

The background of the slide is a photograph of a busy port. A large container ship, the 'YANKI A VALLETTA', is docked at a pier. Several large gantry cranes are visible, some with 'NLE' branding. The ship's hull is blue and white, and it is loaded with many colorful shipping containers. The sky is blue with some clouds.

Report #3

(System Specification and Design)



Berthing Plan/Scheduling Application

Team Members:

- Osborn Collins
- Mark A. Pascual
- Kevin Godoy
- Daniel Garcia
- Driane Perez
- Justin Chuc

CMPS3141 Software Engineering
Submitted to Mr. Manuel Medina
Submission Date: 16th May 2022

Distribution of Work



<u>Signature Block</u>			
Statement	<i>I did my share of the work, and have a general understanding of the contents of the assignment.</i>		
<u>Team Member</u>	<u>Contribution</u>	<u>Signature</u>	<u>Date</u>
Osborn Collins	<ul style="list-style-type: none"> • Customer Statement of Requirements • System Requirements • UC 8 • Small part of Stakeholders • Small part of actors and goals • Glossary • Domain Model • Part of UI Development 	O.C.	16th May, 2022
Driane Perez	<ul style="list-style-type: none"> • 93% of the Casual Description • 100% of the Traceability Matrix • 95 % of the Use Case Diagram • 100% of the work plan • Fully - Dressed Description review and added about 5% 		16th May, 2022
Daniel Garcia	<ul style="list-style-type: none"> • System architecture • Architecture Styles • Mapping Subsystems to Hardware • Connectors and Network 		16th May, 2022

	<ul style="list-style-type: none"> • Protocols • Global Control Flow • Hardware requirements 		
Justin Chuc	<ul style="list-style-type: none"> • Connectors and Network • Use Case • Use Case Diagram Dressed/ Fully Dressed 	<i>J. Chuc</i>	16th May, 2022
Kevin Godoy	<ul style="list-style-type: none"> • Mapping subsystems to hardware • Client Requirements • Preliminary Designs • User Estimation • Website 	<i>Kevin Godoy</i>	16th May, 2022
Mark Pascual	<ul style="list-style-type: none"> • Plan of work • References • Some editing • 2 Fully Dressed Use Case • Actors and Goals • Some Casual Description • Summary of changes • History of Work • 	<i>Mark A. Pascual</i>	16th May, 2022

Distribution of work

Note: See point allocations in the table below.		Team Member Name					
Responsibility Level	See point allocations in the table below.	Osborn Collins	Mark Pascual	Daniel Garcia	Kevin Godoy	Justin Chuc	Driane Perez
	Summary of Changes (5 Points)		100%				
	1. Customer Statement of Requirements (6 Points)	100%					
	2. Glossary of Terms (4 Points)	100%					
	3. System Requirements (6 Points)	70 %	15%				15 %
	4. Functional Requirements Specification (30 Points)	10%	40%				50 %
	5. Effort Estimation (4 Points)	###?	###?	###?	###?		###?
	6. Domain Analysis (25 Points)	50%					50%
	7. Interaction Diagrams (30 Points)	50%					50%
	<i>Design Patterns</i> (10 Points)	100%					
	8. Class Diagram and Interface Specification (10 Points)		30%	60%		10%	
	<i>OCL Contract Specification</i> (10 Points)			50%	50%		
	9. System Architecture and System Design (15 Points)			100%			
	10. Algorithms and Data Structures (4 Points)				20%	80%	
	11. User Interface Design and Implementation (11 Points)	100%					

	12. Design of Tests (12 Points)			90%		10%	
	14. History of Work, Current Status, and Future Work (5 Points)		100%				
	15. References (negative points if missing or inadequate) (5 Points)		100%				
	PROJECT MANAGEMENT (13 Points)	10%	90%				

<u>Signature Block</u>			
Statement	<i>I did my share of the work, and have a general understanding of the contents of the assignment.</i>		
<u>Team Member</u>	<u>Contribution</u>	<u>Signature</u>	<u>Date</u>
Osborn Collins	50% Domain Analysis . 50 % Interaction Diagrams. Created Domain analysis diagram along with Driane Perez. Created Interaction Diagrams, System Operation Contract, Mathematical Model, Persistent Data Storage	O.C	6th April, 2022
Driane Perez	50% Domain Analysis . 50 % Interaction Diagrams. Created Domain analysis diagram along with Osborn Collins. Created Interaction Diagrams, System Operation Contract, Mathematical Model, Persistent Data Storage		6th April, 2022
Daniel Garcia	Class diagram 60% Design of Tests 90%		6th April, 2022

Justin Chuc	80%Algorithms & Data Structures 10%Class Diagram 10%Test Case	<i>Justin Chuc</i>	6th April, 2022
Kevin Godoy	100% User interface 40% Algorithms & Data Structures	<i>Kevin Godoy</i>	6th April, 2022
Mark Pascual	Class diagram, interface specification and Traceability Matrix (30%) Project Management and Plan of work History of work, current status and Future work Table of Contents Part of the proofreading and editing of the final document. References Some functional requirements and system requirements. Summary of Changes	<i>Mark A. Pascual</i>	6th April, 2022

Table of contents

Customer Requirement

- Problem statement.....9
- Glossary of terms.....12

System Requirements

- Enumerated Functional Requirements.....13
- Enumerated Non-Functional Requirement.....14
- On-screen appearance Requirements.....16

Functional Requirements

- Stakeholders.....17
- Actors and Goals.....17
- Use Cases
 - Casual Description.....22
 - Use Case Diagram.....27
 - Traceability Matrix.....28
 - Fully-Dressed Descriptions.....31
- System Sequence Diagrams

Domain Analysis (Analysis and Domain Modeling)

- Conceptual Model
 - Concept Definitions.....37
 - Association Definitions.....39
 - Attribute Definitions.....40
 - Traceability Matrix.....41
- System Operation Contracts.....42
- Data Model and Persistent Data Storage.....45
- Mathematical Model.....47

Interaction Diagrams.....51

Class Diagram and Interface Specification

- Class Diagram.....60
- Data types and Operation Signatures.....61
- Traceability Matrix.....62

System Architecture and System Design

- Identifying Subsystem.....64
- Architecture Styles.....64

• Mapping Subsystems to Hardware.....	64
• Connectors and Network Protocols.....	65
• Global Control Flow.....	65
• Hardware Requirements.....	65
• Client Requirements.....	65
• Mobile.....	65
• Server.....	65
Algorithms and Data Structures	
• Algorithms.....	66
• Data Structures.....	73
Design Patterns.....	74
OCL Contract Specifications	75
User Interface Design and Implementation.....	78
Design of Tests.....	89
History of Work, Current Status, and Future Work.....	93
Project Management and Plan of Work.....	97
1. Merging contributions from individual team members	
2. Project Coordination and progress report	
3. Plan of work (gant chart)	
4. Break down of responsibilities	
Summary of Changes.....	100
References.....	101

Customer Statement Requirement

Problem Statement

Ports are incredibly complex industries that thrive on efficient operations and supply chains. By ensuring that there is no lag between the vessel entering the port, having its cargo loaded or unloaded, refueled and inspected, cleared to depart, and finally pushed out of the berth, these ports allow countries all around the world to bring in vital trade and commerce to the region.

Most ports these days have the latest technology in order to reduce this turnaround time. For instance, the Port of Rotterdam which is the largest port in Europe and 11th worldwide is renowned for the immense importance it has placed on technological advancements. This port is supported by world class machinery such as specialized truck trailers, automated loading system and advanced loading crane. These improvements are a handful of common features implemented throughout a port. However, this technology can only speed up the process once the ship enters the port.

How do ports efficiently process ships waiting to enter or leave the port? This is where the concept of a berthing plan is introduced. Berthing Plans are detailed documents drawn up regarding the complete resource allocation for a port and surrounding facilities. It implements a supply chain and operation plan for all vessels that will dock or berth at the port at least a month before and may go up to 6 months. They are integral in the allocation of a port's resources for both incoming and outgoing vessels by planning ahead.

We have been using an excel sheet to basically plan for our berth. Due to limitations with this option, we are interested in sourcing a more modernized system that will phase out this tedious manual process. The system should help to optimize the Port operations for arriving and departing vessels. This system will be used by a wide variety of employees, for example, employees from the operations department right down to the persons from the mechanical department and as a result it will need to be very user friendly and easy to learn. Due to the fact we will have a wide range of employees using the application, we just want to emphasize that we don't need a system with a ton of features and complexity that will make the learning curve be so

high that it's a burden to be use by an employee, while they are doing the jobs, they were tasked with doing.

First, on a weekly basis a ship schedule in PDF format is manually generated, this schedule basically shows the estimated arrival and departure time of a vessel. This gets the job done, but it causes different delays because we are only planning one week at a time when effective planning can be done for longer periods of time, in some cases, up to six months in advance.. Managing vessels' arrival and departure time is very important because vessels cannot be scheduled to dock simultaneously at our berth because we only have one berth and we wouldn't have the adequate facilities for it. The schedule should be staggered so that ships are able to minimize their waiting times. This by default improves efficiency and reduces strains on the port resources. The main thing we are trying to avoid is a bottleneck.

Secondly, the ship schedules in our current process do not automatically adjust to show recent updates. Another important objective of the port is a reduction in the unexpected arrival or departure of a ship. This throws the entire system into disarray, as it may further delay other ships. For instance, if a ship arrives earlier than expected, it will have a large idling and waiting time that reduces its efficiency. On the other hand, if a ship arrives late, it will also depart late, which creates a propagating delay that spreads to other ships in the port. So, since the schedule is fixed and cannot be updated in real time, there is really no way to minimize early/delayed arrivals and departures of a vessel.

Thirdly, since the schedule is a PDF, the only notification we can send is an email and it makes it very difficult to notify people who aren't around a computer regularly. For instance, if an employee is working out in the field they would not readily have access to updated information about any changes made in the schedule until they return from working out. In such instances there could be many changes to the schedule. So, it's important that everyone working a vessel is notified immediately of any changes made to the schedule.

With these issues in mind, we are requesting for your organization to provide some feedback and present some solutions VIA a system that would help to solve these issues we have described.

We would like to be assigned a project manager to help convert this manual archaic process to a more automated system. The ship schedule should be the main dashboard and the starting point for all operations. This basically means that the ship schedule will need to be dynamic and showing any changes made in real-time, for example, by changing the estimated arrival time of one vessel all other vessels following will be adjusted by either moving up to an earlier time or down to a later time. With this in place this will help to have all resources used in an optimized way.

We expect the system to provide us notification of any changes made in the system, for example if a supervisor, boat-man, foreman and stevedore is scheduled to work on a vessel, they will be sent a notification of when the ship should arrive and what time work needs to begin. If the vessel's time was adjusted and it is showing that it will reach later or earlier, they will get a notification.

One last thing we didn't mention initially but we would like you to keep in mind, is that changes can be tracked in real time and send notification so that everyone knows what's happening and to ensure delay is at a minimum. From that point all data should be compiled into a report. The main objective of working a vessel is to unload and load a vessel as quickly as possible. Therefore, tracking delays would help to show where it went wrong and how to improve for the next vessel.

At the end of the day, the overall objective of the berthing plan app is to efficiently create a timetable of ship arrival and departure, and a list of facilities and services that will have to be provided to a berthed vessel. By optimizing this timetable, the port can achieve the most efficient turnaround time for each vessel.

Glossary of Terms

Terms	Definition
Berth	A ship's allotted place at a wharf or dock
Vessel	A ship or large boat.
Anchorage	An area that is suitable for a ship to anchor in.
Stacker/Crane	A machine that stacks containers in a Port Yard Space.
Port	A town or city with a harbor where ships load or unload, especially one where customs officers are stationed.

System Requirements

Ranking Scheme: From 1 -4 with 1 having the most priority and 4 having the least priority.

Enumerated Functional Requirements

ID	Priority	Requirements
Req1	1	Depending on the user designated dashboard will be shown as well the details of the vessel's duration of up to six months. The system shall provide the user with digital instances of all vessels that will berth for up to 6 months. It should be done in a table format.
Req5	2	The system shall notify designated users of any changes by displaying messages in their respective dashboards. Users shall click to see the notification form the changes that were made by another user. The messaging or notifying tab will be used where the users will see the changes made.
Req6	2	The system and the designated users should allocate all persons, and equipment for working the vessel. Team leaders will be able to assign personnel with the relevant equipment to off load and load a vessel. As soon as the workers for a vessel have been updated the workers will be notified via their respective dashboard with the use of the notifying tab.
Req7	2	Users logged onto the berthing application will be able to view the operations of a vessel via their dashboard and be able to ensure that operations are optimized by viewing updates on the dashboard.
Req8	3	Users, via their dashboard, will be able to generate reports using data stored in the database in order to analyze trends and make reports.
Req9	4	Users will be allowed to see the different status of the vessels that are currently in the port.
Req10	3	Users should be able to be manipulated through the graphical portion and through a manual back end screen.

Req28	1	Users are able to provide their credentials(username/email and password) and once accurate gain access to their dashboard and other features of the berthing application.
Req29	2	Users, on their dashboard, will have a logout button. Upon clicking this button all relevant information the user was working on will be saved and the user will be logged off the system.
Req30	1	Users, once logged in, will be able to view the calendar/schedule of vessels and make updates to schedules whether by adding or deleting.
Req31	1	Administrators, whilst logged in, will be able to update the application.
Req32	1	Administrators, whilst logged in, will be able to add/create users and grant them access to the berthing application.
Req33	1	Whilst working a user will be able to initiate a delay by activating the appropriate button on the dashboard.
Req34	1	Once logged in, a show will have access to generate reports based on their interactions for the day.

Enumerated Nonfunctional Requirements

ID	Priority	Requirements
Req11	3	Only authenticated users will be able to access the system.
Req12	3	The user access will be determined by the user roles and permissions associated with the role.

Req13	1	The system should have some level of intelligence where it is able to track information and report data accurately.
Req14	3	The system will allow user management.
Req15	1	Must be easy to learn and user friendly.
Req16	1	System should not allow any clashing of vessels berthing at the same time.
Req17	2	System should have an auto schedule feature.
Req18	2	Movement in the system should be very easy to follow
Req26	1	The system will keep track of operations, and always ensure that operations are working as optimized as possible. Calculations will be made determining the best possible solution was taken where no time is wasted or the least time was wasted. Comparing the time with the other time with the other vessels.
Req27	1	System should keep as much data as possible to analyze trends and for reporting.
Req2	1	The system shall have all ships in a queue for up to 6 months.
Req3	1	The system should allow the ship schedule to be dynamic and easy to update.
Req4	1	The system will allow user management.

On-Screen Appearance Requirements

ID	Priority	Requirements
Req19	1	The system should be user friendly with a small margin for errors.
Req20	1	The system should display the ship schedule as the main dashboard for the system.
Req21	2	Each vessel listed to berth will display voyage number, vessel name, ETA, ATA, ETD, ATD, operations personnel working and stevedore gang working the vessel.
Req22	3	Notification will be displayed as a message showing up in an inbox like tab.
Req23	4	Vessels will be displayed in various colors to differentiate between ships in the queue.
Req24	3	The system should display delays in minutes on the main dashboard.
Req25	3	The system should be optimized for touching because the schedule would operate similar to a grid in excel.

Functional Requirements

Stakeholders

1. System Administrator
2. CEO & Senior Management
3. Operations Manager
4. Operations Supervisor
5. Operations Team
6. Gang Foreman
7. Gang(Stevedore)
8. Machine Operators
9. Marine Team
10. Pilot
11. Shipping Agents
12. Technical Team (Mechanical, Maintenance, Electrical, Welding etc)

Actors and Goals

Actors	Roles	Types	Goals
System	<ul style="list-style-type: none">● Providing login for the various stakeholders (System Administrator, CEO & Senior Management, Operations Manager, Operations Supervisor, Operations Team, Gang Foreman, Gang(Stevedore), Machine Operators, Marine Team, Pilot, Shipping Agents, Technical Team)● Provide a tailored dashboard for the various users allowing them to see status of ships, report incidents and initiate delay timer.● Provide a weekly/monthly calendar and schedules	Participating	<ul style="list-style-type: none">● Allow login of the various users to the application once their credentials are accurate.● Update calendar and berthing schedules in real time● Allow administrators to make adjustments● Display accurate schedules.

	<p>of ships scheduled to berth.</p> <ul style="list-style-type: none"> • Allowing admin to add users, update status and modify ship schedules • Track delays and inform appropriate personnel. • Access will be determined on the role you have. Example an agent should only see schedules, and make changes to the ETA and ETD. 		
System Administrator	<ul style="list-style-type: none"> • Log into the system • Log off from the system • Input relevant data about shipping schedules • Update the application • Create user accounts and provide access 	<p>Initiating</p> <p>Initiating</p> <p>Initiating</p> <p>Initiating</p> <p>Initiating</p>	<ul style="list-style-type: none"> • Log on to from the system • Log off from the system • Allowed to populate the system with relevant information as it pertains to shipping schedules. • Allowed to update the application. • Allowed to create user accounts and grant access to various levels of users.
CEO, Senior Management	<ul style="list-style-type: none"> • Log into the system • Log off from the system • View Ship schedule and ship details • Gets notification on when a ship starts working and when it is completed. • Gets regular updates on a ship's status(arrival, departure, current status, delays) 	<p>Initiating</p> <p>Initiating</p>	<ul style="list-style-type: none"> • Log on to from the system • Log off from the system • Allowed to view shipping information on the dashboard of the application.

Operations Manager	<ul style="list-style-type: none"> • Log delay in operation whilst off loading of vessel • Log into the system • Log off from the system • Ensure the system is working properly • Update the log and schedules of ships at the port • Mark delay as resolved • User Setup and Access • Get notification on when a vessel starts working and when it completes. • Gets notification of when a delay goes over target. 	Initiating Initiating Initiating Initiating	<ul style="list-style-type: none"> • Log on to from the system • Log off from the system • Report a delay in operations • Update the the system with current information • Keep maintenance log of the system and perform frequent maintenance of the the system
Operations Supervisor	<ul style="list-style-type: none"> • Log delay in operation whilst off loading of vessel • Log into the system • Log off from the system • Ensure the system is working properly • Update the log and schedules of ships at the port • Mark a delay as resolved • Get notifications on which ship they are supposed to work. • Get notification on when a vessel starts 	Initiating Initiating Initiating Initiating	<ul style="list-style-type: none"> • Log on to from the system • Log off from the system • Report a delay in operations • Update the system with current information • Keep maintenance log of the system and perform frequent maintenance of the system

	<p>working and when it completes.</p> <ul style="list-style-type: none"> • Gets notification of when a delay happening • Gets a notification of when the delay time goes over target. • Create, Update and Delete working schedule of crane operators, stevedore gang, Drivers, Gang foreman, Stacker Operators. • Report Delays 		
Operations Team	<ul style="list-style-type: none"> • Log delay in operation whilst off loading of vessel • Log into the system • Log off from the system • Ensure the system is working properly • Update the log and schedules of ships at the port • Mark a delay as resolved • Get notifications on which ship they are supposed to work. • Get notification on when a vessel starts working and when it completes. • Gets notification of when a delay happening • Gets a notification of when the delay time goes over target. 	<p>Initiating</p> <p>Initiating</p> <p>Initiating</p> <p>Initiating</p> <p>Initiating</p> <p>Participating</p> <p>Participating</p> <p>Participating</p> <p>Participating</p>	<ul style="list-style-type: none"> • Log on to from the system • Log off from the system • Report a delay in operations • Report delays • Update schedules • Get various reports
Gang Foreman	<ul style="list-style-type: none"> • Log delay in operation whilst off loading of vessel • Log into the system • Log off from the 	<p>Initiating</p> <p>Initiating</p>	<ul style="list-style-type: none"> • Log on to from the system • Log off from the system • Report a delay in operations • Update schedules • Allowed to view various

	<ul style="list-style-type: none"> system Get notifications on what vessel is to be worked View working schedule of gangs for a particular 	Initiating Participating Participating	reports
Gang (Stevedore)	<ul style="list-style-type: none"> Log into the system Log off from the system View working schedule. Get notification of when they are supposed to work 	Initiating Initiating Participating Participating	<ul style="list-style-type: none"> Log on to from the system Log off from the system Allowed to view working schedule
Machine Operators	<ul style="list-style-type: none"> Log on to the system Log off from the system See which ship he scheduled to work Get notifications on what vessel he is to work. 	Initiating Initiating Participating Participating	<ul style="list-style-type: none"> Log on to from the system Log off from the system See schedule assigned to working gangs
Marine Team	<ul style="list-style-type: none"> Log on to the system Log off from the system View schedules of vessels Make changes to ship schedule and ship details 	Initiating Initiating Participating Initiating	<ul style="list-style-type: none"> Log on to from the system Log off from the system See schedule assigned to working gangs Allowed to make changes to ships' schedules and details.
Pilot	<ul style="list-style-type: none"> Log on to the system Log off from the system View schedules of vessels Get notification of what ship they are scheduled to pilot and when. 	Initiating Initiating Participating Participating	<ul style="list-style-type: none"> Log on to from the system Log off from the system Track the progress of the loading and off loading of a vessel
Shipping Agents	<ul style="list-style-type: none"> Log into system Log off of the system 	Initiating Initiating	<ul style="list-style-type: none"> Log on and off from the system

	<ul style="list-style-type: none"> • See the status of the loading and off loading of vessels • Make changes to ETA and ETD 	Participating Initiating	<ul style="list-style-type: none"> • Track the progress of the loading and off loading of a vessel • Allowed to make changes to ETA and ETD
Technical Team	<ul style="list-style-type: none"> • Log into system • Log off of the system • Report delay • Get notification of individuals from various departments scheduled to work. 	Initiating Initiating Initiating Participating	<ul style="list-style-type: none"> • Log on and off from the system • Report a delay •

Use Cases

Casual Description

Name	Description	Requirements Covered
UC - 1 CreateAccount	<p>Account that will be created depending on the type of user you are.</p> <p>Different account roles will be set depending on what type of stakeholder you are.</p> <p>The administrator for your workplace will be able to create a different user for the roles.</p>	Req 4, Req 11, Req 12, Req 31
UC - 2 Login	<p>Login to see any changes that were made to the table</p> <p>Login to access the system</p>	Req20, Req 11, Req28
UC - 3 ViewDashboard	<p>Accessing the main dashboard to see any changes that were made by users</p> <p>Allowing specific users to change the planning depending on what user you are.</p>	Req1, Req2, Req3, Req5, Req25, Req 23, Req30, Req 29

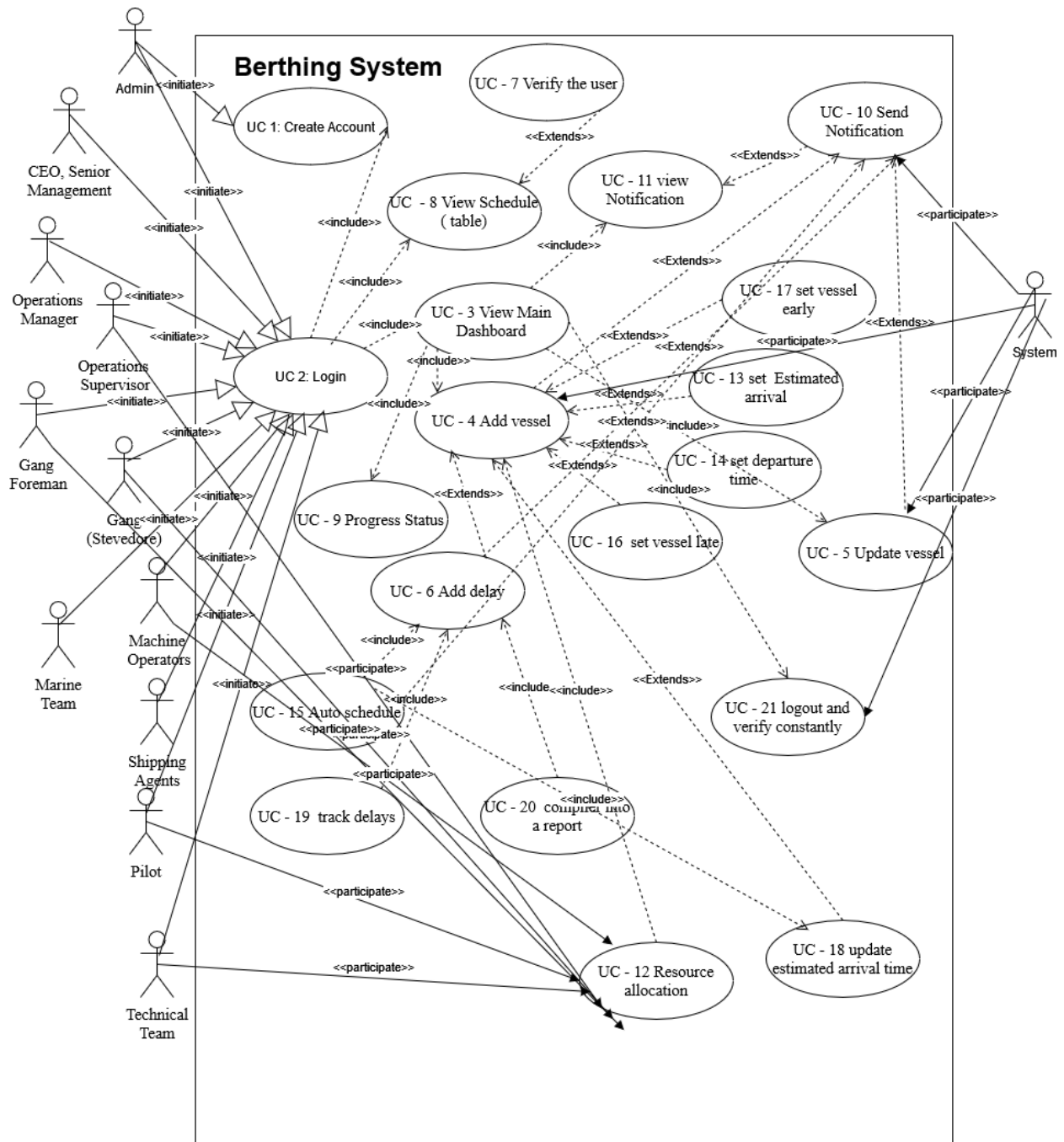
	<p>Once logged into the system a notification should pop showing what recent changes were made to the system.</p> <p>Allowing designated users to make changes dynamically through the dashboard while also notifying the other users.</p>	
UC - 4 AddVessel	<p>Adds a vessel to the dynamic table</p> <p>Add the attribute and information of the vessel</p>	Req. 3, Req. 11, Req. 14
UC - 5 UpdateVessel	<p>Allows the designated user to update all the attribute and information of the vessel</p>	Req. 3, Req. 11, Req. 14
UC - 6 AddDelay	<p>Allows the designated user to enter a delay on the ground whilst a vessel is being loaded or unloaded.</p> <p>Delay added to the system to accommodate vessels that will be affected by the delay</p> <p>Delay will be tractable to be later used to make better decisions by the designated user when adding delays or loading and unloading vessels.</p>	Req 5, Req 7, Req 8, Req 22
UC - 7 VerifyUser	<p>Function use case that will verify that the user is in the correct dashboard, main page.</p> <p>Also will be responsible to determine which table will be assigned to the different users.</p> <p>Will designate specific pages that will be only accessed by designated users.</p>	Req 11, Req 18, Req 19
UC - 8 ViewSchedule	<p>Allows users to view schedules that are relevant to them and other individuals working along with them.</p> <p>Allow user that should not be allow to</p>	Req 2,Req 1, Req 11,Req 12, Req 20, Req 24, Req30

	<p>edit the table to view and see what was change on a tabular form</p> <p>Designated role will login and will be able to see this in the main dashboard along the changes that were made</p>	
UC - 9 DisplayStatus	<p>From the moment a vessel is berthed the status as it relates to loading and unloading is tracked. Delay, if any, are also displayed as part of the progress status</p> <p>The time that is left to arrive or departure time left of the vessel.</p> <p>Will display the progress for each vessels with a friendly user pop up that will be easily learnable and understandable</p>	Req. 4, Req. 9, Req 15, Req 19, Req 21,Req26
UC - 10 SendNotification	<p>Notification sent to a designated user after a change is made by the user that updated the vessel information or any attribute of it.</p>	Req 5, Req 19
UC - 11 viewNotification	<p>A pop up specific to only see changes in a more orderly and understandable manner.</p> <p>Will be a friendly user to make them understandable to the reader.</p>	Req 5, Req 19, Req 15, Req 18, Req 22
UC - 12 ResourceAllocator	<p>The relevant personnel and equipment needed in the process of loading and unloading a berthed vessel is allocated in order for the process to be completed as efficiently as possible.</p> <p>Designated Users will be notify that a vessel needs resources allocation</p> <p>If vessel enter at a earlier state resources will be allocated immediately to avoid delays in the berth planning system</p>	Req 5, Req 6, Req 7, Req 9, Req 26
UC - 13 SetEstimatedArrival	<p>Set the estimated time of arrival.</p> <p>It will also be part of the attribute of a</p>	Req 3, Req 7, Req 9, Req 13, Req 21

	vessel.	
UC - 14 SetDepartureTime	Set an estimated departure time for a vessel.	Req 3, Req 7, Req 9, Req 13, Req 21
UC - 15 AutoSchedule	Reschedule the vessels when changes are made to the attributes of a vessel. If the vessel will arrive late it will auto schedule the vessels again. Same	Req 3, Req 5, Req 10, Req 13, Req 16, Req 17, Req 26
UC - 16 SetLate	Will allow the designated user to add a late attribute to the vessel. Calculation will be made by the system once this is changed. It will update the others estimated times to avoid delays in the system Will also inform the users that a change was made to the system to be prepared and allocate resources.	Req 3, Req 5, Req 9, Req 14, Req 16
UC - 17 setEarly	Will allow the designated user to add a late attribute to the vessel. Calculation will be made by the system once this is changed. It will update the others estimated times to avoid delays in the system Will also inform the users that a change was made to the system to be prepared and allocate resources.	Req 3, Req 5, Req 9, Req 14, Req 16
UC - 18 UpdateEstimatedArrivalTime	updates the estimated time of arrival. It will also be part of the attribute of a vessel and allow you to update it. Will also update the notification	Req 3, Req 5, Req 9, Req 16
UC - 19 trackDelays	Tracking all delays to be later used. Tracking delays would help to show where it went wrong.	Req 5, Req 7, Req 16, Req 26

	To improve for the next vessel and improve the system to reduce the delays between vessels.	
UC - 20 CompilerReport	Compiles a report to be used later to better the system.	Req 4, Req 8, Req 13, Req 27, Req 34
UC - 21 AutoVerify	Verify that every change was reported to the designated user. Notify and verify that every notification is sent, if one or more changes was not notified by the system during the updating or changing phase of the system it will check the notification table and send them once more.	Req 11, Req 12, Req 13
UC- AutoVerify	Calls the AutoVerifyUC and logs out the system after the AutoVerify UC is finished.	Req 11, Req 12, Req 13

Use Case Diagram



Traceability Matrix

P W	Requirement s	Uc -1	Uc -2	Uc -3	Uc -4	Uc -5	Uc -6	Uc -7	Uc -8	Uc -9	Uc -10	Uc -11	Uc -12	Uc -13	Uc -14	Uc -15	Uc -16	Uc -17	Uc -18	Uc -19	Uc -20	Uc -21
1	Req1			X					X													
1	Req2			X					X													
1	Req3			X	X	X								X	X	X	X	X	X			
3	Req4	X								X											X	
2	Req5			X			X				X	X	X			X	X	X	X	X		
2	Req6												X									
2	Req7						X						X	X	X					X		
3	Req8						X														X	
4	Req9									X			X	X	X		X	X	X			
3	Req10															X						
3	Reqq11	X	X		X	X		X	X													X
3	Req12	X							X													X
1	Req13													X	X	X					X	X

3	Req14				X	X											X	X				
1	Reg 15								X		X											
1	Req16															X	X	X	X	X		
2	Req17															X						
2	Reg18							X				X										
1	Req19							X		X	X	X										
1	Req20		X						X													
2	Req21									X					X	X						
3	Req22						X					X										
4	Req23			X																		
3	Req24								X													
3	Req25			X																		
1	Req26									X			X			X				X		
1	Req27																				X	
1	Req28		X																			
1	Req29			X																		
1	Req30			X					X													

1	Req31	X																				
1	Req32																					
1	Req33																					
1	Req34																X					
	Max Value	3	3	4	3	3	3	3	3	4	2	3	4	4	4	3	4	4	4	2	3	3
	Total Weight	10	5	14	7	7	10	6	12	12	3	9	11	10	10	11	11	11	8	6	8	7

Fully - Dressed Descriptions

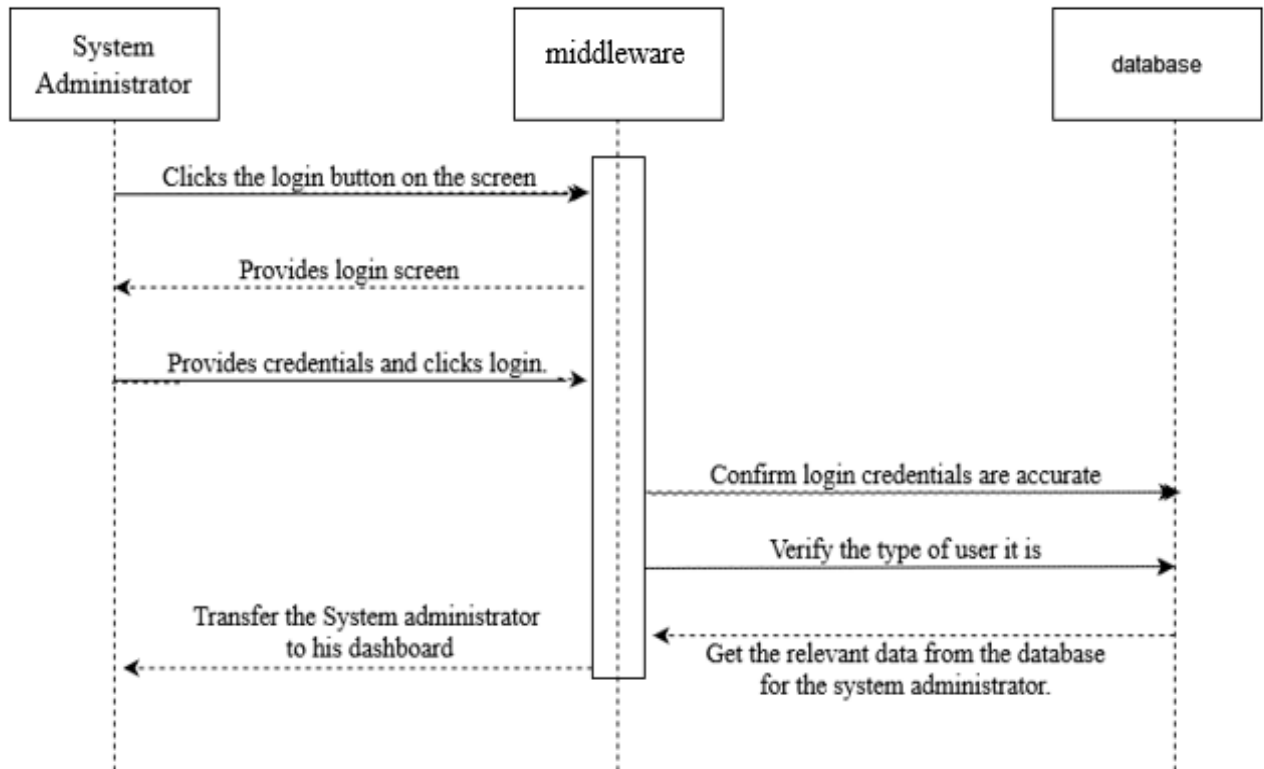
USE CASE #3: View Main Dashboard		
Initiating Actor:	System Administrator	
Actor’s Goal:	View Dashboard	
Participating Actor	System	
Preconditions	The system has been initialized and the database is updated with the latest information.	
Post conditions	Able to make changes and propagate throughout the system for all users.	
Flow of Event for Main Success Scenario		
Step	Actor	Action Description
=>	System Administrator	Clicks the login button on the screen
<=	System	Provides login screen
=>	System Administrator	Provides credentials and clicks login.
<=	System	Confirm login credentials are accurate
<=	System	Verify the type of user it is
<=	System	Get the relevant data from the database for the system administrator.
<=	System	Transfer the System administrator to his dashboard

USE CASE #8: View Schedule		
Initiating Actor	Operations Team, Technical Team, Gang Foreman, Machine Operators, Pilot	
Actor’s Goal	To view/read schedules, they can't create, update or delete.	
Participating Actor	Operations Team, Technical Team, Gang Foreman, Machine Operators, Pilot, System.	
Preconditions	There needs to be at least one vessel with a schedule in the system.	
Post conditions	None, it's just viewing.	
Flow of Event for Main Success Scenario		
Step	Actor	Action Description
=>	Operations Team, Technical Team, Gang Foreman, Machine Operators, Pilot	Taps on screen to view dashboard
<=	System	Display the dashboard with the most updated/current version of the list of vessels that will arrive for the week.
=>	Operations Team, Technical Team, Gang Foreman, Machine Operators, Pilot	Taps on “Vessel Icon” to view vessel details
<=	system	Displays the most updated/current version of the vessel details such as ETA, ETD, gang working, pilot working, Supervisor on shift, gang foreman.

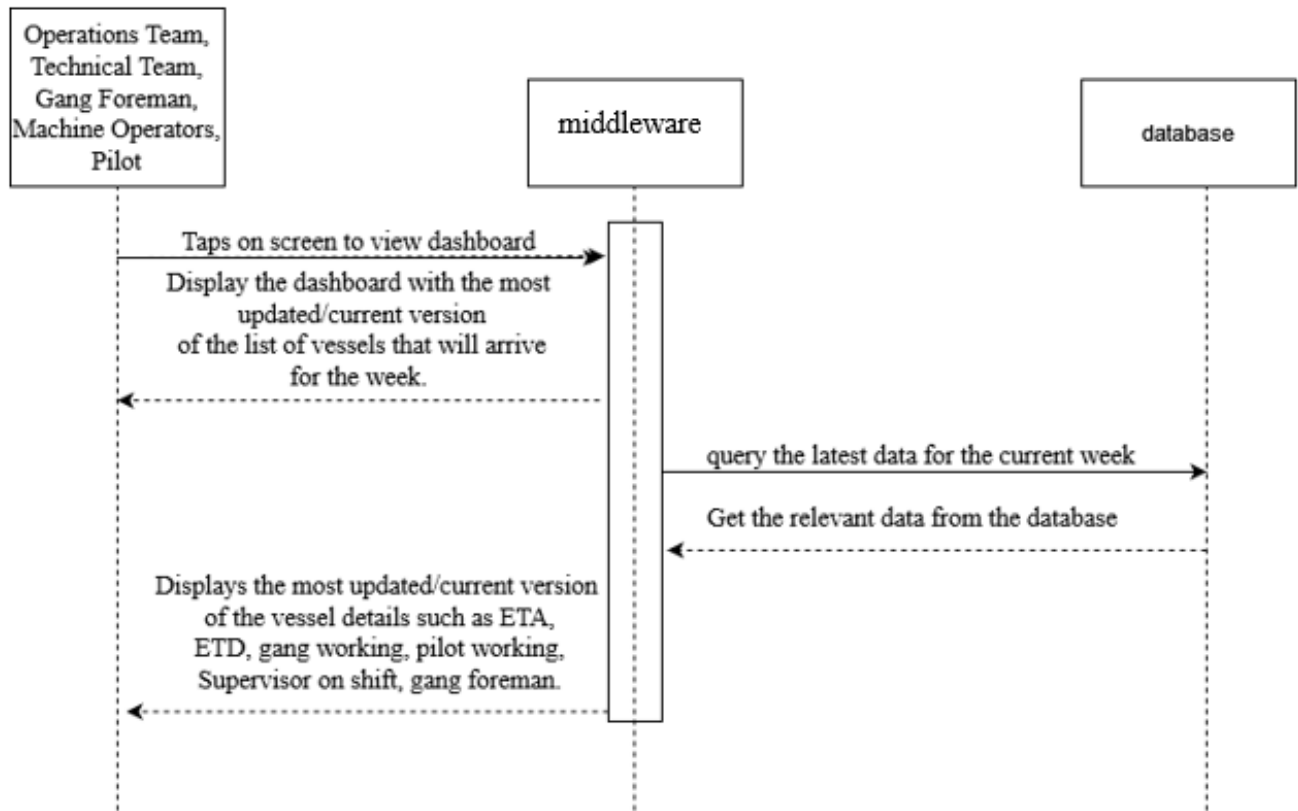
USE CASE #9: Display Progress Status		
Initiating Actor		Operations Manager
Actor’s Goal		To view the status of a berthed vessel at the sea port.
Participating Actor		Operations Manager, system
Preconditions		The system database is updated and a vessel is currently berthed and is being loaded or unloaded. Actor is logged in.
Post conditions		Status of vessel reported and relay to the relevant/interested parties.
Flow of Event for Main Success Scenario		
Step	Actor	Action Description
<=	System	Operations manager is provided with his/her related dashboard.
=>	Operations Manager	Clicks the image representing the berthed vessel
<=	System	Provides a status button of the user along with other options to the user.
=>	Operations Manager	Clicks the status button for the berthed vessel.
<=	System	Displays the latest logged information from the database as it relates to the status of the berthed vessel.

System Sequence Diagrams

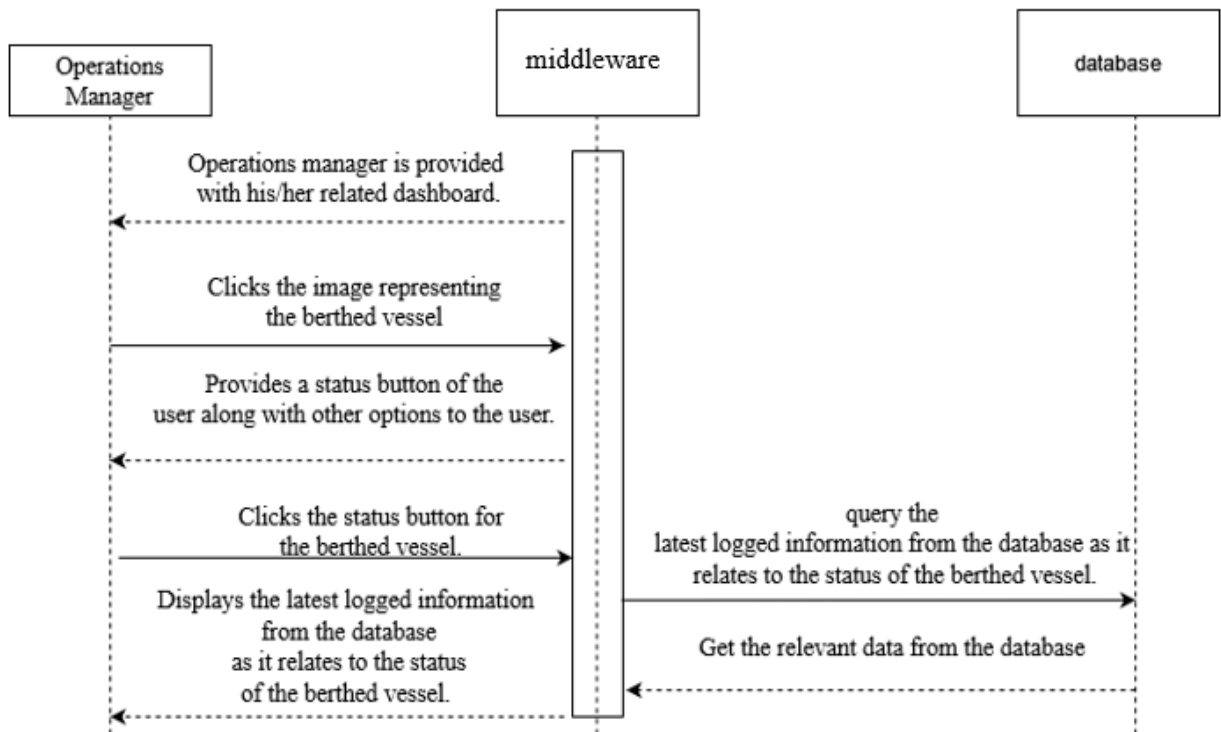
USE CASE #3: View Main Dashboard



USE CASE #8: View Schedule



USE CASE #9: Display Progress Status



Domain Analysis(Analysis and Domain Modeling)

Conceptual Model

Concept Definitions

To examine the domain model, we start by deriving domain model ideas and responsibilities from the previously described system use cases. All of the domain model concepts and their accompanying duties are listed in the table.

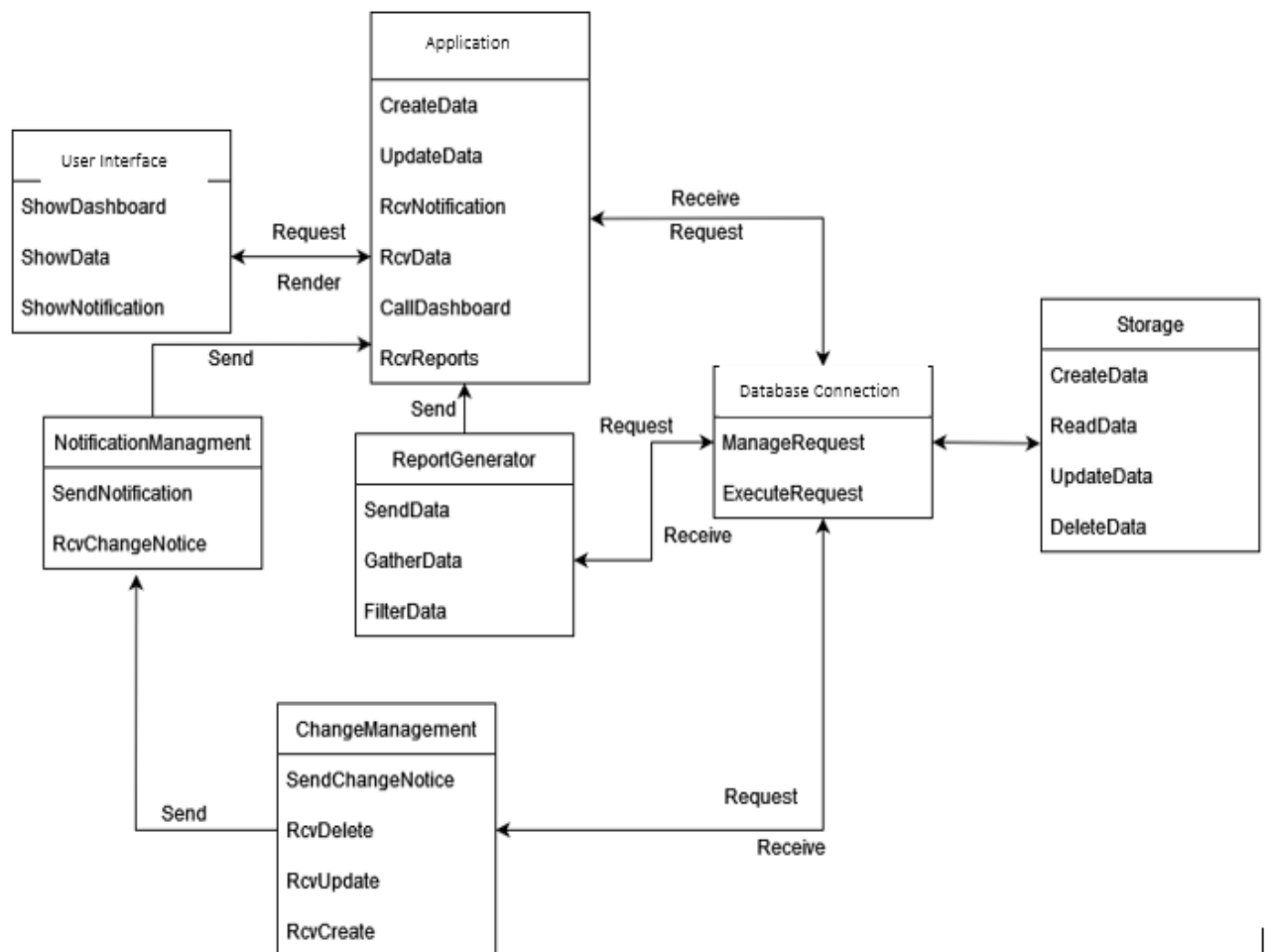


Figure 1 - Concept Definition Diagram

Responsibility	Type	Concept
R1: Read information added by user in the interface	K	Application
R2: Stores information about berthing the plan	K	Storage
R3: Manage request made to the database	D	Database Connection
R4: Execute request made to the database	D	Database Connection
R5: Check for changes made on the database	D	ChangeManagement
R6: Notify user of any changes made to the database that was notified by the change Management	D	NotificationManagemnt
R7: Gather, Filter, and Display the data as a report format	D	ReportGenerator
R8: Display the result from the system	D	User Interface
R9: Display the proper form to create, delete, update data	D	User Interface
R11: Send proper data to report generator	D	DB Connection
R12: Display Dashboard & notifications	D	User Interface
R13: Send a change notice to Notification Management	D	ChangeManagement
R14: Notification is pushed to the interface	D	System

Association definitions

To complete particular target criteria, several of the domain concepts specified above must function in specific ways. Based on the defined domain ideas, the table below shows the associated association definitions.

Concept Pair	Association Definition	Association Name
Application <-> User Interface	The system generates all data, filtered reports, and notifications, renders it to the interface to be displayed	Generate Request Display Data
Application <->Database Connection	The system sends all database request to the DB connection	Render Request
Database Connection <->Storage	DB connection gets access to the storage where it can then create, update, read or delete.	Save Data
ChangeManagement <-> Database Connection	Because the dashboard needs to display data in Realtime, the ChangeManagement checks the database for any changes made.	Check Convey Data
ReportGenerator<->Database connection	ReportGenerator requests for data to be sent by the DB connection to properly filter the data to be able to generate the report.	Request Filter Generate
NotificationManagement <-> ChangeManagement	If there are any changes in the database the notificationManagment sends a notification that there has been a change.	Convey Data
NotificationManagement <->System	Send the pertaining notification to be displayed to the interface through the system.	Display notification Send notification

ReportGenerator <-> System	Filtered data that was collected from the database should be rendered in the system so that it can be displayed on the dashboard.	Convey Data
---	---	-------------

Attribute Definitions

Responsibility	Attribute	Concept
R1: Read information added by user in the interface	ReadData	System
R14: Notification is pushed to the interface	NotificationDisplay	
R2: Stores information about berthing the plan	StoreData	Storage
R3: Manage request made to the database	ManageRequest	DB Connection
R4: Execute request made to the database	ExecuteRequest	
R11: Send proper data to report generator	GenerateReport	
R5: Check for changes made on the database	FindChange	ChangeManagement
R13: Send a change notice to Notification Management	ChangeNotice	
R6: Notify user of any changes made to the database that was notified by the change Management	NotificationCreate	NotificationManagemnt
R7: Gather, Filter, and Display the data as a report format	GenerateReport	ReportGenerator
R8: Display the result from the system	DisplayData	Interface

R9: Display the proper form to create, delete, update data	DisplayForm	
R12: Display Dashboard & notifications	DisplayDashboard	

iii. Traceability Matrix

Domain Concepts							
Use Case	Interface	System	DB Connection	Storage	Report Generator	Change Management	Notification Management
UC1	X	X	X	X			
UC2	X	X	X	X			
UC3	X	X					
UC4	X	X	X	X		X	X
UC5	X	X	X	X		X	X
UC6	X	X	X	X			
UC7	X	X	X	X			
UC8	X	X	X	X	X		
UC9	X	X	X	X	X		
UC10	X	X	X	X			X
UC11	X	X					X
UC12	X	X	X	X	X		
UC13	X	X	X	X			

UC14	X	X	X	X			
UC15	X	X	X	X			
UC16	X	X	X	X			
UC17	X	X	X	X			
UC18	X	X	X	X		X	
UC19	X	X	X	X	X		
UC20	X	X	X	X	X	X	
UC21	X	X	X	X			

System Operation Contracts

Name	compiler into a report
Responsibility	Compiles a report to be used later to better the system.
Use case	UC - 20
Exception	None.
Precondition	Changes must be made to the system for it to be reported to the user and displayed to the interface.
Postcondition	Report will be able to view in the dashboard in full screen everything else will not be shown just the report.

Name	Add Voyage
Responsibility	Adds a voyage to the dynamic table. Add the attribute and information of the voyage.
Use case	UC - 4
Exception	None.
Precondition	Vessels will arrive at our station ready to unpack for us to adjust our schedule.

Postcondition	The necessary resources will be allocated for the vessel. Adjustments will be made to the schedule. Notify the users of the changes that were made to the schedule.
----------------------	--

Name	Update Voyage
Responsibility	Allows the designated user to update all the attribute and information of the voyage
Use case	UC - 5
Exception	None.
Precondition	The voyage must be added before so we would be able to update the voyage attribute.
Postcondition	Notify the users of the updates that were made to the schedule.

Name	track delays
Responsibility	Tracking all delays to make the most educated guess for the delay to be as minimum as possible.
Use case	UC - 19
Exception	Delay wasn't added
Precondition	A change that was made by the user or the system will be tracked until the delay is no longer present or removed.
Postcondition	Depending on if the tracked delay will be delayed more or arrive earlier necessary allocation changes will be made.

Name	Resource allocation
Responsibility	The relevant personnel and equipment needed in the process of loading and unloading a berthed vessel is allocated for the process to be completed as efficiently as possible.
Use case	UC - 12
Exception	None.

Precondition	An update that was made to the schedule either update, add delete or delay was added or remove needs to happen.
Postcondition	Necessary staff will be allocated or removed from a vessel either if the schedule was updated or a vessel got delayed.

Name	Send Notification
Responsibility	Notification sent to a designated user after a change is made by the user that updated the vessel information or any attribute of it or delay was introduced or removed.
Use case	UC - 10
Exception	None.
Precondition	Changes need to be made to the schedule. An update or a vessel that was added or deleted will be saved as a change.
Postcondition	Notification will be sent if there are any changes saved in the database to the proper user.

Name	Display Progress Status
Responsibility	<p>From the moment a vessel is berthed the status as it relates to loading and unloading is tracked. Delays, if any, are also displayed as part of the progress status.</p> <p>The time that is left to arrive or departure time left of the vessel will be calculated.</p>
Use case	UC - 9
Exception	None.
Precondition	<p>Vessel needs to arrive at our station.</p> <p>Vessels need to either be loading or unloading for the status to be displayed.</p>
Postcondition	Information status will be display to interface for the pertaining user to the status of the vessel progress

Data Model and Persistent Data Storage

Data Model

In order to achieve the core requirements of the system. A database system is required to store all information. In this database information that must be stored for the efficient running of our application includes ship's information – id, name, arrival date, departure date, ship's status (on time or delayed), Notification made by workers on the ground, workers schedules and a log of what caused delays amongst other things. The MySQL Database management system and javascript will be more than capable of handling the needs of the application. Our applications, using a database management system, and javascript will make several calls to our database. Users of this application will be required to log in with their credentials provided to them. Upon a successful login user, depending on their role, will be presented with a dashboard showing, for instance, their working schedules, ship currently, status of a vessel, or members in their work crew. All the information presented to any user will be retrieved from that database. The interface provided for the various users will interact with the database and perform the various CRUD operations. Each table was meticulously related to another table to avoid any unnecessary deletion of information. Below is an image of the database schema.

Persistent Data Storage

A database is required to hold the voyage information, as well as information about users, such as user login information, delays etc... Mysql is a database that stores information in a related manner. The wide variety of fully developed features allows us to focus more on the actual organization and management of the data in relation to other modules. All that is needed is a simple call to the database to retrieve the raw data and the custom designed objects shown in the Class Diagram then do their own processing of the data. The data is required to acquire information in this application which makes using this database simple. PhpMyAdmin with the conjunction of Mysql helps us store data in the cloud which is important for us to host a website accessible to everyone. Also javascript also allows us to constantly check the storage for the notifications to work. As soon as the storage has a record a listener syntax detects it and it is displayed. The database is not directly accessible to the other objects in the Class Diagram. As demonstrated in the class diagram, only some classes will have access to the database. Certain classes will be able to create records through database class but won't directly be able to access the storage. As a result, the database class is the sole object with direct storage access.

Mathematical Model

Prediction calculations for Delays

Prediction calculations will be derived after the period of 3 months according to the trend of data. Depending on the trend the system will calculate an average time of delay. Which will then

automatically be able to know or avoid the delay. The sum of the delays will be divided by the number of delays for that month. This will be done for each month on the last day of the current month. In other words, after each end of a month it will calculate the entire average delay. After the calculation is made it will be saved on the system as a const value if a delay needs to be quickly added by the same ship it will use that value. After each month the value will be recalculated and updated depending on its average change value. The user will no longer need to add that attribute or he could still do it but if he is in a hurry the system will be able to do it for him. Reducing the time for adding a delay and making the user less necessary to add delays

$$\text{AVG_delay} = \text{Sum_of_delay} / \text{Number_of_delays}$$

$$\text{New_delay_time} = \text{AVG_delay}$$

Prediction calculations for a given location

Prediction calculations for a given location will be set depending on a ratio of average Estimated time of departure (ETD) and Estimated time of arrival (ETA). If the ETD is close to the Actual time of departure (ATD) by a margin of 30 minutes then the system will save the values of the (ETD) from the location that it is coming from to be used later for a next vessel that may come from that location. The same process will be applied to the ETA, if it is close by a margin of 30 minute to the Actual time of Arrival (ATA) then the system will save the values of the (ETA) from the location that it is coming from to be used later for a next vessel that may come from that location.

ETD AND ATD

```
IF ( ( ETD - ATD ) <= 30 && ( ETD - ATD ) >= 0 ) {  
Default_ETD_“Location” = ETD  
}  
Else IF ( ( ETD - ATD ) >= -30 && ( ETD - ATD ) < 0 ) {  
Default_ETD_“Location” = ATD  
}
```

ETA AND ATA

```
IF ( ( ETA - ATA ) <= 30 && ( ETA - ATA ) >= 0 ) {  
Default_ETA_“Location” = ETA  
}  
Else IF ( ( ETA - ATA ) >= -30 && ( ETA - ATA ) < 0 ) {  
Default_ETA_“Location” = ATA  
}
```

Changes and Notifier sender

An active listener will be programmed with the use of JavaScript to the interface whenever a change is made through the interface a sql syntax will be run to add a new entry to the changes table. This will then notify the user that a change has been made. Another active listener will program for a query to be displayed and notify the designated user. The next active listener will be constantly running in the background to check if a new entry has been to the notification

table. These listeners have to be precise as soon as a new entry is done it needs to display the notification. These notification changes have to be at least 3 seconds to 6 seconds the most for the user to get notified. This was only possible with the active listeners with the use of JavaScript. The most can take 5 seconds and the least is 3 which accomplish our desired goal.

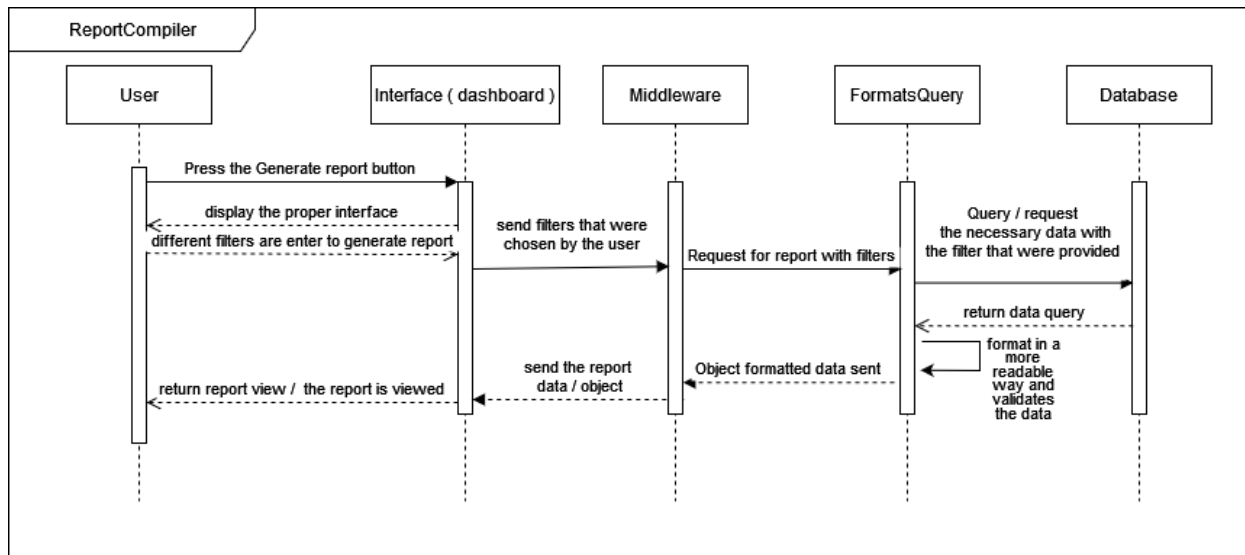
Changes

```
If ( active_listener == Active )
{
Execute Sql ( "INSERT INTO changes (deliver_to_roles, change type, change by)
VALUES
('Gang_Foreman'),
('Delay'),
('Operation_Supervisor');" )
}
```

Notifier sender

```
If ( active_listener ($display = Execute Sql ( "Select FROM Notification (notification message,
changes_made) ) >= 1 )
{
Output $Display
}
```

Interaction Diagrams



UC – Compiler into a report

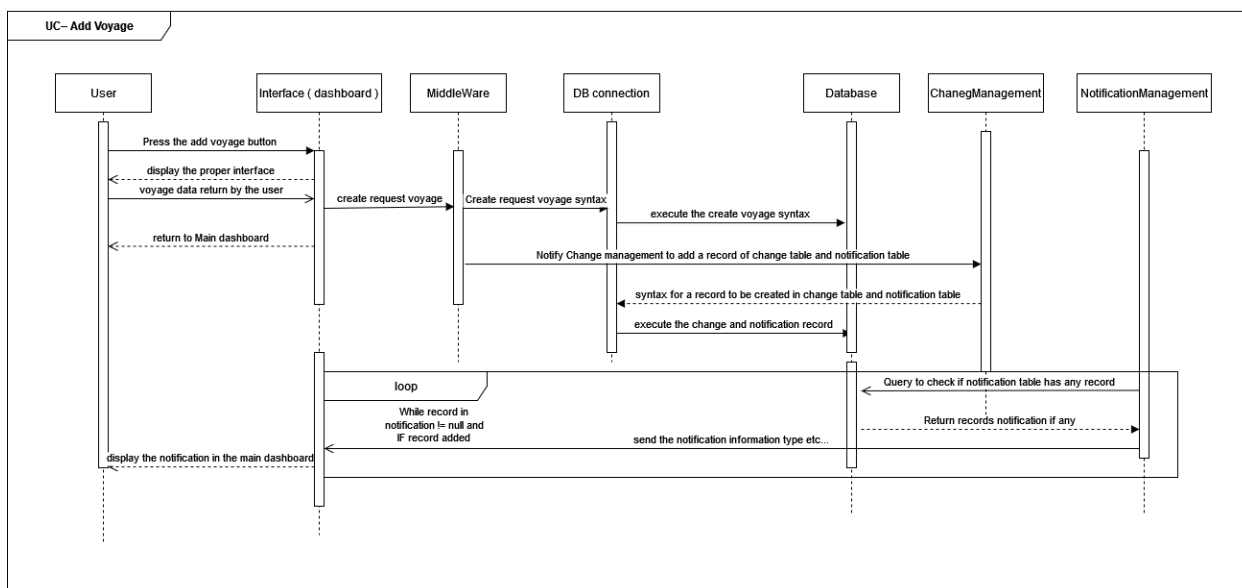
This Use Case shows the process that is happening when the user tries to generate a report in the application. Firstly, after login, the user is greeted with a dashboard page. The information and options they will see in the dashboard will be role specific. In this case the user with the permission to generate a report, will see a report button on the dashboard. When they click this button, they will be taken to a view where they must enter the different filters and criteria of the report.

After selecting the criteria, the application will move from the interface to the system(back-end), where the system then sends a request to the ReportGenerator class. The ReportGenerator then takes the request and tries to gather the data based on the criteria the user enters. The next step is to query the data based on the criteria from the storage/database through the DB connection. The DB connection is the main connection for the CRUD operation in the system. The information is

gathered from the storage and returned to the ReportGenerator where this then packages the data in a more readable format for the user.

Changes: that occurred is that the main system doesn't make the request to the database. The request is sent to the ReportGenerator and that calls the database for the information. This was necessary because the ReportGenerator should have some level of intelligence and it was necessary for formatting.

This report object is then sent back to the System class where it is then rendered to the interface for the user. When this is completed, the system then moves back to the normal state, which is the dashboard state, where it waits for changes to data.



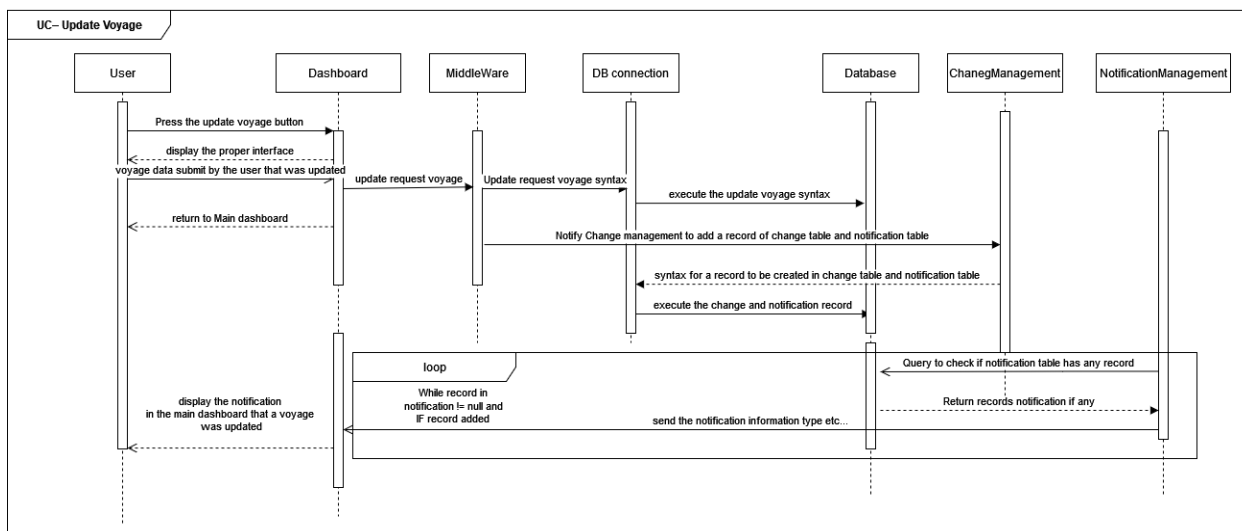
UC – Add Voyage

This Use Case shows the steps to add a voyage to the application. When the user logs into the application, the dashboard will be rendered where it will then display all current voyages stored. A plus sign will be listed above the dashboard view, where the user can click it to add a new

voyage. When this button is clicked, the user will be prompted with a form asking for required voyage details. The required information will be vessel name, voyage number, shipping agent, ETA, ETD and pilot. The next step is to save information.

When that is completed, the application sends a create request to the system with the information entered. The system then sends a create request to the Database through the DB connection and a change notice to the ChangeManagement class. The database stores the information entered and the ChangeManagement class sends a request to the NotificationManagement class to create a notification object. The object is then sent to the system where the system renders the object to the interface for the user to see and be prompted that the voyage has been added successfully.

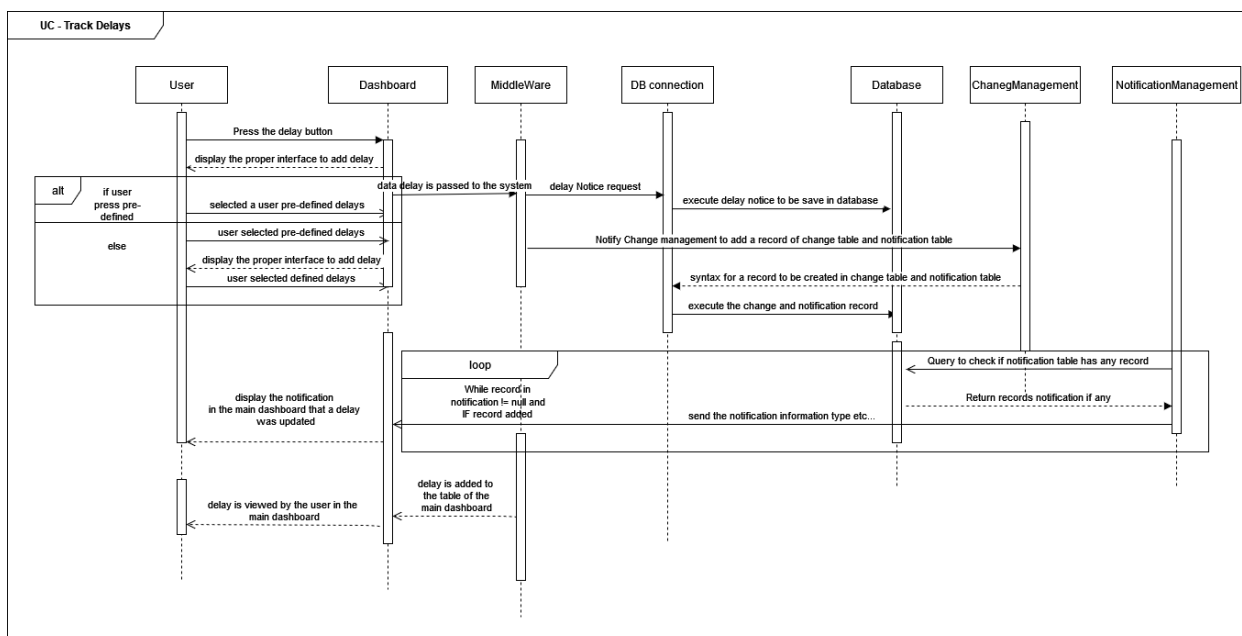
Changes: made was that the system was just doing CRUD operations without having any way to properly track updates and this led to no notifications. All changes are tracked by the ChangeManagement Class now.



UC - Update Voyage

Similar to the Add Voyage Use Case, this Use Case begins from the dashboard when the user clicks on voyage in the schedule. A view is then displayed with all information for the voyage. The user will then click the update button where they will be taken to a form where they can add more information or change existing information. The user will then click the submit button to send the information. The system then takes the information entered by the user and sends an update request with the new data to the storage through the DB connection. A change notice request is then sent to the ChangeManagement Class. The ChangeManagement also creates an object that has what was updated.

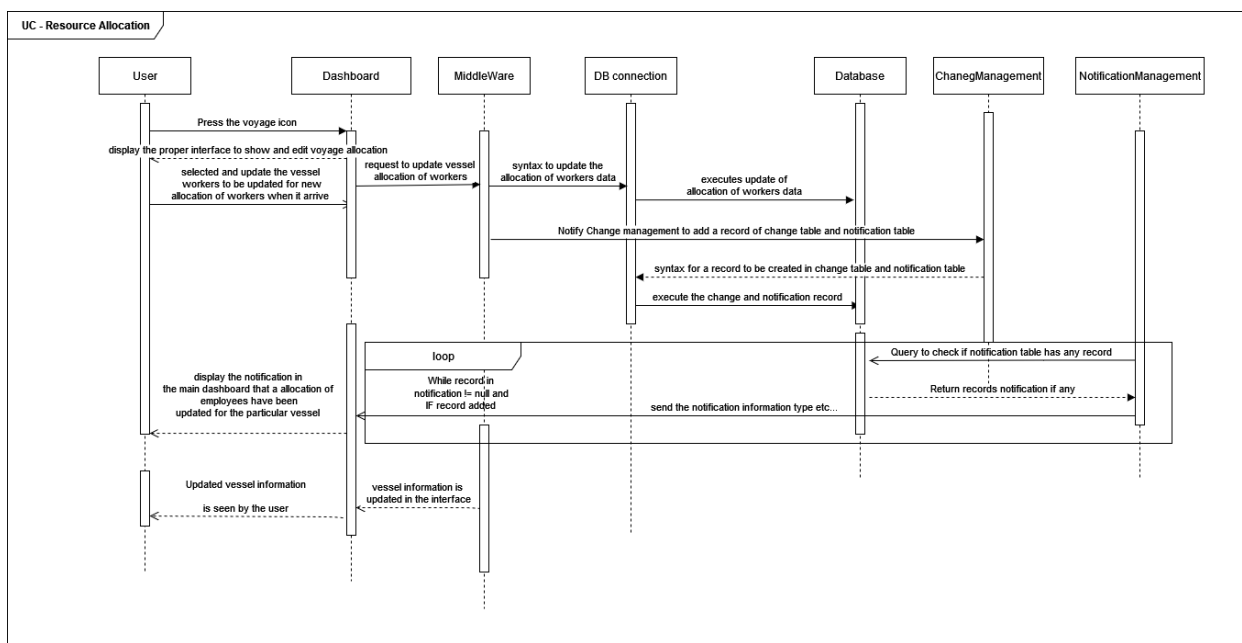
This object and a change notice request are sent to NotificationManagement class where it then sends a Notification object which includes what was updated to the system. The system renders this update to the user through the interface.



UC – Track Delays

This Use Case comes into effect when a ship has berthed and is currently being unloaded and loaded (Worked on). During the operation of a vessel, there are billable delays and non-billable delays. In this case the system is tracking non billable delays. The steps to track these delays starts by the user clicking the currently working ship icon on the dashboard. The voyage information dialog box will display showing all voyage information. When the user scrolls to the bottom of the box, they will see a button labeled delays. When they click the delay button, they will be taken to a form where they will select from a list or pre-defined delays. The selected delays will then be sent to the system through a delay Notice request. The system will then send the notice, which includes the delay type and created time, to the database through the DB connection to be stored. The delay will be listed on the dashboard for the user to see.

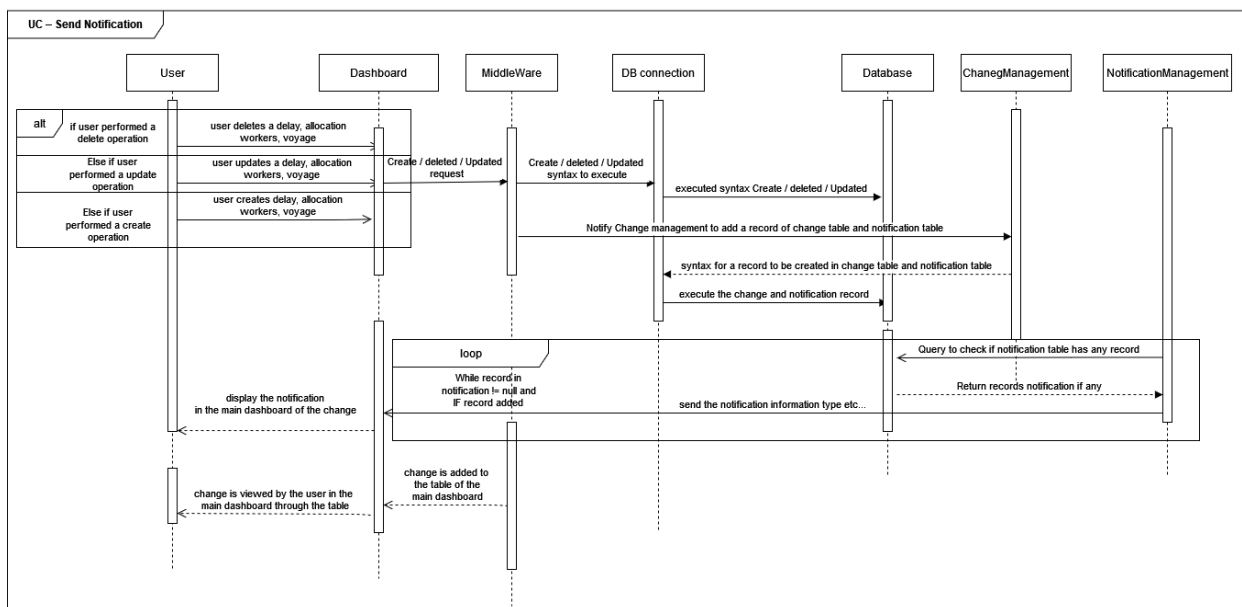
When the delay has been resolved the user will click the delay on the dashboard, and the user will see a button labeled resolved. When the user clicks the button the closing time will be sent to the system, where it follows the same steps as the create delay option.



UC – Resource Allocation

This Use Case indicates how users with the right permission can assign workers and machinery/equipment to be used for a voyage. When a user with the proper permissions log in, they will be prompted with the dashboard showing all voyages. When they click any voyage icon, they will be shown the voyage information dialog box. In this box they will see multiple worker positions such as Pilot, Crane Operator, Operations Supervisors, Truck Drivers, Stevedore Gang, Gang Foreman. They will be able to select from a drop-down list and checkbox any employee they need, to work the vessel when it arrives. In addition to selecting workers, they can select which machinery they will need as well. Machinery such as cranes, stackers, trucks and more. The selection process will be similar to selecting workers.

When all selections have been made by the user, they will submit that information to the system where the system then reads all the selected data and sends a request to the storage through the DB connection to make an update query to a specific voyage.



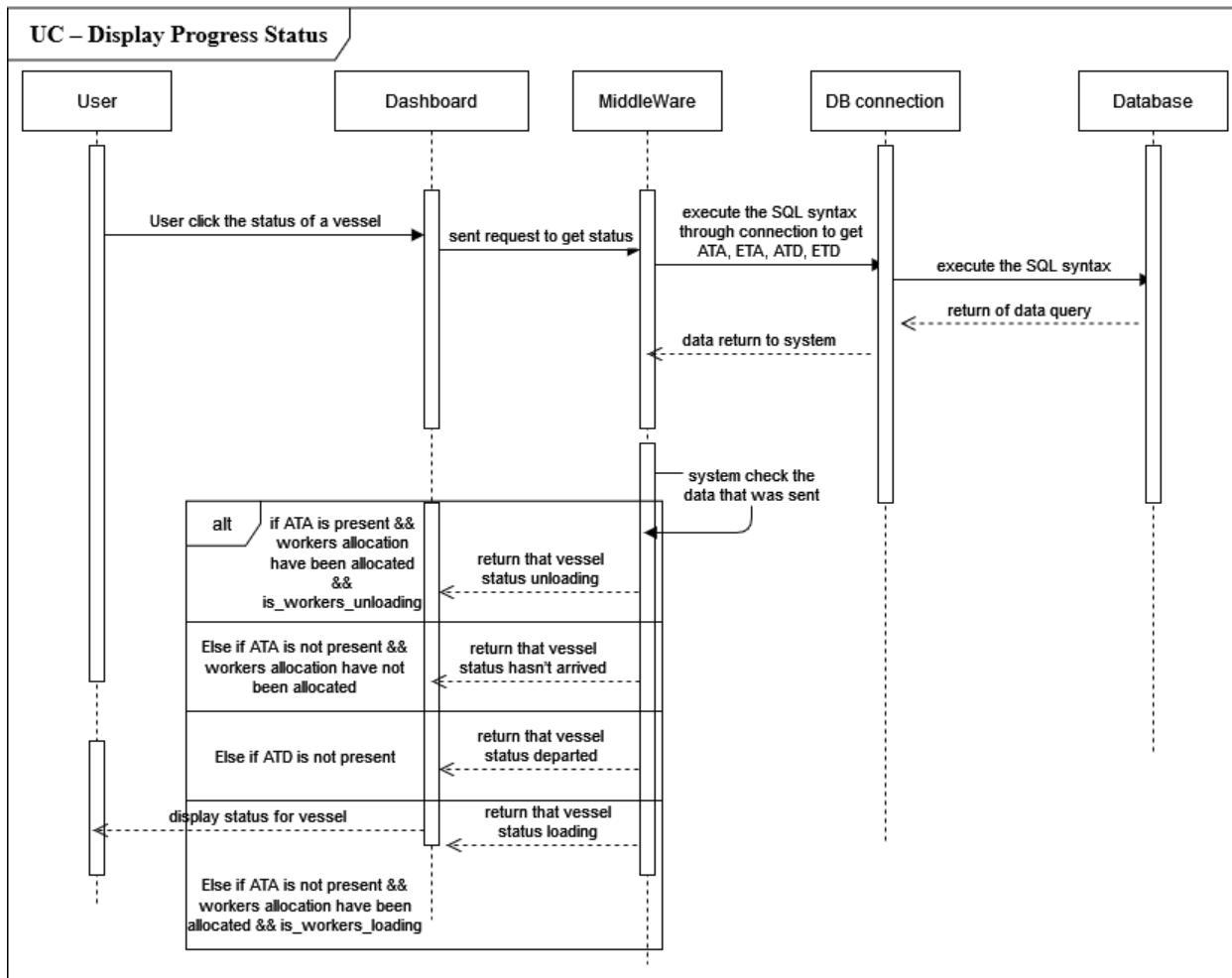
UC – Send Notification

This Use Case has multiple stages. The stages include notifications when performing CRUD operations, notifications to persons allocated to work a voyage and notification for delays.

State 1: notification for CRUD operations, includes prompting the user with a notification screen either saying successfully created, successfully updated or successfully deleted.

State 2: notification to a person allocated to work a voyage. This state occurs after the resource allocation Use Case. After a user enters the resource allocations information and it is stored in the database, the ChangeManagement Class gets a create change notice. That change notice is sent to the NotificationManagment where it creates multiple notification objects for all the users allocated for the voyage. These multiple notifications objects are sent back to the system where the system determines all the users that need to see the notification based on their allocation. After the system finds all the users, it renders individual notifications to all users allocated.

State 3: notification for delays, is when the delay that has been stored goes over target time.



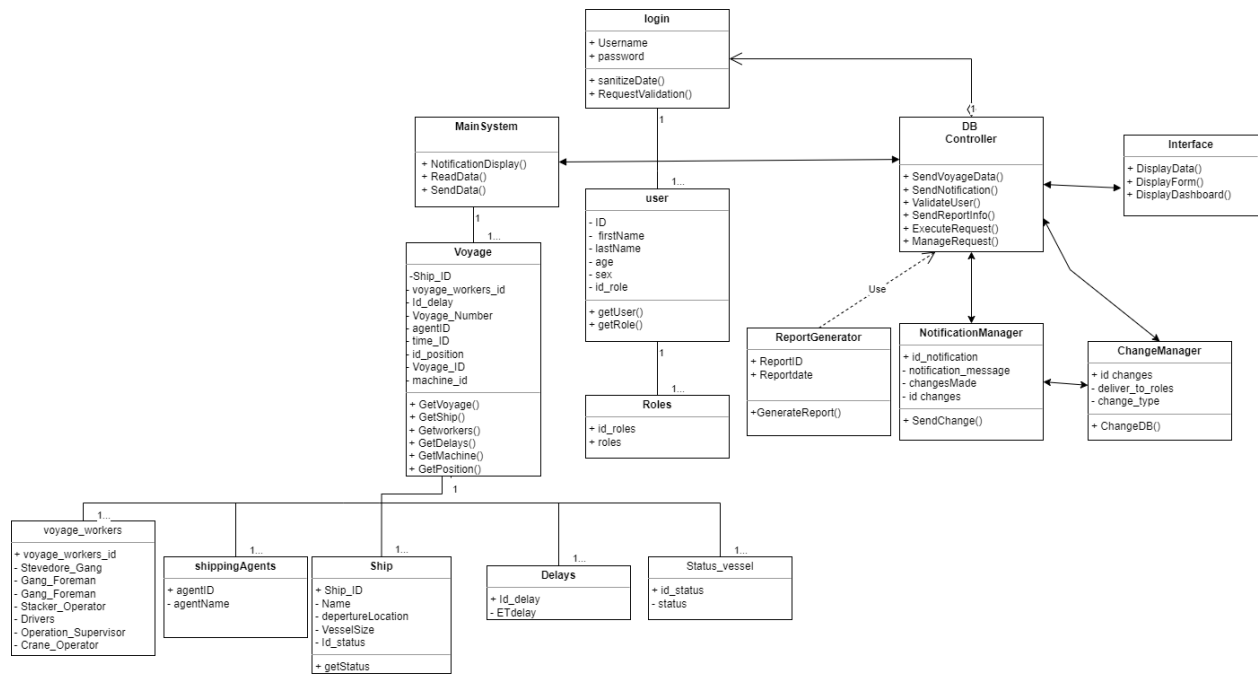
UC – Display Progress Status

This Use Case indicates how the system shows the different states a voyage can be in when it's being worked on. There are 4 states that a voyage can be in, loading, unloading, hasn't arrived and departed. The system updates the status based on the ATA, ETA, ATD, ETD. When the ETA and ETD are the only values for a voyage, that means that the ship hasn't arrived as yet and is in transit, when the ATA is added, the ship is in a state of loading and unloading. When the ATD is added the ship is in a state of departed.

The system is constantly in a state of waiting for notifications about the ATA, ETA, ATD and ETD to be added. When the system receives these notification objects, it then updates the state of the voyage.

Class Diagrams and Interface Specification

Class Diagram



Domain Types and Operation Signatures

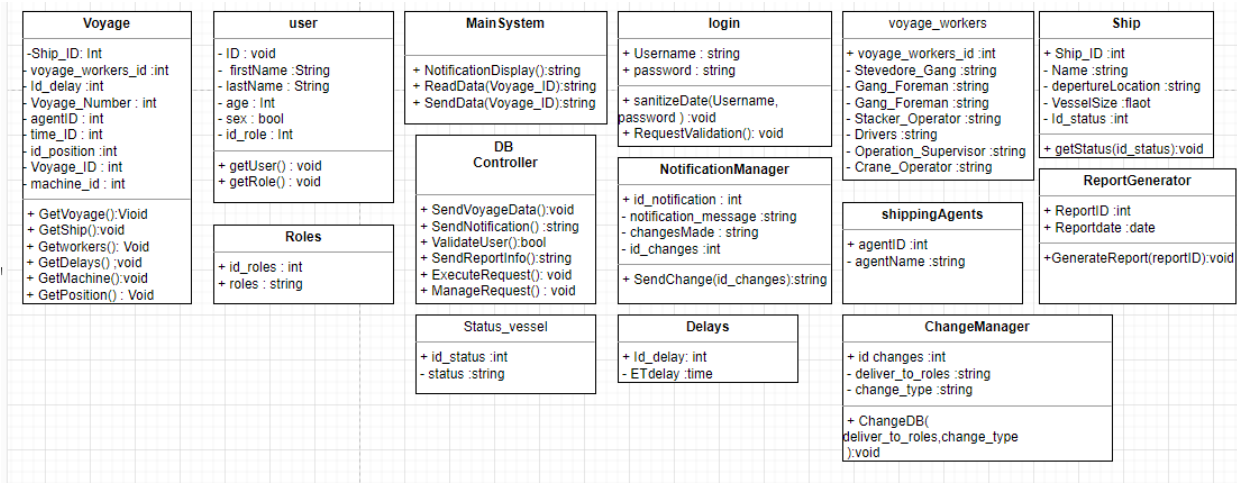


Figure 4-data types and operations signature

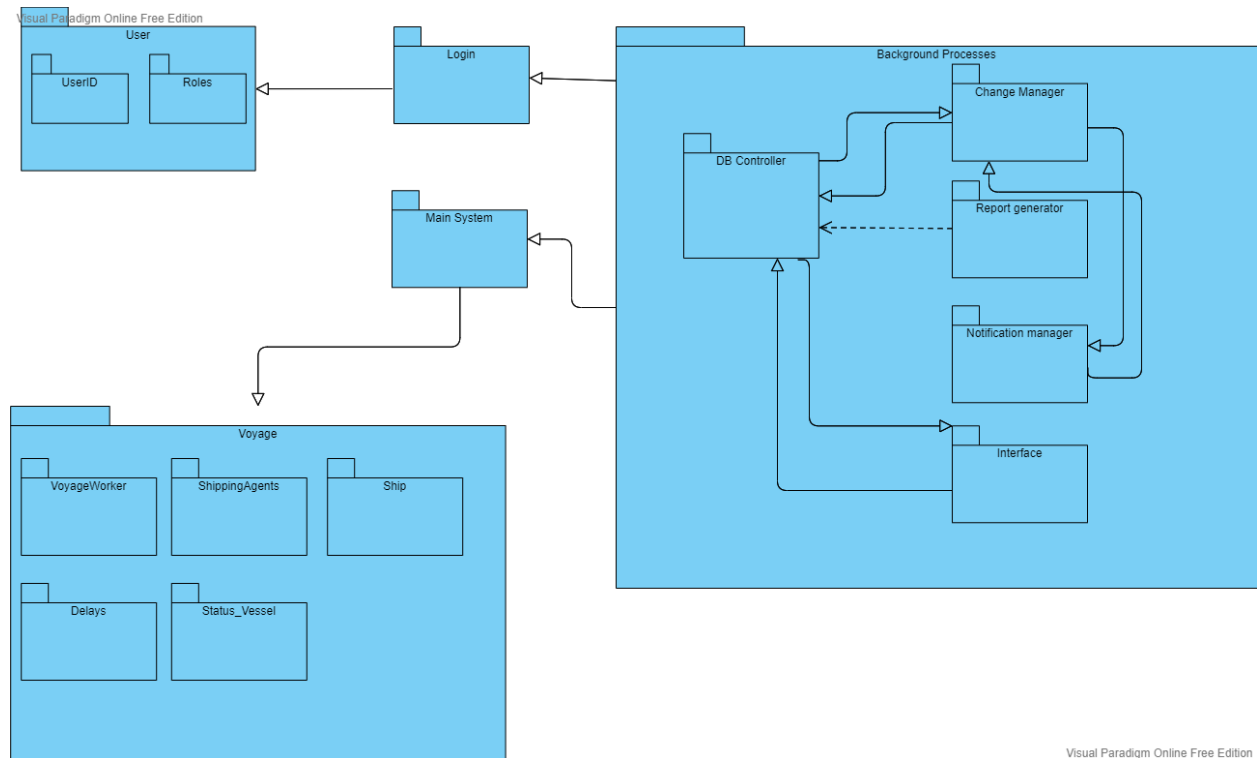
Traceability Matrix

	Classes						
Domain Concepts	Login	User	DB Controller	Report Generator	Main System	Voyage	Roles
Interface	x	x					
System			x	x	x		
DB Connection		x	x			x	x
Change management			x				
Notification Management			x				
Report Generator			x		x		
Storage			x				

	Classes						
Domain Concepts	NotificationManager	Voyage_Worker	Delay	Shipping Agent	Ship	Status Vessel	ChangeManager
Interface	x	x					
System			x	x	x		
DB Connection		x	x	x			x
Change management						x	x
Notification Management		x					
Report Generator				x			
Storage		x					

System architecture and System Design

Identifying Subsystems



Architecture Styles

The architecture styles that will be used are Event-driven architecture, Database centric and client server architecture. The use of event driven is because of the way the Berthing system works that depending on an event happening it sends a notification to the users depending on what change happened to the ships arriving and departing. This will also be database centric because all data will be stored in a database so that it can be displayed in real time to the users and also sending notifications to them if an event is to change. All of this will be done with the help of the server client architecture so that all users will be connected too.

Mapping Subsystems to Hardware

The system will be able to function on multiple functions. Tablets, phones and computers will be able to run the application. Since the system requires an online connection, all devices must have a stable internet connection before they are able to view the updated schedules. On the user side, they will be able to drag and drop the forms within the schedule.

Connectors and Network Protocols

This system will be able to communicate between many devices and for this reason we will use HTTP so all devices can communicate to each other using a database, PHP and mysql. In order for this to work the page should refresh every 5 seconds. Javascript script will be used to refresh the parts of data on the web application.

The notifications on the mobile devices will use the **web push protocol**. This protocol works by using your server to send a push protocol to the push service then the service sends the desired notification to the mobile devices.

Global Control Flow

Execution orderliness: This will be an event driven system because every time the berthing database is updated by any of the users that have the privilege to do so. It will trigger an event depending on the user and the user case they initiated.

Time dependency: This system will be using both event response and real time system. For the event response it will send notifications to the users when an event is about to happen or an event is changed. This aims to facilitate the users to be aware of new updates like if a ship arrives early or if there is an puch back on the departure of a cargo. The system will be updating in real time and refreshing the data every 5 seconds so that users using the web application will always have the most updated data from the database.

Hardware Requirements

Client Requirements:

- Laptop/Desktop/Tablet/Mobile Phone - 1.8 GHz processor, RAM: 2GB

Mobile:

- Web browser, iOS or Android operating system, stable Internet connection

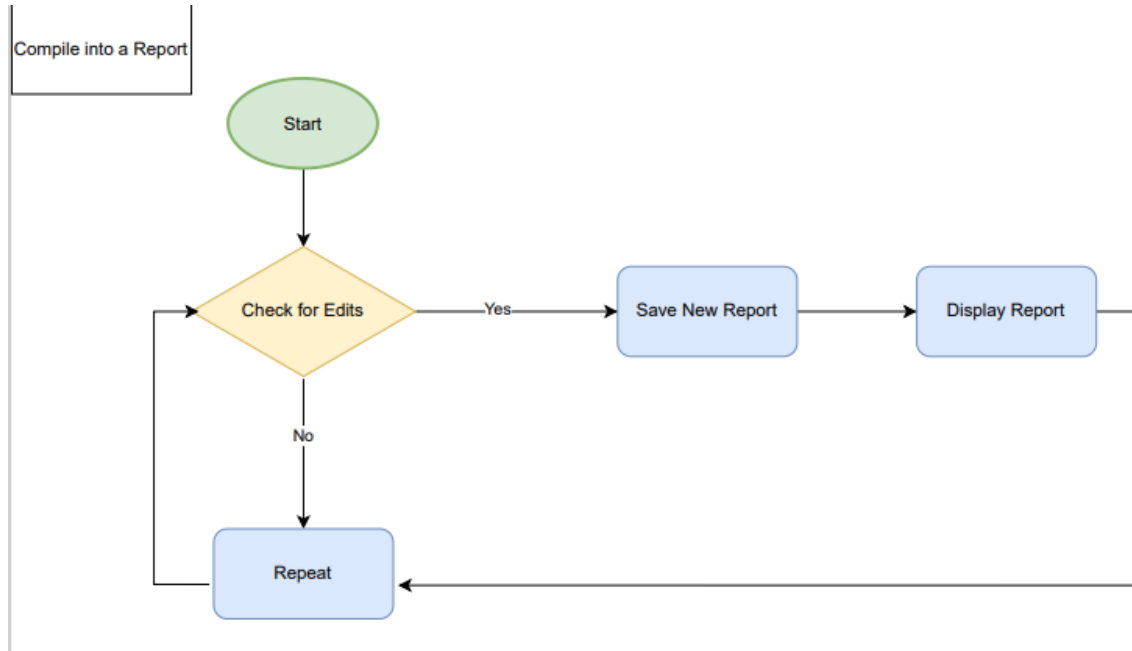
Server:

- CPU: Intel Xeon 2.4GHz, 6GB ECC ram memory, 2 TB HD storage, 30Mbps Internet connection.

Algorithms and Data Structures

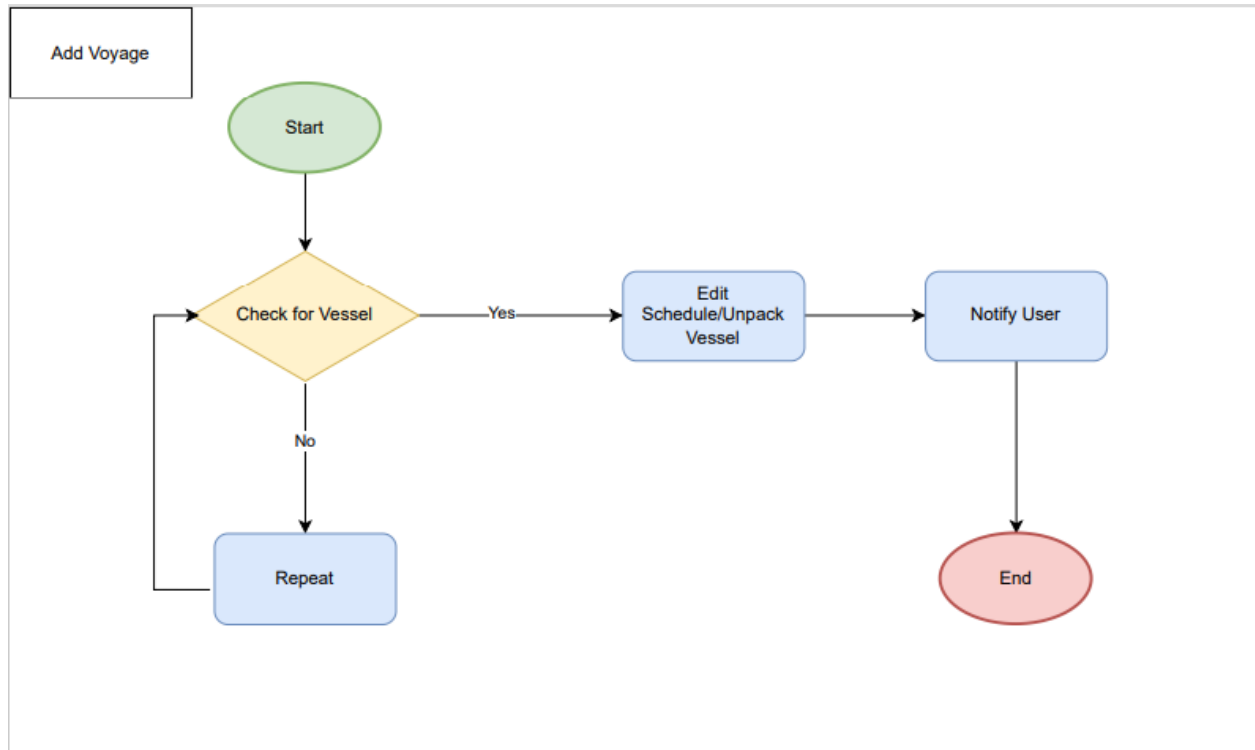
Algorithms

Compile into a Report



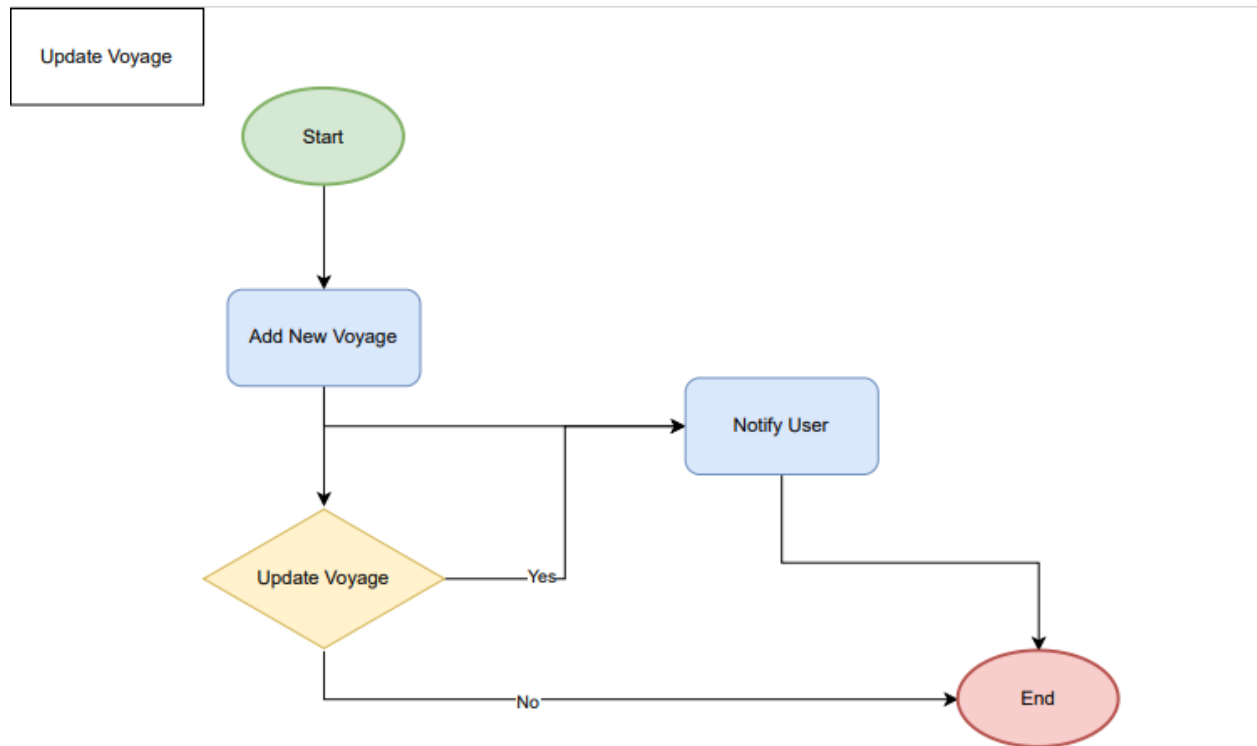
At the start of the algorithm, we check if there are any edits made in the system, if there are, we then save the report and display it. If not, we repeat the check until an edit is made. This allows the system to always be up to date with the latest information.

Add Voyage



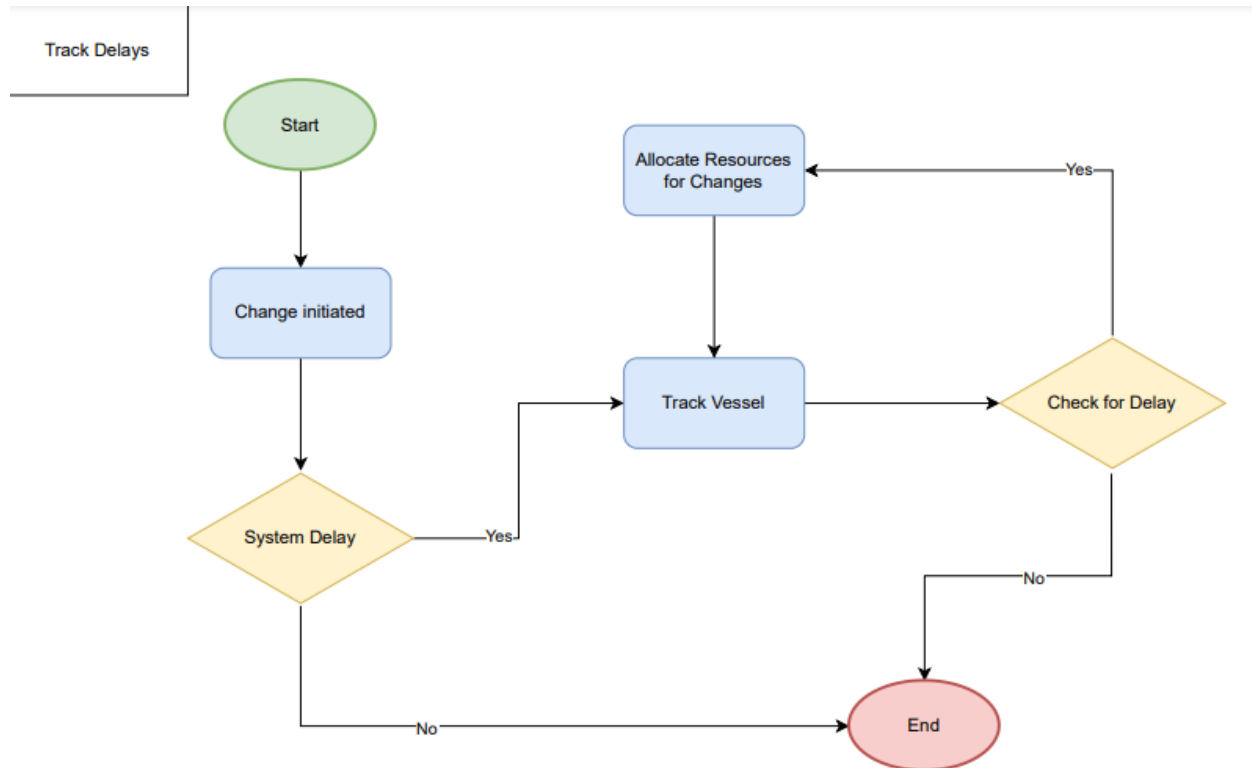
At the start of the algorithm, we check if there is a Vessel coming into the port. If yes, we adjust our schedule to accommodate the vessel and unpack it then we notify Users. If not then we continue to check for vessels coming in.

Update Voyage



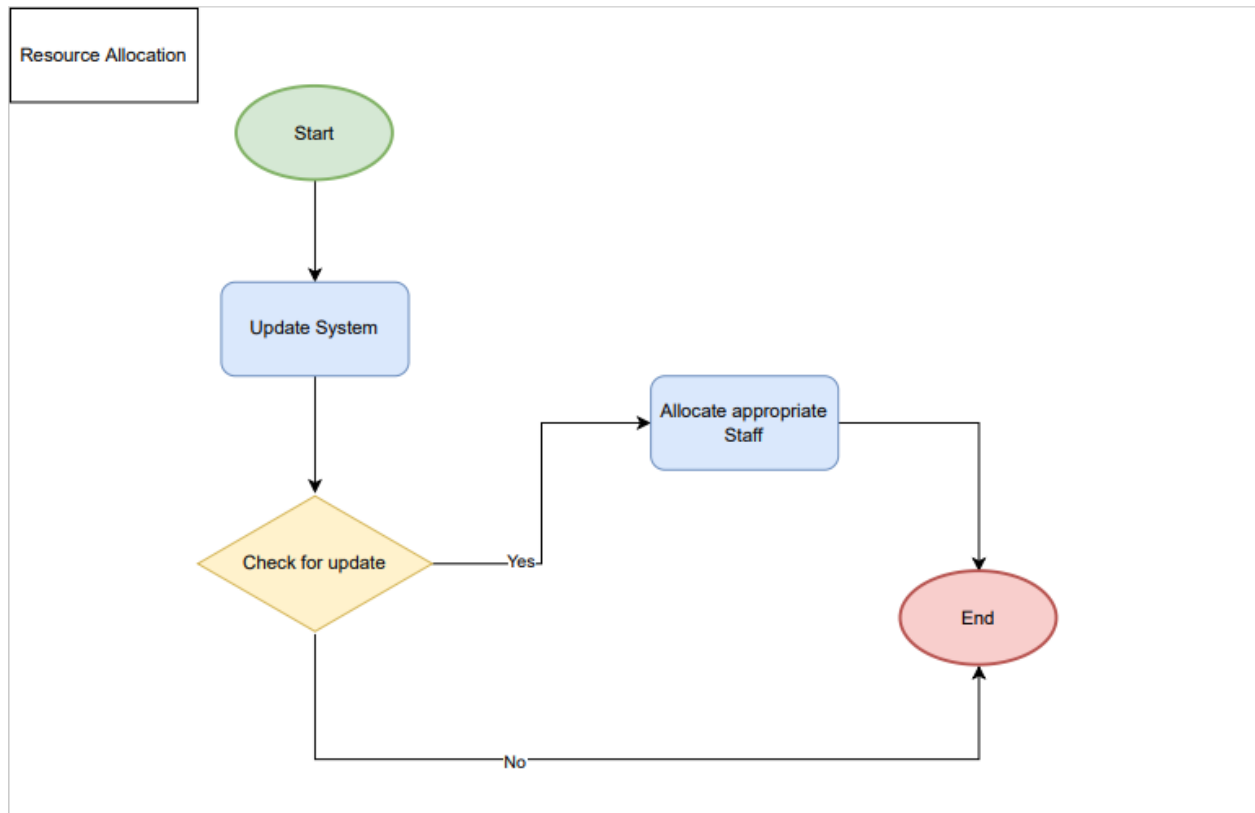
Algorithm starts and adds a new voyage to the system, it then notifies users. Then it checks if the voyage has been edited after being added. If yes then it updates the user again.

Track Delay



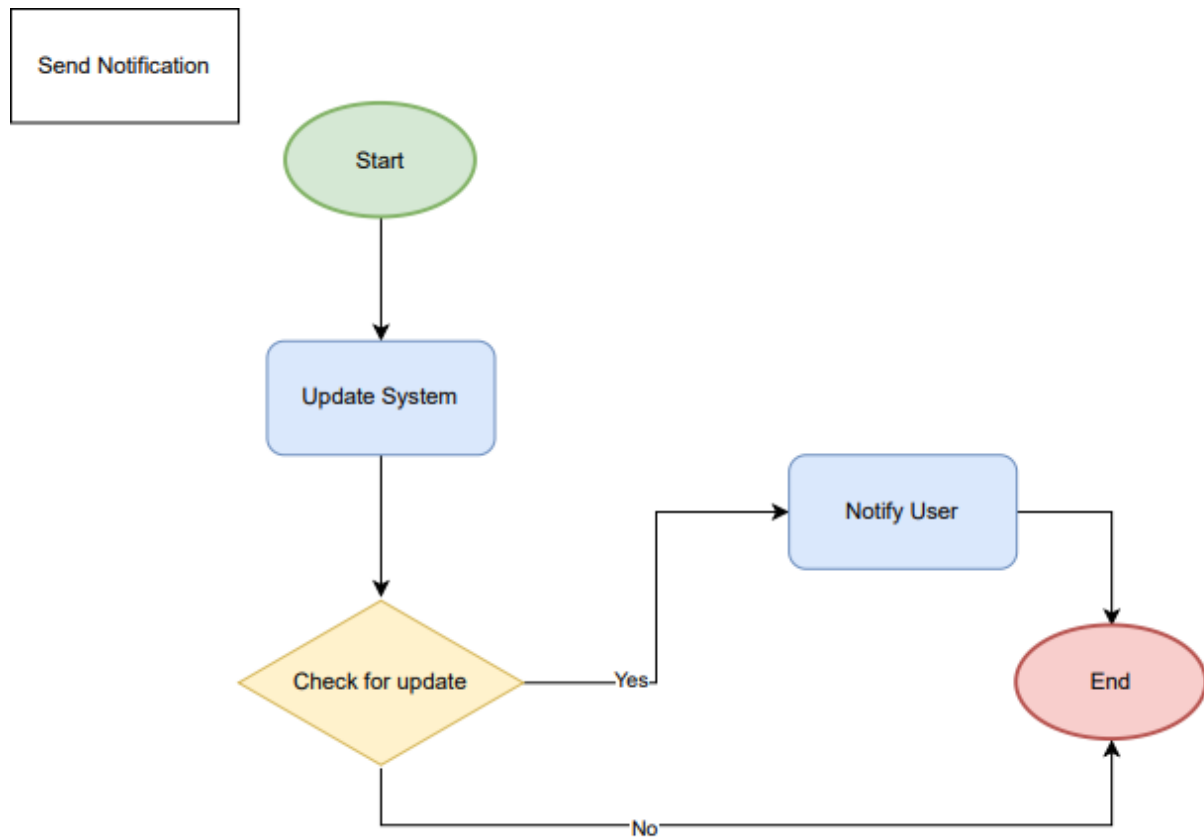
Algorithms start by the user initiating a change in the system. We check if there is a delay with the berthing process? If yes, we track the vessel and check if there is still a delay. If there is then we allocate resources to speed up delay. We then track the vessel and ask again if there is still a delay. If there is no delay, we then end.

Resource Allocation



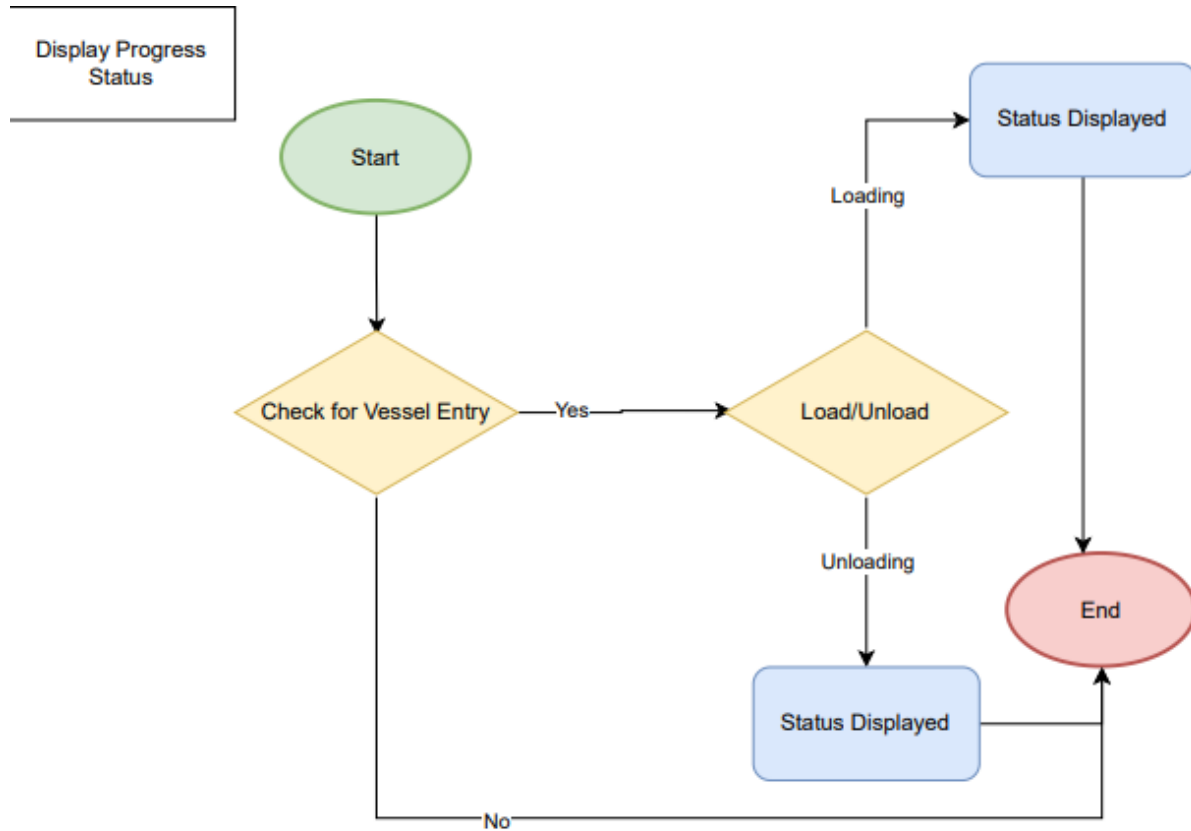
Start algorithm, update system, check if there is an update and notify appropriate staff to be allocated to assist. If not we end.

Send Notification



System is updated, then it checks for what type of update it is, then it notifies users. if no update it ends.

Display progress Status



Algorithm starts and checks for vessel entry; if a boat begins to load or unload either one that occurs, it will display the status of the shipment on screen to view.

Data Structures

The data structure that is used in the berthing schedule project is the array. An array holds elements that can be retrieved based on their index. Within the context of the project, the array stores the data on the “operator information” and “voyage detail.” The array data structure was selected because:

- Array elements can be accessed directly
- Collects multiple values with only one variable
- The size of the array is decided at run time
- The array data is easily manipulated
- Arrays have push, pop, merge and sort functionality in php
- The array can hold both null and duplicate values

Design Patterns

Use Case #3: View Main Dashboard

In this use case we used the Stair Design Pattern. While using this method we are able to make the dynamic behavior be distributed. Each object basically delegates responsibility to other objects. So when viewing the main dashboard, the system knows which object can help with particular requests. In this case the system knows where to validate the login credential and what to display to the user based on the role that they have assigned. The user enters login credentials, systems confirms the credentials with what's stored in the database. If it matches the role, then the dashboard is displayed to the user. With this in mind the system displays High coherence because it has many associations and low coupling because the calling class does not know anything about the internals.

Use Case #8: View schedule

Stair Design Pattern was also used with this use case and it will be a recurring design pattern for all of our use cases. While using this method we were once again able to achieve high coherence and low coupling. When a user is trying to view a schedule, they tap the schedule button. The system makes a call to the database to view most records for the current week. The class that queries the information from the database will be similar to the class for validating the user credentials and role from UC3. Due to the system being designed like this we are able to measure dependencies among subsystems and not worry about other subsystems being changed because one subsystem was changed.

Use Case #9: Display Progress Status

To view the progress of the voyage, the user clicks a vessel that's being worked on in the dashboard screen of the application. The system then makes a call to the database where it does a query to view the latest logged information of the vessel that's at berth. Once again the stair pattern design works perfectly for this use case. It gives us room for our subsystem calls to not be dependent. Which reduces system complexity while allowing change.

In a nutshell, the entire application will follow the stairs design because our subsystems delegate responsibility to the database and our subsystem only knows the database and the main system. The main system only knows the subsystem and the interface where the user sees the forms and data. The application then achieves low coupling and high coherence because our system is separated from the database. The database will handle all the data management and storage and our subsystems and classes will be responsible for querying the data from the database and displaying it to the users.

OCL Contract Specification

Contract Name Report Generator	
Cross Reference	UC - 20 - reportGenerator
Invariants	The report data
Precondition	User must choose the filters they want
Post condition	It compiles a report based on the filters that the user chose.

Contract Name Add Vessel	
Cross Reference	UC - 4 - addVessel
Invariants	User and available vessels
Precondition	-GetShip is initiated and an available ship is chosen -GetWorkers is initiated and the workers are chosen -GetDelays is initiated and gets the delays times.
Post condition	System should add a voyage to the dynamic table and add the attribute and information of the voyage.

Contract Name Notification Sender	
Cross Reference	UC - 10 - notificationsSender
Invariants	System and SendChange method
Precondition	-DBcontroller listening to any changes in the database
Post condition	-Sends a notification to all users .

Contract Name Verify User	
Cross Reference	UC-7 VerifyUser
Invariants	User
Precondition	-GetUser whereby the system identifies the user -GetRole whereby the system identifies the user's role
Post condition	Allows the user to view certain information on the dashboard depending on role.

Contract Name Track Delays	
Cross Reference	UC - 19 - trackDelays
Invariants	Delay
Precondition	DelayTime - Gets how long the delay should take IDdelay - Gets the type of delay
Post condition	Keeps track of the delay and how long it is taking

User Interface Design and Implementation

The first draft of the UI system has changed significantly since the first report. The UI was made to look more modern and more user friendly. Ease of use has increased due to these conscious changes to the UI. After several iterations of mock testing with a paper prototype, we were able to ensure that all information, data, views/screens, forms and reports were further developed to ensure that the application is easy to learn, which leads to users having no issues when using it. Additionally, through testing the paper prototype we were able to ensure that the program was easily recoverable from any errors. The improvement will be listed below:

SignUp/Login/Forgot Password Menu

Signup form was created using a minimal but modern design. All inputs are labeled so that there isn't any wrong input and can help a user successfully signup. In this form to ensure the user is actually putting in the right password we are setting up the form to enter the initial password and to confirm the password. This is best practice when doing the module.

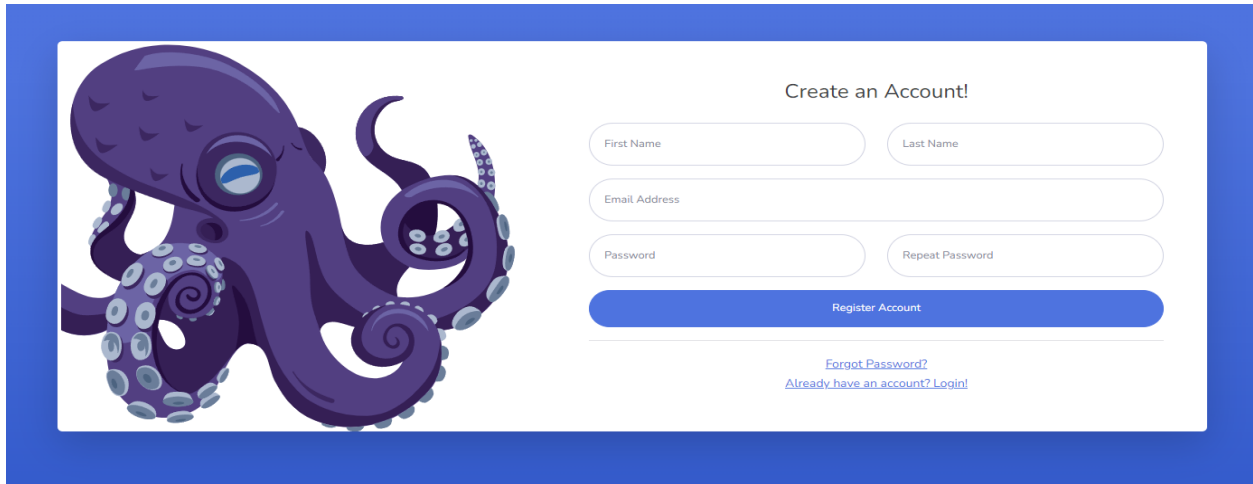
The image shows a 'Create an Account' form on a blue background. On the left is a large, stylized purple octopus illustration. The form itself is white and contains the following elements: a title 'Create an Account!', two input fields for 'First Name' and 'Last Name', a single input field for 'Email Address', two input fields for 'Password' and 'Repeat Password', a blue 'Register Account' button, and two links at the bottom: 'Forgot Password?' and 'Already have an account? Login!'.

Figure 1-1: Create an Account

The Login form basically is just asking for the email and password. It also has links to the other forms. There is a link to reset password if you forgot it and if you don't have an account, users can create one themselves. The forms basically follow the same theme in terms of trying to give a minimalistic design but a design that's effective and easy to use.

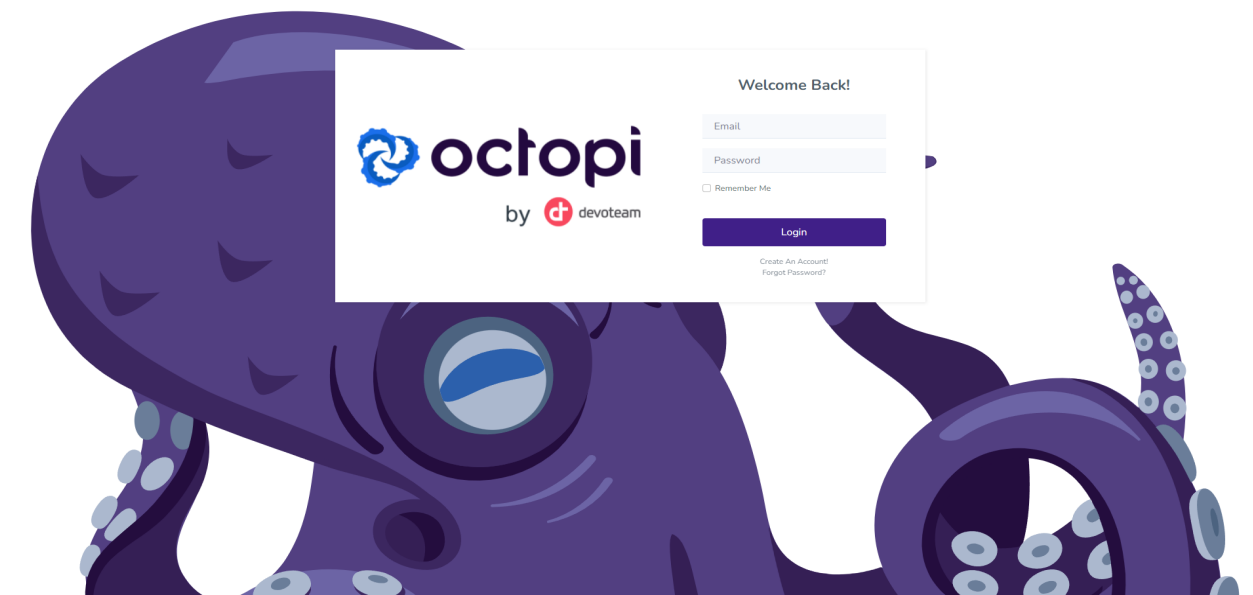
The image shows a login screen with a large purple octopus illustration in the background. A white login form is centered over the octopus. The form includes the 'octopi' logo (a blue octopus head icon) and the text 'by devoteam' (with a red 'd' logo). The form has a title 'Welcome Back!', input fields for 'Email' and 'Password', a 'Remember Me' checkbox, a blue 'Login' button, and two links at the bottom: 'Create An Account!' and 'Forgot Password?'.

Figure 1-2: Login Screen

Reset Password form is basically asking the user to enter their email address that the account is registered and it will send a request to email to reset password.

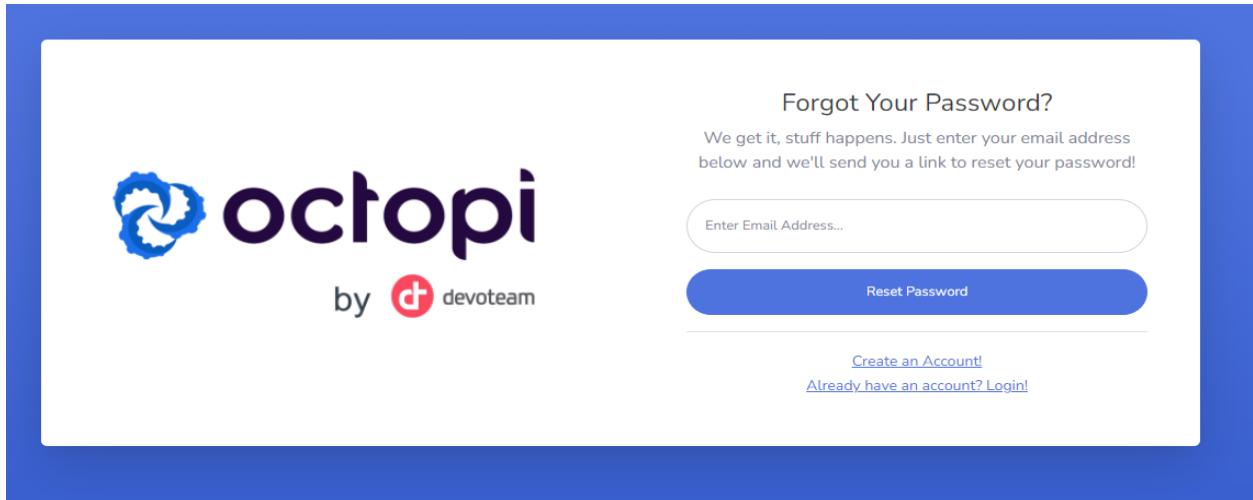


Figure 1-3: Forget Password

Dashboard

The next main section of the UI is the Dashboard. During the mock of the UI, we wanted a main landing page that the user can reach after login, which would have the top 5 upcoming voyages and it would also show the status of the ships that are currently being worked on. The status would be displayed in a card-like widget where it has the progress of the voyage in a percentage unit. From the same card they would be able to see who is working on the ship and the ship information such as ship name, voyage number, agent, and schedule. In addition to the cards, the dashboard shows its Nav Bar at the left and this Nav bar has the links to the other services the application has to offer. The reason for these specific features is to ensure the most data is shown in a clear and concise method. This helps with the learnability and efficiency of the application.

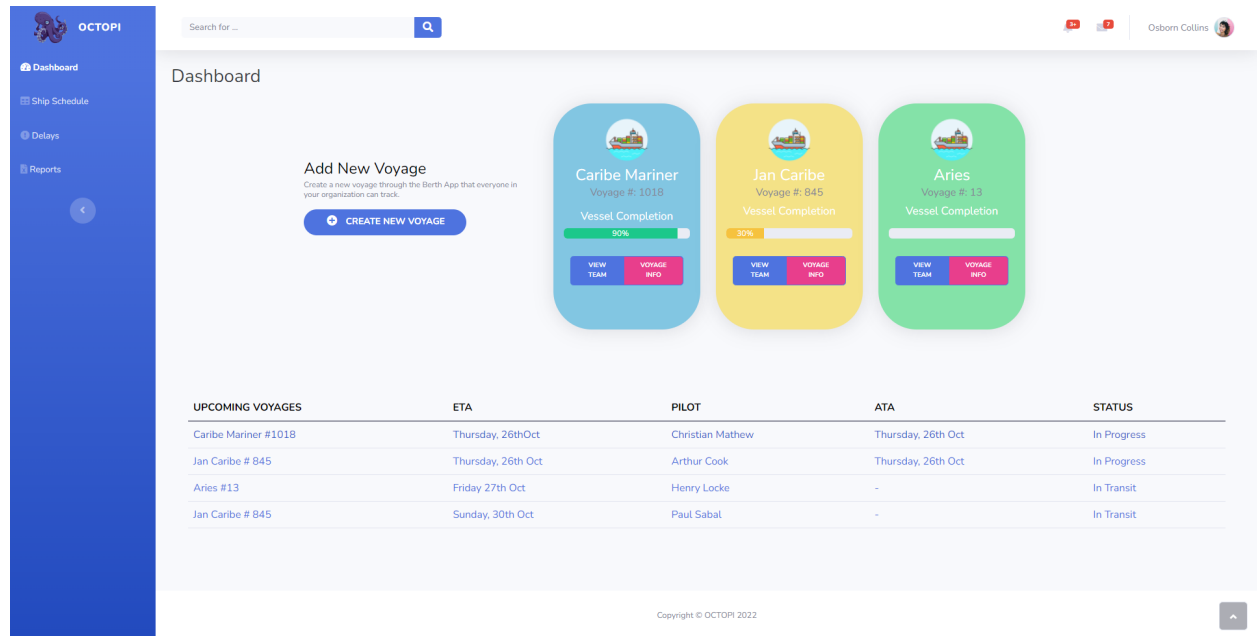


Figure 2-1: Dashboard

Add Voyage/View Schedule

When completing the add voyage form, the idea behind the design was to make things simple as possible, with minimum inputs where the user had to manually type information. So any input that could have been a drop down menu, was made a drop down menu or checkbox to select multiple values without typing it. Using this method, it tries to mitigate human error. During this phase safety was part of the main driving forces. This application will work its best with optimal and accurate data. Therefore setting up such safety measures works well to meet that requirement. Apart from that the form follows the minimalist modern design.

Figure 3-1: Add Voyage

Figure 3-2: Add Voyage

Add Delays/View Delays

When adding the delays per voyage, similar to the add voyage, this form consists mostly of a drop down list of pre populated delay and date and time picker widgets.

OCTOPI

Search for ...

Osborn Collins

Add Voyage Delays

Select Delay Type: AGENTS DELAY

Select Active Voyage: This is item 1

Select Delay Start Time: --:-- --

Submit

Figure 4-1: Add Delays

After adding delays we will be shown the list of all active delays where we can then close off a delay or add a new delay. Delays will be shown by voyage.

OCTOPI

Search for ...

Osborn Collins

Active Delays

ADD NEW DELAY

Delay Type	Voyage	Duration	
AGENT DELAY	Vessel Name Voyage#: 0000	45 Mins	Close
CRANE DELAY	Jan Caribe Voyage#: 1045	10 Mins	Close

Figure 4-1: View Delays




Domain Model

UC - 3 (View Main Dashboard)

Extracting Responsibilities

Responsibilities Description	Type	Concept Name
RS.1 Coordinate actions of concept associated with the use case and delegate the work to other concepts	D	Controller
RS. 2 System displays login screen	D	LOGIN_SCREEN
RS. 3 Enters login credential and click login	K	USER_INFO
RS. 4 System Verifies credentials	K	LOGIN_CONFIR MATION
RS. 5 If logins are correct,this is a successful login and main dashboard is displayed	D	DASHBOARD
RS. 6 If logins are incorrect user gets a prompt that logins are incorrect and to try again	K	PROMPT_MESSA GE

Extracting Association

Concept Pair	Association Description	Association Name
Controller  Login Menu	Controller passes request to login form and gets login page displayed	Login Form Request
Login Menu  Interface	Login form is prepared for interface page	UI Preparation
Interface  Controller	Login form is displayed	Displays

Controller ↓◇ Login Menu	Enters User Credentials	User Input
Controller ↓◇ Login Menu	Submits the credentials in the form	Login
Login Menu ↓◇ Database	Database searches for a match with what was inputted	Data Search
Database ↓◇ Login Menu	Logins match one in DB	Data Comparison.
Login menu ↓◇ Dashboard Menu	Login menu switches to Dashboard Menu	Dashboard Menu
Dashboard Menu ↓◇ Interface	Dashboard Menu is being prepared for interface	UI Preparation
Interface ↓◇ Controller	Dashboard is Displayed	Displays

Extracting Attributes

Concept	Attributes	Attributes Description
GET_LOGIN_MENU		
ENTER_LOGIN_INFO		
VALIDATE_LOGIN		
GET_DASHBOARD		
DISPLAY_DASHBOARD		

User Effort Estimation

Figure 1 - Voyage Detail

Navigation - 1 Click

Click ship icon on schedule

Data Entry - 13 Clicks 5 keystrokes

Click on “ETA” date

Keystroke date in field

Click on “ETA” time

Keystroke time in field

Click on “ETD” date

Keystroke date in field

Click on “ETD” time

Keystroke time in field

Click on “Agent”

Click agent name from list

Click on “Voyage”

Keystroke number of voyages

Click on “Vessel Name”

Click vessel name from list

Click on “Vessel Flag”

Click vessel flag from list

Click on “Vessel Type”

Click vessel type from list

Figure 2 - Operation Detail

Navigation - 1 Click

Click tools icon on schedule

Data Entry - 12 Clicks (Potentially 16 depending on Drivers) 6 Key Strokes

Click on “ATA” date

Keystroke date in field

Click on “ATA” time

Keystroke time in field

Click on “ATD” date

Keystroke date in field

Click on “ATD” time

Click on “Pilot”

Click pilot name from list

Click on “Operation Supervisor”

Click Operation Supervisor name from list

Click on “Stevedore Gang”

Click Stevedore Gang name from list

Click on “Gang Foreman”

Click Gang Foreman name from list

Click on “Crane Operator”

Click Crane Operator name from list

Click on “Stacker Operator”

Click Stacker Operator name from list

Click on “Drivers”

Click (several) driver name(s) from list

Click on “Delay”

Click delay type from list

Click on Delay Date

Keystroke date in field

Click on Delay Time

Keystroke date in field

Figure 3 - Delay Table

Navigation - 1 Click

Click on document icon

Date entry - 1 Click 1 keystroke (Multiple clicks and keystrokes depending on how many delays)

Click on blank Completion Time cell

Keystroke the time

Figure 4 - Berthing Schedule

Data Entry - 3 Clicks 1 Drag 1 double-click

Drag schedule form to change time

Double click on form to bring up the "Voyage Detail"

Click on ship icon to bring up a "Voyage Detail" form

Click on tools icon to bring up "Operation Detail" form

Click on documents icon to bring up "Delay Table"

Design of Tests

For this project we are testing the main 5 use cases that are crucial for the system. All of these five use cases will need the use of the database because our system is mostly database oriented. For this reason in order to test these use cases a database connection must be established. The main use cases that will be used are: , addVessel, updateVoyage, notificationsSender and trackDelays. For this test only the functionality will be tested if it passes or fails the main functionality. Use cases that use the user interface will be tested when the user interface test commences with test users.

Test Case:	TC - 2
Use case being used:	UC - 4 - addVessel
Criteria for success/fail:	For this test to be successful the system should add a voyage to the dynamic table and add the attribute and information of the voyage.
Input Data:	Alphanumeric
Test Procedure:	Expected Result:
Step 1: Establish connection to the database.	Success
Step 2: Call function “addVoyage(voyage)” with an valid data	Success
Step 3: Call function “addVoyage(voyage)” with an invalid data.	Fail
Step 4: Call function “addVoyage(voyage)” with empty data.	Fail

Step 5: Call function “GetVoyage(getVoyage)” with valid voyage attribute info.	Success
Step 6: Call function “GetVoyage(getVoyage)” with an invalid voyage attribute info.	fail

Test Case:	TC - 3
Use case being used:	UC - 5 - updateVoyage & UC - 10 - notificationsSender
Criteria for success/fail:	A successful update of the voyage attribute and info is classified as success.
Test Procedure:	Expected Result:
Step 1: Establish connection to the database.	Success
Step 2: Call function “updateVoyage(voyage_number,update)” with an valid data	Success
Step 3: Call function “updateVoyage(voyage_number,update)” with an invalid data	Fail
Step 4: Call function “updateVoyage(voyage_number,update)” with empty data	Fail
Step 5: Call function “updateVoyage(voyage_number,update)” with an empty update	Fail
Step 6: Call function “notificationsSender(message,voyage_number)” with valid date	Success
Step 7: Call function	Fail

“notificationsSender(message,voyage_number)” with invalid date	
--	--

Test Case:	TC - 4
Use case being used:	UC - 19 - trackDelays
Criteria for success/fail:	This will be a success when it tracks the delay for a vessel.
Input Data:	Alphanumeric, time
Test Procedure:	Expected Result:
Step 1: Establish connection to the database.	Success
Step 2: Call function “trackDelay(voyage_number,id_delay)” with an valid data	success
Step 3: Call function “trackDelay(voyage_number,id_delay)” with missing voyage_number	Fail
Step 4: Call function “trackDelay(voyage_number,id_delay)” with an invalid data	Fail
Step 5: Call function “trackDelay(voyage_number,id_delay)” with an empty id_delay	Fail
Step 6: Call function “trackDelay(voyage_number,id_delay)” with an delay of 0	success

History of Work, Current Status, and Future Work

History of Work and Initial Plan of Work

Initially the team created report one as the guide for our semester project. After reviewing the customer requirements and reviewing the nature of the project we developed report one. It was challenging to set up report one as the guide provided was a bit vague. Even though report one was the initial part of the project the team was simultaneously working on other sections to be incorporated later on. Two of the main components that were worked on were the database and user interfaces.

The gantt chart and the distribution of works table shown below display our initial schedule and how it was intended to sub-divide the work into teams.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Task	subtask	resource name	start	finish	feb			march				april				may	
2						1	2	3	1	2	3	4	1	2	3	4	1	2
3	proposal		osborn	1-Feb-22	5-Feb-22													
4	report 1 - system			1-Feb-22	28-Feb-22													
5	part 1																	
6		proposal statement	osborn	1-Feb-22	5-Feb-22													
7		glossary of terms	mark	5-Feb-22	28-Feb-22													
8		update Requirments	osborn & mark	21-Feb-22	28-Feb-22													
9		on-screen appearance		15-Feb-22	28-Feb-22													
10	part 2			1-Feb-22	28-Feb-22													
11		stakeholders, actors goals	osborn & Mark	1-Feb-22	15-Feb-22													
12		casual description	Driane & Mark	15-Feb-22	28-Feb-22													
13		use case digram	Driane	15-Feb-22	28-Feb-22													
14		traceability matrix	Driane	15-Feb-22	28-Feb-22													
15		fully dressed description	Drinae	15-Feb-22	28-Feb-22													
16		system sequence digram	Driane	15-Feb-22	28-Feb-22													
17		preliminary design	kevin	15-Feb-22	28-Feb-22													
18		user effort estimation	kevin	15-Feb-22	28-Feb-22													
19	part 3			1-Feb-22	28-Feb-22													
20		domain analsis		15-Feb-22	28-Feb-22													
21		plan of work	Driane	21-Feb-22	28-Feb-22													
22		mathematics model		15-Feb-22	28-Feb-22													
23	report 2 system		Justin	1-Mar-22	21-Mar-22													
24	demo 1		Driane & Daniel	24-Mar-22	13-Apr-22													
25	report 3		Daniel	13-Apr-22	29-Apr-22													
26	demo 2		Driane & Daniel	21-Apr-22	12-May-22													

<u>Task</u>	<u>Team members</u>
Database Design, implementation and testing	Mark Pascual and Justin Chuc
Programming	Daniel Garcia and Driane Perez
User interface design and testing	Kevin Godoy and Osborn Collins

As the project and the days moved along adjustments had to be made to both the schedule and the distribution of work. Subsequent to report one the team shifted its efforts to complete report two and demo one. At this point in time other online resources, that could be used as guides, were located. These other resources gave concrete examples of the various parts and their development. In conjunction with report two the team developed the integration test and unit test for demo one. The feedback from report one and two along with the feedback given for demo #1 and other guidelines are what are being incorporated into the final group report and demo.

Final History

The two tables below list the distribution of work and the timeline after completing report two and demo. There have been deviations from the timeline for various reasons.

Timeline

	<u>Task to complete</u>	<u>Personnel in charge</u>	<u>Start Date</u>	<u>Finish Date</u>
1	Revision of Reports #1 and #2	All Group Members	10th/04/2022	13th/04/2022
2	Make corrections of Reports #1 and #2	All Group Members	14th/04/2022	14th/04/2022
3	Combing Report #1 and #2 (Report #3)	All Group Members	15th/04/2022	15th/04/2022
4	Finalizing Report #3	All Group Members	16th/04/2022	30th/04/2022

5	Revision of Demo #1	All Group Members	25th/04/2022	30th/04/2022
6	Finalizing Demo #2	Daniel Garcia Osborn Collins Mark Pascual Kevin Godoy Driane Peres	1st/05/2022	13th/05/2022
7	Brochure Preparation PowerPoint Presentation	Mark Pascual Justin Chuc	9th/05/2022	13th/05/2022
8	Revision of all documents and Demo #2 <ul style="list-style-type: none"> • Brochure • PowerPoint • Report #3 • Demo #2 	All Group Members	14th/05/2022	15th/05/2022

Break Down of Responsibility

<u>Task</u>	<u>Team members</u>
Revision of Report #1 and #2 <ul style="list-style-type: none"> • Marking corrections • Merging Reports #1 and #2 • Making final insertions and adjustments to Report #3 • 	All Team members
Demo #2 <ul style="list-style-type: none"> • Interface • Database • Programming • Testing 	Daniel Garcia Driane Perez Osborn Collins Mark Pascual Kevin Godoy

May 16th - Presentation of Demo #2 <ul style="list-style-type: none"> ● Brochure Preparation ● PowerPoint preparation 	Mark Pascual Justin Chuc
---	-----------------------------

Current accomplishments

The UI for the application was completely revamped from the initial iteration. The design is much more colorful and appears to have a design that is compact and easy to follow. Of course, the final design is one that would need to be tested by users. The team invested quite a considerable amount of time and effort into designing the database. As one would know, behind every great application is a robust and well designed database, at least for the most part. The application, even though working as expected, still needs tweaking.

Future work

It would be great to have one of the ports in Belize pilot the application for a few months and give feedback for further development and improvements. No plans have been made to pilot the application, however, it is something that can be considered for further improvements. The application due to time constraints and available resources was developed as a web based to work on desktop computers and laptops. We did not focus on a mobile version of the application or a responsive design. Knowing that the application is one that is designed to improve the efficiency of a port and be in the hands of workers in the field, these are features that must be developed in future iterations.

Project Management

1. Merging the contributions

As is listed in the history of our work we had various timelines listed out. Tasks were assigned to group members. We started off by listing multiple topics for our semester project. In conjunction with the lecturer we decided to do a berthing system application for a port. After reviewing the documentation for report one, having a session with Osborn Collins to further explain the berthing system we began by selecting various parts to complete. A Google, for report one was opened and everyone was asked to deposit their contribution to the document. After report one, as can be seen in the schedule we began working on report two, demo one and the relevant documentations. Again following the same pattern as with report one we selected various tasks and again made contributions to a shared Google doc. Simultaneously, we worked on the integration and unit tests for Demo one.

The feedback from reports one and two were used to make further adjustments to our final report and demo two. Similar to the previous two reports the work was subdivided. All members of the group were to make corrections and insert the new sections that were required. The UIs initially proposed were scrapped and new designs were created. The application develops show the use cases that are to be implemented.

2. Project Coordination

Group members were assigned tasks and responsibilities. The various tasks had deadlines by which they are to be inputted into a shared Google document. The group kept in contact via a WhatsApp chat group. The chat group was the main mode of communication for the group. Occasionally, meetings were held using Google Meet and Zoom. In these meetings discussion about the progress of the project took place and further clarifications were made on various points. The weekend of the 14th and 15th May, 2022 was used to complete the various documentations and place the final touches on the demo.

3. Work schedules

The diagram below shows the gantt chart outline our initial course of work. After completing report one modifications were made and these changes can be seen reflected in the second table shown below. Even though there were deviations from the second schedule, these were only few.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
1	Task	subtask	resource name	start	finish	feb			march				april				may	
2						1	2	3	1	2	3	4	1	2	3	4	1	2
3	proposal		osborn	1-Feb-22	5-Feb-22													
4	report 1 - system			1-Feb-22	28-Feb-22													
5	part 1																	
6		proposal statement	osborn	1-Feb-22	5-Feb-22													
7		glossary of terms	mark	5-Feb-22	28-Feb-22													
8		update Requirments	osborn & mark	21-Feb-22	28-Feb-22													
9		on-screen appearance		15-Feb-22	28-Feb-22													
10	part 2			1-Feb-22	28-Feb-22													
11		stakeholders, actors goals	osborn & Mark	1-Feb-22	15-Feb-22													
12		casual description	Driane & Mark	15-Feb-22	28-Feb-22													
13		use case digram	Driane	15-Feb-22	28-Feb-22													
14		traceability matrix	Driane	15-Feb-22	28-Feb-22													
15		fully dressed description	Drinae	15-Feb-22	28-Feb-22													
16		system sequence digram	Driane	15-Feb-22	28-Feb-22													
17		preliminary design	kevin	15-Feb-22	28-Feb-22													
18		user effort estimation	kevin	15-Feb-22	28-Feb-22													
19	part 3			1-Feb-22	28-Feb-22													
20		domain analisis		15-Feb-22	28-Feb-22													
21		plan of work	Driane	21-Feb-22	28-Feb-22													
22		mathematics model		15-Feb-22	28-Feb-22													
23	report 2 system		Justin	1-Mar-22	21-Mar-22													
24	demo 1		Driane & Daniel	24-Mar-22	13-Apr-22													
25	report 3		Daniel	13-Apr-22	29-Apr-22													
26	demo 2		Driane & Daniel	21-Apr-22	12-May-22													

	<u>Task to complete</u>	<u>Personnel in charge</u>	<u>Start Date</u>	<u>Finish Date</u>
1	Revision of Reports #1 and	All Group	10th/04/2022	13th/04/2022

	#2	Members		
2	Make corrections of Reports #1 and #2	All Group Members	14th/04/2022	14th/04/2022
3	Combing Report #1 and #2 (Report #3)	All Group Members	15th/04/2022	15th/04/2022
4	Finalizing Report #3	All Group Members	16th/04/2022	30th/04/2022
5	Revision of Demo #1	All Group Members	25th/04/2022	30th/04/2022
6	Finalizing Demo #2	Daniel Garcia Osborn Collins Mark Pascual Kevin Godoy Driane Peres	1st/05/2022	13th/05/2022
7	Brochure Preparation PowerPoint Presentation	Mark Pascual Justin Chuc	9th/05/2022	13th/05/2022
8	Revision of all documents and Demo #2 <ul style="list-style-type: none"> ● Brochure ● PowerPoint ● Report #3 ● Demo #2 	All Group Members	14th/05/2022	15th/05/2022

4. Breakdown of Responsibility

The table below shows the breakdown of tasks. The list shown below was followed for the most part.

<u>Task</u>	<u>Team members</u>
Revision of Report #1 and #2 <ul style="list-style-type: none">● Marking corrections● Merging Reports #1 and #2● Making final insertions and adjustments to Report #3●	All Team members
Demo #2 <ul style="list-style-type: none">● Interface● Database● Programming● Testing	Daniel Garcia Driane Perez Osborn Collins Mark Pascual Kevin Godoy
May 16th - Presentation of Demo #2 <ul style="list-style-type: none">● Brochure Preparation● PowerPoint preparation	Mark Pascual Justin Chuc

Summary of Changes

After reviewing reports one and two several adjustments were made based on the recommendations given by the Lecturer. There were several instances where in the listing of the actors and goals several of the goals included functional requirements that were not listed in the table with the enumerated functional requirements. These additional functional requirements were included in the table. These functional requirements effected changes to the traceability matrix and the fully dressed use cases. The UIs designs incorporated in our first iteration were inadequate and were completely revised. The designs of the UIs incorporated the functional requirements that were updated and included.

References

Menon A. (2021). What are berthing plans - Everything you need to know. Retrieved 25th February, 2022 from <https://www.marineinsight.com/marine-navigation/berthing-plans/#:~:text=Introduction%20to%20Berthing%20Plans,at%20least%20a%20month%20before.>

Lucidchart. (2020, May 13). *Types of UML diagrams*. Introducing Types of UML Diagrams | Lucidchart Blog.

Retrieved February 24, 2022, from <https://www.lucidchart.com/blog/types-of-UML-diagrams>

Rumbli, C. (2021, September 9). *UML diagram types: Learn about all 14 types of UML diagrams*. Creately Blog.

Retrieved February 27, 2022, from <https://creately.com/blog/diagrams/uml-diagram-types-examples/>

Figma. (2020, December 1). *Figma for beginners: Create designs (2/4) - youtube*. Figma.

Retrieved February 27, 2022, from <https://www.youtube.com/watch?v=wwFd-z7jSaA>

Paperform — That's a Wrap. (2021, December 22). YouTube. Retrieved

https://www.youtube.com/watch?v=WQh4ax0li5s&ab_channel=Paperform

<https://berthingschedule.webflow.io/>

<https://www.ece.rutgers.edu/~marsic/books/SE/>