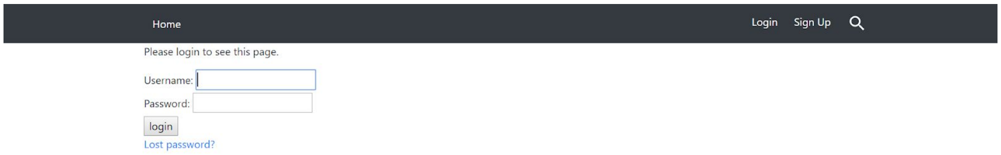
TEAM UBER
NoteTakers
Spring 2018

Overview:
Our application aims to bring the social media aspects of popular sites such as Reddit and Youtube and implement them in a new way for student note sharing. NoteTakers is a hub of college student activity, where students are able to upload their personal notes for their peers, and even for their own later reference.

Team Members:
William Sullivan, Kyle Toohey, Zachary Safir, Jack Bogle, Chigozie Ikoro

Github Repository:
https://github.com/willbond1/CS-326-Project

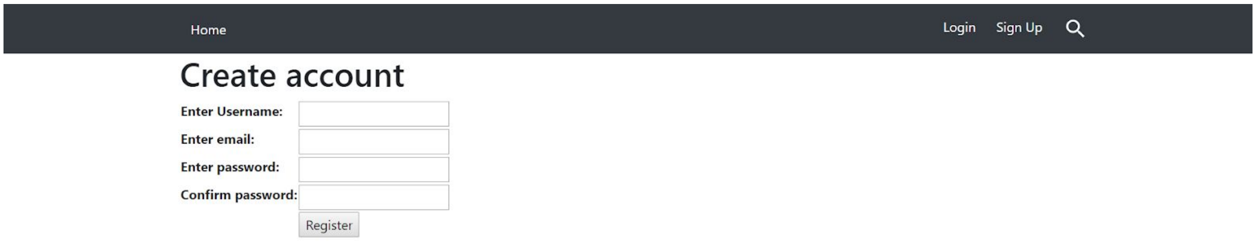
User Interfaces:
Login/Signup Pages: Lets the user sign in or create a profile to use on website.



Please login to see this page.

Username:

Password:



Create account

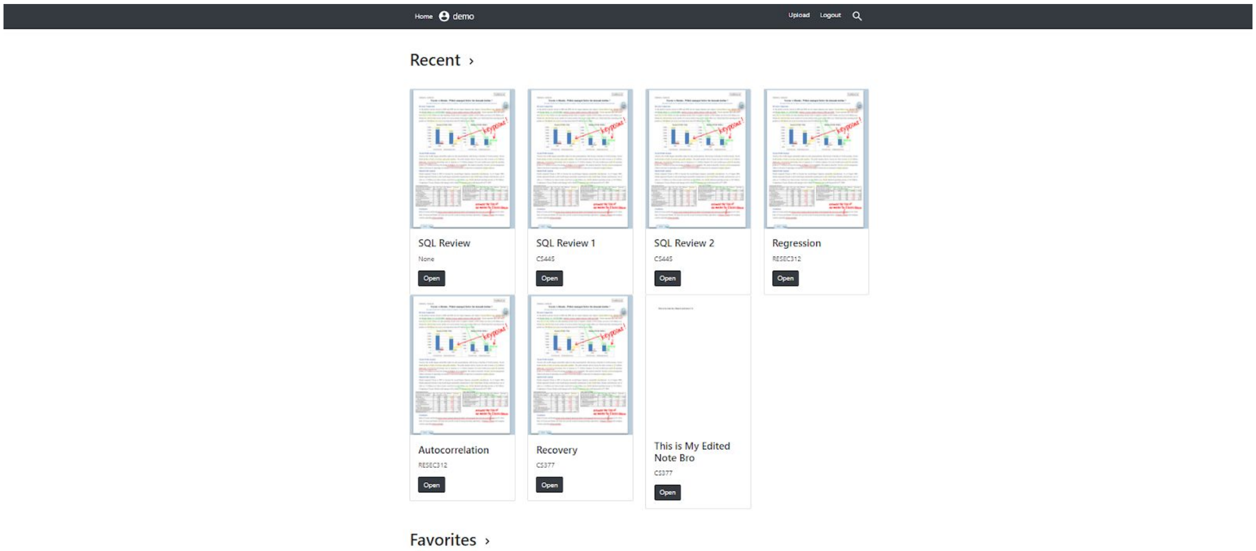
Enter Username:

Enter email:

Enter password:

Confirm password:

Homepage: Lets the user access recently viewed notes as well as any favorited notes. Made to be main point of reference for the user.



Recent ›

SQL Review
None

SQL Review 1
CS445

SQL Review 2
CS445

Regression
RESEC312

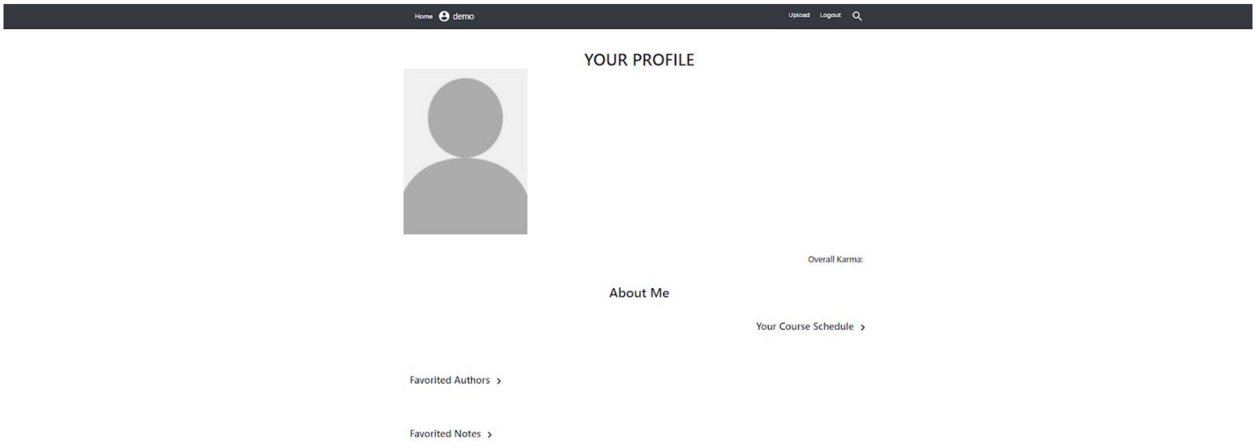
Autocorrelation
RESEC312

Recovery
CS377

This is My Edited Note Bro
CS377

Favorites ›

Profile Page: Hub for each user, displaying personal information such as favorite authors, notes, and a profile picture.



YOUR PROFILE

Overall Karma:

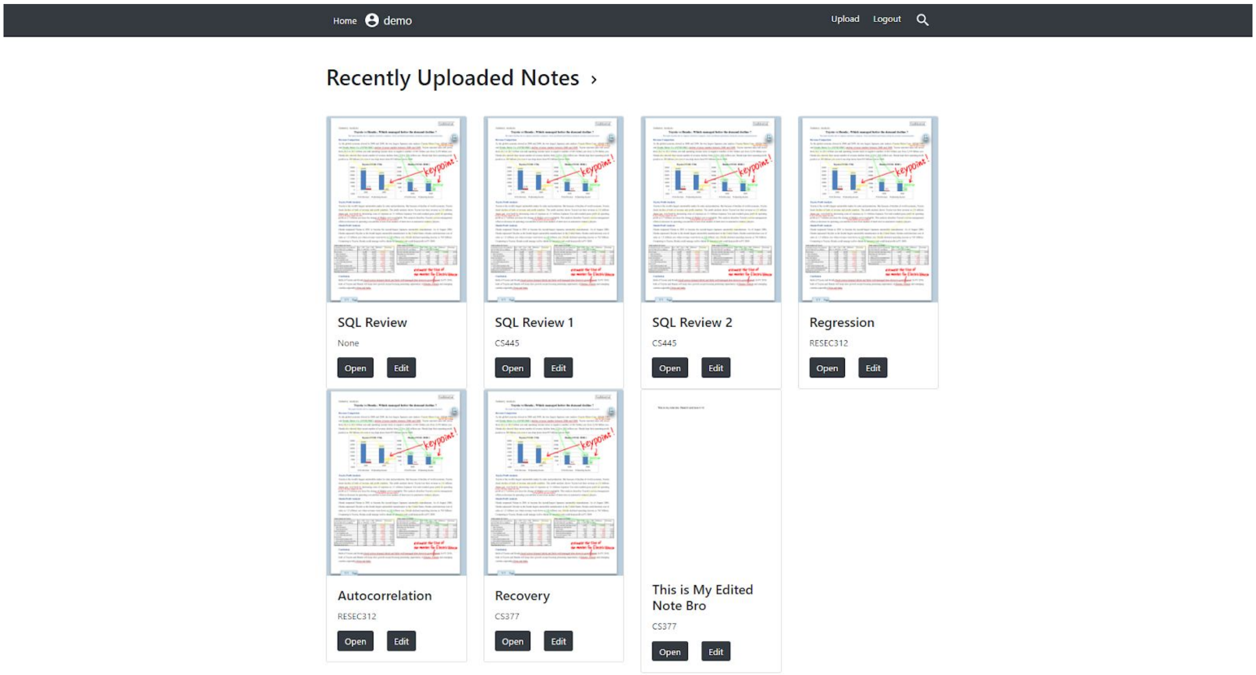
About Me

Your Course Schedule ›

Favorited Authors ›

Favorited Notes ›

My Uploads: Page that lets the user view all of their uploaded notes in list format. User can freely view or edit their own uploaded notes.



Recently Uploaded Notes ›

SQL Review
None

SQL Review 1
CS445

SQL Review 2
CS445

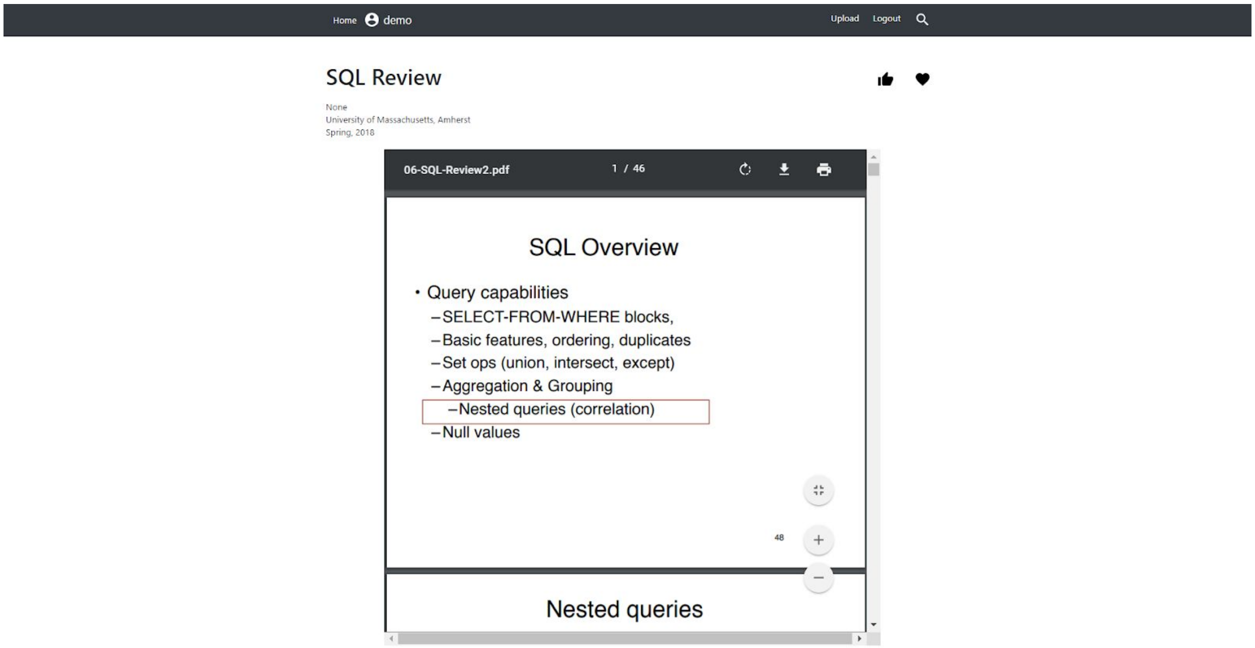
Regression
RESEC312

Autocorrelation
RESEC312

Recovery
CS377

This is My Edited Note Bro
CS377

Note View Page: Page that lets user view a note in PDF format.



SQL Review

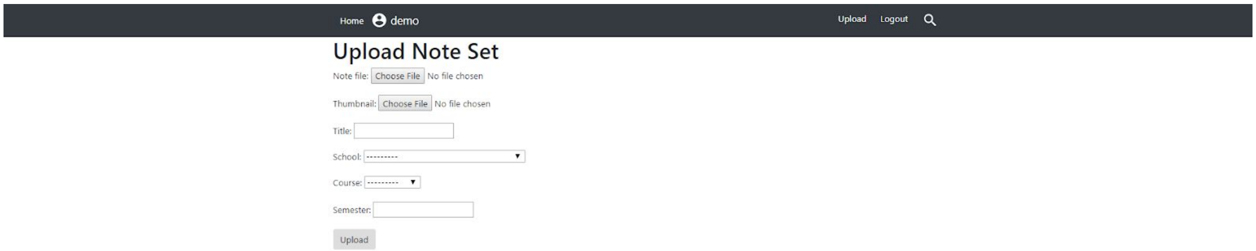
None
University of Massachusetts, Amherst
Spring, 2018

SQL Overview

- Query capabilities
  - SELECT-FROM-WHERE blocks,
  - Basic features, ordering, duplicates
  - Set ops (union, intersect, except)
  - Aggregation & Grouping
    - Nested queries (correlation)
  - Null values

Nested queries

Upload Note Page: This page lets users upload a note to their profile for users to view. All aspects of form are mandatory as of 4/30/18.



Upload Note Set

Note file: Choose File No file chosen

Thumbnail: Choose File No file chosen

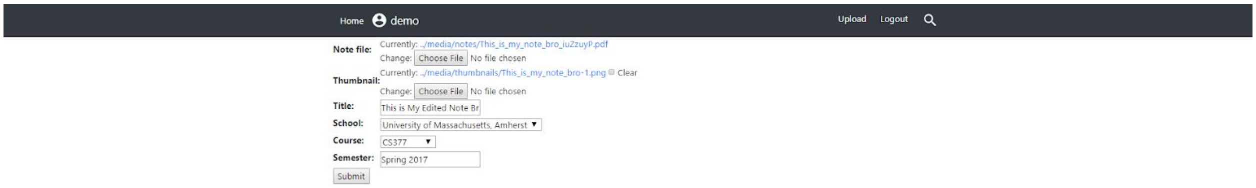
Title:

School: ---------

Course: ---------

Semester:

Edit Note Page: This page lets users edit a previously uploaded note.



Note file: Currently: ../media/notes/This_is_my_note_bro_iuZzuyP.pdf
Change: Choose File No file chosen

Thumbnail: Currently: ../media/thumbnails/This_is_my_note_bro-1.png □ Clear
Change: Choose File No file chosen

Title: This is My Edited Note Br

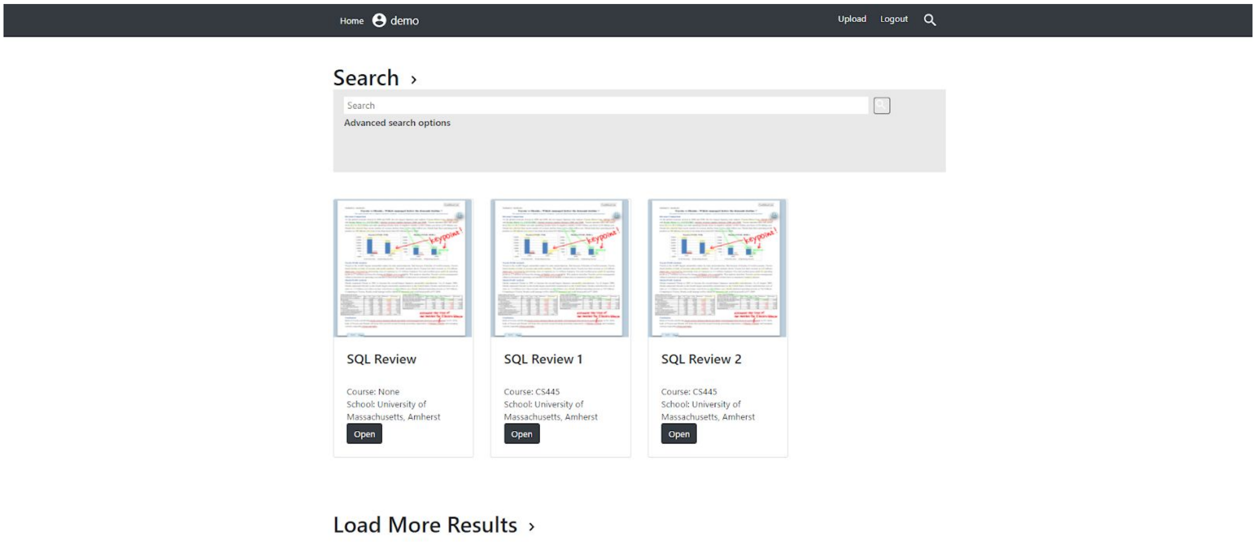
School: University of Massachusetts, Amherst ▼

Course: CS377 ▼

Semester: Spring 2017

Submit

Search Page: Lets the user search by note title.

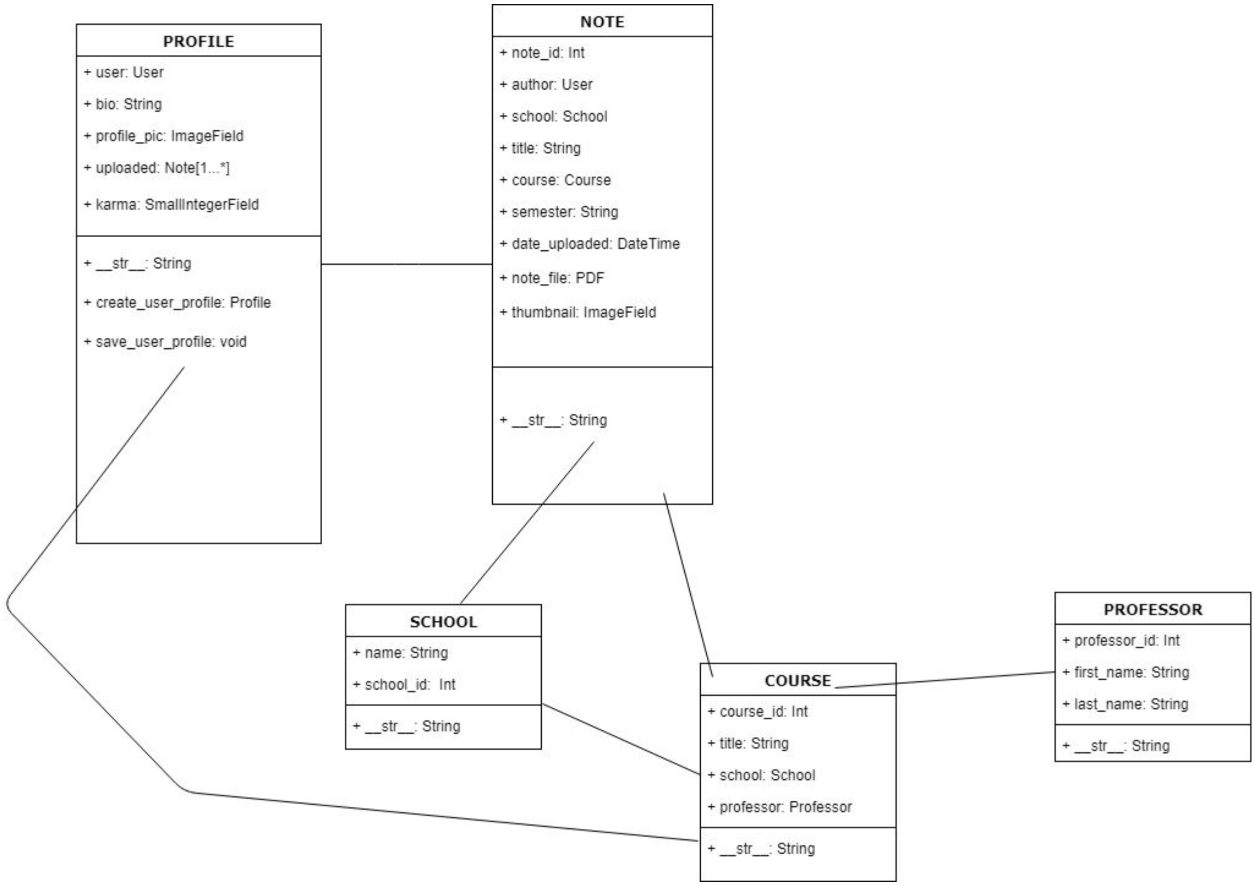


SQL Review
Course: None
School: University of Massachusetts, Amherst
Open

SQL Review 1
Course: CS445
School: University of Massachusetts, Amherst
Open

SQL Review 2
Course: CS445
School: University of Massachusetts, Amherst
Open

Data Model:



Profile: Stores User information and is unique across different users of the application. Associated with Django’s inherent User model.

Note: Stores all relevant metadata pertaining to a note file that is uploaded to the database.

School: Stores data about Schools used by Notes.

Course: Stores information pertaining to the Courses associated with notes.

Professor: Stores info about professors associated with courses.

URL Mappings:

URL Path	View Direct To	Authorization Required
''	Dashboard View (This URL is Used to direct users to the dashboard view when initially logging in.)	No

'note/<int:pk>'	Note Detail View (Used to view specific note as well as certain metadata about note.)	No
'uploaded/'	Note Upload View (Used to view all uploaded notes of specified User.)	Yes
"upload/"	Note Upload Form (Used to have a user upload their own note.)	Yes
“signup/"	User SignUp View (Used to have user signup for a account.)	No
'profile/'	User Profile View (Let’s users view relevant info on their profile.)	Yes
'search/'	Search View (Allows users to search for specific notes based on queries.)	No
'search/results'	Search Results View (Returns results of specified search to user.)	No
'update/<int:pk>'	Update Note Form (Allows users to update already uploaded notes.)	Yes
'profile/<int:pk>/update/'	Update Profile Form (Allows users to update their info for their profiles.)	Yes

Authentication/Authorization:
Users are authenticated by logging into the website using their username and password. The password is encrypted through Django’s User API. An authenticated user has access to all of the site’s features such as note uploading and profile editing. An unauthorized user will have restricted access to the website. Unauthorized users will be able to freely search and view notes, however they will not be able to upload their own notes for other users to view. They also cannot create and edit a profile page.

Team Choice:

Our Team's Choice pertains to extending Django's User model with our own custom model (Profile). We wanted to do this since we needed a way to associate data such as uploaded notes to each user we have. By giving each User a related Profile data model via a one to one field, we were successfully able to create the link needed for us to proceed.

Conclusion:

Designing NoteTakers was a learning experience for everyone involved. Throughout the semester, we were able to learn how to use new tools such as Github and Django which were both essential to the completion of this project. We also communicated with each other well for the most part both inside the classroom during the group meetings and outside of the classroom via Slack. We all also were able to work with Javascript, HTML, and CSS to create webpages for our site.

In terms of difficulties, initially it was hard to get used to using Github, as a majority of the team had no prior experience with using Github collaboratively. We also ran into numerous issues in creating our Django models and making sure they all were working the way we expected them to. Extending Django's User model specifically caused some grief because that idea came towards the end of development and as such, there was no clean way to implement it in a way everyone could agree upon for a long time.

If we could do the project again, our team would definitely liked to know how to properly use and extend Django's User model before starting the project. This would have prevented some of the authorization issues we had run into during the tail end of development. The team also would have liked to know about how to properly add attributes to models.py files before doing the project. During the project, new attributes were added to the Profile model after the model had been initialized, which caused many problems for awhile. The solutions lay in the fact that models must be completely recreated from scratch if new attributes were added to them.