Sample Solutions

1.

- Models: a user and a todo model are needed to provide the schema and ensure persistent storage of the user data and todo list.
- Controllers: then a user controller needs have functions to support user registration and user authentication. A todo controller is needed to search the database and render a view to display a list of todos, when the user is successfully logged in.
- Views: the view files will have forms and hyperlinks to generate the correct HTTP actions. Submit a form should be POST, a hyperlink can be associated with get or delete action.
- Routing: these HTTP verbs combined with the URL are routed to the relevant controller functions for retrieving related data from the database and sending to the view files for rendering and displaying results back to the user.
- Overall, GET request sent to server, the default route of a home page is the authentication page, which is constructed and sent to the client. The client then enters their email and password which is hashed and salted, and sent as a javascript object to the server using a post request to login. This matches a route and the JSON object is turned into a query using mongoose, which authenticates the user by looking for matching username and password. A Mongo query is made to get all the todo jobs of that user, which is returned as a JS object. This object is passed to a template (pug/jade/ejs) and rendered into a html page which is then sent to the client.

2.

CSS linked file (external style sheet), CSS in the head (document-level style sheet) and inline CSS.

Linked file most reusable and maintainable. In the head is limited to a single page and for inline, you can see its effect in context, but style is not separated from content.

3.

- Document object model: language neutral platform and language neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.

- Javascript object notation: a lightweight data-interchange format in plain text format that uses conventions to represent key value pairs.
- Create read update delete: standard database actions to insert a record/document, retrieve/update and delete records based on certain criteria.
- Cascading Style Sheet: a set of style rules that affects the presentation in the rendering of a HTML document.
- Representative State Transfer: an architectural style and approach for the web that describe interactions with web based resources.

4.

- Replaces all(most) gender specific terms with gender neutral terms.
- A closure, a function that takes the given map and applies it to strings.
- he'll will not be mapped.

5.

- listener registration: addEventListener("keydown", eventHandler, false)
- event flow - capture target bubbling: 1) In the capturing phase each node from the document root to the target node, in order, is examined. If the node is not the target node and there is a handler for that event at the node and the handler is enabled for capture for the node, the handler is executed. 2) Then all handlers registered for the target node, if any, are executed. 3) In the bubbling phase each node from the parent of the target node to the root node, in order, is examined. If there is a handler for that event at the node and the handler is not enabled for capture for the node, the handler is executed. Some event types are not allowed to bubble: load, unload, blur and focus among the HTML event types.

6.

- app is the setup and configuration

- package.json record node modules scripts etc.
- views stores jade/ejs templates for building pages
- public stores styles sheets scripts images etc
- routes matches urls to javascript functions that process requests and build responses.

7.

- PUT (updates individuals and collections)
- POST (inserts individual(s))
- GET (requests an individual page, or a list of individuals)
- DELETE (removes an individual, or drops a list)

8.

something like....

```
var review = new mongoose.Schema({
   author: {type: String, required: True},
   date: {type: Date, required:True},
   stars: {type: Number, required:True, min:1, max:5},
   review: {type: String}
   });

var movie = new mongoose.Schema({
    title: {type: String, required:True},
    year: {type:Number, required: True},
    director: {type = String},
    synopsis: {type: text},
    cast: {type:[String]},
    reviews: {type: [review]}
    });
```

9.

```
extends layout

block content
  if (!user)
    a(href="/login") Login
    br
    a(href="/register") Register
  if (user)
    h1= title
    h3 Welcome #{user.name} #[a(href= '/logout') Logout]
    each movie in user.movies
      .movie
        li= movie.title
        -var n = 0
        -var stars = 0
        each review movie.reviews
         if review.author=user.name
             n = review.stars
                - break
        while n < stars
            n++
            img(src='star.jpg')
```

10.

SPA does client side rendering using angular/react etc to render client side models. Two way data-binding synchronises the client side model with the server side database via an API.

A web app hat uses server side rendering receives requests from the client and constructs an html document for the client to display, and delivers the entire document to client for every request.

SPA's can take longer to load but are more responsive and put less load on the server. Navigation can be challenging too (reloads etc).

SSR is faster initially, but puts more load on the server.