|  |  |  |
| --- | --- | --- |
| Icon  Description automatically generated | **CMPS 350 Project Phase 2** | |
| **Group Id:** | | **t6-ahmed-hani-syed** |
| **Group Members:** | | Ahmed Abdelhamid (201907476)  Hani Jassin Jafer (201908619)  Syed Zubair (201708606)  **Emails:** aa1907476@student.qu.edu.qa; hj1908619@student.qu.edu.qa; sz1708606@student.qu.edu.qa; |

## Key Lessons Learnt

* Avoiding Code duplication
  + In our project there are several places where we have used the same code in multiple places thus causing code duplication where it could have been avoided. For example, we copied the same navbar code in each of the html pages. This created unnecessary addition of the same code in all the pages. A better way to implement this would have been to just create a separate html file with the nav bar code and inject this code dynamically in all the pages inside the nav tag.
  + Furthermore, on the JavaScript side, we also had some small functions defined in every file where we used it. What could have been done was to create a common JavaScript file and define all the functions there which were used more than once in the project and import them where needed. This would have greatly reduced the number of lines of code written and increased the modularity alongside as well.
* Importing only what is needed
  + When importing to get something specific, in our project we imported the whole JavaScript file into the destination file. This is quite redundant and can be avoided simply by just importing the functions which are needed and not functions which will not be used. So named imports should more often be used where it is applicable.
* Keeping things simple
  + To generate new id for the customer, in our project we get all the customers and then added one to the length of the array. This is a very data consuming method to get the new id and would be very memory inefficient if there are many customers. A better method to implement this was to use the uuid library which can give unique id each time their method is called. This allows for efficient generation of unique id and is a much quicker process as each time a id in needed to be generated we don’t get the all the data of the customers.
* Filtering on the database
  + In our code there are instances where we could have only fetched from the database directly the document which was needed by passing the key value pair and assigning to a variable. Instead, we got all the documents in the collection and then checked if the value for a specific key matched and returned based on that. This is not a good method and should be avoided as the filtering should be done on the database itself. For example, when validating to check if user exists in the database for login, we get all the users and check if the user exits by checking if there is the user with such an email, then checking if the passed password matched the password in the database. Instead, we can directly try to get the document by querying the password and the email and returning based on that if it exists.

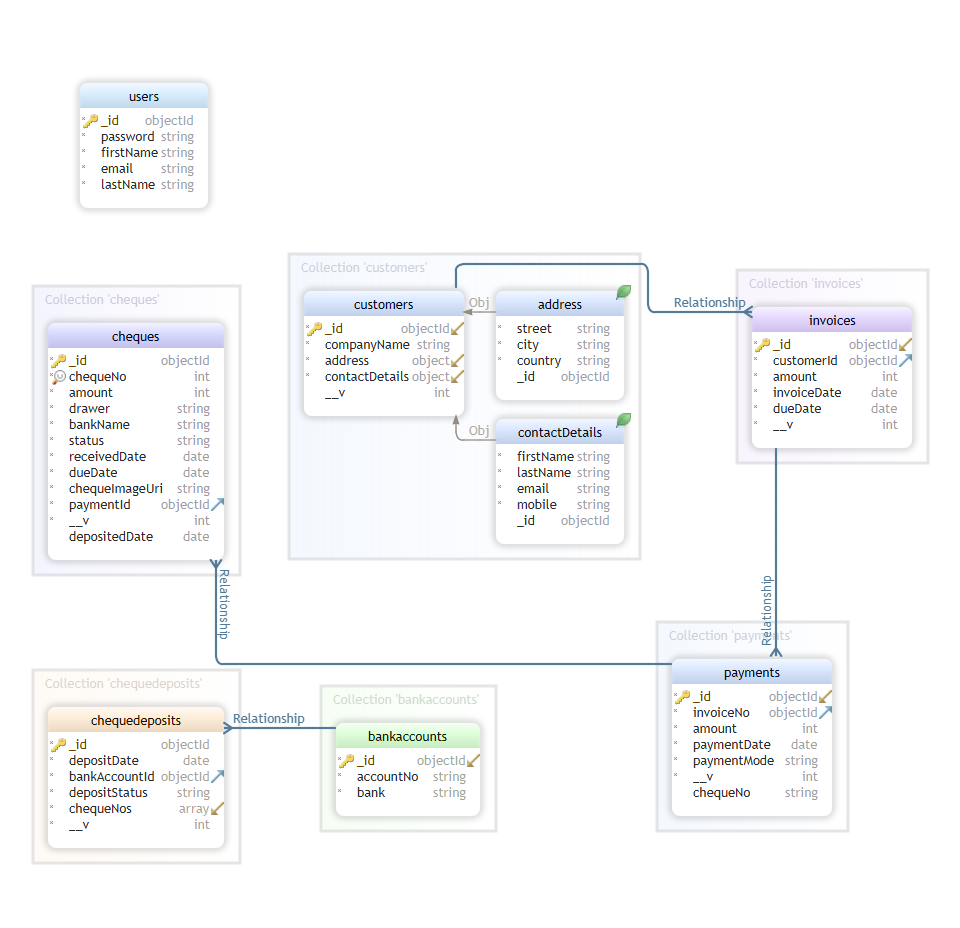
## Web UI Services Diagram

Diagram

Description automatically generated

## MVC Architecture Diagram

## Database Schema Diagram



## Testing

## Project Contribution

* Ahmed Abdelhamid
* Hani Jafer
* Syed Zubair