Internal Message System Final Project

**This final project is copyrighted and may not be shared or posted online in any format! The solution, written by you, is a derivative work of this project and is also copyrighted. It may not be shared, posted or made available in any form, other than to your GitHub repository and production server for the purposes of this class.**

This project is intended to stretch you and be a thorough review of all concepts introduced in the course. You are encouraged to work with your learning team to finish the project, but all work must be your own. No late submissions will be accepted!

The project must:

1. Use an MVC approach for the solution.
2. The database table design is furnished, but will need to be built and added to the existing class database.
3. Build an internal messaging system, which has the following functionality:
4. The active clients who can work within the message system are those created in the JWT Authorization activity: Basic Client, Happy Employee and Manager User. These three should be the only registered accounts used in the development, testing and use of this system. It is recommended that all other accounts be deleted to avoid confusion.
   1. Each of the three accounts has the ability to create an internal message and address it to another registered account.
   2. When the account logs in, they are notified of the number of unread messages waiting for their review.
   3. An account can mark a message as read or leave it as unread. They can also archive the message to remove it from the inbox, or delete it. If archived, the message remains available, via an “archived messages” option in the inbox.
   4. An account can reply to a message.
   5. Each account is restricted to their own messages. There is no super account with access to all messages.
5. System requirements
   1. When creating a new message, only the account to whom the message is being sent, a subject and message body are available in the interface.
   2. When a new message is stored, it should include the three items from the view, but should also store the date and time the message was created and who created the message. The “read” and “archived” options in the database table should both have a default value of “false”. The primary key of the table should be auto\_incrementing, so that a value does not have to be added manually.
   3. Inbox and archive should display the subject, message date and sender, as well as if the message has been read or not.
   4. A displayed message, for reading, should display the sender, the date and time created, the subject, the message body and options to delete and archive the message.
   5. A message (initial or reply) should not be send-able unless the recipient, subject and body are provided. This must be enforced through client-side and server-side validations.

Table name: message

Table structure:

|  |  |  |  |
| --- | --- | --- | --- |
| column\_name | column\_default | null | data\_type |
| message\_id | Identity | NO | integer |
| message\_subject |  | NO | character varying |
| message\_body |  | NO | text |
| message\_created | now() | NO | timestamp with time zone |
| message\_to |  | NO | integer |
| message\_from |  | NO | integer |
| message\_read | FALSE | NO | boolean |
| message\_archived | FALSE | NO | boolean |

**Grading Matrix** - Consult the grade book for objective values

**Objective 1**

* All project views meet the requirements of the frontend checklist.
* Views are laid out in a professional, useable manner.
* Message form inputs are sticky.
* Success and error messages are styled and easy to discern.

**Objective 2**

* Messages are limited to the three designated accounts.
* Message actions: create, send, reply, archive, mark read, delete, unread message count and archive message count are fully functional.

**Objective 3**

* An MVC solution has been created.
* Routes use middleware to limit access to logged-in clients, and function correctly.
* Controller contains all needed logic and function correctly.
* Model contains all database interactions, returns results to controller and functions correctly.
* External server-side validation is applied correctly to routes.

**Objective 4**

* Message table was created and stores all message traffic for the application correctly.
* Message functions in the model use a prepared statement approach and function correctly.

**Objective 5**

* Client-side and server-side validation is present and functions correctly.
* Errors, when detected, are returned and displayed correctly.

**Objective 6**

* The zip file of the CSE Motors application is complete and submitted on time.
* The production URL is submitted on time and operational.