Instructions to Run:

Assuming that Django and Python 2.7 are already installed on the machine to be used for testing, the application can be run by simple changing to the NerdBox directory and executing:

python manage.py runserver --insecure

The “--insecure" flag in this case is required because I am using the Django staticfiles module to serve CSS for the pages. In a production deployment I would want to tap into an Apache/IIS server to provide these resources.

Assumptions and Notes:

* Authentication was not mentioned as a requirement for the management portion of the site – in a real-life situation I would bring this to the customer’s attention
* The app as I sent it uses the DEBUG flag in nerdbox.settings to control a special link and page called “clear\_cookie.html”. When accessed this page will remove the session cookie from the computer.
* I assumed that the list of games (either owned or voting) would never grow large enough that a separate paging structure would be needed. Even if every game were added to one list (about 900 games according to Wikipedia) things would still fit comfortably on a scrolling page. The bigger concern at that point is space to store the games.
* I used the built-in Django object relational mapping for my models. The definition of the tables in the project spec seemed like it was somewhat negotiable, and using the Django ORM provides a lot of benefits
* I used a SQLite3 database because it is easy to create/distribute in this environment. In a production environment I would obtain the customer’s DB settings and use those.
* Per the project spec I did not get any more complicated than using a session cookie to determine if a user has already voted or not. There are two things to note here:
  + Rather than storing any data about the last time that a user voted/added a game in the user’s cookie, I used Django’s built-in cached\_db session type, which stores a unique session id in the user cookie and keeps any persistent data in the database with a read cache front-end.
  + As many people on the internet know – preventing internet voting fraud is hard. The session cookie solution presented is easily worked around – the user just needs to clear their recent browsing history and they will be allowed to vote again. Tracking by IP or MAC address is a little better, but IP addresses can be changed and MAC addresses can be spoofed. If this system were going to be present on a LAN as detailed in the spec, I would hopefully be able to tap into a backend authentication system of some sort such as LDAP, Active Directory, etc. so that voting could be tracked per-user.