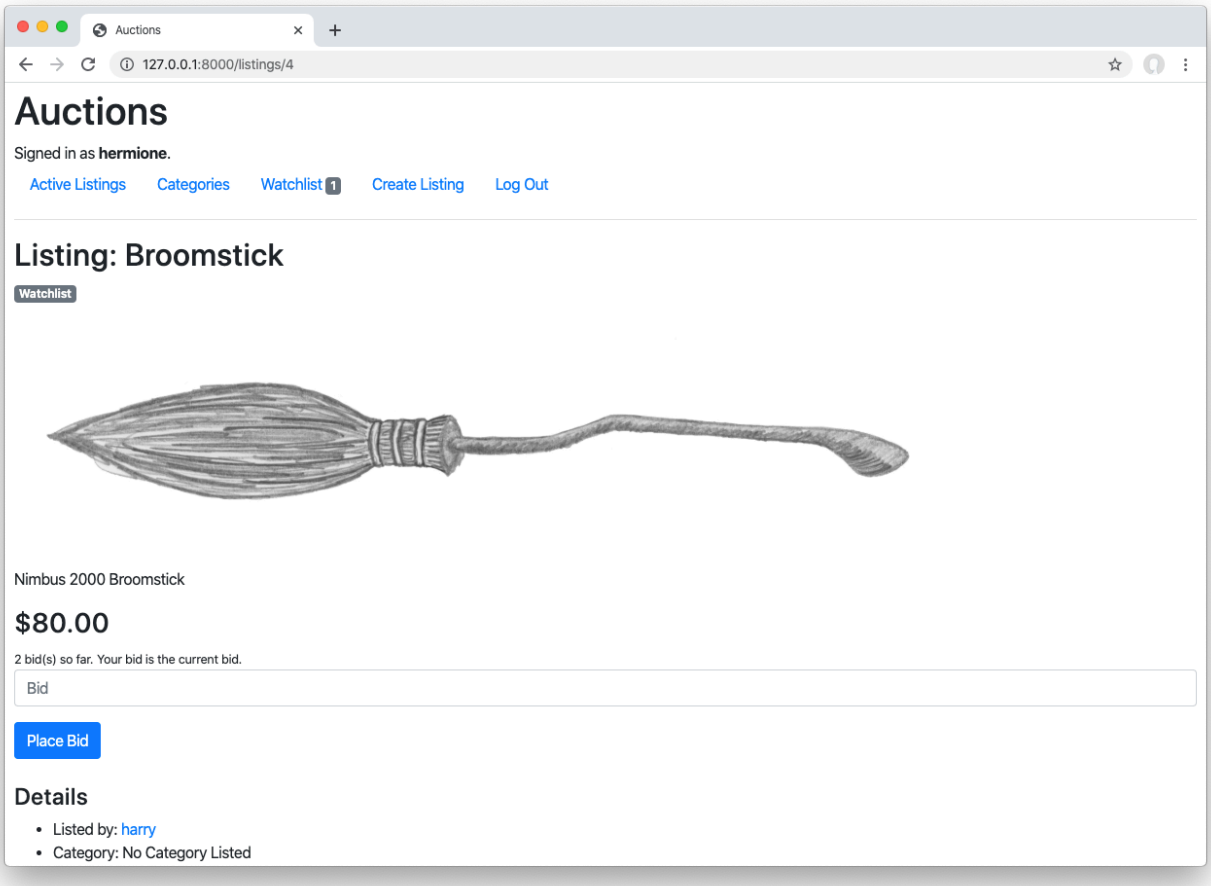
David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.facebook.com/dmalan) (https://github.com/dmalan)
(https://www.instagram.com/davidjmalan/) (https://www.linkedin.com/in/malan/)
(https://www.reddit.com/user/davidjmalan)
(https://www.threads.net/@davidjmalan) (https://twitter.com/davidjmalan)

Comercio

> CS50W no presenta una correspondencia uno a uno entre conferencias y proyectos. Si estás intentando este proyecto sin haber visto al menos la Lección 4, ¡lo estás intentando demasiado pronto!

Diseñe un sitio de subastas de comercio electrónico similar a eBay que permita a los usuarios publicar listados de subastas, realizar ofertas en los listados, comentar sobre esos listados y agregar listados a una "lista de vigilancia".


Auctions
127.0.0.1:8000
Auctions
Signed in as hermione.
Active Listings
Categories
Watchlist
Create Listing
Log Out
Active Listings
Invisibility Cloak
Price: $70.00
Created Feb. 22, 2020, 6:55 p.m.
Broomstick
Price: $50.00
Nimbus 2000 Broomstick
Created Feb. 22, 2020, 6:53 p.m.


Auctions
127.0.0.1:8000/listings/4
Auctions
Signed in as hermione.
Active Listings
Categories
Watchlist 1
Create Listing
Log Out
Listing: Broomstick
Watchlist
Nimbus 2000 Broomstick
$80.00
2 bid(s) so far. Your bid is the current bid.
Bid
Place Bid
Details
Listed by: harry
Category: No Category Listed

Empezando

1. Descargue el código de distribución de https://cdn.cs50.net/web/2020/spring/projects/2/commerce.zip (https://cdn.cs50.net/web/2020/spring/projects/2/commerce.zip) y descomprímalo.
2. En tu terminal, `cd` en el `commerce` directorio.
3. Ejecute `python manage.py makemigrations auctions` para realizar migraciones para la `auctions` aplicación.
4. Ejecute `python manage.py migrate` para aplicar migraciones a su base de datos.

Comprensión

En el código de distribución hay un proyecto Django llamado `commerce` que contiene una única aplicación llamada `auctions`.

Primero, abra `auctions/urls.py`, donde se define la configuración de URL para esta aplicación. Tenga en cuenta que ya hemos escrito algunas URL para usted, incluida una ruta de índice predeterminada, una `/login` ruta, una `/logout` ruta y una `/register` ruta.

Échale un vistazo `auctions/views.py` para ver las vistas asociadas a cada una de estas rutas. La vista de índice por ahora devuelve una plantilla casi vacía `index.html`. La `login_view` vista muestra un formulario de inicio de sesión cuando un usuario intenta OBTENER la página. Cuando un usuario envía el formulario utilizando el método de solicitud POST, el usuario se autentica, inicia sesión y se le redirige a la página de índice. La `logout_view` vista cierra la sesión del usuario y lo redirige a la página de índice. Finalmente, la `register` ruta muestra un formulario de registro al usuario y crea un nuevo usuario cuando se envía el formulario. Todo esto se hace por usted en el código de distribución, por lo que debería poder ejecutar la aplicación ahora para crear algunos usuarios.

Ejecute `python manage.py runserver` para iniciar el servidor web Django y visite el sitio web en su navegador. Haga clic en "Registrarse" y regístrese para obtener una cuenta. Debería ver que ahora ha iniciado sesión como su cuenta de usuario y que los enlaces en la parte superior de la página han cambiado. ¿Cómo cambió el HTML? Eche un vistazo `auctions/templates/auctions/layout.html` al diseño HTML de esta aplicación. Tenga en cuenta que varias partes de la plantilla están incluidas en una marca de verificación if `user.is_authenticated`, de modo que se puede representar contenido diferente dependiendo de si el usuario ha iniciado sesión o no. ¡Puedes cambiar este archivo si deseas agregar o modificar algo en el diseño!

Finalmente, eche un vistazo a `auctions/models.py`. Aquí es donde definirá cualquier modelo para su aplicación web, donde cada modelo representa algún tipo de datos que desea almacenar en su base de datos. Comenzamos con un `User` modelo que representa a cada

usuario de la aplicación. Debido a que hereda de `AbstractUser`, ya tendrá campos para un nombre de usuario, correo electrónico, contraseña, etc., pero puede agregar nuevos campos a la `User` clase si hay información adicional sobre un usuario que desea representar. También deberá agregar modelos adicionales a este archivo para representar detalles sobre listados de subastas, ofertas, comentarios y categorías de subastas. Recuerde que cada vez que cambie algo en `auctions/models.py`, primero deberá ejecutar `python manage.py makemigrations` y luego `python manage.py migrate` migrar esos cambios a su base de datos.

Especificación

Complete la implementación de su sitio de subastas. Debes cumplir con los siguientes requisitos:

- **Modelos** : su aplicación debe tener al menos tres modelos además del `User` modelo: uno para listados de subastas, uno para ofertas y otro para comentarios realizados en listados de subastas. Depende de usted decidir qué campos debe tener cada modelo y cuáles deben ser los tipos de esos campos. Es posible que tenga modelos adicionales si lo desea.
- **Crear listado** : los usuarios deberían poder visitar una página para crear un nuevo listado. Deberían poder especificar un título para el listado, una descripción basada en texto y cuál debería ser la oferta inicial. Opcionalmente, los usuarios también deberían poder proporcionar una URL para una imagen del listado y/o una categoría (por ejemplo, Moda, Juguetes, Electrónica, Hogar, etc.).
- **Página de listados activos** : la ruta predeterminada de su aplicación web debería permitir a los usuarios ver todos los listados de subastas activos actualmente. Para cada listado activo, esta página debe mostrar (como mínimo) el título, la descripción, el precio actual y la foto (si existe una para el listado).
- **Página de listado** : al hacer clic en un listado, los usuarios deberían acceder a una página específica de ese listado. En esa página, los usuarios deberían poder ver todos los detalles sobre el listado, incluido el precio actual del listado.
    - Si el usuario ha iniciado sesión, debería poder agregar el elemento a su "Lista de seguimiento". Si el elemento ya está en la lista de seguimiento, el usuario debería poder eliminarlo.
    - If the user is signed in, the user should be able to bid on the item. The bid must be at least as large as the starting bid, and must be greater than any other bids that have been placed (if any). If the bid doesn't meet those criteria, the user should be presented with an error.
    - If the user is signed in and is the one who created the listing, the user should have the ability to "close" the auction from this page, which makes the highest bidder the winner of the auction and makes the listing no longer active.
    - If a user is signed in on a closed listing page, and the user has won that auction, the page should say so.

    - Users who are signed in should be able to add comments to the listing page. The listing page should display all comments that have been made on the listing.
- **Watchlist**: Users who are signed in should be able to visit a Watchlist page, which should display all of the listings that a user has added to their watchlist. Clicking on any of those listings should take the user to that listing's page.
- **Categories**: Users should be able to visit a page that displays a list of all listing categories. Clicking on the name of any category should take the user to a page that displays all of the active listings in that category.
- **Django Admin Interface**: Via the Django admin interface, a site administrator should be able to view, add, edit, and delete any listings, comments, and bids made on the site.

Hints

- To create a superuser account that can access Django's admin interface, run `python manage.py createsuperuser`.
- See Django's Model field reference (https://docs.djangoproject.com/en/4.0/ref/models/fields/) for possible field types for your Django model.
- You'll likely need to create some Django forms (https://docs.djangoproject.com/en/4.0/topics/forms/) for various parts of this web application.
- Adding the `@login_required` decorator (https://docs.djangoproject.com/en/4.0/topics/auth/default/#the-login-required-decorator) on top of any view will ensure that only a user who is logged in can access that view.
- You're welcome to modify the CSS as much as you'd like, to make the website your own! Some sample screenshots are shown at the top of this page. These are meant only to be examples: your application need not be aesthetically the same as the screenshots here (you're encouraged to be creative!).

How to Submit

1. Visit this link (https://submit.cs50.io/invites/89679428401548238ceb022f141b9947), log in with your GitHub account, and click **Authorize cs50**. Then, check the box indicating that you'd like to grant course staff access to your submissions, and click **Join course**.
2. Install Git (https://git-scm.com/downloads) and, optionally, install `submit50` (https://cs50.readthedocs.io/submit50/).

Cuando envíe su proyecto, el contenido de su `web50/projects/2020/x/commerce` rama debe coincidir con la estructura de archivos del código de distribución descomprimido tal

como se recibió originalmente. Es decir, sus archivos no deben estar anidados dentro de ningún otro directorio de su propia creación. Su rama tampoco debe contener ningún código de ningún otro proyecto, solo este. Si no cumple con esta estructura de archivos, es probable que su envío sea rechazado.

A modo de ejemplo, para este proyecto eso significa que si el personal de calificación visita `https://github.com/me50/USERNAME/tree/web50/projects/2020/x/commerce` (donde `USERNAME` está su propio nombre de usuario de GitHub como se proporciona en el formulario a continuación), deberíamos ver los dos subdirectorios (`auctions` , `commerce`) y el `manage.py` archivo. Si no es así como está organizado su código cuando lo verifica, reorganice su repositorio necesario para que coincida con este paradigma.

3. Si lo ha instalado `submit50` , ejecute

   ```
   submit50 web50/projects/2020/x/commerce
   ```

   De lo contrario, usando Git, envíe su trabajo a `https://github.com/me50/USERNAME.git` , donde `USERNAME` está su nombre de usuario de GitHub, en una rama llamada `web50/projects/2020/x/commerce` .
4. Grabe un screencast (https://www.howtogeek.com/205742/how-to-record-your-windows-mac-linux-android-or-ios-screen/) que no exceda los 5 minutos de duración (y no lo cargue más de un mes antes de enviar este proyecto), en el que demuestre la funcionalidad de su proyecto. Asegúrese de que cada elemento de la especificación anterior se demuestre en su video. No es necesario mostrar su código en este video, solo su aplicación en acción; Revisaremos su código en GitHub. Sube ese video a YouTube (https://www.youtube.com/upload) (como no listado o público, pero no privado) o en otro lugar. En la descripción de su video, debe marcar la hora en la que su video demuestra cada uno de los siete (7) elementos de la especificación. **Esto no es opcional** , los videos sin marcas de tiempo en su descripción serán rechazados automáticamente.
5. Envíe este formulario (https://forms.cs50.io/5659f5b5-7354-4a35-9cc1-d8933840302c) .

Luego puede ir a https://cs50.me/cs50w (https://cs50.me/cs50w) para ver su progreso actual.