Milestone 4 - Routes EJS, and MVC: Database Integration

This milestone is to be completed individually. No group work is allowed.

This milestone is intended to familiarize you with using Databases to persist the data in your web application.

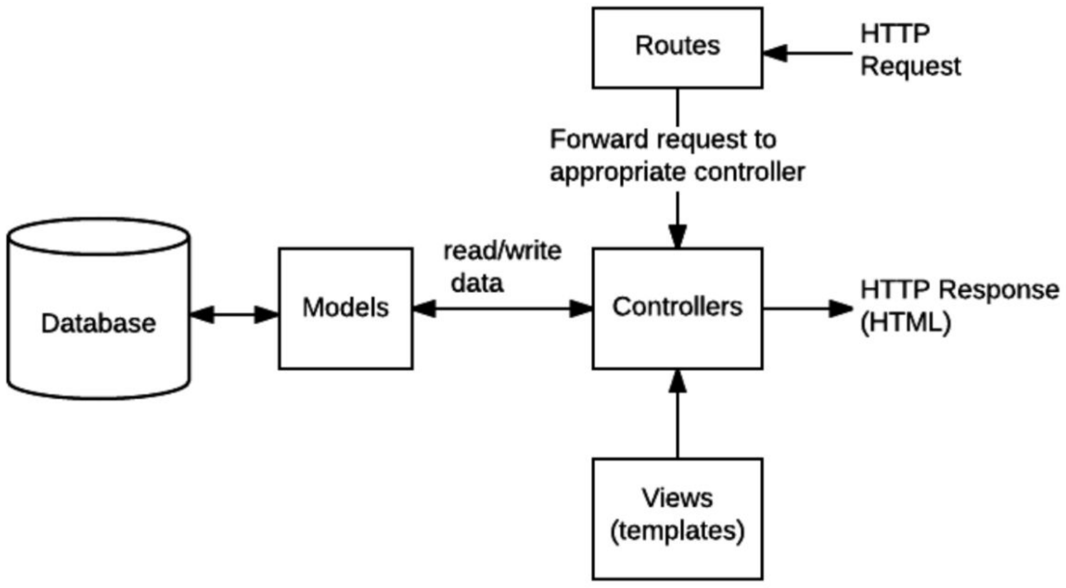
<u>1. Milestone Description:</u>

In this milestone, you will replace the hard-coded "database" with an actual database which entails some modifications to your previous working on your Node.js web application, according to the following specifications:

1. Correct any errors identified or partial/missing functionality from the previous milestone.

If your application developed in milestone 3 has error and not complete it is your responsibility to communicate and coordinate with course staff to help fix those bugs. Keep in mind that you are building in this application in stages. It is important that you have a working application for each stage.

2. All structure, design, and content requirements from previous milestones are mandatory, unless explicitly updated in this milestone description.
3. Use JavaScript to implement the business layer of the application (**<u>model</u>**).
4. Use EJS pages to present the **<u>view</u>** to the browser.
5. Use routes to **<u>control</u>** the flow of the application.
6. Functionality that does not follow the milestone specifications will not receive credit.



2. Database Creation

In addition to the standard project folder for this milestone, you will submit a text file (plain text, NOT Word or RTF or PDF or any other fancy document format). This file will contain all of the MongoDB queries that you use to (1) create, and (2) populate your database. Your database script file must be called:

milestone4_createDB.txt

Create a database in MongoDB to hold your application data. Create the necessary documents and fields to store the information needed in the application (e.g., users, connection, userConnection).

3. Database Population

Your database should be populated with data according to the same requirements from the previous milestones (at least two topics of connections, with at least 3 connection per topic/category).

Within the same script file for database and table creation (milestone4_createDB.txt), add statements at the end of the script to insert your initial data into each of the two tables for connections and users (see #6 for adding a user).

Running the statements in milestone4_createDB.txt script should re-create / reset your database to its initial state with initial data (this means that at any point we can revert back to the beginning as a starting point).

4. Code Changes/Additions

In the following descriptions, data types representing User or Connection fields (code, name, etc) are representative. You may need to adjust some of the functions/methods, depending on the data types you have previously chosen for those fields in your objects. The object types of Connection and User should remain the same.

ConnectionDB - Refactor the "ConnectionDB" module to retrieve connections from the database:

- getConnections() - this function returns an array of Connection objects of all the connections in the connections table from the database
- getConnection(connectionID) - this function returns a Connection object for the provided connection code

UserDB - create the module "UserDB" to retrieve a user from the database:

- getUser(email/userID) - this function returns a User object for the provided user identifier

UserProfileDB – create the module "UserProfileDB" to save and retrieve connections to users and save any feedback they give for connections in the database. This module should provide/support the following functionality:

- retrieve all connections a user has RSVP'd to (a list of UserConnection objects)
- add/update a user RSVP to a connection
- add a new connection to the list of available connections

Sample design:

- getUserProfile(email/userID) - this function returns a collection/list of all UserConnection objects saved
- addRSVP(connectionID, userID, rsvp) – this function associates a connection to the user with userID (when the user saves the connection) and updates the RSVP that user provides for a connection with this connection ID. The corresponding database document can use the userID and connectionID as unique identifiers (keys).
    - **Note**: you need to be able to distinguish between connections that belong to the user (they shared it) vs. connections they RSVP to. If you have not done so already, add a user ID field to each

connection in the database. This will allow you to tell which users own which connections.

- updateRSVP(connectionID, userID, rsvp) – this function updates the rsvp field for a user with userID for connection with connectionID.
- addConnection(connection) - this function handles the new connection form and adds a new connection to the list of connections in the database.

You may include support modules and functions as you see needed. You can also combine functions

5. Changes from Milestone 2

- Use the updated “ConnectionDB” module to retrieve connections form the database.
- Make the necessary changes throughout the project to use this module to retrieve connections from the database.

6. Changes from Milestone 3

- Add at least one user to the database (this is done directly in the database not using the application.
- Use the “UserDB” module to retrieve a user from the database once a user completes the login form (you don't need to validate the entered data. You can use the same user saved in the database for all logins/sessions).
- Use the "UserProfileDB" module to retrieve and update the user connections/ profile from the database. This will require refactoring the functions in the UserProfile module to utilize this module/functions
- Use the "UserProfileDB" class to create and save a new connection to the database. Once user creates a new connection the RSVP value should be "yes". This will also mean that they are the host of that connection/event

Don’t forget that you may also include as many support functions as you might need to make the application work with your design and implementation.

What to submit using Canvas (Email submissions will NOT be accepted):

1. **Milestone4.zip** - An archive of the entire web application (project) stored in a standard ZIP File, you must ensure that all the files are included as part of the archive.
2. **milestone4_createDB.txt** – text file to create and populate your database tables.

3. **Milestone4Info.pdf** – PDF document with the following milestone information:
    1. Screenshot of the application’s home page displayed in your browser.
    2. Explanation of additional features, if any.
    3. Explanation of status, stopping point, and issues if incomplete.
    4. Discuss the easy and challenging parts of the milestone. How did you overcome all or some of the challenges?