Github Repo: https://github.com/UMM-CSci-3601-S20/lab-2-client-server-kyle-aaron.git

1. Gitignore Specifies intentionally untracked files to ignore. Basically it's useful to hide things that could potentially break the build such as files or tests that are not yet implemented within the project skeleton, but may be later.
2. Gradle itself is a 'general purpose build management system'. When run, gradle will create your build environment with the related dependencies. Gradle builds are comprised of at least one build.gradle file which are specific to each project. Projects consist of tasks, which are individual build components e.g. source compilation, downloading remote dependencies.
3. GitHub Actions is used to automatically test a build after it is committed to GitHub.
4. Endpoints (or routes) are used to resolve http url requests and serve the correct html file. They essentially bind url paths to html files.
5. The umm3601Server class sets up a server on port 4567 with the html/javascript content inside of the CLIENT_DIRECTORY dir and the "database" json file in /users.json. It creates an instance of the UserController class, which loads and creates an interface for the json database our server will be serving data from.
    a. The page hocalhost:4567/users shows the users tab as specified in the html file for the page.
    b. Adding the /api/users dups the "database" .json file onto the page by calling the server.get("api/users") method on the server.java.
    c. Adding ?age=25 to the above filters the results of .getusers(ctx) by the age of 25.
    d. The last one matches the specific id to one specified in the .json "database"
6. The 'client' folder holds the css, javascript, and html files used to make the pages when the server is running. Index.html sets up the main page holding links to the other html files for the user to use. Users.html has a page with fields for the user to fill out in order to filter users by company or age, along with a button to initiate the process of the sort (users.js holds the functions that do the actual sorting.) todos.html has a page that is very similar to the users.html file, but this one uses the database of 'todos.json' rather than 'users.json', and has more fields to sort by, such as owner, status, contains, and order by.
7. The users.js file sets up the interface to load data from the users.json file. The users.html page takes input from the client side and calls functions within the users.html file. The information is read from the 'age' field of the users.html page. When the button is pressed, the event listener invokes a call to getFilteredUsers in users.js, which then builds a url with a series of if statements, ignoring the fields of 'age' or 'company' if blank. The server then replies back to the client with an http get request with the constructed url, which returns the filtered json file entries. When the client receives a response from the server, it displays is the 'jsonDump' element specified in the users.html file.
8. The client-side JS is defined in the client/javascript folder. The users.html file defines the javascript by adding a script element whose source is javascript/users.js. The todos.html file defines the javascript by adding a script element whose source is javascript/todos.js. They both have script elements whose source is javascript/util.js.