Uniform Distribution System and Database

evaluation report 14/04/2025

James Bullen

Contents

[Feature Analysis 2](#_Toc195551739)

[Validity 2](#_Toc195551740)

[Reliability 2](#_Toc195551741)

[Scalability 2](#_Toc195551742)

[Requirements 2](#_Toc195551743)

[Performance Analysis 3](#_Toc195551744)

[Query speeds 3](#_Toc195551745)

[Reliability 3](#_Toc195551746)

[Scalability 3](#_Toc195551747)

[Issue Analysis 4](#_Toc195551748)

[Areas of Improvement 4](#_Toc195551749)

[Planned Updates 5](#_Toc195551750)

[Conclusion 6](#_Toc195551751)

[Appendix 7](#_Toc195551752)

# Feature Analysis

## Validity

Minimal validation tests where required in the front end due to the use of combo boxes, eliminating the need for range or datatype test cases and significantly reducing chances for human error. User input fields also have strict validation on their range and datatype. Making the only possible errors being incorrectly formatted names, and accidental size entry due to using the smallest size as the default instead of none which can be picked up on with null validation.

## Reliability

Issues were found during development when using connection pools, where functions would not return the connection to the pool. This often resulted in a crash when a method would try to use empty data from a failed query after a number of calls depending what the pool size was set to. This severally effected the reliability and thus switched the function responsible for calling procedures to a direct connection. While resolving the issue of reliability, will have impact on its performance in scaling in the future.

## Scalability

Due to how the program grabs the relational tables on launch and uses them to generate combo box items, whenever the tables are expanded upon in the database, the front end automatically uses the new additions or changes. This makes the validation of the front end very scalable and easy to maintain through future updates.  
  
The database is also designed with ‘ON DELETE CASCADE’ and ‘ON UPDATE CASCADE’. Allowing changes and expansions to be made to an existing database with minimal interference from leftover legacy data in the system.

# Requirements

There where five requirements lined out for the program, and five met (Bullen J., Appendix, Requirements sheet). The program consistently issues allocated uniform when a new staff is added, with no method to add without issuing uniform. Extra uniform can order and marked as not for reissue, at a limit of 99 per item per order. Valid reissues are detected and can be individually selected as needed to minimise ordering unnecessary uniform.

# Performance Analysis

## Query speeds

The longest duration time from any procedure was 32ms which is well within acceptable response time (Bullen J., Appendix, Fetch Testing sheet). The shortest duration being smaller than the 3 decimal places the tests recorded at (Bullen J., Appendix, Fetch Testing sheet), but by using the profiler we can see that is averages to roughly 0.45ms to execute. For now, no further optimisation is needed necessary until future stress testing proves otherwise.

## Reliability

Random fluctuation in duration time have been found ranging from 15ms±1 to 31ms±1 (Bullen J., Appendix, Fetch Testing sheet). This is believed to be due to factors and influences from the system executing the queries, judging by the inconsistency versus workload. Further testing on different systems not currently available would be required to confirm. The impact is otherwise considered minimal, and no course of action is required.

## Scalability

When scaling table by 100 entries per test case, no reasonable pattern of increase in duration was found across all of the procedures. This suggests that the database either scales very well or increments of 100 entries is not enough to properly measure the database.

# Issue Analysis

## Areas of Improvement

* The visual design of the GUI resembles that of a program from the Win98 era minus. While functional and does not create any issues, it is below current industry standards and expectations from end users. Improving on this would result in a better UX design and satisfaction rate.
* How buttons are placed in the layout and stretch fill various between different popup menus, this is results in a worse UX design. This is a result of different methods of adding the buttons to the layout being used, in combination with how the QT framework auto sizes based on other widgets and columns.
* Documents created to display report tables are barebones in design, giving the suer the bare minimum information to understand the content.
* Hunting down and resolving the issue of connections not being returned to the pool will be needed if the program is to be ever expanded upon to have multiple users interact at the same time.
* If a user wishes to search for a specific staffer or order to check its information, there is no way to narrow down table rows to the specific entry they are looking for.
* Issues in the testing methodology where duration time is only being measured to 3 decimal places, resulting in many zero seconds values recorded. Using the profiler in the MySQL workbench returns to 8 decimal places but also returns a different result to that displayed in the action output. A more accurate and consistent methods need to be identified before more testing. The test cases should also be redesigned to include a ‘Rows Returned’ column to better track variances due to uncontrolled variables in the mock data.

## Planned Updates

* Wrap the project in a Docker package for easer end user deployment.
* A login homepage is the next step, allowing the program to track users, their actions, and permissions to enable other features.
* An ‘Admin’ tab where configuration details to the database connection can be entered and changed, as well as set user permissions. This would allow administrators to switch what database they want to use while the program is live and would remove dotenv as a dependency.
* A custom search widget class to be added to the Staff and Orders tabs layout, allowing for a dynamic search function through table data.
* A new table which will track the actions performed and by which user. This would allow administrators to track down sources of human errors found in the system and potentially revert mistakes or sabotage to the database.

# Conclusion

The program has been found to meet all function requirements set out, provides a GUI with limitations to prevent human error or SQL injection, and executes queries in milliseconds under various loads. With the majority of shortfalls come from the UX designs consistency and lack of visual interest or colour.

The Qt framework which the front end is built off supports CSS stylesheets, which is the main recommended action to take and implement for future versions. Boosting end user satisfaction and ease of navigation finding specific data.

The following planned updates will focus on adding administrator permissions and tools to maintain the database for the long term on a user level.

In conclusion, the program meets all needs for a minimal viable product and can begin end user testing and reviews.

# Appendix

Bullen J. (2025) Change Log. Available at: https://github.com/JamesBullen/Uniform-Distribution-System/blob/main/documents/Change%20Log%20and%20Tests.xlsx