Press F to Pay Respects

DnD Pal Fall 2018

Overview

Our application is meant to be an easy way for people to get started with D&D (5e), specifically to create a character, and not be completely overwhelmed by the amount of weird naming conventions and lookups involved in the creation of characters and progression through the game. This application is different from other dungeons and dragons character creators because it allows you to create the character but also progress, modify attributes (spells etc.) in accordance to official handbook rules, and view suggestions for good spells to pick.

Team members:

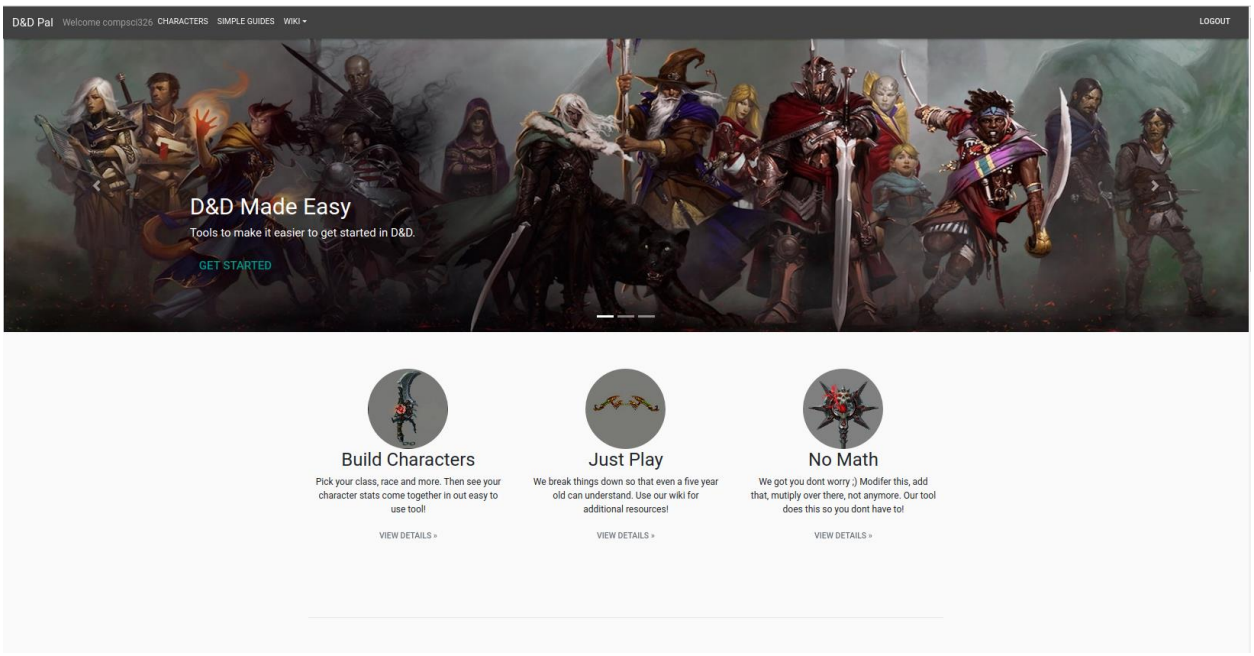
Alan Ruiz-Castro, Niko Giraud, Manuel Moquete, Nicole Morales, and Connor Wong

Github Repository:

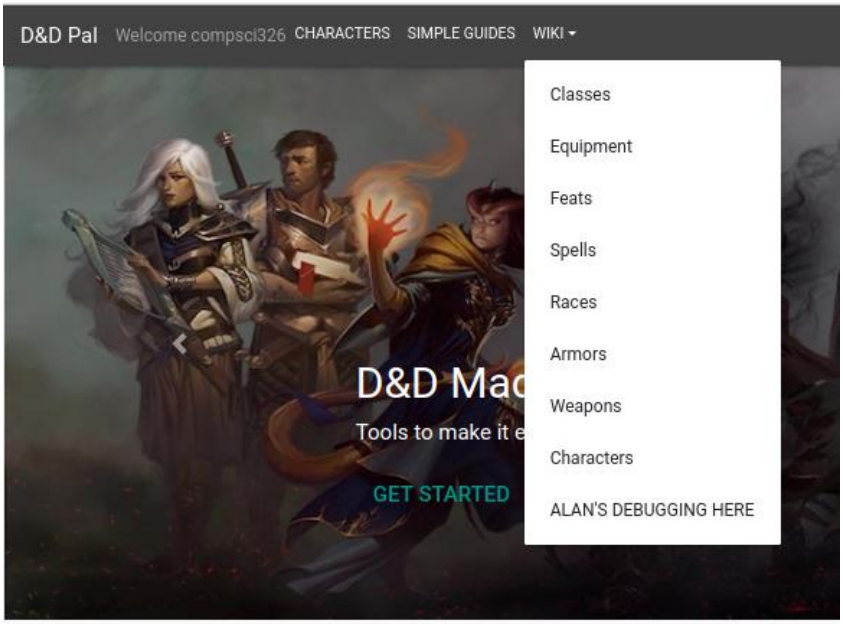
https://github.com/nicmorales/CS326F

User Interface:

Landing page: Pretty self explanatory, on the actual page there is more stuff as you scroll down

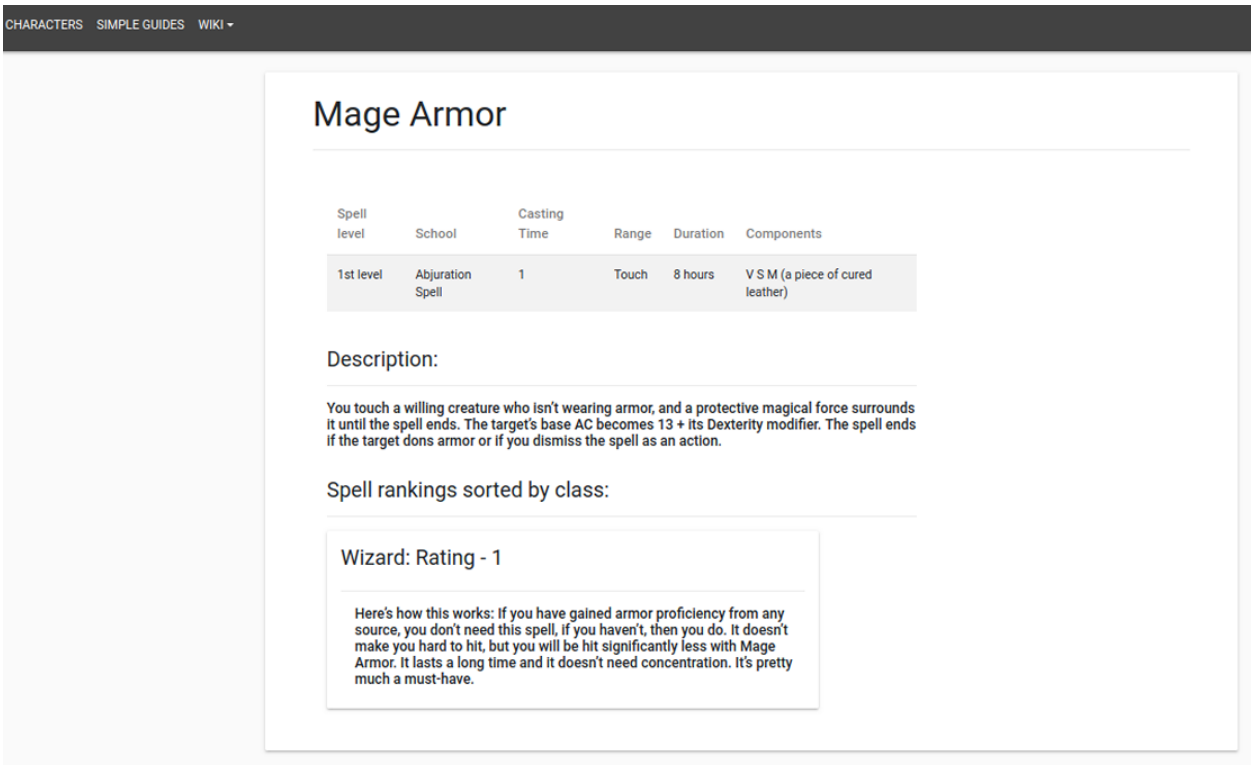


Wiki: The wiki is basically just a display of stuff we had to have in our database, but is a nice local guide for further detail on what stuff is. More is explained in the URLs section, but for various things shown in the drop down, there is a list of all instances of that thing and then when you click on one there is a page with much more detail on that instance of the thing. An example with spells is shown in the screencaps.



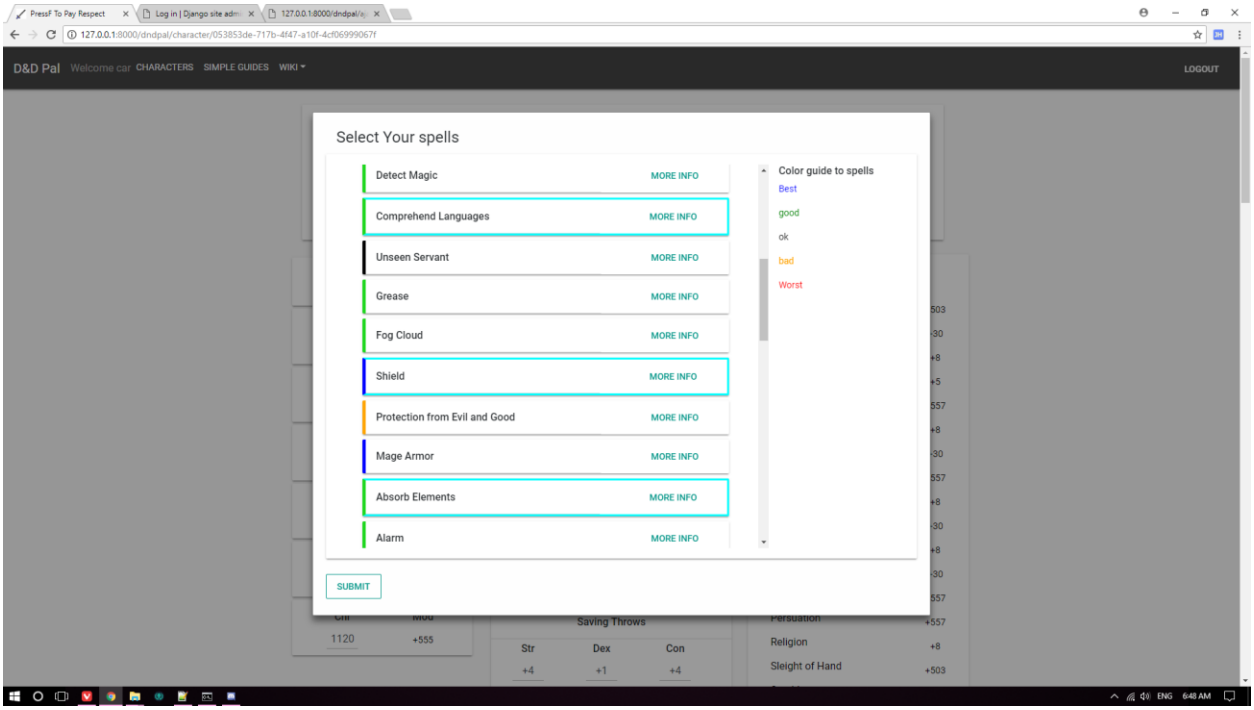
List of Spells

- Absorb Elements
- Alarm
- Burning Hands
- Aganazzar's Scorcher
- Alter Self
- Blindness Deafness
- Acid Splash
- Blade Ward
- Chill Touch
- Dancing Lights
- Fire Bolt
- Friends
- Light
- Mage Hand
- Mending
- Message
- Minor Illusion
- Poison Spray
- Prestidigitation
- Ray of Frost
- Shocking Grasp
- True Strike
- Charm Person
- Chromatic Orb
- Color Spray
- Comprehend Languages
- Detect Magic
- Disguise Self
- Expeditious
- False Life
- Feather Fall
- Find Familiar
- Fog Cloud
- Grease
- Identify
- Illusory Script
- Jump
- Longstrider
- Mage Armor
- Magic Missile
- Protection from Evil and Good
- Ray of Sickness
- Shield
- Silent Image
- Sleep
- Tasha's Hideous Laughter
- Thunderwave
- Unseen Servant

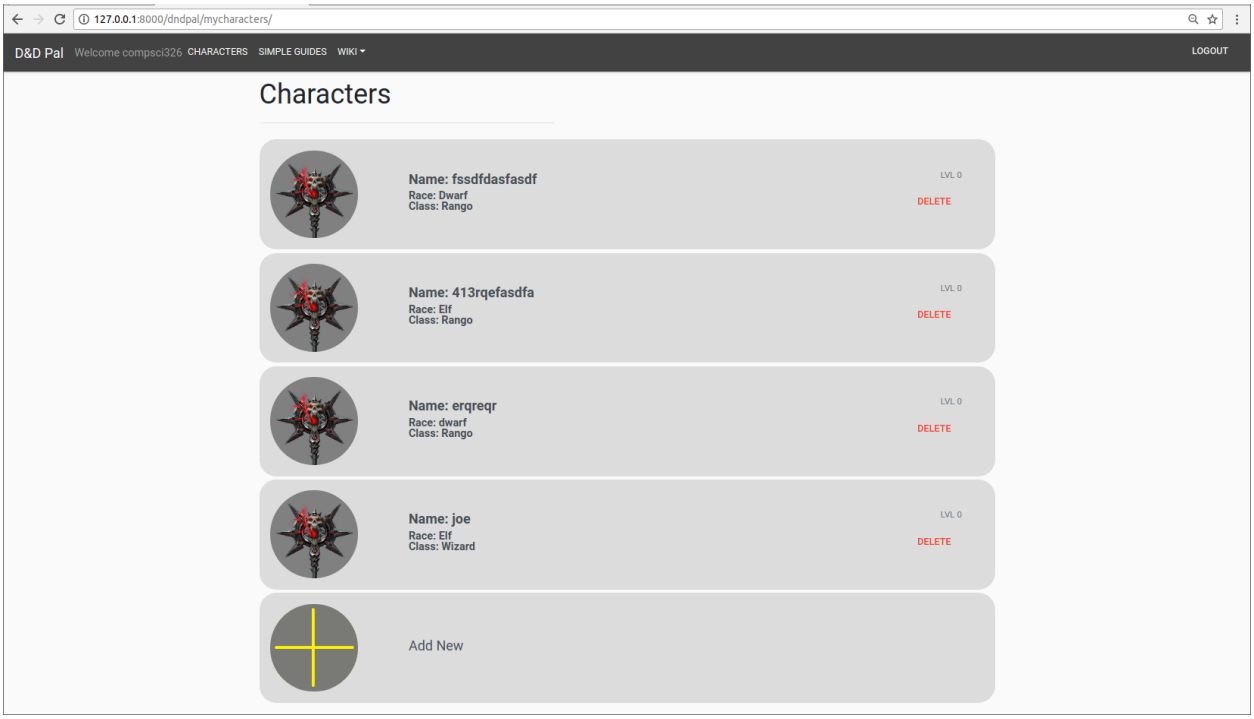


Guided char creation: This has become the default and only character creation mode since it is the only one that is actually useful or unique to our app and because we are busy people :(But it basically allows you to create a DnD 5e character according to official rulebook constraints (didn't get to enforcing all, because way too many) and shows suggestions for spells to pick as shown below and you can enter in your info and level up. I failed to make a proper parser of the actual wiki/ data models changed very often so only Wizard data was input and so **only the Wizard class works.** There is also a save functionality for the characters but it **doesn't work when the server is run on some of our computers** (all run in virtual environments, lubuntu

doesn’t seem to work, described more in Authentication section).



Character list: This is a view of the characters tied to the logged in account. It displays “No characters” when the account has none and is not reachable when not logged in.



Manual/Simple: We decided to scrap other character creation views for time reasons and because it wasn't very useful compared to just using a plain editable character sheet pdf.

Data Model: A final up-to-date diagram of your data model including a brief description of each of the entities in your model and their relationships. WAY too big to describe each thing or to put here, it's pretty self explanatory and is best viewed in the actual models.py or the slightly outdated model diagram in the Writeups folder (it's way too big to keep up to date especially since we constantly make changes to it).

URL Mapping:

Root URLS	
Name of Mapping	**URL Route**
admin	admin/
accounts (django.contrib.auth.url)	accounts/
dndpal (dndpal.urls)	dndpal/

Basically everything in our app that we did / is related to DnD is under 'dndpal/', certainly everything the user will use.

dndpal/	
armors	armors/
armor-detail <str:pk>	armor/<str:pk>
equipment	equipment/
equipment-detail <str:pk>	equipment/<str:pk>
races	races/

race-detail <str:pk>	race/<str:pk
spells	spells/
spell-detail <str:pk>	spellr/<str:pk
classes	classes/
class-detail <str:pk>	class/<str:pk
weapons	weapons/
weapon-detail <str:pk>	weapon/<str:pk>
armor_create	armor/create/
armor_update	armor/<str:pk>/update/
armor_delete	armor/<str:pk>/delete/
character list page	mycharacters/
character_create	character/create/
character-detail <uuid:pk>	character/<uuid:pk>

For armor, equipment, races, weapons, spells, and classes we have views for the corresponding list of all of that thing, and for specific things, for example to view all armor we have an ‘armors’ view mapped to ‘armors/’ and an ‘armor-detail’ view for a specific armor identified by its name (string pk, meant to be user specific but we didn’t get around to it) so for an armor instance called “Chainmail” the url to view specifics about that armor would be ‘armor/Chainmail’. All other of these ‘thing’ views are not user specific because they do not have any creation, deletion, or update forms like armor does.

The character list is basically the only account changed view. Character's are meant to be shareable so they aren't actually locked to a specific user account. We had some issues with saving (explained at end of Authentication section), so unfortunately anyone can edit and save any character.

Authentication: Users are authenticated using Django's authentication system like in the homeworks and a user account basically just saves characters to an account. This means that logging in just changes what characters are viewable (only their own) on the character list page. This affects what characters are displayed on the character list page and what is changed when they create a new character.

When not logged in at all the header bar does not display the "Characters" link and you can not view the character list page, or create characters. But you can view characters with the link for that character (and also edit, which is a bug explained further below)

There is currently a problem with being able to edit characters that you don't own. On some of our computers (lubuntu) there is a json typeError problem where things that should be strings are passed as bytes objects when run on lubuntu (and maybe other distros) but works fine on Windows. It seems to be a problem with the json encoding (django) for the platforms it breaks on and we don't want to edit django stuff to probably not get it working. For this reason and other similar entanglements, we weren't able to get save working only for specific users.

Team Choice:

For the team choice component we implemented ajax functionality to the character page. This makes the page smoother as leveling up no longer requires a reload. Specifically the asking for spells and health , populating the ability list , and loading/ saving the character all depend on ajax. To implement this the passing of parameters was done through the urls as shown below to the views described

url(r'^ajax/get_health/(?P<stub>[-\w]+)$', views.get_health, name='get_health'),
    Gets the hit die size from a character passes through stub
url(r'^ajax/get_skills/(?P<cname>[-\w]+)$', views.get_skills),
    Gets the skills a class <cname> can be proficient in to limit the choices the user has in the skills popup

url(r'^ajax/get_spells/(?P<cname>[-\w]+)/(?P<lvl>[-\w]+)/$', views.get_spells),
    Gets the spells a class<cname> at level <lvl> can use from the spell_list model
    Limits choices the player has when prompting for new spells

Same issue as getskills

url(r'^ajax/get_cantrip/(?P<cname>[-\w]+)/$', views.get_cantrip),
return s the cantrips a class <cname> can use. Unimplimented
Could not figure out join statements in django so used raw sql query instead
With helper function dictfetchall in views to be able to return result as json

url(r'^ajax/get_features/(?P<cname>\w+)/(?P<lvl>\w+)/$', views.get_features),
Gets the features of <cname> at <lvl> used to determine what needs to be done for that lvl up. Ie what models to display, if spell lots need to be added to, or abilities appended using the feature_type field in the character class features table

url(r'^ajax/get_abilites/(?P<cname>\w+)/(?P<lvl>\w+)/$', views.get_abilites),
Used to retrieve all abilities to append to the ability list on loading the page

url(r'^ajax/test_post/$', views.testing_post),
Used to save the character. is given a json object containing field information

url(r'^ajax/get_character/(?P<id>[-\w]+)/$', views.get_character),
Used to load character by returning the character by <id>

Conclusion

I think overall we learned a lot about how web servers work, how difficult it is to work with other people on software projects, and got a good feel of how design should work compared to implementation. Design isn't the easiest thing, but we learned how hard it is to get the implementation to reflect design and that it's very easy to get ahead of yourself if you're missing implementation details at the design board. Our team mostly worked around the software is hard in groups by having people do almost all of certain parts, which was kinda bad but was hard to avoid mid semester working in separate places at different times. I think it worked out well enough and we did learn enough about the different parts from Mozilla / limited interaction with things other group members made to do it if we needed to though. Due to the large project, specific tools needed for things, and time constraints, we did not really get the site very functional but it shows concepts and I know that we could get it working if we had the time.

I wish we had prior Git knowledge!!! Above all else by miles, before starting the project we should all have been taught/ensured that we knew how to use Git properly and what

is does / does not do. Significant amounts of work were lost because of improperly resolved merge conflicts, and from multiple people working concurrently on the sqlite.db binary and creating incompatible files. I highly doubt that these issues would have occurred had we been forced to do the Git intro and preferably some quizzing (or something to ensure we actually learned) about Git basics/important knowledge. In particular, that binary files are not actually mergeable like text files are, Git pushes do not make backups every time, particulars of what exactly branches are, about stashes, and how merges work. I think people learned it as they went thinking it'd be fine since it wasn't covered too much in class, but it really screws people over when mistakes are made.

Also it might be useful to encourage use of VBoxes so that we don't have to worry about if the TA's will have certain platform-specific issues or not.