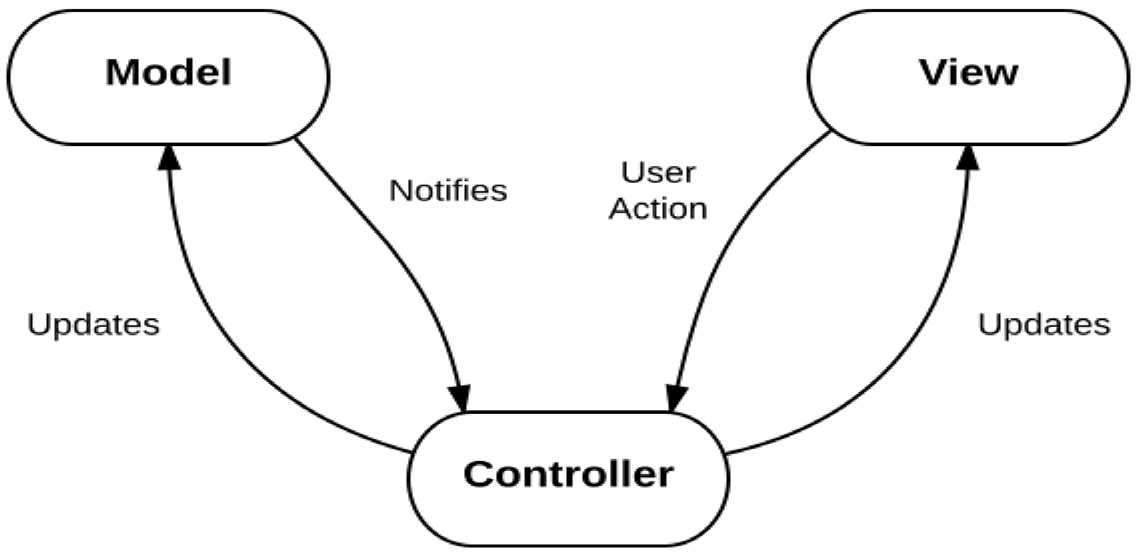
1. High-level system architecture

Architectural Design Pattern

- Application will use the MVC (Model-View-Controller) Pattern to reduce coupling.
- A query file will be our model. The model will take data from the user or MySQL and manipulate that data to be to viewed or stored. The model should validate any incoming/outgoing data constraints. The model will also notify the controller for updates in the database.
- Pug files will be our view for our application’s server-side/client-side rendering. Pug is a template engine, and it gets data from the controller and sends it to the browser, filling out dynamic content for the user to view. This is where the client can interact with the server.
- ExpressJS will be our controller that processes a user’s request and communicates with the model. The controller will also take data directly from model and send it back to the user. The controller will also send user data for model to store.

MVC Model



Database Organization:

[*Put a pic of the data schema and a short description here*]

Media Storage:

We will be using Cloudinary for media storage. Cloudinary is a platform that uses Amazon’s Web Service for media storage for backend image hosting. Their API is specifically focused on efficient content delivery for websites.
[Needs Anthony’s written approval]

Search/filter architecture and implementation:

The search algorithm will use Sequel’s “LIKE” command to find any related characters from the query string that matches with the attributes city, address, and zip code.

Non-trivial algorithm:

We may integrate some machine learning APIs to return some suggested or related views to the user.