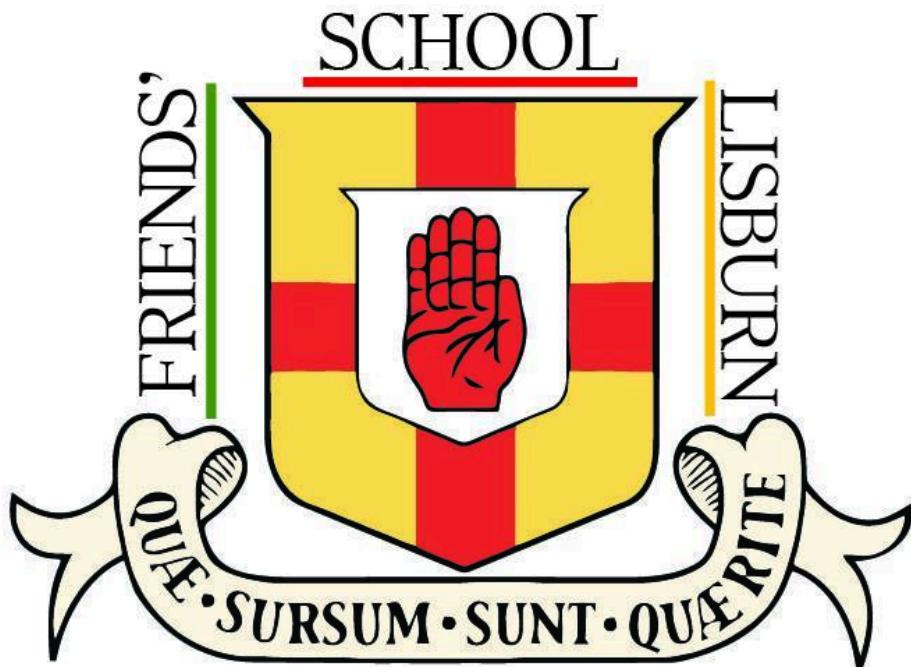


A2 Software Systems Development Movers (2024 - 2025)



Candidate Number: 8346

Contents

Contents	2
Planning	11
Case Study Overview	11
Company Background	11
Business Case for a New IT System	11
Identification of Stakeholders	11
Internal Stakeholders	11
External Stakeholders	12
Microsystem	12
Potential Solutions	13
Application	13
Web Based System	13
Advantages	13
Disadvantages	13
Conclusion	13
Mobile	14
Advantages	14
Disadvantages	14
Conclusion	14
Desktop	15
Advantages	15
Disadvantages	15
Conclusion	15
Flat-File	16
Advantages	16
Disadvantages	16
Conclusion	16
Heirarchical Database	17
Advantages	17
Disadvantages	17
Conclusion	17
Relational Database	18
Advantages	18
Disadvantages	18
Conclusion	18
Project Management Techniques	19
Gantt Chart	19
Advantages	19
Disadvantages	19
Conclusion	19
PERT Chart	21
Advantages	21

Disadvantages	21
Conclusion	21
Design Methodologies	23
Agile vs Linear	23
Scrum	24
Scrum Team	24
Developers	24
Product Owner	24
Scrum Master	24
Parts of Scrum	25
Sprints	25
Sprint Planning	25
Daily Scrum	25
Sprint Review	25
Sprint Retrospective	25
Product Backlog	25
Sprint Backlog	25
Increment	26
Advantages	26
Disadvantages	26
Conclusion	26
Extreme Programming	27
Extreme Programming Lifecycle	27
Rules of Extreme Programming	27
Planning	28
Managing	28
Coding	28
Designing	29
Testing	29
Advantages	29
Disadvantages	30
Conclusion	30
Waterfall	31
Phases of Waterfall	31
Requirements	31
Design	31
Implementation	32
Testing	32
Maintenance	32
Advantages	32
Disadvantages	32
Conclusion	33
User Requirements	34
Functional Requirements	34

Non-functional requirements	38
Project Architecture	40
Model View Controller	40
Components of MVC	40
Model	40
View	40
Controller	41
Advantages	41
Disadvantages	41
Model View Presenter	42
Variants of MVP	42
Supervising Controller	42
Observing Presenter	42
Passive View	42
Advantages	42
Disadvantages	43
Model View View-Model	44
Components of MVVM	44
View	44
View-Model	44
Model	44
Advantages	44
Disadvantages	45
Conclusion	45
Object Oriented Design	46
Classes	46
Encapsulation	46
Abstraction	46
Inheritance	46
Polymorphism	46
Data Access and Storage	47
Microsoft SQL Server	47
Stored Procedures	47
Data Access Layer	47
Integration of Technologies	48
Forms	48
MVP Architecture	48
Risk Mitigation	49
Testing	49
Refactoring	49
Comments and Documentation	49
Data Design	49
Normalisation	50

First Normal Form	50
Second Normal Form	50
Third Normal Form	50
Boyce-Codd Normal Form (BCNF), Fourth Normal Form and Fifth Normal Form	50
Database Normalisation	50
Unnormalised Form (0NF)	51
First Normal Form (1NF)	51
Second Normal Form (2NF)	52
Third Normal Form (3NF)	52
Entity Relationship Diagram	54
Advantages of ERDs	54
Disadvantages of ERDs	54
Conclusion	54
Data Dictionary	56
Uniform Modelling Language	63
Types of Unified Modeling Language Diagrams	63
Behavioural Diagrams	63
Interaction Diagrams	63
Structural Diagrams	63
Advantages	64
Disadvantages	64
Usage of Unified Modeling Language	64
Use Case Diagram	65
Class Diagram	66
Presenter Classes	67
Interfaces	68
Forms	69
Design	70
Storyboards	70
General Styling	70
Colour Scheme	70
Fonts	71
Other Styling	71
Form Storyboards	72
Splash Screen	72
Initial Design	72
Description	72
Feedback	72
Response	73
Final Design	73
Description	74
Sign In Page	75
Initial Design	75
Description	75

Feedback	76
Final Design	77
Description	77
Main Page	80
Initial Design	80
Description	80
Feedback	81
Final Design	81
Description	82
Dashboard	83
Initial Design	83
Description	83
Feedback	83
Final Design	84
Description	84
Add Page	87
Initial Design	87
Description	87
Final Design	89
Description	89
Edit View	91
Initial Design	91
Description	91
Final Design	93
Description	93
Manage Stock Page	95
Initial Design	95
Description	95
Feedback	96
Final Design	97
Description	97
Stock Details Page	101
Initial Design	101
Description	101
Feedback	103
Final Design	104
Description	104
Stock Warning Page	107
Initial Design	107
Description	107
Feedback	108
Final Design	109
Description	109
Stock Quantity Page	111

Initial Design	112
Description	112
Feedback	113
Final Design	114
Description	114
Stock Unit Cost Page	117
Initial Design	118
Description	118
Feedback	119
Final Design	119
Description	120
Manage Stock Quantity Changes Page	120
Initial Design	121
Description	121
Feedback	122
Final Design	123
Description	123
Stock Quantity Change Page	126
Initial Design	126
Description	126
Feedback	127
Final Design	128
Description	128
Manage Orders Page	131
Initial Design	131
Description	131
Feedback	132
Final Design	133
Description	133
Approve Orders Page	136
Initial Design	136
Description	136
Feedback	137
Final Design	138
Description	138
Upcoming Deliveries Page	141
Initial Design	141
Description	141
Feedback	142
Final Design	142
Description	143
Select Order Stock & Select Cleaning Job Customer & Select Cleaning Job Option Page	145
Initial Design	145

Description	145
Final Design	146
Description	147
Select Order Stock Quantities Page & Select Cleaning Job Cleaning Options Page	149
Initial Design	149
Description	149
Feedback	150
Final Design	151
Description	151
Manage Staff Page	154
Initial Design	154
Description	154
Feedback	155
Final Design	156
Description	156
Staff Personal Details & Customer Personal Details Page	160
Initial Design	160
Description	160
Feedback	161
Final Design	162
Description	162
Staff Contact Details & Customer Contact Details Page	164
Initial Design	164
Description	164
Feedback	165
Final Design	166
Description	166
Staff Account Security Page	168
Initial Design	168
Description	168
Feedback	169
Final Design	170
Description	170
Cleaning Job Option Details Page	174
Initial Design	174
Description	174
Feedback	175
Final Design	176
Description	176
Book Cleaning Job Page	179
Initial Design	179
Description	179
Feedback	181

Final Design	181
Description	182
Reports Page	184
Initial Design	184
Description	184
Feedback	184
Final Design	185
Description	185
Report Storyboards	187
Current Stock Report	187
Initial Design	187
Description	187
Feedback	187
Final Design	188
Description	188
Current Staff Report	189
Initial Design	189
Description	189
Feedback	189
Final Design	190
Description	190
Cleaning Job Report	191
Initial Design	191
Description	191
Feedback	191
Final Design	192
Description	192
Pseudocode	193
Levenshtein Distance Algorithm	193
Recursive Implementation	193
Dynamic Programming Implementation	194
Usage	195
Password Cryptography	196
Cryptography Class	196
Usage	197
Assigning Cleaners To Cleaning Jobs	198
Implementation	199
Displaying A Child-View	201
Control Methods	201
Recursive Control Methods	202
Database	203
Connection String	203
Data Access Layer Queries	203
Example: Get Data	203

Example: Add Data	204
Example Update Data	204
Example Delete Data	204
Forms	206
Signing In	206
Password Requirements	208
Dashboard Message	210
Input-Output-Process (IPO)	211
Staff	211
Stock	211
Orders	212
Bookings	212
Customers	212
Cleaning Job Options	213
User Feedback Evaluation	214
User Feedback Form	214
User Feedback Results	216
Question 1	216
Question 2	216
Question 3	217
Question 4	217
Question 5	218
Conclusion	218
Testing	219
Types of Testing	219
Black Box Testing	219
White Box Testing	219
Parts of Testing	219
Unit Testing	219
Integration Testing	219
System Testing	219
Acceptance Testing	219
Data Entry Testing	220
Link Testing	220
Conclusion	220
Test Plan	221
Testing	251
Link Testing Fixes	281
10.3 - Cleaning Manager Menu	281
Description	281
Code	281
Solution	282
11 - Change Password Button Redirect	283
Description	283

Code	283
Solution	283
53.4 - Adding a New Staff Member	285
Description	285
Code	285
Solution	286
32.1 - Stock ID Sorting	287
Description	287
Code	287
Solution	288
76 - Cleaning Job Booking	290
Description	290
Code	290
Solution	290
81 - Navigation Back from Adding a Cleaning Job	292
Description	292
Code	292
Solution	292
Data Entry Testing Fixes	294
26 - Length of Extra Information in a Cleaning Job.	294
Description	294
Code	294
Solution	294
User Acceptance Testing	296
User Feedback	296
User Feedback Form	296
User Feedback Form Results	300
Question 1	300
Question 2	300
Question 3	301
Question 4	301
Question 5	302
Question 6	302
Question 7	302
Question 8	303
Question 9	303
Conclusion	303
Evaluation	304
User Requirements	304
Functional Requirements	304
Non-functional requirements	312
Conclusion	315
Approach	316
Architecture	317

Design Methodology	319
Project Management	321
Testing Process	323
Solution	325
Own Performance	327
Closing Remarks	329
Appendix 1	330
Developer Diary	330
Appendix 2	333
Code	333
Database	333
Tables	333
Staff	333
Login Attempt	334
Stock	334
Stock Quantity	334
Order	335
Order Stock	335
Order Customer	335
Customer	335
Cleaning Job Option	336
Cleaning Job	336
Cleaning Job Cleaning Option	336
Cleaning Job Staff	337
Stored Procedures	337
AddCleaningJobCleaningOption	337
AddCleaningJobStaff	337
AddCustomer	338
AddOrderStock	338
CountCustomersInFutureJobs	338
CreateCleaningJob	338
CreateCleaningJobOption	339
CreateOrder	339
CreateStaff	340
CreateStock	341
DeleteAllCleaningJobCleaningJobOptions	341
DeleteAllCleaningJobStaff	341
DeleteAllOrderStock	342
DeleteCleaningJob	342
DeleteOrder	342
GetCleaningJobById	342
GetCleaningJobCleaningOptionIds	342
GetCleaningJobCleaningOptions	342

GetCleaningJobOptions	343
GetCleaningJobs	343
GetCleaningJobsAfterDate	343
GetCleaningJobsByDate	343
GetCleaningJobStaffIds	343
GetCustomerById	343
GetCustomers	344
GetJobOptionByName	344
GetLoginAttempts	344
GetNonArchivedCleaningJobOptions	344
GetNonArchivedCustomers	344
GetNonArchivedStock	344
GetOrders	344
GetOrderStockItems	344
GetStaffById	345
GetStaffByPrivilege	345
GetStaffByUsername	345
GetStaffCredentials	346
GetStock	346
GetStockBySku	347
GetStockQuantityChanges	347
GetUpcomingCleaningJobs	347
LogLoginAttempt	347
UpdateCleaningJobCleaningOptionQuantity	347
UpdateCleaningJobCustomer	348
UpdateCleaningJobDetails	348
UpdateCleaningJobDetails	348
UpdateCleaningJobOptionDetails	348
UpdateCleaningJobTimes	349
UpdateCustomerArchived	349
UpdateCustomerContactDetails	349
UpdateCustomerPersonalDetails	349
UpdateOrderDescription	349
UpdateOrderDiscrepancies	350
UpdateOrderStatus	350
UpdateOrderStockQuantity	350
UpdateStaffAppearanceSettings	350
UpdateStaffArchived	350
UpdateStaffContactDetails	351
UpdateStaffCredentials	351
UpdateStaffEmergencyContact	351
UpdateStaffPassword	351
UpdateStaffPersonalInformation	352
UpdateStockArchived	352

UpdateStockDetails	352
UpdateStockQuantity	352
UpdateStockWarnings	353

Planning

Case Study Overview

Company Background

“Movers” is a removal company, owned by the Ross family, that specialises in home content relocation. They have expanded to cover a large geographical area with several services, including home clearing, relocating possessions from one property to another (including electrical appliances), providing a range of cleaning options, and assisting students in moving belongings to and from the university. They are also planning to add commercial removals and full office relocation services. To facilitate this, Mr Ross employs 6 administrative staff members, including 1 manager and 1 assistant manager, who are involved in customer enquiries, billing, scheduling, and general administration. In addition to this, there are 40 drivers and assistant drivers who operate Mover’s fleet of 30 vans (8 small, 12 medium-sized, 10 large) that assist with loading and unloading the vans at a service. The vans are maintained by 6 staff who handle regular checks, servicing and repairs. They also manage the training of drivers. Apart from this, there are 8 cleaning staff who work in teams to fulfil the cleaning service. Mr Ross is also considering hiring additional part-time staff to support the cleaning service.

Business Case for a New IT System

For several reasons, Mr Ross has recognised the need for an IT system to help manage his family business. His administration staff constantly deal with enquiries, phone calls and billing. In addition to this, they also need to organise each job from start to finish. Poor organisation of paperwork has led to difficulties for both staff and customers. Reliable pricing, booking, billing, scheduling, maintenance, stock management and efficient staff deployment are nearly impossible to guarantee. In addition, with plans for expanding Movers, Mr Ross needs access to reliable business metrics to help inform his decision. A new IT system to help administration would allow him to maintain a competitive edge and ensure efficiency. This could improve revenue. Detailed analytics, in the form of reports, from the system could also provide insight into Mover’s performance to inform future decisions.

Identification of Stakeholders

The stakeholders with a vested interest in the new system are outlined below. Stakeholders are those individuals and organisations affected by the project.

Internal Stakeholders

The internal stakeholders include Movers’ employees (specifically the cleaners, cleaning manager, admin staff and managers), William Ross and any shareholders

in the company. The employees have a vested interest due to their use of the system to help manage the Movers business.

External Stakeholders

External stakeholders include the government, customers, and suppliers. The government collects company taxes and is influenced by Movers' success. Customers and suppliers are managed internally with the Movers system, which means they should be successfully managed and are affected by the system's performance.

Microsystem

This project seeks to design a microsystem to manage Movers' "Cleaning" service. The system will streamline the booking process and attempt to replace the paper-based booking system fully. It will also address issues such as the lack of analytics and strive to accurately track the stock of cleaning supplies. It will also record the hours each cleaning staff member worked to help calculate monthly bills.

The system will be used by managers (including the cleaning manager and the company owner) and staff in the office. Cleaners themselves will also have a dedicated account on the system.

Potential Solutions

Application

Numerous different software solutions could be suited for designing the Movers microsystem application. Several options are outlined below.

Web Based System

A website could provide a solution for the Movers microsystem. Frameworks like React (within Next.js) or Svelte allow for component-based website development using Typescript or Javascript, HTML, and Tailwind CSS. Alternatively, Electron, a free and open-source framework maintained by Github, also allows cross-platform desktop apps to be designed from web applications.

Advantages

1. Portable: Websites are extremely portable because they can run across devices within a browser. In addition, modern CSS frameworks, such as Tailwind CSS, allow for easy development for various screen sizes.
2. Storage Requirements: Storage requirements are offloaded to the server and can be expanded rapidly.
3. Centralised Updates: Only the server needs to be updated, so client-side updates are non-existent.
4. Experience: I have experience with several web frameworks and have created a website.

Disadvantages

1. Security: Web applications can be compromised by bad actors remotely.
2. Internet: Internet access is vital to allow the Movers' staff to access the system. This means the system will be ineffective if there are any Internet issues.
3. Centralised Issues: If there is an issue with the server, the whole application will be affected.
4. Less Performant: Web applications can be less performant than desktop applications. They can have much higher memory requirements. This is especially obvious with Electron desktop apps, which have a larger installed size and are slower than other desktop development technologies such as WPF.

Conclusion

In conclusion, despite the many advantages and prevalence of web development, I will not use a web application. First, the portability of web applications is unnecessary for the Movers' microsystem because it is a small local company. Second, web applications can have large memory requirements (RAM) that may not be compatible with older computers and could significantly impede performance.

Mobile

A mobile app could provide a solution for the Movers microsystem. Cross-platform mobile apps can be built with technologies such as Flutter, .NET Maui or React Native.

Advantages

1. Ease of Updating: Updates can be rolled out easily if the app is registered on the devices' native app stores.
2. Ease of Access: Mobile systems can be accessed from anywhere due to the portability of devices. Employees could update stock remotely or even record cleaning progress on the worksite.

Disadvantages

1. Security: If the application is registered on an app store, people could download and access it maliciously.
2. Device Compatibility: Different operating systems may have device-specific issues, which could pose a serious risk of cross-platform compatibility issues. Testing the application would also require a wide range of test devices.
3. Experience: I have limited experience making mobile applications.

Conclusion

In conclusion, I will not use a mobile application due to potential device compatibility issues. In addition, updating stock remotely would be impractical, and the new hardware (e.g., iPads or mobile phones) required for the cleaning staff to record cleaning jobs remotely would be expensive. They could also easily be misplaced, which would pose a security risk.

Desktop

A native desktop application could provide a solution. Desktop solutions could be implemented with Windows Forms or the Windows Presentation Foundation (WPF).

Advantages

1. Easy UI Creation: Windows Forms offers an easy-to-use interface.
2. Experience: I have a lot of experience with Windows Forms.
3. Security: A desktop application can run in a controlled environment and is less accessible to web-based attacks. To compromise the system, attackers would have to enter the Movers' office and interact with one of their machines.
4. Long-Term Stability: Windows Forms offers a long-lasting, stable solution.
5. Performance: Direct access to the system resources (i.e. rather than through a web browser) allows for high performance. This could provide a large advantage to the microsystem startup requirements as Windows Forms can offer good performance on old hardware.

Disadvantages

1. Portability: Windows Forms applications can only run on Windows computers, which restricts the types of devices Movers may purchase in the future.
2. Outdated UI: Although Windows Forms facilitates the swift creation of Forms, the default controls appear antiquated.
3. Updates: Updating a desktop application would be tedious. Each machine would have to be updated individually, or another solution would have to be implemented to manage the mover's computer system.
4. User Adoption: Modern users prefer mobile and web applications, so they are less keen to adopt a desktop solution.

Conclusion

In conclusion, I will use a Windows Forms desktop application due to my extensive experience with the Windows Forms ecosystem. A desktop application would be well-suited for the Movers office. It would also offer increased performance compared to mobile and web-based solutions, and it will be supported for a long time within the Windows ecosystem. Finally, many staff members are already familiar with the Windows operating system, allowing them to adopt the new system quickly.

Data Storage Solutions

In addition to deciding a software ecosystem to build the application, it is also necessary to decide on a data storage solution

Flat-File

A flat file is a database stored within a file. Every entry has the same format, and there are no links between data. Flat files can be comma-separated value files (CSV), tab-separated value files (TSV), or simply text files.

Advantages

1. Easy to Understand: Flat-file databases are simple to use and understand, making them suitable for simple applications or storing configuration information.
2. Minimal Overhead: Flat-file databases are quick to read and write. Small portions of the file can be read in at one time for further performance gains.
3. Human-readable: Data within a flat-file database can be quickly inspected and

Disadvantages

1. Data Redundancy: Flat-file databases only have one table, so several entries containing the same information can be created, resulting in data duplication.
2. Limited Functionality: No relationships can be created between entries within a flat-file database. This means that data management and association is cumbersome.
3. Security: Due to the human-readable nature of flat-file databases, they can easily be edited by malicious actors.
4. Limited standardisation: Within flat-file databases, there are many different options with no standardisation. For example, CSV does not have standards for escaping characters, which can lead to issues when reading in data.

Conclusion

In conclusion, I will not use a flat-file database for my primary data storage within the application. I will, however, use flat files for configuration files. These will be XML files, as XML is well documented.

Heirarchical Database

A hierarchical database is a data model in which data is organised into a tree-like structure. The data is stored as records represented by nodes. Each node is linked to its children in a parent-child relationship. Parents can have multiple children, with the top-level parent called the “root node.” Examples of hierarchical databases include MongoDB, a hierarchical document-based database.

Advantages

1. Improved Data Retrieval: The hierarchical structure makes data retrieval easy. It is simple to inquire about records in a parent-child relationship.
2. Intuitive structure: Data in a heirarchical structure is easy to comprehend, making it easier to use.

Disadvantages

1. Inflexible: The inflexible nature of the hierarchical model makes changes difficult without disrupting the structure.
2. Only One-to-Many Relationships: Hierarchical databases can only store one to many relationships, but they cannot use any other associations.
3. Deletions: Removing a parent node from within a heirarchical structure can result in removal of all child nodes.

Conclusion

In conclusion, I will not be using a heirarchical database due to difficulties associated with deletion in a hierarchical database and the restriction of one-to-many relationships.

Relational Database

Relational databases is a database in which data is structured in a format of tables using rows and columns, with relationships between data structures. Each entry within a table is uniquely identified by a “primary key”. Entries can then be linked together through referencing another entries primary keys. An example of a relational database is Microsoft SQL Server. This database provides T-SQL (Transaction Structured Query Language) to help manage database opeations

Advantages

1. Simple Model: Relational databases provide a simple model of data storage, making them easy and efficient to work with.
2. Easy to use: Many relational databases provide SQL to manage data. This makes it easy to update and query data in various forms.
3. Accurate: Relational databases prevent data integrity loss and avoid duplication. This makes data more accurate.
4. Security: Many relational database software solutions provide access management to prevent destructive attacks on the database.

Disadvantages

1. Performance: As a relational database gets larger, performance can decrease.

Conclusion

In conclusion, I will be using a relational database (MS SQL Server) solution for the application. This is due to the benefits of SQL for managing the database and the simple model. In addition, performance issues should be limited as the application will require a small number of tables. I will be using a local database for demonstration purposes; however, this will be later updated and moved to a server when the application is deployed.

Project Management Techniques

Several project management techniques, explained below, were employed to ensure the project remained on target for completion by the deadline.

Gantt Chart

A Gantt Chart is a horizontal bar chart (time on the x-axis and tasks/activities on the y-axis). Tasks are represented by horizontal bars whose length represents their duration. Henry Gantt developed Gantt charts in 1917 to schedule work in factories. Arrows may represent dependencies between tasks; however, this is optional. Milestones, objectives that indicate the completion of a deliverable, may also be represented by a diamond-shaped task of zero duration.

Advantages

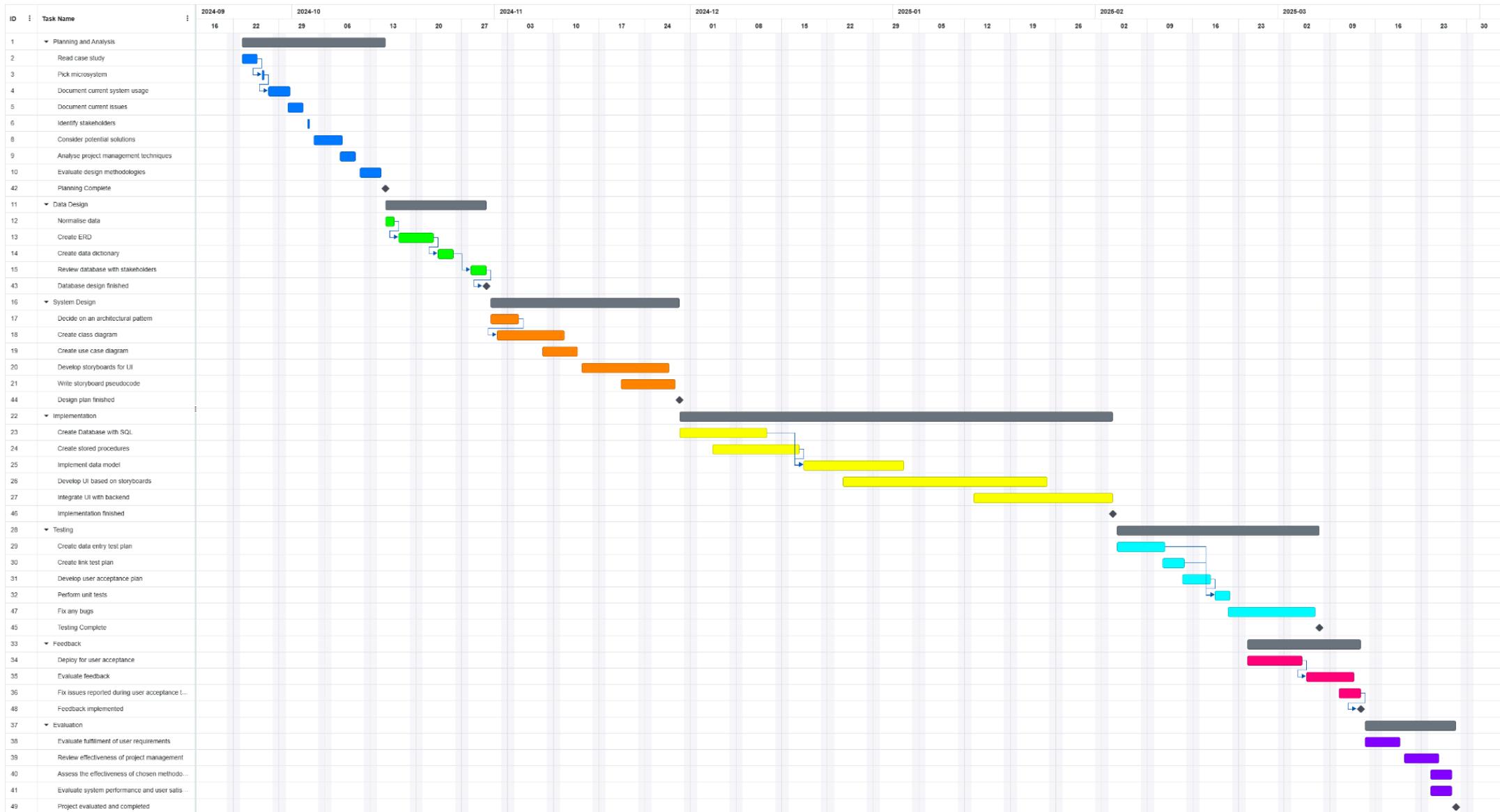
1. Visual Representation: Gantt charts provide a clear visual timeline of tasks. This makes the tasks easily understood visually, which helps communicate the project management plan to the stakeholders.
2. Task Dependencies: Task dependencies can be seen by arrows between tasks.
3. Resource Allocation: Gantt charts allow resources to be allocated efficiently.
4. Progress Tracking: Gantt charts can help manage and record each task's progress. Tools such as [Online Gantt](#) assign each task a progress value.
5. Time Management: They help to set realistic deadlines by breaking down the project into smaller tasks.
6. Flexibility: Gantt charts can be updated during the project development, providing flexibility if requirements change.

Disadvantages

1. Limited detail: Gantt charts do not capture the details of each task and only provide a brief overview through the task name. This limited detail may also mean oversimplification that does not account for unexpected issues.
2. Complexity for Larger Projects: Large projects can become complicated to manage. This could be overwhelming and defeat the objective of breaking the project into manageable tasks.
3. Focus on Time: Gantt charts focus on time, not the outcome, which can misrepresent the quality of task completion.

Conclusion

In conclusion, I will use the Gantt chart as a project management tool to track the completion of the Movers' microsystem. The project is relatively small but has a tight time constraint, so time must be managed effectively. As the sole developer of this project, a Gantt chart also allows me to have a clear overview of all tasks that must be completed.



PERT Chart

Program Evaluation Review Technique (PERT) charts model the relationship of dependencies between tasks. They consist of nodes (boxes) and arrows in a networked diagram. From 1958, the US Navy Polaris missile/submarine project used network diagrams. This approach allows the critical path, a sequence of networked work packages in the path with the least amount of slack, to be easily viewed. A delay in any task on the critical path will delay the project's completion.

A box for each task must be drawn before constructing a PERT chart. These are placed in an ordered sequence. Arrows are then used between these boxes to show dependencies. After this, the duration of each task is entered. From here the critical path is calculated by a forward pass and a backward pass. The critical path is a sequence of tasks for which the earliest start time equals the latest start time, and the earliest finish time equals the latest finish time.

Advantages

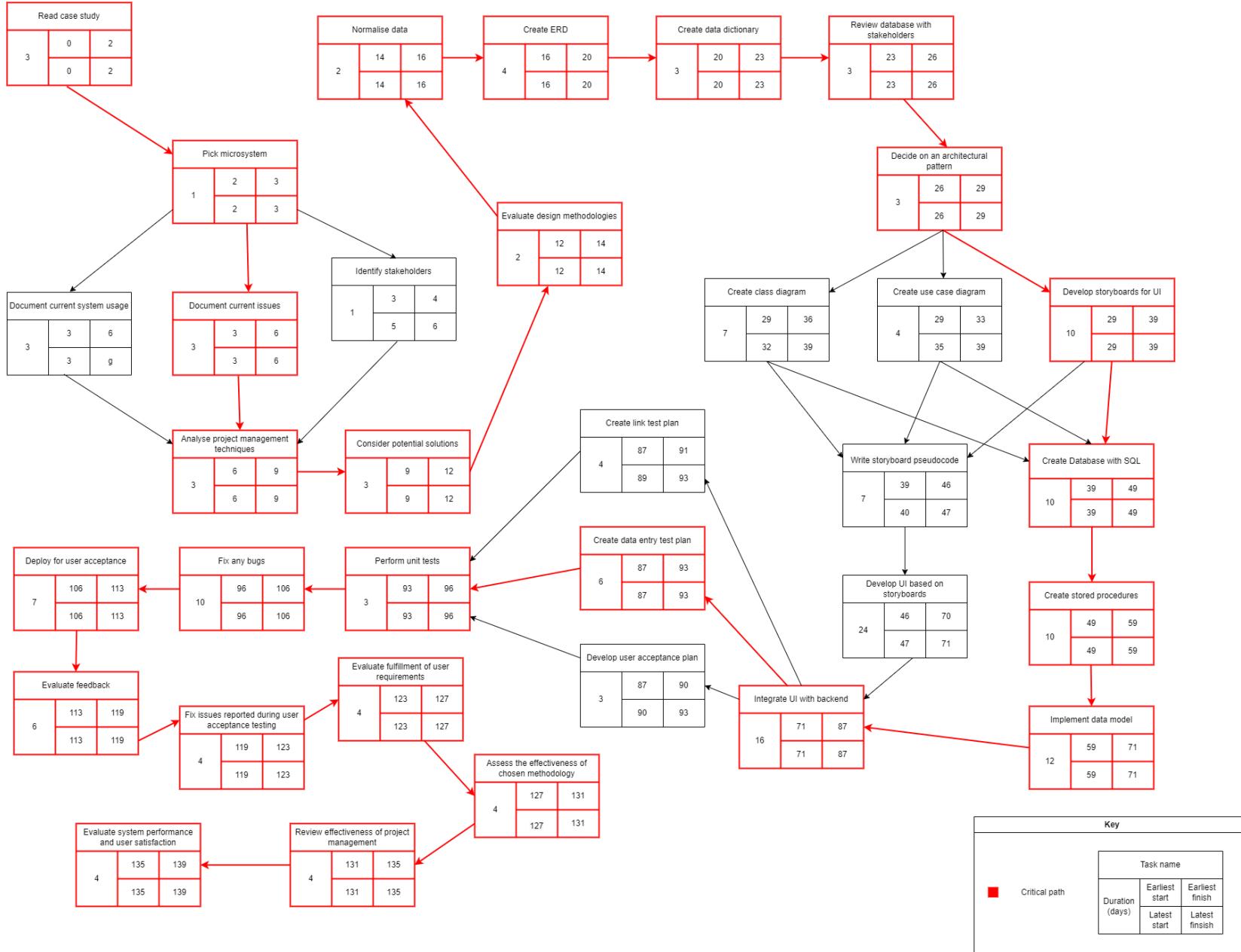
1. Critical Path Visibility: The critical path within a PERT chart is easy to identify. This helps project managers identify tasks that cannot be delayed without affecting the project's completion time.
2. Shows Dependencies: Arrows within a PERT chart make the dependencies between tasks easily visible.
3. Unexpected Delays: Project managers can simulate unplanned delays in task delivery and see their effect on the project timeline. This allows managers to identify where they may need more float.
4. Resource Efficiency: PERT charts easily show tasks on the critical path that may require more resources to introduce additional float. This makes them especially useful for helping project managers allocate resources efficiently.

Disadvantages

1. Requires Duration Estimates: The critical path calculation relies on realistic estimates of task durations, requiring a skilled project manager to create a reliable schedule.
2. Specialised Software: Creating and maintaining a PERT chart may require specialised software. This is especially evident in large software projects, where the critical path is too large to calculate by hand.
3. Technical Appearance: The appearance of the PERT chart might make it difficult for stakeholders to understand.

Conclusion

In conclusion, I will use a PERT chart to help manage resources for critical tasks and stay within the allocated time. The project is relatively small, so the critical path should be easily calculated by hand.



Design Methodologies

Design methodologies are systematic approaches to problem-solving. They provide a set of principles and guidelines for completing a project and help designers structure the delivery of a solution. Several design methodologies were considered to help control the project's development.

Agile vs Linear

Design methodologies can generally be divided into “agile”, and “non-agile” or “linear”. Non-agile methodologies follow a traditional sequential workflow with a fixed scope and user requirements created before the project is implemented. This planning means a large amount of documentation is produced, but changes to the project plan are expensive and discouraged. Agile methodologies adopt a flexible iterative and incremental approach. Agile emerged in response to the limitations of traditional strategies. The Agile Manifesto outlines the principles of prioritising individuals and interactions over processes and tools, working software over comprehensive documentation, customer collaboration over contract negotiation and responding to change over following a plan.

Scrum

Scrum is an agile design methodology which offers a flexible strategy with the development team working as a unit. Team self-organisation is encouraged by the close collaboration of all team members with daily face-to-face communications. This collaboration aims to improve how the team solves complex problems. Work is carried out in short “Sprints”, in which items from the “Product Backlog” are implemented. Scrum is outlined in detail within the [Scrum Guides](#).



Scrum Team

A Scrum Team is a small team of people (usually no more than 10) responsible for making progress on the project. Teams are self-managing and are responsible for all product-related activities such as research, development, maintenance and stakeholder collaboration. There are three specific roles within the Scrum Team:

Developers

Developers within a Scrum team are accountable for creating the Sprint Plan and Backlog and adapting their plans to progress towards the Sprint Goal. They also have to maintain quality. Since developers within a Scrum Team have no specific role (e.g., programmer or tester), they must have a wide range of skills.

Product Owner

They are accountable for maintaining the Product Backlog. This includes ordering Product Backlog items in order of priority. They must also communicate the product goal.

Scrum Master

They are accountable for establishing and leading the Scrum team. They are involved in leading, training and organising.

Parts of Scrum

Sprints

Sprints are consecutive work cycles of fixed length (timeboxed for one month or less) in which work is done within the Scrum. All necessary work for the Sprint, such as planning and review, happens during the sprint. During the Sprint, no changes are made that may interfere with or damage the Sprint Goal. The Product Backlog may be refined during the sprint, and the Product Owner may renegotiate the scope.

Sprint Planning

This part of Scrum initiates a Sprint by laying out the work to be performed during a sprint (in the Sprint Backlog). The plan is the collective work of the Scrum Team. All important Product Backlog items are discussed. Sometimes, other people may be invited to attend Sprint Planning to provide expertise and advice.

Daily Scrum

The Daily Scrum is a meeting that inspects the progress towards the Sprint Goal. The Sprint Backlog may be modified. The Daily Scrum aims to be a 15-minute event for the Developers and is usually held at the same place and time every day.

Sprint Review

Inspecting the outcome of a completed Sprint and looking for future adaptations and improvements is the Sprint Review's goal. The Scrum Team presents their current progress to the Stakeholders.

Sprint Retrospective

The Sprint Retrospective is a meeting to plan ways to increase quality and effectiveness. The team reviews the success of the previous Sprint. The tools used, processes, interactions and Definition of Done may all be inspected. It is usually timeboxed for a maximum of 3 hours for a one-month Sprint.

Product Backlog

This is an ordered list of what is needed to improve the product. It is the source of work for the Scrum Team. Product Backlog items are refined into small, precise items that need to be added. Sprint Backlogs are equivalent to the Product Backlog but only applicable to a specific sprint. The Product Backlog is constantly evolving to meet the stakeholders' needs.

Sprint Backlog

This is composed of the Sprint Goal, the reason for and single objective of the Sprint, and a subset of the Product Backlog selected to be implemented. Developers need to keep the Sprint goal in mind while they work.

Increment

This is how the unit of completed work is measured within the Scrum. When an item of the product backlog reaches the Definition of Done, a formal description of the quality of a deliverable required for a project, it is considered an Increment. The Definition of Done ensures that sprint items that do not meet its quality standards cannot be released prematurely or reviewed at a Sprint Review.

Advantages

1. **Adaptability:** Scrum is flexible due to its iterative and incremental structure, created through the Sprints. This means it is suited when clear requirements are difficult to identify.
2. **Creativity:** Scrum's emphasis on collaboration and teams means high levels of innovation and new ideas. The daily meetings and developer autonomy also help to encourage this.
3. **Fast Development:** The prioritisation of the Product Backlog and short duration of Sprints means that key parts of the application are implemented quickly, which can reduce time to market.
4. **Self-Management:** Scrum is a self-managing methodology, so it can require less documentation and control.
5. **High-Quality Work:** The Definition of Done provides a transparent standard for work. This means sub-standard items cannot be released and enforces a level of quality.
6. **Continuous Feedback:** Frequent meetings with stakeholders and constant product refinements mean stakeholders and clients are often happy with completed projects.

Disadvantages

1. **Experienced Team:** Scrum involves intense periods of work. All team members must be capable of successfully performing their tasks.
2. **Scalability Issues:** Scrum Teams are designed to include no more than 10 members. This means Scrum can be difficult to scale.
3. **Scope Creep:** Constant revisions of the Product Backlog can lead to scope creep. This could cause the overall project to run over time and budget.
4. **Non-Standard Working Environment:** A Scrum Team may be difficult to integrate with a company's organisational structure.
5. **Stakeholder Time Commitment:** Due to frequent meetings, Scrum is intensive on stakeholders' time.

Conclusion

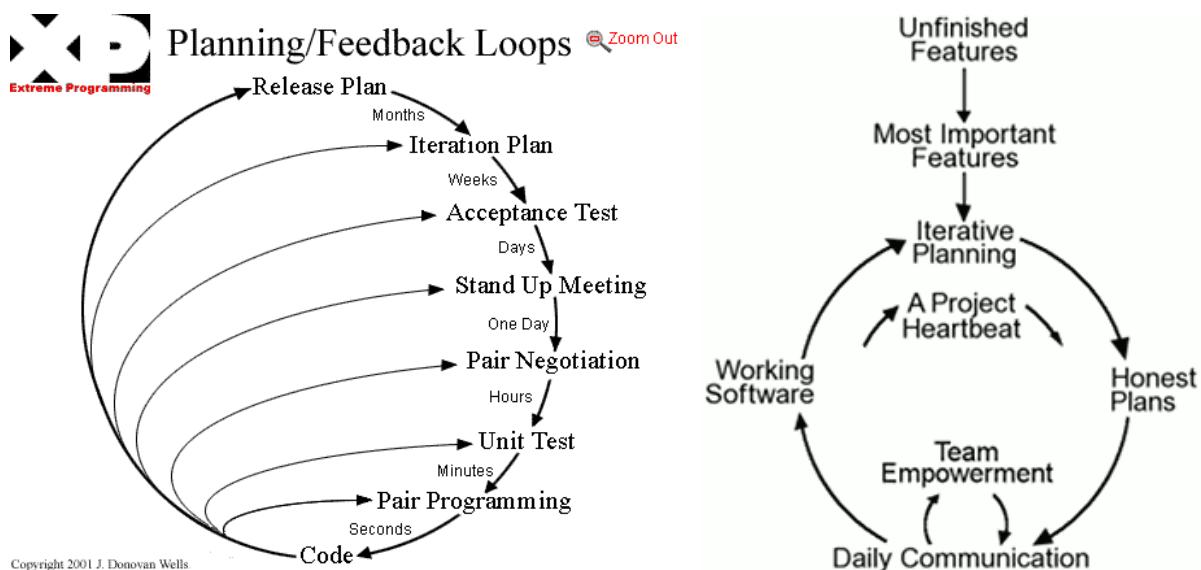
I will not be using the Scrum design methodology for the Movers Microsystem due to its requirement for a skilled team and constant stakeholder interactions.

Extreme Programming

Extreme programming (XP) is an agile design methodology. It intends to improve software quality by focusing on responsiveness to changing customer requirements rather than predictability. Teamwork is emphasised, and all managers, customers, and developers are equals. The team should self-organise around the problem to solve it efficiently. The five core values of extreme programming are communication, simplicity, feedback, courage and respect. XP is outlined in detail at the [Extreme Programming website](#).

Extreme Programming Lifecycle

The lifecycle of XP can be broken into several phases. Firstly, there is an exploration phase in which the development team tries to gather information about the initial requirements and the technical feasibility of the project. This can involve coding to research different solutions. Secondly, there is a planning phase. Work is prioritised and expectations of quality for deliverables are defined. What upcoming releases of the system should contain is decided, and these releases are then broken down into iterations (1-2 weeks). Thirdly, there is an iteration cycle. Working software is delivered through short focused cycles, planned at the start of the iteration in the planning phase. Once a release is ready, it is delivered to the customer. Bugs will be fixed and new features will be added in later iterations. Customer collaboration



through all phases is vital. There will be many iterations and releases within the project's lifetime. Finally, when a project is obsolete, it will be retired. Any necessary documentation will be finalised, and knowledge will be transferred to the stakeholders. The XP team then reflects on the lessons learned during the project.

Rules of Extreme Programming

Several key rules must be followed during the development of a project with XP.

Planning

User stories and simplified use case diagrams are written about the system by the client. They may include but are not limited to, descriptions of the user interface. They are in the format of short (three-sentence) text written by the customer without specific technical terminology. These are backed up by release planning. Release planning meetings are used to create a release plan which lays out what will be released during the project. Small, frequently tested releases with good business sense are prioritised. Features are implemented in iterations. Iteration planning is used at the start of each iteration to plan programming tasks.

Managing

XP teams should be given a dedicated open workspace to improve communication. Whiteboards could be added to help express ideas. Each day should be started with a stand-up meeting. These are short meetings that encourage team participation and attempt to limit the amount of time wasted in overly long meetings. In addition to this, there should be a sustainable pace set for each iteration. This ensures code is not rushed and developers are not overworked. Finally, developers should be regularly moved around to share fresh experiences throughout the team. This also makes the team more flexible, as everyone should know enough about each part of the system to be able to work on it.

Coding

During XP, customers should always be available to help and participate with the development team. It is generally recommended that the customer be on-site for immediate feedback and to refine the User Stories.

When developers write code during XP, there are several requirements to ensure quality. Firstly, code must be written to agreed standards to allow ease of refactoring and reading. This also encourages collective ownership of the code as a whole by the team. Secondly, unit tests should be written before the actual code. This encourages developers to create bug-free code and aims to improve the speed at which code is written. Thirdly, all production-level code must be pair-programmed. Pair programming is when two people work together at the same computer. This means that the code is of increased quality and checked as written. Fourthly, integration should be limited to only one pair at a time. This ensures that there is a clear-cut latest version of the software and that all code is tested together once it is integrated, minimising undetected bugs. This may be further reinforced by the use of a dedicated computer for integration. Finally, integration should happen often. This ensures everyone can work with the latest version and avoids fragmented development efforts.

Designing

Simplicity should be prioritised when designing the system. Simple code is easier and faster to maintain and develop, so this minimises wasted time. Complex code should always be simplified whenever possible. Furthermore, no functionality should be implemented until it is necessary to avoid redundant code. Spike Solutions is a simple program to explore a potential solution. These can reduce the risks of difficult-to-solve problems arising and delaying the project. Finally, code should be refactored whenever possible to save time in the future. This also ensures that all the code is maintainable and understandable.

Testing

To ensure code quality, all code must have unit tests. Automated unit test frameworks should be used to allow for automated unit test suites. All classes within the system should be tested, and all code must pass all unit tests before it can be released for production. If a bug is found, tests are written to guard against it coming back. Moreover, acceptance tests created from User Stories are run frequently to ensure a User Story has been implemented correctly. Customers are responsible for verifying the correctness of acceptance tests and reviewing test scores to prioritise failed tests.

Advantages

1. Speed of Development: During extreme programming, wasted time is minimised. Continuous integration and development also mean that XP projects are often finished quickly.
2. Customer Satisfaction: Frequent feedback loops and continuous customer collaboration (by having a customer on site) ensure the product aligns with the client's requirements. Frequent releases also allow customers to see progress.
3. Increases Quality: The test-driven development XP demands ensures that bugs are caught early. Constant refactoring and code standards being set from the beginning also maintain a clean codebase.
4. Risk Limitation: Short iterations and regular releases allow risks to be addressed early in the development cycle. Continuous integration also removes risks associated with a "Big Bang" integration at the end of a project's development lifecycle.
5. Improved Productivity: Techniques such as Pair Programming allow technical knowledge to be shared. XP proponents would also argue that pair programming increases productivity beyond that of two developers working alone. The sustainable work pace encouraged by XP also ensures that developers avoid burnout.
6. Flexible: Short development cycles, constant customer feedback and user acceptance testing allow user requirements to be updated or added as required.

Disadvantages

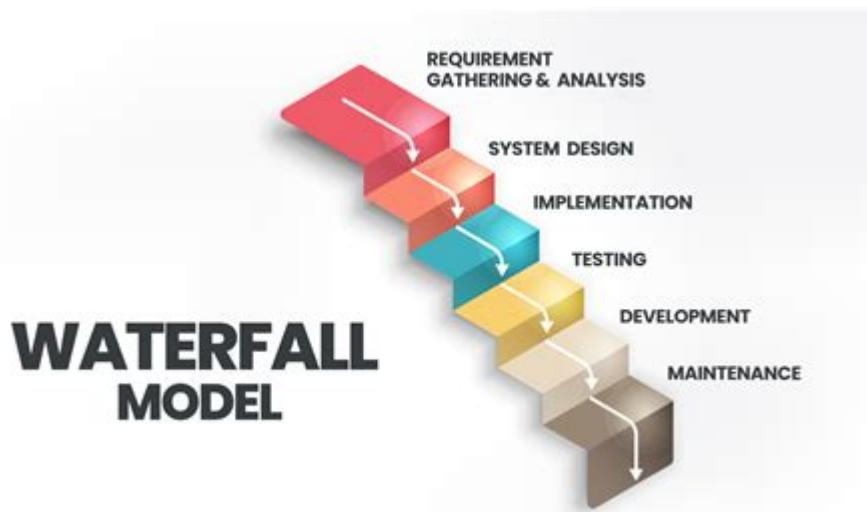
1. Limited Documentation: XP's focus on code over documentation can result in less comprehensive documentation. This could lead to scalability and maintainability issues.
2. Scalability: XP is designed to be used with tight-knit small teams. Close collaboration can be more difficult as team size increases.
3. Less Predictable: XP's iterative and incremental nature can make estimating a timeline and budget difficult. In addition, XP may not be suited to highly complex or heavily regulated projects.
4. Team Requirements: XP requires skilled developers and high levels of collaboration. If a team dynamic is poor, it can negatively affect the overall project.
5. Intensive of Customers' Time: XP demands a high level of commitment from customers. XP aims to have an expert on-site at all times.

Conclusion

In conclusion, I will not be using XP for this project. User requirements have already been provided and are fixed, so XP's flexibility is not necessary. In addition, I am the sole developer of the microsystem, so I will not benefit from techniques such as pair programming.

Waterfall

The waterfall model is a traditional, structured design methodology. Development is broken down into linear sequential phases. Each phase depends on the completion of the previous phase and can only start once the previous phase has been completed and validated.



Phases of Waterfall

Requirements

The first phase involves gathering requirements from stakeholders and analysing them to understand the scope and objectives of the project. All requirements are gathered from the user and then these are analysed to remove any inconsistent (contradictory) requirements or requirements that are incomplete. Requirements may be documented in a Software Requirement Specification (SRS). The SRS serves as part of the contract between the developers and the clients. The SRS is also useful for settling disputes. A Feasibility Study is also used to determine whether it is technically and financially viable to build the software. It involves attempting to understand the problem and determining various possible strategies to solve the problem.

Design

During this phase of the Waterfall methodology, the SRS is converted to a plan that can be implemented in a programming language. This includes high-level design, such as UI, and software architectural design through formats such as class diagrams and Entity Relationship Diagrams. The design phase is documented in a Software Design Document.

Implementation

During the coding phase, the Software Design Document is converted to functional code. Small unit tests may also be conducted during this time.

Testing

After the application's components are created, they undergo integration and system testing. Testing is initially undertaken by the development team (alpha testing), followed by beta testing with customers. Once the software has been delivered, it undergoes user acceptance testing. Changes to the software may be negotiated through addendums.

Maintenance

Once the software has been deployed, the development team will pivot to maintenance. There are three main types of maintenance. Firstly, there is Corrective Maintenance. Corrective Maintenance involves fixing errors that were not discovered within the development and testing process. Secondly, there is Perfective Maintenance. This involves adding new features to the system that the client requests after the application has been deployed. Finally, there is Adaptive Maintenance. Adaptive Maintenance involves porting software to new computers or operating systems. Once the contract expires and the project is ended or after deployment, the project's success is analysed, and the insight gained is recorded.

Advantages

1. Easy to understand: The Waterfall design methodology is easy to understand, so it is suited to less experienced teams.
2. Clear milestones: The linear nature of the Waterfall model makes it easy to see project milestones.
3. Documentation: A large volume of detailed documentation for all parts of the development cycle is produced.
4. Limited scope creep: The rigidity of the Waterfall methodology prevents the project scope from growing beyond what was defined at the outset.
5. Limited Risk: The extensive planning at the start of this methodology, through the Feasibility Study, ensures that time and budget can be managed effectively, minimising risk.

Disadvantages

1. Limited Feedback: The lack of user involvement during the execution of the project means that there may be a large number of feature requests after launch.
2. Inflexible: Inflexibility means that this design methodology is not suited to situations in which the client cannot provide well-defined user requirements or does not fully know their requirements.

3. Deadline Creep: Due to the linear nature of the Waterfall model, if an issue occurs within one phase of the project, all the other phases must be delayed until it is solved.
4. Late Defect Detection: Since testing and integration are done late in the development cycle, in a "Big Bang," errors may not be discovered until later, which can be expensive and cause delays.

Conclusion

In conclusion, I will be using the Waterfall Design Methodology due to the fixed requirements I have received from Movers. Its simple nature is also suited to my level of experience, whereas agile development would be more suited to developers with several years of experience. In addition, the lack of an end-user to facilitate iterative and incremental development means the project is not suitable for agile. Furthermore, the project has a limited time frame of 6 months to develop. Agile methodologies would require a longer timeframe for iteration. Finally, this project is an academic exercise designed to simulate systems design, so the waterfall methodology provides a strong base to showcase the design process.

User Requirements

I will aim to implement the following user requirements in the microsystem:

Functional Requirements

General	
ID	Description
FR1	There should be a light and dark mode for every part of the user interface.
FR2	The application should be menu-driven with a side menu.
FR3	The side menu should have drop-down tabs for clarity.
Sign In	
ID	Description
FR4	Staff members should be able to log in with a username and password.
FR5	Staff members should have unique usernames and strong passwords.
FR6	Staff members should be able to show their password on the login page.
FR7	Staff members should be able to toggle between light and dark modes on the login page.
FR8	The sign-in page should pause after the staff member submits a password to prevent rapid password and username entry.
FR9	When a user logs in with an incorrect password or username, there should be a descriptive error message.
Settings	
ID	Description
FR10	Staff members should be able to edit some of their settings by navigating with the side menu.
FR11	Staff members should be able to edit their forename and surname and provide an optional date of birth.
FR12	There should be validation of forename and surname to prevent the user from leaving them blank.
FR13	Staff members should be able to edit their contact details (email, phone

	number and, optionally, an address).
FR14	Staff members should only be able to provide a valid email and phone number. If it is incorrect, the user should be informed by an error message.
FR15	Staff members should be able to provide an optional emergency contact (forename, surname and phone number).
FR16	Staff members should be able to view their username and privilege level.
FR17	Staff members should be able to change their password.
FR18	Staff members should be able to change their preferences for displaying tooltips.
FR19	Staff members should be able to change their font.
FR20	Staff members should be able to change their appearance theme by choosing between light and dark modes.
Dashboard	
ID	Description
FR21	Staff members should have a welcome message shown when they sign in.
FR22	Staff members should be able to see when they should change their password.
Cleaning Job Management	
ID	Description
FR23	Staff members should be able to book a cleaning job for a time over 2 weeks in the future.
FR24	Bookings should be able to be edited until they are complete.
FR25	Past bookings should be viewable.
FR26	Bookings can be deleted if the customer cancels their cleaning service.
FR27	Bookings should be able to be rescheduled for a time over 2 weeks in the future.
FR28	Bookings should have a location.
FR29	Staff members should be able to select several cleaning options for each job. They should have an associated quantity.

FR30	Bookings should have an associated cleaning team that can be selected by the staff.
FR31	The application should allow bookings to have a start time and an end time to allow for scheduling of cleaning staff.
FR32	Cleaning team members should only be selectable if they are not already on the job at that time.
Customer Management	
ID	Description
FR33	Customers should have a name, surname, email and phone number.
FR34	Office staff should be able to add customers.
FR35	Office staff should be able to edit existing customers.
FR36	Customers should be archivable so that only current customers are visible and old customers can be hidden.
Cleaning Job Options Management	
ID	Description
FR37	Cleaning job options should have a name, description and unit cost.
FR38	Office staff should be able to add, edit and delete cleaning job options.
FR39	Cleaning job options should have their prices recorded historically for cleaning jobs.
FR40	Cleaning job options should be able to be archived when they are no longer in use.
FR41	Only active cleaning job options should be able to be used in a cleaning job.
Stock	
ID	Description
FR42	Stock items should have a name, description and quantity.
FR43	Stock items should display different warnings indicating whether they have high, low or medium stock.
FR44	Stock quantities should be recorded over time to allow stock usage to be

	seen.
FR45	Stock items should be able to be archived when they are no longer in use.
FR46	All stock quantity changes should have a staff member attached to track which staff member made each quantity change.
Stock Reordering	
ID	Description
FR47	Stock managers should be able to request stock to be bought at the office.
FR48	Orders should be able to be drafted and then sent to the office with a staff member attached to them
FR49	Office staff should have to approve orders and mark them as deliveries when they are requested
FR50	Deliveries should have a “pending” and “delivered” state.
FR51	When a delivery is received, the cleaning manager should record any discrepancies.
FR52	Cleaning managers should update the stock quantities once an order is received.
Administration	
ID	Description
FR53	There should be different types of users, i.e. cleaning staff should not be able to access the same parts of the application as managers. There should be administrative accounts, manager accounts, cleaning staff accounts, office staff accounts and cleaning manager accounts.
FR54	Administrative staff should be able to change other users' passwords.
FR55	Administrative staff should be able to add new staff.
FR56	Administrative staff should be able to edit all existing staff settings.
FR57	Administrative staff should be able to archive old user accounts and reactivate them if necessary.
FR58	Administrative staff should be able to view all sign-in attempts and see if they were successful.

Non-functional requirements

Navigation	
ID	Description
NF1	The application should be laid out in such a way as to make it easy to use and understand for new employees.
NF2	The amount of mouse movement and clicks should be minimised to improve productivity.
NF3	The application should be menu-driven to allow easy, self-explanatory navigation.
NF4	Users should be able to tab between controls.
NF5	A search functionality should be available for searching through different items, such as stock.
Accessibility	
ID	Description
NF6	All the controls should have accessibility options such as sensible tab orders.
NF7	The font should be easily legible, and there should be an option to change the font to something easier to read for those with Dyslexia.
NF8	There should be a consistent margin between controls to improve readability.
NF9	There should be accessibility options on the login menu
NF10	A light mode and dark mode should be implemented. Dark mode should be enabled by default to reduce eye strain.
Performance	
ID	Description
NF11	The application should be fast to load.
NF12	The application should have a low memory footprint.
NF13	The application should be able to handle multiple users interacting with it. It should also be able to switch between numerous users quickly.

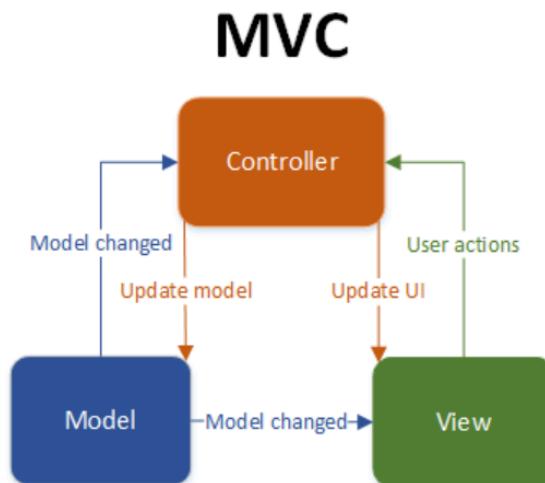
NF14	The application should run at a reasonable speed, even on a low-end computer.
NF15	The application should have a 99.99% uptime and be reliable.
NF16	Power efficiency should be good.
Portability and extensibility	
ID	Description
NF17	The application should be portable and run on Windows computers.
NF18	The application should be engineered to allow the rapid development and integration of other microsystems.
NF19	The application should be able to function as a standalone microsystem.
NF20	The application should be able to be used on all Windows computers.
NF21	The application should never stall or freeze. This includes long-running processes such as database queries.

Project Architecture

There are numerous different ways to develop a C# Windows Forms application. Project architectures are programming patterns which aim to improve maintainability while limiting class coupling.

Model View Controller

Model View Controller (MVC) was originally popularised for web development. It consists of several components: the Model, the View and the Controller. It aims to solve the problem of maintaining a large and complex application by splitting it into smaller sections.



Components of MVC

Model

This is part of the application (a group of classes) that corresponds to all the data-related logic that the user controls underneath the UI. The Model can refer to data objects passed through the View and Controller layer or business logic. It interacts with the database, performing CRUD (Create, Read, Update, Delete) operations and notifying the View and Controller of state changes. This would include the Data Access Layer and related classes.

View

The View is the part of the application that handles all the UI logic. Data creates views, but they are only responsible for displaying it. The View only interacts with the Controller.

Controller

The Controller enables communication between the View and the Model. It does not contain business logic and simply tells the viewer what to do. It is responsible for receiving user input and interpreting it, updating the model based on user actions and selecting and displaying Views.

Advantages

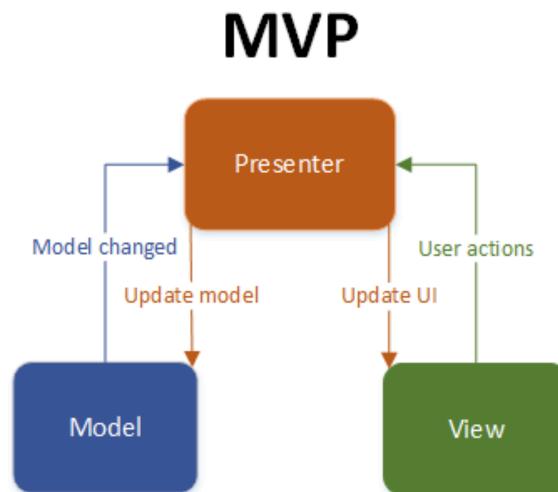
1. Scalable: Suited for small to medium-sized applications.
2. Simplicity: It is easy to understand and implement.
3. Separation of Concerns: Separates data and UI through the Model and View.
4. Reusability: Allows for models and views to be reused.

Disadvantages

1. Tight Coupling: The View and the Controller are often tightly coupled (i.e. the view class and controller class are extensively dependent on one another). Tight coupling makes it more difficult to modify views and their corresponding controllers without affecting each other.
2. Can become difficult to manage: Code can become very complex in larger applications.
3. Bloat: Controller classes can often have huge amounts of logic. This violates the Single Responsibility Principle, which states that classes or methods should have only one responsibility and, hence, only one reason for change. This can make development slower as controllers become more complex.

Model View Presenter

Model View Presenter (MVP) is a derivative of the MVC architecture. It attempts to improve the MVC framework by dictating how the View should be structured and encouraging loose coupling. The Model behaves the same way in MVP as it does in MVC. There are several different variants of MVP.



Variants of MVP

Supervising Controller

The Supervising Controller variant of MVP divides the logic of the application so that only complex logic is held within the Presenter. This means that the View is more independent and can handle some user interaction. While this approach can be suited to small, less complex projects due to lower code complexity, it is less testable due to the tighter coupling between the classes.

Observing Presenter

Within this variant of MVP, the View notifies the presenter of changes, which then updates the model, and vice versa. All logic is held within the presenter, but the View handles some UI display logic. This makes this variant general purpose with moderately good levels of testability with slightly increased complexity.

Passive View

Within the Passive View variant of MVP, the View is considered “dumb” and has no awareness of the model. All view updates must go through the presenter. This variant has the most code complexity but is ideal for string testing. It provides the maximum separation of concerns.

Advantages

1. Scalable: Suited for small to medium-sized applications.

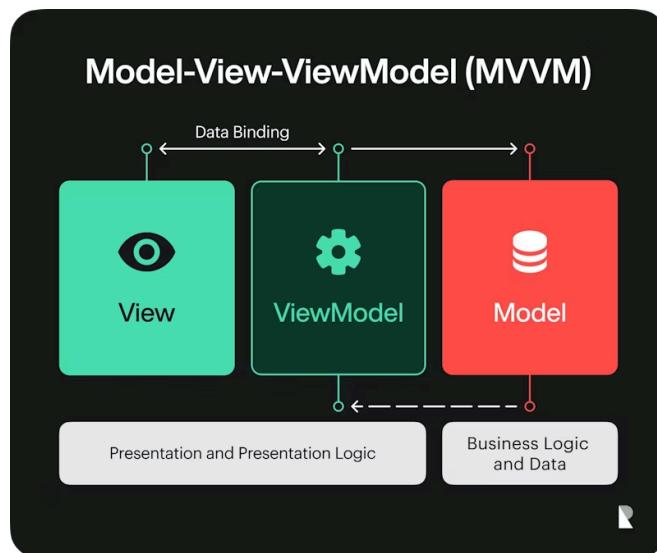
2. Simplicity: It is Easy to understand and implement.
3. Separation of Concerns: Separates data and UI through the Model and View.
4. Reusability: Allows for models and views to be reused.
5. Loose coupling: There is loose coupling between the views.
6. Testability: Variants of MVP generally have high levels of testability.

Disadvantages

1. Higher learning curve: MVP has a higher learning curve than MVC, and more initial setup is required.
2. Increased boilerplate: There is increased boilerplate within MVP as more code is required to ensure loose coupling.

Model View View-Model

The Model View View-Model (MVVM) architecture is a descendant of the MVC that aims to fully separate application logic from the user interface, making the application code more maintainable. Similar to MVC and MVP, it features three parts to help with this separation: the View, the View-Model and the Model. MVVM is most suited to WPF, React and Xamarin.



Components of MVVM

View

The View is what the user sees on the screen and simply contains the UI. It takes user input and provides feedback.

View-Model

The View-Model is responsible for providing data for the View from the Model and provides a layer of abstraction to separate them. It can handle interactions with the user and controls.

Model

The Model represents the application's data. It is a collection of objects, properties, and methods that interact with and contain database data.

Advantages

1. Separation of concerns: MVVM provides the best separation of concerns over MVC and MVP.
2. Maintenance: MVVM allows for better maintenance and testability.
3. Code Reusability: Code can be reused throughout the application.
4. Improved Collaboration: MVVM is the most well-defined so it allows increased collaboration within a team due to the predictable structure.

Disadvantages

1. Learning curve: MVVM has a steep learning curve, so it can be challenging for new developers to use it effectively.
2. Increased complexity: MVVM has increased complexity to completely separate the View and the Model through the View-Model abstraction.
3. Over-engineering: The complexity and increased boilerplate of MVVM can lead to over-engineering. Over-engineering can lead to increased “technical debt”, i.e., it becomes more difficult for a developer to add new functionality, slowing down the development of the project.

Conclusion

In conclusion, I will be using the MVP pattern with an Observing Presenter. This is due to the increased testability of this variant of MVP compared with other patterns, such as MVC. I have also chosen MVP over MVVM due to the less steep learning curve. I will be using a data access layer to separate the database queries within the Model.

Object Oriented Design

C# is an object-oriented programming (OOP) language. To fully exploit this, I will employ techniques of encapsulation, abstraction, inheritance, and polymorphism within several classes.

Classes

Classes are the key building blocks within OOP. Classes will contain related data for each model, form, and presenter. Examples of classes include the “Order” class, which will handle information related to stock ordering, and the “Staff” class, which will contain information related to a staff member’s account.

Encapsulation

Encapsulation hides internal implementations and controls access through public methods and properties. It allows for improved data integrity. To achieve these benefits, encapsulation will be used within every class through the use of public, protected, and private modifiers.

Abstraction

Abstraction is the generalisation of features to allow the hiding of complex details. Within the code, I will define clean interfaces to allow contracts between classes. Abstract classes (classes that are not instantiated but act as a base class from which child classes derive) will also be used to extract common behaviour. Within the project, several abstract classes will be used to handle CRUD operations. These classes will feature virtual and abstract methods to allow for overriding within children to define different implementations.

Inheritance

Inheritance is the creation of more specialised types and classes from base classes. Inheritance allows the reuse of common behaviours and implementations, allowing the creation of “dry” code. It also allows the establishment of hierarchies of related types necessary for polymorphism. There will be inheritance within custom control classes and also within the CRUD operation classes.

Polymorphism

Polymorphism is a fundamental concept of OOP, in which objects of different types are treated as objects of a common parent type. This is achieved through the use of late binding, where the method to be called is determined at runtime. Polymorphism is also achieved through method overriding by marking methods as virtual or abstract.

Data Access and Storage

It is vital to store and access data efficiently and securely within the application. This will be achieved by using a Microsoft SQL Server database, accessed with stored procedures through a Data Access Layer (DAL) within the application.

Microsoft SQL Server

Microsoft SQL Server (MS SQL) is a relational database management system developed by Microsoft. It is written in C and C++, so it offers high performance and works on multiple operating systems. It uses a structured query language called Transact-SQL that complies with the SQL standard. I will be using a single local database file within the project. This can later be moved to a dedicated server by configuring the connection string (the path to the database).

Stored Procedures

A stored procedure is an SQL statement saved within the database. They accept input parameters and can be used simultaneously by multiple requests at once. Using stored procedures increases performance by limiting the traffic directed to a database. This is due to shorter queries as the whole procedure is not sent to the database, only the parameters. This makes stored procedures ideal for optimising common queries. Stored procedures also improve security within the database by preventing SQL injection, reducing the risk of malicious attacks. Finally, stored procedures improve maintainability. If a query needs to be updated, only the stored procedure needs to be updated once within the database, rather than updating every instance of a query within the code base.

Data Access Layer

The Data Access Layer (DAL) is a class or series of classes (e.g. OrderDAL and StaffDAL classes) that handle database interactions. The DAL structure enforces separation of concerns by only allowing specific classes to interact with the database, improving security with encapsulation and maintainability. Business logic and database logic are completely separated.

Integration of Technologies

Windows Forms and the SQL Server database will work together within the MVP architecture to build a functional application.

Forms

Windows Forms will provide the front end for user interaction through the use of “Forms” (also known as “windows” or “views”). Forms will contain controls such as text boxes, check boxes and buttons to help the user interact with the application. Controls can also be grouped into “User Controls” to create more complex modular controls. They can be dragged and dropped onto the form to improve ease of development.

MVP Architecture

The Model View Presenter architecture will be implemented by using each form class as a View. It will then have a corresponding IView interface to separate its internal workings from the Presenter class, which will control the business logic and interact with the SQL database through the DAL. The Presenter and View will communicate through events, making the implementation a loosely coupled observing presenter.

Risk Mitigation

Several key strategies will be used to mitigate risk and overcome any challenges related to implementing the solution.

Testing

All code will be tested thoroughly during development and at the end of the process. This should minimise any bugs and prevent major bugs from appearing late in the development and testing phases.

Refactoring

During development, code will be refactored frequently to improve ease of development and to ensure a clean, “dry” codebase. This should also prevent any bloat of the code base or redundant classes or methods.

Comments and Documentation

Comments and method documentation strings will be used where necessary within the code base to explain clearly what the code is doing. This will help me later when testing and also during the development process.

Data Design

Normalisation

Normalisation is the process of organising a database to reduce redundancy (duplication) and improve data integrity. It also simplifies the database design, achieving an optimal structure of atomic elements (elements that cannot be broken down further). E.F. Codd, the inventor of relational databases, developed the normalisation process in the 1970s. The process is divided into several levels known as “normal forms.”

First Normal Form

In the first normal form (1NF), each column in a table is unique (i.e., groups are not repeated), and each record (row in the database) has an associated unique identifier called a primary key. Composite keys, which are primary keys composed of multiple fields, may also exist.

Second Normal Form

The second normal form (2NF) improves upon 1NF. In 2NF, all partial key dependencies are removed. Partial key dependencies occur when fields within a database entry do not depend on the full composite key. This means that for a table to be in 2NF, non-key columns should rely entirely on the candidate key.

Third Normal Form

Within the third normal form (3NF), any transitive dependencies are removed. Transitive dependencies are when a non-key field depends on another non-key field. In other words, all non-key columns within a table should depend on the primary key.

Boyce-Codd Normal Form (BCNF), Fourth Normal Form and Fifth Normal Form

The next levels of normal forms fix several other issues that can occur within a database design. Boyce-Codd Normal Form removes multiple overlapping candidate keys, the fourth normal form removes multivalued dependencies, and the fifth normal form, the most advanced normal form, removes “join” dependencies to minimise data redundancy and also helps fix update anomalies. In practice, higher normal forms are rarely needed for smaller databases.

Database Normalisation

Below are each of the steps to the database normalisation. To improve readability, fields used within the normalisation are named explicitly (e.g., “StockId” rather than “Id”).

Unnormalised Form (0NF)

Repeating groups of fields are indicated with curly braces. Round braces are used for clarity to show how some fields are related.

```
CleaningJobId, StartDate, EndDate, Address, ExtraInformation, (StaffId, Username,
HashedPassword, Salt, LastPasswordChange, Archived, Forename, Surname,
DateOfBirth, Email, PhoneNumber, Address, EmergencyContactForename,
EmergencyContactSurname, EmergencyContactPhoneNumber, PrivilegeLevel,
AppearanceSettings), {StaffId, Username, HashedPassword, Salt, LastPasswordChange,
Archived, Forename, Surname, DateOfBirth, Email, PhoneNumber, Address,
EmergencyContactForename, EmergencyContactSurname,
EmergencyContactPhoneNumber, PrivilegeLevel, AppearanceSettings},
{{CleaningJobOptionId, Name, Description, CurrentUnitCost, Archived}, Quantity,
UnitCostAtTime}, (CustomerId, Archived, Forename, Surname, Email, PhoneNumber,
Address)OrderId, StaffId, Description, Status, Discrepancies, {{StockId, Name,
Description, Archived, HighQuantity, LowQuantity, Sku, CurrentUnitCost, {Date, Quantity,
EditedByStaff, ReasonForQuantityChange}}, Quantity, UnitCostAtTime}, LoginAttemptId,
Username, AttemptTime, Successful
```

First Normal Form (1NF)

For 1NF and higher, tables will be denoted before the corresponding comma-separated fields with a colon. Primary keys are marked with an underline, and foreign keys are appended with an asterisk. For example...

TableA: fieldOne, fieldTwo, fieldThree

TableB: fieldOne^{*}, fieldFour

CleaningJob: CleaningJobId, StartDate, EndDate, Address, ExtraInformation, StaffId*,
(CustomerId, Archived, Forename, Surname, Email, PhoneNumber, Address)

Staff: StaffId, Username, HashedPassword, Salt, LastPasswordChange, Archived,
Forename, Surname, DateOfBirth, Email, PhoneNumber, Address,
EmergencyContactForename, EmergencyContactSurname,
EmergencyContactPhoneNumber, PrivilegeLevel, AppearanceSettings

CleaningJob_CleaningJobOption: CleaningJobId*, CleaningJobOptionId*, Quantity,
UnitCostAtTime

CleaningJobOption: CleaningJobOptionId, Name, Description, CurrentUnitCost, Archived

Order : OrderId, StaffId*, Description, Status, Discrepancies

Order_Stock: OrderId*, StockId*, Quantity, UnitCostAtTime

Stock: StockId, Name, Description, Archived, HighQuantity, LowQuantity, Sku, CurrentUnitCost

Stock_Quantity: StockId*, Date, Quantity, EditedByStaff, ReasonForQuantityChange

LoginAttempts: LoginAttemptId, Username, AttemptTime, Successful

Second Normal Form (2NF)

CleaningJob: CleaningJobId, StartDate, EndDate, Address, ExtraInformation, StaffId*, (CustomerId, Archived, Forename, Surname, Email, PhoneNumber, Address)

Staff: StaffId, Username, HashedPassword, Salt, LastPasswordChange, Archived, Forename, Surname, DateOfBirth, Email, PhoneNumber, Address, EmergencyContactForename, EmergencyContactSurname, EmergencyContactPhoneNumber, PrivilegeLevel, AppearanceSettings

CleaningJob_CleaningJobOption: CleaningJobId*, CleaningJobOptionId*, Quantity, UnitCostAtTime

CleaningJobOption: CleaningJobOptionId, Name, Description, CurrentUnitCost, Archived

Order : OrderId, StaffId*, Description, Status, Discrepancies

Order_Stock: OrderId*, StockId*, Quantity, UnitCostAtTime

Stock: StockId, Name, Description, Archived, HighQuantity, LowQuantity, Sku, CurrentUnitCost

Stock_Quantity: StockId*, Date, Quantity, EditedByStaff, ReasonForQuantityChange

LoginAttempts: LoginAttemptId, Username, AttemptTime, Successful

Third Normal Form (3NF)

CleaningJob: CleaningJobId, StartDate, EndDate, Address, ExtraInformation, StaffId*, CustomerId*

Customer: CustomerId, Archived, Forename, Surname, Email, PhoneNumber, Address

Staff: StaffId, Username, HashedPassword, Salt, LastPasswordChange, Archived, Forename, Surname, DateOfBirth, Email, PhoneNumber, Address, EmergencyContactForename, EmergencyContactSurname, EmergencyContactPhoneNumber, PrivilegeLevel, AppearanceSettings

CleaningJob_CleaningJobOption: CleaningJobId*, CleaningJobOptionId*, Quantity, UnitCostAtTime

CleaningJobOption: CleaningJobOptionId, Name, Description, CurrentUnitCost, Archived

Order : OrderId, StaffId*, Description, Status, Discrepancies

Order_Stock: OrderId*, StockId*, Quantity, UnitCostAtTime

Stock: StockId, Name, Description, Archived, HighQuantity, LowQuantity, Sku, CurrentUnitCost

Stock_Quantity: StockId*, Date, Quantity, EditedByStaff, ReasonForQuantityChange

LoginAttempts: LoginAttemptId, Username, AttemptTime, Successful

Entity Relationship Diagram

An entity relationship diagram (ERD) is a type of structural diagram, similar to a flowchart, that shows a database design. It contains the major entities (database tables and fields) and the relationship between each of them. ERDs use a defined set of symbols, including rectangles, ovals and diamonds, to help represent relationships with lines. They are a high-level, top-down approach to database design, commonly used in industry over the more traditional normalisation approach. ERDs also provide a solid foundation for further analysis and advanced design.

Advantages of ERDs

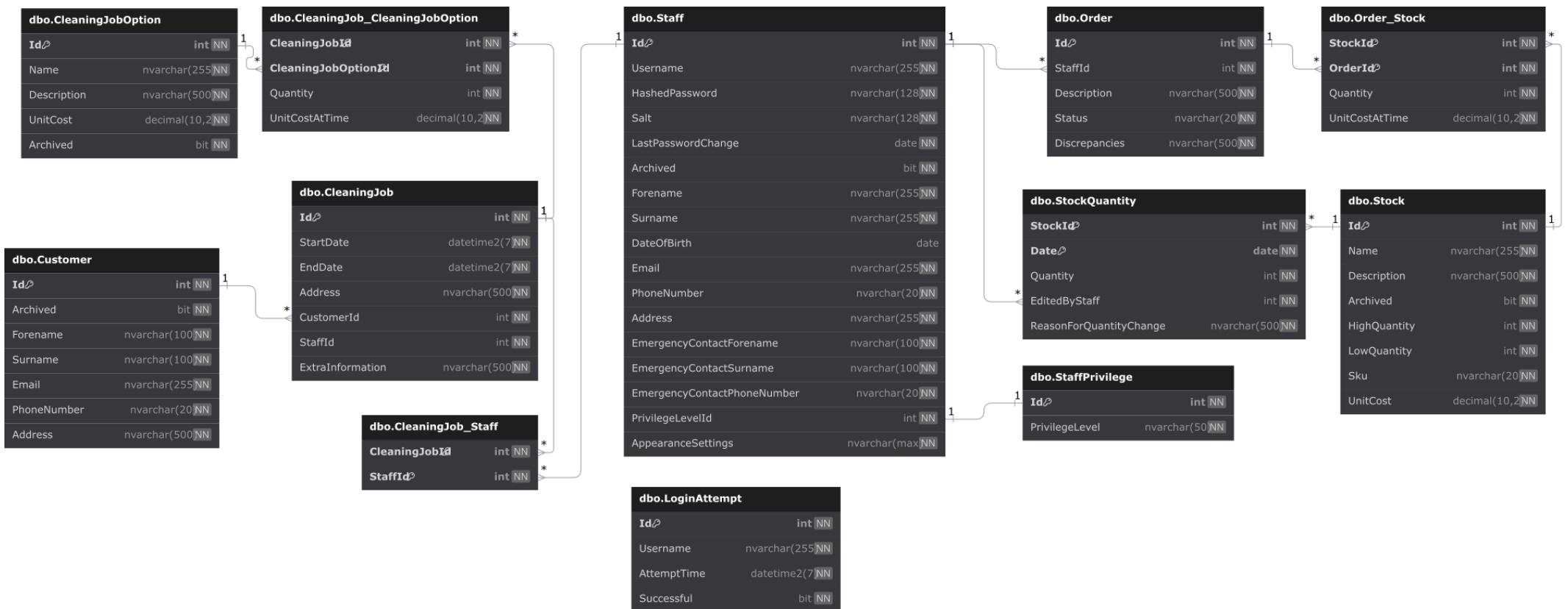
1. Visual: ERDs provide a visual approach to database design and can also be an effective communication tool when working with stakeholders.
2. Debugging: ERDs allow engineers to see problems within a database design quickly.
3. Allows for quick comparison of designs: ERDs' visual nature allows for quick comparison of different database designs. This can prove vital for large engineering projects when databases are re-engineered.

Disadvantages of ERDs

1. Loss of information: ERDs have a more limited amount of information than other approaches, such as data dictionaries.
2. No industry standard: There is no industry standard notation for ERD diagrams, which could lead to confusion.
3. Difficult to scale: As databases grow larger, ERDs become less useful as they are harder to understand due to the increased size of the diagram.
4. Risks of incorrect normalisation: Designing databases from the top-down approach of an ER diagram has an increased risk of incorrect normalisation.

Conclusion

In conclusion, I will use an entity relationship diagram to design the microsystem's database. Firstly, the cleaning microsystem is a relatively small project, so an ERD will add value and will not be overwhelming. Secondly, the use of a data dictionary will combat the loss of information within the ER diagram. Normalisation has already taken place, so there is a limited risk of incorrect normal forms.



Data Dictionary

Staff				
Field Name	Constraint	Data Type	Description	Example Data
Id	PRIMARY KEY	INT IDENTITY (1, 1) NOT NULL	Unique identifier of a staff member.	1
Username	UNIQUE	NVARCHAR (255) NOT NULL	The username that staff members use to log in.	'jdoe1'
HashedPassword		NVARCHAR (128) NOT NULL	The user's hashed password.	'1BEF2BCC749B BD637278CA7C6 99703C725EB230 28B5752F75565D 4BE8F8270D0B1 FF794C44BD9946 92989B4AA8B729 3101F0DDBEA30 884AB5099BCFB 54039B6B'
Salt		NVARCHAR (128) NOT NULL	A random salt for the hashed password.	'DF40270A641D2 60F92685782E02 C5F298C236995F E5849648908563 A9F6061F29DBF1 7DAAFAE631318 C0BA1CABB7AD 286F9A31440699 AD0C5DE98E331 6A63E55'
LastPasswordChange		DATE DEFAULT (getDate()) NOT NULL	The date of the user's last password change.	03/06/2024
Archived		BIT DEFAULT (0) NOT NULL	Determines if a user account is active. (1 means archived, 0 means not archived)	0
Forename		NVARCHAR (255) NOT NULL	Staff member's forename.	'John'

Surname		NVARCHAR (255) NOT NULL	Staff member's surname.	'Doe'
DateOfBirth		DATE NULL	The staff member's date of birth.	14-09-2006
Email		NVARCHAR (100) NOT NULL	The staff member's email.	'jdoe@example.co m'
PhoneNumber		NVARCHAR (20) NOT NULL	The staff member's phone number.	'+441234567890'
Address		NVARCHAR (500) DEFAULT ("") NOT NULL	The staff member's address.	'3 Oak Tree Lane, Lisburn'
EmergencyContactForename		NVARCHAR (255) DEFAULT ("") NOT NULL	The staff member's emergency contact forename.	'Jane'
EmergencyContactSurname		NVARCHAR (255) DEFAULT ("") NOT NULL	The staff member's emergency contact surname.	'Doe'
EmergencyContactPhoneNumber		NVARCHAR (20) DEFAULT ("") NOT NULL	The staff member's emergency contact phone number.	'+449876543210'
PrivilegeLevelId	FOREIGN KEY	INT NOT NULL	The privilege level of the account.	3
AppearanceSettings		NVARCHAR (MAX) DEFAULT (N'{"AppearanceThe me": "dark", "ShowToolTips": true, "FontName" : "Bahnschrift"}) NOT NULL	The appearance settings of the staff member's account in a JSON string.	'{"AppearanceThe me": "dark", "ShowToolTips": true, "FontName" : "Bahnschrift"}'
StaffPrivilege				
Field Name	Constraint	Data Type	Description	Example Data
Id	PRIMARY KEY	INT NOT NULL IDENTITY (1, 1)	A unique identifier for the privilege level.	1
PrivilegeLevel	UNIQUE	NVARCHAR (50) NOT NULL	The privilege level name.	'Manager'
LoginAttempt				

Field Name	Constraint	Data Type	Description	Example Data
Id	PRIMARY KEY	INT IDENTITY (1,1) NOT NULL	Unique identifier for each login attempt.	1
Username	FOREIGN KEY UNIQUE	NVARCHAR (500) NOT NULL	The username of the staff member attempting to log in.	'jdoe1'
AttemptTime		DATETIME2 (7) NOT NULL	The timestamp when the login attempt was made.	2024-03-06 12:30:45.1234567
Successful		BIT NOT NULL	Indicates whether the login attempt was successful (1 for success, 0 for failure).	1
Stock				
Field Name	Constraint	Data Type	Description	Example Data
Id	PRIMARY KEY	INT IDENTITY (1,1) NOT NULL	Unique identifier for each stock item.	1
Name		NVARCHAR(255) NOT NULL	The name of the stock item.	'Dettol Spray'
Description		NVARCHAR(500) NOT NULL	A description of the stock item.	'Kills 99.9% of bacteria'
Archived		BIT DEFAULT (0) NOT NULL	Indicates if the stock item is archived (1 for archived, 0 for active).	0
HighQuantity		INT DEFAULT (0) NOT NULL	The high quantity threshold for the stock item.	100
LowQuantity		INT DEFAULT (0) NOT NULL	The low quantity threshold for the stock item.	10
Sku	UNIQUE	NVARCHAR(20) NOT NULL	The SKU (Stock Keeping Unit) identifier.	'DET-001'
UnitCost		DECIMAL(10,2) DEFAULT (0) NOT	The unit cost of the stock item (in	15.99

		NULL	pounds).	
StockQuantity				
Field Name	Constraint	Data Type	Description	Example Data
StockId	FOREIGN KEY	INT NOT NULL	The unique identifier of the stock item.	1
Date	PRIMARY KEY	DATE NOT NULL	The date when the stock quantity was recorded.	2024-03-06
Quantity		INT NOT NULL	The quantity of stock recorded on that date.	50
EditedByStaff		INT NOT NULL	The staff member who edited the stock quantity.	2
ReasonForQuantityChange		NVARCHAR(500) NOT NULL	The reason for the quantity change.	'Stock received'
Order				
Field Name	Constraint	Data Type	Description	Example Data
Id	PRIMARY KEY	INT IDENTITY (1,1) NOT NULL	Unique identifier for each order.	1
StaffId		INT NOT NULL	The staff member who created the order.	2
Description		NVARCHAR(500) NOT NULL	A description of the order.	'Order for supplies'
Status	CHECK ([Status]='Delivered' OR [Status]='Pending' OR [Status]='Rejected' OR [Status]='Submitted' OR [Status] = 'Draft')	NVARCHAR(20) NOT NULL DEFAULT ('Draft')	The current status of the order. It can only be one of 'Draft', 'Rejected', 'Pending', 'Delivered' or 'Submitted'.	'Pending'

)			
Discrepancies		NVARCHAR(500) NOT NULL	Any discrepancies related to the order.	'Delayed delivery'
Order_Stock				
Field Name	Constraint	Data Type	Description	Example Data
StockId	FOREIGN KEY	INT NOT NULL	The unique identifier of the stock item in the order.	1
OrderId	FOREIGN KEY	INT NOT NULL	The unique identifier of the order.	1
Quantity		INT NOT NULL	The quantity of the stock item in the order.	10
UnitCostAtTime		DECIMAL(10,2) NOT NULL	The unit cost of the stock item at the time of the order.	12.99
Customer				
Field Name	Constraint	Data Type	Description	Example Data
Id	PRIMARY KEY	INT IDENTITY (1,1) NOT NULL	Unique identifier for each customer.	1
Archived		BIT DEFAULT (0) NOT NULL	Indicates if the customer is archived (1 for archived, 0 for active).	0
Forename		NVARCHAR(100) NOT NULL	The customer's first name.	'Bob'
Surname		NVARCHAR(100) NOT NULL	The customer's last name.	'Jones'
Email	UNIQUE	NVARCHAR(255) NOT NULL	The customer's email address.	'bobjones@example.com'
PhoneNumber		NVARCHAR(20) NOT NULL	The customer's phone number.	'+441234567890'

Address		NVARCHAR(500) DEFAULT ("") NOT NULL	The customer's address.	'123 Pine St, Lisburn'
CleaningJob				
Field Name	Constraint	Data Type	Description	Example Data
Id	PRIMARY KEY	INT IDENTITY (1,1) NOT NULL	Unique identifier for each cleaning job.	1
StartDate		DATETIME2 (7) NOT NULL	The start date and time of the cleaning job.	2024-03-06 08:00:00.000
EndDate		DATETIME2(7) NOT NULL	The end date and time of the cleaning job.	2024-03-06 10:00:00.000
Address		NVARCHAR(500) NOT NULL	The address where the cleaning job takes place.	'123 Pine St, Lisburn'
CustomerId	FOREIGN KEY	INT NOT NULL	The unique identifier for the customer requesting the cleaning job.	1
StaffId	FOREIGN KEY	INT NOT NULL	The unique identifier for the staff member assigned to the job.	2
ExtraInformation		NVARCHAR(500) DEFAULT ("") NOT NULL	Additional information about the cleaning job.	'Customer has a dog. Close the gate.'
CleaningJob_Staff				
Field Name	Constraint	Data Type	Description	Example Data
CleaningJobId	FOREIGN KEY	INT NOT NULL	The unique identifier for the cleaning job.	1
StaffId	FOREIGN KEY	INT NOT NULL	The unique identifier for the staff assigned to the cleaning job.	2
CleaningJob_CleaningJobOption				

Field Name	Constraint	Data Type	Description	Example Data
CleaningJobId	FOREIGN KEY	INT NOT NULL	The unique identifier for the cleaning job.	1
CleaningJobOptionId	FOREIGN KEY	INT NOT NULL	The unique identifier for the cleaning job option.	1
Quantity		INT DEFAULT (1) NOT NULL	The quantity of the cleaning job option.	2
UnitCostAtTime		DECIMAL(10,2) NOT NULL	The unit cost of the cleaning job option at the time of the job.	10.50
CleaningJobOption				
Field Name	Constraint	Data Type	Description	Example Data
Id	PRIMARY KEY	INT IDENTITY (1,1) NOT NULL	Unique identifier for each cleaning job option.	1
Name		NVARCHAR(255) NOT NULL	The name of the cleaning job option.	'Deep Cleaning'
Description		NVARCHAR(500) DEFAULT ("") NOT NULL	A description of the cleaning job option.	'Extra deep cleaning of carpets'
UnitCost		DECIMAL(10,2) NOT NULL	The unit cost of the cleaning job option.	50.00
Archived		BIT DEFAULT (0) NOT NULL	Indicates if the cleaning job option is archived.	0

Uniform Modelling Language

Unified Modeling Language (UML) is a multi-purpose modeling language that aims to standardise the visualisation of software system design. It began development in 1994 by Gary Booch, Ivar Jacobson and James Rumbaugh to standardise the different notational approaches to systems design for software engineering at the time.



UML provides a set of notations to create models of a software system. It has the foundation of its design in object-oriented programming (OOP), which makes it suitable for projects using object-oriented programming languages, such as C#. The standardisation from UML also enables better communication with stakeholders and within the development team, reducing confusion and errors.

Types of Unified Modeling Language Diagrams

UML provides notation for three main groups of diagrams to help visualise blueprints for system design.

Behavioural Diagrams

Behavioural diagrams describe how a system behaves during execution. They include communication diagrams, state machine diagrams and activity diagrams. These all focus on control and data flow through the system.

Interaction Diagrams

These diagrams emphasise how objects interact with each other through the exchange of messages. Interaction diagrams include sequence diagrams, communication diagrams, and timing diagrams, all of which are useful for modelling the interaction between components of the system.

Structural Diagrams

Structural diagrams show how a system is structured and the relationships between different components. These include class diagrams, component diagrams and package diagrams.

Advantages

1. Improved visualisation: UML helps to provide a clear representation of all system components and their relationships.
2. Communication: The standardised format enables effective communication between developers and stakeholders.
3. Analysis and design: UML allows for the design of a wide array of system components.
4. Documentation: The usage of UML diagrams in documentation means there are clear system schematics for future maintenance.

Disadvantages

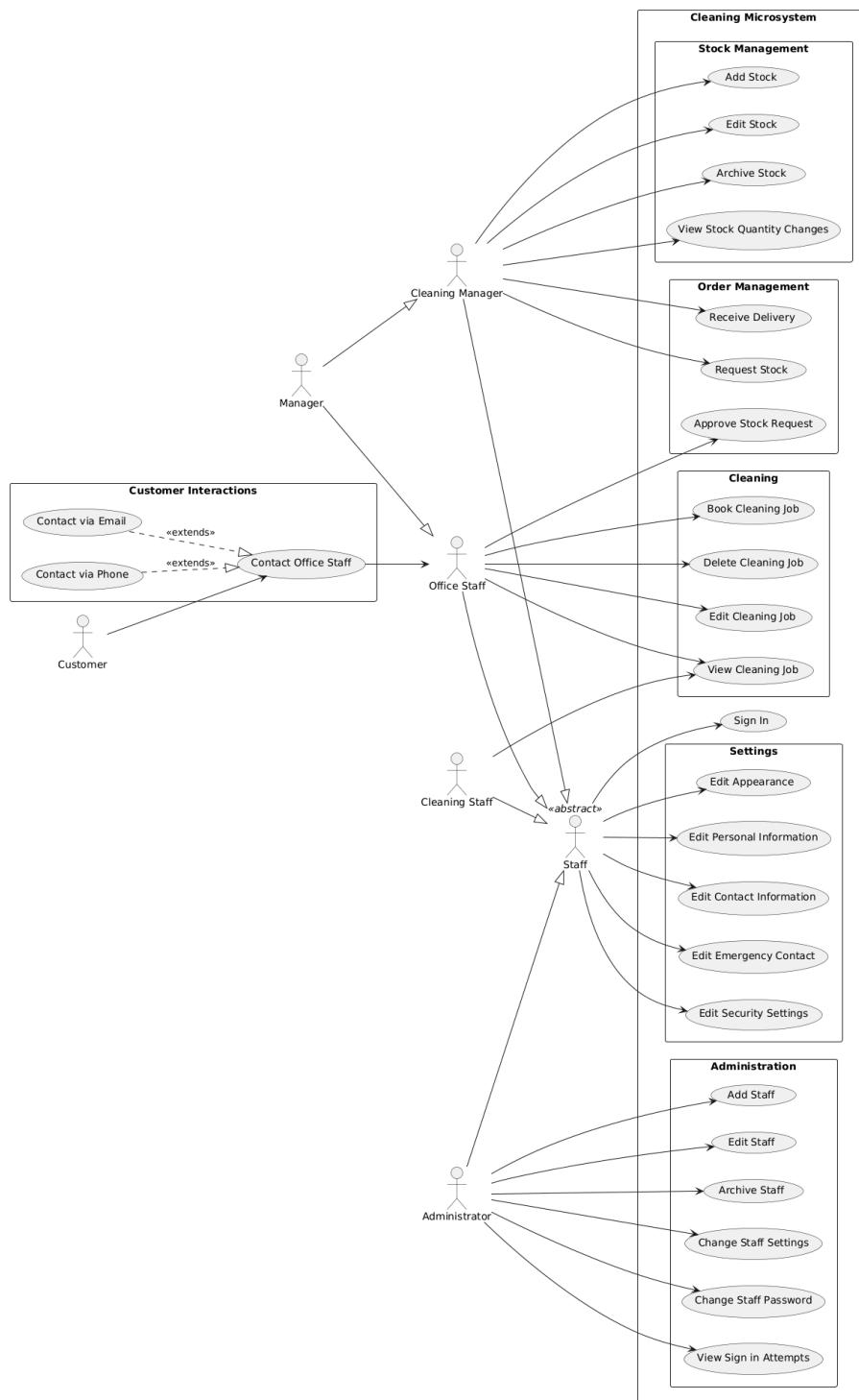
1. Complexity: The wide variety of UML notation means there is a steep learning curve due to the extensive features.
2. Time: Creating and maintaining accurate diagrams can be time-consuming and may require extensive updates.
3. Ambiguities: UML diagrams can cause confusion for team members unfamiliar with the notation and lead to different interpretations of the diagram.

Usage of Unified Modeling Language

Although UML is widely used to create diagrams and has been adopted by various organisations, including the International Organization for Standardization (ISO), it is not always used explicitly within software engineering. I will be using UML for my microsystem architecture planning due to its numerous benefits, such as improved communication with stakeholders.

Use Case Diagram

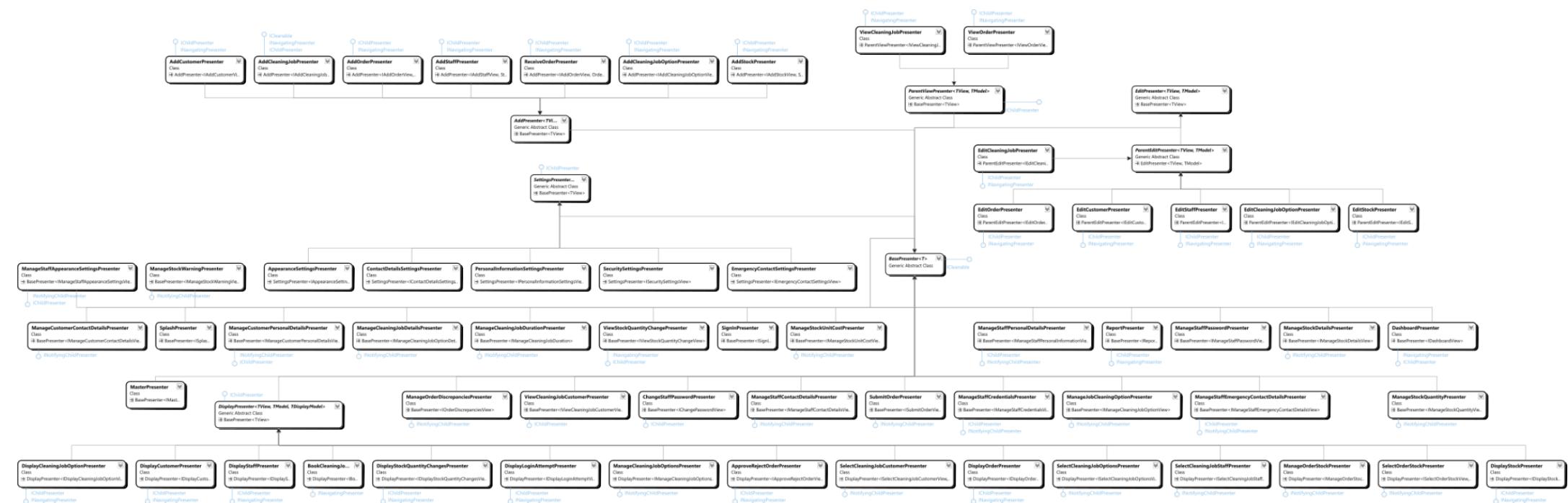
Use case diagrams are a type of UML diagram that focuses on how a system behaves from a user perspective. They represent the ways in which “actors” (users or external systems) interact with the system with arrows and lines called associations. Actors are represented by stick figures. Use cases, the different operations or functionalities of the system, are represented by ovals. The use case diagram for the cleaning microsystem can be seen below:



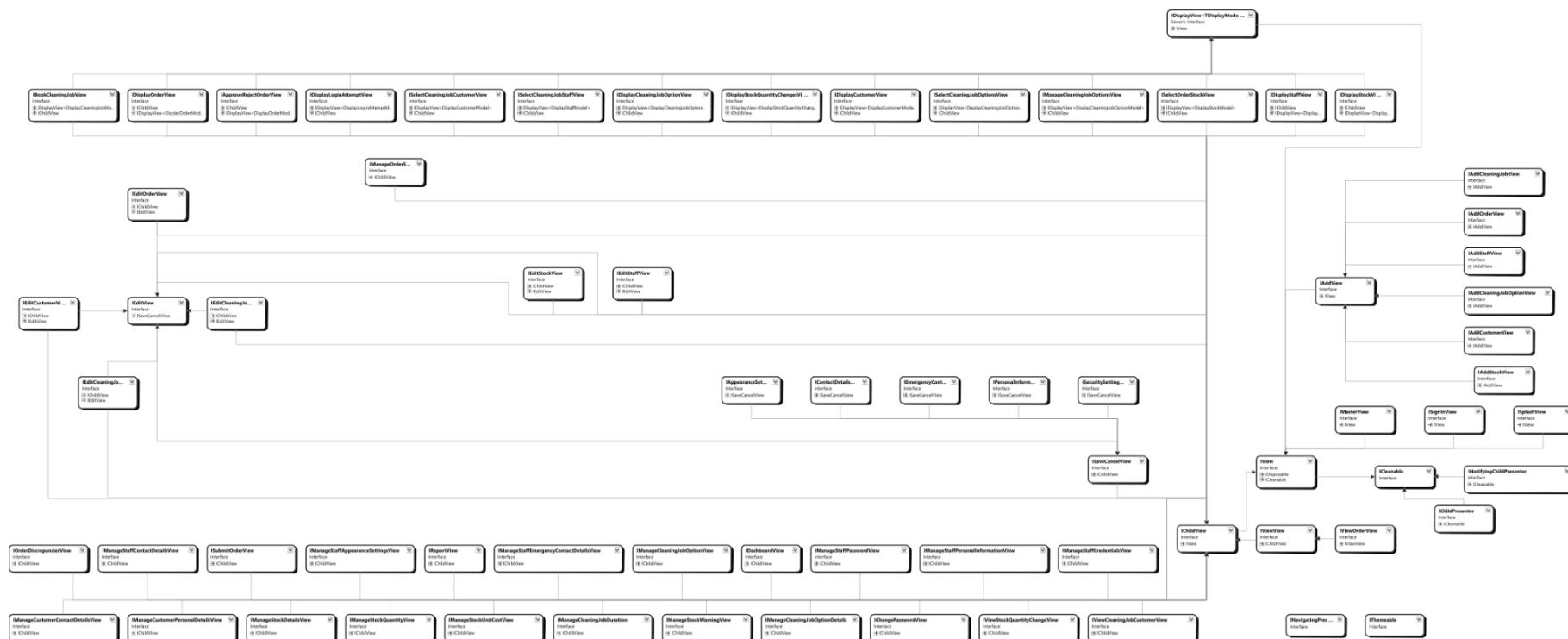
Class Diagram

A class diagram is a type of static structure diagram that represents the various classes in a system's associations and relationships. Each class includes attributes and methods, each defined with specific access modifiers. The diagram provides an accurate representation of the program's class structure, ensuring consistency between design and implementation. The class diagram for the cleaning microsystem can be seen below (a higher-resolution version has also been included as a separate file). Arrows represent the association of base classes/interfaces and their corresponding child classes. Classes that implement interfaces are represented by blue circles. Structs are represented by yellow boxes without rounded corners, and enums are represented by blue boxes.

Presenter Classes



Interfaces



Forms



Design

Storyboards

Storyboards are a visual representation of how the user interface will appear within an application. They are useful to help designers, developers, and stakeholders understand the user experience. Once completed, they will serve as a visual guide for building the user interface (UI) within Windows Forms. Storyboards will aim to illustrate all UI elements (e.g. buttons, images, textboxes, etc.). Each storyboard will include both a dark and light mode in the final design.

General Styling

Colour Scheme

The application will use both a light and dark mode to improve the user experience. The colours used are specified below. Throughout storyboards, colours will be referred to by their names from the table to improve readability (after the “Color” property). Not all colours may feature frequently within the storyboards. The bottom five colours are generally reserved for programmatic colouring of data grid view cells.

Colour Name	Light Mode	Dark Mode
Background	(250, 250, 250)	(9, 9, 10)
Foreground	(9, 9, 10)	(250, 250, 250)
Primary	(200, 200, 204)	(39, 39, 42)
Primary Foreground	(133, 133, 134)	(168, 171, 174)
Secondary	(226, 226, 226)	(24, 24, 27)
Secondary Foreground	(24, 24, 27)	(226, 226, 226)
Danger	(225, 29, 72)	(190, 18, 60)
Warning	(251, 191, 36)	(245, 158, 11)
Info	(34, 197, 94)	(22, 163, 74)
Data	(56, 189, 248)	(14, 165, 233)
Other	(168, 85, 247)	(192, 132, 252)

Fonts

Throughout the application, one of three main fonts will be used. The fonts are as follows:

Font Name	Sample text
Bahnschrift (sans serif)	<i>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</i>
Century (serif)	<i>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</i>
Comic sans (sans serif, dyslexia friendly)	<i>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.</i>

For clarity, there will not be a storyboard for each font. All controls have the same font within a storyboard.

Other Styling

Several other key styling motifs will be used throughout the design. These include:

Property	Description
CornerRadii	On all panels, buttons and textboxes, the corner radius for all corners will be set to 10.
DataGridView	On all datagrid view column headers, the font will be set to bold. In addition, the selected forecolour will be the same as the normal forecolour. The selected row color will be PrimaryForeground.

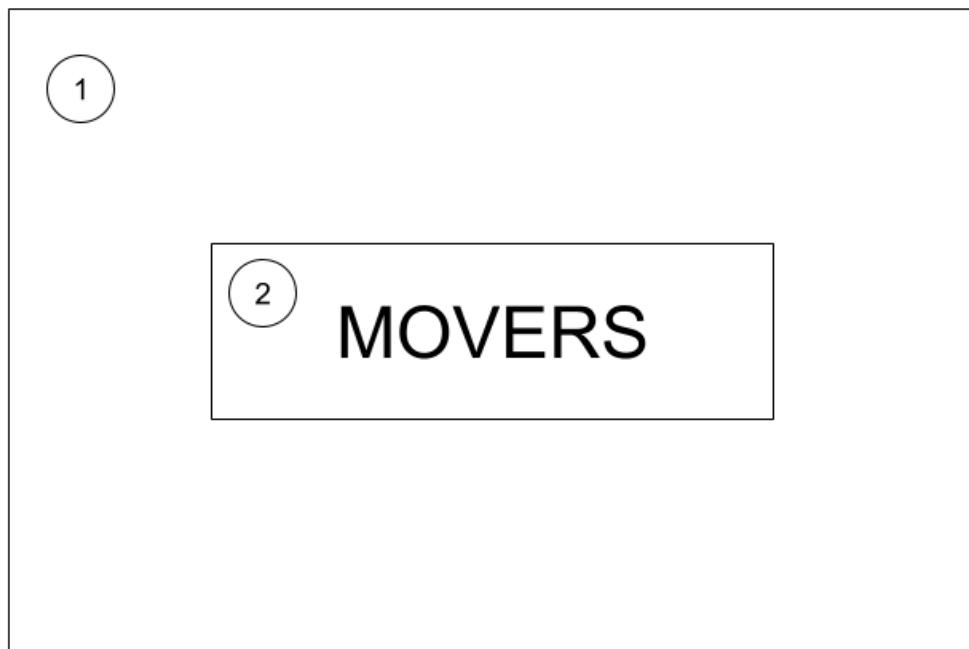
Form Storyboards

Below is a selection of form storyboards. These provide an overview of how the final UI will appear. Some storyboards have been grouped together for conciseness.

Splash Screen

The splash screen is the first user interface element the user sees when they start the application. It displays briefly when the application is loading and then disappears as the sign-in view appears.

Initial Design



Description

ID	Item	Walk Through	Styling
1	SplashView	The main form for the splash screen.	BackColor: Background Size: 500, 300
2	pbLogo	The movers logo.	BackColor: Background Size: 300, 100 Location: 100, 100 Anchor: None

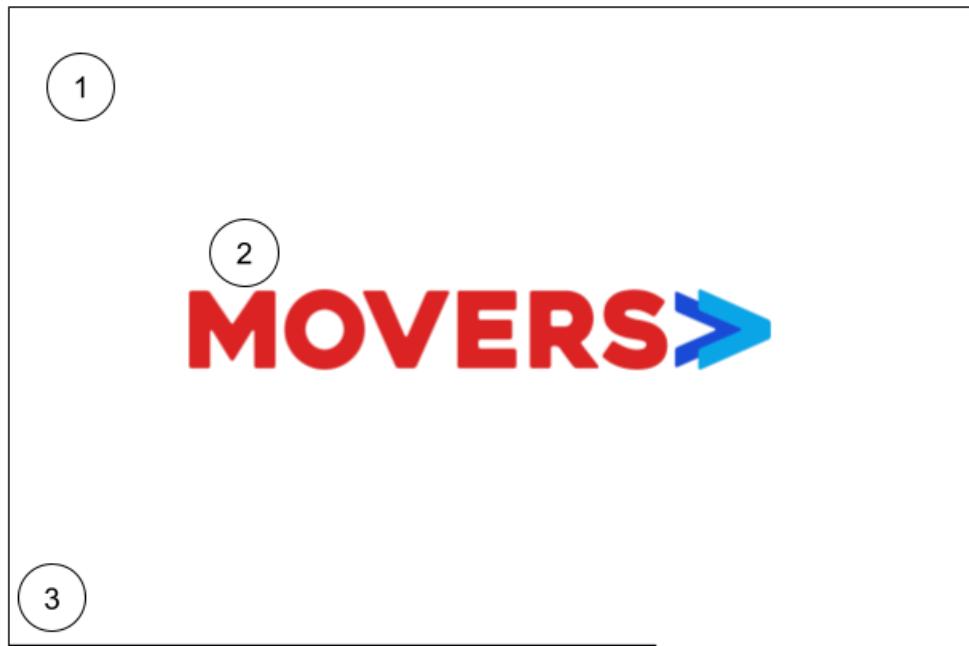
Feedback

Include a way for users to be able to see how close the application is to being loaded, to give staff members a sense of how long until the application will be ready.

Response

The storyboard will be updated to include a progress bar at the bottom. It will move from left to right across the screen while the application is loading. This should take approximately one second.

Final Design



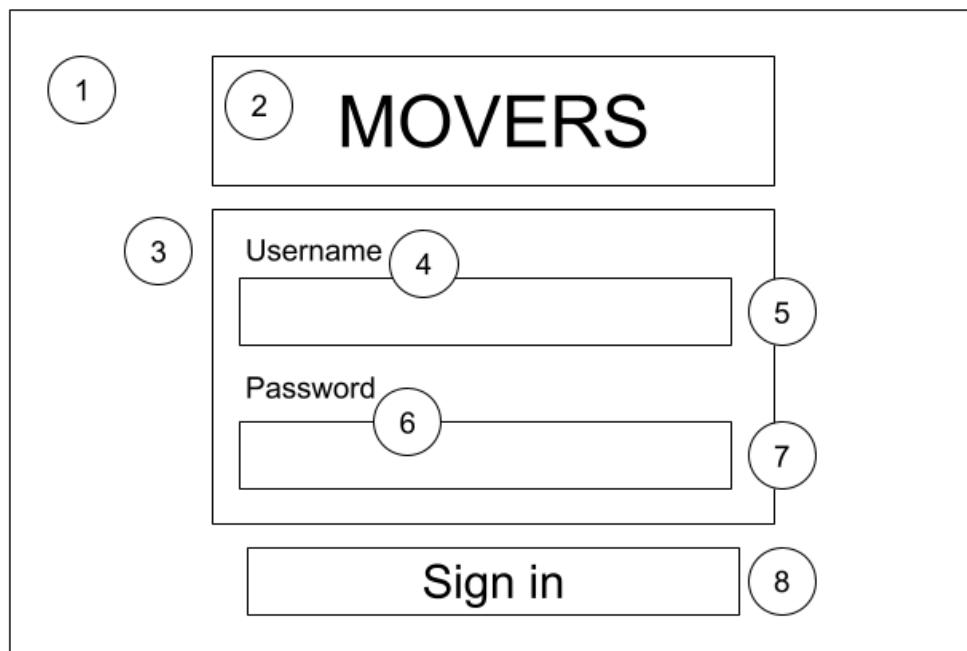
Description

ID	Item	Walk Through	Styling
1	SplashView	The main form for the splash screen.	BackColor: Background Size: 500, 300
2	pbLogo	The Movers logo.	BackColor: Background Size: 300, 100 Location: 100, 100 Anchor: None
3	pnlProgress	A progress bar that fills up as the application loads.	BackColor: Foreground Size: 500, 3 (variable width) Location: 0, 297 Anchor: Bottom, Left

Sign In Page

The sign-in page of the application is where the staff member will type in their credentials to access their account. If they correctly enter their username and password, they will be redirected to the main page where they can use the application.

Initial Design



Description

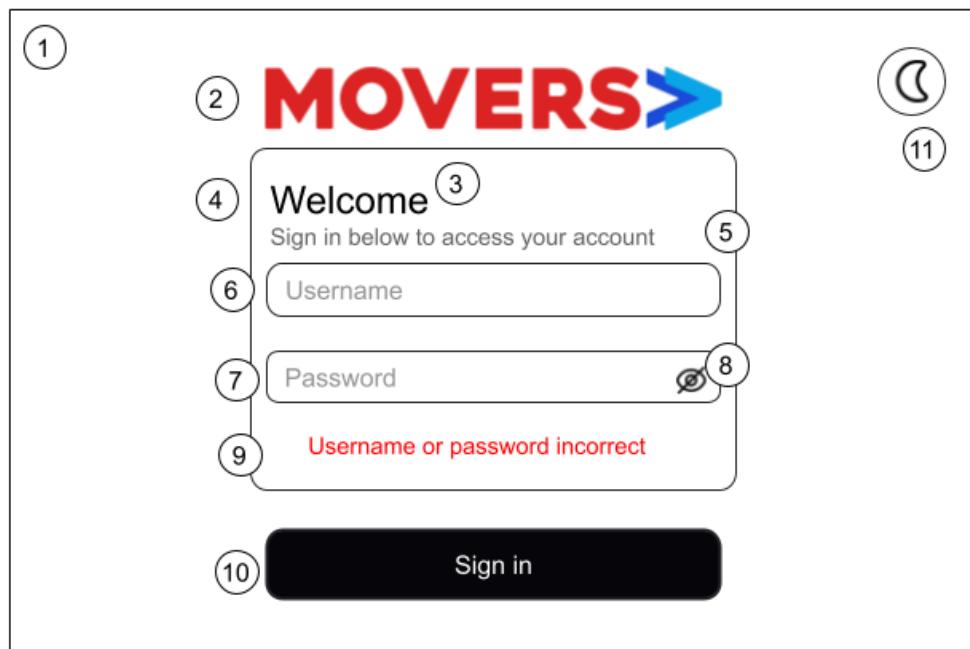
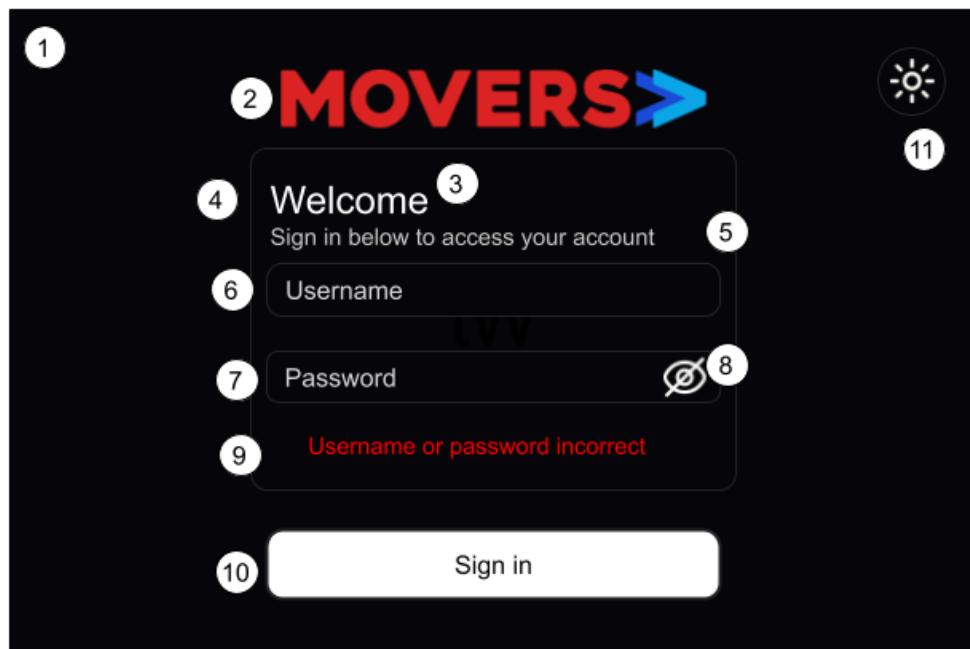
ID	Item	Walk Through	Styling
1	SignInView	The main form for the sign in screen.	BackColor: Background Size: 600, 500
2	pbLogo	The Movers logo.	BackColor: Background Size: 300, 100 Location: 100, 100 Anchor: None
3	pnlSignIn	The panel that holds all the sign-in controls.	BackColor: Background Size: 340, 340 Location: 220, 80 Anchor: None
4	lblUsername	The title label indicates where the user should put in their username.	ForeColor: Foreground AutoSize: True Anchor: Top, Left Location: 10, 10

			FontSize: 12pt Text: "Username"
5	tbUsername	The text box where the staff member types in their username.	Forecolor: Foreground Size: 300, 20 Location: 10, 40 Anchor: Top, Left FontSize: 12pt
6	lblPassword	The title label indicates where the user should type in their password.	Forecolor: Foreground AutoSize: True Anchor: Top, Left Location: 10, 10 FontSize: 12pt Text: "Password"
7	tbPassword	The text box where the staff member types in their password.	Forecolor: Foreground Size: 300, 20 Location: 10, 40 Anchor: Top, Left FontSize: 12pt UseSystemPasswordChar: True
8	btnSignIn	The button the user clicks to submit their password.	BackColor: Foreground ForeColor: Background HoverColor: SecondaryForeground FontSize: 20pt Text: "Sign in"

Feedback

The overall design is good, and the use of the Movers logo is a positive addition. Add a way for the user to view their password once they type it into the box and a way for staff members to toggle the appearance theme on the sign-in page so they can use their preferred appearance from the start. In addition, add a welcome message above the fields and include placeholder text within the text boxes. Finally, add descriptive error messages for when the user incorrectly types in their sign-in credentials.

Final Design



Description

ID	Item	Walk Through	Styling
1	SignInView	The main form for the sign in screen.	BackColor: Background Size: 600, 500
2	pbLogo	The Movers logo.	BackColor: Background

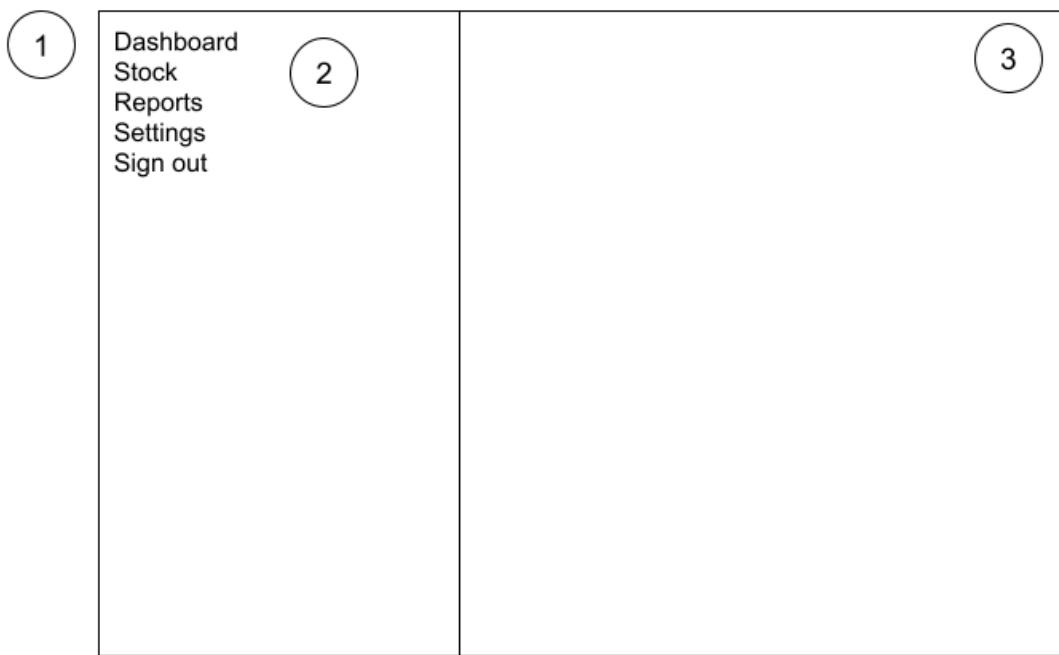
			Size: 300, 100 Location: 100, 100 Anchor: None
3	pnlSignIn	The panel that holds all the sign-in controls.	BackColor: Background Size: 340, 340 Location: 220, 80 Anchor: None CornerRadii: 10, 10, 10, 10 BorderColor: PrimaryForeground
4	lblWelcome	A label to display a welcome message.	ForeColor: Foreground AutoSize: True Anchor: Top, Left Location: 25, 26 FontSize: 24pt Text: "Username"
5	lblSignIn	The label that displays the sign-in prompt	Text: "Sign in below to access your account" ForeColor: PrimaryForeground Anchor: Top, Left Size: 282, 24 Location: 28, 65
6	tbUsername	The text box where the staff member types in their username.	ForeColor: Foreground Size: 300, 20 Location: 10, 40 Anchor: Top, Left FontSize: 12pt Placeholder: "Username" CornerRadii: 10, 10, 10, 10 BorderColor: PrimaryForeground
7	tbPassword	The text box where the staff member types in their password.	ForeColor: Foreground Size: 300, 20 Location: 10, 40 Anchor: Top, Left FontSize: 12pt UseSystemPasswordChar: True Placeholder: "Password" CornerRadii: 10, 10, 10, 10 BorderColor:

			PrimaryForeground
8	pbShowPassword	A clickable picture box containing an icon to determine if the password is shown.	BackColor: Background Size: 21, 21 Location: 278, 177
9	lblError	The error message that is shown when a user types in their password incorrectly	BackColor: Foreground ForeColor: Danger FontSize: 12pt Text: "" (Set programmatically)
10	btnSignIn	The button the user clicks to submit their password.	BackColor: Foreground ForeColor: Background HoverColor: SecondaryForeground FontSize: 20pt Text: "Sign in"
11	btnSwitchTheme	The button the user clicks when they want to change the theme	BackColor: Background ForeColor: SecondaryForeground HoverColor: SecondaryForeground

Main Page

The main page is the page of the application where the user can access the features required to complete their work. This is accessible from the side menu that consists of drop-down tabs.

Initial Design



Description

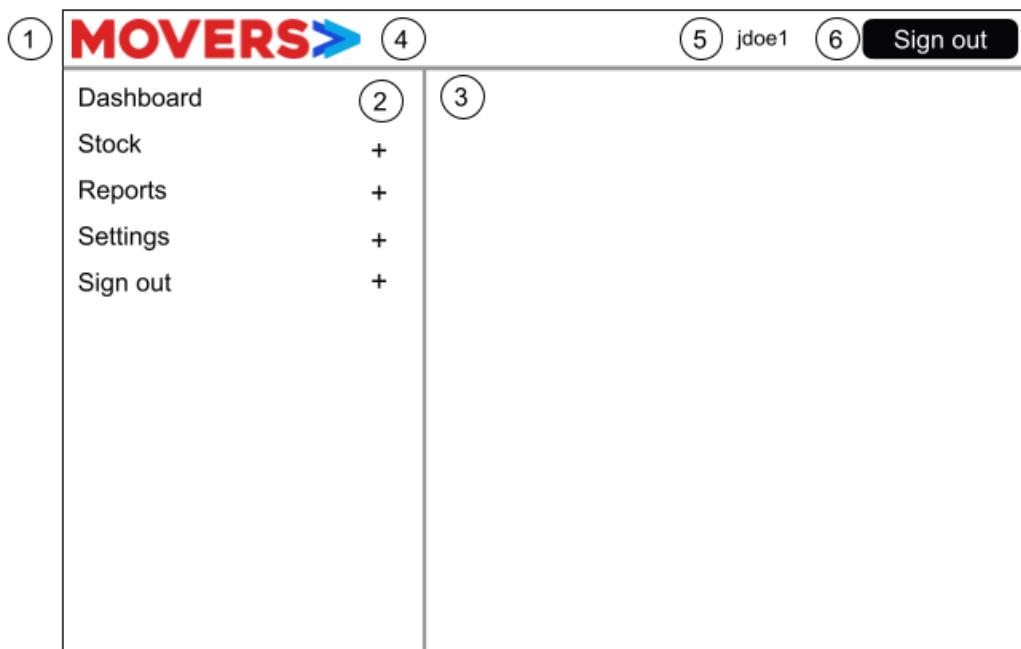
ID	Item	Walk Through	Styling
1	MasterView	The main menu form for the application.	BackColor: Background Size: 900, 650 MinimumSize: 900, 650
2	sideMenu (user control)	The side menu from which the staff member will navigate through the application.	BackColor: Background Size: 230, 570 Location: 0, 0 Dock: Left
3	pnlHolder	The panel that holds each sub-page of the application.	BackColor: Background Size: 654, 571 Location: 230, 40 Dock: Fill

Feedback

It would be useful for users to have a sign-out button at the top so they do not need to search through the menu to find it. The movers logo, along with the username of the currently logged-in staff member, would also improve the design.

Final

Design



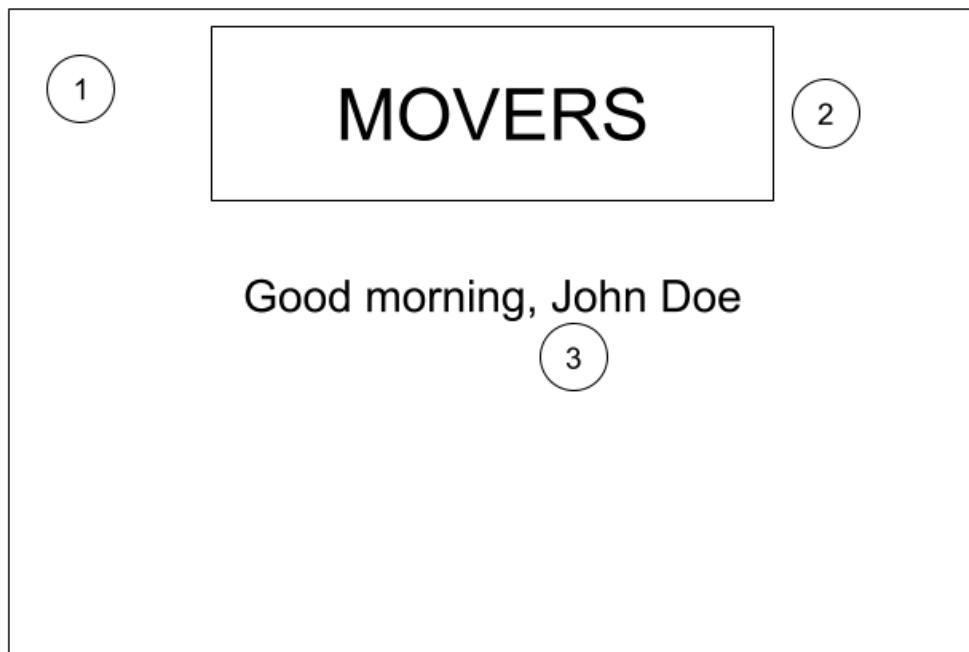
Description

ID	Item	Walk Through	Styling
1	MasterView	The main menu form for the application.	BackColor: Background Size: 900, 650 MinimumSize: 900, 650
2	sideMenu (user control)	The side menu from which the staff member will navigate through the application.	BackColor: Background Size: 230, 570 Location: 0, 0 Dock: Left
3	pnlHolder	The panel that holds each sub-page of the application.	BackColor: Background Size: 654, 571 Location: 230, 40 Dock: Fill BorderColor: PrimaryForeground
4	pbLogo	A picture box showing the Movers logo	BackColor: Background Size: 100, 20 Location: 10
5	lblUsername	The username of the currently signed in staff member.	ForeColor: Foreground BackColor: Background FontSize: 12pt AutoSize: True Location: 700, 10
6	btnSignOut	The button to sign out the currently signed-in staff member.	BackColor: Foreground ForeColor: Background HoverColor: SecondaryForeground FontSize: 20pt Text: "Sign out"

Dashboard

The dashboard is the first page the staff member will see when they sign in to their account.

Initial Design



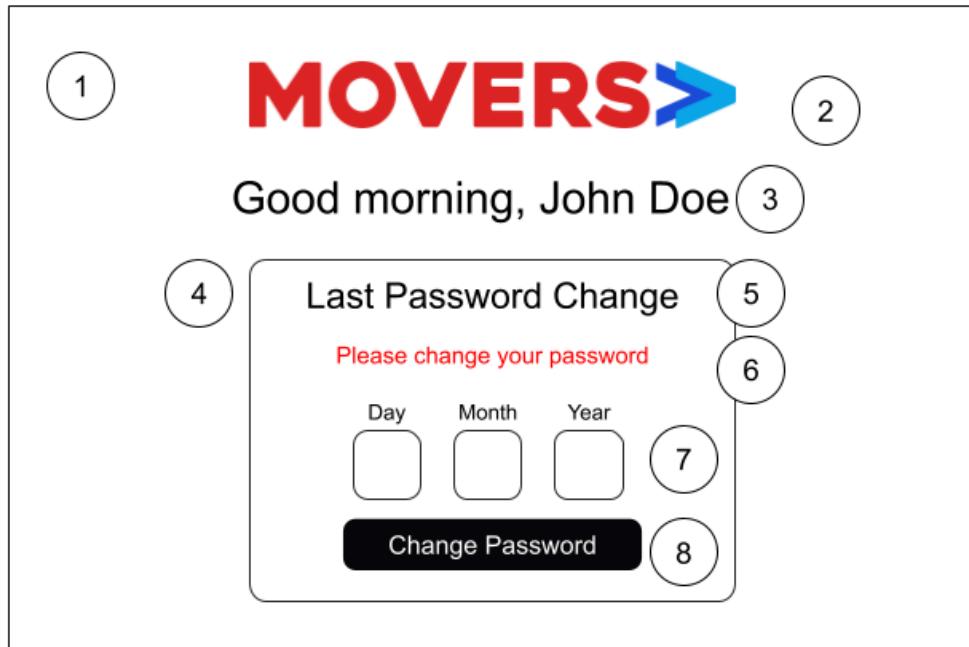
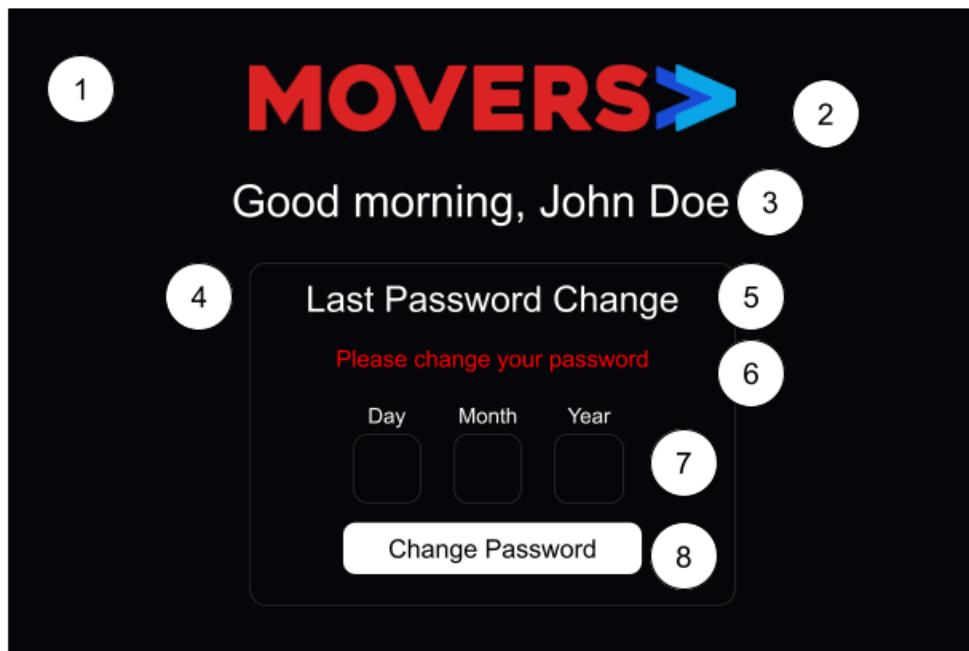
Description

ID	Item	Walk Through	Styling
1	DashboardView	The dashboard form.	BackColor: Background Size: 638, 500 MinimumSize: 600, 500
2	pbLogo	A picture box showing the Movers logo	BackColor: Background Size: 150, 50 Location: 150, 75 Anchor: None
3	lblWelcome	A label showing a welcome message to the staff member that is signed in.	BackColor: Background FontSize: 24pt AutoSize: True Size: 150, 50 Location: 300, 75

Feedback

The dashboard could be improved by the inclusion of some other details about the user's account.

Final Design



Description

ID	Item	Walk Through	Styling
1	DashboardView	The dashboard form.	BackColor: Background

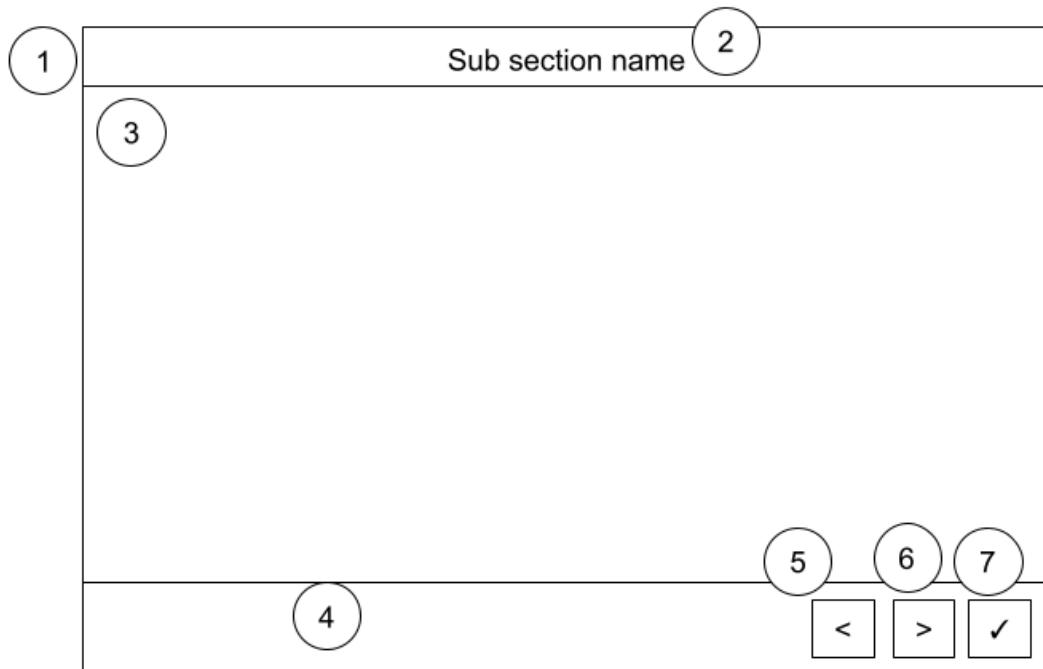
	w		Size: 638, 500 MinimumSize: 600, 500
2	pbLogo	A picture box showing the Movers logo	BackColor: Background Size: 282, 50 Location: 178, 22 Anchor: None
3	lblWelcome	A label showing a welcome message to the staff member that is signed in.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 150, 50 Location: 200, 75 Text: "Good afternoon, John Doe" (programmatically edited)
4	pnlLastPasswordChange	The panel for the password change information.	BackColor: Background BorderColor: PrimaryForeground Location: 163, 162 Size: 312, 270 CornerRadii: 10, 10, 10, 10
5	lblLastPasswordChange	The title for the password change information.	BackColor: Background ForeColor: Foreground FontSize: 20pt AutoSize: True Size: 287, 33 Location: 10, 10 Text: "Last Password Change"
6	lblChangePasswordPrompt	A prompt that appears to recommend that the user change their password when it is greater than thirty days old.	BackColor: Background ForeColor: Danger FontSize: 12pt AutoSize: True Size: 228, 19 Location: 39, 53 Text: "Please change your password"
7	diLastPasswordChange	A date input displaying the last password change	BackColor: Background ForeColor: Foreground FontSize: 12pt Size: 180, 70 Location: 61, 110 BorderColor:

			PrimaryForeground ReadOnly: True
8	btnChangePas sword	A button to redirect the user to their change password page.	BackColor: Foreground ForeColor: Background HoverColor: SecondaryForeground FontSize: 12pt Size: 252, 36 Location: 30, 200 CornerRadii: 10, 10, 10, 10 Text: "Change Password"

Add Page

The add page will be used to display any pages needed to add an item. A general add page is shown below. There will be different add pages for different items, but they will all have a similar structure.

Initial Design

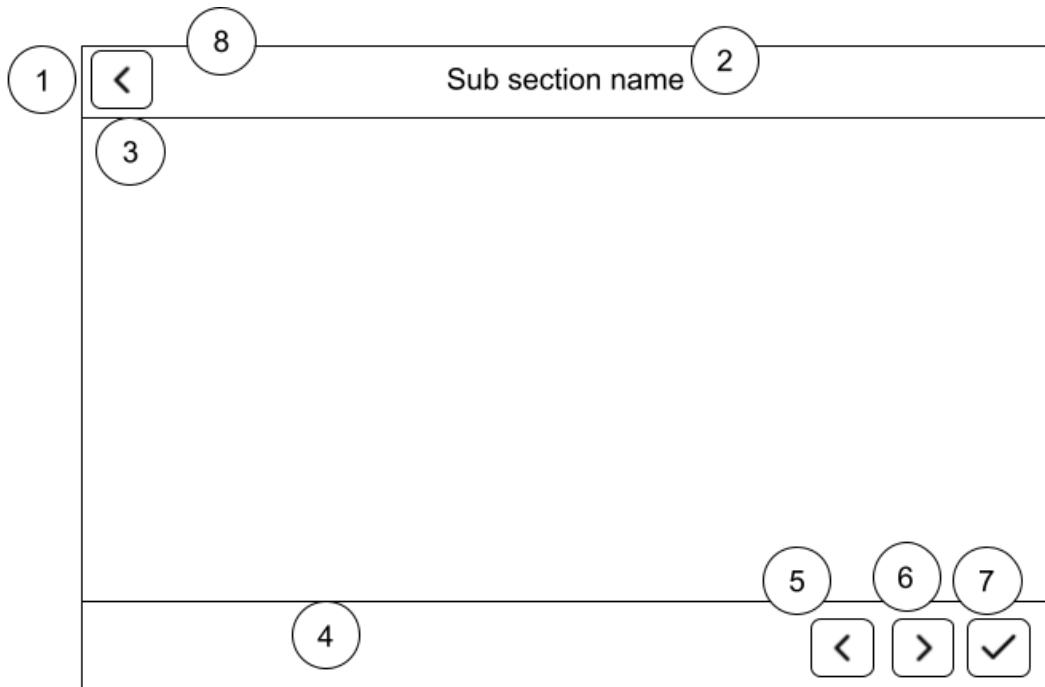
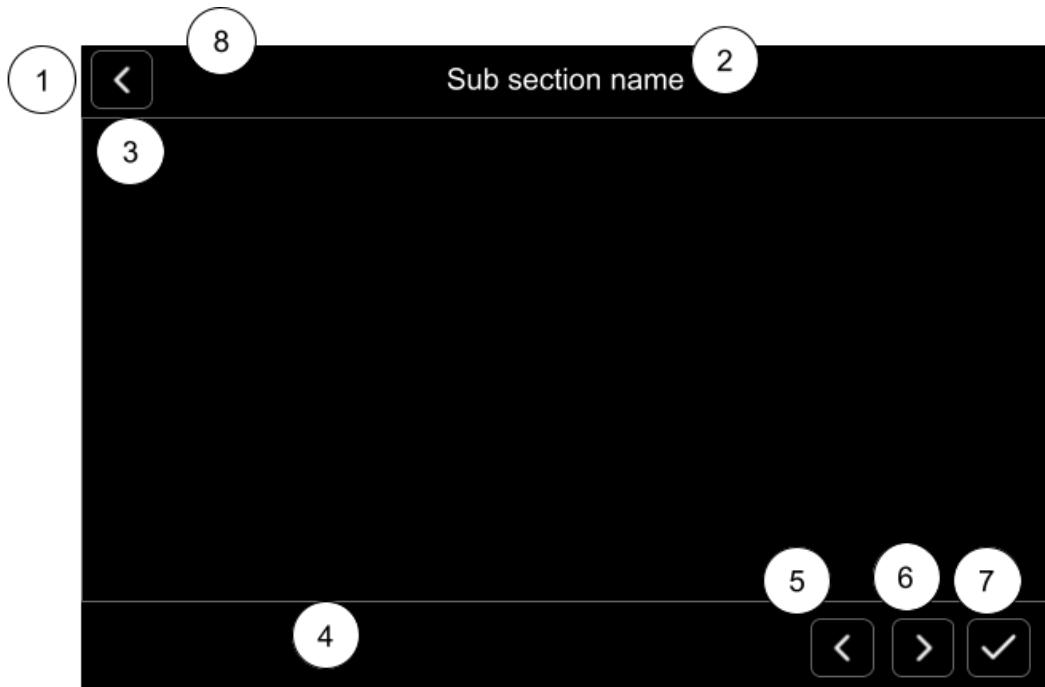


Description

ID	Item	Walk Through	Styling
1	AddView	The form for adding items	BackColor: Background Size: 700, 500
2	lblTitle	The title for the subsection.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23
3	pnlHolder	The panel that holds the search, add, edit, etc. controls.	BackColor: Background ForeColor: Foreground Size: 600, 50 Location: 10, 50
4	pnlBottom	The panel at the bottom of the page that holds the add, edit and delete.	BackColor: Background ForeColor: Foreground Dock: Bottom Size: 700, 40

5	btnBack	The button to move to the previous add page.	BackColor: Background ForeColor: Foreground Size: 40, 40 Location: 500, 10
6	btnNext	The button to move to the next add page. It is visible on all pages apart from the last.	BackColor: Background ForeColor: Foreground Size: 40, 40 Location: 545, 10
7	btnDone	The button to add the item. It is only visible on the final page.	BackColor: Background ForeColor: Foreground Size: 40, 40 Location: 550, 10

Final Design



Description

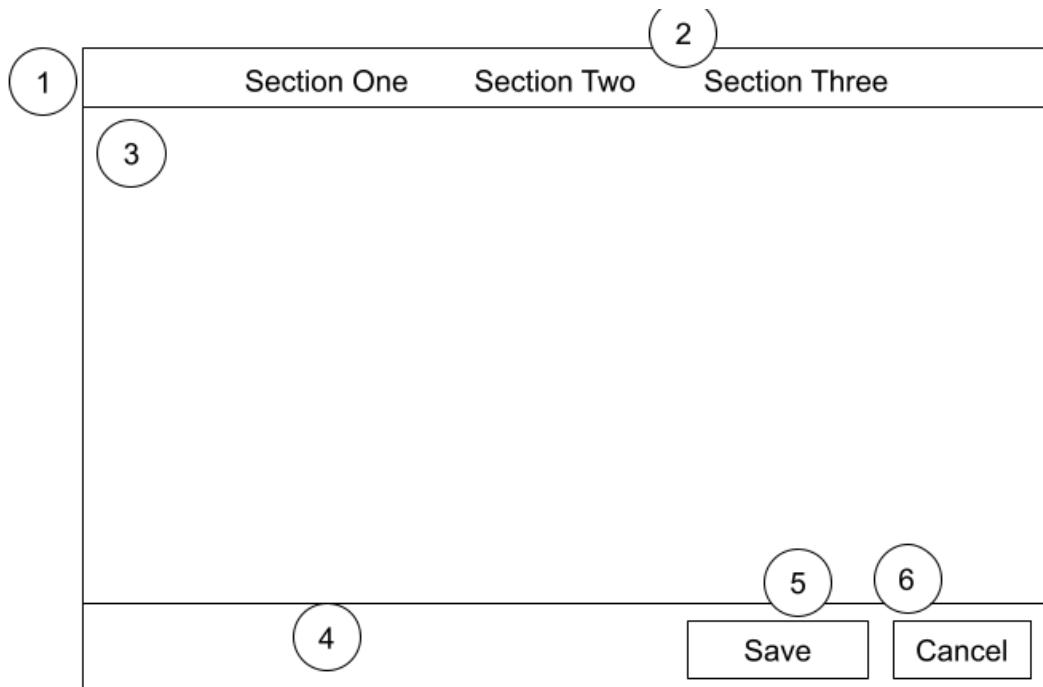
ID	Item	Walk Through	Styling
1	AddView	The general form for adding items.	BackColor: Background Size: 700, 500

2	lblTitle	The title for the subsection.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23
3	pnlHolder	The panel that holds the child view (Subview) contains data related to the specific sub-section.	BackColor: Background ForeColor: Foreground Size: 600, 50 Location: 10, 50
4	pnlBottom	The panel at the bottom of the page that holds the add, edit and delete.	BackColor: Background ForeColor: Foreground Dock: Bottom Size: 700, 40
5	btnBack	The button to move to the previous add page.	BackColor: Background ForeColor: Foreground Size: 40, 40 Location: 500, 10
6	btnNext	The button to move to the next add page. It is visible on all pages apart from the last.	BackColor: Background ForeColor: Foreground Size: 40, 40 Location: 545, 10
7	btnDone	The button to add the item. It is only visible on the final page.	BackColor: Background ForeColor: Foreground Size: 40, 40 Location: 550, 10

Edit View

The edit view will be used to edit an item. A general edit page is shown below. Edit pages will only differ in their menu items.

Initial Design

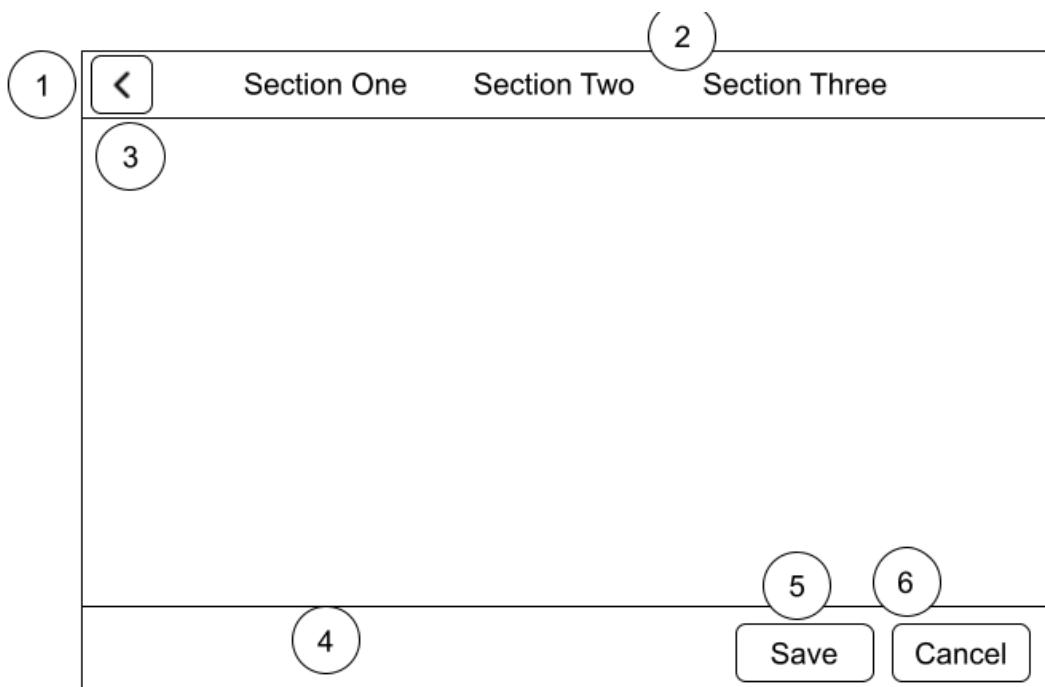
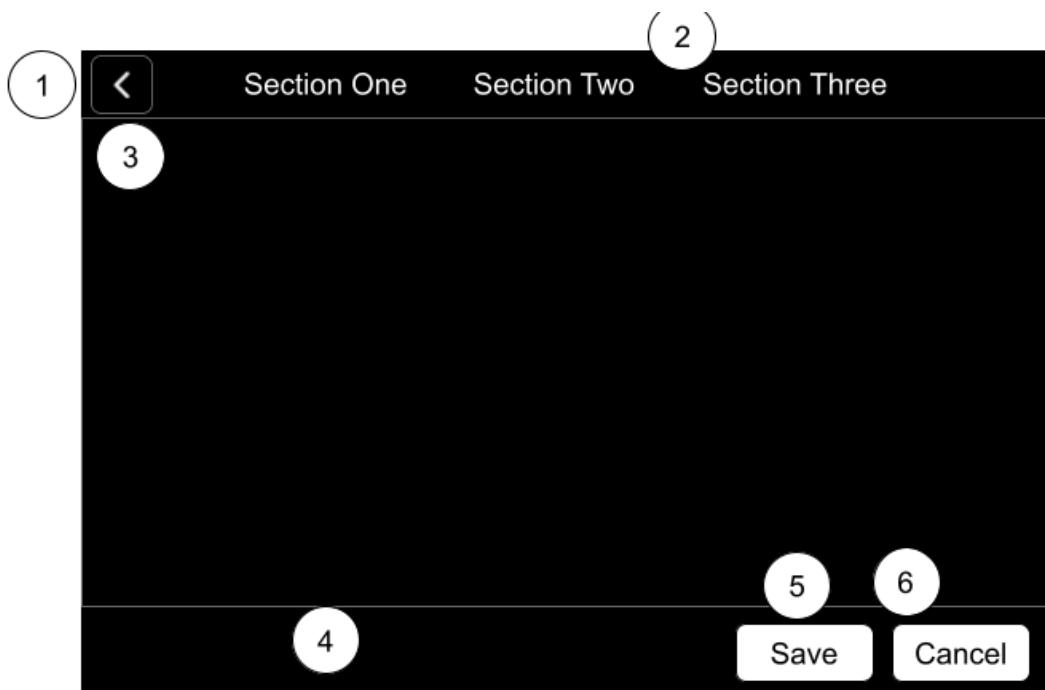


Description

ID	Item	Walk Through	Styling
1	EditView	The general form for editing items.	BackColor: Background Size: 700, 500
2	topMenu	A user control that holds a list of selectable menu items.	BackColor: Background ForeColor: Foreground FontSize: 24pt Size: 97, 39 Location: 276, 23
3	pnlHolder	The panel that holds the child view (Subview) that contains data related to the specific sub section.	BackColor: Background ForeColor: Foreground Size: 600, 50 Location: 10, 50
4	pnlBottom	The panel at the bottom of the page that holds the save and cancel buttons. This bar is only visible when there are changes to the data held within the edit subview.	BackColor: Background ForeColor: Foreground Dock: Bottom Size: 700, 40

5	btnSave	The button to save any changed data.	BackColor: Foreground ForeColor: Background Size: 40, 40 Location: 500, 10
6	btnCancel	The button to reset the data within the sub section.	BackColor: Foreground ForeColor: Background Size: 40, 40 Location: 545, 10

Final Design



Description

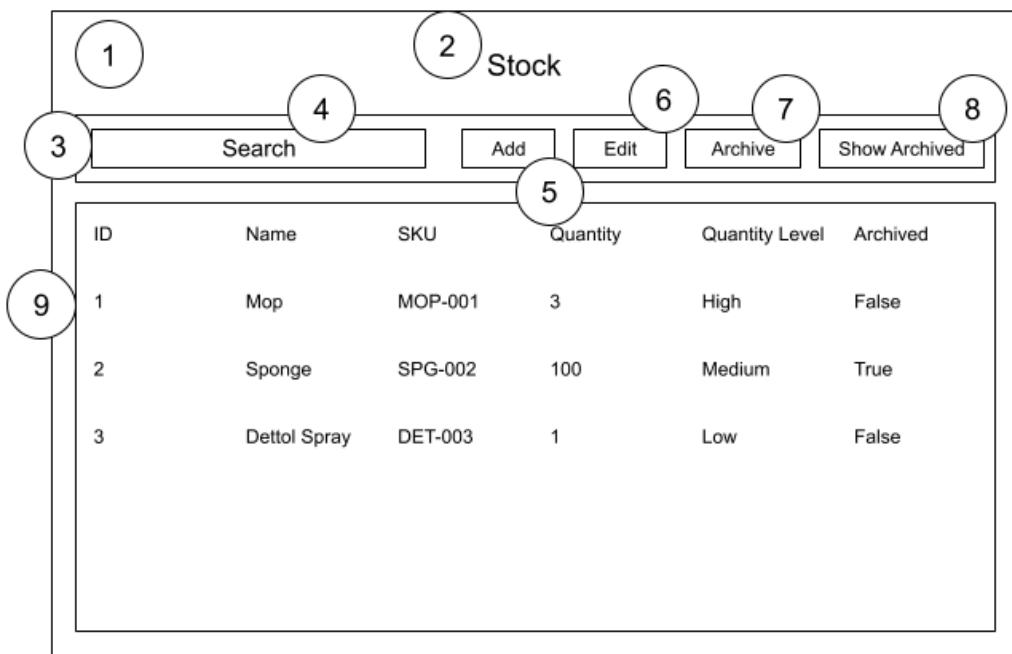
ID	Item	Walk Through	Styling
1	EditView	The general form for editing items.	BackColor: Background Size: 700, 500

2	topMenu	A user control that holds a list of selectable menu items.	BackColor: Background ForeColor: Foreground FontSize: 24pt Size: 97, 39 Location: 276, 23
3	pnlHolder	The panel that holds the child view (Subview) that contains data related to the specific sub section.	BackColor: Background ForeColor: Foreground Size: 600, 50 Location: 10, 50
4	pnlBottom	The panel at the bottom of the page that holds the save and cancel buttons. This bar is only visible when there are changes to the data held within the edit subview.	BackColor: Background ForeColor: Foreground Dock: Bottom Size: 700, 40
5	btnSave	The button to save any changed data.	BackColor: Foreground ForeColor: Background Size: 40, 40 Location: 500, 10
6	btnCancel	The button to reset the data within the sub section.	BackColor: Foreground ForeColor: Background Size: 40, 40 Location: 545, 10

Manage Stock Page

The add stock page is available to all managers. It allows the user to view all current stock and provides a gateway for adding , editing and deleting stock.

Initial Design



Description

ID	Item	Walk Through	Styling
1	DisplayStockView	The form for displaying stock.	BackColor: Background Size: 649, 500
2	lblStock	The title for the display stock form.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23
3	pnlTopBar	The panel that holds the search, add, edit, etc. controls.	BackColor: Background ForeColor: Foreground Size: 600, 50 Location: 10, 50
4	tbSearch	The text box into which the user inputs their search term to order the table by.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top

			FontSize: 12pt Placeholder: "Search"
5	btnAdd	The button the staff member clicks if they want to add a new stock item.	Location: 400, 5 Size 100, 40 Anchor: Top FontSize: 12pt Text: "Add"
6	btnEdit	The button the staff member clicks if they want to edit an existing stock item.	Location: 500, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Text: "Edit"
7	btnArchive	The button the staff member clicks if they want to archive a stock item.	Location: 600, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Text: "Archive"
8	btnShowArchived	The button the staff member clicks if they want to show the archived stock.	Location: 650, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Text: "Show Archived"
9	dataGridView	The table in which the stock items are displayed. Column headers can be clicked to sort the corresponding column. This is scrollable.	Location: 100, 100 Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor: Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold

Feedback

The initial design is good but several aspects could be improved. Firstly, icons could be more concise and intuitive than text but could be supplemented by tooltips. In addition, a way to clear the search bar could be useful. Secondly, a scroll bar would be useful for users at the side of the data grid view. Finally, the formatting of the data grid view could be improved. The addition of different colours for different levels of stock could improve the ability for staff members to see the stock level at a glance. It would also make more sense to use "yes" and "no" for archive status to improve

readability. It would also be useful to be able to edit a staff account by double-clicking the row.

Final Design

This screenshot shows the final design of a stock management application. The interface has a dark background with white text and icons. At the top center is the title "Stock" (labeled 2). To the left of the title are two small circles labeled 3 and 4, which likely represent search and sort functions. To the right of the title are several icons: a magnifying glass for search (labeled 5), a plus sign for adding items (labeled 6), a pencil for editing (labeled 7), a trash can for deleting (labeled 8), and a cube for archiving (labeled 9). Below these icons is a vertical scroll bar (labeled 10). The main content area displays a table of stock items:

ID	Name	SKU	Quantity	Quantity Level	Archived
1	Mop	MOP-001	3	High	No
2	Sponge	SPG-002	100	Medium	Yes
3	Dettol Spray	DET-003	1	Low	No

The table rows are numbered 1, 2, and 3 from top to bottom. A large circle labeled 11 is positioned in the middle of the table area. A small circle labeled 12 is located at the bottom left corner of the form.

This screenshot shows the same stock management application as above, but with a specific row highlighted. The third row, which contains the item "Dettol Spray", is shaded in gray (labeled 11). This indicates that the user has selected or double-clicked on that particular item. The rest of the interface remains the same, with the title "Stock", search and sort functions, and the scroll bar.

Description

ID	Item	Walk Through	Styling
1	DisplayStockVi	The form for displaying stock.	BackColor: Background

	ew		Size: 649, 500
2	lblStock	The title for the display stock form.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23 Text: "Stock"
3	pbSearch	A search icon to indicate the search box	BackColor: Background ForeColor: Foreground Size: 10, 10 Location: 12, 75
4	tbSearch	The text box into which the user inputs their search term to order the table by.	Forecolor: Foreground BorderColor: PrimaryForeground PlaceholderColor: Primary Size: 449, 40 Location: 12, 75 Anchor: Top FontSize: 12pt Placeholder: "Search"
5	pbClear	The picture box containing a cross icon. This can be clicked to clear the search bar.	BackColor: Background Location: 400, 5 Size 40, 40 Anchor: Top FontSize: 12pt Image: Cross
6	btnAdd	The button the staff member clicks if they want to add a new stock item.	BackColor: Background ForeColor: SecondaryForeground HoverColor: SecondaryForeground Location: 400, 5 Size 40, 40 Anchor: Top FontSize: 12pt Image: Plus
7	btnEdit	The button the staff member clicks if they want to edit an existing stock item.	BackColor: Background ForeColor: SecondaryForeground HoverColor: SecondaryForeground Location: 500, 5

			Size: 40, 40 Anchor: Top FontSize: 12pt Image: Pencil
8	btnArchive	The button the staff member clicks if they want to archive a stock item.	BackColor: Background ForeColor: SecondaryForeground HoverColor: SecondaryForeground Location: 600, 5 Size: 40, 40 Anchor: Top FontSize: 12pt Image: Archive
9	btnShowArchived	The button the staff member clicks if they want to show the archived stock.	BackColor: Background ForeColor: SecondaryForeground HoverColor: SecondaryForeground Location: 650, 5 Size: 40, 40 Anchor: Top FontSize: 12pt Image: Box
10	scrollBar	The scroll bar linked to the data grid view. It is only visible if the columns overflow the displayable size.	BackColor: Background ThumbColor: PrimaryForeground CornerRadii: 10, 10, 10, 10
11	dataGridView	The table in which the stock items are displayed. Column headers can be clicked to sort the corresponding column.	Location: 100, 100 Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor: Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold
12	pnlDataGridView	The panel that holds the data grid view to improve the user interface by adding a rounded border.	BackColor: Background BorderColor: PrimaryForeground Location: 163, 162

Candidate number: 8346

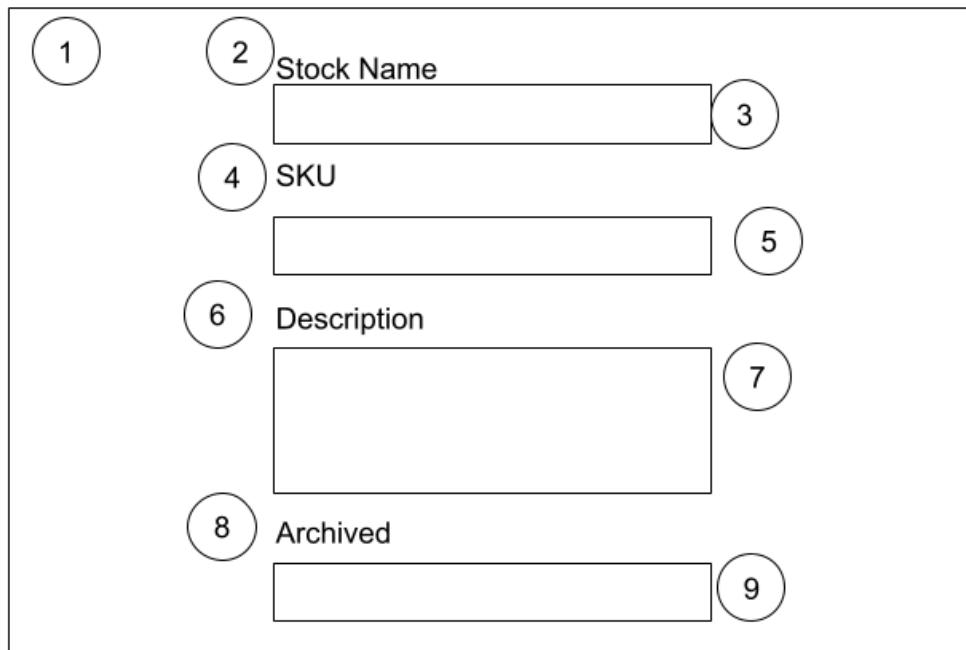
Centre Number: 71571

		Size: 312, 270 CornerRadii: 10, 10, 10, 10
--	--	--

Stock Details Page

This is the page where the staff member will be able to edit or view the stock item details.

Initial Design



Description

ID	Item	Walk Through	Styling
1	StockDetailView	The form for viewing and editing stock details.	BackColor: Background Size: 654, 419
2	lblStockName	Label for the stock name input.	FontSize: 14pt ForeColor: Foreground Location: 135, 5 AutoSize: True
3	tbStockName	Textbox for entering the stock name.	BackColor: Background BorderColor: PrimaryForeground Size: 355, 40

			Placeholder: "Dettol Spray"
4	lblSKU	Label for the SKU input.	FontSize:14pt ForeColor: White Location: 135, 88 AutoSize: True
5	tbSKU	Textbox for entering the stock SKU.	BackColor: Background, BorderColor: PrimaryForegroun d Size: 355, 40 Placeholder: "DET-123"
6	lblStockDescription	Label for the stock description input.	FontSize: 14pt ForeColor: White Location: 135, 224 AutoSize: True
7	tbStockDescription	Multi-line textbox for stock description.	BackColor: Background, BorderColor: PrimaryForegroun d, Size: 355, 83 Placeholder: "Dettol antibacterial surface cleaning spray..."
8	tbArchived	A text box for the staff member to type the archived status ("yes" or "no").	BackColor: Background, BorderColor: PrimaryForegroun d Size: 355, 83

Feedback

It would be useful to include a character limit label in the description text box so the staff member knows how concise they need to be. It would also be more intuitive if the archived field were a checkbox rather than a text box.

Final

Design

1

2 Stock Name
Dettol Spray

3

4 SKU
DET-001

5

6 Description
Kills 99% of bacteria

7

8 Archived
22/500

9

10

11

1

2 Stock Name
Dettol Spray

3

4 SKU
DET-001

5

6 Description
Kills 99% of bacteria

7

8 Archived
22/500

9

10

11

Description

ID	Item	Walk Through	Styling
1	StockDetailView	The form for viewing and editing stock details.	BackColor: Background Size: 654, 419

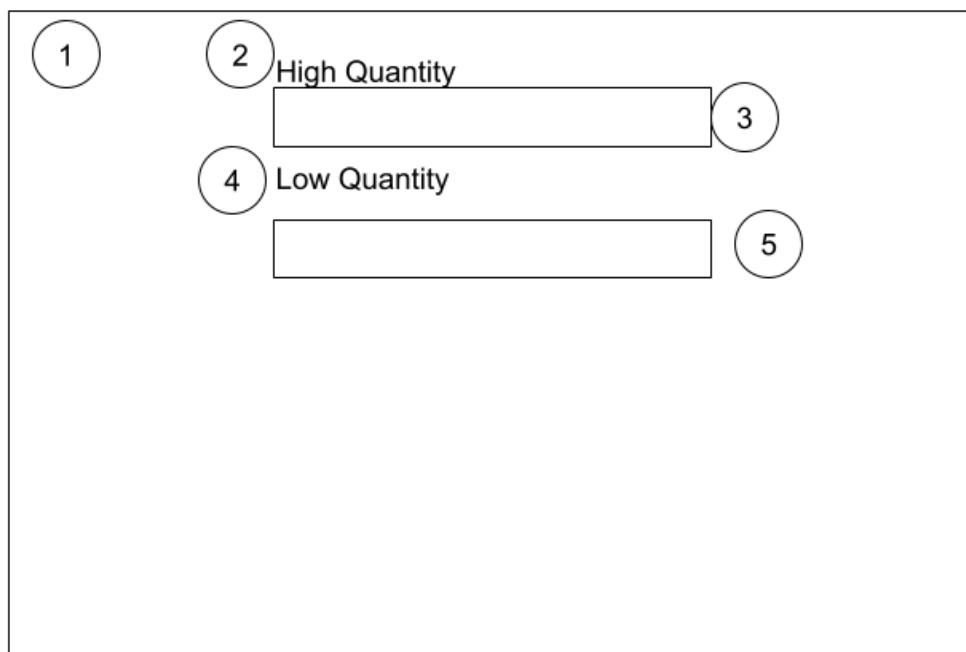
2	lblStockName	Label for the stock name input.	FontSize: 14pt ForeColor: Foreground Location: 135, 5 AutoSize: True
3	tbStockName	Textbox for entering the stock name.	BackColor: Background BorderColor: PrimaryForegroun d CornerRadius: 10, 10, 10, 10 Size: 355, 40 Placeholder: "Dettol Spray"
4	lblSKU	Label for the SKU input.	FontSize:14pt ForeColor: White Location: 135, 88 AutoSize: True
5	tbSKU	Textbox for entering the stock SKU.	BackColor: Background, BorderColor: PrimaryForegroun d CornerRadius: 10, 10, 10, 10 Size: 355, 40 Placeholder: "DET-123"
6	lblStockDescriptio n	Label for the stock description input.	FontSize: 14pt ForeColor: White Location: 135, 224 AutoSize: True
7	tbStockDescription	Multi-line textbox for stock description.	BackColor: Background, BorderColor: PrimaryForegroun d, CornerRadius: 10, 10, 10, 10

			Size: 355, 83 Placeholder: "Dettol antibacterial surface cleaning spray..."
8	lblArchived	Label for the archived status.	FontSize: 14pt ForeColor: White Location: 10, 9 AutoSize: True
9	lblCharacterLimit	Label showing character limit for description.	Font: 12pt ForeColor: Foreground Size: 150, 24 Location: 340, 345 Text: "0/500"
10	pnlArchived	Panel containing archive controls.	BorderColor: PrimaryForeground, BorderThickness: 1, CornerRadius: 10 Size: 35541, Location: 135, 374
11	rbArchived	Radio button for archived status.	CheckColor: White BorderThickness: 1.7 Size: 17, 17, Location: 324, 12

Stock Warning Page

The page where the staff member can edit or view the levels for which a stock warning is displayed.

Initial Design



Description

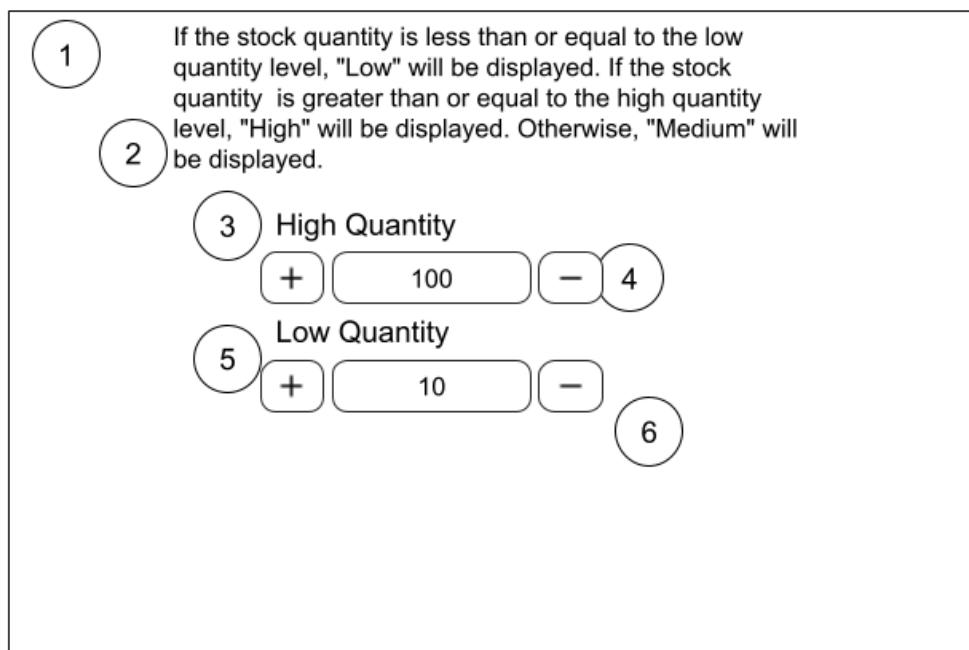
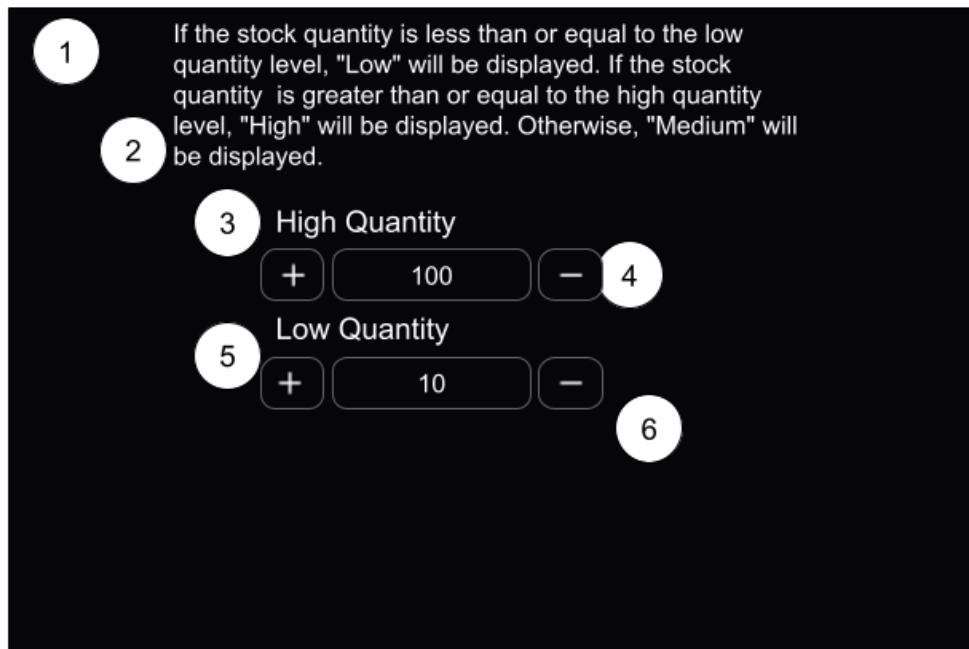
ID	Item	Walk Through	Styling
1	ManageStockWarningView	The main form for managing stock warnings.	BackColor: Background Size: 654, 419
2	lblHighQuantity	A label indicating the high quantity level.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
3	tbHighQuantity	A text box for setting the high quantity level.	BackColor: Background Location: 202, 240 Size: 250, 41

4	lblLowQuantity	A label indicating the low quantity level.	FontSize: 14pt ForeColor: Foreground Location: 202, 124 Size: 173, 23
5	tbLowQuantity	A text box for setting the low quantity level.	BackColor: Background Location: 202, 155 Size: 250, 41

Feedback

The initial design could perhaps be improved by the addition of a help message describing how each quantity level works. It could also be useful to add an up-down instead of a text box for setting the quantity.

Final Design



Description

ID	Item	Walk Through	Styling
1	ManageStockWarningView	The main form for managing stock warnings.	BackColor: Background Size: 654, 419

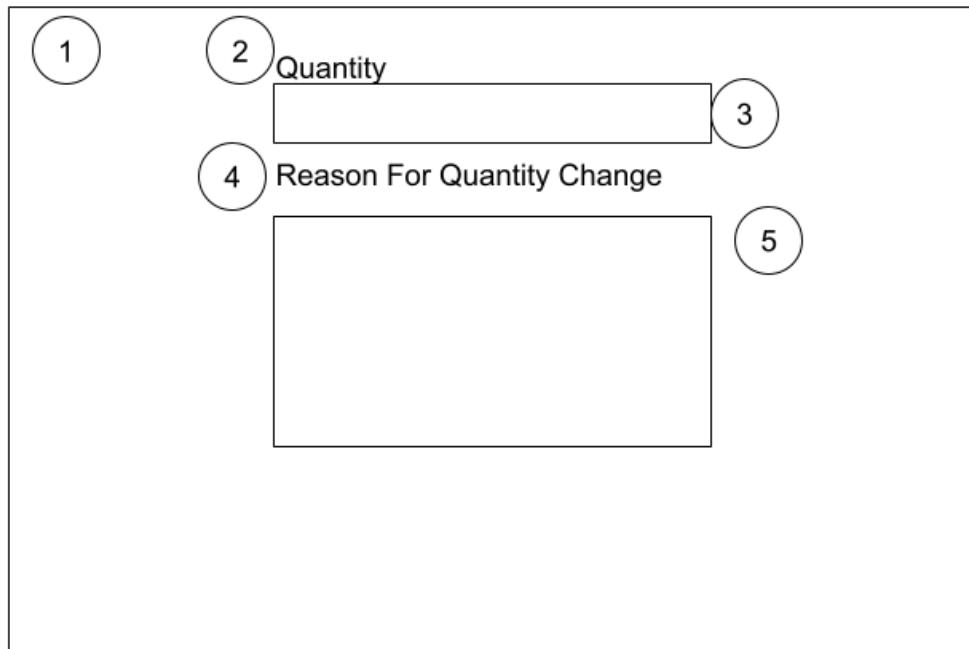
2	lblHighQuantity	A label indicating the high quantity level.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
3	nudHighQuantity	A numeric up-down control for setting the high quantity level.	BackColor: Background Location: 202, 240 Size: 250, 41 Maximum: 1000000
4	lblLowQuantity	A label indicating the low quantity level.	FontSize: 14pt ForeColor: Foreground Location: 202, 124 Size: 173, 23
5	nudLowQuantity	A numeric up-down control for setting the low quantity level.	BackColor: Background Location: 202, 155 Size: 250, 41 Maximum: 1000000
6	lblInfo	A label providing additional information about stock levels.	FontSize: 12pt ForeColor: Foreground Location: 132, 8 Size: 391, 98 Text: "If the stock quantity is less than or equal to the low quantity level, "Low" will be displayed. If the stock quantity is greater than or equal to the high quantity level, "High" will be displayed. Otherwise,

			"Medium" will be displayed."
--	--	--	------------------------------

Stock Quantity Page

This is the page where the user can edit or view the stock quantity.

Initial Design



Description

ID	Item	Walk Through	Styling
1	ManageStockQuantityView	The main form for managing stock quantity.	BackColor: Background Size: 654, 419
2	lblQuantity	A title label indicating the quantity.	FontSize: 14pt ForeColor: Foreground Location: 200, 200 Size: 200, 40
3	tbQuantity	A text box for the current stock quantity.	BackColor: Background Location: 202, 240 Size: 250, 41
4	lblLowQuantity	A label indicating the low quantity level.	FontSize: 14pt ForeColor: Foreground

			Location: 202, 124 Size: 173, 23
5	tbLowQuantity	A text box for setting the low quantity level.	BackColor: Background Location: 202, 155 Size: 250, 41

Feedback

To improve the design, numeric up-downs could be used instead of text boxes. A way to add or remove stock in bulk could also help prevent mathematical errors when changing stock values.

Final Design

1 2 Quantity
 + 100 - 3

4 Bulk Add
 + 100 - 6 Add

7 Bulk Remove
 + 100 - 8 9 Remove

10 Reason for Quantity Change (optional)
 Your staff account is linked to this quantity change 11

12 Extra stock used in a cleaning job.

1 2 Quantity
 + 100 - 3

4 Bulk Add
 + 100 - 5 6 Add

7 Bulk Remove
 + 100 - 8 9 Remove

10 Reason for Quantity Change (optional)
 Your staff account is linked to this quantity change 11

12 Extra stock used in a cleaning job.

Description

ID	Item	Walk Through	Styling
1	ManageStockQuantityView	The main form for managing stock quantity.	BackColor: Background Size: 654, 419

2	lblQuantity	A title label for the current stock quantity.	Location: 135, 3 Size: 80, 23 AutoSize = true, FontSize = 14, ForeColor = Foreground
3	nudQuantity	The numeric up-down to view or edit the current quantity level.	Location: 135, 34 Size: 267, 41 BackColor = Background, Maximum = 999999999
4	lblBulkAdd	The label indicating the bulk add field.	Location: 135, 83 Size: 86, 23 AutoSize = true, FontSize = 14, ForeColor = Foreground
5	nudBulkAdd	The numeric up-down to set the quantity to bulk add.	Location: 135, 114 Size: 267, 41 BackColor = Background Maximum = 999999999
6	btnAdd	The button to initiate the add.	Location: 415, 114 Size: 86, 40 FontSize = 12 ForeColor = Background Cursor = Hand CornerRadii = 10, 10, 10, 10
7	lblBulkRemove	The label indicating the bulk remove field below.	Location: 135, 166 Size: 122, 23 AutoSize = true, FontSize = 14, ForeColor = Foreground

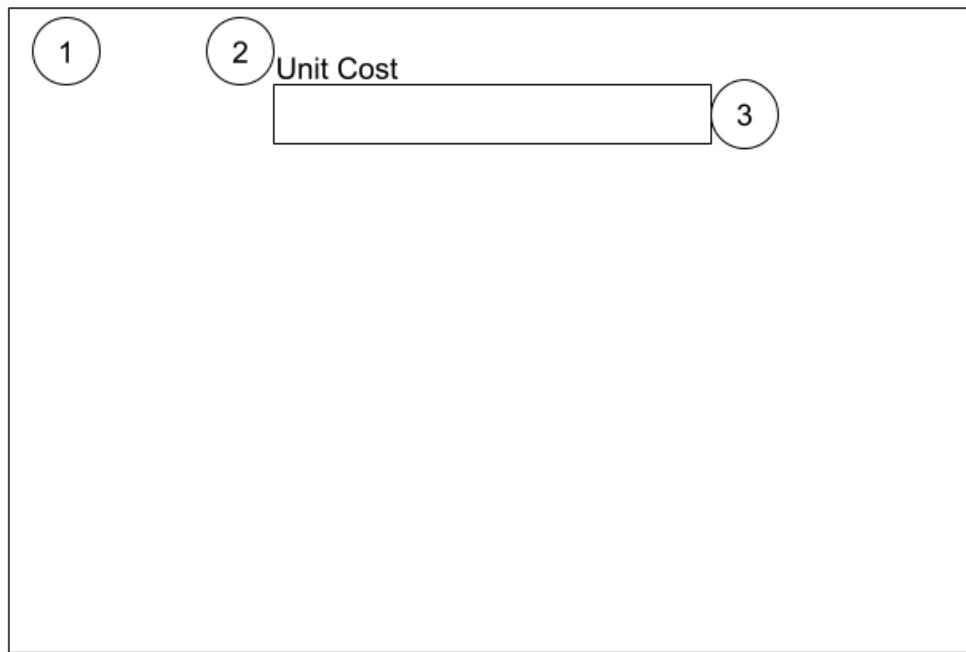
8	nudBulkRemove	The numeric up-down to set the quantity to bulk remove.	Location: 135, 197 Size: 267, 41 BackColor = Background Maximum = 999999999
9	btnRemove	The button to initiate the bulk remove.	Location: 415, 197 Size: 86, 40 FontSize = 12 ForeColor = Background Cursor = Hand CornerRadii = 10, 10, 10, 10
10	lblReasonForChange	The title label for the reason for quantity change text input.	Location: 135, 249 Size: 331, 23 AutoSize = true, FontSize = 14, ForeColor = Foreground
11	lblStaffLink	The label reminds the staff member that their staff account is linked to the quantity change.	Location: 135, 27 Size: 366, 19 AutoSize = true FontSize = 12 ForeColor = PrimaryForeground
12	tbReasonForQuantityChange	The textbox where the staff member can optionally input the reason for the quantity change.	Location: 135, 307 Size: 366, 80 BackColor = Background BorderColor = PrimaryForeground, MultiLine = true MaxLength = 500 PlaceholderTextColor = PrimaryForeground

	lblCharacterLimit	The label indicates to the staff member how long their reason for quantity change message is.	Location: 351, 392 Size: 150, 24 FontSize = 12 ForeColor = PrimaryForeground
--	-------------------	---	---

Stock Unit Cost Page

The stock unit cost page is where the staff member can change the cost of the stock item they are currently editing.

Initial Design



Description

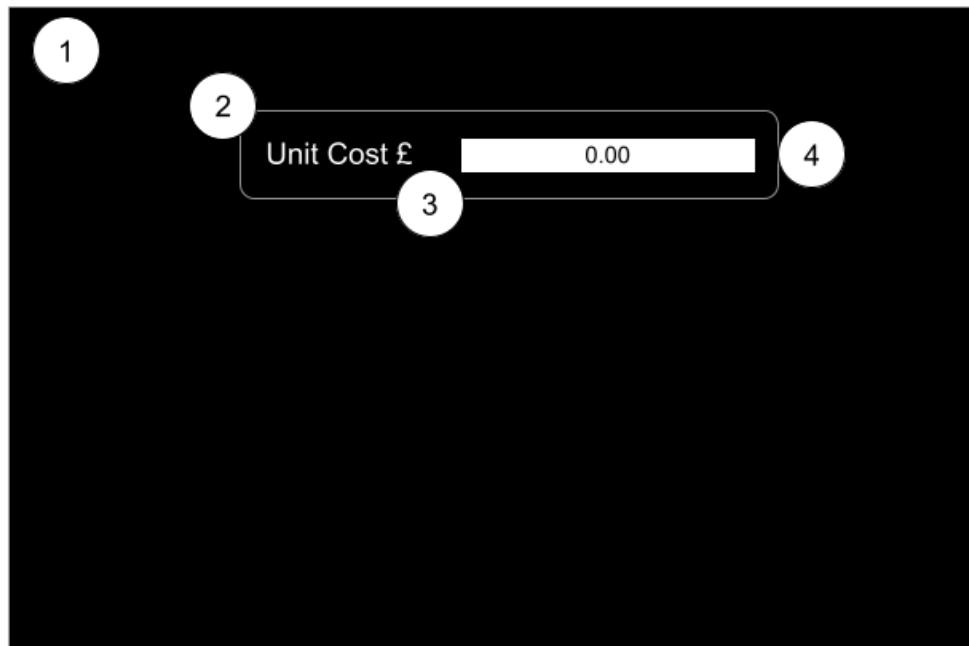
ID	Item	Walk Through	Styling
1	ManageStockUnitCostView	The form for managing a stock item's unit cost.	BackColor: Background Size: 654, 419
2	lblUnitCost	A title label for the stock unit cost.	Location: 135, 3 Size: 80, 23 AutoSize = true, FontSize = 14, ForeColor = Foreground
3	tbUnitCost	A text box for the staff member to view or type in the unit cost.	Location: 135, 34 Size: 267, 41 BackColor = Background

Feedback

This is a good design. It could be improved by adding a numeric up down to allow for the currency to be input more easily. The pound symbol (£) could also be added before the input for the unit cost to show that the cost is in pounds.

Final Design

A wireframe diagram showing a rectangular input field with rounded corners. Inside the field, the text "0.00" is centered. To the left of the input field, the text "Unit Cost £" is displayed. Four numbered callouts point to specific parts of the interface: callout 1 points to the top-left corner of the input field; callout 2 points to the "Unit Cost £" label; callout 3 points to the center of the input field; and callout 4 points to the top-right corner of the input field.



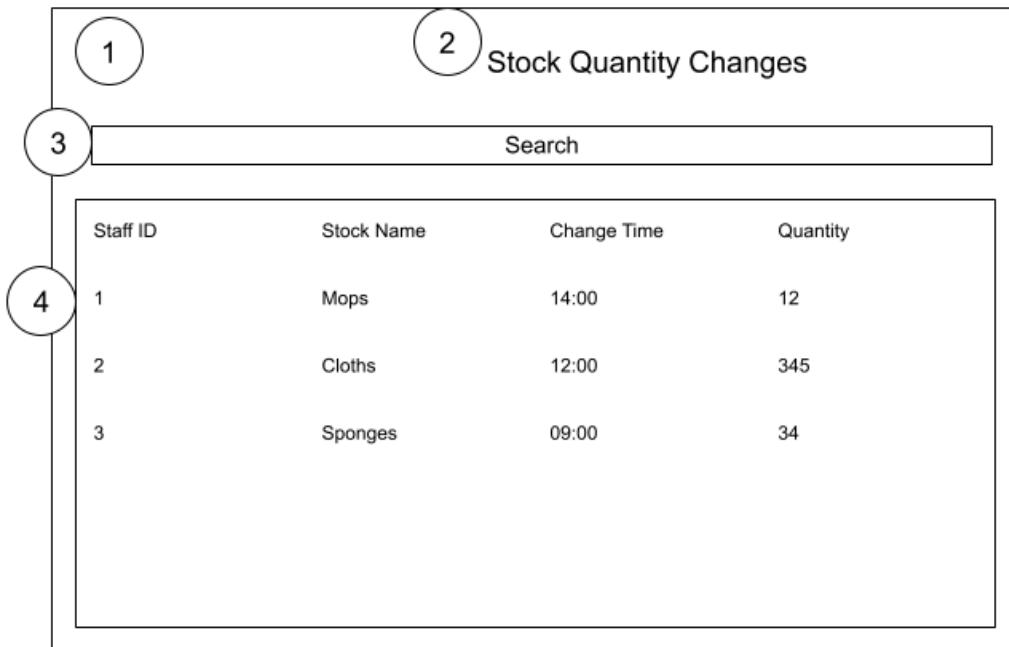
Description

ID	Item	Walk Through	Styling
1	ManageStockUnitCostView	The form for managing a stock item's unit cost.	BackColor: Background Size: 654, 419
2	lblUnitCost	A title label for the stock unit cost.	Location: 5, 13 Size: 103, 23 AutoSize = true, FontSize = 14, ForeColor = Foreground
3	nudUnitCost	A numeric up-down control allows the staff member to view or type the unit cost.	Location: 116, 13 Size: 228, 27 BackColor = Background Maximum: 1000000 Minimum: 0
4	pnlUnitCost	A panel that holds the controls.	Location 146, 44 Size: 355, 49 CornerRadii: 10, 10, 10, 10 BorderColor: PrimaryForeground

Manage Stock Quantity Changes Page

The stock quantity changes page allows staff members to see when the stock quantity changed, who was responsible and what the quantity changed to.

Initial Design



Description

ID	Item	Walk Through	Styling
1	DisplayStockQuantityChangesView	The form for displaying stock quantity changes.	BackColor: Background Size: 649, 500
2	lblStockQuantityChanges	The title for the display stock quantity changes form.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23
3	tbSearch	The text box into which the user inputs their search term to order the table by.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt Placeholder: "Search"
4	dataGridView	The table in which the stock quantity change items are displayed. Column	Location: 100, 100 Size: 600, 400

		headers can be clicked to sort the corresponding column. This is scrollable.	Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor: Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold
--	--	--	--

Feedback

The design is a good start but could be improved by having a way to view a specific stock quantity change in more detail. Additionally, more columns to provide more data could be useful. Finally, it would be good if only relevant stock quantity changes were shown, not the archived ones.

Final Design

This image shows the final design of a 'Stock Quantity Changes' form. The interface has a dark background with white text and icons. At the top center is the title 'Stock Quantity Changes' (2). To its left is a search bar with a magnifying glass icon (3) and the word 'Search' (4). To its right are three icons: a close button (5), a refresh/circular arrow (6), and a 3D cube (7). Below the title is a table with the following columns: Username, Stock Name, SKU, Date, Quantity, and Archived. The table contains three rows of data:

Username	Stock Name	SKU	Date	Quantity	Archived
jdoe	Mops	MOP-001	12/09/2025	12	12
jdoe	Cloths	CLH-002	13/09/2023	345	345
bjones	Sponges	SPG-003	01/01/2020	34	34

On the far right edge of the table area is a vertical scroll bar (8). In the bottom right corner of the table area is a small circle containing the number 9 (9).

This image shows the initial design of the same 'Stock Quantity Changes' form. It has a light gray background with black text and icons. The layout and data are identical to the final design, including the table structure and the three data rows. The scroll bar on the right is also present.

Description

ID	Item	Walk Through	Styling
1	DisplayStockQuantityChange	The form for displaying stock quantity changes.	BackColor: Background Size: 649, 500

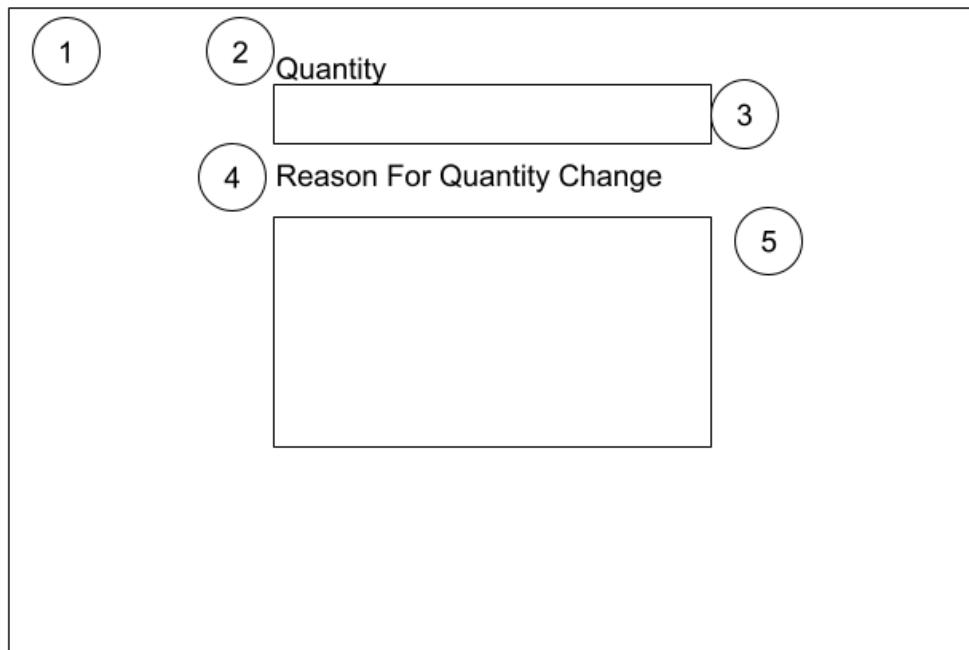
	sView		
2	lblStockQuantityChanges	The title for the display stock quantity changes form.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23
3	pbSearch	A search icon to indicate the search box.	BackColor: Background ForeColor: Foreground Size: 10, 10 Location: 12, 75
4	tbSearch	The text box into which the user inputs their search term to order the table by.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt Placeholder: "Search"
5	btnView	The button the staff member clicks if they want to view a stock quantity change.	BackColor: Background ForeColor: SecondaryForeground HoverColor: SecondaryForeground Location: 400, 5 Size 40, 40 Anchor: Top FontSize: 12pt Image: Eye
6	btnShowArchived	The button the staff member clicks if they want to show the archived stock quantity changes.	BackColor: Background ForeColor: SecondaryForeground HoverColor: SecondaryForeground Location: 650, 5 Size: 40, 40 Anchor: Top FontSize: 12pt Image: Box
7	dataGridView	The table in which the stock quantity change items are displayed. Column headers can be clicked to sort the corresponding column. This is scrollable.	Location: 100, 100 Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor: Foreground

			SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold
8	pnlDataGridView	The panel that holds the data grid view to improve the user interface by adding a rounded border.	BackColor: Background BorderColor: PrimaryForeground Location: 163, 162 Size: 312, 270 CornerRadii: 10, 10, 10, 10
9	scrollBar	The scroll bar is linked to the data grid view. It is only visible if the columns overflow the displayable size.	BackColor: Background ThumbColor: PrimaryForeground CornerRadii: 10, 10, 10, 10

Stock Quantity Change Page

This page will be displayed when a staff member wants to view a specific stock quantity change in greater detail.

Initial Design



Description

ID	Item	Walk Through	Styling
1	StockQuantityChangeView	The form for viewing a specific stock quantity change in more detail.	BackColor: Background Size: 654, 419
2	lblQuantity	The label to display the title for the value for which the quantity was changed.	FontSize: 14pt ForeColor: Foreground Location: 135, 5 AutoSize: True
3	tbQuantity	The textbox that displays the quantity after the quantity change.	BackColor: Background BorderColor: PrimaryForeground Size: 355, 40

			Readonly: True
4	lblReasonForQuantityChange	Label for the quantity change text box.	FontSize:14pt ForeColor: White Location: 135, 88 AutoSize: True
5	tbReasonForQuantityChange	The textbox that displays the reason for the quantity change.	BackColor: Background, BorderColor: PrimaryForeground Size: 355, 40 Readonly: True

Feedback

This page could be improved by adding more information about the quantity change, such as the username of the staff member who made the change and the date.

Final Design

A form design with a black background and white text. The form consists of twelve numbered fields:

- 1: A small circle containing the number 1.
- 2: "Stock Name" followed by a text input field.
- 3: A small circle containing the number 3.
- 4: "Quantity" followed by a text input field.
- 5: A small circle containing the number 5.
- 6: "Username" followed by a text input field.
- 7: A small circle containing the number 7.
- 8: "Date Of Change" followed by three input fields labeled "Day", "Month", and "Year".
- 9: A small circle containing the number 9.
- 10: "Reason For Quantity Change" followed by a large text input field.
- 11: A small circle containing the number 11.
- 12: A small circle containing the number 12.

A form design with a white background and black text. The form consists of twelve numbered fields:

- 1: A small circle containing the number 1.
- 2: "Stock Name" followed by a text input field.
- 3: A small circle containing the number 3.
- 4: "Quantity" followed by a text input field.
- 5: A small circle containing the number 5.
- 6: "Username" followed by a text input field.
- 7: A small circle containing the number 7.
- 8: "Date Of Change" followed by three input fields labeled "Day", "Month", and "Year".
- 9: A small circle containing the number 9.
- 10: "Reason For Quantity Change" followed by a large text input field.
- 11: A small circle containing the number 11.
- 12: A small circle containing the number 12.

Description

ID	Item	Walk Through	Styling
----	------	--------------	---------

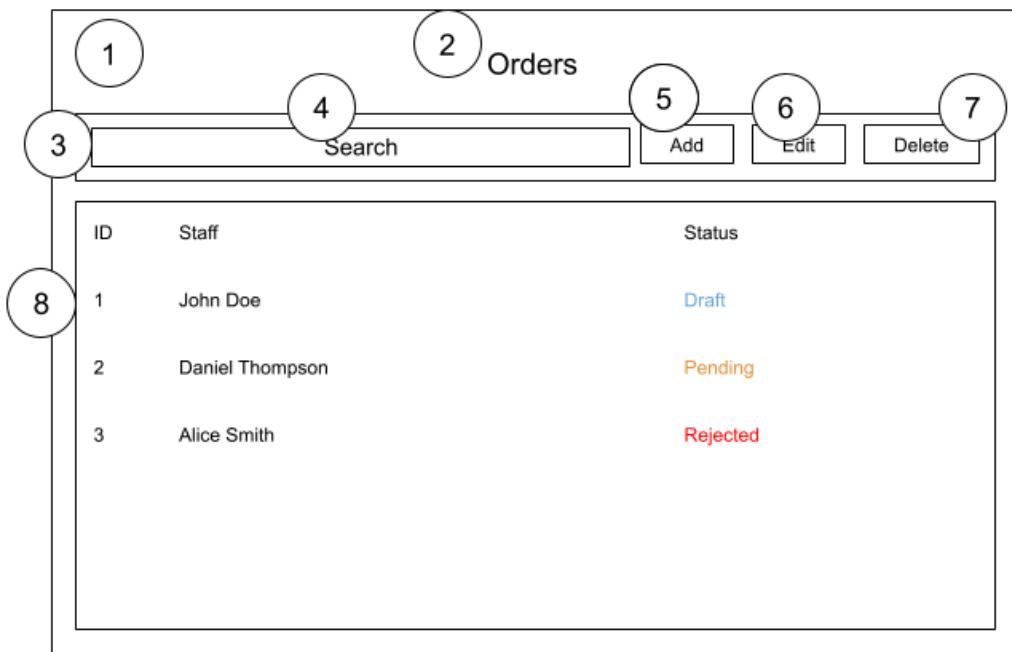
1	StockQuantityChangeView	The form for viewing a specific stock quantity change in more detail.	BackColor: Background Size: 654, 419
2	lblStockName	The label displays the stock name title.	FontSize: 14pt ForeColor: Foreground Location: 135, 5 AutoSize: True
3	tbStockName	The textbox that displays the quantity after the quantity change.	BackColor: Background BorderColor: PrimaryForeground Size: 355, 40 ReadOnly: True
4	lblQuantity	The label to display the title for the value for which the quantity was changed.	FontSize: 14pt ForeColor: Foreground Location: 135, 5 AutoSize: True
5	tbQuantity	The textbox that displays the quantity after the quantity change.	BackColor: Background BorderColor: PrimaryForeground Size: 355, 40 ReadOnly: True
6	lblUsername	The label to display the title for the value for the username.	FontSize: 14pt ForeColor: Foreground Location: 135, 5 AutoSize: True
7	tbUsername	The textbox that displays the username of the staff member who made the quantity change.	BackColor: Background BorderColor: PrimaryForeground Size: 355, 40

			Readonly: True
9	lblDate	The label to display the title for the date of the quantity change.	FontSize: 14pt ForeColor: Foreground Location: 135, 5 AutoSize: True
10	diQuantityChange Date	A date input displaying the date of the quantity change	BackColor: Background ForeColor: Foreground FontSize: 12pt Size: 180, 70 Location: 61, 110 BorderColor: PrimaryForeground ReadOnly: True
11	lblReasonForQuantityChange	Label for the quantity change text box.	FontSize:14pt ForeColor: White Location: 135, 88 AutoSize: True
12	tbReasonForQuantityChange	The textbox that displays the reason for the quantity change.	BackColor: Background, BorderColor: PrimaryForeground Size: 355, 40 Readonly: True

Manage Orders Page

This is the page where the cleaning manager can manage the upcoming orders, draft orders and view historic order data.

Initial Design



Description

ID	Item	Walk Through	Styling
1	DisplayOrderView	The form for displaying orders.	BackColor: Background Size: 649, 500
2	lblOrders	The title for the display order form.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23
3	pnlTopBar	The panel that holds the search, add, edit, etc. controls.	BackColor: Background ForeColor: Foreground Size: 600, 50 Location: 10, 50
4	tbSearch	The text box into which the user inputs their search term to order the table by.	Forecolor: Foreground Size: 300, 40 Location: 5, 5

			Anchor: Top FontSize: 12pt Placeholder: "Search"
5	btnAdd	The button the staff member clicks if they want to add a new order.	Location: 400, 5 Size 100, 40 Anchor: Top FontSize: 12pt Text: "Add"
6	btnEdit	The button the staff member clicks if they want to edit an existing drafted order.	Location: 500, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Text: "Edit"
7	btnDelete	The button the staff member clicks if they want to delete a drafted order.	Location: 600, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Text: "Archive"
8	dataGridView	The table in which the stock items are displayed. Column headers can be clicked to sort the corresponding column. This is scrollable.	Location: 100, 100 Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor: Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold

Feedback

The design could be improved by adding different colours to indicate the state of an order. Additionally, the use of icons would make the screen more user-friendly.

Final Design

This screenshot shows the final design of the 'Orders' application interface. The title 'Orders' is at the top center. Below it is a search bar with a magnifying glass icon and a placeholder 'Search'. To the right of the search bar are four icons: a trash can (Delete), a pencil (Edit), a plus sign (Add), and a close button (X). A vertical scroll bar is on the right side of the table.

ID	Staff	Status
1	John Doe	Draft
2	Daniel Thompson	Pending
3	Alice Smith	Rejected

Annotations numbered 1 through 11 are placed around the interface:

- 1: Top-left corner of the main container.
- 2: Title 'Orders'.
- 3: Search bar icon.
- 4: Search bar placeholder.
- 5: Delete icon.
- 6: Edit icon.
- 7: Add icon.
- 8: Close icon.
- 9: Vertical scroll bar handle.
- 10: Bottom-right corner of the table.
- 11: Bottom-left corner of the main container.

This screenshot shows a proposed design for the 'Orders' application interface, similar to the final design but with color-coded status labels.

ID	Staff	Status
1	John Doe	Draft
2	Daniel Thompson	Pending
3	Alice Smith	Rejected

Annotations numbered 1 through 11 are placed around the interface:

- 1: Top-left corner of the main container.
- 2: Title 'Orders'.
- 3: Search bar icon.
- 4: Search bar placeholder.
- 5: Delete icon.
- 6: Add icon.
- 7: Edit icon.
- 8: Close icon.
- 9: Vertical scroll bar handle.
- 10: Bottom-right corner of the table.
- 11: Bottom-left corner of the main container.

Description

ID	Item	Walk Through	Styling
1	DisplayOrderView	The form for displaying orders.	BackColor: Background Size: 649, 500

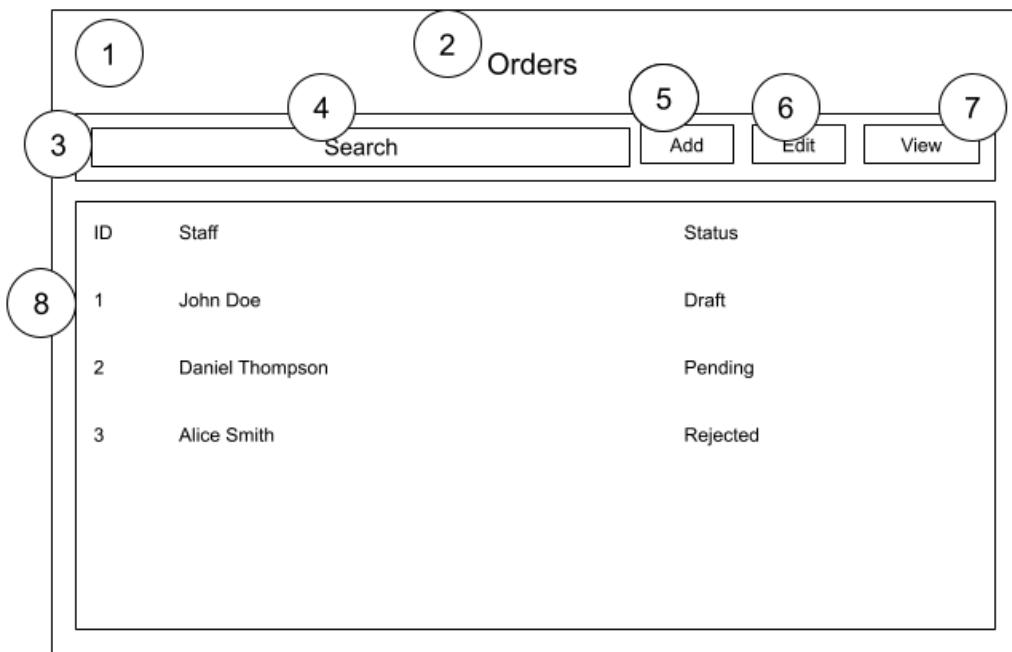
2	lblOrders	The title for the display order form.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23
3	pbSearch	A search icon to indicate the search box	BackColor: Background ForeColor: Foreground Size: 10, 10 Location: 12, 75
4	tbSearch	The text box into which the user inputs their search term to order the table by.	Forecolor: Foreground BorderColor: PrimaryForeground PlaceholderColor: Primary Size: 449, 40 Location: 12, 75 Anchor: Top FontSize: 12pt Placeholder: "Search"
5	pbClear	The picture box containing a cross icon. This can be clicked to clear the search bar.	BackColor: Background Location: 400, 5 Size 40, 40 Anchor: Top FontSize: 12pt Image: Plus
6	btnAdd	The button the staff member clicks if they want to add a new order.	Location: 400, 5 Size 100, 40 Anchor: Top FontSize: 12pt Image: Plus
7	btnEdit	The button the staff member clicks if they want to edit an existing drafted order.	Location: 500, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Image: Pencil
8	btnDelete	The button the staff member clicks if they want to delete a drafted order.	Location: 600, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Image: Bin
9	scrollBar	The scroll bar is linked to the data grid view. It is only visible if the	BackColor: Background ThumbColor:

		columns overflow the displayable size.	PrimaryForeground CornerRadii: 10, 10, 10, 10
10	dataGridView	The table in which the stock items are displayed. Column headers can be clicked to sort the corresponding column.	Location: 100, 100 Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor: Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold
11	pnlDataGridView	The panel that holds the data grid view to improve the user interface by adding a rounded border.	BackColor: Background BorderColor: PrimaryForeground Location: 163, 162 Size: 312, 270 CornerRadii: 10, 10, 10, 10

Approve Orders Page

This is the page where staff who work in the office can approve or reject orders submitted by the cleaning manager.

Initial Design



Description

ID	Item	Walk Through	Styling
1	ApproveOrder View	The form for displaying orders for approval.	BackColor: Background Size: 649, 500
2	lblOrders	The title for the display order form.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23
3	pnlTopBar	The panel that holds the search, add, edit, etc. controls.	BackColor: Background ForeColor: Foreground Size: 600, 50 Location: 10, 50
4	tbSearch	The text box into which the user inputs their search term to order the table by.	Forecolor: Foreground Size: 300, 40 Location: 5, 5

			Anchor: Top FontSize: 12pt Placeholder: "Search"
5	btnApprove	The button the staff member clicks if they want to approve a submitted order.	Location: 400, 5 Size 100, 40 Anchor: Top FontSize: 12pt Text: "Approve"
6	btnReject	The button the staff member clicks if they want to reject a submitted order.	Location: 500, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Text: "Reject"
7	btnView	The button the staff member clicks if they want to view an order.	Location: 600, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Text: "View"
8	dataGridView	The table in which the stock items are displayed. Column headers can be clicked to sort the corresponding column. This is scrollable.	Location: 100, 100 Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor: Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold

Feedback

The design could be improved by adding different colours to indicate the state of an order. Additionally, the use of icons would make the screen more user-friendly.

Final Design

This image shows the final design of the 'Orders' view. It features a dark-themed interface with white text and icons. At the top center is the title 'Orders' (2). To the left of the title are three numbered circles (1, 3, 4). To the right are five numbered circles (5, 6, 7, 8) followed by four small action buttons: a red 'X', a green checkmark, another red 'X', and a circular refresh icon. Below the title is a search bar with a magnifying glass icon and the word 'Search' (3). The main content area contains a table with columns 'ID', 'Staff', and 'Status'. The data rows are: ID 1, Staff John Doe, Status Delivered; ID 2, Staff Daniel Thompson, Status Pending; and ID 3, Staff Alice Smith, Status Rejected. A vertical scroll bar is visible on the right side of the table. At the bottom left is a large numbered circle (11), and at the bottom center is a smaller numbered circle (10).

ID	Staff	Status
1	John Doe	Delivered
2	Daniel Thompson	Pending
3	Alice Smith	Rejected

This image shows a design mockup for the 'Orders' view, similar in structure to the final design but with different styling. The background is white, and the text is in a light gray or black font. The title 'Orders' (2) is at the top center. To the left of the title are three numbered circles (1, 3, 4). To the right are five numbered circles (5, 6, 7, 8) followed by four small action buttons: a red 'X', a green checkmark, another red 'X', and a circular refresh icon. Below the title is a search bar with a magnifying glass icon and the word 'Search' (3). The main content area contains a table with columns 'ID', 'Staff', and 'Status'. The data rows are: ID 1, Staff John Doe, Status Draft; ID 2, Staff Daniel Thompson, Status Pending; and ID 3, Staff Alice Smith, Status Rejected. A vertical scroll bar is visible on the right side of the table. At the bottom left is a large numbered circle (11), and at the bottom center is a smaller numbered circle (10).

ID	Staff	Status
1	John Doe	Draft
2	Daniel Thompson	Pending
3	Alice Smith	Rejected

Description

ID	Item	Walk Through	Styling
1	ApproveOrder View	The form for displaying orders for approval.	BackColor: Background Size: 649, 500

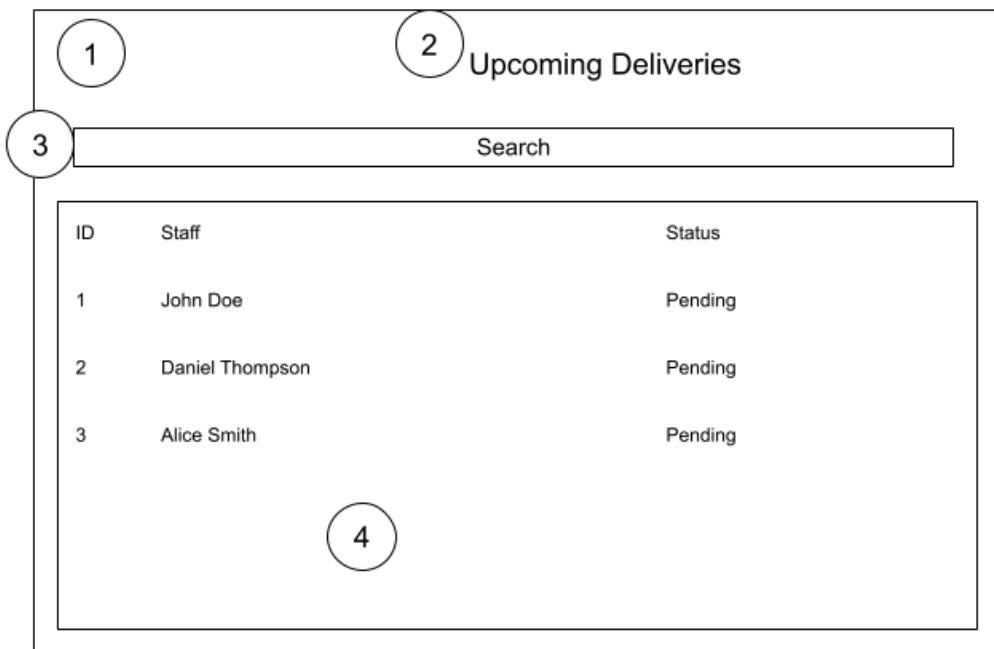
2	lblOrders	The title for the display order form.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23
3	pbSearch	A search icon to indicate the search box	BackColor: Background ForeColor: Foreground Size: 10, 10 Location: 12, 75
4	tbSearch	The text box into which the user inputs their search term to order the table by.	Forecolor: Foreground BorderColor: PrimaryForeground PlaceholderColor: Primary Size: 449, 40 Location: 12, 75 Anchor: Top FontSize: 12pt Placeholder: "Search"
5	pbClear	The picture box containing a cross icon. This can be clicked to clear the search bar.	BackColor: Background Location: 400, 5 Size 40, 40 Anchor: Top FontSize: 12pt Image: Plus
6	btnApprove	The button the staff member clicks if they want to approve a submitted order.	Location: 400, 5 Size 100, 40 Anchor: Top FontSize: 12pt Image: Tick
7	btnReject	The button the staff member clicks if they want to reject a submitted order.	Location: 500, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Image: Cross
8	btnView	The button the staff member clicks if they want to view an order.	Location: 600, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Image: Eye
9	scrollBar	The scroll bar is linked to the data grid view. It is only visible if the	BackColor: Background ThumbColor:

		columns overflow the displayable size.	PrimaryForeground CornerRadii: 10, 10, 10, 10
10	dataGridView	The table in which the stock items are displayed. Column headers can be clicked to sort the corresponding column.	Location: 100, 100 Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor: Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold
11	pnlDataGridView	The panel that holds the data grid view to improve the user interface by adding a rounded border.	BackColor: Background BorderColor: PrimaryForeground Location: 163, 162 Size: 312, 270 CornerRadii: 10, 10, 10, 10

Upcoming Deliveries Page

This is the page where the cleaning manager can view upcoming pending orders (deliveries).

Initial Design



Description

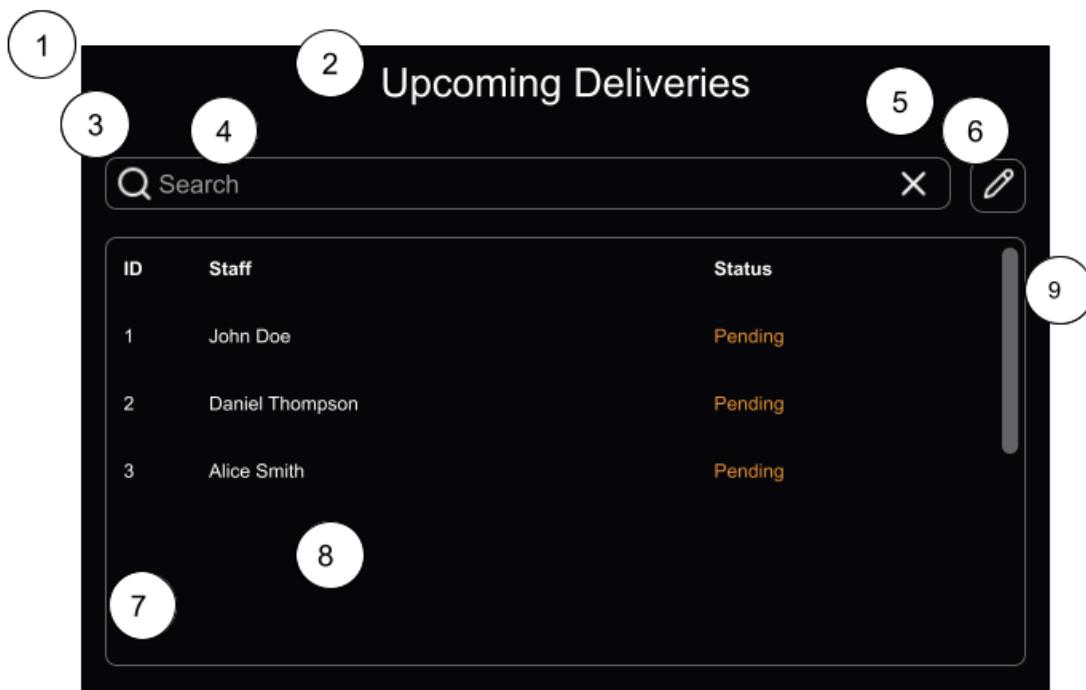
ID	Item	Walk Through	Styling
1	UpcomingDeliveriesView	The form for displaying upcoming deliveries.	BackColor: Background Size: 649, 500
2	lblUpcomingDeliveries	The title label for the upcoming deliveries page.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23
3	tbSearch	The text box into which the user inputs their search term to order the table by.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt Placeholder: "Search"
4	dataGridView	This is the table in which the	Location: 100, 100

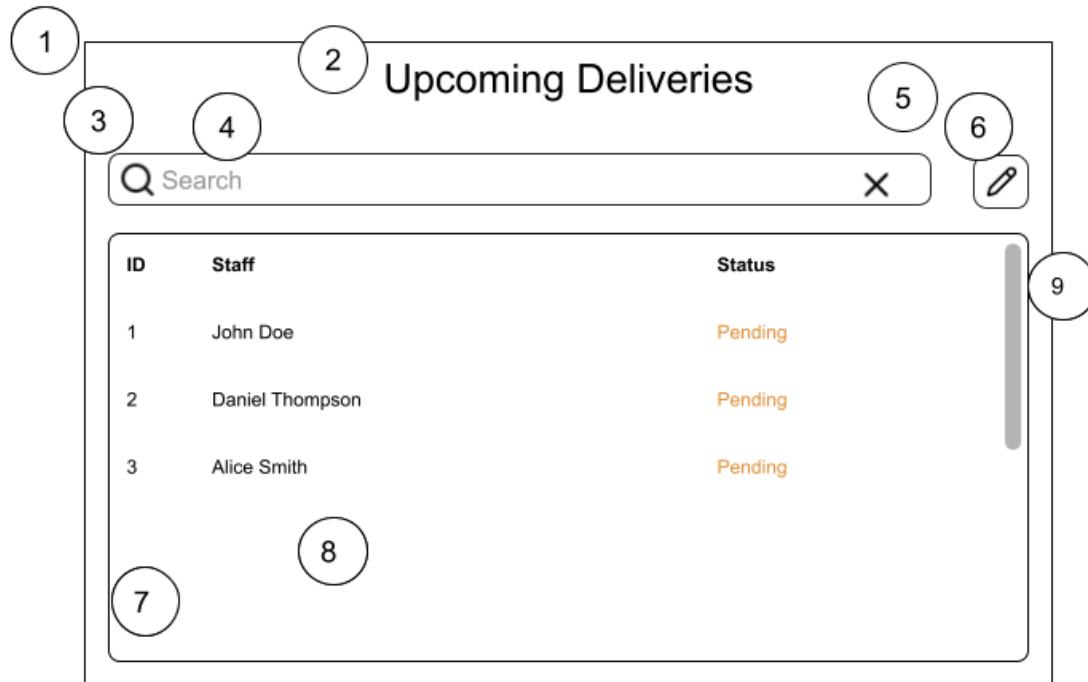
		<p>pending orders are displayed. Column headers can be clicked to sort the corresponding column. It is scrollable.</p>	Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor: Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold
--	--	--	--

Feedback

It would be useful to have a way to directly edit upcoming deliveries from this page.

Final Design





Description

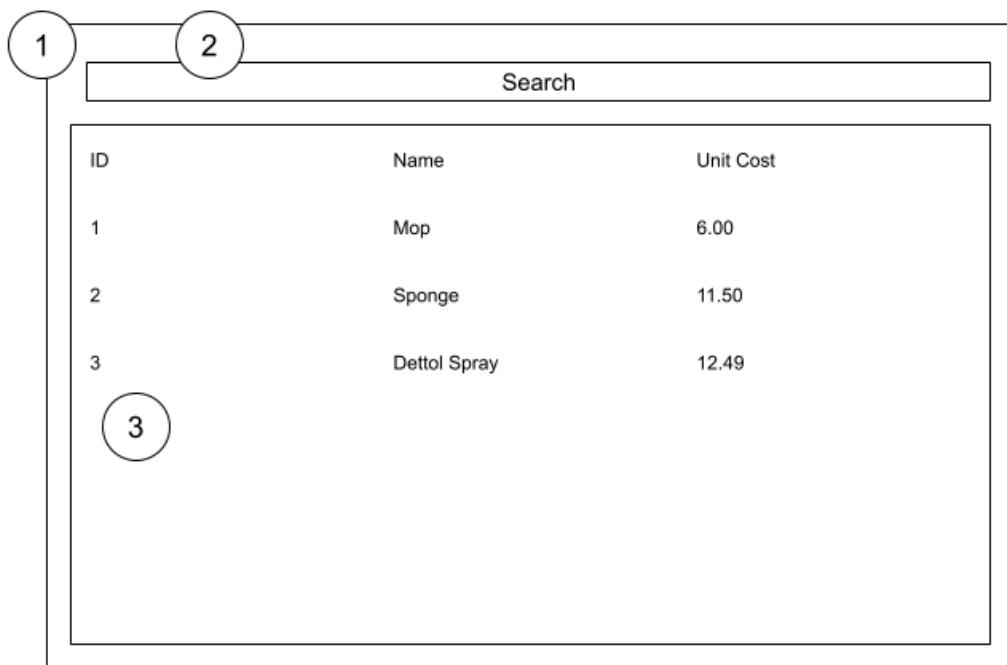
ID	Item	Walk Through	Styling
1	UpcomingDeliv eriesView	The form for displaying upcoming deliveries.	BackColor: Background Size: 649, 500
2	lblUpcomingDe liveries	The title label for the upcoming deliveries page.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23
3	pbSearch	A search icon to indicate the search box.	BackColor: Background ForeColor: Foreground Size: 10, 10 Location: 12, 75
4	tbSearch	The text box into which the user inputs their search term to order the table by.	ForeColor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt Placeholder: "Search"
5	pbClear	The picture box containing a cross icon. This can be clicked to clear the	BackColor: Background Location: 400, 5

		search bar.	Size 40, 40 Anchor: Top FontSize: 12pt Image: Plus
6	btnEdit	The button the staff member clicks if they want to view a stock quantity change.	BackColor: Background ForeColor: SecondaryForeground HoverColor: SecondaryForeground Location: 400, 5 Size 40, 40 Anchor: Top FontSize: 12pt Image: Eye
7	dataGridView	The table in which the pending orders are displayed. Column headers can be clicked to sort the corresponding column. This is scrollable.	Location: 100, 100 Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor: Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold
8	pnlDataGridView	The panel that holds the data grid view to improve the user interface by adding a rounded border.	BackColor: Background BorderColor: PrimaryForeground Location: 163, 162 Size: 312, 270 CornerRadii: 10, 10, 10, 10
9	scrollBar	The scroll bar is linked to the data grid view. It is only visible if the columns overflow the displayable size.	BackColor: Background ThumbColor: PrimaryForeground CornerRadii: 10, 10, 10, 10

Select Order Stock & Select Cleaning Job Customer & Select Cleaning Job Option Page

This page will be used to select stock for an order when creating an order. It will also be used to select a customer for a cleaning job and cleaning job options for a cleaning job.

Initial Design

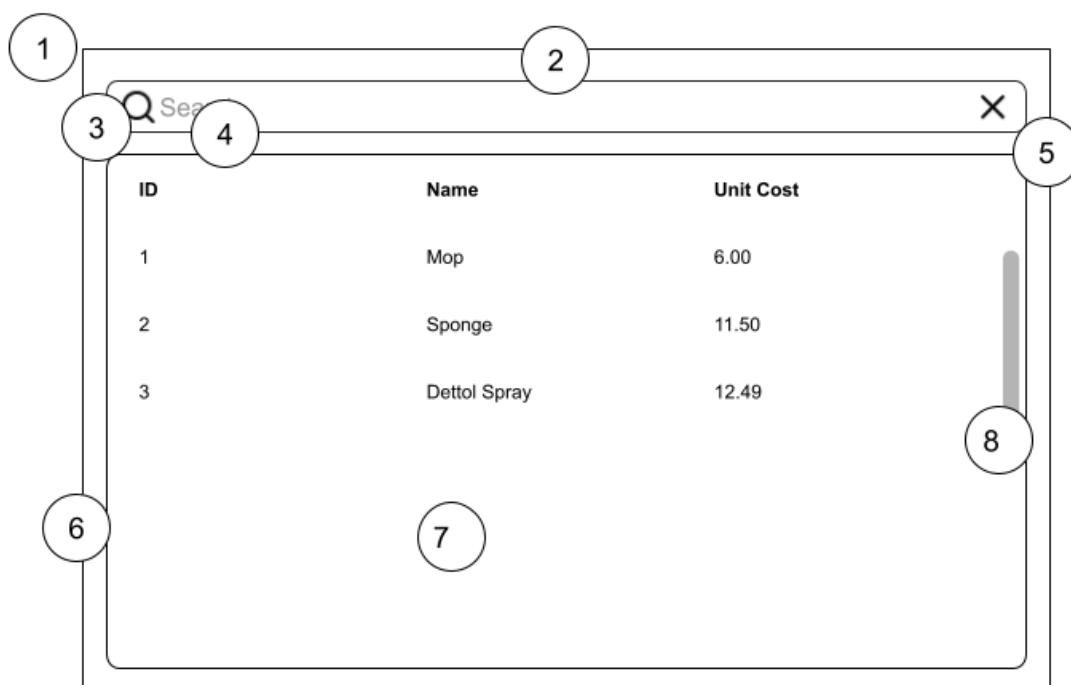
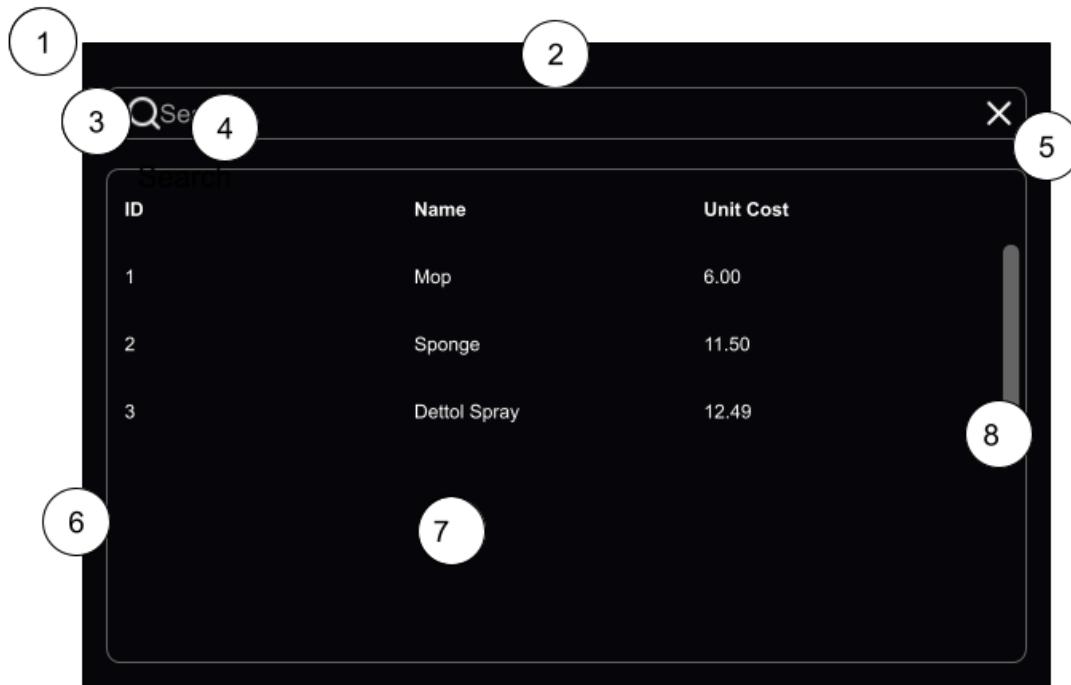


Description

ID	Item	Walk Through	Styling
1	SelectView	The form for displaying selecting items.	BackColor: Background Size: 649, 500
2	tbSearch	The text box into which the user inputs their search term to order the table by.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt Placeholder: "Search"
3	dataGridView	The table in which the selectable items are displayed. Column headers can be clicked to sort the corresponding column. This is scrollable.	Location: 100, 100 Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground

			SelectedForeColor: Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold
--	--	--	---

Final Design



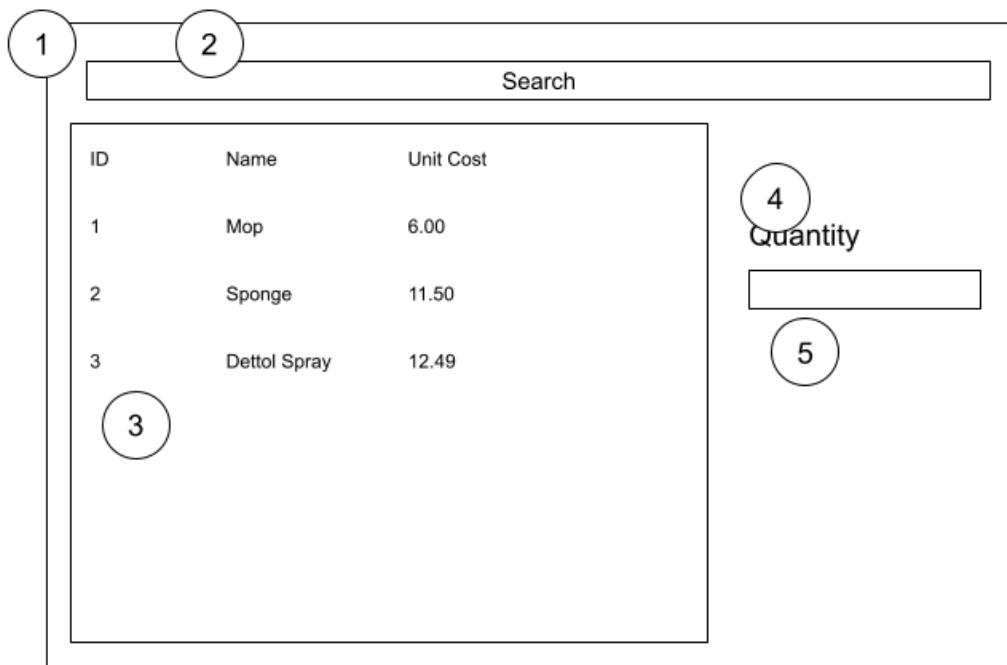
Description

ID	Item	Walk Through	Styling
1	UpcomingDeliveriesView	The form for displaying upcoming deliveries.	BackColor: Background Size: 649, 500
2	lblUpcomingDeliveries	The title label for the upcoming deliveries page.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23
3	pbSearch	A search icon to indicate the search box.	BackColor: Background ForeColor: Foreground Size: 10, 10 Location: 12, 75
4	tbSearch	The text box into which the user inputs their search term to order the table by.	ForeColor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt Placeholder: "Search"
5	pbClear	The picture box containing a cross icon. This can be clicked to clear the search bar.	BackColor: Background Location: 400, 5 Size 40, 40 Anchor: Top FontSize: 12pt Image: Plus
6	dataGridView	The table in which the pending orders are displayed. Column headers can be clicked to sort the corresponding column. This is scrollable.	Location: 100, 100 Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor: Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold
7	pnlDataGridView	The panel that holds the data grid view to improve the user interface by adding a rounded border.	BackColor: Background BorderColor: PrimaryForeground

			Location: 163, 162 Size: 312, 270 CornerRadii: 10, 10, 10, 10
8	scrollBar	The scroll bar is linked to the data grid view. It is only visible if the columns overflow the displayable size.	BackColor: Background ThumbColor: PrimaryForeground CornerRadii: 10, 10, 10, 10

Select Order Stock Quantities Page & Select Cleaning Job Cleaning Options Page
 This page will be used to set the quantity of each stock item to be ordered.
 Alternatively, a similar page will be used to select cleaning job options for a cleaning job.

Initial Design



Description

ID	Item	Walk Through	Styling
1	SelectQuantity View	The form for displaying selecting a quantity for a list of items.	BackColor: Background Size: 649, 500
2	tbSearch	The text box into which the user inputs their search term to order the table by.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt Placeholder: "Search"
3	dataGridView	The table in which the selectable items are displayed. Column headers can be clicked to sort the corresponding column. This is scrollable.	Location: 100, 100 Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor:

			Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold
4	lblQuantity	A title label indicating the quantity.	FontSize: 14pt ForeColor: Foreground Location: 200, 200 Size: 200, 40
5	tbQuantity	A text box for the currently selected item quantity.	BackColor: Background Location: 202, 240 Size: 250, 41

Feedback

This page could be improved by providing more detailed information about the selected item, such as the description, name, and subtotal. A numeric control to input the quantity instead of simply a text box would also be useful. Finally, a total cost should be displayed in the bottom right corner.

Final Design

1

ID	Name	Unit Cost
1	Mop	6.00
2	Sponge	11.50
3	Dettol Spray	12.49

2 Search 3 X 8 Name 9
10 Description 11
12 13 Quantity + 1 14
15 Subtotal: 16
17 Total: 18

1

ID	Name	Unit Cost
1	Mop	6.00
2	Sponge	11.50
3	Dettol Spray	12.49

2 Search 3 X 8 Name 9
10 Description 11
12 13 Quantity + 1 14
15 Subtotal: 16
17 Total: 18

Description

ID	Item	Walk Through	Styling
1	SelectQuantity View	The form for displaying selecting a quantity for a list of items.	BackColor: Background Size: 649, 500

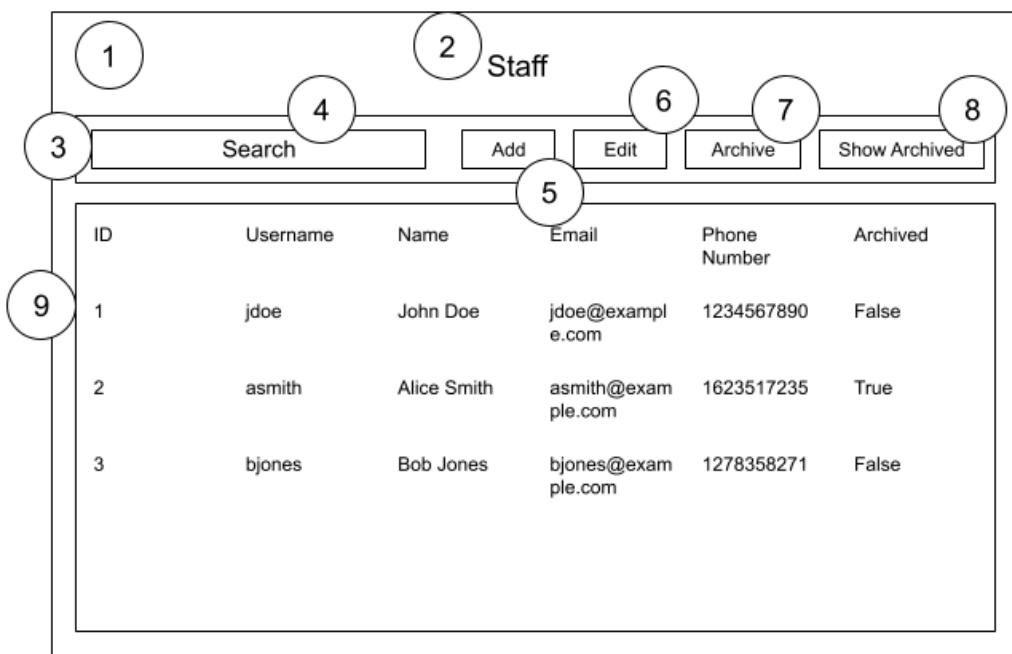
2	pbSearch	A search icon to indicate the search box	BackColor: Background ForeColor: Foreground Size: 10, 10 Location: 12, 75
3	tbSearch	The text box into which the user inputs their search term to order the table by.	Forecolor: Foreground BorderColor: PrimaryForeground PlaceholderColor: Primary Size: 449, 40 Location: 12, 75 Anchor: Top FontSize: 12pt Placeholder: "Search"
4	pbClear	The picture box contains a cross icon. This can be clicked to clear the search bar.	BackColor: Background Location: 400, 5 Size 40, 40 Anchor: Top FontSize: 12pt Image: Plus
5	dataGridView	The table in which the selectable items are displayed. Column headers can be clicked to sort the corresponding column. This is scrollable.	Location: 100, 100 Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor: Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold
6	pnlDataGridView	The panel that holds the data grid view to improve the user interface by adding a rounded border.	BackColor: Background BorderColor: PrimaryForeground Location: 163, 162 Size: 312, 270 CornerRadii: 10, 10, 10, 10
7	scrollBar	The scroll bar is linked to the data grid view. It is only visible if the columns overflow the displayable size.	BackColor: Background ThumbColor: PrimaryForeground CornerRadii: 10, 10, 10, 10

8	pnlInfo	The panel that holds the controls provides the information for the selected item.	BackColor: Background BorderColor: PrimaryForeground Location: 163, 162 Size: 312, 270 CornerRadii: 10, 10, 10, 10
4	lblQuantity	A title label indicating the quantity.	FontSize: 14pt ForeColor: Foreground Location: 200, 200 Size: 200, 40
5	nudQuantity	A numeric up-down for the currently selected item quantity.	BackColor: Background Location: 202, 240 Size: 250, 41

Manage Staff Page

The staff display page is where the administrator can view all staff within the application and edit their details.

Initial Design



Description

ID	Item	Walk Through	Styling
1	DisplayStaffView	The form for displaying stock.	BackColor: Background Size: 649, 500
2	lblStaff	The title for the display staff form.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23
3	pnlTopBar	The panel that holds the search, add, edit, etc. controls.	BackColor: Background ForeColor: Foreground Size: 600, 50 Location: 10, 50
4	tbSearch	The text box into which the user inputs their search term to order the table by.	ForeColor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt

			Placeholder: "Search"
5	btnAdd	The button the staff member clicks if they want to add a new staff account.	Location: 400, 5 Size 100, 40 Anchor: Top FontSize: 12pt Text: "Add"
6	btnEdit	The button the staff member clicks if they want to edit an existing staff account.	Location: 500, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Text: "Edit"
7	btnArchive	The button the staff member clicks if they want to archive an existing staff account.	Location: 600, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Text: "Archive"
8	btnShowArchived	The button the staff member clicks if they want to show the archived staff accounts.	Location: 650, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Text: "Show Archived"
9	dataGridView	The table in which the staff accounts are displayed. Column headers can be clicked to sort the corresponding column. This is scrollable.	Location: 100, 100 Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor: Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold

Feedback

The initial design is good but could be improved in several aspects. Firstly, icons could be more concise and intuitive than text but could be supplemented by tooltips. In addition to this, a way to clear the search bar could be useful. Secondly, a scroll bar would be useful for users at the side of the data grid view. Finally, the formatting of the data grid view could be improved. The addition of different colours for different levels of stock could improve the ability of staff members to see the stock level at a glance. It would also make more sense to use "yes" and "no" for archive status to improve readability. In addition to this, it would also be useful to be able to edit a staff account by double-clicking the row.

Final Design

This screenshot shows the final design of a staff list view. The interface has a dark header bar with the title "Staff". Below the header is a search bar containing a magnifying glass icon and the word "Search". To the right of the search bar are several icons: a white circle with a black "X", a white circle with a black plus sign, a pencil icon, a trash bin icon, a cube icon, and a question mark icon. A vertical scroll bar is visible on the right side of the list area.

ID	Username	Name	Email	Phone Number	Archived
1	jdoe	John Doe	jdoe@example.com	1234567890	No
2	asmith	Alice Smith	asmith@example.com	1623517235	Yes
3	bjones	Bob Jones	bjones@example.com	1278358271	No

Callouts numbered 1 through 12 are overlaid on the interface:

- 1: Top-left corner of the header bar.
- 2: Header bar text "Staff".
- 3: Search bar icon.
- 4: Search bar text "Search".
- 5: White circle with black "X" icon.
- 6: White circle with black plus sign icon.
- 7: Pencil icon.
- 8: Trash bin icon.
- 9: Cube icon.
- 10: Vertical scroll bar handle.
- 11: Bottom center of the list area.
- 12: Bottom-left corner of the list area.

This screenshot shows the initial design of the same staff list view. The layout and data are identical to the final design. Callouts numbered 1 through 12 are overlaid on the interface:

- 1: Top-left corner of the header bar.
- 2: Header bar text "Staff".
- 3: Search bar icon.
- 4: Search bar text "Search".
- 5: White circle with black "X" icon.
- 6: White circle with black plus sign icon.
- 7: Pencil icon.
- 8: Trash bin icon.
- 9: Cube icon.
- 10: Vertical scroll bar handle.
- 11: Bottom center of the list area.
- 12: Bottom-left corner of the list area.

Description

ID	Item	Walk Through	Styling
1	DisplayStaffView	The form for displaying stock.	BackColor: Background Size: 649, 500

2	lblStaff	The title for the display staff form.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23 Text: "Staff"
3	pbSearch	A search icon to indicate the search box	BackColor: Background ForeColor: Foreground Size: 10, 10 Location: 12, 75
4	tbSearch	The text box into which the user inputs their search term to order the table by.	Forecolor: Foreground BorderColor: PrimaryForeground PlaceholderColor: Primary Size: 449, 40 Location: 12, 75 Anchor: Top FontSize: 12pt Placeholder: "Search"
5	pbClear	The picture box containing a cross icon. This can be clicked to clear the search bar.	BackColor: Background Location: 400, 5 Size 40, 40 Anchor: Top FontSize: 12pt Image: Plus
6	btnAdd	The button the staff member clicks if they want to add a new stock item.	BackColor: Background ForeColor: SecondaryForeground HoverColor: SecondaryForeground Location: 400, 5 Size 40, 40 Anchor: Top FontSize: 12pt Image: Plus
7	btnEdit	The button the staff member clicks if they want to edit an existing stock item.	BackColor: Background ForeColor: SecondaryForeground HoverColor: SecondaryForeground Location: 500, 5 Size: 40, 40 Anchor: Top

			FontSize: 12pt Image: Pencil
8	btnArchive	The button the staff member clicks if they want to archive a stock item.	BackColor: Background ForeColor: SecondaryForeground HoverColor: SecondaryForeground Location: 600, 5 Size: 40, 40 Anchor: Top FontSize: 12pt Image: Archive
9	btnShowArchived	The button the staff member clicks if they want to show the archived stock.	BackColor: Background ForeColor: SecondaryForeground HoverColor: SecondaryForeground Location: 650, 5 Size: 40, 40 Anchor: Top FontSize: 12pt Image: Box
10	scrollBar	The scroll bar linked to the data grid view. It is only visible if the columns overflow the displayable size	BackColor: Background ThumbColor: PrimaryForeground CornerRadii: 10, 10, 10, 10
11	dataGridView	The table in which the stock items are displayed. Column headers can be clicked to sort the corresponding column.	Location: 100, 100 Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor: Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold
12	pnlDataGridView	The panel that holds the data grid view to improve the user interface by adding a rounded border.	BackColor: Background BorderColor: PrimaryForeground Location: 163, 162 Size: 312, 270 CornerRadii: 10, 10, 10,

Candidate number: 8346

Centre Number: 71571

			10
--	--	--	----

Staff Personal Details & Customer Personal Details Page

This page will be used by staff members to update their personal details. It will also be used by customers so that office staff can set their personal details. This is similar to the emergency contact page, but with a date of birth instead of a phone number.

Initial Design

Description

ID	Item	Walk Through	Styling
1	PersonalDetailsView	The form for managing personal details.	BackColor: Background Size: 654, 419
2	lblForename	A label indicating the forename.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
3	tbForename	A text box for setting the forename.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top

			FontSize: 12pt
4	lblSurname	A label indicating the surname.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
5	tbSurname	A text box for setting the surname.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt
6	lblDateOfBirth	A label indicating the date of birth.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
7	tbDateOfBirth	A text box for setting the date of birth.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt

Feedback

This could be improved by using a dedicated date input instead of a textbox.

Final Design

A screenshot of a mobile application interface showing a form for personal information. The background is black. The form fields are white with black text. Circular numbered callouts point to specific elements:

- 1: A small circle at the top left.
- 2: Points to the "Forename" label and the input field containing "John".
- 3: A small circle at the top right.
- 4: Points to the "Surname" label and the input field containing "Doe".
- 5: A small circle on the right side.
- 6: Points to the "Date Of Birth (optional)" label and the "Day Month Year" placeholder text.
- 7: Points to the three separate input fields for Day (23), Month (06), and Year (1968).

A screenshot of a mobile application interface showing a form for personal information. The background is white. The form fields are white with black text. Circular numbered callouts point to specific elements:

- 1: A small circle at the top left.
- 2: Points to the "Forename" label and the input field containing "John".
- 3: A small circle at the top right.
- 4: Points to the "Surname" label and the input field containing "Doe".
- 5: A small circle on the right side.
- 6: Points to the "Date Of Birth (optional)" label and the "Day Month Year" placeholder text.
- 7: Points to the three separate input fields for Day (23), Month (06), and Year (1968).

Description

ID	Item	Walk Through	Styling
----	------	--------------	---------

1	PersonalDetailsView	The form for managing personal details.	BackColor: Background Size: 654, 419
2	lblForename	A label indicating the forename.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
3	tbForename	A text box for setting the forename.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt
4	lblSurname	A label indicating the surname.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
5	tbSurname	A text box for setting the surname.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt
6	lblDateOfBirth	A label indicating the date of birth.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
7	diDateOfBirth	A date input for setting the date of birth.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt

Staff Contact Details & Customer Contact Details Page

This is the page where the staff member can edit their own contact details or those of a customer.

Initial Design



Description

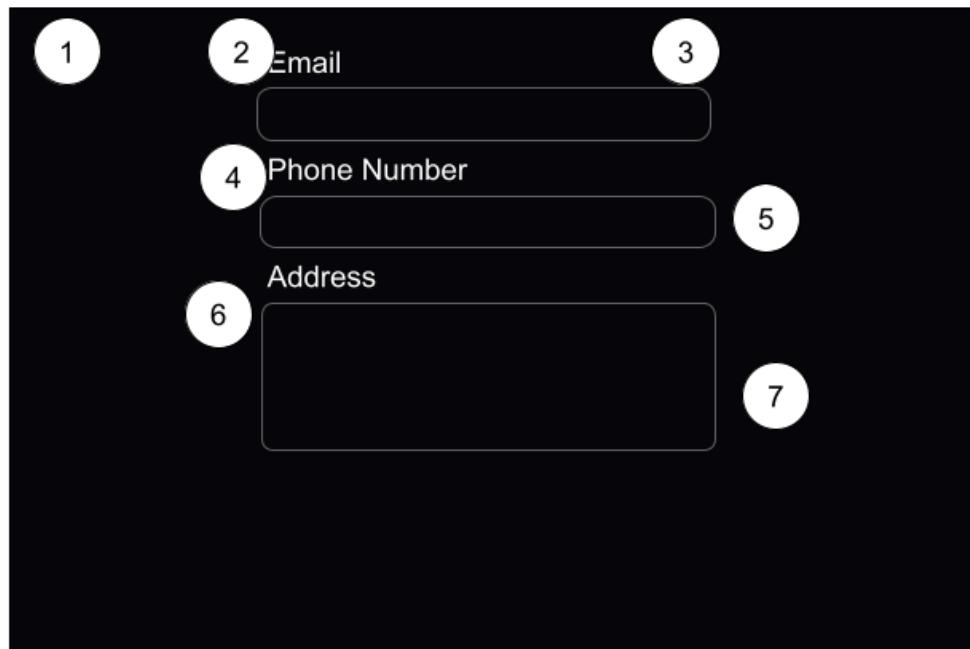
ID	Item	Walk Through	Styling
1	PersonalDetailsView	The form for managing personal details.	BackColor: Background Size: 654, 419
2	lblEmail	A label indicating the email.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
3	tbEmail	A text box for setting the email.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt

4	lblPhoneNumber	A label indicating the phone number.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
5	tbPhoneNumber	A text box for setting the phone number.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt
6	lblAddress	A label indicating the address.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
7	tbAddress	A text box for setting the address.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt

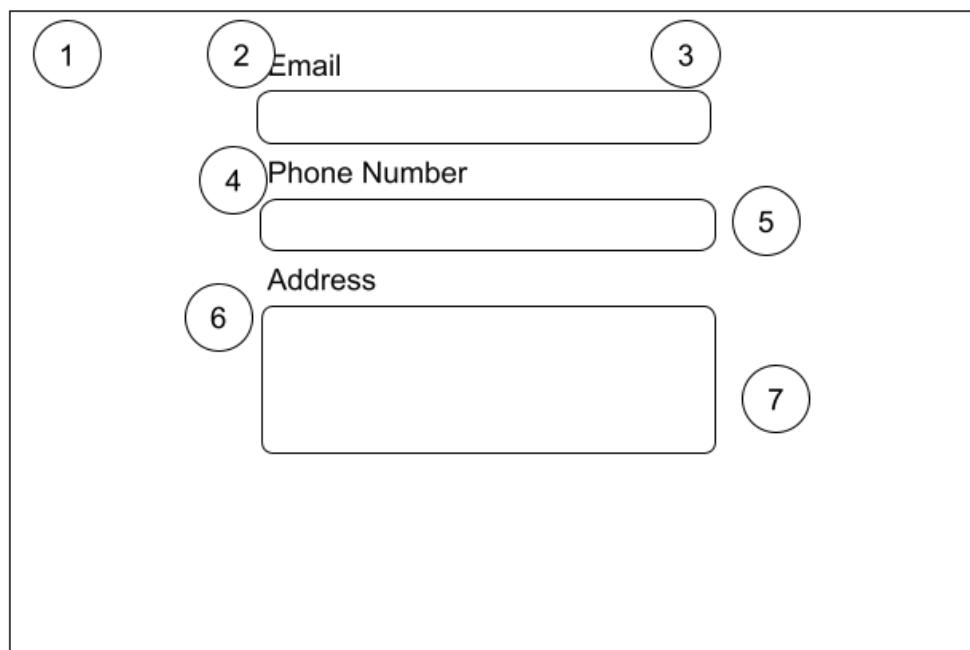
Feedback

This page is good. However, it could be improved by making the address section larger to make multiline addresses more readable.

Final Design



A form design with a black background. It features three input fields: a short one labeled 'Email' (2), a medium one labeled 'Phone Number' (4), and a large one labeled 'Address' (6). Each label is preceded by a numbered circle (1, 2, 3, 4, 5, 6, 7) and followed by a horizontal line for input.



A form design with a white background, identical in structure to the 'Final Design' version above it. It includes three input fields labeled 'Email' (2), 'Phone Number' (4), and 'Address' (6), each preceded by a numbered circle (1, 2, 3, 4, 5, 6, 7) and a corresponding input line.

Description

ID	Item	Walk Through	Styling
----	------	--------------	---------

1	PersonalDetailsView	The form for managing personal details.	BackColor: Background Size: 654, 419
2	lblEmail	A label indicating the email.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
3	tbEmail	A text box for setting the email.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt
4	lblPhoneNumber	A label indicating the phone number.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
5	tbPhoneNumber	A text box for setting the phone number.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt
6	lblAddress	A label indicating the address.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
7	tbAddress	A text box for setting the address.	Forecolor: Foreground Size: 300, 100 Location: 5, 5 Anchor: Top FontSize: 12pt

Staff Account Security Page

This page allows staff members to update their password. This page will also be used by administrators to allow them to change passwords for any staff account. When it is used with the staff settings page, the username and privilege level will be displayed in read-only text boxes. When it is used by an administrator to change a staff account password, a username input box will be visible, and there will be no current password input.

Initial Design

1 2 Current Password
 3
 4 New Password
 5
 6 Requirements:
 - Eight characters long
 - Numbers
 - Uppercase and lowercase
 - Special character
 7 Change Password

Description

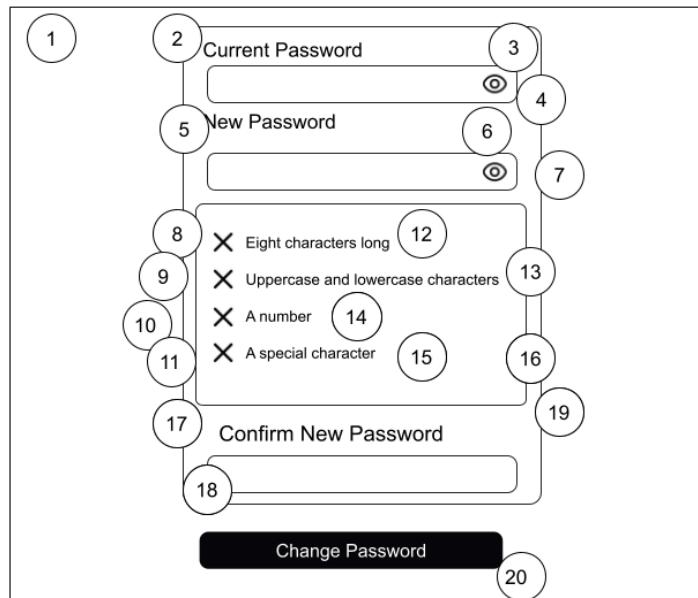
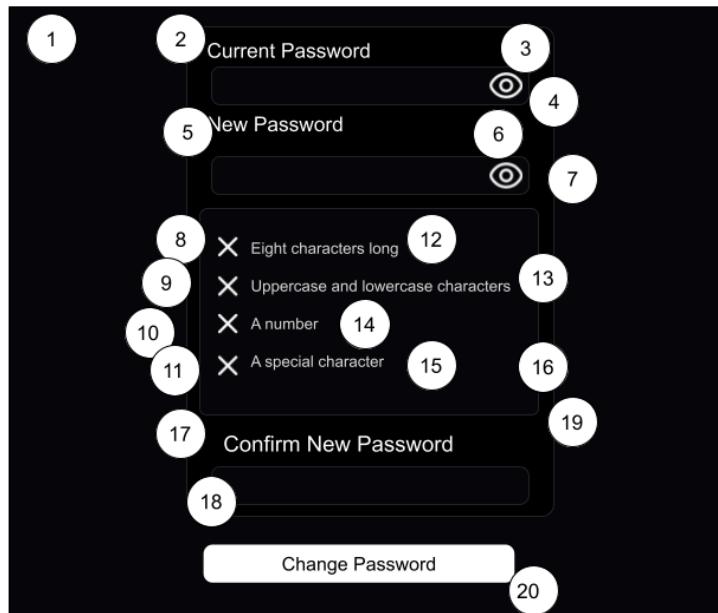
ID	Item	Walk Through	Styling
1	SecurityView	The form for managing personal details.	BackColor: Background Size: 654, 419
2	lblCurrentPassword	A label indicating the current password input.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
3	tbCurrentPassword	A text box for inputting the current password.	Forecolor: Foreground Size: 300, 40 Location: 5, 5

			Anchor: Top FontSize: 12pt UsePasswordCharacter: True
4	lblNewPassword	A label indicating the new password input.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
5	tbNewPassword	A text box for inputting the new password.	ForeColor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt
6	lblRequirements	A label indicating password requirements.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
7	btnChangePassword	A button to change the password (if it is valid).	BackColor: Foreground ForeColor: Background HoverColor: SecondaryForeground FontSize: 12pt Size: 252, 36 Location: 30, 200 CornerRadii: 10, 10, 10, 10 Text: "Change Password"

Feedback

This page could be improved by adding a confirm password box to help reduce the chance of accidentally typing the new password in incorrectly. Furthermore, it would be helpful for users to be able to turn off the password character to view their new password and current password. Finally, an interactive way to show if the user requirements have been fulfilled would allow for more intuitive password creation.

Final Design



Description

ID	Item	Walk Through	Styling
----	------	--------------	---------

1	SecurityView	The form for managing personal details.	BackColor: Background Size: 654, 419
2	lblCurrentPassword	A label indicating the current password input.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
3	tbCurrentPassword	A text box for inputting the current password.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt UsePasswordCharacter: True
4	pbShowCurrentPassword	A clickable picture box containing an icon to determine if the current password is shown.	BackColor: Background Size: 21, 21 Location: 278, 177
5	lblNewPassword	A label indicating the new password input.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
6	tbNewPassword	A text box for inputting the new password.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt
7	pbShowNewPassword	A clickable picture box containing an icon (cross vs tick) to determine corresponding password requirement has been met.	BackColor: Background Size: 21, 21 Location: 278, 177
8	pbLength	A clickable picture box containing an icon (cross vs tick) to determine corresponding password requirement has been met.	BackColor: Background Size: 21, 21 Location: 30, 30

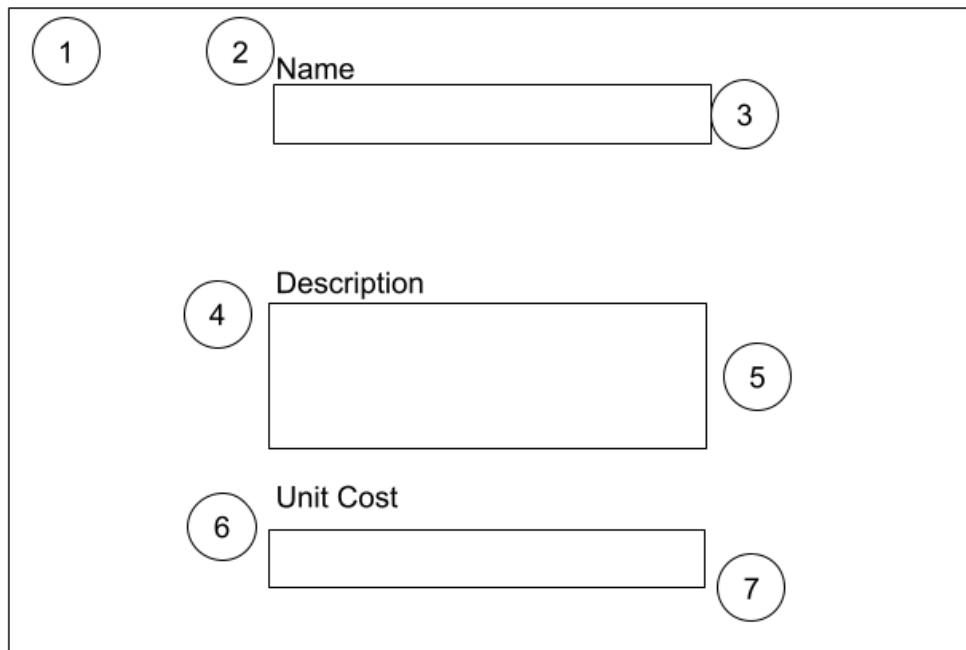
9	pbCase	A clickable picture box containing an icon (cross vs tick) to determine corresponding password requirement has been met.	BackColor: Background Size: 21, 21 Location: 30, 60
10	pbNumber	A clickable picture box containing an icon (cross vs tick) to determine corresponding password requirement has been met.	BackColor: Background Size: 21, 21 Location: 30, 90
11	pbSpecial	A clickable picture box containing an icon (cross vs tick) to determine corresponding password requirement has been met.	BackColor: Background Size: 21, 21 Location: 30, 120
12	lblLength	A label displaying the length requirement.	FontSize: 10pt ForeColor: Primary Location: 202, 209 Size: 176, 23
13	lblCase	A label displaying the uppercase and lowercase requirement.	FontSize: 10pt ForeColor: Primary Location: 202, 209 Size: 176, 23
14	lblNumber	A label displaying the number requirement.	FontSize: 10pt ForeColor: Primary Location: 202, 209 Size: 176, 23
15	lblSpecial	A label displaying the special character requirement.	FontSize: 10pt ForeColor: Primary Location: 202, 209 Size: 176, 23
16	pnlRequirements	A panel holding the requirements controls.	Location 246, 44 Size: 300, 200 CornerRadii: 10, 10, 10, 10

			BorderColor: PrimaryForeground
17	lblConfirmPassword	A label indicating the confirm password input.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
18	tbConfirmPassword	A text box for inputting the password for confirmation.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt UsePasswordCharacter: True
19	pnl	The panel that holds all the controls.	Location 146, 44 Size: 400, 500 CornerRadii: 10, 10, 10, 10 BorderColor: PrimaryForeground
20	btnChangePassword	A button to change the password (if it is valid).	BackColor: Foreground ForeColor: Background HoverColor: SecondaryForeground FontSize: 12pt Size: 252, 36 Location: 30, 200 CornerRadii: 10, 10, 10, 10 Text: "Change Password"

Cleaning Job Option Details Page

The page that displays the data associated with a cleaning job option for editing, viewing or adding.

Initial Design



Description

ID	Item	Walk Through	Styling
1	CleaningJobOptionDetailView	The form for viewing and editing cleaning job option details.	BackColor: Background Size: 654, 419
2	lblName	Label for the cleaning job option name input.	FontSize: 14pt ForeColor: Foreground Location: 135, 5 AutoSize: True
3	tbName	Text box for entering the cleaning job option name.	BackColor: Background BorderColor: PrimaryForeground Size: 355, 40

			Placeholder: "Dettol Spray"
4	lblDescription	Label for the cleaning job option description input.	FontSize: 14pt ForeColor: White Location: 135, 224 AutoSize: True
5	tbDescription	Multi-line textbox for cleaning job option description.	BackColor: Background, BorderColor: PrimaryForeground, Size: 355, 83 Placeholder: "Dettol antibacterial surface cleaning spray..."
6	lblUnitCost	Label for the cleaning job option unit cost input.	FontSize: 14pt ForeColor: White Location: 135, 224 AutoSize: True
7	tbUnitCost	A text box for the staff member to type the unit cost.	BackColor: Background, BorderColor: PrimaryForeground Size: 355, 83

Feedback

This could be improved by adding a numeric-up down instead of a text box for the unit cost. Additionally, it would be useful to indicate the maximum size for the description.

Final Design

A form with the following fields and controls:

- 1: A small circle at the top left.
- 2: "Stock Name" label with a text input field below it.
- 3: A small circle at the end of the Stock Name input field.
- 4: A small circle below the Stock Name label.
- 5: A small circle at the end of the Description input field.
- 6: A small circle below the Description input field.
- 7: "22/500" label with a small circle at its end.
- 8: A small circle at the top left of the Unit Cost input field.
- 9: A small circle at the end of the Unit Cost input field.
- 10: A small circle at the right end of the Unit Cost input field.

A form with the same layout as the final design, but with a light gray background:

- 1: A small circle at the top left.
- 2: "Stock Name" label with a text input field below it.
- 3: A small circle at the end of the Stock Name input field.
- 4: A small circle below the Stock Name label.
- 5: A small circle at the end of the Description input field.
- 6: A small circle below the Description input field.
- 7: "22/500" label with a small circle at its end.
- 8: A small circle at the top left of the Unit Cost input field.
- 9: A small circle at the end of the Unit Cost input field.
- 10: A small circle at the right end of the Unit Cost input field.

Description

ID	Item	Walk Through	Styling
----	------	--------------	---------

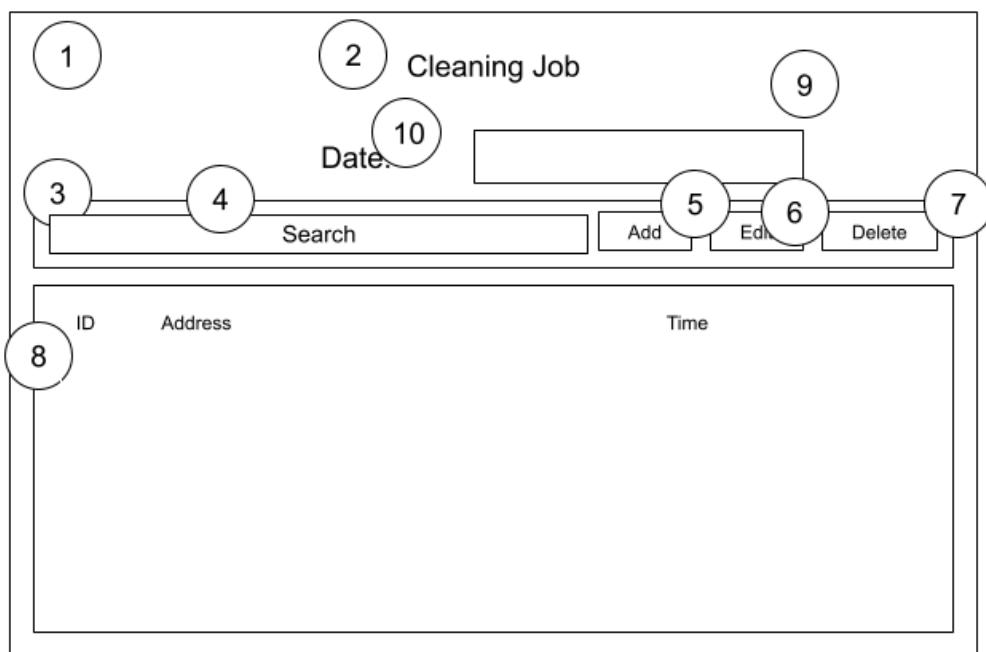
1	CleaningJobOptionDetailView	The form for viewing and editing cleaning job option details.	BackColor: Background Size: 654, 419
2	lblName	Label for the cleaning job option name input.	FontSize: 14pt ForeColor: Foreground Location: 135, 5 AutoSize: True
3	tbName	Text box for entering the cleaning job option name.	BackColor: Background BorderColor: PrimaryForeground Size: 355, 40 Placeholder: "Dettol Spray"
4	lblDescription	Label for the cleaning job option description input.	FontSize: 14pt ForeColor: White Location: 135, 224 AutoSize: True
5	tbDescription	Multi-line textbox for cleaning job option description.	BackColor: Background, BorderColor: PrimaryForeground, Size: 355, 83 Placeholder: "Dettol antibacterial surface cleaning spray..."
6	lblUnitCost	Label for the cleaning job option unit cost input.	FontSize: 14pt ForeColor: White Location: 135, 224 AutoSize: True
7	lblCharacterLimit	Label showing character limit for description.	Font: 12pt ForeColor: Foreground

			Size: 150, 24 Location: 340, 345 Text: "0/500"
8	pnlUnitCost	A panel that holds the controls.	Location 146, 44 Size: 355, 49 CornerRadii: 10, 10, 10, 10 BorderColor: PrimaryForeground
9	lblUnitCost	A title label for the cleaning job option unit cost.	Location: 5, 13 Size: 103, 23 AutoSize = true, FontSize = 14, ForeColor = Foreground
10	nudUnitCost	A numeric up-down control allows the staff member to view or type the unit cost.	Location: 116, 13 Size: 228, 27 BackColor = Background Maximum: 1000000 Minimum: 0

Book Cleaning Job Page

This is the page where a cleaning job booking will be made by office staff.

Initial Design



Description

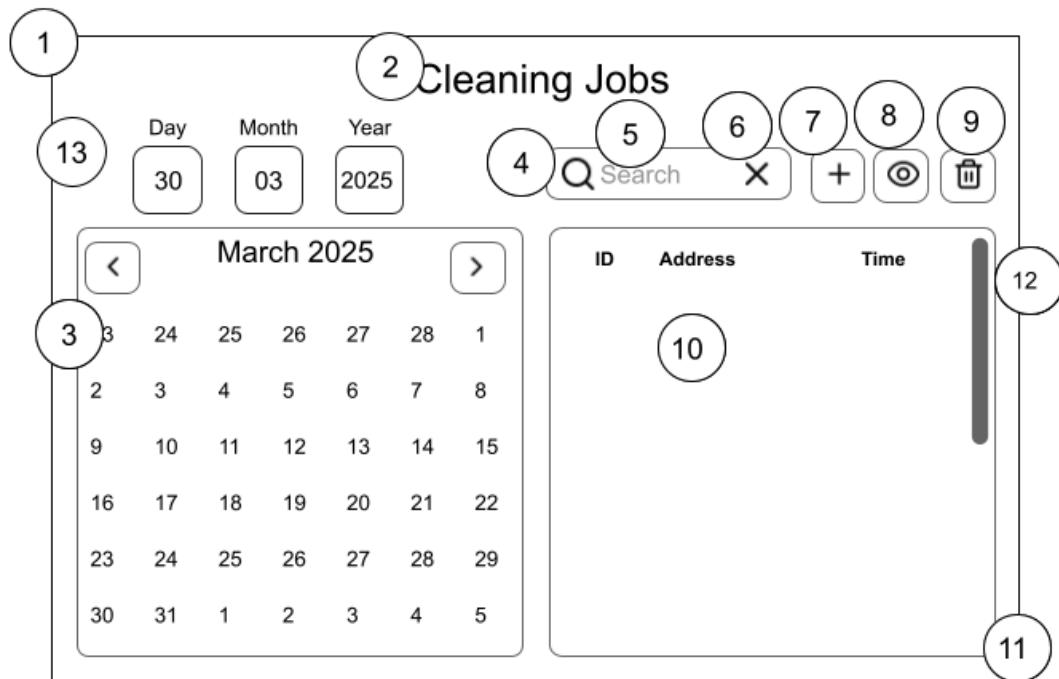
ID	Item	Walk Through	Styling
1	BookCleaningJobView	The form for displaying and booking cleaning jobs.	BackColor: Background Size: 649, 500
2	lblCleaningJob	The title for the book cleaning job form.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23
3	pnlTopBar	The panel that holds the search, add, edit, etc. controls.	BackColor: Background ForeColor: Foreground Size: 600, 50 Location: 10, 50
4	tbSearch	The text box into which the user inputs their search term to order the table by.	Forecolor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top

			FontSize: 12pt Placeholder: "Search"
5	btnAdd	The button the staff member clicks if they want to add a new cleaning job.	Location: 400, 5 Size 100, 40 Anchor: Top FontSize: 12pt Text: "Add"
6	btnEdit	The button the staff member clicks if they want to edit an existing cleaning job.	Location: 500, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Text: "Edit"
7	btnDelete	The button the staff member clicks if they want to delete a cleaning job.	Location: 600, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Text: "Archive"
8	dataGridView	The table in which the stock items are displayed. Column headers can be clicked to sort the corresponding column. This is scrollable.	Location: 100, 100 Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor: Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold
9	lblDate	A label indicating the date for the currently displayed cleaning jobs.	FontSize: 14pt ForeColor: Foreground Location: 202, 209 Size: 176, 23
10	tbDate	A text box for setting the date for the currently displayed cleaning jobs. This is also the date that will be populated when creating a cleaning job by default	ForeColor: Foreground Size: 300, 40 Location: 5, 5 Anchor: Top FontSize: 12pt

Feedback

This page could be improved by the usage of a dedicated control to select the date, (i.e. a calendar-esque control). Additionally, icons would improve the user experience.

Final Design



Description

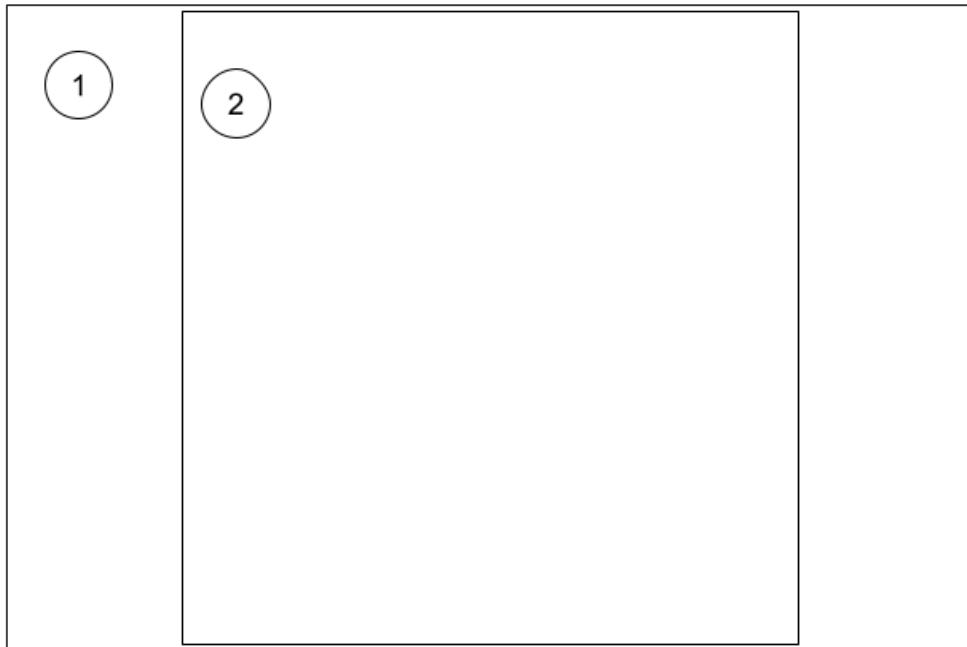
ID	Item	Walk Through	Styling
1	BookCleaningJobView	The form for displaying and booking cleaning jobs.	BackColor: Background Size: 649, 500
2	lblCleaningJob	The title for the book cleaning job form.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 97, 39 Location: 276, 23
3	calendar	The calendar from which the staff member can select a date.	BackColor: Background ForeColor: Foreground FontSize: 12pt Size: 300, 300 Location: 10, 50
4	pbSearch	A search icon to indicate the search box	BackColor: Background ForeColor: Foreground Size: 10, 10 Location: 12, 75
5	tbSearch	The text box into which the user inputs their search term to order the table by.	ForeColor: Foreground BorderColor: PrimaryForeground PlaceholderColor: Primary Size: 449, 40 Location: 12, 75 Anchor: Top FontSize: 12pt Placeholder: "Search"
6	pbClear	The picture box containing a cross icon. This can be clicked to clear the search bar.	BackColor: Background Location: 400, 5 Size 40, 40 Anchor: Top FontSize: 12pt Image: Plus
7	btnAdd	The button the staff member clicks if they want to add a new cleaning job.	Location: 400, 5 Size 100, 40 Anchor: Top FontSize: 12pt Image: Tick

8	btnEdit	The button the staff member clicks if they want to edit an existing cleaning job.	Location: 500, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Image: Cross
9	btnDelete	The button the staff member clicks if they want to delete a cleaning job.	Location: 600, 5 Size: 100, 40 Anchor: Top FontSize: 12pt Image: Eye
10	dataGridView	The table in which the stock items are displayed. Column headers can be clicked to sort the corresponding column.	Location: 100, 100 Size: 600, 400 Font Size: 12pt BackColor: Background ForeColor: Foreground SelectedForeColor: Foreground SelectedBackColor: PrimaryForeground BorderStyle: None EnableVisualStyles: True HeaderFontStyle: Bold
11	scrollBar	The scroll bar is linked to the data grid view. It is only visible if the columns overflow the displayable size.	BackColor: Background ThumbColor: PrimaryForeground CornerRadii: 10, 10, 10, 10
12	pnlDataGridView	The panel that holds the data grid view to improve the user interface by adding a rounded border.	BackColor: Background BorderColor: PrimaryForeground Location: 163, 162 Size: 312, 270 CornerRadii: 10, 10, 10, 10
13	diLastPasswordChange	A date input for setting the date for the currently displayed cleaning jobs. This is also the date that will be populated when creating a cleaning job by default.	BackColor: Background ForeColor: Foreground FontSize: 12pt Size: 180, 70 Location: 61, 110 BorderColor: PrimaryForeground ReadOnly: True

Reports Page

The report page will hold and display any PDF-based report.

Initial Design



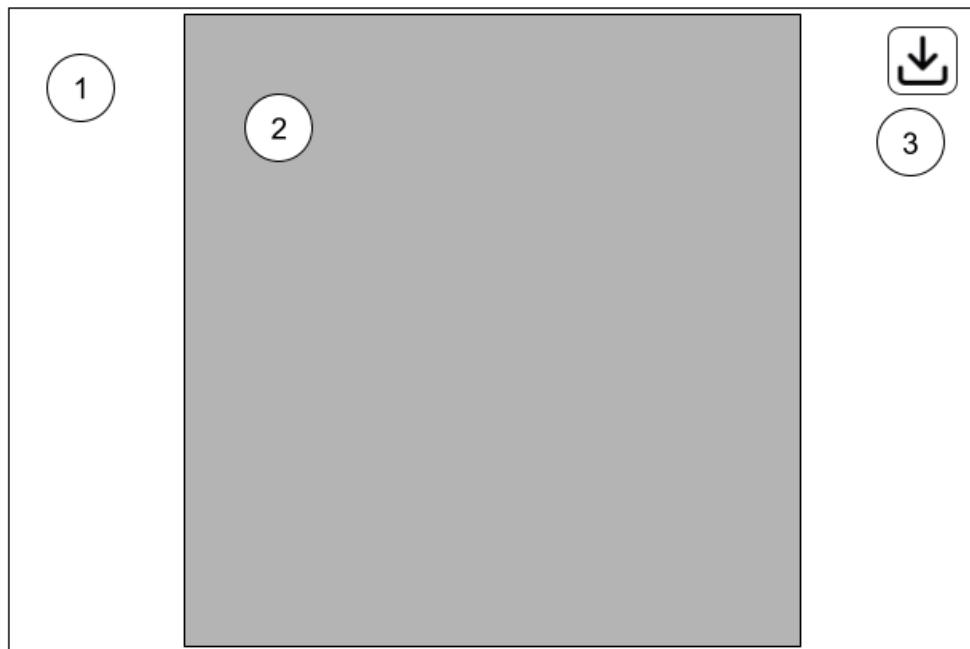
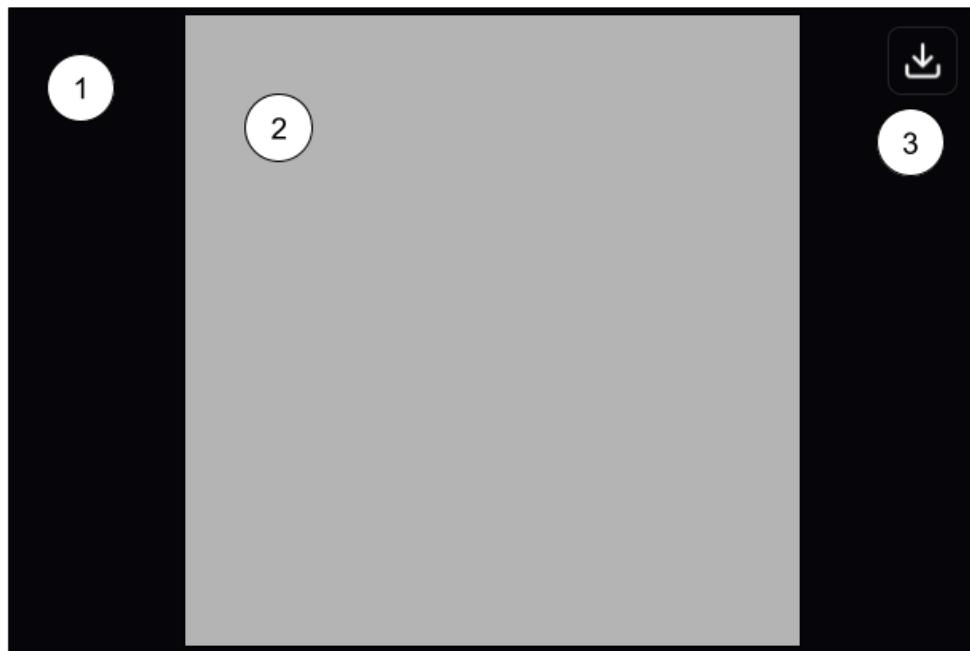
Description

ID	Item	Walk Through	Styling
1	ReportView	The form for displaying a report PDF.	BackColor: Background Size: 649, 500
2	pdfViewer.	The control displays a PDF and allows scrolling, zooming, etc.	BackColor: Background ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 400, 500 Location: 276, 23

Feedback

It would be useful to have a way to download the currently displayed report.

Final Design



Description

ID	Item	Walk Through	Styling
1	ReportView	The form for displaying a report PDF.	BackColor: Background Size: 649, 500
2	pdfViewer.	The control displays a PDF and	BackColor: Background

		allows scrolling, zooming, etc.	ForeColor: Foreground FontSize: 24pt AutoSize: True Size: 400, 500 Location: 276, 23
8	btnDownload	The button the staff member clicks if they want to save the currently displayed report.	BackColor: Background ForeColor: SecondaryForeground HoverColor: SecondaryForeground Location: 600, 5 Size: 40, 40 Anchor: Top FontSize: 12pt Image: Download

Report Storyboards

Current Stock Report

The current stock report will display the current stock levels for every stock item.

Initial Design

Stock Report			
ID	Name	SKU	Quantity
1	Dettol	DET-001	3
2	Mop	MOP-002	2
3	Soap	SOP-003	1

Description

The current stock report includes a title followed by a table containing all current stock items and their quantities.

Feedback

The report could be further improved by adding the Movers logo to the top of each page, along with more specific information about when the report was generated. This would be useful for auditing purposes. In addition, a column showing the quantity level could provide an overview of what stock might need to be reordered. It would also be useful if the quantity levels were coloured.

Final

Design

Report Name: Stock Report		MOVERS >		
ID	Name	SKU	Quantity	Quantity Level
1	Dettol	DET-001	3	High
2	Mop	MOP-002	2	Medium
3	Soap	SOP-003	1	Low

Description

The heading now appears at the top of the page and includes more detailed information about the report, the Movers logo, and a quantity-level column with colouring.

Current Staff Report

The current staff report will display all currently active staff on the Movers system.

Initial Design

Staff Report				
ID	Forename	Surname	Username	Privilege Level
1	John	Doe	jdoe	Cleaner
2	Bob	Jones	bjones	Manager
3	Alice	Smith	asmith	Office

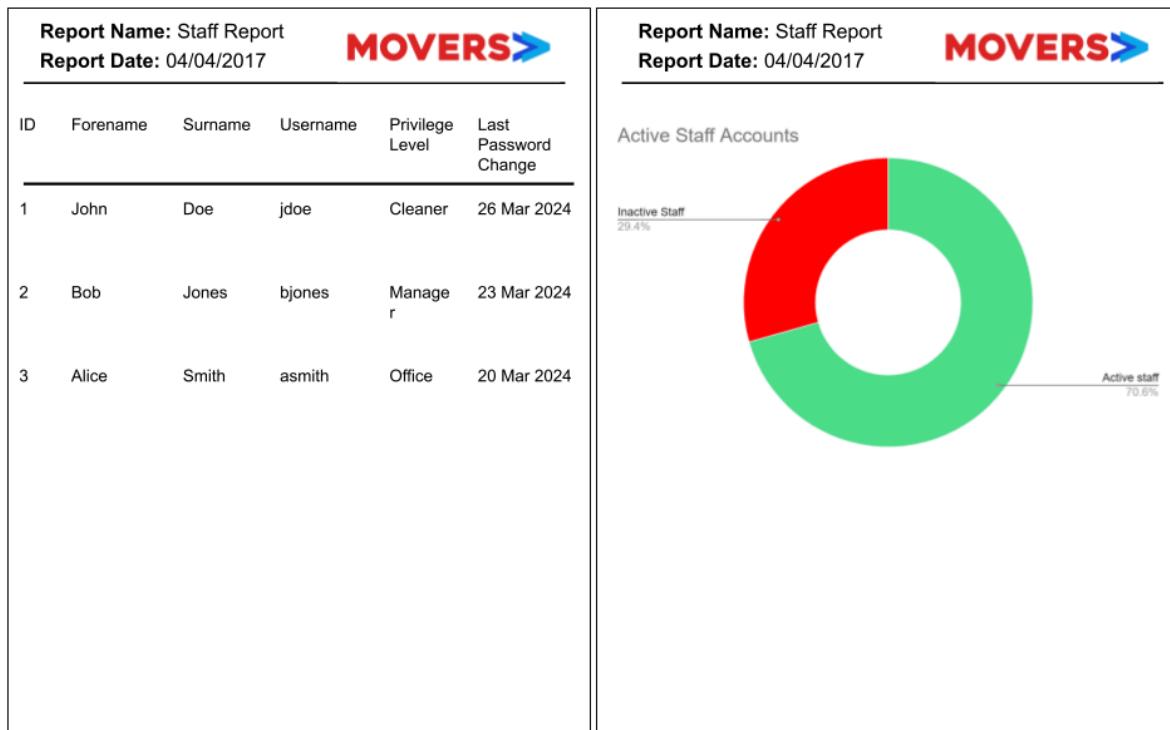
Description

The report features a title followed by a table showing the current staff (i.e., those who are not archived) registered in the Movers system. The table includes their privilege level name and username and can flow over multiple pages.

Feedback

The report could further be improved by adding the Movers logo to the top of each page, along with more specific information about when the report was generated. This would be useful for auditing purposes. In addition to this, a column showing the last password change date for each account could be useful to help monitor the system security. Finally, a chart showing all staff (both archived and not archived) could be useful to give a more general overview of the current and previous staff at Movers (i.e. employee retention).

Final Design



Description

The heading now appears at the top of the page and includes more detailed information about the report and the Movers logo. The final design has also been updated from the original to include a "Last Password Change" column within the staff table. After a page break, a pie chart showing the activity status of all staff accounts has been added.

Cleaning Job Report

Cleaning Job Reports are used to display a cleaning job in an invoice-like format.

Initial Design

Cleaning Job Report				
ID	Forename	Surname	Username	Privilege Level
1	John	Doe	jdoe	Cleaner
2	Bob	Jones	bjones	Cleaner
3	Alice	Smith	asmith	Cleaner
ID	Name	Unit Cost (£)	Quantity	
1	Window Cleaning	1.00	10	

Description

The cleaning job report displays cleaning job data on the first page, followed by a page break, then both tables displaying the cleaners on the job and the selected cleaning job options for the job in two tables.

Feedback

The report could be further improved by adding the Movers logo to the top of each page, along with more specific information about when the report was generated. This would be useful for auditing purposes. More data related to the customer who booked the cleaning job would be useful, e.g., customer name and contact details. In addition to this, address and extra information could be formatted within boxes to allow for more text (if an address or extra information spans multiple lines). Titles for each table would be useful, and page breaks between them would improve clarity. A total cost calculation would be useful, along with a subtotal column.

Final Design

Report Name: Cleaning Job Report Report Date: 04/04/2017	
Job ID: 23 Booked by: James doe	
Customer ID: 3 Customer name: Sarah Johnson Customer Email: sjohnson@outlook.com Customer Phone Number: 09876543212	
Date: 13/02/2025 Start Time: 11:00 End Time: 12:00	
Address <div style="border: 1px solid black; padding: 5px;">3 Oak Tree avenue Lisburn</div>	
Extra Information <div style="border: 1px solid black; padding: 5px;">Close the gate, the owner has a dog</div>	

Report Name: Cleaning Job Report Report Date: 04/04/2017											
Cleaning Job Options											
Total Cost: £10.00											
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>ID</th> <th>Name</th> <th>Unit Cost (£)</th> <th>Quantity</th> <th>Subtotal (£)</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Window Cleaning</td> <td>1.00</td> <td>10</td> <td>10.00</td> </tr> </tbody> </table>		ID	Name	Unit Cost (£)	Quantity	Subtotal (£)	1	Window Cleaning	1.00	10	10.00
ID	Name	Unit Cost (£)	Quantity	Subtotal (£)							
1	Window Cleaning	1.00	10	10.00							

Description

The heading now appears at the top of the page and includes more detailed information about the report and the Movers logo. The final design has also been updated from the original to include more customer information and better formatting. Tables have been formatted to improve clarity, and a total value and subtotal column has been created. Finally, the unnecessary privilege level column has been removed.

Pseudocode

Pseudocode has been included for some key algorithms within the project. Within pseudocode, the wait word shows waiting for an asynchronous process.

Levenshtein Distance Algorithm

The Levenshtein Distance Algorithm determines the minimum distance between two strings. To move from one string to another, characters can be added, substituted, or deleted. This algorithm orders items based on a search query and is used in any view that contains search functionality. The algorithm can be shown in two forms: a recursive and a non-recursive dynamic programming approach.

Recursive Implementation

The recursive implementation is simple but naive. It has an exponential growth time of $O(3^{\max(m, n)})$ for two strings of length m and n . This is because, in the worst case, each functional call can result in three more recursive calls. This implementation is also inefficient due to repeatedly calculating the same string distance for the same sub-string. This can be improved upon with memoisation, in which the string distance of each sub-string is stored for future use within the algorithm (e.g., in a hash map or dictionary).

```

FUNCTION LevenshteinDistance(a STRING, b STRING):
    IF LENGTH(a) EQUALS 0 THEN
        RETURN LENGTH(b)
    END IF
    IF LENGTH(b) EQUALS 0 THEN
        RETURN LENGTH(a)
    END IF

    IF a[LENGTH(a) - 1] EQUALS b[LENGTH(b) - 1] THEN
        cost ← 0
    ELSE
        cost ← 1
    END IF

    shortA ← SUBSTRING(a, 0, LENGTH(a) - 1)
    shortB ← SUBSTRING(b, 0, LENGTH(b) - 1)

    RETURN MIN(
        LevenshteinDistance(shortA, b) + 1,           // Deletion
        LevenshteinDistance(a, shortB) + 1,           // Insertion
        LevenshteinDistance(shortA, shortB) + cost // Substitution
    )
END FUNCTION

```

Dynamic Programming Implementation

The dynamic programming implementation is more efficient within C#. It has no recursive function calls, so it is unlikely to cause a stack overflow. In addition, there is less overhead for each function call, as there is only one initial function call before the loop. This algorithm greatly improves upon the recursive implementation by using arrays to store previous substring calculations. This implementation has a time complexity $O(m \times n)$. I will be using the following algorithm within the application.

```
FUNCTION LevenshteinDistance(a STRING, b STRING):
    rowOne  $\leftarrow$  NEW CHARACTER ARRAY of size LENGTH(b) + 1
    rowTwo  $\leftarrow$  NEW CHARACTER ARRAY of size LENGTH(b) + 1

    IF a is empty THEN
        RETURN LENGTH(b)
    END IF
    IF b is empty THEN
        RETURN LENGTH(a)
    END IF

    FOR i  $\leftarrow$  0 TO LENGTH(b):
        rowOne[i]  $\leftarrow$  i
    END FOR

    FOR i  $\leftarrow$  0 TO LENGTH(a) - 1:
        rowOne[0]  $\leftarrow$  i + 1

        FOR j  $\leftarrow$  0 TO LENGTH(b) - 1:
            deletionCost  $\leftarrow$  rowOne[j + 1] + 1
            insertionCost  $\leftarrow$  rowTwo[j] + 1
            substitutionCost  $\leftarrow$  0

            IF a[i] = b[j] THEN
                substitutionCost  $\leftarrow$  rowOne[j]
            ELSE
                substitutionCost  $\leftarrow$  rowOne[j] + 1
            END IF

            rowTwo[j + 1]  $\leftarrow$  MIN(
                deletionCost,
                insertionCost,
                substitutionCost
            )
        END FOR
```

```

    SWAP(rowOne, rowTwo) // Swap the arrays for the next iteration
END FOR

RETURN rowOne[LENGTH(b)]
END FUNCTION

```

Usage

The Levenstein distance algorithm is then used as part of a ranking algorithm which matches substrings to produce a final heuristic for how closely a user's search query matches the other string. This function will be called from within classes that have a search functionality.

```

FUNCTION SubstringLevenshteinDistance(a STRING, b STRING):
    // Ensure b is the shorter string
    IF LENGTH(a) < LENGTH(b) THEN
        SWAP(a, b)
    END IF

    IF LENGTH(b) EQUALS 0 THEN
        RETURN 0
    END IF

    minDistance ← MAXIMUM_INTEGER_VALUE

    // Slide the shorter string across the longer string
    FOR i ← 0 TO LENGTH(a) - LENGTH(b):
        subA ← SUBSTRING(a, i, LENGTH(b))
        distance ← LevenshteinDistance(subA, b) // Assuming
LevenshteinDistance function exists
        minDistance ← MIN(minDistance, distance)

        // Early exit if exact match found
        IF minDistance EQUALS 0 THEN
            BREAK
        END IF
    END FOR

    RETURN minDistance
END FUNCTION

```

Password Cryptography

Passwords are stored securely within the database using hashing and salting. This process involves converting a password into a seemingly random string of bytes of a fixed length through a hashing algorithm (SHA-512). The same password input consistently produces the same hash. Hashing algorithms are designed to be computationally infeasible to reverse; however, their deterministic nature renders them vulnerable to lookup tables known as "Rainbow Tables," which contain hundreds of thousands of precomputed hashes for common passwords. This is combatted by using a randomly generated salt with the password. This ensures the user's password is completely unique and cannot be found within any lookup tables.

Cryptography Class

The cryptographic operations will be contained within a cryptography class.

```
CLASS CryptographyManager:

CONSTANT INTEGER ITERATIONS ← 100000
CONSTANT INTEGER SIZE ← 64
CONSTANT HASH ALGORITHM HASH_ALGORITHM ← SHA512

FUNCTION GetHash(text STRING, OUT salt BYTE ARRAY) :
    salt ← GetRandomBytes(SIZE)
    RETURN GetHash(text, salt)
END FUNCTION

FUNCTION GetHash(text STRING, salt BYTE ARRAY) :
    textBytes ← ConvertStringToBytes(text)
    // Use the password-based key derivation function from the RSA
laboratory
    hash ← PBKDF2(textBytes, salt, ITERATIONS, HASH_ALGORITHM, SIZE)
    RETURN hash
END FUNCTION

FUNCTION VerifyHashEquality(text STRING, hash BYTE ARRAY, salt BYTE
ARRAY) :
    textBytes ← ConvertStringToBytes(text)
    hashedText ← PBKDF2(textBytes, salt, ITERATIONS, HASH_ALGORITHM,
SIZE)
    RETURN FixedTimeEquals(hash, hashedText)
END FUNCTION

FUNCTION GetRandomBytes(size INTEGER) :
    // Use a cryptographically secure random number generator.
    bytes ← NEW BYTE ARRAY of size 'size'
    FILL bytes WITH cryptographically secure random values
    RETURN bytes
```

```

END FUNCTION

FUNCTION ConvertStringToBytes(text STRING):
    bytes ← CONVERT text TO BYTE ARRAY USING UTF-8 encoding
    RETURN bytes
END FUNCTION

FUNCTION FixedTimeEquals(hash1 BYTE ARRAY, hash2 BYTE ARRAY):
    // Should compare the byte arrays in a way that takes a fixed
    amount of time, regardless of the inputs. This prevents timing-based
    attacks.
    result ← COMPARE hash1 and hash2 WITH fixed-time comparison
    algorithm.
    RETURN result
END FUNCTION
END CLASS

```

Usage

The cryptography class will be used mainly on the application's sign-in page. The code below shows how the sign-in method will use the cryptography class methods and provides some basic error handling.

```

FUNCTION SignIn():
    username ← view.Username
    password ← view.Password

    isUsernameEmpty ← EMPTY(username)
    isPasswordEmpty ← EMPTY(password)

    IF isUsernameEmpty AND isPasswordEmpty THEN
        view.ErrorText ← "Please fill in a username and password"
        RETURN // Exit and do no further validation
    END IF

    IF isUsernameEmpty THEN
        view.Password ← "" // Clear the password field
        view.ErrorText ← "Please fill in a username and password"
        RETURN // Exit and do no further validation
    END IF

    IF isPasswordEmpty THEN
        view.ErrorText ← "Please fill in a password"
        RETURN // Exit and do no further validation
    END IF

    // Put the UI in loading mode to prevent interactions.
    view.Loading ← TRUE

```

```
TRY
    Hash, salt ← WAIT FOR GetStaffCredentialsFromDatabase(username) AND
    WAIT FOR DELAY(1000)

    IF hash EQUALS NULL OR salt EQUALS NULL THEN
        view.ErrorText ← "Invalid username or password"
        CALL LogLoginAttemptInDatabase(username, CURRENT_TIME, FALSE)
        RETURN // Exit and do no further validation
    END IF

    IF CryptographyManager.VerifyHashEquality(password, hash, salt) THEN
        // Tell the other parts of the application the login was
        successful by navigating to the home page
        CALL NavigateToHomePage()
        CALL LogLoginAttemptInDatabase(username, CURRENT_TIME, TRUE)
    ELSE
        view.ErrorText ← "Invalid username or password"
        CALL LogLoginAttemptInDatabase(username, CURRENT_TIME, FALSE)
        RETURN // Exit and do no further validation
    END IF

    CATCH
        view.DisplayErrorMessageBox("Error connecting to database")
    FINALLY
        view.Loading ← FALSE
    END TRY
END FUNCTION
```

Assigning Cleaners To Cleaning Jobs

When assigning cleaners to clean jobs, there is a requirement to prevent double-booking (i.e., booking the same cleaners onto different jobs at the same time). To see which cleaners are free for a particular job at a particular time, the cleaners on all other jobs that occur on that day must be checked to see if they intersect with the cleaning job. If they intersect, all cleaners on the other job are considered unavailable for the job for which the staff member selects cleaners.

To determine if a cleaning job intersects, there are three cases to consider. This includes if the cleaning job's start time lies within the other cleaning job's start and end time (case 1), if the cleaning job's end time lies within the other cleaning job's start and end time (case 2), or if the cleaning job starts before the other job but ends after it (case 3). This is illustrated below:



Implementation

```

FUNCTION GetAvailableCleaners(cleaningJob):
    cleaningStaff ← CALL GetNonArchivedCleaningStaffFromDatabase()
    cleaningJobsOnDay ← CALL GetCleaningJobsByDate(cleaningJob.StartDate)

    FOR EACH model IN cleaningJobsOnDay:
        IF model.Id EQUALS cleaningJob.Id THEN

```

```
CONTINUE // Skip this job (it's the same job input into the method)
END IF

// Check for date/time overlap
IF (model.StartDate <= cleaningJob.EndDate AND model.EndDate >=
cleaningJob.StartDate) OR
    (model.StartDate <= cleaningJob.StartDate AND model.EndDate >=
cleaningJob.EndDate) THEN

    FOR EACH staffId IN model.StaffIds:
        REMOVE FROM cleaningStaff WHERE staff.Id EQUALS staffId
    END FOR
END IF
END FOR

RETURN cleaningStaff
END FUNCTION
```

Displaying A Child-View

Displaying a child-view is one of the most essential methods in the entire application. It allows forms to display subforms inside panels. This method is implemented in any forms that have sub-views.

```
FUNCTION DisplayChildView(childView CHILDVIEW_INTERFACE):
    // Remove the previous child form
    CALL ClearCurrentSubView()

    ChildView ← childView;

    // Setup the child form to be displayed
    ChildView.TopLevel ← FALSE;
    ChildView.Width ← pnlHolder.Width;
    ChildView.Anchor ← AnchorStyles.Left AND AnchorStyles.Right AND
    AnchorStyles.Top;
    ChildView.Dock ← Fill;

    // Display the form
    pnlHolder.Controls.Add(ChildView as Form);
    CALL Show();
END FUNCTION
```

Control Methods

Recursive Control Methods

This method recursively searches through a control's child controls and applies a function. It is useful for setting properties on all controls within a parent control.

```
FUNCTION ExecuteRecursive(control CONTROL, action FUNCTION(CONTROL)):
    queue ← NEW QUEUE
    ENQUEUE control INTO queue

    WHILE LENGTH(queue) > 0:
        current ← DEQUEUE from queue
        FOR EACH child IN current.Controls:
            ENQUEUE child INTO queue
        END FOR
        CALL action(current)
    END WHILE
END FUNCTION
```

Database

Connection String

A connection string is required to connect to a database. This string will be dynamically created based on the application's root directory.

```
CLASS ExampleDAL:
    CONSTANT workingDirectoryPath ← CALL GetCurrentDirectoryPath()
    CONSTANT projectDirectoryPath ← CALL
workingDirectoryPath.GetParent().GetParent()
    CONSTANT connectionString ← FormatString(CALL AppConfig.ConnectionString(),
projectDirectoryPath)
```

Data Access Layer Queries

Methods that gather data from the database will exist within the Data Access Layer (DAL) classes. Examples are shown below.

Example Get Data

This pseudocode shows the example get method to get customers from the database.

```
FUNCTION GetCustomers():
    connection ← CALL StartConnection(connectionString)

    reader ← CALL executeStoredProcedureReader("GetCustomers")

    customers ← NEW LIST

    WHILE CALL reader.ReadAsync():
        customer ← NEW CustomerModel(
            CALL reader.GetInteger(reader.GetOrdinal("Id")),
            CALL reader.GetString(reader.GetOrdinal("Forename")),
            CALL reader.GetString(reader.GetOrdinal("Surname")),
            CALL reader.GetString(reader.GetOrdinal("Email")),
            CALL reader.GetString(reader.GetOrdinal("PhoneNumber")),
            CALL reader.GetString(reader.GetOrdinal("Address")),
            CALL reader.GetBoolean(reader.GetOrdinal("Archived"))
        )
        ADD customer T0 customers
    END WHILE

    RETURN customers
END FUNCTION
```

Example Add Data

Here is an example add method, which adds an order to the database with the “CreateOrder” stored procedure.

```
FUNCTION CreateOrder(model ORDER_MODEL):
    connection ← CALL StartConnection(connectionString)

    connection.parameters["@status"] ← model.Status
    connection.parameters["@discrepancies"] ← model.Discrepancies
    connection.parameters["@description"] ← model.Description
    connection.parameters["@staffId"] ← model.Staff.Id

    results ← CALL executeStoredProcedureReader("CreateOrder",
parameters)

    IF LENGTH(results) EQUALS 0 THEN
        RETURN FALSE
    END IF

    RETURN TRUE
END FUNCTION
```

Example Update Data

Here is an example update method that shows the quantity of a stock item.

```
FUNCTION UpdateStockQuantity(stockId INTEGER, staffId INTEGER, quantity
INTEGER, date DATETIME, reasonForQuantityChange STRING):
    connection ← CALL StartConnection(connectionString)

    CALL command.Parameters.AddWithValue("@stockId", stockId)
    CALL command.Parameters.AddWithValue("@staffId", staffId)
    CALL command.Parameters.AddWithValue("@quantity", quantity)
    CALL command.Parameters.AddWithValue("@date", date)
    CALL command.Parameters.AddWithValue("@reasonForQuantityChange",
reasonForQuantityChange)

    rowsAffected ← CALL executeStoredProcedureReader("UpdateStockQuantity")

    RETURN rowsAffected > 0
END FUNCTION
```

Example Delete Data

Below is an example of deleting cleaning job options from a cleaning job.

```
FUNCTION DeleteCleaningJob(id INTEGER):
    connection ← CALL StartConnection(connectionString)

    CALL command.Parameters.AddWithValue("@id", id)

    rowsAffected ← CALL
executeStoredProcedureReader("DeleteCleaningJobOptions")

    RETURN rowsAffected > 0
END FUNCTION
```

Forms

Below is the pseudocode for vital validation processes within forms and other UI updates.

Signing In

Signing in references the cryptography class to check password validity. This will be located within the sign-in presenter class.

```
FUNCTION AttemptSignIn():
    username ← view.Username
    password ← view.Password

    isUsernameEmpty ← username IS NULL OR username IS EMPTY
    isPasswordEmpty ← password IS NULL OR password IS EMPTY

    IF isUsernameEmpty AND isPasswordEmpty THEN:
        view.ErrorText ← "Please fill in a username and password"
        RETURN
    END IF

    IF isUsernameEmpty THEN:
        view.Password ← ""
        view.ErrorText ← "Please fill in a username and password"
        RETURN
    END IF

    IF isPasswordEmpty THEN:
        view.ErrorText ← "Please fill in a password"
        RETURN
    END IF

    view.Loading ← TRUE

TRY:
    (hash, salt) ← WAIT CALL GetStaffCredentialsFromDatabase()

    IF hash IS NULL OR salt IS NULL THEN:
        CALL SignInError()
        WAIT CALL LogLoginAttemptInDatabase()
        RETURN
    END IF

    IF CryptographyManager.VerifyHashEquality(password, hash, salt) THEN:
        CALL TellOtherClassesSignInSuccessful()
    ENDIF

    ELSE:
        CALL SignInError()
        WAIT CALL LogLoginAttemptInDatabase()
```

```
    RETURN
END IF

CATCH:
    CALL MessageBox WITH "Error connecting to database")
END CATCH
FINALLY:
    view.Loading ← FALSE
END TRY

FUNCTION SignInError():
    view.Password ← ""
    view.ErrorText ← "Username or password incorrect"
    view.Loading ← FALSE
END FUNCTION
```

Password Requirements

When creating or updating password requirements, some code similar to the pseudocode below will be used.

```
FUNCTION ValidatePasswordRequirements():
    view.EightLong ← LENGTH(view.NewPassword) >= 8
    view.UppercaseLowercase ← view.NewPassword CONTAINS UPPERCASE AND
    view.NewPassword CONTAINS LOWERCASE
    view.Number ← _view.NewPassword CONTAINS DIGIT
    view.SpecialCharacter ← NOT view.NewPassword CONTAINS ALPHANUMERIC
    CHARACTER

    IF view.EightLong AND view.UppercaseLowercase AND view.Number AND
    view.SpecialCharacter THEN:
        _view.SetNewPasswordBorderError(FALSE)
        RETURN TRUE
    ELSE:
        _view.SetNewPasswordBorderError(TRUE)
        RETURN FALSE
    END IF

FUNCTION ValidatePasswordInputs():
    newPasswordEmpty ← LENGTH(view.NewPassword) EQUALS 0
    confirmPasswordEmpty ← LENGTH(view.ConfirmPassword) EQUALS 0

    IF newPasswordEmpty OR confirmPasswordEmpty THEN:
        CASE (newPasswordEmpty, confirmPasswordEmpty) OF:
            (TRUE, TRUE):
                view.PasswordError ← "New and confirm password are required"
            (TRUE, FALSE):
                view.PasswordError ← "New password is required"
            (FALSE, TRUE):
                view.PasswordError ← "Confirm password is required"
        END CASE
        RETURN FALSE
    END IF

    IF NOT CALL ValidatePasswordRequirements() THEN:
        view.PasswordError ← "Password does not satisfy the requirements"
        RETURN FALSE
    END IF

    IF view.NewPassword NOT EQUALS view.ConfirmPassword THEN:
        view.PasswordError ← "Passwords did not match"
        RETURN FALSE
    END IF

    view.PasswordError ← ""
    RETURN TRUE
```

```
FUNCTION ValidateUsername():
    usernameValid ← LENGTH(view.Username) NOT EQUALS 0

    view.SetUsernameBorderError(NOT usernameValid)
    view.UsernameError ← IF usernameValid THEN "" ELSE "Fill in a username"

    IF NOT usernameValid THEN RETURN FALSE

    staff ← WAIT CALL GetStaffFromDatabaseByUsername(view.Username)
    usernameValid ← staff IS NOT NULL
    view.SetUsernameBorderError(NOT usernameValid)
    view.UsernameError ← IF usernameValid THEN "" ELSE "The username does
not exist"

    IF NOT usernameValid THEN RETURN FALSE

    RETURN TRUE
```

Dashboard Message

On the application's dashboard page, a welcome message will be displayed, and a warning will be shown based on the date of the staff member's last password change. The pseudocode for this is shown below.

```
FUNCTION PopulateDefaultValues():
    IF CURRENT_TIME.Hour < 12 THEN:
        welcomeMessage ← "Good Morning"
    ELSE IF CURRENT_TIME.HOUR < 16 THEN:
        welcomeMessage ← "Good afternoon"
    ELSE:
        welcomeMessage ← "Good evening"
    END IF

    view.Welcome ← welcomeMessage + ", " + _staff.Forename
    view.LastPasswordChange ← _staff.LastPasswordChange
    view.ShowChangePassword ← _staff.LastPasswordChange < CURRENT_TIME -
NEW TIMESPAN(30) // Check if the last password change is older than
thirty days ago
```

Input-Output-Process (IPO)

IPO is an approach within software engineering and systems analysis that describes the structure of a program and how information is captured and processed within a diagram. Input is captured from the user and then processed (by the Process) and converted to an Output. The Output is what the system displays when the process is complete.

The main inputs within the microsystem are staff members, stock information, orders, bookings, customers and cleaning job options.

Staff

Input	Process	Output
Staff details (forename, surname, email, etc.).	Store in the Staff table in the database.	Staff records are available for display and reporting.
Update staff details (e.g., changing the archived status).	Update the corresponding staff data within the Staff table.	Updated staff record.
Sign in (username and password provided).	Store login attempts in the LoginAttempt table and verify credentials.	Authentication result (success or failure).

Stock

Input	Process	Output
Add new stock item (name, description, SKU, unit cost, quantity).	Store in the Stock table in the database.	A new stock record was created. Stock records are available for display and reporting.
Update stock details (e.g., cost, quantity, archived status).	Modify the corresponding entry in the Stock table.	Updated stock record.
Record stock quantity changes (e.g., restocking, usage).	Store changes in the StockQuantity table with the reason and staff ID.	The stock history is recorded and displayable.

Orders

Input	Process	Output
Order stock items (some details and a selection of stock items and quantities).	Store order details in the Order and Order_Stock tables.	Stock linked to an order.
Receive order (the number of each item received and the quantity).	Update the order status to "Delivered" and update the stock by adding the items that were received.	The order is displayed as received. The stock items are displayed with their new quantities.

Bookings

Input	Process	Output
Create a new booking (customer details, date, time, services required).	Store booking details in the CleaningJob table.	A new booking record was created. Booking is available for display and reporting.
Assign staff to a booking (cleaning job ID, staff ID).	Store assignment in the CleaningJob_Staff table.	Assigned staff displayed for the booking.
Update booking details (e.g., reschedule, add extra information).	Modify the corresponding entry in the CleaningJob table.	Updated booking record.
Cancel a booking.	Remove the entry from the CleaningJob table.	Booking removed from records.
View upcoming bookings.	Retrieve scheduled jobs from the CleaningJob table based on date filters.	List of upcoming bookings displayed.

Customers

Input	Process	Output
Add a new customer (forename, surname, email, phone number, address).	Store customer details in the Customer table.	A new customer record was created. Customers are available for display and reporting.

Update customer details (e.g., change phone number, update address).	Modify the corresponding entry in the Customer table.	Updated customer record.
Archive a customer.	Update the Archived field in the Customer table.	The customer is marked as archived and hidden from the active list.
Add a new customer (forename, surname, email, phone number, address).	Store customer details in the Customer table.	A new customer record is created. Customers are available for display and reporting.

Cleaning Job Options

Input	Process	Output
Add a new cleaning job option (name, description, unit cost).	Store option details in the CleaningJobOption table.	A new cleaning job option is created. Option available for selection in bookings.
Update cleaning job option details (e.g., name, cost, description).	Modify the corresponding entry in the CleaningJobOption table.	Updated cleaning job option record.
Archive a cleaning job option.	Update the Archived field in the CleaningJobOption table.	The cleaning job option is marked as archived and hidden from selection.
Add a new cleaning job option (name, description, unit cost).	Store option details in the CleaningJobOption table.	A new cleaning job option is created. Option available for selection in bookings.

User Feedback Evaluation

I created and distributed a form to gather users' responses to gauge how they felt about the planning process. This allowed me to receive feedback on its strengths and weaknesses before implementing the solution. The link to the form can be found [here](#). Screenshots of the form have also been included below.

User Feedback Form

Movers Design Feedback Form

Please give constructive feedback on the Movers design process.



* Indicates required question

What is your role in the company? *

- Manager
- Office Staff
- Cleaner
- Cleaning Manager

Does the data dictionary contain all relevant information? *

- Yes
- No

Are there any missing use cases within the use case diagram? *

Yes

No

Rate how aesthetically pleasing the storyboards' user interface designs are. *

1

2

3

4

5

Poor

Good



Rate how easy you think the storyboards would be to navigate within the application. *

1

2

3

4

5

Difficult

Easy



Do you feel any requirements are missing? *

Your answer

Submit

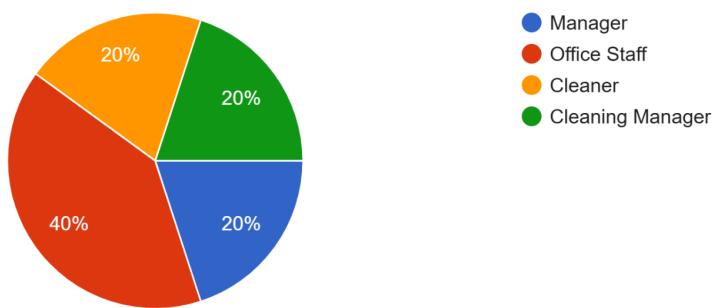
Clear form

User Feedback Results

Question 1

What is your role in the company?

5 responses

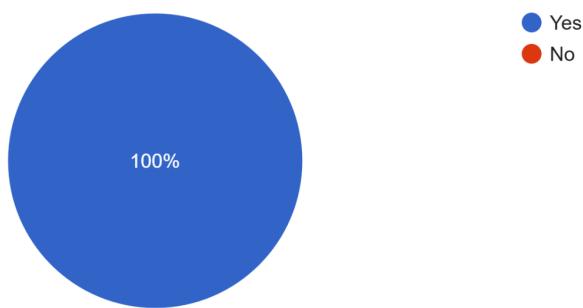


The first question ensures that a wide variety of staff with different roles within Movers have been consulted. Therefore, the feedback provides views from a range of experiences and will be representative of the whole company's collective feedback.

Question 2

Does the data dictionary contain all relevant information?

5 responses

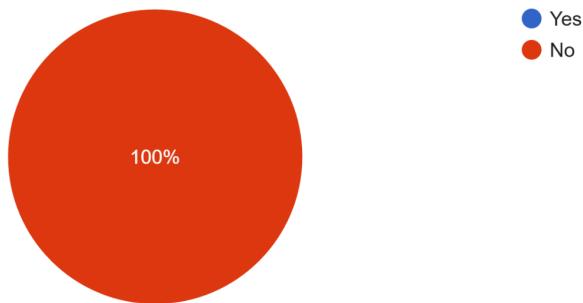


Question 2 shows that all those who completed the form thought the data dictionary contained relevant information. This gives me confidence that the application will have enough information to be useful in relieving the issues within the company.

Question 3

Are there any missing use cases within the use case diagram?

5 responses

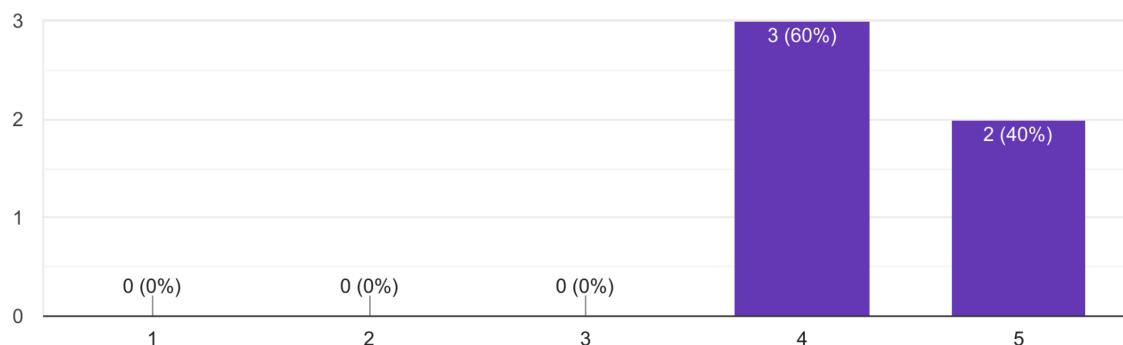


Based on this positive feedback, the use-case diagram clearly covered all possible use cases. This provides another good indicator that the final application will be useful for running and administering Movers.

Question 4

Rate how aesthetically pleasing the storyboards' user interface designs are.

5 responses

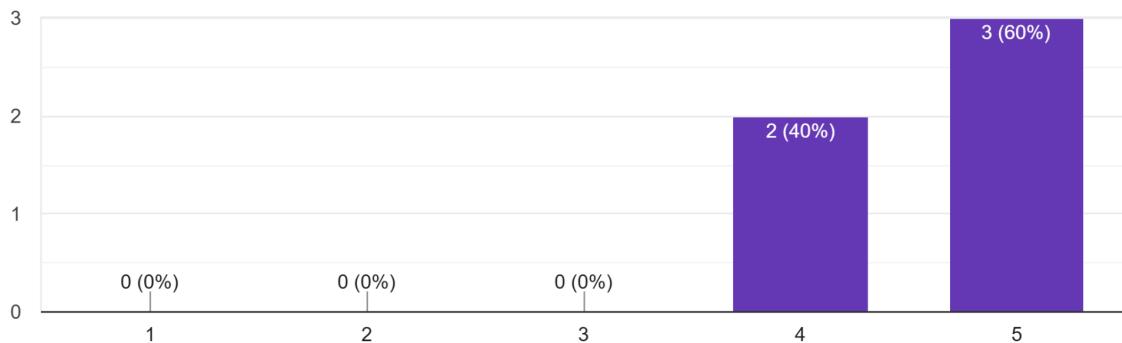


This feedback reveals that the storyboards provide a very strong design on which to base the application's user interface. I am confident the design will be positively received within the completed application.

Question 5

Rate how easy you think the storyboards would be to navigate within the application.

5 responses



The storyboards have also received positive feedback on how easy they will be to navigate and use. This provides me with confidence that the final application will be well received.

Conclusion

Overall, the feedback form has given me confidence that my application will solve the problems faced by Movers and will be well received by staff throughout the company. This provides an early indicator that the application will please the stakeholders and that the project will be successful.

Testing

Types of Testing

Testing is a vital part of the development process. It has several key goals, including detecting defects in the application and gathering user feedback. There are two main types of testing.

Black Box Testing

Black box testing involves testing the outer structure of the application. The tester does not know the inner structure of the code. Testers put inputs into the application and observe the outputs. If the outputs differ from what should be expected, they are flagged and fixed.

White Box Testing

White box testing involves techniques that analyse the inner structures of code and data structures within the application. White box testers have access to the codebase and perform their tests directly within the code.

Parts of Testing

Unit Testing

Unit testing involves checking each part of the code. This improves quality by testing each unit individually. Many different frameworks help automatically run unit tests whenever code changes. Unit testing also promotes modular code.

Integration Testing

Integration testing describes tests that run as the development modules of the application are coupled together. There are several different types of integration such as big-bang, mixed, top-down and bottom-up. Waterfall generally uses big-bang integration. This involves all of the development modules being coupled at once and then tested.

System Testing

System testing is when a fully integrated system is tested to evaluate its compliance with the agreed-upon user requirements. It tests the system's design and behaviour to ensure it aligns with the customer's expectations. System testing includes performance testing, load testing, stress testing, and scalability testing.

Acceptance Testing

Acceptance testing is the final stage of the Waterfall testing pathway. It is a crucial step in ensuring the completeness and quality of a software project. It is also necessary to ensure stakeholder satisfaction, which determines the project's

success. User acceptance testing, a subsection of acceptance testing, is generally performed by end-users or clients who will use the software. User acceptance testing includes beta testing, black box testing, operational acceptance testing and contract acceptance testing. Business acceptance testing, another subsection of acceptance testing, aims to ensure the project provides a solution that aligns with the business's desires, supplying assurance to stakeholders that the system is suited for its intended purpose.

Data Entry Testing

Data entry testing involves inserting data into the application and observing how the data is treated and eventually stored. It does not require direct access to the application's internal code.

Link Testing

Link testing involves testing links and navigation within an application's user interface. It helps ensure that all redirects to different pages or views within an application work as intended.

Conclusion

For the application, I will be using mainly black box testing to ensure everything functions correctly from the user's point of view. The testing will be divided into three key areas: link testing, data entry testing, and user acceptance testing (user acceptance testing is located at the start of the evaluation testing). It is in the form of a survey and evaluation of the success of implementing user requirements. I will be using black box testing since it is easier to plan and complete than full white box testing.

Test Plan

Testing: Link Testing					
Personnel:		Matthew Gracey			
Environment:		Windows 11 Laptop (Intel i7, 16GB RAM, 1TB SSD)			
Splash Screen					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
1	The splash screen appears.	App start.	The splash screen is visible and centered on the screen.		
2	Splash screen timeout.	App start.	The splash screen disappears after 1 second (approx.).		
3	Splash screen transition.	App start.	Sign-in page opens		
Sign In Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
4	A successful sign in directs to the main menu.	Username: "jdoe1" Password: "password123!"	Sign-in page disappears and main page opens		
5	Users can toggle the show password.	Click the show password icon. Password: "password"	Password character is removed, and password is visible in plain text.		
6	Switch theme in the login page	Click the switch theme button.	The view changes from one theme to the other, and the icon changes from a sun (dark mode) to a moon (light mode).		

7	Tab order	Press the tab key repeatedly.	The controls change focus in a sensible manner.		
Main Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
8	Sign out works.	Click the sign-out button at the top right.	The application reloads.		
9	Menu tabs work.	Click the menu tabs.	The tabs open downward to expose the menu items. The icon changes from a plus to a minus.		
10.1	Correct menu items display	Sign in to the application with a cleaning staff account.	<p>The menu items are:</p> <p>Dashboard</p> <p>Cleaning (Upcoming jobs)</p> <p>Settings (Personal information, Contact details, Emergency contact, Account security, Appearance)</p>		
10.2		Sign in to the application with an office staff account.	<p>The menu items are:</p> <p>Dashboard</p> <p>Cleaning (Book cleaning, Upcoming jobs, Manage customers, Manage options)</p> <p>Stock (Orders)</p> <p>Settings (Personal information, Contact details, Emergency contact, Account security, Appearance)</p>		

10.3	Sign in to the application with a cleaning manager account.	The menu items are: Dashboard Stock (Manage stock, Order stock, Quantity changes, Upcoming deliveries) Reports (Stock) Settings (Personal information, Contact details, Emergency contact, Account security, Appearance)		
10.4	Sign in to the application with an administrator account.	The menu items are: Dashboard Security (Manage staff, Login attempts, Change password) Reports (Staff) Settings (Personal information, Contact details, Emergency contact, Account security, Appearance)		
10.5	Sign in to the application with a manager account.	The menu items are: Dashboard Stock (Manage stock, Orders, Quantity changes) Cleaning (Book cleaning, Upcoming jobs, Manage customers, Manage options) Reports (Stock, Staff)		

			Settings (Personal information, Contact details, Emergency contact, Account security, Appearance)		
Dashboard Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
11	Check the change password button redirects to the security settings page.	Sign in to a staff account with a password that is older than 30 days. Click the Change Password button.	Redirected to the security settings page.		
Personal Information Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
12.1	Click the save button.	Change any input so the save and cancel buttons appear. Ensure there is valid data. Click the save button.	Personal information is saved to the database.		
12.2		Change any input so the save and cancel buttons appear. Ensure there is invalid data. Click the save button.	The personal information is not saved to the database and a message box appears indicating the error.		
13	Click the cancel button	Change any input. Click the cancel button.	The personal information all resets.		
14	Tab order	Press the tab key repeatedly.	The controls change focus in a sensible manner.		

Contact Information Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
15.1	Click the save button.	Change any input so the save and cancel buttons appear. Ensure there is valid data. Click the save button.	The contact information is saved to the database.		
15.2		Change any input so the save and cancel buttons appear. Ensure there is invalid data. Click the save button.	The contact information is not saved to the database, and a message box appears indicating the error.		
16	Click the cancel button	Change any input. Click the cancel button.	The contact information all resets.		
17	Tab order	Press the tab key repeatedly.	The controls change focus in a sensible manner.		
Emergency Contact Information Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
18.1	Click the save button.	Change any input so the save and cancel buttons appear. Ensure there is valid data. Click the save button.	The emergency contact information is saved to the database.		
18.2		Change any input so the save and cancel buttons appear. Ensure there is invalid	The emergency contact information is not saved to the database, and a message box appears indicating the error.		

		data. Click the save button.			
19	Click the cancel button	Change any input. Click the cancel button.	The emergency contact information all resets.		
20	Tab order	Press the tab key repeatedly.	The controls change focus in a sensible manner.		

Appearance Settings Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
21	Appearance settings toggle.	Click the appearance settings radio button twice.	The radio button toggles (gives a full circle) then untoggles.		
22.1	Font selection.	Click each font radio button.	The clicked font radio button toggles (full circle) and the others untoggle (empty circle).		
22.2		Click the currently toggled font radio button.	Nothing.		
23	Tool tip selection.	Click the tooltip radio button twice.	The radio button toggles (gives a full circle) then untoggles.		
24.1	Click the save button.	Change any input so the save and cancel buttons appear. Ensure there is valid data. Click the save button.	The appearance information is saved to the database.		
24.2		Change any input so the save and cancel buttons appear. Ensure there is invalid	The appearance information is not saved to the database and a message box appears indicating the error.		

		data. Click the save button.			
25	Click the cancel button	Change any input. Click the cancel button.	The emergency contact information is all reset.		
26	Tab order	Press the tab key repeatedly.	The controls change focus in a sensible manner.		

Account Security Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
27.1	Password visibility.	Type some text into the password input and click the show password icon.	The password becomes visible in plain text.		
27.2		Type some text into the new password input and click the show password icon.	The new password becomes visible in plain text.		
28.1	Change password.	Click the change password button with a valid new password.	The password is hashed, salted and updated on the database.		
28.2		Click the Change Password button with an invalid new password.	The password is not updated, and an error message is displayed.		
29	Tab order	Press the tab key repeatedly.	The controls change focus in a sensible manner.		
30	Scrolling.	Use the mouse wheel and the	The whole page scrolls.		

		scroll bar on the side.			
Manage Stock Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
31	Data grid view scrolling.	Use the scroll bar and the scroll wheel on the data grid view.	The items in the data grid view scroll.		
32.1	Sort the columns.	Click the ID column header. (left mouse button is ascending and right mouse button is descending).	The stock items are sorted in numerical order of ID.		
32.2		Click the Name column header. (left mouse button is ascending and right mouse button is descending).	The stock items are sorted in alphabetical order by name.		
32.3		Click the SKU column header. (left mouse button is ascending and right mouse button is descending).	The stock items are sorted alphabetically by SKU.		
32.4		Click the Quantity column header. (left mouse button is ascending and right mouse button is descending).	The stock items are sorted by quantity.		

32.5		Click the Quantity Level column header. (left mouse button is ascending and right mouse button is descending).	The stock items are sorted by quantity level ("Low", "Medium", "High").		
32.6		Click the Archived column header. (left mouse button is ascending and right mouse button is descending).	The stock items are sorted by archived status.		
33	Add stock.	Click the add stock button.	The add stock view opens.		
34.1	Edit stock.	Click the edit stock button.	The edit stock view opens for the selected stock item.		
34.2		Double-click a stock item.	The edit stock view opens for the selected stock item.		
35	Archived stock.	Click the archive stock button.	The selected stock item becomes archived.		
36	Show archived stock.	Click the show archived stock button.	The button changes to show the archived state. The data grid view displays archived items.		

Add Stock View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
37	Initial sub view.	Open the add view.	The initial view is displayed. (Stock Details View).		

38	Back button navigates back.	Click the back button in the top left corner.	Navigates to the stock display view.		
39	View titles.	Any child view is displayed.	The corresponding heading appears above it.		
40.1	Navigation between subviews.	Click the forward button with valid data.	Navigates to the next subview.		
40.2		Click the forward button with invalid data.	Does not navigate to the next subview.		
40.3		Click the back button.	Navigates to previous subview.		
40.4		Click the done button.	If the data is valid, the new stock item is inserted into the database and navigates back to the display stock view.		

Edit Stock View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
41.1	Saving and cancelling.	Change any bit of data within any subview.	The save and cancel button appear.		
41.2		Change data within each subview to valid data and click save.	The data is saved to the database.		
41.3		Click cancel after changing some data.	The default values from the database are populated.		
42.1	Navigation.	Click the back button with the default data.	Navigate back.		

42.2		Click the back button with the different data than the default.	The confirmation message box appears to allow navigation.		
42.3		Click the items in the top menu.	Navigate to the clicked item.		

Stock Details View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
43	Archiving.	Click the archive radio button twice.	The radio button toggles (gives a full circle) then untoggles.		

Stock Quantity View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
44.1	Bulk add/remove.	Click the bulk add button after inserting several values to be bulk added, e.g. 10, 100, 1000.	The corresponding quantity is added to the current stock level.		
44.2		After inserting several values to be removed, e.g., 10, 100, 1000, click the bulk remove button.	The corresponding quantity is removed from the current stock level.		
44.3		Click the bulk remove button with a quantity greater than the current quantity.	The corresponding quantity is reduced to 0.		

Manage Staff Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
---------	------------------	------------	------------------	----------------	------

45	Data grid view scrolling.	Use the scroll bar and the scroll wheel on the data grid view.	The items in the data grid view scroll.		
46.1	Sort the columns.	Click the ID column header. (left mouse button is ascending and right mouse button is descending).	The staff members are sorted in numerical order of ID.		
46.2		Click the Username column header. (left mouse button is ascending and right mouse button is descending).	The staff are sorted in alphabetical order by username.		
46.3		Click the Name column header. (left mouse button is ascending and right mouse button is descending).	The staff are sorted alphabetically by name.		
46.4		Click the Email column header. (left mouse button is ascending and right mouse button is descending).	The staff are sorted by email.		
46.5		Click the Phone Number column header. (left mouse button is ascending and right mouse	The staff are sorted by phone number.		

		button is descending).			
46.6		Click the Archived column header. (left mouse button is ascending and right mouse button is descending).	The staff are sorted by archived status.		
47	Add staff.	Click the add button.	The add staff view opens.		
48.1	Edit staff.	Click the edit button.	The edit staff view opens for the selected staff item.		
48.2		Double-click a staff item.	The edit staff view opens for the selected staff item.		
49	Archived staff.	Click the archive staff button.	The selected staff item becomes archived.		
50	Show archived staff.	Click the show archived staff button.	The button changes to show the archived state. The data grid view displays archived items.		

Add Staff View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
51	Initial sub view.	Open the add view.	The initial view is displayed. (Staff Details View)		
52	Back button navigates back.	Click the back button in the top left corner.	Navigates to the staff display view.		
53.1	Navigation between subviews.	Click the forward button with valid data.	Navigates to the next subview.		

53.2		Click the forward button with invalid data.	Does not navigate to the next subview.		
53.3		Click the back button.	Navigates to previous subview.		
53.4		Click the done button.	If the data is valid, the new staff item is inserted into the database and navigated back to the display staff view.		

Edit Staff View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
54.1	Saving and cancelling.	Change any bit of data within any subview.	The save and cancel button appear.		
54.2		Change data within each subview to valid data and click save.	The data is saved to the database.		
54.3		Click cancel after changing some data.	The default values from the database are populated.		
55.1	Navigation.	Click the back button with the default data.	Navigate back.		
55.2		Click the back button with the different data than the default.	The confirmation message box appears to allow navigation.		
55.3		Click the items in the top menu.	Navigate to the clicked item.		

Manage Order Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
56	Data grid view scrolling.	Use the scroll bar and the scroll wheel on the data grid view.	The items in the data grid view scroll.		
57.1	Sort the columns.	Click the ID column header. (left mouse button is ascending and right mouse button is descending).	The orders are sorted in numerical order of ID.		
57.2		Click the Staff column header. (left mouse button is ascending and right mouse button is descending).	The orders are sorted in alphabetical order by staff name.		
57.3		Click the Status column header. (left mouse button is ascending and right mouse button is descending).	The orders are sorted alphabetically by status.		
58	Add order.	Click the add button.	The add order view opens.		
59.1	Edit order.	Click the edit button.	The edit orderview opens for the selected order item.		
59.2		Double-click an order item.	The edit order view opens for the selected order item.		
60.1	Delete an order.	Click the delete order button with a drafted order selected.	The order is deleted.		

60.2		Click the delete order button with a drafted order not selected.	Nothing.		
------	--	--	----------	--	--

Add Order View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
61	Initial sub view.	Open the add view.	The initial view is displayed. (Order Details View)		
62	Back button navigates back.	Click the back button in the top left corner.	Navigates to the stock display view.		
63.1	Navigation between subviews.	Click the forward button with valid data.	Navigates to the next subview.		
63.2		Click the forward button with invalid data.	Does not navigate to the next subview.		
63.3		Click the back button.	Navigates to previous subview.		
63.4		Click the done button.	If the data is valid, the new order item is inserted into the database and navigates back to the display order view.		

Edit Order View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
64.1	Saving and cancelling.	Change any bit of data within any subview.	The save and cancel button appear.		
64.2		Change data within each subview to valid	The data is saved to the database.		

		data and click save.			
64.3		Click cancel after changing some data.	The default values from the database are populated.		
65.1	Navigation.	Click the back button with the default data.	Navigate back.		
65.2		Click the back button with the different data than the default.	The confirmation message box appears to allow navigation.		
65.3		Click the items in the top menu.	Navigate to the clicked item.		

Manage Cleaning Job Option Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
66	Data grid view scrolling.	Use the scroll bar and the scroll wheel on the data grid view.	The items in the data grid view scroll.		
67.1	Sort the columns.	Click the ID column header. (left mouse button is ascending and right mouse button is descending).	The options are sorted in numerical order of ID.		
67.2		Click the Name column header. (left mouse button is ascending and right mouse button is descending).	The options are sorted in alphabetical order by staff name.		

67.3		Click the Status column header. (left mouse button is ascending and right mouse button is descending).	The cleaning job options are sorted alphabetically by status.		
67.3		Click the Status column header. (left mouse button is ascending and right mouse button is descending).	The cleaning job options are sorted alphabetically by status.		
67.2		Double-click a cleaning job option item.	The edit cleaning job option view opens for the selected order item.		
68	Archived cleaning job option.	Click the archive cleaning job option button.	The selected cleaning job option item becomes archived.		
69	Show archived cleaning job option.	Click the show archived cleaning job options button.	The button changes to show the archived state. The data grid view displays archived items.		

Add Cleaning Job Option View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
70	Initial sub view.	Open the add view.	The initial view is displayed. (Cleaning Job Option Details View)		
71	Back button navigates back.	Click the back button in the top left corner.	Navigates to the stock display view.		
72.1	Navigation between subviews.	Click the forward button with valid data.	Navigates to the next subview.		

72.2		Click the forward button with invalid data.	Does not navigate to the next subview.		
72.3		Click the back button.	Navigates to previous subview.		
72.4		Click the done button.	If the data is valid, the new stock item is inserted into the database and navigates back to the display stock view.		

Book Cleaning Job Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
73	Data grid view scrolling.	Use the scroll bar and the scroll wheel on the data grid view.	The items in the data grid view scroll.		
74.1	Date	Select a date with the calendar.	Jobs for that date are populated in the table.		
74.2		Use the date navigation buttons.	The month changes by going forward or back. Jobs for that date are populated in the table.		
75.1	Sort the columns.	Click the ID column header. (left mouse button is ascending and right mouse button is descending).	The jobs are sorted in numerical order of ID.		
75.2		Click the Address column header. (left mouse button is ascending and	The cleaning jobs are sorted in alphabetical order by staff name.		

		right mouse button is descending).			
75.3		Click the Status column header. (left mouse button is ascending and right mouse button is descending).	The jobs are sorted alphabetically by status.		
76	Add booking.	Click the add button. (with a job less than two weeks in advance and a job more than two weeks in advance).	The add cleaning job view opens if the cleaning job date is less than two weeks in advance.		
78.1	Edit cleaning job.	Click the edit button.	The edit cleaning job view opens for the selected order item.		
78.2		Double-click a cleaning job item.	The edit cleaning job view opens for the selected cleaning job item.		
79	Delete a job.	Click the delete cleaning job button.	The selected job is deleted (if it is upcoming).		

Add Cleaning Job View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
80	Initial sub view.	Open the add view.	The initial view is displayed. (Cleaning Job Details View)		
81	Back button navigates back.	Click the back button in the top left corner.	Navigates to the stock display view.		
82.1	Navigation between subviews.	Click the forward button with valid data.	Navigates to the next subview.		

82.2		Click the forward button with invalid data.	Does not navigate to the next subview.		
82.3		Click the back button.	Navigates to previous subview.		
82.4		Click the done button.	If the data is valid, the new cleaning job is inserted into the database and navigates back to the display cleaning job view.		

Edit Cleaning Job View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
83.1	Saving and cancelling.	Change any bit of data within any subview.	The save and cancel button appear.		
83.2		Change data within each subview to valid data and click save.	The data is saved to the database.		
83.3		Click cancel after changing some data.	The default values from the database are populated.		
84.1	Navigation.	Click the back button with the default data.	Navigate back.		
84.2		Click the back button with the different data than the default.	The confirmation message box appears to allow navigation.		
84.3		Click the items in the top menu.	Navigate to the clicked item.		

Manage Customer Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
85	Data grid view scrolling.	Use the scroll bar and the scroll wheel on the data grid view.	The items in the data grid view scroll.		
86.1	Sort the columns.	Click the ID column header. (left mouse button is ascending and right mouse button is descending).	The customers are sorted in numerical order of ID.		
87.2		Click the Name column header. (left mouse button is ascending and right mouse button is descending).	The customers are sorted in alphabetical order by customer name.		
87.3		Click the Status column header. (left mouse button is ascending and right mouse button is descending).	The customers are sorted alphabetically by status.		
87.3		Click the Status column header. (left mouse button is ascending and right mouse button is descending).	The customers are sorted alphabetically by status.		
87.2		Double-click a customer item.	The edit customer view opens for the selected customer item.		

88	Archived customer.	Click the archive customer button.	The selected customer item becomes archived.		
89	Show archived customer.	Click the show archived customers button.	The button changes to show the archived state. The data grid view displays archived items.		

Add Customer View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
90	Initial sub view.	Open the add view.	The initial view is displayed. (Customer Details View)		
91	Back button navigates back.	Click the back button in the top left corner.	Navigates to the stock display view.		
92.1	Navigation between subviews.	Click the forward button with valid data.	Navigates to the next subview.		
92.2		Click the forward button with invalid data.	Does not navigate to the next subview.		
92.3		Click the back button.	Navigates to previous subview.		
92.4		Click the done button.	If the data is valid, the new stock item is inserted into the database and navigates back to the display customer view.		

Edit Customer View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
---------	------------------	------------	------------------	----------------	------

93.1	Saving and cancelling.	Change any bit of data within any subview.	The save and cancel button appear.		
93.2		Change data within each subview to valid data and click save.	The data is saved to the database.		
93.3		Click cancel after changing some data.	The default values from the database are populated.		
94.1	Navigation.	Click the back button with the default data.	Navigate back.		
94.2		Click the back button with the different data than the default.	The confirmation message box appears to allow navigation.		
94.3		Click the items in the top menu.	Navigate to the clicked item.		

Testing: Data Entry					
Personnel:		Matthew Gracey			
Environment:		Windows 11 Laptop (Intel i7, 16GB RAM, 1TB SSD)			
Sign In Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
1	Username.	256 characters.	The last character cannot be typed.		
2	Password.	256 characters.	The last character cannot be typed.		
Personal Information Settings Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
3.1	Name.	Nothing.	Error displayed.		
3.2		Forename: “John” Surname: “Doe”	No error displayed.		
3.3		Forename: “sy192yr” Surname: “1283he”	No error displayed		
4.1	Date of birth.	Nothing	No error displayed.		
4.2		Partial input: Day: 01 Month: 06	The error message “Invalid date of birth” is displayed.		
4.3		Valid date: Day: 01 Month: 02 Year: 2005	No error displayed.		
4.4		Erroneous date: Day: 32 Month: 31 Year: 1000	The error message “Invalid date of birth” is displayed.		

Staff Contact Information Settings Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
5.1	Email.	Nothing.	Error message displayed.		
5.2		Erroneous: "sjkadhaksjhd"	Error message displayed.		
5.3		Normal: "test@example.com"	No error message is displayed.		
6.1	Phone number.	Nothing.	Error message displayed.		
6.2		Erroneous: "sjkadhaksjhd"	No text is inputted.		
6.3		Incorrect: "1234"	Error message displayed.		
6.4		Correct: "07863427265"	No error message is displayed.		
6.5		Length: "098765432123456789"	Limited to 20 characters.		
7	Address.	Length: Type 256 characters.	Limited to 255 characters.		

Emergency Contact Information Settings Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
8	Forename	Length: Type 256 characters.	Limited to 255 characters.		
9	Surname	Length: Type 256 characters.	Limited to 255 characters.		
10.1	Phone number.	Nothing.	Error message displayed.		
10.2		Erroneous:	No text is inputted.		

		"sjkadhsjhd"			
10.3		Incorrect: "1234"	Error message displayed.		
10.4		Correct: "07863427265"	No error message is displayed.		
10.5		Length: "098765432123 456789"	Limited to 20 characters.		

Security Settings Page & Change Staff Password Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
11	New password	Password: "Password123!"	As the password is typed in, the password requirements change from a cross to a tick.		

Stock Details Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
12.1	Name	Nothing.	Error displayed.		
12.2		Length: Type 256 characters.	Limited to 255 characters.		
13.1	SKU.	Duplicate SKU.	Error displayed.		
13.2		Nothing.	Error displayed.		
14	Description	Length: Type 1001 characters.	Limited to 1000 characters.		

Stock Quantity Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
15	Reason for quantity change.	Length: Type 1001 characters.	Limited to 1000 characters.		

Order Quantity Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
16	Quantity.	Select a stock item and change the quantity to below zero.	The quantity cannot be put below zero.		
Order Details Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
17	Description.	Type 256 characters in the description.	Limited to 255 characters.		
Order Discrepancies Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
18	Description.	Type 256 characters in the description.	Limited to 255 characters.		
Customer Details Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
19.1	Name.	Nothing.	Error displayed.		
19.2		Forename: "John" Surname: "Doe"	No error displayed.		
19.3		Forename: "sy192yr" Surname: "1283he"	No error displayed		
Customer Contact Information Settings Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
20.1	Email.	Nothing.	Error message displayed.		

20.2		Erroneous: “sjkadhaksjhd”	Error message displayed.		
20.3		Normal: “test@example.com”	No error message is displayed.		
21.1	Phone number.	Nothing.	Error message displayed.		
21.2		Erroneous: “sjkadhaksjhd”	No text is inputted.		
21.3		Incorrect: “1234”	Error message displayed.		
21.4		Correct: “07863427265”	No error message is displayed.		
21.5		Length: “098765432123 456789”	Limited to 20 characters.		
22	Address.	Length: Type 256 characters.	Limited to 255 characters.		

Cleaning Job Options Details Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
23.1	Name	Nothing.	Error displayed.		
23.2		Length: Type 256 characters.	Limited to 255 characters.		
24	Description	Length: Type 1001 characters.	Limited to 255 characters.		

Cleaning Job Details Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
25.1	Address	Length: Type 501 characters.	Limited to 500 characters.		

25.2		Clear the text box after inputting something or try to click Next when it is empty.	Error message appears		
26	Extra Information	Length: Type 501 characters.	Limited to 500 characters.		

Cleaning Job Date and Time Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
27.1	Date	Nothing	An error message is displayed.		
27.2		Partial input: Day: 01 Month: 06	The error message “Invalid date of birth” is displayed.		
27.3		Valid date: Day: 01 Month: 02 Year: 2005	No error displayed.		
27.4		Erroneous date: Day: 32 Month: 31 Year: 1000	The error message “Invalid date of birth” is displayed.		
28.1	Time	Nothing.	An error message is displayed.		
28.2		Erroneous: 25:76	An error message is displayed.		
28.3		Start time after end time.	The error message “Ensure the end time is after the start time” appears.		

Testing

Testing: Link Testing					
Personnel:		Matthew Gracey			
Environment:		Windows 11 Laptop (Intel i7, 16GB RAM, 1TB SSD)			
Splash Screen					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
1	The splash screen appears.	App start.	The splash screen is visible and centered on the screen.	The splash screen is visible and centered on the screen.	✓
2	Splash screen timeout.	App start.	The splash screen disappears after 1 second (approx.).	The splash screen disappears after 1 second.	✓
3	Splash screen transition.	App start.	Sign-in page opens	Transition to sign-in page	✓
Sign In Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
4	A successful sign in directs to the main menu.	Username: "jdoe1" Password: "password123!"	Sign-in page disappears and main page opens	Sign-in page disappears, and main page opens	✓
5	Users can toggle the show password.	Click the show password icon. Password: "password"	Password character is removed, and password is visible in plain text.	The password character is removed, and the password is visible in plain text.	✓
6	Switch theme in the login page	Click the switch theme button.	The view changes from one theme to the other, and the icon changes from a sun (dark mode) to a moon (light mode).	The view changes from one theme to the other, and the icon changes from a sun (dark mode) to a moon (light mode).	✓
7	Tab order	Press the tab key repeatedly.	The controls change focus in a sensible manner.	The controls change focus in a sensible manner.	✓

Main Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
8	Sign out works.	Click the sign-out button at the top right.	The application reloads.	The application reloads.	✓
9	Menu tabs work.	Click the menu tabs.	The tabs open downward to expose the menu items. The icon changes from a plus to a minus.	The tabs open downward to expose the menu items. The icon changes from a plus to a minus.	✓
10.1	Correct menu items display	Sign in to the application with a cleaning staff account.	The menu items are: Dashboard Cleaning (Upcoming jobs) Settings (Personal information, Contact details, Emergency contact, Account security, Appearance)	The menu items are: Dashboard Cleaning (Upcoming jobs) Settings (Personal information, Contact details, Emergency contact, Account security, Appearance)	✓
		Sign in to the application with an office staff account.	The menu items are: Dashboard Cleaning (Book cleaning, Upcoming jobs, Manage customers, Manage options) Stock (Orders) Settings (Personal information, Contact details, Emergency contact, Account security, Appearance)	The menu items are: Dashboard Cleaning (Book cleaning, Upcoming jobs, Manage customers, Manage options) Stock (Orders) Settings (Personal information, Contact details, Emergency contact, Account security, Appearance)	✓
10.3		Sign in to the application with a cleaning	The menu items are: Dashboard	The menu items are different: Dashboard	□

		<p>manager account.</p> <p>Stock (Manage stock, Order stock, Quantity changes, Upcoming deliveries)</p> <p>Reports (Stock)</p> <p>Settings (Personal information, Contact details, Emergency contact, Account security, Appearance)</p>	<p>Cleaning (Book cleaning, Upcoming jobs, Manage customers, Manage options)</p> <p>Stock (Manage stock, Order stock, Quantity changes, Upcoming deliveries)</p> <p>Reports (Stock)</p> <p>Settings (Personal information, Contact details, Emergency contact, Account security, Appearance)</p>	
10.4		<p>Sign in to the application with an administrator account.</p> <p>The menu items are:</p> <p>Dashboard</p> <p>Security (Manage staff, Login attempts, Change password)</p> <p>Reports (Staff)</p> <p>Settings (Personal information, Contact details, Emergency contact, Account security, Appearance)</p>	<p>The menu items are:</p> <p>Dashboard</p> <p>Security (Manage staff, Login attempts, Change password)</p> <p>Reports (Staff)</p> <p>Settings (Personal information, Contact details, Emergency contact, Account security, Appearance)</p>	✓
10.5		<p>Sign in to the application with a manager account.</p> <p>The menu items are:</p> <p>Dashboard</p> <p>Stock (Manage stock, Orders, Quantity changes)</p> <p>Cleaning (Book cleaning, Upcoming jobs, Manage customers, Manage options)</p> <p>Reports (Stock, Staff)</p>	<p>The menu items are:</p> <p>Dashboard</p> <p>Stock (Manage stock, Orders, Quantity changes)</p> <p>Cleaning (Book cleaning, Upcoming jobs, Manage customers, Manage options)</p> <p>Reports (Stock, Staff)</p>	✓

			Settings (Personal information, Contact details, Emergency contact, Account security, Appearance)	Settings (Personal information, Contact details, Emergency contact, Account security, Appearance)	
Dashboard Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
11	Check the change password button redirects to the security settings page.	Sign in to a staff account with a password that is older than 30 days. Click the Change Password button.	Redirected to the security settings page.	Nothing happens.	<input type="checkbox"/>
Personal Information Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
12.1	Click the save button.	Change any input so the save and cancel buttons appear. Ensure there is valid data. Click the save button.	Personal information is saved to the database.	Personal information is saved to the database.	<input checked="" type="checkbox"/>
12.2		Change any input so the save and cancel buttons appear. Ensure there is invalid data. Click the save button.	The personal information is not saved to the database and a message box appears indicating the error.	The personal information is not saved to the database, and a message box appears indicating the error.	<input checked="" type="checkbox"/>
13	Click the cancel button	Change any input. Click the cancel button.	The personal information all resets.	The personal information all resets.	<input checked="" type="checkbox"/>

14	Tab order	Press the tab key repeatedly.	The controls change focus in a sensible manner.	The controls change focus in a sensible manner.	<input checked="" type="checkbox"/>
----	-----------	-------------------------------	---	---	-------------------------------------

Contact Information Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
15.1	Click the save button. Change any input so the save and cancel buttons appear. Ensure there is valid data. Click the save button.	Change any input so the save and cancel buttons appear. Ensure there is valid data. Click the save button.	The contact information is saved to the database.	The contact information is saved to the database.	<input checked="" type="checkbox"/>
15.2		Change any input so the save and cancel buttons appear. Ensure there is invalid data. Click the save button.	The contact information is not saved to the database, and a message box appears indicating the error.	The contact information is not saved to the database, and a message box appears indicating the error.	<input checked="" type="checkbox"/>
16	Click the cancel button	Change any input. Click the cancel button.	The contact information all resets.	The contact information has all reset.	<input checked="" type="checkbox"/>
17	Tab order	Press the tab key repeatedly.	The controls change focus in a sensible manner.	The controls change focus in a sensible manner.	<input checked="" type="checkbox"/>

Emergency Contact Information Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
18.1	Click the save button. Change any input so the save and cancel buttons appear. Ensure there is valid data. Click the save button.	Change any input so the save and cancel buttons appear. Ensure there is valid data. Click the save button.	The emergency contact information is saved to the database.	The emergency contact information is saved to the database.	<input checked="" type="checkbox"/>
18.2		Change any input so the	The emergency contact information is	The emergency contact information is not saved	<input checked="" type="checkbox"/>

		save and cancel buttons appear. Ensure there is invalid data. Click the save button.	not saved to the database, and a message box appears indicating the error.	to the database, and a message box indicates the error.	
19	Click the cancel button	Change any input. Click the cancel button.	The emergency contact information all resets.	The emergency contact information all resets.	✓
20	Tab order	Press the tab key repeatedly.	The controls change focus in a sensible manner.	The controls change focus in a sensible manner.	✓

Appearance Settings Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
21	Appearance settings toggle.	Click the appearance settings radio button twice.	The radio button toggles (gives a full circle) then untoggles.	The radio button toggles (gives a full circle) then untoggles.	✓
22.1	Font selection.	Click each font radio button.	The clicked font radio button toggles (full circle) and the others untoggle (empty circle).	The clicked font radio button toggles (full circle) and the others untoggle (empty circle).	✓
22.2		Click the currently toggled font radio button.	Nothing.	Nothing.	✓
23	Tool tip selection.	Click the tooltip radio button twice.	The radio button toggles (gives a full circle) then untoggles.	The radio button toggles (gives a full circle) then untoggles.	✓
24.1	Click the save button.	Change any input so the save and cancel buttons appear. Ensure there is valid data. Click the save button.	The appearance information is saved to the database.	The appearance information is saved to the database.	✓
24.2		Change any input so the	The appearance information is not	The appearance information is not saved	✓

		save and cancel buttons appear. Ensure there is invalid data. Click the save button.	saved to the database and a message box appears indicating the error.	to the database and a message box appears indicating the error.	
25	Click the cancel button	Change any input. Click the cancel button.	The emergency contact information is all reset.	The emergency contact information has all reset.	✓
26	Tab order	Press the tab key repeatedly.	The controls change focus in a sensible manner.	The controls change focus in a sensible manner.	✓

Account Security Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
27.1	Password visibility.	Type some text into the password input and click the show password icon.	The password becomes visible in plain text.	The password becomes visible in plain text	✓
27.2		Type some text into the new password input and click the show password icon.	The new password becomes visible in plain text.	The new password becomes visible in plain text.	✓
28.1	Change password.	Click the change password button with a valid new password.	The password is hashed, salted and updated on the database.	The password is hashed, salted and updated on the database.	✓
28.2		Click the Change Password button with an invalid new password.	The password is not updated, and an error message is displayed.	The password is not updated, and an error message is displayed.	✓
29	Tab order	Press the tab key repeatedly.	The controls change focus in a sensible manner.	The controls change focus in a sensible manner.	✓

30	Scrolling.	Use the mouse wheel and the scroll bar on the side.	The whole page scrolls.	The whole page scrolls.	<input checked="" type="checkbox"/>
----	------------	---	-------------------------	-------------------------	-------------------------------------

Manage Stock Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
31	Data grid view scrolling.	Use the scroll bar and the scroll wheel on the data grid view.	The items in the data grid view scroll.	The items in the data grid view scroll.	<input checked="" type="checkbox"/>
32.1	Sort the columns.	Click the ID column header. (left mouse button is ascending and right mouse button is descending).	The stock items are sorted in numerical order of ID.	An exception occurs.	<input type="checkbox"/>
32.2		Click the Name column header. (left mouse button is ascending and right mouse button is descending).	The stock items are sorted in alphabetical order by name.	The stock items are sorted in alphabetical order by name.	<input checked="" type="checkbox"/>
32.3		Click the SKU column header. (left mouse button is ascending and right mouse button is descending).	The stock items are sorted alphabetically by SKU.	The stock items are sorted alphabetically by SKU.	<input checked="" type="checkbox"/>
32.4		Click the Quantity column header. (left mouse button is ascending and right mouse	The stock items are sorted by quantity.	The stock items are sorted by quantity.	<input checked="" type="checkbox"/>

		button is descending).			
32.5		Click the Quantity Level column header. (left mouse button is ascending and right mouse button is descending).	The stock items are sorted by quantity level ("Low", "Medium", "High").	The stock items are sorted by quantity level ("Low", "Medium", "High").	✓
32.6		Click the Archived column header. (left mouse button is ascending and right mouse button is descending).	The stock items are sorted by archived status.	The stock items are sorted by archived status.	✓
33	Add stock.	Click the add stock button.	The add stock view opens.	The add stock view opens.	✓
34.1	Edit stock.	Click the edit stock button.	The edit stock view opens for the selected stock item.	The edit stock view opens for the selected stock item.	✓
34.2		Double-click a stock item.	The edit stock view opens for the selected stock item.	The edit stock view opens for the selected stock item.	✓
35	Archived stock.	Click the archive stock button.	The selected stock item becomes archived.	The selected stock item becomes archived.	✓
36	Show archived stock.	Click the show archived stock button.	The button changes to show the archived state. The data grid view displays archived items.	The button changes to show the archived state. The data grid view displays archived items.	✓

Add Stock View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass

37	Initial sub view.	Open the add view.	The initial view is displayed. (Stock Details View).	The initial view is displayed. (Stock Details View).	✓
38	Back button navigates back.	Click the back button in the top left corner.	Navigates to the stock display view.	Navigates to the stock display view.	✓
39	View titles.	Any child view is displayed.	The corresponding heading appears above it.	The corresponding heading appears above it.	✓
40.1	Navigation between subviews.	Click the forward button with valid data.	Navigates to the next subview.	Navigates to the next subview.	✓
40.2		Click the forward button with invalid data.	Does not navigate to the next subview.	Does not navigate to the next subview.	✓
40.3		Click the back button.	Navigates to previous subview.	Navigates to previous subview.	✓
40.4		Click the done button.	If the data is valid, the new stock item is inserted into the database and navigates back to the display stock view.	If the data is valid, the new stock item is inserted into the database and navigates back to the display stock view.	✓

Edit Stock View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
41.1	Saving and cancelling.	Change any bit of data within any subview.	The save and cancel button appear.	The save and cancel button appear.	✓
41.2		Change data within each subview to valid data and click save.	The data is saved to the database.	The data is saved to the database.	✓
41.3		Click cancel after changing some data.	The default values from the database are populated.	The default values from the database are populated.	✓

42.1	Navigation.	Click the back button with the default data.	Navigate back.	Navigate back.	✓
42.2		Click the back button with the different data than the default.	The confirmation message box appears to allow navigation.	The confirmation message box appears to allow navigation.	✓
42.3		Click the items in the top menu.	Navigate to the clicked item.	Navigate to the clicked item.	✓

Stock Details View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
43	Archiving.	Click the archive radio button twice.	The radio button toggles (gives a full circle) then untoggles.	The radio button toggles (gives a full circle) then untoggles.	✓

Stock Quantity View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
44.1	Bulk add/remove.	Click the bulk add button after inserting several values to be bulk added, e.g. 10, 100, 1000.	The corresponding quantity is added to the current stock level.	The corresponding quantity is added to the current stock level.	✓
44.2		After inserting several values to be removed, e.g., 10, 100, 1000, click the bulk remove button.	The corresponding quantity is removed from the current stock level.	The corresponding quantity is removed from the current stock level.	✓
44.3		Click the bulk remove button with a quantity greater than the current quantity.	The corresponding quantity is reduced to 0.	The corresponding quantity is reduced to 0.	✓

Manage Staff Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
45	Data grid view scrolling.	Use the scroll bar and the scroll wheel on the data grid view.	The items in the data grid view scroll.	The items in the data grid view scroll.	✓
46.1	Sort the columns.	Click the ID column header. (left mouse button is ascending and right mouse button is descending).	The staff members are sorted in numerical order of ID.	The staff members are sorted in numerical order of ID.	✓
46.2		Click the Username column header. (left mouse button is ascending and right mouse button is descending).	The staff are sorted in alphabetical order by username.	The staff are sorted in alphabetical order by username.	✓
46.3		Click the Name column header. (left mouse button is ascending and right mouse button is descending).	The staff are sorted alphabetically by name.	The staff are sorted alphabetically by name.	✓
46.4		Click the Email column header. (left mouse button is ascending and right mouse button is descending).	The staff are sorted by email.	The staff are sorted by email.	✓
46.5		Click the Phone Number column	The staff are sorted by phone number.	The staff are sorted by phone number.	✓

		header. (left mouse button is ascending and right mouse button is descending).			
46.6		Click the Archived column header. (left mouse button is ascending and right mouse button is descending).	The staff are sorted by archived status.	The staff are sorted by archived status.	✓
47	Add staff.	Click the add button.	The add staff view opens.	The add staff view opens.	✓
48.1	Edit staff.	Click the edit button.	The edit staff view opens for the selected staff item.	The edit staff view opens for the selected staff item.	✓
48.2		Double-click a staff item.	The edit staff view opens for the selected staff item.	The edit staff view opens for the selected staff item.	✓
49	Archived staff.	Click the archive staff button.	The selected staff item becomes archived.	The selected staff item becomes archived.	✓
50	Show archived staff.	Click the show archived staff button.	The button changes to show the archived state. The data grid view displays archived items.	The button changes to show the archived state. The data grid view displays archived items.	✓

Add Staff View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
51	Initial sub view.	Open the add view.	The initial view is displayed. (Staff Details View)	The initial view is displayed. (Staff Details View)	✓
52	Back button navigates back.	Click the back button in the top left corner.	Navigates to the staff display view.	Navigates to the staff display view.	✓

53.1	Navigation between subviews.	Click the forward button with valid data.	Navigates to the next subview.	Navigates to the next subview.	✓
53.2		Click the forward button with invalid data.	Does not navigate to the next subview.	Does not navigate to the next subview.	✓
53.3		Click the back button.	Navigates to previous subview.	Navigates to previous subview.	✓
53.4		Click the done button.	If the data is valid, the new staff item is inserted into the database and navigated back to the display staff view.	The data is not inserted. An error message box says saving changes failed.	□

Edit Staff View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
54.1	Saving and cancelling.	Change any bit of data within any subview.	The save and cancel button appear.	The save and cancel button appear.	✓
54.2		Change data within each subview to valid data and click save.	The data is saved to the database.	The data is saved to the database.	✓
54.3		Click cancel after changing some data.	The default values from the database are populated.	The default values from the database are populated.	✓
55.1	Navigation.	Click the back button with the default data.	Navigate back.	Navigate back.	✓
55.2		Click the back button with the different data than the default.	The confirmation message box appears to allow navigation.	The confirmation message box appears to allow navigation.	✓
55.3		Click the items in the top menu.	Navigate to the clicked item.	Navigate to the clicked item.	✓

Manage Order Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
56	Data grid view scrolling.	Use the scroll bar and the scroll wheel on the data grid view.	The items in the data grid view scroll.	The items in the data grid view scroll.	✓
57.1	Sort the columns.	Click the ID column header. (left mouse button is ascending and right mouse button is descending).	The orders are sorted in numerical order of ID.	The orders are sorted in numerical order of ID.	✓
57.2		Click the Staff column header. (left mouse button is ascending and right mouse button is descending).	The orders are sorted in alphabetical order by staff name.	The orders are sorted in alphabetical order by staff name.	✓
57.3		Click the Status column header. (left mouse button is ascending and right mouse button is descending).	The orders are sorted alphabetically by status.	The orders are sorted alphabetically by status.	✓
58	Add order.	Click the add button.	The add order view opens.	The add order view opens.	✓
59.1	Edit order.	Click the edit button.	The edit orderview opens for the selected order item.	The edit orderview opens for the selected order item.	✓
59.2		Double-click an order item.	The edit order view opens for the selected order item.	The edit order view opens for the selected order item.	
60.1	Delete an order.	Click the delete order button	The order is deleted.	The order is deleted.	✓

		with a drafted order selected.			
60.2		Click the delete order button with a drafted order not selected.	Nothing.	Nothing.	✓

Add Order View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
61	Initial sub view.	Open the add view.	The initial view is displayed. (Order Details View)	The initial view is displayed. (Order Details View)	✓
62	Back button navigates back.	Click the back button in the top left corner.	Navigates to the stock display view.	Navigates to the stock display view.	✓
63.1	Navigation between subviews.	Click the forward button with valid data.	Navigates to the next subview.	Navigates to the next subview.	✓
63.2		Click the forward button with invalid data.	Does not navigate to the next subview.	Does not navigate to the next subview.	✓
63.3		Click the back button.	Navigates to previous subview.	Navigates to previous subview.	✓
63.4		Click the done button.	If the data is valid, the new order item is inserted into the database and navigates back to the display order view.	If the data is valid, the new order item is inserted into the database and navigates back to the display order view.	✓

Edit Order View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
64.1	Saving and cancelling.	Change any bit of data within any subview.	The save and cancel button appear.	The save and cancel button appear.	✓
64.2		Change data within each	The data is saved to the database.	The data is saved to the database.	✓

		subview to valid data and click save.			
64.3		Click cancel after changing some data.	The default values from the database are populated.	The default values from the database are populated.	✓
65.1	Navigation.	Click the back button with the default data.	Navigate back.	Navigate back.	✓
65.2		Click the back button with the different data than the default.	The confirmation message box appears to allow navigation.	The confirmation message box appears to allow navigation.	✓
65.3		Click the items in the top menu.	Navigate to the clicked item.	Navigate to the clicked item.	✓

Manage Cleaning Job Option Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
66	Data grid view scrolling.	Use the scroll bar and the scroll wheel on the data grid view.	The items in the data grid view scroll.	The items in the data grid view scroll.	✓
67.1	Sort the columns.	Click the ID column header. (left mouse button is ascending and right mouse button is descending).	The options are sorted in numerical order of ID.	The options are sorted in numerical order of ID.	✓
67.2		Click the Name column header. (left mouse button is ascending and right mouse button is descending).	The options are sorted in alphabetical order by staff name.	The options are sorted in alphabetical order by staff name.	✓

67.3		Click the Status column header. (left mouse button is ascending and right mouse button is descending).	The cleaning job options are sorted alphabetically by status.	The cleaning job options are sorted alphabetically by status.	✓
67.3		Click the Status column header. (left mouse button is ascending and right mouse button is descending).	The cleaning job options are sorted alphabetically by status.	The cleaning job options are sorted alphabetically by status.	✓
67.2		Double-click a cleaning job option item.	The edit cleaning job option view opens for the selected cleaning job option item.	The edit cleaning job option view opens for the selected cleaning job option item.	
68	Archived cleaning job option.	Click the archive cleaning job option button.	The selected cleaning job option item becomes archived.	The selected cleaning job option item becomes archived.	✓
69	Show archived cleaning job option.	Click the show archived cleaning job options button.	The button changes to show the archived state. The data grid view displays archived items.	The button changes to show the archived state. The data grid view displays archived items.	✓

Add Cleaning Job Option View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
70	Initial sub view.	Open the add view.	The initial view is displayed. (Cleaning Job Option Details View)	The initial view is displayed. (Cleaning Job Option Details View)	✓
71	Back button navigates back.	Click the back button in the top left corner.	Navigates to the stock display view.	Navigates to the stock display view.	✓
72.1	Navigation between subviews.	Click the forward button with valid data.	Navigates to the next subview.	Navigates to the next subview.	✓

72.2		Click the forward button with invalid data.	Does not navigate to the next subview.	Does not navigate to the next subview.	✓
72.3		Click the back button.	Navigates to previous subview.	Navigates to previous subview.	✓
72.4		Click the done button.	If the data is valid, the new stock item is inserted into the database and navigates back to the display stock view.	If the data is valid, the new stock item is inserted into the database and navigates back to the display stock view.	✓

Book Cleaning Job Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
73	Data grid view scrolling.	Use the scroll bar and the scroll wheel on the data grid view.	The items in the data grid view scroll.	The items in the data grid view scroll.	✓
74.1	Date	Select a date with the calendar.	Jobs for that date are populated in the table.	Jobs for that date are populated in the table.	✓
74.2		Use the date navigation buttons.	The month changes by going forward or back. Jobs for that date are populated in the table.	The month changes by going forward or back. Jobs for that date are populated in the table.	✓
75.1	Sort the columns.	Click the ID column header. (left mouse button is ascending and right mouse button is descending).	The jobs are sorted in numerical order of ID.	The jobs are sorted in numerical order of ID.	✓
75.2		Click the Address column header. (left mouse button is ascending and	The cleaning jobs are sorted in alphabetical order by staff name.	The cleaning jobs are sorted in alphabetical order by staff name.	✓

		right mouse button is descending).			<input checked="" type="checkbox"/>
75.3		Click the Status column header. (left mouse button is ascending and right mouse button is descending).	The jobs are sorted alphabetically by status.	The jobs are sorted alphabetically by status.	<input checked="" type="checkbox"/>
76	Add booking.	Click the add button. (with a job less than two weeks in advance and a job more than two weeks in advance).	The add cleaning job view opens if the cleaning job date is less than two weeks in advance.	The add cleaning job view opens for both.	<input type="checkbox"/>
78.1	Edit cleaning job.	Click the edit button.	The edit cleaning job view opens for the selected cleaning job item.	The edit cleaning job view opens for the selected cleaning job item.	<input checked="" type="checkbox"/>
78.2		Double-click a cleaning job item.	The edit cleaning job view opens for the selected cleaning job item.	The edit cleaning job view opens for the selected cleaning job item.	<input checked="" type="checkbox"/>
79	Delete a job.	Click the delete cleaning job button.	The selected job is deleted (if it is upcoming).	The selected job is deleted (if it is upcoming).	<input checked="" type="checkbox"/>

Add Cleaning Job View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
80	Initial sub view.	Open the add view.	The initial view is displayed. (Cleaning Job Details View)	The initial view is displayed. (Cleaning Job Details View)	<input checked="" type="checkbox"/>
81	Back button navigates back.	Click the back button in the top left corner.	Navigates to the cleaning display view.	Navigates to the stock display view.	<input type="checkbox"/>

82.1	Navigation between subviews.	Click the forward button with valid data.	Navigates to the next subview.	Navigates to the next subview.	✓
82.2		Click the forward button with invalid data.	Does not navigate to the next subview.	Does not navigate to the next subview.	✓
82.3		Click the back button.	Navigates to previous subview.	Navigates to previous subview.	✓
82.4		Click the done button.	If the data is valid, the new cleaning job is inserted into the database and navigates back to the display cleaning job view.	If the data is valid, the new cleaning job is inserted into the database and navigates back to the display cleaning job view.	✓

Edit Cleaning Job View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
83.1	Saving and cancelling.	Change any bit of data within any subview.	The save and cancel button appear.	The save and cancel button appear.	✓
83.2		Change data within each subview to valid data and click save.	The data is saved to the database.	The data is saved to the database.	✓
83.3		Click cancel after changing some data.	The default values from the database are populated.	The default values from the database are populated.	✓
84.1	Navigation.	Click the back button with the default data.	Navigate back.	Navigate back.	✓
84.2		Click the back button with the different data than the default.	The confirmation message box appears to allow navigation.	The confirmation message box appears to allow navigation.	✓

84.3		Click the items in the top menu.	Navigate to the clicked item.	Navigate to the clicked item.	<input checked="" type="checkbox"/>
Manage Customer Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
85	Data grid view scrolling.	Use the scroll bar and the scroll wheel on the data grid view.	The items in the data grid view scroll.	The items in the data grid view scroll.	<input checked="" type="checkbox"/>
86.1	Sort the columns.	Click the ID column header. (left mouse button is ascending and right mouse button is descending).	The customers are sorted in numerical order of ID.	The customers are sorted in numerical order of ID.	<input checked="" type="checkbox"/>
87.2		Click the Name column header. (left mouse button is ascending and right mouse button is descending).	The customers are sorted in alphabetical order by customer name.	The customers are sorted in alphabetical order by customer name.	<input checked="" type="checkbox"/>
87.3		Click the Status column header. (left mouse button is ascending and right mouse button is descending).	The customers are sorted alphabetically by status.	The customers are sorted alphabetically by status.	<input checked="" type="checkbox"/>
87.3		Click the Status column header. (left mouse button is ascending and right mouse button is descending).	The customers are sorted alphabetically by status.	The customers are sorted alphabetically by status.	<input checked="" type="checkbox"/>

87.2		Double-click a customer item.	The edit customer view opens for the selected customer item.	The edit customer view opens for the selected customer item.	✓
88	Archived customer.	Click the archive customer button.	The selected customer item becomes archived.	The selected customer item becomes archived.	✓
89	Show archived customer.	Click the show archived customers button.	The button changes to show the archived state. The data grid view displays archived items.	The button changes to show the archived state. The data grid view displays archived items.	✓

Add Customer View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
90	Initial sub view.	Open the add view.	The initial view is displayed. (Customer Details View)	The initial view is displayed. (Customer Details View)	✓
91	Back button navigates back.	Click the back button in the top left corner.	Navigates to the stock display view.	Navigates to the stock display view.	✓
92.1	Navigation between subviews.	Click the forward button with valid data.	Navigates to the next subview.	Navigates to the next subview.	✓
92.2		Click the forward button with invalid data.	Does not navigate to the next subview.	Does not navigate to the next subview.	✓
92.3		Click the back button.	Navigates to previous subview.	Navigates to previous subview.	✓
92.4		Click the done button.	If the data is valid, the new stock item is inserted into the database and navigates back to the display customer view.	If the data is valid, the new stock item is inserted into the database and navigates back to the display customer view.	✓

Edit Customer View

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
93.1	Saving and cancelling.	Change any bit of data within any subview.	The save and cancel button appear.	The save and cancel button appear.	✓
93.2		Change data within each subview to valid data and click save.	The data is saved to the database.	The data is saved to the database.	✓
93.3		Click cancel after changing some data.	The default values from the database are populated.	The default values from the database are populated.	✓
94.1	Navigation.	Click the back button with the default data.	Navigate back.	Navigate back.	✓
94.2		Click the back button with the different data than the default.	The confirmation message box appears to allow navigation.	The confirmation message box appears to allow navigation.	✓
94.3		Click the items in the top menu.	Navigate to the clicked item.	Navigate to the clicked item.	✓

Testing: Data Entry					
Personnel:		Matthew Gracey			
Environment:		Windows 11 Laptop (Intel i7, 16GB RAM, 1TB SSD)			
Sign In Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
1	Username.	256 characters.	The last character cannot be typed.	The last character cannot be typed.	✓
2	Password.	256 characters.	The last character cannot be typed.	The last character cannot be typed.	✓
Personal Information Settings Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
3.1	Name.	Nothing.	Error displayed.	Error displayed.	✓
3.2		Forename: "John" Surname: "Doe"	No error displayed.	No error displayed.	✓
3.3		Forename: "sy192yr" Surname: "1283he"	No error displayed	No error displayed.	✓
4.1	Date of birth.	Nothing	No error displayed.	No error displayed.	✓
4.2		Partial input: Day: 01 Month: 06	The error message "Invalid date of birth" is displayed.	The error message "Invalid date of birth" is displayed.	✓
4.3		Valid date: Day: 01 Month: 02 Year: 2005	No error displayed.	No error displayed.	✓
4.4		Erroneous date: Day: 32 Month: 31 Year: 1000	The error message "Invalid date of birth" is displayed.	The error message "Invalid date of birth" is displayed.	✓

Staff Contact Information Settings Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
5.1	Email.	Nothing.	Error message displayed.	Error message displayed.	✓
5.2		Erroneous: "sjkadhaksjhd"	Error message displayed.	Error message displayed.	✓
5.3		Normal: "test@example.com"	No error message is displayed.	No error message is displayed.	✓
6.1	Phone number.	Nothing.	Error message displayed.	Error message displayed.	✓
6.2		Erroneous: "sjkadhaksjhd"	No text is inputted.	No text is inputted.	✓
6.3		Incorrect: "1234"	Error message displayed.	Error message displayed.	✓
6.4		Correct: "07863427265"	No error message is displayed.	No error message is displayed.	✓
6.5		Length: "098765432123456789"	Limited to 20 characters.	Limited to 20 characters.	✓
7	Address.	Length: Type 256 characters.	Limited to 255 characters.	Limited to 255 characters.	✓

Emergency Contact Information Settings Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
8	Forename	Length: Type 256 characters.	Limited to 255 characters.	Limited to 255 characters.	✓
9	Surname	Length: Type 256 characters.	Limited to 255 characters.	Limited to 255 characters.	✓
10.1	Phone number.	Nothing.	Error message displayed.	Nothing.	□

10.2		Erroneous: “sjkadhaksjhd”	No text is inputted.	The text is inputted (not a valid phone number).	<input type="checkbox"/>
10.3		Incorrect: “1234”	Error message displayed.	Error message displayed.	<input checked="" type="checkbox"/>
10.4		Correct: “07863427265”	No error message is displayed.	No error message is displayed.	<input checked="" type="checkbox"/>
10.5		Length: “098765432123 456789”	Limited to 20 characters.	Limited to 20 characters.	<input checked="" type="checkbox"/>

Security Settings Page & Change Staff Password Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
11	New password	Password: “Password123!”	As the password is typed in, the password requirements change from a cross to a tick.	As the password is typed in, the password requirements change from a cross to a tick.	<input checked="" type="checkbox"/>

Stock Details Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
12.1	Name	Nothing.	Error displayed.	Error displayed.	<input checked="" type="checkbox"/>
12.2		Length: Type 256 characters.	Limited to 255 characters.	Limited to 255 characters.	<input checked="" type="checkbox"/>
13.1	SKU.	Duplicate SKU.	Error displayed.	Error displayed.	<input checked="" type="checkbox"/>
13.2		Nothing.	Error displayed.	Error displayed.	<input checked="" type="checkbox"/>
14	Description	Length: Type 1001 characters.	Limited to 1000 characters.	Limited to 1000 characters.	<input checked="" type="checkbox"/>

Stock Quantity Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
15	Reason for quantity change.	Length: Type 1001 characters.	Limited to 1000 characters.	Limited to 1000 characters.	<input checked="" type="checkbox"/>

Order Quantity Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
16	Quantity.	Select a stock item and change the quantity to below zero.	The quantity cannot be put below zero.	The quantity cannot be put below zero.	✓
Order Details Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
17	Description.	Type 256 characters in the description.	Limited to 255 characters.	Limited to 255 characters.	✓
Order Discrepancies Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
18	Description.	Type 256 characters in the description.	Limited to 255 characters.	Limited to 255 characters.	✓
Customer Details Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
19.1	Name.	Nothing.	Error displayed.	Error displayed.	✓
19.2		Forename: "John" Surname: "Doe"	No error displayed.	No error displayed.	✓
19.3		Forename: "sy192yr" Surname: "1283he"	No error displayed	No error displayed.	✓
Customer Contact Information Settings Page					
Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass

20.1	Email.	Nothing.	Error message displayed.	Error message displayed.	✓
20.2		Erroneous: "sjkadhaksjhd"	Error message displayed.	Error message displayed.	✓
20.3		Normal: "test@example.com"	No error message is displayed.	No error message is displayed.	✓
21.1	Phone number.	Nothing.	Error message displayed.	Error message displayed.	✓
21.2		Erroneous: "sjkadhaksjhd"	No text is inputted.	The text is inputted.	✓
21.3		Incorrect: "1234"	Error message displayed.	Error message displayed.	✓
21.4		Correct: "07863427265"	No error message is displayed.	No error message is displayed.	✓
21.5		Length: "098765432123 456789"	Limited to 20 characters.	Limited to 20 characters.	✓
22	Address.	Length: Type 256 characters.	Limited to 255 characters.	Limited to 255 characters.	✓

Cleaning Job Options Details Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
23.1	Name	Nothing.	Error displayed.	Error displayed.	✓
23.2		Length: Type 256 characters.	Limited to 255 characters.	Limited to 255 characters.	✓
24	Description	Length: Type 1001 characters.	Limited to 255 characters.	Limited to 255 characters.	✓

Cleaning Job Details Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
25.1	Address	Length:	Limited to 500 characters.	Limited to 500 characters.	✓

		Type 501 characters.			
25.2		Clear the text box after inputting something or try to click Next when it is empty.	Error message appears	Error message appears	✓
26	Extra Information	Length: Type 501 characters.	Limited to 500 characters.	Not limited to 500 characters (limited to 256).	□

Cleaning Job Date and Time Page

Test ID	Test Description	Input Data	Expected Outcome	Actual Outcome	Pass
27.1	Date	Nothing	An error message is displayed.	An error message is displayed.	✓
27.2		Partial input: Day: 01 Month: 06	The error message “Invalid date of birth” is displayed.	The error message “Invalid date of birth” is displayed.	✓
27.3		Valid date: Day: 01 Month: 02 Year: 2005	No error displayed.	No error displayed.	✓
27.4		Erroneous date: Day: 32 Month: 31 Year: 1000	The error message “Invalid date of birth” is displayed.	The error message “Invalid date of birth” is displayed.	✓
28.1	Time	Nothing.	An error message is displayed.	An error message is displayed.	✓
28.2		Erroneous: 25:76	An error message is displayed.	An error message is displayed.	✓
28.3		Start time after end time.	The error message “Ensure the end time is after the start time” appears.	The error message “Ensure the end time is after the start time” appears.	✓

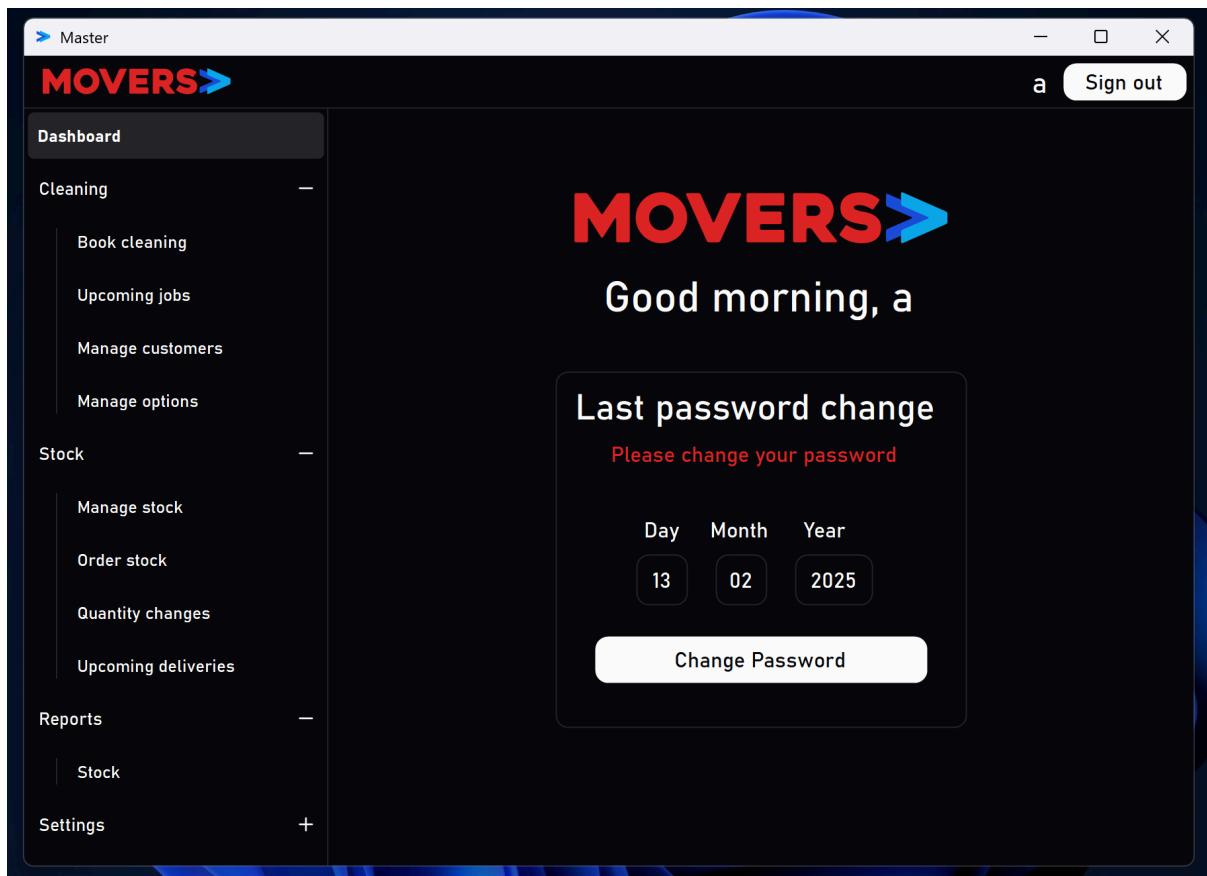
Link Testing Fixes

Below is a list of the tests that failed and the corresponding fixes.

10.3 - Cleaning Manager Menu

Description

This test failed due to an extra entry in the menu when signed in as a cleaning manager. Cleaning managers should not have access to the part of the application that allows staff members to make bookings.



Code

The code causing the issue occurs within the `MasterPresenter` class. There is an extra entry for the cleaning manager menu, left over from development.

```
MasterPresenter.cs

    private string[][] GetMenuItems(PrivilegeLevel staffPrivilegeLevel) =>
    staffPrivilegeLevel switch {
```

```
PrivilegeLevel.Office => [
    ["Dashboard"],
    ["Cleaning", "Book cleaning", "Upcoming jobs", "Manage customers",
     "Manage options"],
    ["Stock", "Orders"],
    ["Reports", "Cleaning job"],
    ["Settings", "Personal information", "Contact details", "Emergency
     contact", "Account security", "Appearance"]],

    PrivilegeLevel.Cleaner => [
        ["Dashboard"],
        ["Cleaning", "Upcoming jobs"],
        ["Reports", "Cleaning job"],
        ["Settings", "Personal information", "Contact details", "Emergency
         contact", "Account security", "Appearance"]],

    PrivilegeLevel.CleaningManager => [
        ["Dashboard"],
        ["Cleaning", "Book cleaning", "Upcoming jobs", "Manage customers",
         "Manage options"],
        ["Stock", "Manage stock", "Order stock", "Quantity changes",
         "Upcoming deliveries"],
        ["Reports", "Stock"],
        ["Settings", "Personal information", "Contact details", "Emergency
         contact", "Account security", "Appearance"]],

        // Rest of the switch statement here
    ]
}
```

Solution

To fix this bug, the extra list is removed.

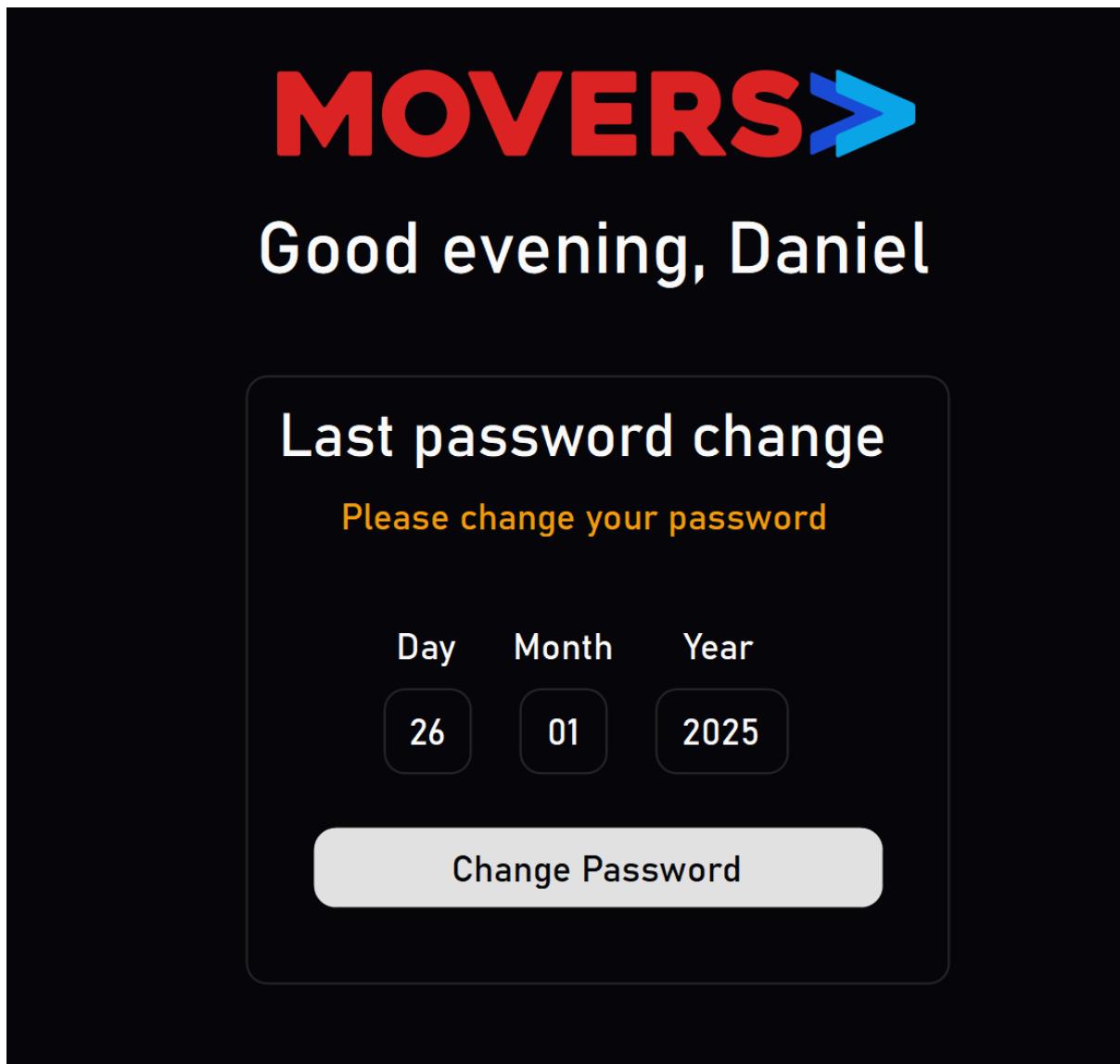
```
MasterPresenter.cs

    PrivilegeLevel.CleaningManager => [
        ["Dashboard"],
        // Cleaning list removed here
        ["Stock", "Manage stock", "Order stock", "Quantity changes",
         "Upcoming deliveries"],
        ["Reports", "Stock"],
        ["Settings", "Personal information", "Contact details", "Emergency
         contact", "Account security", "Appearance"]],
```

11 - Change Password Button Redirect

Description

When clicking the change password button in the dashboard page (with a password older than thirty days old), the view does not direct to the change password page.



Code

This bug happened due to lack of binding within the presenter. The presenter had the infrastructure to navigate, but did not have an event to trigger it.

Solution

The solution is to add the binding so the navigation will work.

```
DashboardPresenter.cs
```

```
private void OnChangePasswordClicked(object? sender, EventArgs e) =>
    NavigateChangePassword();

    // Other code here

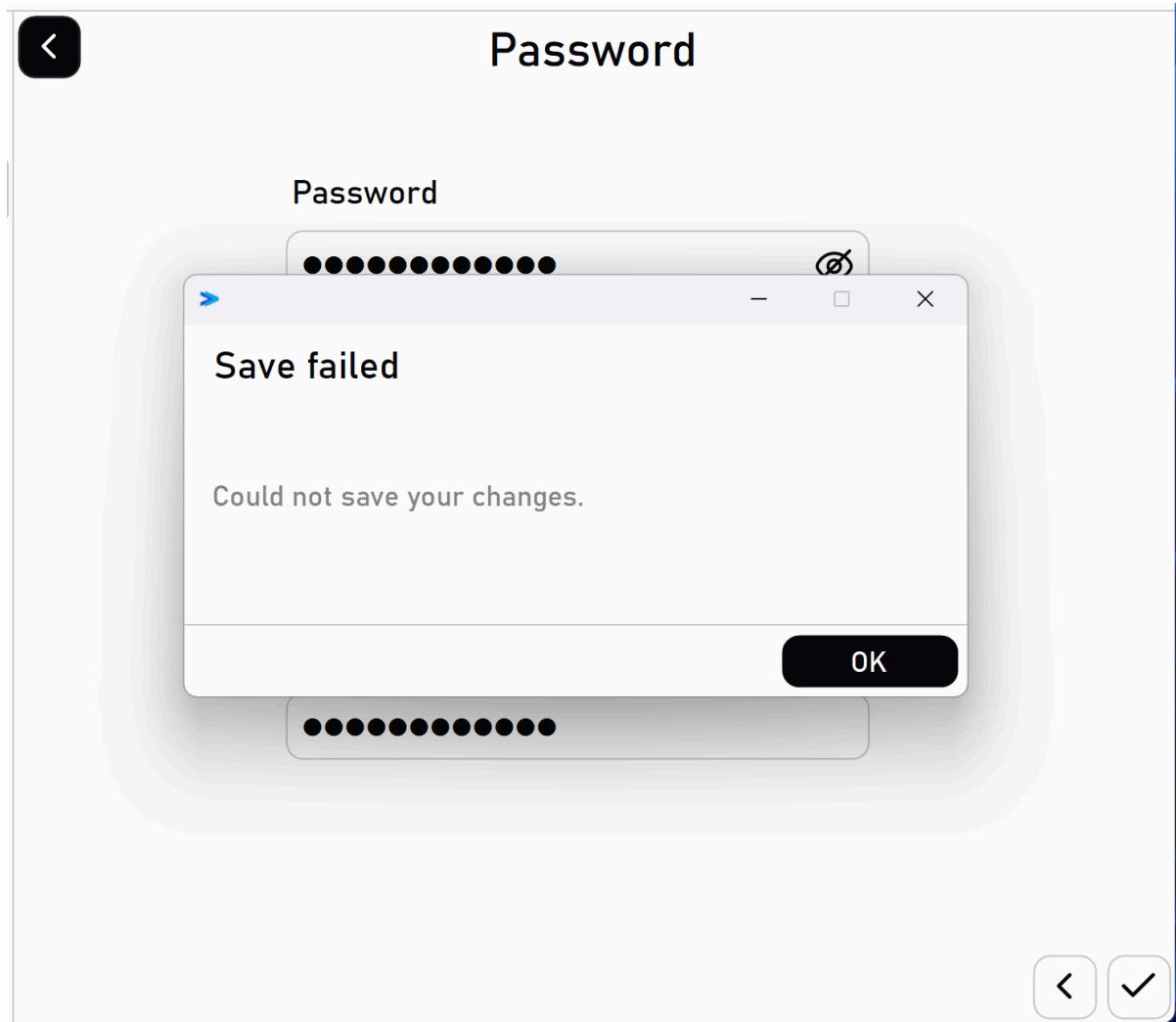

private void NavigateChangePassword() {
    (IChildView view, IChildPresenter presenter) =
        StaffFactory.CreateSecuritySettings(_staff);

    NavigationRequest?.Invoke(this, new NavigationEventArgs(view, presenter)
    { NavigateMenu = true, MenuItemName = "Account security" });
}
```

53.4 - Adding a New Staff Member

Description

This test failed as the staff member was unable to be added successfully after the values were populated.



Code

The code causing this issue was within the `StaffDAL` class. The code was incorrectly handling null values when passing them to stored procedures in the database. The problematic code lies in the line where the date of birth is added as a stored procedure parameter.

```
StaffDAL.cs
```

```
    public static async Task<bool> CreateStaff(string username, string  
hashedPassword, string salt, string forename, string surname, DateTime?  
dateOfBirth, string email, string phoneNumber, string address, string
```

```

emergencyContactForename, string emergencyContactSurname, string
emergencyContactPhoneNumber, int privilegeLevelId) {
    await using SqlConnection connection = new(_connectionString);
    await connection.OpenAsync();

    await using SqlCommand command = new("CreateStaff", connection);
    command.CommandType = CommandType.StoredProcedure;
    command.Parameters.AddWithValue("@username", username);
    command.Parameters.AddWithValue("@hashedPassword", hashedPassword);
    command.Parameters.AddWithValue("@salt", salt);
    command.Parameters.AddWithValue("@forename", forename);
    command.Parameters.AddWithValue("@surname", surname);
    command.Parameters.AddWithValue("@dateOfBirth", dateOfBirth);
    command.Parameters.AddWithValue("@email", email);
    command.Parameters.AddWithValue("@phoneNumber", phoneNumber);
    command.Parameters.AddWithValue("@address", address);
    command.Parameters.AddWithValue("@emergencyContactForename",
emergencyContactForename);
    command.Parameters.AddWithValue("@emergencyContactSurname",
emergencyContactSurname);
    command.Parameters.AddWithValue("@emergencyContactPhoneNumber",
emergencyContactPhoneNumber);
    command.Parameters.AddWithValue("@privilegeLevelId", privilegeLevelId);

    int rowsAffected = await command.ExecuteNonQueryAsync();
    return rowsAffected > 0;
}

// Rest of the StaffDAL methods

```

Solution

To fix this problem, null DateTime values must be explicitly converted to DBNull.Value. This has been fixed below.

```

StaffDAL.cs

// Start of the method code

    command.Parameters.AddWithValue("@surname", surname);
    command.Parameters.AddWithValue("@dateOfBirth", (object?)dateOfBirth ??
DBNull.Value);
    command.Parameters.AddWithValue("@email", email);

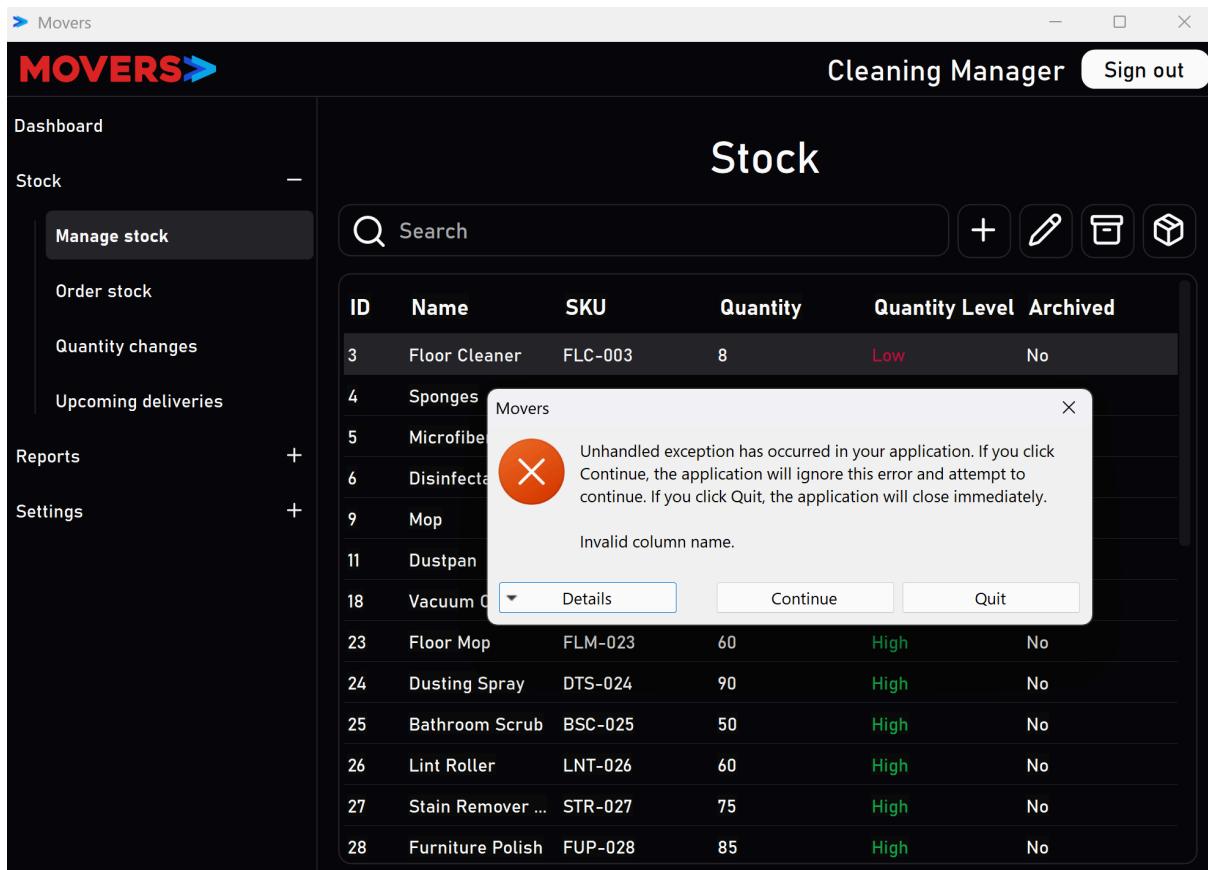
// Rest of the method code

```

32.1 - Stock ID Sorting

Description

When trying to sort by stock ID within the stock management view, an exception is thrown.



Code

This error occurred due to the columnID being incorrectly named. This meant the code fell through to the default case and threw an exception.

```
DisplayStockPresenter.cs

    protected override void SortByColumn(string columnName, bool sortAscending)
{
    if (_isAsyncRunning) return;

    switch (columnName) {
        case "columnID":
            SortBy(x => x.Id, sortAscending);
            break;
        case "columnName":
            SortBy(x => x.Name, sortAscending);
            break;
```

```
        case "columnSku":
            SortBy(x => x.Sku, sortAscending);
            break;
        case "columnQuantity":
            SortBy(x => x.Quantity, sortAscending);
            break;
        case "columnQuantityLevel":
            SortBy(ConvertQuantityLevelToInt, sortAscending);
            break;
        case "columnArchived":
            SortBy(x => x.Archived, sortAscending);
            break;

        default:
            throw new NotImplementedException("Invalid column name");
    }

    DisplayItems();
}
```

Solution

This code can be simply fixed by renaming the “columnID” case to “columnId”.

```
DisplayStockPresenter.cs

    protected override void SortByColumn(string columnName, bool sortAscending)
{
    if (_isAsyncRunning) return;

    switch (columnName) {
        case "columnId":
            SortBy(x => x.Id, sortAscending);
            break;
        case "columnName":
            SortBy(x => x.Name, sortAscending);
            break;
        case "columnSku":
            SortBy(x => x.Sku, sortAscending);
            break;
        case "columnQuantity":
            SortBy(x => x.Quantity, sortAscending);
            break;
        case "columnQuantityLevel":
            SortBy(ConvertQuantityLevelToInt, sortAscending);
            break;
        case "columnArchived":
            SortBy(x => x.Archived, sortAscending);
```

```
        break;

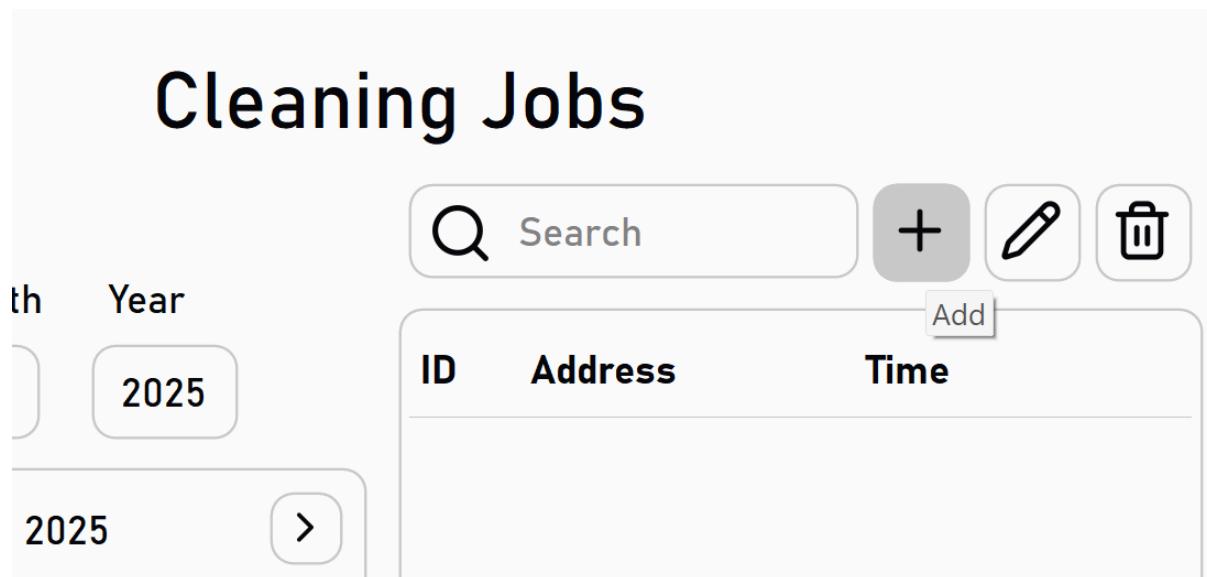
    default:
        throw new NotImplementedException("Invalid column name");
    }

    DisplayItems();
}
```

76 - Cleaning Job Booking

Description

Booking a cleaning job should be limited to at least two weeks in advance. This was not happening, as bookings could be made for any time in the future. This test initially failed.



Code

The code only contains logic for checking if the selected date is before today. The required logic has not been implemented. Instead of disabling the button, I will add a message box to tell the user that they cannot add a booking less than two weeks in advance.

```
BookCleaningJobPresenter.cs
```

```
private void SetAddEditDeleteView() {
    bool beforeToday = _view.Date <= DateTime.Today;
    _view.AddEnabled = !beforeToday;
    _view.DeleteEnabled = !beforeToday;
    _view.ViewMode = beforeToday;
}
```

Solution

The code has been added to the add method (which is called when the add is clicked in the view) to display a message box telling the staff member that they cannot book the cleaning job less than two weeks in advance. The early return also prevents the navigation event from being fired, preventing the adding from taking place.

```
BookCleaningJobPresenter.cs
```

```
private void Add() {
```

```
_cancellationTokenSource.Cancel();

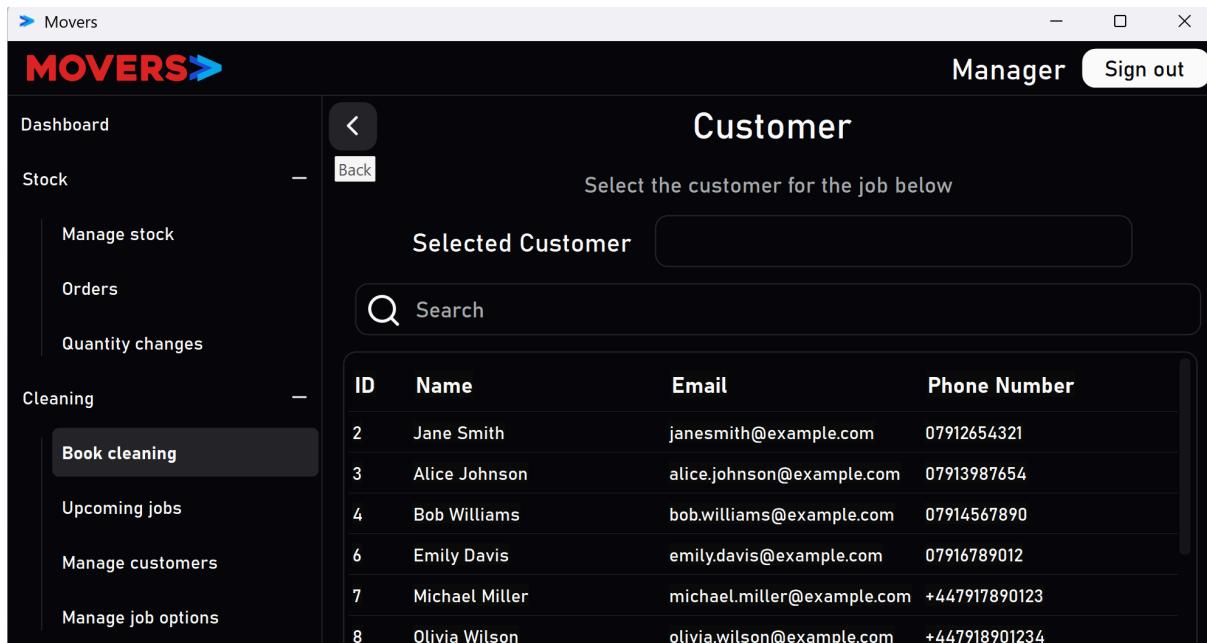
if (_view.Date < DateTime.Today.AddDays(14)) {
    _view.ShowMessageBox("Bookings must be made at least two weeks in
advance", "Cannot place booking", MessageBoxButtons.OK);
    return;
}

(IChildView view, IChildPresenter presenter) =
CleaningJobOptionFactory.CreateAddCleaningJob(_view.Date, _staff);
NavigationRequest?.Invoke(this, new NavigationEventArgs(view,
presenter));
}
```

81 - Navigation Back from Adding a Cleaning Job

Description

When clicking the back button when adding a cleaning job, it redirects to the wrong view.



Code

The code in the navigate method, creates the wrong sub view.

AddCleaningJobPresenter.cs

```
private void NavigateBack() {
    (IChildView view, IChildPresenter presenter) =
    StockFactory.CreateStockDisplay(_staff);
    NavigationRequest?.Invoke(this, new
    NavigationEventArgs(view, presenter));
}
```

Solution

Fix the call to the factory method.

AddCleaningJobPresenter.cs

```
private void NavigateBack() {
    (IChildView view, IChildPresenter presenter) =
```

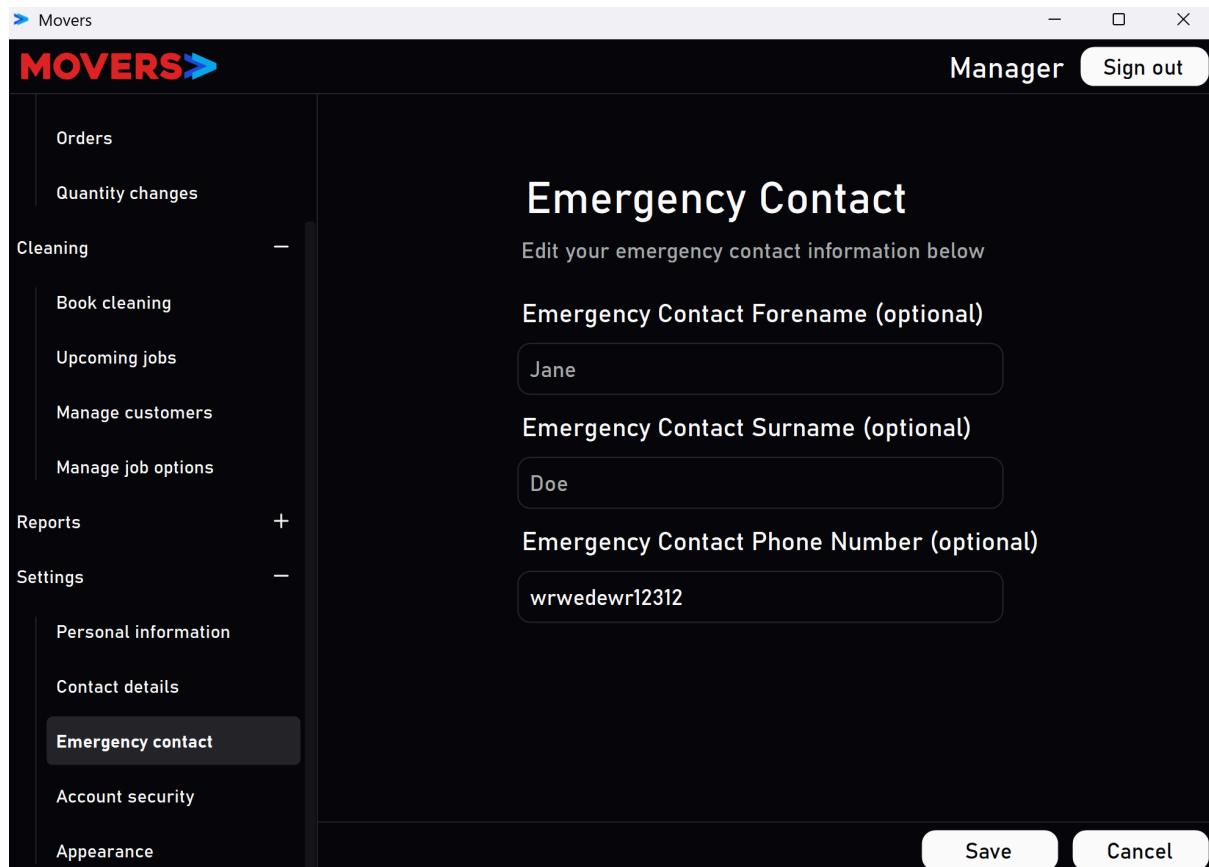
```
CleaningJobFactory.CreateDisplayCleaningJob(_staff);
    NavigationRequest?.Invoke(this, new
NavigationEventArgs(view, presenter));
}
```

Data Entry Testing Fixes

10.1 & 10.2 - Emergency Contact Phone Number

Description

When incorrectly typing in a phone number (e.g. text instead of words) the error message did not appear.



Code

These issues were caused by two separate incomplete methods. The prevention of letters in the view and the failed binding within the presenter.

Solution

I used binding within the constructor and also ensured the text box could only receive digit characters.

```
EmergencyContactSettingsView.cs

    private void tbEmergencyContactPhoneNumber_KeyPress(object sender,
    KeyPressEventArgs e) {
        if (!char.IsDigit(e.KeyChar) && !char.IsControl(e.KeyChar)) e.Handled =
    true;
```

```
}
```

```
EmergencyContactSettingsPresenter.cs
```

```
    private void OnEmergencyContactPhoneNumberChanged(object? sender, EventArgs  
e) => InputChanged();
```

26 - Length of Extra Information in a Cleaning Job.

Description

When typing text in the description, it is unintentionally limited to 256 characters.

Code

The error resulted from an incorrect value of the MaxLength within the designer.

⊕ Margin	11, 11, 11, 11
⊕ MaximumSize	0, 0
MaxLength	256
⊕ MinimumSize	0, 0

Solution

This is fixed by adjusting the MaxLength to 500.

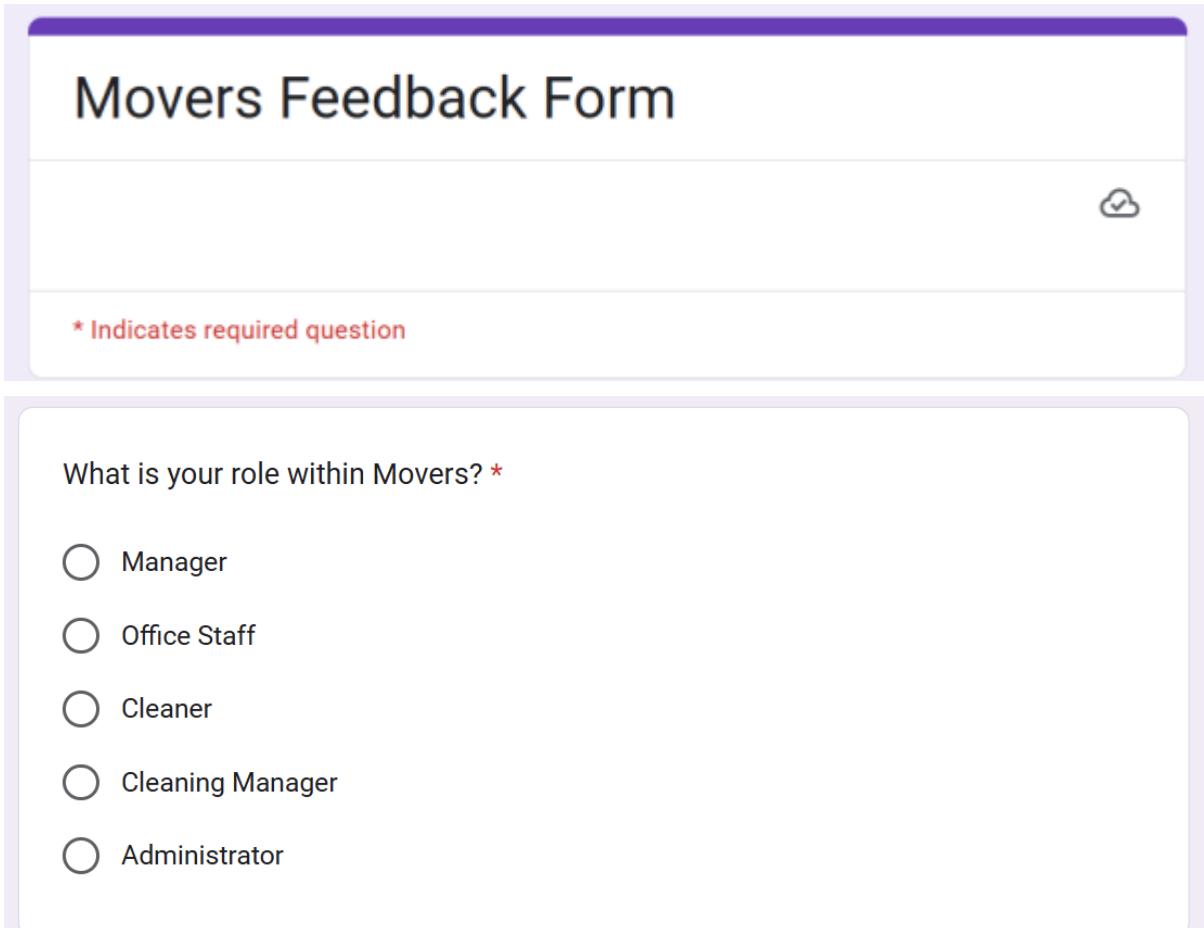
Properties	Value
⊕ Lines	String[] Array
⊕ Location	145, 92
Locked	False
⊕ Margin	5, 5, 5, 5
⊕ MaximumSize	355, 166
MaxLength	500
⊕ MinimumSize	0, 0
Modifiers	Private

User Acceptance Testing

User Feedback

To understand the microsystem's success, feedback must be gathered from those who will be using it. I compiled a series of questions to gather this data from the Movers staff who use the system. The form is included below and can also be found [here](#).

User Feedback Form



The screenshot shows a digital feedback form titled "Movers Feedback Form". At the top right is a small icon of a cloud with a checkmark. Below the title, a note in red text says "* Indicates required question". The main question is "What is your role within Movers? *". Five radio button options are listed: Manager, Office Staff, Cleaner, Cleaning Manager, and Administrator.

Movers Feedback Form

* Indicates required question

What is your role within Movers? *

- Manager
- Office Staff
- Cleaner
- Cleaning Manager
- Administrator

How long have you worked at Movers? *

- Less than 6 months
- 6 months to a year
- 1 - 2 years
- 3+ years

How would you rate the current system? *

1 2 3 4 5



[Next](#)

[Clear form](#)

Application Feedback

Do you feel that the application increases your productivity? *

- Yes
- No
- Maybe

Does the application satisfy your expectations? *

- Yes
- No

Do you have any suggestions for potential improvements to the system?

Your answer

[Back](#)[Next](#)[Clear form](#)

Appearance and Ease of Use

Is the application easy to navigate? *

1

2

3

4

5

Easy



Hard

How appealing is the user interface? *

1

2

3

4

5



Was the application responsive? *

1

2

3

4

5

Unresponsive



Responsive

Any suggestions to improve the user interface?

Your answer

Back

Submit

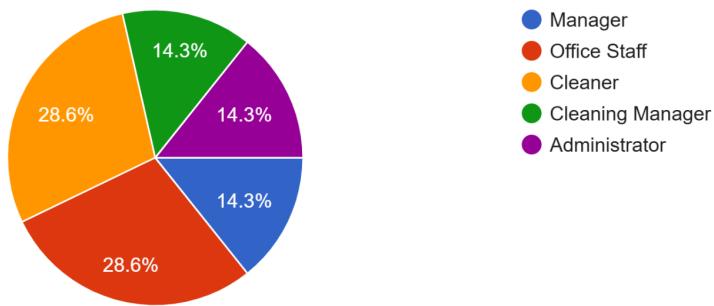
Clear form

User Feedback Form Results

Question 1

What is your role within Movers?

7 responses

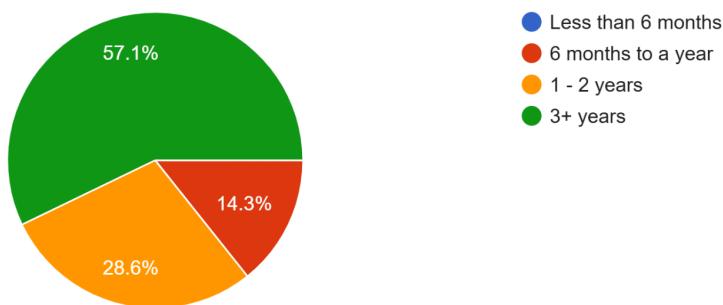


The first question shows that responses have come from a variety of staff members within Movers. This indicates that the survey will provide a good overview of how effective the application is in different environments within Movers and ensures there is a fair representation of all staff members.

Question 2

How long have you worked at Movers?

7 responses

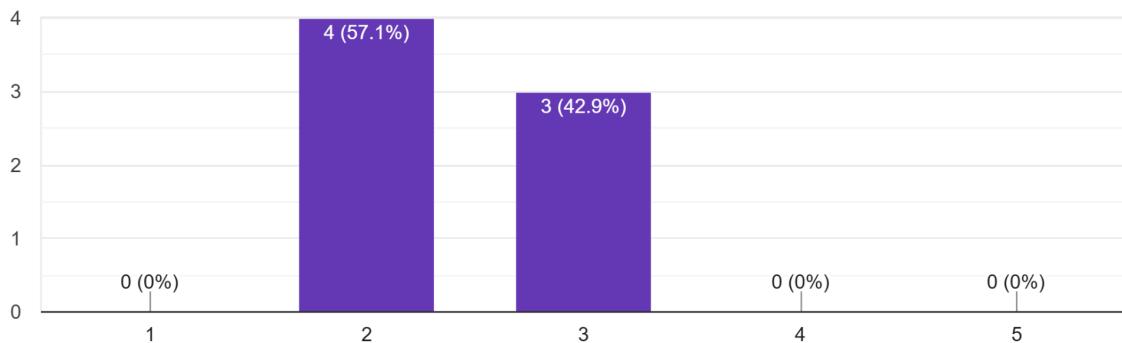


The second question shows that we are receiving responses from both newer and older employees. This also indicates the survey will be a good overview of how effective the application was for both new employees and more experienced ones.

Question 3

How would you rate the current system?

7 responses

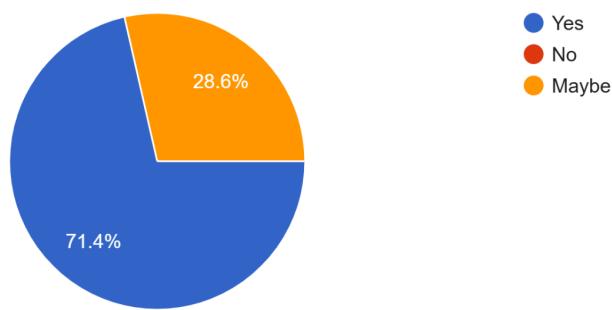


This question confirms that the current system is not rated highly by any Movers employees (average 2.43). This provides a baseline for which the application can improve upon.

Question 4

Do you feel that the application increases your productivity?

7 responses

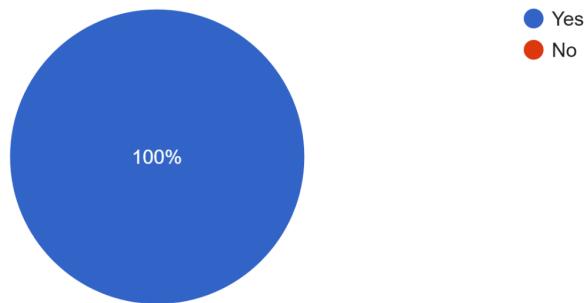


This question provides evidence that the application generally improved productivity within the company. The majority of the responses were positive, with some responses of "Maybe". The "Maybe" responses came from cleaning staff. This could be because they have less usage of the application in their work (they only need to log in to see the upcoming cleaning job and generate the cleaning job report).

Question 5

Does the application satisfy your expectations?

7 responses



This question clearly shows that the application satisfied all user expectations, implying that the project was a success.

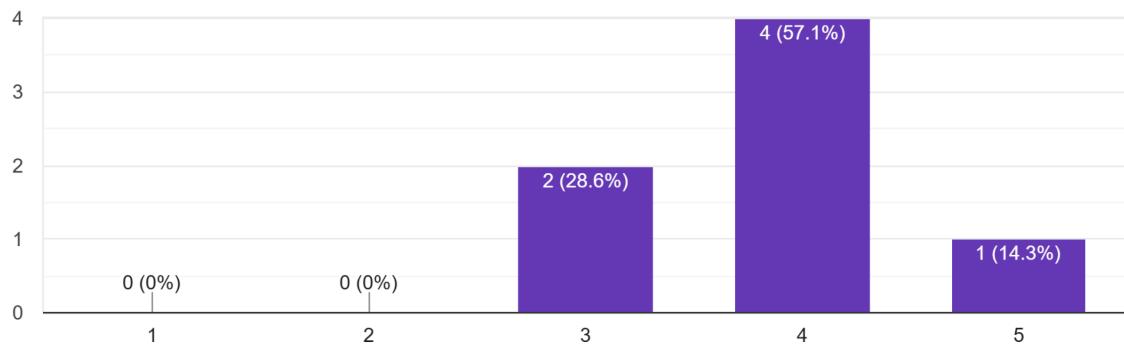
Question 6

One response from question six asked for more advanced reporting facilities. Due to its modular nature, this is an area within the application that could be easily improved upon in the future.

Question 7

Is the application easy to navigate?

7 responses

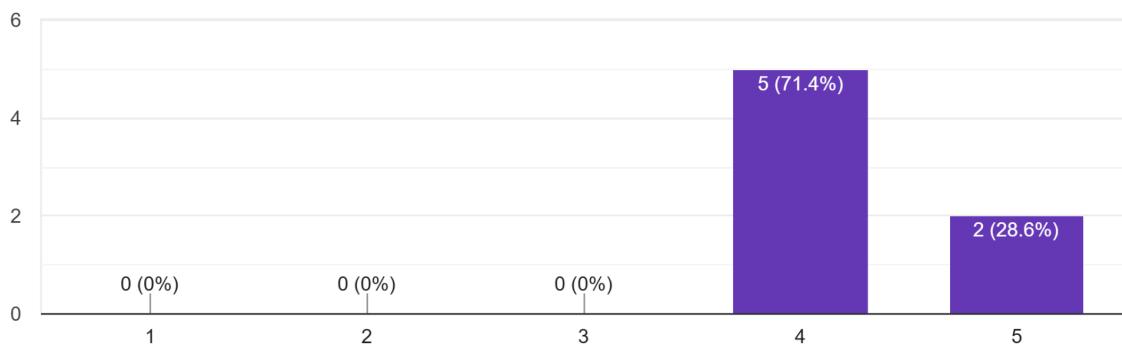


Question 7 shows a generally positive reaction to how easy the application was to navigate. Higher responses also came from managers and office staff, the primary users of the system.

Question 8

How appealing is the user interface?

7 responses



This question shows that the user interface received very positive feedback. This is surprising since I am not a user interface developer, but it is encouraging for the application's reception.

Question 9

Question 9 asked for any suggestions for the user interface (UI). None were proposed, suggesting that very few improvements could be made to the UI.

Conclusion

The application was received very well by the Moving company, which provides strong grounds to state that the project was successful. Overall, I am happy with the responses and will aim to improve the reporting facilities in the future.

Evaluation

User Requirements

User requirements are a key area that determines the success of the project. All user requirements are evaluated below.

Functional Requirements

General			
ID	Description	Success of fulfilment	Pass
FR1	There should be a light and dark mode for every part of the user interface.	Each view within the application has both a dark and a light mode, which staff members can change in settings. The default theme can also be changed for the sign-in page through the app config.	✓
FR2	The application should be menu-driven with a side menu.	A side menu allows users to access different parts of the application.	✓
FR3	The side menu should have drop-down tabs for clarity.	The menu is divided into sections that drop down to expose the controls inside when clicked. A plus or minus sign indicates when the drop-down menu is open.	✓
Sign In			
ID	Description	Success of fulfilment	Pass
FR4	Staff members should be able to log in with a username and password.	Each staff member has a dedicated account from which they can access the services within the application.	✓
FR5	Staff members should have a unique username and a strong password.	The staff member's unique name has been enforced successfully by making the username unique within the database. The password has been made strong through password requirements and then stored securely on the database with password hashing and salting.	✓
FR6	Staff members should be able to show their	This has been achieved by using a button on the password bar to toggle the password visibility in plain	✓

	password in the login page.	text. The button icon also changes to show whether the password is visible.	
FR7	Staff members should be able to toggle between light and dark modes on the login page.	The change theme button at the top right corner of the change password page achieves this. The default theme can also be customised in the AppConfig file.	✓
FR8	The sign-in page should pause after the staff member submits a password to prevent rapid password and username entry.	There is a short pause of at least 1.5 seconds after the sign-in button is clicked after the credentials have been entered.	✓
FR9	When a user logs in with an incorrect password or username, there should be a descriptive error message.	Under the password text box, descriptive error messages are shown. The messages are as follows: "Please fill in a username and password," "Please fill in a username," "Please fill in a password," and "Username or password incorrect."	✓
Settings			
ID	Description	Success of fulfilment	Pass
FR10	Staff members should be able to edit some of their settings by navigating with the side menu.	Staff members can access the settings page from the side menu and edit their preferences.	✓
FR11	Staff members should be able to edit their forename and surname and provide an optional date of birth.	Staff members can update their forename and surname and optionally enter a date of birth.	✓
FR12	There should be validation of forename and surname to prevent the user from leaving them blank.	Validation has been successfully implemented. If a user tries to leave these fields blank, an error message (e.g., "Invalid Forename") is displayed.	✓

FR13	Staff members should be able to edit their contact details (email, phone number and optionally, an address).	Staff members can update their contact details (including email, phone number and optionally an address) in the account security section of the settings menu.	✓
FR14	Staff members should only be able to provide a valid email and phone number. If it is incorrect, the user should be informed by an error message.	Staff members can edit their contact details from the contact details subsection within the settings menu. They can provide an email, phone number (required), and optionally an address. The fields are also validated. If an invalid email or phone number is entered (UK numbers are the default; other numbers must have an international prefix), an error message is shown below the phone number text box.	✓
FR15	Staff members should be able to provide an optional emergency contact (forename, surname and phone number).	Staff members edit their emergency contact. Any field emergency contact settings page can be left blank, but the phone number must be valid.	✓
FR16	Staff members should be able to view their username and privilege level.	The username and privilege level are displayed in the settings section but cannot be edited.	✓
FR17	Staff members should be able to change their password.	The account security tab in the settings section of the menu has a change password section where users can enter their current and new passwords.	✓
FR18	Staff members should be able to change their preferences for displaying tooltips.	There is a setting to enable or disable tooltips in the user interface.	✓
FR19	Staff members should be able to change their font.	Users can select their preferred font by selecting one of Bahnschrift, Century or Comic Sans, with a radio button toggle group.	✓
FR20	Staff members should be able to change their appearance theme by choosing between light and dark modes.	Users can toggle between light and dark modes in the appearance settings.	✓

Dashboard			
ID	Description	Success of fulfilment	Pass
FR21	Staff members should have a welcome message shown when they sign in.	A personalised welcome message is displayed on the dashboard based on the time of day.	✓
FR22	Staff members should be able to see when they should change their password.	If a password is older than thirty days, a warning message appears on the dashboard.	✓
Cleaning Job Management			
ID	Description	Success of fulfilment	Pass
FR23	Staff members should be able to book a cleaning job for a time over 2 weeks in the future.	When booking a cleaning job, adding a booking for a date less than two weeks in the future is disabled (editing is available instead).	✓
FR24	Bookings should be able to be edited until they are complete.	Staff members can edit bookings up until the date and time have passed. This allows for minor adjustments to the booked cleaning job. If major changes are required, the cleaning job can be rescheduled for over two weeks in the future.	✓
FR25	Past bookings should be viewable.	Past bookings are stored in the database, and past dates can be viewed in the bookings page (in read-only mode).	✓
FR26	Bookings can be deleted if the customer cancels their cleaning service.	Bookings can be permanently deleted if a customer cancels their cleaning job.	✓
FR27	Bookings should be able to be rescheduled for a time over 2 weeks in the future.	Staff members can reschedule bookings, provided the new date is at least two weeks in the future. Rescheduling moves the date without affecting the existing details, such as location and cleaning job options.	✓

FR28	Bookings should have a location.	Each booking requires a location, which can be manually entered or selected from predefined locations. The system validates that a location is provided before confirming the booking.	✓
FR29	Staff members should be able to select several cleaning options for each job. They should have an associated quantity.	Staff members can select cleaning job options for the cleaning job and then add the required quantity for each item (default 1).	✓
FR30	Bookings should have an associated cleaning team that can be selected by the staff.	Each booking must have an assigned cleaning team (the office staff can select which cleaners go on which jobs), which is selected from the cleaning staff accounts on the system. The application also prevents assigning the same cleaners to multiple jobs happening at the same time.	✓
FR31	The application should allow bookings to have a start time and an end time to allow for scheduling of cleaning staff.	Bookings include both a start time and an end time.	✓
FR32	Cleaning team members should only be selectable if they are not already on a job at that time.	The application prevents assigning the same cleaners to multiple jobs simultaneously.	✓

Customer Management

ID	Description	Success of fulfilment	Pass
FR33	Customers should have a name, surname, email and phone number.	Each customer record includes mandatory fields for first name, surname, email, and phone number. Input validation ensures that all required fields are completed before saving.	✓
FR34	Office staff should be able to add customers.	Office staff can add new customers by clicking the add button on the display customer page.	✓

FR35	Office staff should be able to edit existing customers.	Office staff can edit existing customer details (forename, surname, email, and phone number).	✓
FR36	Customers should be archivable so that only current customers are visible and old customers can be hidden.	Customers can be archived, making them invisible in the display customer page while retaining their records for historical reference. Archived customers can also be restored if necessary.	✓

Cleaning Job Options Management

ID	Description	Success of fulfilment	Pass
FR37	Cleaning job options should have a name, description and unit cost.	Each cleaning job option includes a name, a description and a unit cost to help price cleaning jobs.	✓
FR38	Office staff should be able to add, edit and delete cleaning job options.	Office staff can add, edit, and archive (non-destructive deleting to preserve previous cleaning job history) cleaning job options.	✓
FR39	Cleaning job options should have their prices recorded historically for cleaning jobs.	The system stores historical price data for cleaning job options linked to the cleaning job. When a cleaning job is booked, the price at the time is recorded to ensure accurate billing and to record historical prices if the cleaning job option's price changes later.	✓
FR40	Cleaning job options should be able to be archived when they are no longer in use.	Cleaning job options can be archived when they are no longer in use. Archived options are hidden from selection but remain in the system for historical reference. They can be restored if needed.	✓
FR41	Only active cleaning job options should be able to be used in a cleaning job.	Only non-archived cleaning job options are available when adding them to a cleaning job.	✓

Stock

ID	Description	Success of fulfilment	Pass

FR42	Stock items should have a name, description and quantity.	Each stock item has a name, a description (allows for purpose to be recorded), and a quantity.	✓
FR43	Stock items should display different warnings indicating whether they have high, low or medium stock.	On the stock display page, stock quantities are displayed beside a quantity level with “Low”, “Medium” and “High” quantity levels. Low stock warnings appear in red, medium in yellow, and high in green to improve readability.	✓
FR44	Stock quantities should be recorded over time to allow stock usage to be seen.	This feature has been fulfilled. Stock quantities are recorded over time, allowing staff to see stock usage.	✓
FR45	Stock items should be able to be archived when they are no longer in use.	Stock items can be archived when they are no longer in use. Archived items are hidden in the display stock data grid by default but can be shown when the view archived stock button is clicked. Archived stock items can be unarchived if they are needed again.	✓
FR46	All stock quantity changes should have a staff member attached to track which staff member made each quantity change.	Every stock quantity change is recorded, along with the staff member who made the change. This ensures accountability. Staff members can also include a reason for the change in quantity.	✓
Stock Reordering			
ID	Description	Success of fulfilment	Pass
FR47	Stock managers should be able to request stock to be bought at the office.	Stock managers can request stock (in the form of an order) to be bought by the office staff. Orders can be created by selecting stock items and inputting the required quantity.	✓
FR48	Orders should be able to be drafted, then sent to the office with a staff member attached to them.	When orders are created, they are initially saved as drafts. Once the cleaning manager has finished drafting the order, they can submit it for approval or rejection by the office staff. The staff member who created the order is linked to it.	✓

FR49	Office staff should have to approve orders and mark them as deliveries when they are requested.	Office staff must reject or approve orders before they are marked as deliveries or rejected.	✓
FR50	Deliveries should have a "pending" and "delivered" state.	Deliveries have a "pending" state while waiting to be received and can be updated to "delivered" once they arrive.	✓
FR51	When a delivery is received, the cleaning manager should record any discrepancies.	When a delivery is received, the cleaning manager can record discrepancies, such as missing or incorrect items, ensuring accurate stock tracking.	✓
FR52	Cleaning managers should update the stock quantities once an order is received.	Cleaning managers are required to update stock quantities once an order is marked as received. Cleaning managers record the stock received, which is then automatically added to the current stock quantities. The reason for the quantity change is "Order x received," where x is the order number.	✓

Administration

ID	Description	Success of fulfilment	Pass
FR53	There should be different types of users, i.e. cleaning staff should not be able to access the same parts of the application as managers. There should be administrative accounts, manager accounts, cleaning staff accounts, office staff accounts and cleaning manager accounts.	Different account privilege levels for different staff accounts have been created to reflect the roles of staff within Movers. These roles include administrator, manager, office staff, cleaning staff and cleaning manager. Role-based access control ensures that staff members only have access to the parts of the application they need to do their job.	✓
FR54	Administrative staff should be able to change other users' passwords.	Administrative staff can reset and change other users' passwords.	✓

FR55	Administrative staff should be able to add new staff.	Administrative staff can add new staff members and assign a role, username and password.	✓
FR56	Administrative staff should be able to edit all existing staff settings.	This requirement has been implemented successfully. Administrative staff can edit all staff settings, including contact information, roles, and privilege levels.	✓
FR57	Administrative staff should be able to archive old user accounts and reactivate them if necessary.	When staff members leave Movers, user accounts can be archived, preventing them from accessing the application. Archived accounts can be restored if necessary.	✓
FR58	Administrative staff should be able to view all sign-in attempts and see if they were successful.	A sign-in activity log allows administrative staff to view all sign-in attempts. This includes data for the username entered, the date and time and whether the attempt was successful.	✓

Non-functional requirements

Navigation			
ID	Description	Success of fulfilment	Pass
NF1	The application should be laid out in such a way as to make it easy to use and understand for new employees.	The application layout is designed for ease of navigation, ensuring new employees can quickly understand and navigate it. User interface elements are consistent throughout.	✓
NF2	The amount of mouse movement and clicks should be minimised to improve productivity.	This requirement was partially realised by binding the enter key on many text boxes to submit the data, and in each display view, double-clicking on an item performed an action (e.g., edit the selected item). Key bindings/shortcuts for controls were not fully realised due to the constraints of focus handling within Windows Forms. Focus determines which control will receive input from the keyboard, which is unreliable within the Windows Forms event system.	□

NF3	The application should be menu-driven to allow easy, self-explanatory navigation.	The application is fully menu-driven, featuring a structured and intuitive navigation system that allows users to access different parts of the application.	✓
NF4	Users should be able to tab between controls.	Users can navigate between input fields and controls using the Tab key, improving accessibility and workflow efficiency. Focus order is logically arranged to match expected user interactions.	✓
NF5	A search functionality should be available for searching through different items, such as stock.	All display views have search functionality. This allows users to search through various items, including stock, customer records, and bookings.	✓

Accessibility

ID	Description	Success of fulfilment	Pass
NF6	All the controls should have accessibility options such as sensible tab orders.	All controls have logical tab orders to improve navigation. Key controls have screen reader-compatible audio descriptions to assist visually impaired users.	✓
NF7	The font should be easily legible, and there should be an option to change the font to something easier to read for those with Dyslexia.	The default font is chosen for high readability (other fonts are also provided), and users can change it to a Dyslexia-friendly font (Comic Sans) in the settings.	✓
NF8	There should be a consistent margin between controls to improve readability.	There are consistent margins and spacing between controls.	✓
NF9	There should be accessibility options on the login menu.	Users can change between dark and light mode in the sign in page.	✓
NF10	A light mode and dark mode should be implemented. Dark mode should be enabled by default to reduce eye strain.	Both light and dark modes are available.	✓

Performance

ID	Description	Success of fulfilment	Pass
NF11	The application should be fast to load.	The loading time is normally never over 1 second.	✓
NF12	The application should have a low memory footprint.	The application has a higher-than-normal memory footprint to achieve dark and light modes; however, it still has a relatively small memory footprint.	✓
NF13	The application should be able to handle multiple users interacting with it simultaneously. It should also be able to switch between numerous user accounts quickly.	Users can sign out from any point in the application by clicking the “Sign out” button on the top right hand corner. The application has been designed to handle partial updates, allowing for multiple users to interact with it simultaneously.	✓
NF14	The application should run at a reasonable speed, even on a low-end computer.	Windows Forms have relatively low memory and computational requirements. This means the application can run on even a low-end computer.	✓
NF15	The application should have a 99.99% uptime and be reliable.	The application has been built in Windows Forms, which means it is reliable.	✓
NF16	Power efficiency should be good.	Windows Forms is desktop-based, so it is power-efficient.	✓

Portability and extensibility

ID	Description	Success of fulfilment	Pass
NF17	The application should be portable and run on Windows computers.	This requirement has been fulfilled by using Windows Forms, which can run on any Windows computer.	✓
NF18	The application should be engineered in such a way as to allow the development and integration of other microsystems quickly and efficiently.	The application uses MVP, so it is extensible and modular. This will allow easy future system integration.	✓
NF19	The application should be able to function as a	The application can function by itself without requiring other microsystems.	✓

	standalone microsystem.		
NF20	The application should be able to be used on all Windows computers.	The application is built using Windows Forms, is performant, and can run on all Windows computers.	✓
NF21	The application should never stall or freeze. This includes long-running processes such as database queries.	The application is implemented using asynchronous programming to minimise freezing.	✓

Conclusion

Overall, I have successfully implemented the user requirements and effectively satisfied nearly all user needs. The only requirement that was not completely met was due to limitations within Windows Forms.

Approach

Overall, my approach to the solution for this project was successful.

Before choosing a solution, I did extensive research into different technologies that could be used to build the application. This included research into web development, mobile development, and desktop development. I also considered different technologies that I would use to develop the application for each of these formats. In addition to this, I evaluated different database approaches to ensure that I had chosen the optimal solution for data management.

Firstly, using a desktop application proved to be a good choice as the Movers' Cleaning Application did not require portability. This also simplified the installation and updating process.

Secondly, Windows Forms offered strong performance gains and lower memory usage than a potential web-based solution. This allowed me to fulfill the user requirement of having a lightweight, performant application and also ensured that the application would function on older, less powerful hardware. Windows Forms also has strong backwards compatibility with Windows 10.

Thirdly, Windows Forms allowed for rapid development as the application UI could be created from within the Windows Forms designer. This greatly increases the rate of development over other technologies, such as HTML and CSS, due to the less steep learning curve.

Finally, many users are familiar with the Windows operating system. So, using Windows Forms allowed direct development on the operating system that users will be most comfortable working on.

While my approach to the Movers' Cleaning Application had many successes, there was one difficulty. When working with Windows Forms, I struggled to correctly manage focus within the application due to the unreliability and outdated nature of the development framework. This meant I could not easily clear focus from controls in an elegant manner and could not reliably detect key presses when no controls were selected. This issue could be resolved by using a more modern alternative to Windows Forms, such as WPF.

Overall, I am happy with my decision to develop the application using Windows Forms. This is mainly due to the performance gains related to desktop applications and the backward compatibility. I would be confident in developing future Windows Forms applications. I would consider using a modern alternative to the Windows Forms framework for future projects.

Architecture

The solution's use of the Model View Presenter (MVP) architecture proved very effective. This is evident for several reasons.

Firstly, MVP allowed me to produce modular code. This helped me satisfy the requirement that the code is easy to expand upon and to add extra features. The separation of code through interfaces and events meant that classes were loosely coupled and easy to implement due to the rigid, well-planned structure.

Secondly, MVP increased the rate of development. Once I understood the structure of MVP, I was able to implement any features quickly. Views proved to be incredibly easy to implement as all business logic was refactored into the presenter. Presenters were also easy to implement as they did not require complex UI interactions. Finally, the usage of Data Access Layer classes improved the code structure to encapsulate the database queries safely.

Thirdly, MVP reduced overall code complexity. The use of a "dumb" view and the observing presenter helped reduce overall complexity by separating code into sensible locations. The limited code within views also improved ease of debugging, as I instantly knew where to look when I encountered a bug.

Despite MVP having many successes, I did also experience some issues with the architecture.

Firstly, I did not fully follow MVP from the start of the project, which meant that some earlier classes had to be refactored. Due to MVP's steep learning curve, tightly coupled code was necessary.

Secondly, I experienced a major memory leak due to my decision to use an observing presenter. I sometimes neglected to correctly unbind from view events in the presenter, creating a cyclic reference that was unable to be disposed of by the garbage collector. This led to the application consuming more memory than it actually required; however, as Windows Forms is quite performant, this was less serious than I initially thought. In order to fix this, I created an abstract class that all presenters would inherit from. This further improved the structure of the presenters by forcing me to implement event unbinding.

Thirdly, MVP proved to be slow to implement initially. Each view required an interface, which slowed development at the start of the project. In addition, this structure made my initial planning with the class diagram more involved and required a lot of forward thinking.

Overall, the Model-View-Presenter architecture served me well within the project and I would definitely use it for future projects. I have gained confidence in the MVP structure and would feel confident quickly developing the project further with this

architecture. The reduced overall code complexity also meant there was less mental overhead required for development.

Design Methodology

The Movers' microsystem was developed using the Waterfall Methodology. I followed the Waterfall Methodology's core principle of linear development. The Waterfall Design Methodology was successful in this project for several reasons.

Firstly, the Waterfall Methodology's focus on clear planning allowed for effective decisions on how much time to spend on each part of the project. This proved to be quite effective, as time spent planning the project with both a Gantt chart and PERT chart, common techniques used within the Waterfall Methodology, allowed me to stay within the time-constrained six-month period for the project's completion.

Secondly, the strict user requirements in the case study meant there was no need for iteration. The Waterfall Methodology complemented this rigid structure, forcing me to thoroughly define the functional and non-functional user requirements before beginning development. This also allowed me to evaluate my performance in implementing the user requirements upon completing the project. In addition to this, the rigid user requirements limited the opportunity for scope creep and helped maintain a balance of how I invested my resources, in line with the project manager's "Iron Triangle".

Finally, the Waterfall Methodology helped me produce a large amount of documentation. This documentation is necessary for a school project to showcase my ability to manage the project and implement the solution.

While the waterfall methodology was well-suited to this project, I experienced several issues related to this methodology.

The Waterfall Methodology has an extremely rigid linear approach. This meant that I had to be certain of initial planning, such as storyboards, as they were ideally subject to no changes once the design phase was complete. This was not the case, and there were final variations in storyboard design from the proposed final designs. This proved challenging to get right the first time, as I knew I could not review them later.

Secondly, the waterfall methodology allows limited stakeholder interaction. This meant I could not get feedback on the application until it was fully complete and could not make any changes.

Finally, the Waterfall Methodology produced large, unnecessary amounts of documentation that would not be useful in a real-world application. For example, detailed storyboard descriptions are not useful, and the evaluation of a large variety of different methodologies and design options have limited usefulness.

In conclusion, the Waterfall Design Methodology served me well when undertaking this project and provided numerous advantages, such as time management and documentation production. If I were to undertake a similar project again, I would use an Agile methodology such as SCRUM or RAD to interact more with the

stakeholders. This would also give me the opportunity to make iterative and incremental improvements to the application, leading to a higher-quality solution as a result.

Project Management

Overall, my project management techniques helped me use my time efficiently. The Gantt chart allowed me to see what I needed to do and the time allocated to complete it. This gave me clear guidance for which section of the project I should work on next. It also helped prevent me from spending excessive time on any one section of the project or delaying tasks when I experienced difficulties. This was further complemented by using the PERT chart, which I used to manage task dependencies. The PERT chart ensured that no tasks on the critical path were delayed, ensuring the project was completed within the allocated time.

Another project management technique that would have been useful for the implementation is a burndown chart. A burndown chart would have allowed me to see clearly which features I had successfully implemented, given me a sense of accomplishment as I completed tasks, and allowed me to see the estimated finish time based on the current progress.

In addition to this, the MoSCoW method. The MoSCoW method is a prioritisation method that categorises user requirements into an order of priority based on requirements that the project must, should, could, could not, and won't have. This could have greatly helped minimise the time spent developing requirements such as the application's dark and light modes. Light and dark modes greatly improved the overall aesthetics and accessibility of the application but provided minimal value from a business perspective. The time spent fulfilling this requirement could have been better spent on other application parts, such as developing more advanced reporting facilities.

While my project management generally proved successful, there were some issues. Firstly, I spent a lot of time developing custom controls at the start of the development phase, which proved to be time-consuming initially but later increased the pace of development. This put me under unnecessary pressure as this time was not effectively accounted for within the Gantt and PERT Chart, and the front-end (user interface) development was more time-consuming than initially expected. Secondly, I did not update my Gantt and PERT charts with minor variations in the project timeline, which led to them deviating slightly from the actual timeline of the project. This was in accordance with the waterfall methodology, which dictated a linear flow of progress. If I had updated them, I would have had a more accurate project timeline plan, relieving any stress and confusion associated with slight variations.

If I were to undertake a similar project again, I would be more disciplined with my project management. I would update the Gantt and PERT charts if the project timeline deviated from what was planned and consider the different stages of the development phase more carefully, e.g., allocate time to make custom controls. In addition to this, I would also consider using other project management techniques,

such as a burndown chart, to track progress more effectively and categorise my user requirements using the MoSCoW method.

Testing Process

The testing process was successful overall and effectively improved the application's robustness by removing any bugs created during development. This was for several reasons.

Firstly, as testing continued, the number of bugs detected improved significantly. Near the end of testing, the project seemed to function very reliably, and only minor bugs were found.

Secondly, the testing plan was comprehensive and covered all aspects of the application. Testing was broken down into link testing, data entry testing and user acceptance testing. This ensured that all pages within the application received extreme scrutiny within these three sections, ensuring they functioned reliably and limiting the risks of undetected bugs. Creating a test plan first also ensured I knew what I needed to test, so I was efficient with my time.

Thirdly, user acceptance testing (through user feedback) allowed me to ensure the application was successful among those who would actually use it. This ensured the final product was both useful and effective. From the results, it was clear that the application improved productivity and was easy to use. In addition, the application fulfilled all but one minor user requirement.

Fourthly, I used extensive link testing to ensure that all navigation within the system functioned properly. This included link testing with all states of data and for every possible click on every page.

Fifthly, I effectively used data entry testing to ensure that only the correct data could be inputted. This ranged from normal cases of input data to more extreme cases. I also checked inputs for length, ensuring the input data would not become too long.

While the testing process had many successes, there were also some difficulties I experienced and improvements I would make.

Firstly, I could have used different testing techniques to ensure more rigorous testing. I mainly focused on black box testing, which ensured that the application appeared to work from a user's perspective but did not validate any code. I could have used white box testing with detailed unit tests to ensure that all classes and methods only interacted with data in the way that was intended.

Secondly, testing could have been more efficient if I had been able to give it to another person working on the application. This would have made it less likely for me to miss bugs, as there would have been a different perspective for the testing and development. This is one of the reasons larger companies use dedicated testing teams to have multiple "testing engineers" who test the system much more quickly and efficiently.

Thirdly, while I tested each link and input, I did not test every possible combination of inputs and clicks due to the extreme number of possible options. In the future, I could develop an even more thorough test plan to help combat this issue.

Finally, due to my decision to develop the project using the Waterfall Methodology, I risked uncovering serious errors within the application that would be difficult to fix. This is due to the fact that testing only occurred near the end of the project, so any major errors could have prevented the project's completion. Fortunately, no major errors were encountered, however, I would consider using a more continuous testing strategy that an agile methodology would propose to minimise this risk and my own stress in future projects.

In conclusion, I think my testing process was well-structured and comprehensive; however, I would consider using different testing techniques for future projects.

Solution

The solution I provided for Movers has met the required functionality that was expected.

Firstly, the application manages all cleaning job-related functionality and also provides several security features. This includes managing cleaning stock, stock orders, staff, customers, cleaning jobs, stock deliveries, login attempts, and cleaning job options. Additionally, it provides reports for stock, staff, and cleaning jobs. Each of these detailed CRUD operations (Create, read, update, delete) is for each distinct section. Additionally, the application has privilege-based access to each of these areas. This means that staff members with different jobs can only see what they need to within the application, improving productivity by removing distractions and also improving security. Staff members can only access what they need to complete their jobs.

Secondly, the user interface has shown to be of high quality (as seen in the positive feedback from the user acceptance testing survey). There is both a dark and a light mode, several fonts to select from, and the ability to toggle tool tips. This greatly improved the overall user interface experience and accessibility.

Thirdly, the application was well received by those who will actually use it, as seen through the user acceptance testing results and the evaluation of the implementation of the user requirements. This provides clear evidence that the application served its intended purpose efficiently and effectively.

Fourthly, the application's navigation is intuitive. It consists of a drop-down side menu that allows quick navigation from anywhere within the application. This helps minimise mouse clicks and improve productivity.

Fifthly, throughout the application, there are descriptive error messages in the event of bad user inputs. These prevent incorrect data from being saved while also providing information to the staff member indicating which data is incorrect and why.

Sixthly, the application uses robust asynchronous database operations to prevent the user interface from freezing during a database operation (e.g., when a large amount of text is displayed in a data table). In the event of a failed database connection, e.g., when using an internet-based database on a server, detailed error messages are also provided so the application will not crash. In addition to this, database updates are allowed for partial record changes on the server. This means multiple users could edit different parts of the same record at one time and not cause one update to overwrite the other.

Finally, the solution provided comes with a detailed, instructional user guide to help any new staff members learn how to use the application if they are stuck. The application has been designed to be intuitive without the user guide through the use

of tool tips and icons, so the user guide serves as an effective backup if users are still unsure.

Overall, my solution had many strong points of success; however, I would make several improvements in future releases of the Movers' Cleaning Microsystem Application.

Firstly, I would develop colour coding within the calendar used to make bookings. This would allow for specific dates to be highlighted based on Movers' cleaning staff availability and to allow for more intuitive booking of cleaning jobs. I would highlight dates with low booking availability in red and those with high availability in green.

Secondly, I would implement automatic stock reordering. This would involve drafting an order to restock any low-stock items at the end of each month. The cleaning manager could then check this order and send it to the office so that they could replenish the supplies of those stock items.

Thirdly, I would improve the search algorithm to make it efficient with larger datasets (millions instead of hundreds of thousands). I would do this by using algorithms such as early pruning of the search space with a BK tree and also enabling full text search within the database.

Finally, I would further develop the reporting facilities to provide more summary statistics, such as key performance indicators like monthly revenue, staff activity, and most used stock items. This would further improve the business insights generated with the system.

In conclusion, the solution is overall very successful due to its extensive feature set and fulfillment of the user requirements. The positive response from user acceptance testing also suggests that the solution met Movers' requirements and provides an indicator that the overall project was successful. If I were to develop the application further, I would make several minor improvements. I would also work with the Movers' staff in greater detail to ensure the application is even more suited to their needs.

Own Performance

Overall, my performance proved to be very effective.

Firstly, I have successfully created an application that has fulfilled Movers' requirements. It provides a large amount of functionality for every user requirement bar one, which was impossible to fulfill due to the nature of Windows Forms.

Secondly, I rapidly learned SQL and the techniques for creating a relational database to store data for the Movers' Cleaning Management Microsystem. The database was successfully planned, normalised, and displayed with an Entity Relation Diagram. It was then implemented in practice using T-SQL within the Microsoft SQL Server. I experienced minimal difficulties while doing this despite having no initial experience with SQL and databases.

Thirdly, I effectively used version control within the project to ensure any changes I made were tracked. I achieved this through the use of a Git repository hosted on Github. I am experienced with Git and experienced no issues using the Git tool through the command line. I also regularly saved my progress after every development session with descriptive "commit messages" (to record my progress) and uploaded it to the remote repository to ensure no changes were lost in the event of an emergency, such as the local file becoming corrupted.

Fourthly, I have effectively documented the whole project's development process from start to finish. I provided detailed documentation for the design phase, database development, system architecture, and various other parts of the project.

Finally, I effectively understood the prompt from a business perspective. This led to the development of role-based access control, which ensured staff could only access the parts of the application they needed to do their jobs. Additionally, I ensured all bookings and stock quantity changes had an attached staff member to ensure accountability from staff members.

While my performance was generally successful, there are some improvements I would make in the future.

Firstly, I used the Gantt Chart as a checklist to manage my progress. However, this did not prove detailed enough to provide complete guidance on what I needed to complete each day. This led to some development phases overrunning their allocated time. For example, making customer modular controls was not accounted for, leading to the translation of storyboards into final forms being slightly delayed. A burndown chart would be a more suitable solution and would have provided more insight into how the project was progressing.

Secondly, due to my inexperience with the MVP architecture, I experienced several issues. I accidentally created a memory leak, which delayed development by several days as I had to refactor the code. Additionally, my initial MVP code was not strict

enough, which meant I needed to refactor it to achieve loose coupling and separation of concerns. Despite these initial issues, I now feel confident in developing the MVP and would also be assured in working on other projects with this architecture. Furthermore, I did not thoroughly comment the code, meaning it will be harder to understand for other developers.

Finally, my decision to use Google Docs to document the project proved inefficient due to the large memory requirements within the browser. This meant that the application became slower as the documentation grew larger. The document took longer to load, and typing lagged behind the keyboard input. For future projects, I would consider using a more lightweight framework to document the project, perhaps with AsciiDoc, a plain-text system based on a markup language for writing technical content. This would also allow me to more easily manage code within my documentation and allow me to more coherently break my documentation into multiple smaller files that could be easily composed at the end. Additionally, I would have been able to edit the documentation with Vim to leverage improved keyboard shortcuts.

In conclusion, I am happy with my performance overall. I effectively used version control to manage the code base and also overcame challenges such as learning SQL without difficulties. Overall, I could have improved my time management.

Closing Remarks

The application to help manage the Movers microsystem is considered a success for several reasons.

Firstly, the users and stakeholders appear to be happy with the application, which can be seen through the generally very positive overall feedback from the form. The new application also significantly improves how well it is liked compared to the original. In addition to this, there was only one request for more features (more reports). This

Secondly, all user requirements, bar one, have been completely satisfied. This high rate of fulfillment means the system has achieved what was required of it. In addition to this, only one request for more features (more reports) from the survey was made. Due to the modular code base, this can easily be added in future releases of the application. The one partially fulfilled user requirement, key binds, could only be partially fulfilled due to the constraints of Windows Forms.

Thirdly, project management proved to be generally successful, with minimal delays and issues. The project was completed on time and was not delivered late.

Fourthly, the testing process was very thorough, and there should be minimal minor bugs left within the system, if not none at all.

Finally, the Movers' Cleaning Microsystem functions as a standalone system and can easily be incorporated with other microsystems in the future. This has been achieved by using the Model-View-Presenter architect and separating database queries into stored procedures, which are then accessed through the Data Access Layer classes.

In conclusion, the Movers' Cleaning Microsystem Application has been a successful project for the reasons above. I am confident that the microsystem will continue to meet Movers' demands successfully in years to come, and I look forward to future work on the system.

Appendix 1

Developer Diary

Date	Actions
September 23rd - 24th (2024)	Read the case study to understand how Movers works as a company and what problems they face.
September 25th (2024)	Selected the cleaning microsystem.
September 25th - 27th (2024)	Documented current system usage.
September 27th - 30th (2024)	Identified current system issues.
September 30th (2024)	Completed stakeholder identification and analysis.
October 1st - 8th (2024)	Researched and evaluated potential solutions for the identified issues. These included mobile, web and desktop solutions.
October 8th - 10th (2024)	Analysed various project management techniques. Created both a Gantt and Pert chart using Online Gantt and draw.io .
October 11th - 14th (2024)	Looked at design methodologies for system implementation to determine a structured strategy to complete the project (chose waterfall).
October 15th (2024)	The planning phase is completed.
October 15th - 16th (2024)	Normalised data structures for efficient storage and retrieval.
October 17th - 22th (2024)	Created Entity Relationship Diagram (ERD) for database design.
October 23th - 25th (2024)	Developed a data dictionary for all database elements.
October 28th - 30th (2024)	Reviewed database design.

October 30th (2024)	Database design finalised.
October 31st - November 4th (2024)	Selected architectural pattern for system implementation. Chose MVP, but would also have considered MVC.
November 1st - 11th (2024)	Developed class diagrams to model inheritance.
November 8th - 13th (2024)	Created use case diagrams.
November 14th - 27th (2024)	Created storyboards.
November 20th - 28th (2024)	Wrote pseudocode for storyboards.
November 29th (2024)	System design phase completed.
November 29th - December 12th (2024)	Created database structure using SQL within visual studio. Started developing modular user controls.
December 4th - 17th (2024)	Created necessary stored procedures to access database information.
December 22th (2024) - January 24th (2025)	Developed user interface based on storyboards. Discovered memory leak on 14th and spent two days refactoring the presenters. The memory leak occurred due to incorrectly unbinding from events.
December 23th - January 12th	Christmas.
January 13th - February 3rd (2025)	Integrated user interface with backend DAL.
February 3rd (2025)	Implementation completed.
February 4th - 11th (2025)	Created data entry test plan for validation
February 11th - 14th (2025)	Developed link test plan.
February 14th - 18th (2025)	Created user acceptance test plan.
February 19th - 21th (2025)	Performed testing.

February 21st - March 6th (2025)	Fixed bugs discovered during testing. Fortunately, no major changes required
March 7th (2025)	Testing completed.
February 24th - March 4th (2025)	Deployed system for user acceptance testing.
March 5th - 13th (2025)	Collected and evaluated user feedback.
March 13th (2025)	User feedback implementation completed.
March 14th - 19th (2025)	Evaluated fulfillment of user requirements.
March 20th - 25th (2025)	Reviewed effectiveness of project management approach.
March 24th - 27th (2025)	Assessed effectiveness of chosen methodology and evaluated system performance and user satisfaction.
March 27th (2025)	Project evaluation completed. Project finished.

Appendix 2

Code

The code documentation has been included in another document called “code_documentation.md”.

Database

The SQL scripts required to create the database are inserted below. In addition, a file to create the database from one script has been included.

Tables

The tables used within the database are included below.

Staff

```
CREATE TABLE [dbo].[Staff] (
    [Id] INT IDENTITY (1, 1) NOT NULL,
    [Username] NVARCHAR (255) NOT NULL,
    [HashedPassword] NVARCHAR (128) NOT NULL,
    [Salt] NVARCHAR (128) NOT NULL,
    [LastPasswordChange] DATE DEFAULT (getdate()) NOT NULL,
    [Archived] BIT DEFAULT ((0)) NOT NULL,
    [Forename] NVARCHAR (255) NOT NULL,
    [Surname] NVARCHAR (255) NOT NULL,
    [DateOfBirth] DATE NULL,
    [Email] NVARCHAR (255) NOT NULL,
    [PhoneNumber] NVARCHAR (20) NOT NULL,
    [Address] NVARCHAR (255) NOT NULL,
    [EmergencyContactForename] NVARCHAR (100) DEFAULT ('') NOT NULL,
    [EmergencyContactSurname] NVARCHAR (100) DEFAULT ('') NOT NULL,
    [EmergencyContactPhoneNumber] NVARCHAR (20) DEFAULT ('') NOT NULL,
    [PrivilegeLevelID] INT NOT NULL,
    [AppearanceSettings] NVARCHAR (MAX) DEFAULT (N'{"AppearanceTheme": "dark", "ShowToolTips": true, "FontName": "Bahnschrift"}') NOT NULL,
    PRIMARY KEY CLUSTERED ([Id] ASC),
    UNIQUE NONCLUSTERED ([Username] ASC),
    CONSTRAINT [FK_PrivilegeLevel_To_Staff] FOREIGN KEY ([PrivilegeLevelID])
    REFERENCES [dbo].[StaffPrivilege] ([Id]),
    CHECK (isjson([AppearanceSettings])=(1))
);
```

Staff Privilege

```
CREATE TABLE [dbo].[StaffPrivilege] (
    [Id] INT NOT NULL,
```

```

    [PrivilegeLevel] NVARCHAR (50) NOT NULL,
    CONSTRAINT [PK_StaffPrivilege] PRIMARY KEY CLUSTERED ([Id] ASC),
    UNIQUE NONCLUSTERED ([PrivilegeLevel] ASC)
);

```

Login Attempt

```

CREATE TABLE [dbo].[LoginAttempt] (
    [Id]             INT            IDENTITY (1, 1) NOT NULL,
    [Username]       NVARCHAR (255) NOT NULL,
    [AttemptTime]    DATETIME2 (7)  NOT NULL,
    [Successful]    BIT            NOT NULL,
    CONSTRAINT [PK_LoginAttempt] PRIMARY KEY CLUSTERED ([AttemptTime] ASC)
);

```

Stock

```

CREATE TABLE [dbo].[Stock] (
    [Id]             INT            IDENTITY (1, 1) NOT NULL,
    [Name]           NVARCHAR (255) NOT NULL,
    [Description]    NVARCHAR (500) NOT NULL,
    [Archived]       BIT            DEFAULT ((0)) NOT NULL,
    [HighQuantity]   INT            DEFAULT ((0)) NOT NULL,
    [LowQuantity]    INT            DEFAULT ((0)) NOT NULL,
    [Sku]            NVARCHAR (20)  NOT NULL,
    [UnitCost]       DECIMAL (10, 2) DEFAULT ((0)) NOT NULL,
    CONSTRAINT [PK_Stock] PRIMARY KEY CLUSTERED ([Id] ASC),
    UNIQUE NONCLUSTERED ([Sku] ASC)
);

```

Stock Quantity

```

CREATE TABLE [dbo].[StockQuantity] (
    [Id]             INT            IDENTITY (1, 1) NOT NULL,
    [StockId]        INT            NOT NULL,
    [Date]           DATETIME2 (7) NOT NULL,
    [Quantity]       INT            NOT NULL,
    [EditedByStaff]  INT            NOT NULL,
    [ReasonForQuantityChange] NVARCHAR (500) NOT NULL,
    CONSTRAINT [PK_StockQuantity] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_Stock_To_Staff] FOREIGN KEY ([EditedByStaff]) REFERENCES
    [dbo].[Staff] ([Id]),
    CONSTRAINT [FK_StockQuantity_To_Stock] FOREIGN KEY ([StockId]) REFERENCES
    [dbo].[Stock] ([Id])
);

```

Order

```
CREATE TABLE [dbo].[Order] (
    [Id]             INT            IDENTITY (1, 1) NOT NULL,
    [StaffId]        INT            NOT NULL,
    [Description]   NVARCHAR (500) NOT NULL,
    [Status]         NVARCHAR (20)  DEFAULT ('Draft') NOT NULL,
    [Discrepancies] NVARCHAR (500) NOT NULL,
    CONSTRAINT [PK_Order] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_Order_To_Staff_Id] FOREIGN KEY ([StaffId]) REFERENCES
    [dbo].[Staff] ([Id]),
    CHECK ([Status]='Delivered' OR [Status]='Pending' OR [Status]='Rejected' OR
    [Status]='Draft' OR [Status]='Submitted')
);
```

Order Stock

```
CREATE TABLE [dbo].[Order_Stock] (
    [StockId]        INT            NOT NULL,
    [OrderId]        INT            NOT NULL,
    [Quantity]       INT            DEFAULT ((1)) NOT NULL,
    [UnitCostAtTime] DECIMAL (10, 2) NOT NULL,
    CONSTRAINT [PK_Order_Stock] PRIMARY KEY CLUSTERED ([OrderId] ASC, [StockId]
    ASC),
    CONSTRAINT [FK_Order_Stock_To_Order] FOREIGN KEY ([OrderId]) REFERENCES
    [dbo].[Order] ([Id]) ON DELETE CASCADE,
    CONSTRAINT [FK_Order_Stock_To_Stock] FOREIGN KEY ([StockId]) REFERENCES
    [dbo].[Stock] ([Id]) ON DELETE CASCADE
);
```

Order Customer

```
CREATE TABLE [dbo].[Customer] (
    [Id]             INT            IDENTITY (1, 1) NOT NULL,
    [Archived]       BIT            DEFAULT ((0)) NOT NULL,
    [Forename]       NVARCHAR (100) NOT NULL,
    [Surname]        NVARCHAR (100) NOT NULL,
    [Email]          NVARCHAR (255) NOT NULL,
    [PhoneNumber]    NVARCHAR (20)  NOT NULL,
    [Address]        NVARCHAR (500) DEFAULT ('') NOT NULL,
    CONSTRAINT [PK_Customer] PRIMARY KEY CLUSTERED ([Id] ASC),
    UNIQUE NONCLUSTERED ([Email] ASC)
);
```

Customer

```
CREATE TABLE [dbo].[Customer] (
    [Id]             INT            IDENTITY (1, 1) NOT NULL,
    [Archived]       BIT            DEFAULT ((0)) NOT NULL,
    [Forename]       NVARCHAR (100) NOT NULL,
```

```

    [Surname]      NVARCHAR (100) NOT NULL,
    [Email]        NVARCHAR (255) NOT NULL,
    [PhoneNumber]  NVARCHAR (20)  NOT NULL,
    [Address]      NVARCHAR (500) DEFAULT ('') NOT NULL,
    CONSTRAINT [PK_Customer] PRIMARY KEY CLUSTERED ([Id] ASC),
    UNIQUE NONCLUSTERED ([Email] ASC)
);

```

Cleaning Job Option

```

CREATE TABLE [dbo].[CleaningJobOption] (
    [Id]           INT            IDENTITY (1, 1) NOT NULL,
    [Name]         NVARCHAR (255) NOT NULL,
    [Description] NVARCHAR (500)  DEFAULT ('') NOT NULL,
    [UnitCost]     DECIMAL (10, 2) NOT NULL,
    [Archived]    BIT             DEFAULT ((0)) NOT NULL,
    CONSTRAINT [PK_CleaningJobOption] PRIMARY KEY CLUSTERED ([Id] ASC)
);

```

Cleaning Job

```

CREATE TABLE [dbo].[CleaningJob] (
    [Id]           INT            IDENTITY (1, 1) NOT NULL,
    [StartDate]    DATETIME2 (7)  NOT NULL,
    [EndDate]      DATETIME2 (7)  NOT NULL,
    [Address]      NVARCHAR (500) NOT NULL,
    [CustomerId]   INT            NOT NULL,
    [StaffId]      INT            NOT NULL,
    [ExtraInformation] NVARCHAR (500) DEFAULT ('') NOT NULL,
    CONSTRAINT [PK_CleaningJob] PRIMARY KEY CLUSTERED ([Id] ASC),
    CONSTRAINT [FK_CleaningJob_To_Customer] FOREIGN KEY ([CustomerId])
    REFERENCES [dbo].[Customer] ([Id]),
    CONSTRAINT [FK_CleaningJob_To_Staff] FOREIGN KEY ([StaffId]) REFERENCES
    [dbo].[Staff] ([Id])
);

```

Cleaning Job Cleaning Option

```

CREATE TABLE [dbo].[CleaningJob_CleaningJobOption] (
    [CleaningJobId]    INT            NOT NULL,
    [CleaningJobOptionId] INT           NOT NULL,
    [Quantity]          INT            DEFAULT ((1)) NOT NULL,
    [UnitCostAtTime]    DECIMAL (10, 2) NOT NULL,
    CONSTRAINT [PK_CleaningJob_CleaningJobOption] PRIMARY KEY CLUSTERED
    ([CleaningJobId] ASC, [CleaningJobOptionId] ASC),
    CONSTRAINT [FK_CleaningJob_CleaningJobOption_To_CleaningJob] FOREIGN KEY
    ([CleaningJobId]) REFERENCES [dbo].[CleaningJob] ([Id]) ON DELETE CASCADE,
    CONSTRAINT [FK_CleaningJob_CleaningJobOption_To_CleaningJobOption] FOREIGN
    KEY ([CleaningJobOptionId]) REFERENCES [dbo].[CleaningJobOption] ([Id])
);

```

Cleaning Job Staff

```
CREATE TABLE [dbo].[CleaningJob_Staff] (
    [CleaningJobId] INT NOT NULL,
    [StaffId] INT NOT NULL,
    CONSTRAINT [PK_CleaningJobStaff] PRIMARY KEY CLUSTERED ([CleaningJobId] ASC,
    [StaffId] ASC),
    CONSTRAINT [FK_CleaningJobStaff_To_CleaningJob] FOREIGN KEY
    ([CleaningJobId]) REFERENCES [dbo].[CleaningJob] ([Id]) ON DELETE CASCADE,
    CONSTRAINT [FK_CleaningJobStaff_To_Staff] FOREIGN KEY ([StaffId]) REFERENCES
    [dbo].[Staff] ([Id])
);
```

Stored Procedures

AddCleaningJobCleaningOption

```
CREATE PROCEDURE [dbo].[AddCleaningJobCleaningOption]
    @cleaningJobId INT,
    @cleaningJobOptionId INT
AS
BEGIN
    DECLARE @unitCost DECIMAL(10, 2);

    SELECT @unitCost = UnitCost
    FROM CleaningJobOption
    WHERE Id = @cleaningJobOptionId;

    INSERT INTO CleaningJob_CleaningJobOption (CleaningJobId,
    CleaningJobOptionId, UnitCostAtTime)
    VALUES (@cleaningJobId, @cleaningJobOptionId, @unitCost);
END
```

AddCleaningJobStaff

```
CREATE PROCEDURE [dbo].[AddCleaningJobStaff]
    @cleaningJobId INT,
    @staffId INT
AS
    INSERT INTO CleaningJob_Staff (CleaningJobId, StaffId) VALUES
    (@cleaningJobId, @staffId);
```

AddCustomer

```

CREATE PROCEDURE [dbo].[AddCustomer]
    @forename NVARCHAR (100),
    @surname NVARCHAR (100),
    @email NVARCHAR (255),
    @phoneNumber NVARCHAR (20),
    @archived BIT,
    @address NVARCHAR (500)
AS
    INSERT INTO Customer (Forename, Surname, Email, PhoneNumber,
Archived, Address) VALUES (@forename, @surname, @email,
@phoneNumber, @archived, @address);

```

AddOrderStock

```

CREATE PROCEDURE [dbo].[AddOrderStock]
    @orderId INT,
    @stockId INT
AS
BEGIN
    DECLARE @unitCost DECIMAL(10, 2);

    SELECT @unitCost = UnitCost
    FROM Stock
    WHERE Id = @stockId;

    INSERT INTO Order_Stock (OrderId, StockId, UnitCostAtTime)
    VALUES (@orderId, @stockId, @unitCost);
END

```

CountCustomersInFutureJobs

```

CREATE PROCEDURE [dbo].[CountCustomersInFutureJobs]
    @customerId INT
AS
    SELECT COUNT(*) AS [Count] FROM CleaningJob WHERE StartDate
    >= GETDATE() AND CustomerId = @customerId;

```

CreateCleaningJob

```

CREATE PROCEDURE [dbo].[CreateCleaningJob]
    @startDate DATETIME2,

```

```

    @endDate DATETIME2,
    @address NVARCHAR (500),
    @customerId INT,
    @staffId INT,
    @extraInformation NVARCHAR (500)
AS
BEGIN
    INSERT INTO CleaningJob (StartDate, EndDate, Address,
CustomerId, StaffId, ExtraInformation) VALUES (@startDate,
@endDate, @address, @customerId, @staffId, @extraInformation);

    SELECT * FROM CleaningJob WHERE Id = SCOPE_IDENTITY();
END

```

CreateCleaningJobOption

```

CREATE PROCEDURE [dbo].[CreateCleaningJobOption]
    @name NVARCHAR (255),
    @description NVARCHAR (500),
    @unitCost DECIMAL (10, 2),
    @archived BIT
AS
    INSERT INTO CleaningJobOption (Name, Description, UnitCost,
Archived) OUTPUT Inserted.Id VALUES (@name, @description,
@unitCost, @archived);

```

CreateOrder

```

CREATE PROCEDURE [dbo].[CreateOrder]
    @status NVARCHAR (20),
    @discrepancies NVARCHAR (500),
    @description NVARCHAR (500),
    @staffId INT
AS
BEGIN
    INSERT INTO [Order] (Status, Discrepancies, Description,
StaffId) VALUES (@status, @discrepancies, @description, @staffId);

    SELECT SCOPE_IDENTITY() AS Id;
END

```

CreateStaff

```
CREATE PROCEDURE [dbo].[CreateStaff]
    @username NVARCHAR(255),
    @hashedPassword NVARCHAR(128),
    @salt NVARCHAR(128),
    @forename NVARCHAR(255),
    @surname NVARCHAR(255),
    @dateOfBirth DATE,
    @email NVARCHAR(255),
    @phoneNumber NVARCHAR(15),
    @address NVARCHAR(256),
    @emergencyContactForename NVARCHAR(255),
    @emergencyContactSurname NVARCHAR(255),
    @emergencyContactPhoneNumber NVARCHAR(255),
    @privilegeLevelId INT
AS
    INSERT INTO Staff (
        Username,
        HashedPassword,
        Salt,
        Forename,
        Surname,
        DateOfBirth,
        Email,
        PhoneNumber,
        Address,
        EmergencyContactForename,
        EmergencyContactSurname,
        EmergencyContactPhoneNumber,
        PrivilegeLevelId
    )
    VALUES (
        @username,
        @hashedPassword,
        @salt,
        @forename,
        @surname,
        @dateOfBirth,
        @email,
        @phoneNumber,
        @address,
        @emergencyContactForename,
```

```

    @emergencyContactSurname,
    @emergencyContactPhoneNumber,
    @privilegeLevelId
);

```

CreateStock

```

CREATE PROCEDURE [dbo].[CreateStock]
    @name NVARCHAR(255),
    @description NVARCHAR(500),
    @sku VARCHAR(20),
    @quantity INT,
    @staffId INT,
    @highQuantity INT,
    @lowQuantity INT,
    @archived BIT,
    @date DATE,
    @reasonForQuantityChange NVARCHAR (500),
    @unitCost DECIMAL(10,2)
AS
    INSERT INTO Stock (Name, Description, Sku, HighQuantity,
LowQuantity, Archived, UnitCost) VALUES (@name, @description,
@sku, @highQuantity, @lowQuantity, @archived, @unitCost);
    INSERT INTO StockQuantity (StockId, Date, Quantity,
EditedByStaff, ReasonForQuantityChange) VALUES (SCOPE_IDENTITY(),
@date, @quantity, @staffId, @reasonForQuantityChange);

```

DeleteAllCleaningJobCleaningJobOptions

```

CREATE PROCEDURE [dbo].[DeleteAllCleaningJobCleaningJobOptions]
    @id INT
AS
    DELETE FROM CleaningJob_CleaningJobOption WHERE CleaningJobId
= @id;

```

DeleteAllCleaningJobStaff

```

CREATE PROCEDURE [dbo].[DeleteAllCleaningJobStaff]
    @id INT
AS
    DELETE FROM CleaningJob_Staff WHERE CleaningJobId = @id;

```

DeleteAllOrderStock

```
CREATE PROCEDURE [dbo].[DeleteAllOrderStock]
    @id INT
AS
    DELETE FROM Order_Stock WHERE OrderId = @id;
```

DeleteCleaningJob

```
CREATE PROCEDURE [dbo].[DeleteCleaningJob]
    @id INT
AS
    DELETE FROM CleaningJob WHERE Id = @id;
```

DeleteOrder

```
CREATE PROCEDURE [dbo].[DeleteOrder]
    @id INT
AS
    DELETE FROM [Order] WHERE Id = @id;
```

GetCleaningJobById

```
CREATE PROCEDURE [dbo].[GetCleaningJobById]
    @id INT
AS
    SELECT * FROM CleaningJob WHERE Id = @id;
```

GetCleaningJobCleaningOptionIds

```
CREATE PROCEDURE [dbo].[GetCleaningJobCleaningOptionIds]
    @id INT
AS
    SELECT * FROM CleaningJob_CleaningJobOption WHERE
CleaningJobId = @id;
```

GetCleaningJobCleaningOptions

```
CREATE PROCEDURE [dbo].[GetCleaningJobCleaningOptions]
    @id INT
AS
    SELECT cjo.*, Quantity, UnitCostAtTime FROM
CleaningJob_CleaningJobOption INNER JOIN CleaningJobOption cjo ON
CleaningJobOptionId = cjo.Id WHERE CleaningJobId = @id;
```

GetCleaningJobOptions

```
CREATE PROCEDURE [dbo].[GetCleaningJobOptions]
AS
    SELECT * FROM CleaningJobOption;
```

GetCleaningJobs

```
CREATE PROCEDURE [dbo].[GetCleaningJobs]
AS
    SELECT * FROM CleaningJob;
```

GetCleaningJobsAfterDate

```
CREATE PROCEDURE [dbo].[GetCleaningJobsAfterDate]
    @date DATE
AS
    SELECT * FROM CleaningJob WHERE CAST(StartDate as DATE) >=
@date;
```

GetCleaningJobsByDate

```
CREATE PROCEDURE [dbo].[GetCleaningJobsByDate]
    @date DATE
AS
    SELECT * FROM CleaningJob WHERE CAST(StartDate as DATE) =
@date;
```

GetCleaningJobStaffIds

```
CREATE PROCEDURE [dbo].[GetCleaningJobStaffIds]
    @id INT
AS
    SELECT * FROM CleaningJob_Staff WHERE CleaningJobId = @id;
```

GetCustomerById

```
CREATE PROCEDURE [dbo].[GetCustomerById]
    @id INT
AS
    SELECT * FROM Customer WHERE Id = @id;
```

GetCustomers

```
CREATE PROCEDURE [dbo].[GetCustomers]
AS SELECT * FROM Customer;
```

GetjobOptionByName

```
CREATE PROCEDURE [dbo].[GetJobOptionByName]
@name NVARCHAR (255)
AS
SELECT * FROM CleaningJobOption WHERE Name = @name;
```

GetLoginAttempts

```
CREATE PROCEDURE [dbo].[GetLoginAttempts]
AS
SELECT TOP 100 * FROM LoginAttempt ORDER BY AttemptTime DESC;
```

GetNonArchivedCleaningJobOptions

```
CREATE PROCEDURE [dbo].[GetNonArchivedCleaningJobOptions]
AS
SELECT * FROM CleaningJobOption WHERE Archived = 0;
```

GetNonArchivedCustomers

```
CREATE PROCEDURE [dbo].[GetNonArchivedCustomers]
AS
SELECT * FROM Customer WHERE Archived = 0;
```

GetNonArchivedStock

```
CREATE PROCEDURE [dbo].[GetNonArchivedStock]
AS
SELECT * FROM Stock WHERE Archived = 0;
```

GetOrders

```
CREATE PROCEDURE [dbo].[GetOrders]
AS
SELECT * FROM [Order];
```

GetOrderStockItems

```
CREATE PROCEDURE [dbo].[GetStaff]
```

AS

```
    SELECT Staff.Id, Staff.HashedPassword, Staff.Salt,
Staff.LastPasswordChange, Staff.Forename, Staff.Surname,
Staff.DateOfBirth, Staff.Email, Staff.PhoneNumber,
Staff.EmergencyContactForename, Staff.EmergencyContactSurname,
Staff.EmergencyContactPhoneNumber, Staff.Address, Staff.Username,
StaffPrivilege.PrivilegeLevel, Staff.Archived,
Staff.AppearanceSettings
    FROM Staff
    INNER JOIN StaffPrivilege ON
Staff.PrivilegeLevelId=StaffPrivilege.Id;
```

GetStaffById

```
CREATE PROCEDURE [dbo].[GetStaffById]
    @id int
AS
    SELECT * FROM Staff s INNER JOIN StaffPrivilege sp ON
s.PrivilegeLevelId = sp.Id WHERE s.Id = @id;
```

GetStaffByPrivilege

```
CREATE PROCEDURE [dbo].[GetStaffByPrivilege]
    @privilegeLevelId INT
AS
    SELECT * FROM Staff WHERE PrivilegeLevelId =
@privilegeLevelId;
```

GetStaffByUsername

```
CREATE PROCEDURE [dbo].[GetStaffByUsername]
    @username NVARCHAR (255)
AS
    SELECT Staff.Id, Staff.HashedPassword, Staff.Salt,
Staff.LastPasswordChange, Staff.Forename, Staff.Surname,
Staff.DateOfBirth, Staff.Email, Staff.PhoneNumber,
Staff.EmergencyContactForename, Staff.EmergencyContactSurname,
Staff.EmergencyContactPhoneNumber, Staff.Address, Staff.Username,
StaffPrivilege.PrivilegeLevel, Staff.Archived,
Staff.AppearanceSettings
    FROM Staff
    INNER JOIN StaffPrivilege ON
Staff.PrivilegeLevelId=StaffPrivilege.Id
```

```
WHERE [Username] = @username;
```

GetStaffCredentails

```
CREATE PROCEDURE [dbo].[GetStaffCredentials]
    @username NVARCHAR (255)
AS
BEGIN
    SELECT [HashedPassword], [Salt] FROM [dbo].[Staff] WHERE
    ([Username]) = @username AND ([Archived]) = 0;
END
```

GetStock

```
CREATE PROCEDURE [dbo].[GetStock]
AS
SELECT
    s.Id,
    s.Name,
    s.Description,
    s.Sku,
    s.Archived,
    s.LowQuantity,
    s.HighQuantity,
    sq.Quantity,
    sq.Date,
    s.UnitCost
FROM
    Stock s
INNER JOIN
    StockQuantity sq
ON
    s.Id = sq.StockId
AND
    sq.Id = (
        SELECT MAX(Id)
        FROM StockQuantity
        WHERE StockId = s.Id
    )
ORDER BY s.Id;
```

GetStockBySku

```
CREATE PROCEDURE [dbo].[GetStockBySku]
    @Sku NVARCHAR (20)
AS
    SELECT * FROM Stock WHERE Sku = @Sku;
```

GetStockQuantityChanges

```
CREATE PROCEDURE GetStockQuantityChanges
AS
    SELECT sqc.Id, sqc.Quantity, sqc.ReasonForQuantityChange,
    sqc.Date, sqc.StockId, s.Name AS StockName, s.Sku AS StockSku,
    s Archived AS StockArchived, sqc.EditedByStaff, st.Forename AS
    StaffForename, st.Surname AS StaffSurname, st.Username AS
    StaffUsername
        FROM StockQuantity sqc
        INNER JOIN Stock s ON sqc.StockId = s.Id
        INNER JOIN Staff st ON sqc.EditedByStaff = st.Id;
```

GetUpcomingCleaningJobs

```
CREATE PROCEDURE [dbo].[GetUpcomingCleaningJobs]
AS
    SELECT * FROM CleaningJob WHERE [StartDate] >= GETDATE();
```

LogLoginAttempt

```
CREATE PROCEDURE [dbo].[LogLoginAttempt]
    @username NVARCHAR (255),
    @attemptTime DATETIME2,
    @success BIT
AS
    INSERT INTO LoginAttempt (Username, AttemptTime, Successful)
    VALUES (@username, @attemptTime, @success);
```

UpdateCleaningJobCleaningOptionQuantity

```
CREATE PROCEDURE [dbo].[UpdateCleaningJobCleaningOptionQuantity]
    @cleaningJobId INT,
    @cleaningJobOptionId INT,
    @Quantity INT
AS
    UPDATE CleaningJob_CleaningJobOption SET Quantity = @Quantity
```

```
WHERE CleaningJobId = @cleaningJobId AND CleaningJobOptionId =  
@cleaningJobOptionId;
```

UpdateCleaningJobCustomer

```
CREATE PROCEDURE [dbo].[UpdateCleaningJobCustomer]  
    @id INT,  
    @customerId INT  
AS  
    UPDATE CleaningJob SET CustomerId = @customerId WHERE Id =  
@id;
```

UpdateCleaningJobDetails

```
CREATE PROCEDURE [dbo].[UpdateCleaningJobDetails]  
    @id INT,  
    @address NVARCHAR (500),  
    @extraInformation NVARCHAR (500)  
AS  
    UPDATE CleaningJob SET Address = @address, ExtraInformation =  
@extraInformation WHERE Id = @id;
```

UpdateCleaningJobDetails

```
CREATE PROCEDURE [dbo].[UpdateCleaningJobOptionArchived]  
    @id INT,  
    @archived BIT  
AS  
    UPDATE CleaningJobOption SET Archived = @archived WHERE Id =  
@id;
```

UpdateCleaningJobOptionDetails

```
CREATE PROCEDURE [dbo].[UpdateCleaningJobOptionDetails]  
    @id INT,  
    @name NVARCHAR (255),  
    @description NVARCHAR (500),  
    @unitCost DECIMAL (10, 2)  
AS  
    UPDATE CleaningJobOption SET Name = @name, Description =  
@description, UnitCost = @unitCost WHERE Id = @id;
```

UpdateCleaningJobTimes

```
CREATE PROCEDURE [dbo].[UpdateCleaningJobTimes]
    @id INT,
    @startDate DATETIME2,
    @endDate DATETIME2
AS
    UPDATE CleaningJob SET StartDate = @startDate, EndDate =
    @endDate WHERE Id = @id;
```

UpdateCustomerArchived

```
CREATE PROCEDURE [dbo].[UpdateCustomerArchived]
    @id INT,
    @archived BIT
AS
    UPDATE Customer SET Archived = @archived WHERE Id = @id
```

UpdateCustomerContactDetails

```
CREATE PROCEDURE [dbo].[UpdateCustomerContactDetails]
    @id INT,
    @email NVARCHAR (255),
    @phoneNumber NVARCHAR (20),
    @address NVARCHAR (500)
AS
    UPDATE Customer SET PhoneNumber = @phoneNumber, Email =
    @email, Address = @address WHERE Id = @id;
```

UpdateCustomerPersonalDetails

```
CREATE PROCEDURE [dbo].[UpdateCustomerPersonalDetails]
    @id INT,
    @forename NVARCHAR (100),
    @surname NVARCHAR (100),
    @archived BIT
AS
    UPDATE Customer SET Forename = @forename, Surname = @surname,
    Archived = @archived WHERE Id = @id;
```

UpdateOrderDescription

```
CREATE PROCEDURE [dbo].[UpdateOrderDescription]
    @id INT,
```

```
    @description NVARCHAR(500)
AS
    UPDATE [Order] SET Description = @description WHERE Id = @id
```

UpdateOrderDiscrepancies

```
CREATE PROCEDURE [dbo].[UpdateOrderDiscrepancies]
    @id INT,
    @discrepancies NVARCHAR (500)
AS
    UPDATE [Order] SET Discrepancies = @discrepancies WHERE Id =
@id;
```

UpdateOrderStatus

```
CREATE PROCEDURE [dbo].[UpdateOrderStatus]
    @id INT,
    @status NVARCHAR(20)
AS
    UPDATE [Order] SET Status = @status WHERE Id = @id;
```

UpdateOrderStockQuantity

```
CREATE PROCEDURE [dbo].[UpdateOrderStockQuantity]
    @orderId INT,
    @stockId INT,
    @quantity INT
AS
    UPDATE Order_Stock SET Quantity = @quantity WHERE OrderId =
@orderId AND StockId = @stockId;
```

UpdateStaffAppearanceSettings

```
CREATE PROCEDURE [dbo].[UpdateStaffAppearanceSettings]
    @appearanceSettings NVARCHAR (MAX),
    @id int
AS
    UPDATE Staff SET AppearanceSettings = @appearanceSettings
WHERE Id = @id;
```

UpdateStaffArchived

```
CREATE PROCEDURE [dbo].[UpdateStaffArchived]
```

```

@Id INT,
@archived BIT
AS
    UPDATE Staff SET Archived = @archived WHERE Id = @Id;

```

UpdateStaffContactDetails

```

CREATE PROCEDURE [dbo].[UpdateStaffContactDetails]
    @email NVARCHAR (255),
    @phoneNumber NVARCHAR (255),
    @address NVARCHAR (255),
    @staffId int
AS
    UPDATE Staff SET Email = @email, PhoneNumber = @phoneNumber,
Address = @address WHERE Id = @staffId

```

UpdateStaffCredentials

```

CREATE PROCEDURE [dbo].[UpdateStaffCredentials]
    @id INT,
    @privilegeLevel INT,
    @username NVARCHAR (255)
AS
    UPDATE Staff SET PrivilegeLevelId = @privilegeLevel, Username
= @username WHERE Id = @id;

```

UpdateStaffEmergencyContact

```

CREATE PROCEDURE [dbo].[UpdateStaffEmergencyContact]
    @emergencyContactForename NVARCHAR (100),
    @emergencyContactSurname NVARCHAR (100),
    @emergencyContactPhoneNumber NVARCHAR (20),
    @id int
AS
    UPDATE Staff SET EmergencyContactForename =
@emergencyContactForename, EmergencyContactSurname =
@emergencyContactSurname, EmergencyContactPhoneNumber =
@emergencyContactPhoneNumber WHERE Id = @id

```

UpdateStaffPassword

```

CREATE PROCEDURE [dbo].[UpdateStaffPassword]
    @hashedPassword NVARCHAR (128),

```

```

@salt NVARCHAR (128),
@lastPasswordChange DATE,
@Id int
AS
    UPDATE Staff SET HashedPassword = @hashedPassword, Salt =
@salt, LastPasswordChange = @lastPasswordChange WHERE Id = @Id

```

UpdateStaffPersonalinformation

```

CREATE PROCEDURE [dbo].[UpdateStaffPersonalInformation]
    @dateOfBirth DATE,
    @forename NVARCHAR (255),
    @surname NVARCHAR (255),
    @id int
AS
    UPDATE Staff SET DateOfBirth = @dateOfBirth, Forename =
@forename, Surname = @surname WHERE Id = @id;

```

UpdateStockArchived

```

CREATE PROCEDURE [dbo].[UpdateStockArchived]
    @id INT,
    @archived BIT
AS
    UPDATE Stock SET Archived = @archived WHERE Stock.Id = @id;

```

UpdateStockDetails

```

CREATE PROCEDURE [dbo].[UpdateStockDetails]
    @id INT,
    @name NVARCHAR (256),
    @description NVARCHAR (1000),
    @sku NVARCHAR (15),
    @archived BIT
AS
    UPDATE Stock SET Name = @name, Description = @description,
Sku = @sku, Archived = @archived WHERE Id = @id;

```

UpdateStockQuantity

```

CREATE PROCEDURE [dbo].[UpdateStockQuantity]
    @date DATE,
    @staffId INT,

```

```
@reasonForQuantityChange NVARCHAR (1000),  
@quantity INT,  
@stockId INT  
AS  
    INSERT INTO StockQuantity (StockId, Date, Quantity,  
EditedByStaff, ReasonForQuantityChange) VALUES (@stockId, @date,  
@quantity, @staffId, @reasonForQuantityChange);
```

```
CREATE PROCEDURE [dbo].[UpdateStockUnitCost]  
    @id INT,  
    @unitCost DECIMAL(10, 2)  
AS  
    UPDATE Stock SET UnitCost = @unitCost WHERE Id = @id
```

UpdateStockWarnings

```
CREATE PROCEDURE [dbo].[UpdateStockWarnings]  
    @id INT,  
    @lowQuantity INT,  
    @highQuantity INT  
AS  
    UPDATE Stock SET LowQuantity = @lowQuantity, HighQuantity =  
@highQuantity WHERE Id = @id;
```