Unmanned Surface Vessel (USV) Fleet Management System

Project Step 4

Alexander Jones and Alexander Lane
Team 45

URL: http://classwork.engr.oregonstate.edu:10067/

Table of Contents

Project Overview and Database Outline

Overview

The United States Navy currently owns an ever growing fleet of approximately 25 Unmanned Surface Vessels (USVs) ranging from small 24-foot reconnaissance craft to medium-sized autonomous vessels roughly 180 ft in length. All of these vessels are modular, meaning that they can be outfitted with one or many payloads or sensor packages. Payloads are either mounted on the back of the vessel or fixed to the mast and include sensors such as a sonar towed array, electronic warfare (EW) suites, and high definition optical sensors. Currently, the personnel assigned to each vessel and the payloads installed on them are all tracked by hand and usually there are many versions of the same file floating around, with minor or major differences. This database-driven website will allow the commands in charge of USVs to have a centralized interface to track USV status (deployed, deployable, training, etc.), which of the 200 crew members are assigned to each vessel, and whether each crew member is qualified or not. Qualifications tracked are limited to USV Operator and USV Supervisor for this application, but could be expanded in the future with the limitless qualifications afforded to the US Navy. With 25 USVs capable of carrying anywhere from one to six payloads, commands can keep track of nearly 150 different mission packages in this database.

Example questions that could be answered (with relative ease) with this project:

- Which deployable USVs currently have a sonar payload installed and a qualified operator assigned?
- Which USV is a particular crew member assigned to?
- When was a particular payload installed on its assigned USV?
- What USVs are currently deployed?
- Is the entire crew of a specified USV fully qualified to operate it?
- What is the vessel class for a USV with a given name?
- What USVs are currently assigned to a mission in a specific location (e.g., "Mediterranean Sea")?
- What is the priorityLevel and location of the mission assigned to a specific USV?

Database Outline

USVs

- Description: Stores the core details and current operational readiness of each Unmanned Surface Vessel (USV).
- Attributes:
    - usvID: int(11), auto_increment, unique, not NULL, PK
    - name: varchar(50), not NULL (e.g. “Seahunter”)
    - class: varchar(50), not NULL (e.g. “MUSV”)
    - status: varchar(25), not NULL (e.g. “Deployable”, “Maintenance”)
    - missionID: int(11), FK, identifies mission(missionID) the USV is assigned to
- Relationships:
    - Has a 0..1 USVs: 0..M Payloads; usvID is a FK in Payloads.
    - Has a 1..M USVs: 0..1 Missions; missionID is a FK in USVs.
    - Has a 1 USVs: 1..M CrewMembers; usvID is a FK in CrewMembers.

Payloads

- Description: Tracks modular sensors and hardware equipment capable of being mounted on various USVs.
- Attributes:
    - payloadID: int(11), auto_increment, unique, not NULL, PK
    - type: varchar(50), not NULL (e.g. “Sonar Towed Array”)
    - serialNumber: varchar(50), unique, not NULL
    - condition: varchar(25), not NULL (e.g. “Operable or Inoperable”)
    - installedUSV: int(11), FK, identifies the primary USV(usvID) the payload is installed on
    - installationDate: date
- Relationships:
    - Has a 0..M Payloads: 0..1 USVs; usvID is a FK in Payloads

CrewMembers

- Description: Records details of the Sailors (Operators and Technicians) responsible for fleet operations.
- Attributes:
    - crewMemberID: int(11), auto_increment, unique, not NULL, PK
    - firstName: varchar(50), not NULL
    - lastName:varchar(50), not NULL
    - rank: varchar(10), not NULL (e.g. “E6” or “O3”)
    - usvID: int(11), FK, identifies the primary USV(usvID) the member is currently assigned to
- Relationships:

    - Has an 0..M CrewMembers: 0..M Qualifications facilitated via the CrewQualifications intersection table.
    - Has a 1..M CrewMembers: 1 USVs; usvID is a FK in CrewMembers.

Qualifications

- Description: Stores a master list of qualifications available to personnel (“USV Operator”, “USV Supervisor”
- Attributes:
    - qualificationID: int(11), auto_increment, unique, not NULL, PK
    - name: varchar(50), not NULL (e.g. “USV Operator”)
- Relationships:
    - Has an 0..M Qualifications: 0..M CrewMembers via the CrewQualifications intersection table.

CrewMemberQualifications (Intersection Table)

- Description: Facilitates the M:M relationship between CrewMembers and Qualifications, allowing a Sailor to hold multiple qualifications.
- Attributes:
    - crewMemberQualificationID: int(11), auto_increment, unique, not NULL, PK
    - crewMemberID: int(11), not NULL, FK; links to CrewMembers(crewMemberID)
    - qualificationID: int(11), not NULL, FK; links to Qualifications(qualificationID).
    - earnedDate: date, not NULL
- Relationships:
    - Facilitates 0..M CrewMembers: 0..M Qualifications relationship.

Missions

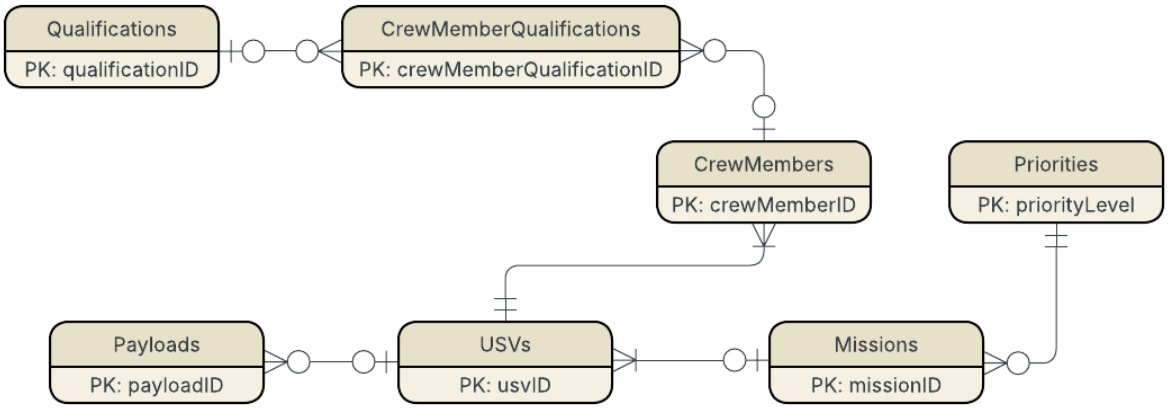
- Description: Records operational objectives, locations, and priority for USV deployment.
- Attributes:
    - missionID: int(11), auto_increment, unique, not NULL, PK
    - title: varchar(100), not NULL (e.g., 'Operation Ghost Net')
    - location: varchar(100), not NULL (e.g. “Southern California”, “Mediterranean Sea”)
    - priorityLevel: int(11), not NULL, FK; links to Priorities(priorityLevel)
- Relationships:
    - Has a 0..1 Missions: 1..M USVs; missionID is a FK in USVs.
    - Has a 0..M Missions: 1 Priorities, FK constraint priorityLevel in Missions.

Priorities

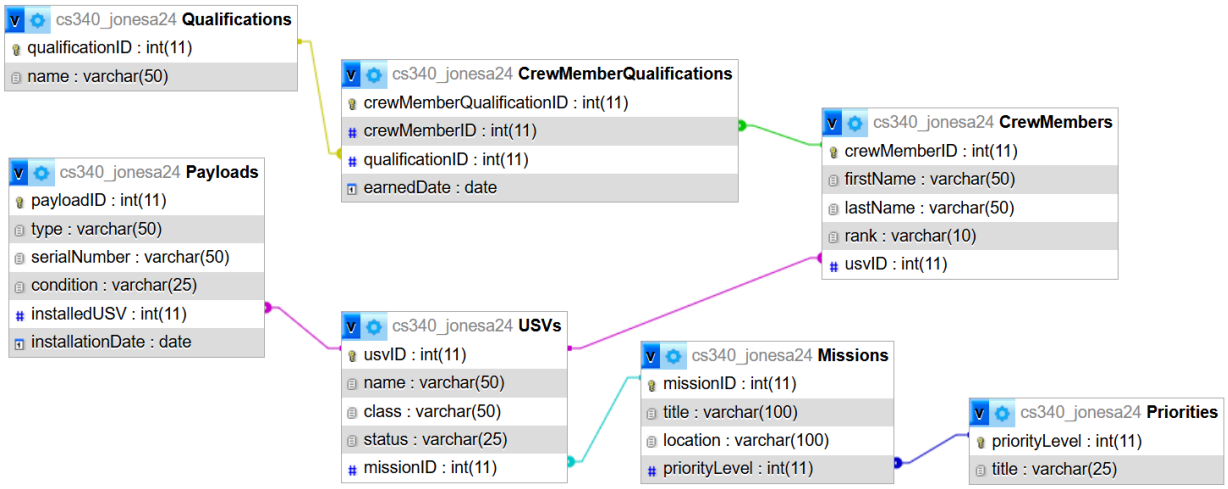
- Description: designates level of “Low”, “Medium”, or “High” based on ID 1, 2, and 3 respectively.
- Attributes:

    - priorityID: int(11), auto_increment, unique, not NULL, PK
    - title: varchar(25), not NULL
- Relationships:
    - Has a 1 Priorities: 0..M Missions, FK constraint priorityLevel in Missions.

Entity-Relationship Diagram



Schema Diagram



Sample Data

Table Name: USVs

usvID	name	class	status	missionID
1	Sentinel	MASC	Deployed	1
2	Striker	MASC	Training	NULL
3	Wraith	GARC	Deployed	3
4	Ghost	GARC	Deployed	3
5	Raider	GARC	Maintenance	NULL

Table Name: Missions

missionID	title	location	priorityLevel
1	Silent Aegis	South China Sea	3
2	Swift Talon	Red Sea	1
3	Pacific Watch	Hawaiian Islands	2

Table Name: Payloads

payloadID	type	serialNumber	condition	installedUSV	installationDate
1	EW	SA-EW-26-001	Operable	1	2025-05-31
2	EW	SA-EW-26-002	Operable	3	2026-01-12
3	EW	SA-EW-26-003	Operable	4	2025-12-20
4	SONAR	DL2-SN-26-101	Operable	1	2025-05-31
5	SONAR	DL2-SN-26-102	Inoperable	2	2025-10-15
6	SONAR	DL2-SN-26-103	Operable	NULL	NULL
7	EO/IR	A7-HD-26-501	Inoperable	NULL	NULL
8	EO/IR	A7-HD-26-502	Operable	1	2025-05-31
9	EO/IR	A7-HD-26-503	Operable	2	2025-10-15

Table Name: Priorities

priorityLevel	title
1	Low
2	Medium
3	High

Table Name: CrewMembers

crewMemberID	firstName	lastName	rank	usvID
1	Marcus	Thorne	O-3	1
2	Sarah	Jenkins	RW1	1
3	Felipe	Torres	ET2	1
4	Davis	Miller	O-3	2
5	James	Shaw	ET1	2
6	Maya	Rodriguez	RW2	2
7	Christopher	Evans	O-3	3
8	Liam	Foster	RW1	3
9	Samantha	Reed	RW2	3
10	Hannah	White	O-3	4
11	Caleb	Wright	ET1	4
12	Sophia	Morales	ET2	4
13	Michael	Sterling	O-3	5
14	Aaron	Choi	RW1	5
15	Ryan	Bennett	RW2	5

Table Name: Qualifications

qualificationID	name
1	USV Craft Master
2	USV Supervisor
3	USV Operator

Table Name: CrewMemberQualifications

crewMemberQualificationID	crewMemberID	qualificationID	earnedDate
1	1	1	2024-10-01
2	1	2	2024-06-01
3	1	3	2024-03-01
4	2	2	2024-12-01
5	2	3	2024-08-01
6	3	3	2025-08-20
7	4	1	2024-10-01
8	4	2	2024-06-01
9	4	3	2024-03-01
10	5	2	2024-12-01
11	5	3	2024-08-01
12	7	1	2024-10-01
13	7	2	2024-06-01
14	7	3	2024-03-01
15	8	2	2024-12-01
16	8	3	2024-08-01
17	9	3	2025-08-20
18	10	1	2024-10-01
19	10	2	2024-06-01
20	10	3	2024-03-01
21	11	2	2024-12-01
22	11	3	2024-08-01
23	13	1	2024-10-01
24	13	2	2024-06-01
25	13	3	2024-03-01
26	14	2	2024-12-01
27	14	3	2024-08-01

Feedback by Peer Reviewers

Step 1 Feedback

Reviewer: Sy Soto

> Looks like a well-scoped design, and the ERD matches the written description clearly. The use of intersection tables for both USV–Payloads and CrewMembers–Qualifications makes sense and shows a good understanding of M:M relationships from what I've read and undertstand. One thing to think about is whether CrewMembers should really be limited to a single “primary” USV via a 1:M relationship, since in real operations sailors might rotate or be temporarily assigned elsewhere. Solid work from my point of view.

Reviewer: Charles Gilbert

> Project Step 1 Review for Group 45
> (Unmanned Surface Vessel (USV) Fleet Management System)
>
> Reviewed by: Charles Gilbert
>
> • The overview does a great job explaining the problem. The problem being that the Navy is tracking USV crews and payloads manually with multiple files, which can easily get inconsistent. This idea of building a centralized database to manage USV status, crew assignments, and payload configurations makes a lot of sense. One small suggestion would be to include an example question the system could answer, such as: “Which deployable USVs currently have a sonar payload installed and a qualified operator assigned?” (if that’s something this interface would support).
>
> • The overview also includes useful specific facts, for example, that there are about 25 USVs, each capable of carrying multiple payloads, along with roughly 150 crew members and many mission packages. These details help show why a database is needed instead of spreadsheets.
>
> • They describe more than four entities, so they definitely meet the assignment requirement. Their entities include: USVs, Payloads, USVPayloads (the junction table), CrewMembers, Qualifications, CrewMemberQualifications (another junction table), and Missions. I don’t see any major issues here, but I’d just caution them not to keep expanding the scope too much with additional tables.
>
> • The database outline is very clear. For each entity, they explain what it represents, list attributes with data types, and include constraints like PK, FK, NOT NULL, and

UNIQUE. I also liked that they explained design choices. For instance, using a “status” field in USVs to track readiness (deployable, maintenance, etc.). One minor thought is that CrewMembers → USVs is modeled as 1:M, but in real life a sailor might work on multiple vessels over time. That could eventually be modeled as M:M, but for this class, keeping it simple is totally reasonable.

• The relationships look correct overall. There are clear 1:M relationships and at least one M:M relationship (USVs ↔ Payloads, and CrewMembers ↔ Qualifications). The ERD matches the written outline and seems logically sound.

• Naming is mostly consistent. The entities are plural (USVs, Payloads, CrewMembers, Qualifications, Missions) and attributes are singular. A small stylistic suggestion would be to rename USVPayloads to USV_Payloads for readability, and maybe shorten CrewMemberQualifications to Crew_Qualifications, but these are just cosmetic changes.

• Overall, this is a strong, realistic, and well-structured project. The problem is clearly motivated, the entities make sense, the relationships are logical, and the ERD aligns nicely with the written outline. Great Job!

Originality: This review is my own analysis. I used AI (ChatGPT) only to check my writing for major grammatical or clarity issues with the prompt: “Please review my writing for grammatical issues that would cause clarity problems.”

Reviewer: Jordan Smith

Hello Alexanders (Sorry),

This overview hurts my soul, only because it hits a little too close to home with my current job. I unfortunately am completely able to understand why this database would be needed. “tracked by hand and usually there are many versions of the same file floating around, with minor or major differences” these words cut deep…

I see plenty of specific facts: 25 Unmanned USV’s, ~150 feet long, 150 crew, payload capacity, 150 mission packages. I think you both covered quite well the generic facts that would be needed/included. Maybe include the potential number of qualifications? Also, maybe considered using different numbers? It looks like 150 is someone’s favorite number. I usually try to keep all my numbers fairly different (order of magnitude if possible), that way I can just eyeball real quick to see if things are making sense.

Plenty of entities are included in this proposal: USVs, Payloads, CrewMembers, Qualifications, and Missions. Plus a few intersection tables. All of these entities make sense and appear to have reasonable attributes with the correct datatypes. The corresponding relationship descriptions also look logical to me.

Relationships appear correct to me. They follow a logical pathway that is easy to follow along with. Numerous M:M relationships are present.

Everything looks consistent to me. Entity names are consistent along with attribute formats. No recommendations.

Great work! I have no major recommendations, only minor personal preferences.

Everything above is my own work/effort.

Reviewer: Bryanna Rosales-Hernandez

Reviewed by Bryanna Rosales-Hernandez

The problem you are describing is the United Stated Navy tracking personnel assignments and payloads on each vessel by hand. The database website you are proposing will which will allow the commands to have a centralized interface to track the USV status, 150 crew members and qualifications appears to be a great way of solving this issue.

The overview that was a provided gives specific facts of what the solution will provide, for instance the group stated their database website will be able to keep track of "nearly 150 different mission packages." Data revolving the issue was also provided, stating "25 USVs capable of carrying anywhere from one to six payloads,..."

I believe the group did a great job at detailing the four entities with their intersecting tables provided if the relationship was M:M. The 5 main entities were USVs, Payloads, CrewMembers, Qualifications and Missions.

The outline provided does specify in details the description of each entity, which was similar to how it was described in the overview. For instance, the Payloads entity was described as, stores the core details and current operational readiness of each Unarmed

Surface Vessel(USV). The group also described each intersecting table, with it's purpose to ease the M:M relationship.

The group provided two M:M relationships: one between USVs and payload, and another between CrewMembers and Qualifications. The 1:M also appear to be accurately represented in the ERDiagram.

The ERD is consistent in the naming, which makes it easier to read. The attributes are also consistent, it appears that multiple words are used with no spacing but the usage of capitalization helps distinguish words, for example: serialNumber.

Feedback was all original, great job on the overview, it was very detailed and the topic is interesting!

TA Reviewer: Lindsey Clement

The modular payload and crew qualification tracking is a cool and unique angle for this project!

Step 2 Feedback

Reviewer: Shayle Sonoron

Hey Group 45! Great work on this! Here's some things I noticed first and foremost,

Schema, outline, and ERD match each other, no inconsistencies found

Naming conventions are consistent throughout except (maybe?) perhaps operable vs. operational

The outline states "operational" as the example condition while your DDL uses "operable" very minute detail though!

Schema and ERD are clean and readable

Intersection tables are properly formed, CrewMemberQualifications has 2 FKs and properly supports the M:N relationship

Sample data shows no issues, tables seem to be 3NF

A minor suggestion however, add a UNIQUE(crewMemberID, qualificationID) to prevent duplicate qualification entries per person

SQL files has correct syntax, I had no problem importing it

Data types are appropriate

All example data from PDF is correctly inserted in SQL

SQL file is well structured, adding dividers with labels between your create/insert/etc. statements might also be a good addition for comments, great work overall nonetheless!

All work is my own.

Reviewer: Quinton Gonzalez

The schema presents a good outline, matching the ERD with all core entities and attributes addressed. It is easy to read and looks relatively organized. Within the intersection tables I would suggest removing NOT NULL from crewMemberID and qualificationID, as there will be cases where one or the other can be null. The normalization looks great, and I see no issues given the scope of your project. Besides that, intersection tables are made.

In regards to the SQL file, everything seems in good order. I had no issue importing and running the queries. The syntax seemed appropriate, and followed the naming consistency in the pdf file. I have no complaints for the structure of the program.

Step 3 Feedback

Reviewer: Samuel Mock

Does the UI utilize a SELECT for every table in the schema? In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.

Yes, it does! Each page of the web app has a table for the relevant information, which is all grabbed by SELECTs.

Does the UI implement an INSERT form for at least one table in the schema? In other words, there should be UI input fields that correspond to at least one table.

Yep, each page except for the Priorities page has an insert form.

Does the UI have at least one DELETE for any one entity? In other words, is there a form/link/button that will allow the deletion of a row in at least one table?

Yes, USVs, Crew Members, Qualifications, and Qualifications all have delete buttons.

Does the UI have at least one DELETE that will remove things from a M:M relationship? In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

Yes, the Crew Qualifications page has a remove feature that, in the SQL, deletes only a specific relationship from that intersection table without deleting any data from the Crew Members or Qualifications tables.

Is there at least one UPDATE form in the UI for any one entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?

Sort of. Both USVs and Payloads have an update button, but it doesn't take me to an update form, so while the structure is there to get you to an update section, it doesn't actually do so.

Is there at least one UPDATE form in the UI to modify an M:M relationship? In other words, does the UPDATE allow the user to select a different foreign key value to update the intersection table with?

No, the edit buttons (which are for updates I think) don't function so it doesn't allow me to select anything.

Do you have any other suggestions for the team to help with their HTML UI? For example, using AS aliases to replace obscure column names such as fname with First Name.

No, no suggestions. This is actually the best-looking web app from any of the projects I've looked at. Really excellent job on the UI, it looks great and is super easy to navigate.

The only thing I'm not super sure about with the UI is what the update form actually looks like because the edit buttons don't take me anywhere, but everything else is perfect. One other thing that did throw me off a little bit is that the DELETEs are all called delete on every page except for the Crew Qualifications page. I'm not sure if that's by design or accident though, it just seemed odd to me.

This review is fully my own work.

Reviewer: Matthew Gomez

Does the UI utilize a SELECT for every table in the schema? In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.

Yes, there exists a SELECT for each table and entity in the schema.

Does the UI implement an INSERT form for at least one table in the schema? In other words, there should be UI input fields that correspond to at least one table.

Yes, there are several tables that employ a form for INSERTs.

Does the UI have at least one DELETE for any one entity? In other words, is there a form/link/button that will allow the deletion of a row in at least one table?

Yes, there exists several tables wherein a DELETE can be performed for an entity.

Does the UI have at least one DELETE that will remove things from a M:M relationship? In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

Crew Qualifications is an intersection table for an M:N relationship between Crew <-> Qualifications and does, in fact, have a <button> to DELETE a row from the table.

Is there at least one UPDATE form in the UI for any one entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?

Yes, the USVs form has an "edit" button wherein an entity might be updated.

Is there at least one UPDATE form in the UI to modify an M:M relationship? In other words, does the UPDATE allow the user to select a different foreign key value to update the intersection table with?

There is no update form, just an update <button> that, hypothetically, I image will eventually translate to an update form as this team's project continues to evolve. But at this time, there exists no update form in this project's UI.

Do you have any other suggestions for the team to help with their HTML UI? For example, using AS aliases to replace obscure column names such as fname with First Name.

I know a React SPA when I see one! Excellent work. Each tab is instantly responsive to each of my inputs thanks in full to your employment of SPA. One thing that I would instantly remark on is that the form is simply too prominent on your pages. Viewing on my 13" laptop, I'd realized that I had to scroll down to view the actual table, an exemplary point where for I moment I felt that clicking a navigation tab did not give me my intended result.

I do, in fact, like the form, and its design; however I would recommend it be invoked by some sort of event behavior from the user. I recognize that is beyond the scope of the assignment, however, and the issue could entirely be circumvented by simply putting the form below the table.

Reviewer: Jordan Smith

Does the UI utilize a SELECT for every table in the schema? In other words, data from each table in the schema should be displayed on the UI. Note: it is generally not acceptable for just a single query to join all tables and displays them.

Yep, every schema table has been selected and is shown on a webpage.

Does the UI implement an INSERT form for at least one table in the schema? In other words, there should be UI input fields that correspond to at least one table.

Yes, each page (except priorities) has an insert form on the very top of the page.

Does the UI have at least one DELETE for any one entity? In other words, is there a form/link/button that will allow the deletion of a row in at least one table?

Yes, I counted four tables with delete capabilities.

Does the UI have at least one DELETE that will remove things from a M:M relationship? In other words, if an order is deleted from the Orders table, it should also delete the corresponding rows from the OrderDetails table, BUT it should not delete any Products or Customers.

Yes, the crew qualifications are able to be deleted

Is there at least one UPDATE form in the UI for any one entity? In other words, in the case of Products, can productName, listPrice, qtyOnHand, e.g. be updated for a single ProductID record?

Yes there is an edit button on two tables.

Is there at least one UPDATE form in the UI to modify an M:M relationship? In other words, does the UPDATE allow the user to select a different foreign key value to update the intersection table with?

There is a edit button on a M:M table, however it isn’t functionalyet.

Do you have any other suggestions for the team to help with their HTML UI? For example, using AS aliases to replace obscure column names such as fname with First Name.

Great work, you both should be pleased with what has been put together so far. I have no recommendations, you both are clearly taking this assignment above and beyond.

As a reviewer, clearly describe to what extent your feedback to the team was original (e.g. "all my work") or non-original (e.g. used AI tools per Code Citation Tips).

All work is my own, no AI was used.

Reviewer: Collin Vassallo

Hello Group 45!

Your UI shows evidence of SELECT queries, as there is sample data supplied across all the pages' tables.

INSERT queries are present in the UI, as data can readably be added into the tables for each page, besides Priorities.

The UI supports deletion of data in the tables with delete buttons, supporting DELETE queries. The M:M relationship crew qualifications is able to have deletes.

There is an edit button in two tables, including the M:M table, but they are not functional at least for me yet. These would support UPDATE queries

I have no additional recommendations for this site, as it has a lot of the functionality and design laid out well.

This response was all my original work.

TA Feedback: Lindsey Clement

Hi Group 45,

Your DML file is organized and includes the required SELECT, INSERT, UPDATE, and DELETE queries, along with proper JOINs and backend variables. There are a couple of syntax issues in the Payloads queries, like a formatting error in the INSERT and a variable mismatch in the UPDATE, which would stop those statements from running. Please review the Payloads INSERT and UPDATE sections, and check the JOIN and foreign key references to make sure all queries work as expected.
Nice work!

Actions Based on the Feedback

Step 1

From the feedback we received from the class and TAs, we made minor changes from Step 1. There were no changes to our project files other than the Overview section in our Step 1 Final Version. We added example questions that our project will be capable of answering under the Overview. We also changed some of the values to better define the scope of numbers of rows the project will be handling (e.g. increased potential qualifications beyond 2, and crew size up to 200).

In the down time and while Alex Jones was writing the DDL, some design decisions were updated.

- Relationship between **USVs and Missions** was changed from
    - 1 USVs: 1..N Missions to
    - 1..N USVs: 0..1 Missions
    - Rather than a USV having many missions, a mission can have many USVs and a USV is assigned 1 or 0 missions at a time.
    - Added the FK missionID to USVs and removed the FK usvID from Missions
- Relationship between **CrewMembers and Qualifications** was slightly changed from
    - 1..N CrewMembers: 1..N Qualifications to
    - 0..N CrewMembers: 0..N Qualifications
    - CrewMembers can exist without a qualification and a qualification can exist belonging to no CrewMember
- Relationship between **Payloads and USVs** was changed from
    - 1..N Payloads: 1..N USVs to
    - 0..N Payloads: 0..1 USVs
    - Removed intersection table, now a 1-M relationship. A payload can exist on its own without being installed on a USV. Not categorical so each instance is unique. A USV can have no payloads or N payloads.
    - Added FK constraint to Payloads to indicate whether and which USV it is installed on.
- Added **Priorities** table and **separated priorityLevel from Missions**
    - Added 1 Priorities: 0..N Missions
    - A mission must have a priority level. A priority level can appear in 0 to N missions.
    - Added FK constraint to Missions to indicate what the priority level is.

Step 2

Based on feedback received from the step 2 draft, minor changes were made to the overview and DDL SQL file. In the overview, one of the examples for use of the ‘condition’ attribute did not exactly match the wording used in the rest of the files, so that was updated to match. In the DDL SQL file, heading comments were made to separate each section.

Thank you Shayle Senoron and Quinton Gonzales!

Step 3

To address Samuel’s suggestions, we implemented hard-coded queries running in the backend to enable the use of our add, update, and delete buttons. Every button on our webpage is now fully functional. These queries will be replaced by stored procedures in subsequent steps.

To address Matthew’s comments on the form being too prominent on the webpage, we have added an update to the layout of each page to our backlog for the next step. Instead of having the full form featured at the top of each page, we will simply have an “Add” button above our tables that reveals the form once clicked.

To address Lindsey’s comment we fixed the syntax error (missing comma) in the Payload INSERT query. The variable mismatch in UPDATE as far as we could tell was a mismatch between the Part 3 Final version of this doc’s Database Outline rather than a mismatch with the actual schema. We edited the doc to reflect the name change (originally Payload had a “usvID” FK but it was renamed to “installedUSV”).

Upgrades to the Draft Version

There are no changes to the files based on our own changed design decisions. All the PL/SQL procedures, CRUD operations planned for the project, and RESET have been implemented.