**Database Design:**

**Table**: ship\_position\_Details

Primary Key: Id (Django model framework automatically adds)

Unique Key: IMO, position\_dt\_tm

Constraints: All the columns are not null

Indexes: Default index will be created since the key columns are primary and unique keys.

**Script:**

CREATE TABLE ship\_position\_details (

ID int primary key,

IMO bigint (7) NOT NULL,

POSITION\_DT\_TM TIMESTAMP NOT NULL,

LATITUDE DECIMAL (10,8),

LONGITUDE DECIMAL (11,8),

UNIQUE KEY (IMO, POSITION\_DT\_TM),

CREATED\_DATE DATETIME DEFAULT CURRENT\_TIMESTAMP

);

**Table**: Ship\_Details

Primary Key: Id (Django model framework automatically adds)

Unique Key: IMO, Name

Constraints: All the columns are not null

Indexes: Default index will be created since the key columns are primary and unique keys.

**Script**: CREATE TABLE ship\_ detail (

ID int primary key,

IMO bigint (7) NOT NULL,

Name Varchar(30) NOT NULL

UNIQUE KEY (IMO, POSITION\_DT\_TM));

**Backend-Design**:

**Service**:

ImportService class designed to process ships, position import and data clean up from the respective tables.

**API:**

**Endpoint:** api/ships/, api/ships/<IMO>

Designed serializers returns the JSON response.

Classes: getShipDetails, getPositionDetails class-based views fetch the data from the back-end and return json response.

**Testcases:**

TestImportService class runs the testcases of data import into ship\_detail and ship\_position\_details tables, validate the api end points.

Note: Since the application not depending on the user input, written api testcases irrespective of default database creation as part of testcase execution.