<Todo>

Implementer: Dylan Gaines

API name: Todo

# Problems and Solutions

Problems

* Current Layout and features do not provide functionality of a todo list. Tasks cannot be marked as complete and referenced later, only created or deleted.

Solutions

* Make UI into a more compact, responsive list view.
* Add a status to track Tasks to be done and done tasks.
* Add description to hold important information.

# Requirements (User Stories)

* As a User, I want to have a visible list of completed tasks so I can track my progress and refer to notes from old tasks
* As a User, I want to have more detailed views of tasks so I can store information I may need to reference as I complete it.

# Design

* Data flow is simple:
  + Load information from MongoDB, render it server side and send it to the client.
  + Anytime a task is complete, send the server the status update, the server updates MongoDB.

# Usage Example

* <http://todoApp.com/todo>
  + Renders the todo page using data from mongoDB
  + **Graphical user interface, text, application, email

    Description automatically generated**A put request can be made to the same url. If the request contains a valid ID and status, updates the record. If viewing in the browser, once this request is complete, the task is moved into the Done section

# User Manuals

Simply GET the /todo endpoint once authenticated in order to view the User interface, all other interactions are handled behind the scenes.

# Implementation Details

Added endpoint for todo API that accepts PUT/GET requests. Used Bootstrap for front-end redesign, implementing data status updates was handled via AJAX requests and MongoDB system updates. Updated form fields and loading using EJS and AJAX for edit, detail and add endpoints.

# Test

* Was not able to properly develop a test (was going into todo.test.js
* While testing the HTML document’s contents is difficult because they’re generated via ejs, testing it would involve checking the response’s content type.
* For the put requests to the todo endpoint, requests would be made for requests that are invalid for multiple reasons (invalid ID, invalid value, invalid content-type, etc)
* All of this would be implemented via supertest and jest’s asynchronous testing framework.

# Deployment

1. GitHub URL: [*https://github.com/jmlpamonag/ASE-285-final-project*](https://github.com/jmlpamonag/ASE-285-final-project)