Contents

1 Dependencies

We are dependent on the following:

1.1 Ship

The `Ship` class is one of the core components to the entire project. From the Ship class, we require some sort of ship ID which can then be looked up in a shared database consisting of various Ships that the port is able to hold as well as their properties. It is vital that we're given the correct `Ship` object from the previous group as checks will need to be ran on the properties of the ship to find out which pilot needs to be sent out. Pilots have different skillsets and training in regards to the ships that they're able to pilot; this is discussed further below. Checking whether current tide levels are safe for the ship to be brought in is also our responsibility, so we require the ship's tolerance to tide to be recorded somewhere. The following attribute **must** be retrieved from a REST endpoint:

- **Ship ID**: Used to lookup the Ship in the core database and find a pilot with the skills to lead it in.

From here, the **Ship ID** is then used to lookup information about the ship in a database. The following properties must be present:

- **Length**: Needed to find a suitable berth to house the ship and to assign a pilot with the appropriate skill.
- **Width**: Needed to find a suitable berth to house the ship and to assign a pilot with the appropriate skill.
- **Minimum water depth**: Needed to ensure the ship is safe to bring in with current weather conditions.
- **Maximum water depth**: Needed to ensure the ship is safe to bring in with current weather conditions.

1.2 Booking

Similiar to the `Ship` class, `Booking` is a core component which is a class that's likely to be referenced to by multiple groups. One of our uses of the `Booking` class could be to confirm that the ship which is ready to be piloted has a valid booking in place. It's possible that multiple customers refer to the same ship ID, so double-checking could be done as a possible security measure. The following attribute **must** be retrieved from a REST endpoint:

- **Booking ID**: Similiar to the attribute in the `Ship` class; used to verify the authenticity of the ship and to retrieve the **start time** and **end time** of the booking. The booking ID should be retrieved from the same REST endpoint as the ship ID.

From here, the **Booking ID** is used to retrieve the following data:

- **Start Time**: Needed to know when a pilot must be sent out.
- **End Time**: Assuming that a pilot stays with the ship the entire time, needed to know when the pilot is free again. Ideally, a pilot would spend ~20 minutes guiding the ship, bring in others, and then come back at the end time to lead them out.

1.3 Berth

It's uncertain as to whether we are responsible for finding an appropriate berth at the time of bringing in the ship. An ideal scenario would be having ship dimension checks take place at the time of booking, but if this is not the case, then we may be responsible for doing so ourselves.

If we are not responsible for finding an appropriate berth, the following attribute **must** be retrieved from a REST endpoint:

- **Berth ID**: Needed so that the ship can be directed to a berth and to mark the berth as 'occupied'.

More has been written about this in the database section.

2 Classes

2.1 Pilot

The only new class that we need to create is the `Pilot` class. As handling pilots is a job that only the harbour master needs to do, it is not necessary to expose any `Pilot` objects to other sub-modules via REST. The pilot class will require the attributes as follows:

- **Pilot ID**: Used to refer to each unique pilot.
- **Training Level**: A means of identifying a pilot's level of training. Needed to know whether or not the pilot is able to guide the inbound ship.

Other attributes for quality-of-life may include:

- **Forename**: To output their name to the dashboard.
- **Surname**: To output their name to the dashboard.

3 MySQL Database

We will be making use of MySQL as the main means of data storage and handling. We will need to be able to access a database with the **ships**, **bookings**, and (potentially) **berth** tables. These tables should be pre-populated and we should, using an ID retrieved from a REST endpoint, be able to **read** from them. One table that we're currently unsure of as to whether we're responsible for its creation is the **berths** table. It is included below as this will make things easier in the near future in the case that we *are* responsible for it. Tables that we must be able to **read** from include:

- **ships**
- **bookings**

Tables that we are responsible for **creating** include:

- **pilots**
- **pilots_bookings**
- **pilots_training**
- **logins**
- **tides**
- **berths** (potentially)

3.1 pilots

The **pilots** table will contain the following fields:

- **pilot_id** INT (**PRIMARY KEY**, AUTO_INCREMENT, NOT NULL): As explained within the 'classes' section, this is needed as a way to uniquely identify a pilot.
- **forename** VARCHAR (NOT NULL): Not required for computation, but will make the dashboard interface easier to read.
- **surname** VARCHAR, (NOT NULL): Not required for computation, but will make the dashboard interface easier to read.
- **training_id** INT, (NOT NULL): Holds a pilot's training level on a scale from one to ten, with the maximum ship size increasing per increase in level of training. This could be implemented in a different way, such as by explicitly listing the ship IDs a pilot is capable of guiding or perhaps the maximum length / width in centimetres.

An example record:

Pilot ID	Forename	Surname	Level of Training
39286034	John	Smith	6

3.2 pilots_bookings

The **pilots_bookings** table will be used to link the **pilots** table with **bookings**. The table will contain the following fields:

- **pb_id** INT (**PRIMARY_KEY**, AUTO_INCREMENT, NOT NULL): Unique ID of the pilot-booking assignment.
- **pilot_id** INT (FOREIGN KEY [pilots], NOT NULL): ID of the pilot handling the booking.

- **booking_id** INT (FOREIGN KEY [bookings], NOT NULL): ID of the booking that is being handled by the pilot.
- **start_time** DATE ([bookings] NOT NULL): Start time of the booking.
- **end_time** DATE ([bookings] NOT NULL): End time of the booking.

An example record:

Pilots_Booking ID	Pilot ID	Booking ID	start_time	end_time
59236985	39286034	95842175	2021-10-16 16:00:00	2021-10-16 17:00:00

The **start_time** and **end_time** are subject to change as it's currently unclear as to how the time slots will be represented in the **bookings** table. This method of scheduling assumes that a single pilot has to stay with the ship the entire time of unloading. This may prove to be the wrong approach as a pilot could deal with other ships whilst the current one is being unloaded; how long it takes for the piloting process is unclear. Computing whether or not a pilot is currently available may prove to be somewhat difficult with this method of scheduling.

3.3 pilots_training

This table maps a '**training_id**' to its values. Each pilot will have a '**training_id**' as a numerical value which is looked up within this table to determine whether or not the pilot has sufficient training to handle the inbound ship. The table will contain the following fields:

- **training_id** INT (**PRIMARY_KEY**, AUTO_INCREMENT, NOT NULL): ID of the training level.
- **maximum_length** INT (NOT NULL): Maximum length of the ship that the pilot is permitted to board.
- **maximum_width** INT (NOT NULL): Maximum width of the ship that the pilot is permitted to board.
- **maximum_weight** INT (NOT NULL): Maximum weight of the ship that the pilot is permitted to board.

3.4 logins

This table will contain the logins for the variety of users that will be accessing the harbour master system. The types of users may include: **admins**, **pilots**. There will only be a single **admin** user as admins have no need to distinguish themselves from one another. **pilots** will need access to the dashboard so that they can view their schedule. The table will contain the following fields:

- **Login ID** INT (**PRIMARY KEY**, AUTO_INCREMENT, NOT NULL): Unique ID of the login details.
- **Username** VARCHAR (NOT NULL): Used to log in to the system.

- **Password** VARCHAR (NOT NULL): In an encrypted form for logging in to the main user interface.
- **Role** VARCHAR (NOT NULL): Used to determine whether the user is a 'PILOT' or an 'ADMIN'.
- **Pilot ID** INT: If the user is has the 'PILOT' role, their ID is also recorded. This field may be NULL if the user is an admin.

Example records:

Login ID	Username	Password	Role	Pilot ID
48297843	jsmith87	825a50df0054e5c72d90ebee7cd...	PILOT	39286034
19396854	admin	a0f1b5ce8c9d848f7c42d82f76b...	ADMIN	NULL

3.5 tides

The **tides** table will store the past and near future available tide data. The table will contain the following fields:

- **tide_id** INT (**PRIMARY KEY**. AUTO_INCREMENT, NOT NULL)
- **height** FLOAT (NOT NULL): Height of the tide in metres; needed to check whether it's safe for the ship to come in.
- **until** DATE (NOT NULL): The time at which the current tide height changes.
- **type** VARCHAR (NOT NULL): The type of tide (HIGH or LOW).

Example records:

Tide ID	Height	Until	Type
74	1.00	2021-10-19 10:36	LOW
75	4.39	2021-10-19 13:36	HIGH

3.6 berths

We may or may not be responsible for creating the **berths** table. It is recommended that a berth is approximately 10% longer than the longest vessel. The table will contain the following fields:

- **berth_id** INT (**PRIMARY KEY**, AUTO_INCREMENT, NOT NULL): Needed as a way to reference different berths.
- **length** INT (NOT NULL): Used to check whether or not the ship is safe to bring in to this berth (in metres).
- **width** INT (NOT NULL): Used to check whether or not the ship is safe to bring in to this berth (in metres).

Example records:

Berth ID	Length	Width
4	350	50

3.7 Entity Relationship Diagram