

# DBMS MINI PROJECT

## Space Research Station Management System (SRSMS)

### Team

NAME	SRN
SURABHI M	PES1UG23AM325
SMRITHI A S	PES1UG23AM306
TANUU SHREE M	PES1UG23AM336
GAHNAVI B	PES1UG23AM900

### Abstract

The Space Research Station Management System (SRSMS) is designed to efficiently manage astronauts, missions, experiments, and supply allocations in a simulated space station database. The system offers a GUI built using Python's ttkbootstrap library (Cyborg theme) integrated with a MySQL backend. Users can perform CRUD operations, execute stored procedures, and trigger audit logging to monitor database activities in real time.

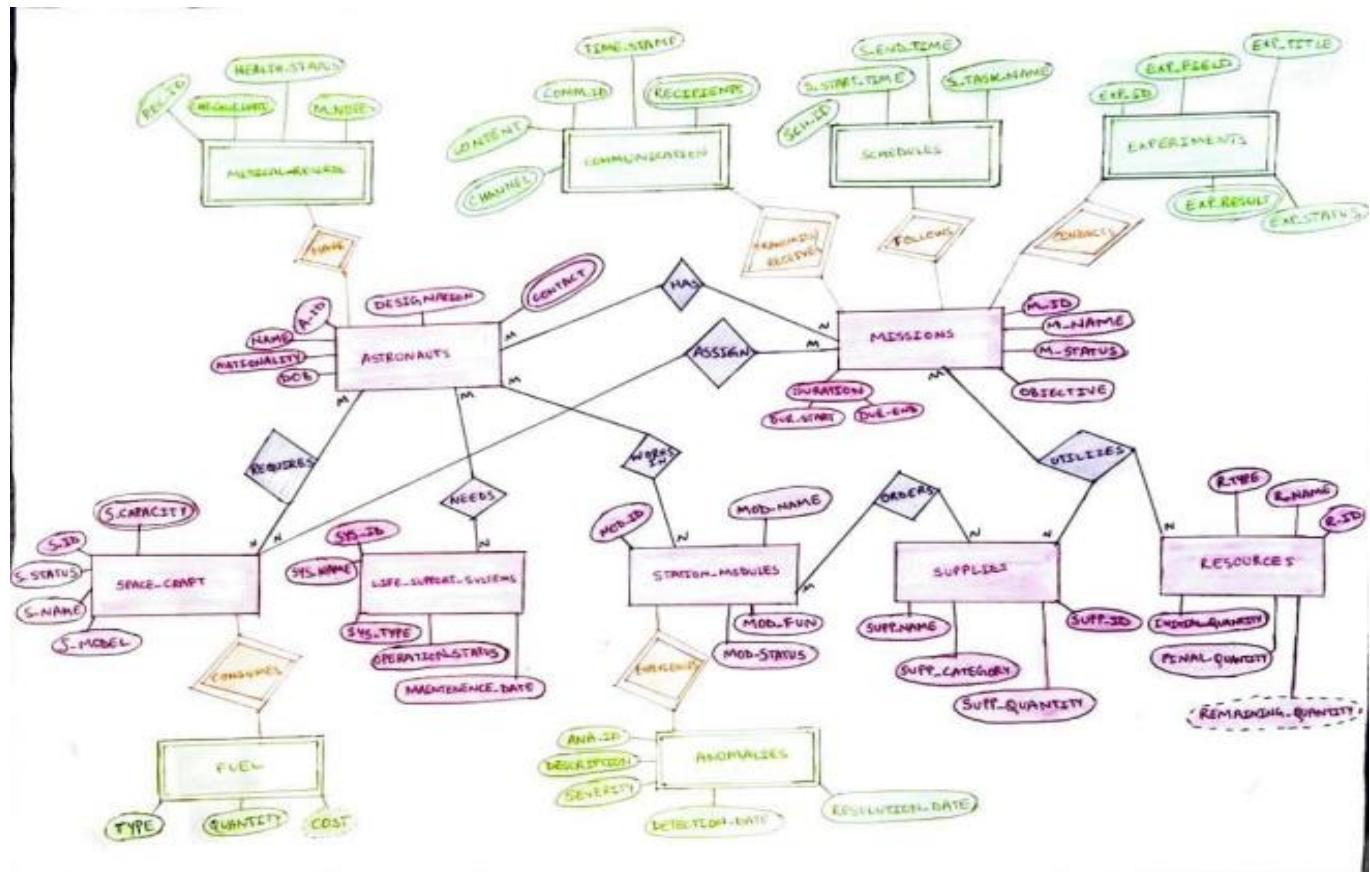
### User Requirement Specification

1. The system must support three user roles: *Admin, Operator, and Viewer*.
2. Admins can perform full CRUD operations and manage audit logs.
3. Operators can insert and update data but cannot delete.
4. Viewers can only view database records.
5. Each table modification must be recorded in the Audit\_Log table using triggers.
6. The GUI must allow viewing, inserting, updating, and exporting table data.

## Tools / Programming Languages Used

1. Python 3.11
2. ttkbootstrap (Cyborg theme)
3. MySQL Workbench 8.0
4. mysql-connector-python
5. pandas (for CSV export)
6. tkinter for GUI layout

## ER Diagram



## **Relational Schema**

### Step I : Mapping Strong Entities

#### Astronauts

A-ID	name	nationality	designation	DOB	contact
------	------	-------------	-------------	-----	---------

#### Missions

M-ID	M-name	M-status	Objective	dur-start	dur-end
------	--------	----------	-----------	-----------	---------

#### Spacecraft

S-ID	s-status	s-name	s-model	s-capacity
------	----------	--------	---------	------------

#### Station-modules

Mod-ID	Mod-name	Mod-fun	Mod-status
--------	----------	---------	------------

#### Life-Support-systems

Sys-ID	Sys-name	Sys-type	operat?.	status	maintenance-date
--------	----------	----------	----------	--------	------------------

#### Resources

R-ID	R-name	R-type	initial-quantity	final-quantity	remaining-quantity
------	--------	--------	------------------	----------------	--------------------

#### Supplies

Supp-ID	Supp-name	Supp-category	Supp-quantity
---------	-----------	---------------	---------------

### Step II : Mapping Weak Entities

#### Experiments

Exp-ID	Exp-field	Exp-field	Exp-title	Exp-result	Exp-status	M-ID
--------	-----------	-----------	-----------	------------	------------	------

#### Schedules

Sch-ID	S-start-time	S-end-time	S-task-name	M-ID
--------	--------------	------------	-------------	------

#### Medical Records

Rec-ID	checkup-date	health-status	m-notes	A-ID
--------	--------------	---------------	---------	------

#### Anomalies

Ana-ID	description	severity	detection-date	resolution-date	Mod-ID
--------	-------------	----------	----------------	-----------------	--------

#### Communication

Comm-ID	channel	recipients	content	time-stamp	M-ID
---------	---------	------------	---------	------------	------

#### Fuel

Type	Quantity	Cost	S-ID
------	----------	------	------

### Step III : Mapping binary 1:1 relationship types

None [fuel is weak entity]

### Step IV : Mapping 1:1 relationships

#### Life-support-systems

Sys-ID	Sys-name	Sys-type	Ops-status	mant-date	Mod-ID
--------	----------	----------	------------	-----------	--------

### Step V : Mapping M:M relationships

#### Has

A-ID	M-ID
------	------

#### Assign

M-ID	S-ID
------	------

#### Requires

A-ID	S-ID
------	------

#### Works-in

A-ID	Mod-ID
------	--------

#### Utilizes

M-ID	R-ID	Supp-ID
------	------	---------

#### Needs

A-ID	Sys-ID
------	--------

#### Creates

Mod-ID	Supp-ID
--------	---------

#### Step VI : Multivalued attributes

#### Astro-no

A-ID	A-no
------	------

#### Spacecraft-capacity

S-ID	S-capacity
------	------------

#### Exp-results

M-ID	Exp-results
------	-------------

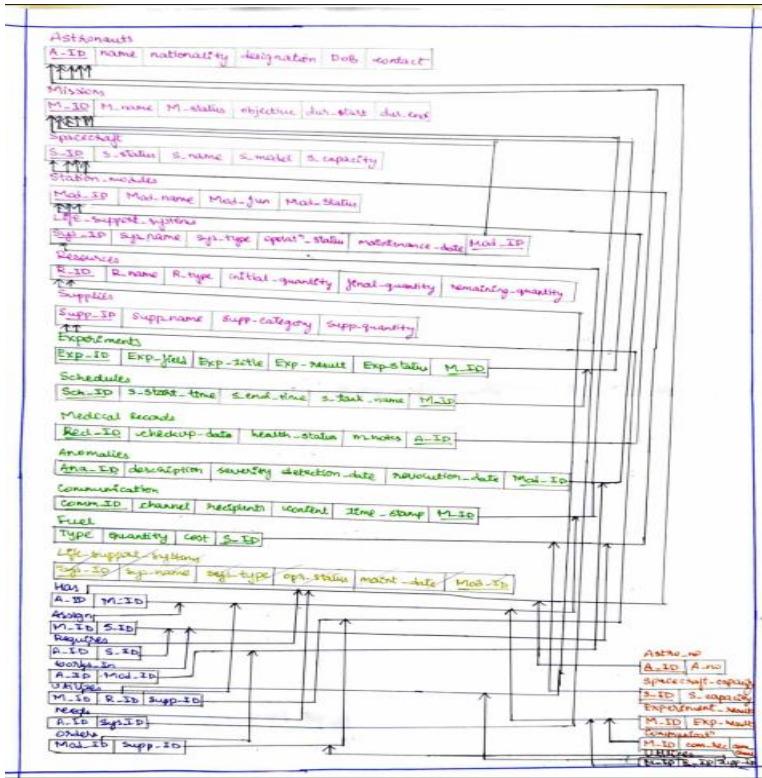
#### Communication

Comm-ID	comm-recipients	comm-channel
---------	-----------------	--------------

### Step VII : N-way relationships

#### Utilizes

M-ID	R-ID	Supp-ID
------	------	---------



## DDL Commands

```
-- =====
```

-- 1) Strong entities

```
-- =====
```

```
CREATE TABLE Astronauts (
```

```
    AstronautID INT AUTO_INCREMENT PRIMARY KEY,
```

```
    FirstName VARCHAR(100) NOT NULL,
```

```
    LastName VARCHAR(100) NOT NULL,
```

```
    DOB DATE,
```

```
    Nationality VARCHAR(80),
```

```
    JobTitle VARCHAR(80),
```

```
    MedicalStatus VARCHAR(60)
```

```
) ENGINE=InnoDB;
```

Astro-no
A-ID   Astro
Spacecraft-capacity
S-ID   S-capacity
Experiment-results
M-ID   EXP-results
Communication
Mod-ID   S-ID
Mod-ID   Supp-ID

```
CREATE TABLE AstronautSkills (
    AstronautID INT NOT NULL,
    Skill      VARCHAR(100) NOT NULL,
    PRIMARY KEY (AstronautID, Skill),
    CONSTRAINT fk_as_sk_as FOREIGN KEY (AstronautID) REFERENCES
    Astronauts(AstronautID) ON DELETE CASCADE
) ENGINE=InnoDB;
```

```
CREATE TABLE Missions (
    MissionID   INT AUTO_INCREMENT PRIMARY KEY,
    MissionName VARCHAR(200) NOT NULL UNIQUE,
    LaunchDate   DATE,
    ReturnDate   DATE,
    MissionType  VARCHAR(80),
    CurrentStatus VARCHAR(20) DEFAULT 'Planned'
) ENGINE=InnoDB;
```

```
CREATE TABLE StationModules (
    ModuleID   INT AUTO_INCREMENT PRIMARY KEY,
    ModuleName  VARCHAR(150) NOT NULL UNIQUE,
    ModuleType  VARCHAR(80),
    Capacity    INT UNSIGNED,
    CurrentStatus VARCHAR(40),
    Location    VARCHAR(120)
) ENGINE=InnoDB;
```

```
CREATE TABLE Resources (
    ResourceID INT AUTO_INCREMENT PRIMARY KEY,
    ResourceName VARCHAR(150) NOT NULL,
    Unit      VARCHAR(50),
    Description TEXT
) ENGINE=InnoDB;
```

```
CREATE TABLE Supplies (
    SupplyID      INT AUTO_INCREMENT PRIMARY KEY,
    ResourceID    INT NOT NULL,
    Quantity      DECIMAL(18,3) NOT NULL DEFAULT 0,
    Unit          VARCHAR(50),
    ExpiryDate    DATE,
    SupplierName  VARCHAR(150),
    StorageModuleID INT,
    CONSTRAINT fk_supplies_resource FOREIGN KEY (ResourceID) REFERENCES
    Resources(ResourceID) ON DELETE CASCADE,
    CONSTRAINT fk_supplies_module FOREIGN KEY (StorageModuleID) REFERENCES
    StationModules(ModuleID) ON DELETE SET NULL
) ENGINE=InnoDB;
```

```
CREATE TABLE LifeSupportSystems (
    SystemID   INT AUTO_INCREMENT PRIMARY KEY,
    ModuleID   INT NOT NULL UNIQUE,
    SystemType VARCHAR(80),
    OxygenLevel DECIMAL(10,3),
    Pressure    DECIMAL(10,3),
```

```
Temperature DECIMAL(6,2),
CO2Level  DECIMAL(10,3),
CurrentStatus VARCHAR(40),
CONSTRAINT fk_lss_module FOREIGN KEY (ModuleID) REFERENCES
StationModules(ModuleID) ON DELETE CASCADE
) ENGINE=InnoDB;
```

```
CREATE TABLE Spacecrafts (
SpacecraftID INT AUTO_INCREMENT PRIMARY KEY,
Name      VARCHAR(150) NOT NULL UNIQUE,
SystemType VARCHAR(80),
CrewCapacity INT UNSIGNED,
CargoCapacity DECIMAL(12,2),
LaunchDate DATE,
CurrentStatus VARCHAR(40)
) ENGINE=InnoDB;
```

```
-- =====
```

```
-- 2) Weak / dependent entities
```

```
-- =====
```

```
CREATE TABLE Experiments (
ExperimentID INT AUTO_INCREMENT PRIMARY KEY,
MissionID   INT NOT NULL,
Title       VARCHAR(250) NOT NULL,
Objective   TEXT,
Category    VARCHAR(80),
```

```
    CurrentStatus VARCHAR(40),  
    ModuleID INT,  
    LeadAstronautID INT,  
    CONSTRAINT fk_experiments_mission FOREIGN KEY (MissionID) REFERENCES  
    Missions(MissionID) ON DELETE CASCADE,  
    CONSTRAINT fk_experiments_module FOREIGN KEY (ModuleID) REFERENCES  
    StationModules(ModuleID) ON DELETE SET NULL,  
    CONSTRAINT fk_experiments_lead FOREIGN KEY (LeadAstronautID) REFERENCES  
    Astronauts(AstronautID) ON DELETE SET NULL  
 ) ENGINE=InnoDB;
```

```
CREATE TABLE Schedules (  
    ScheduleID INT AUTO_INCREMENT PRIMARY KEY,  
    MissionID INT NOT NULL,  
    TaskDescription TEXT,  
    TaskType VARCHAR(80),  
    StartTime DATETIME,  
    EndTime DATETIME,  
    AstronautID INT,  
    CurrentStatus VARCHAR(40),  
    CONSTRAINT fk_schedules_mission FOREIGN KEY (MissionID) REFERENCES  
    Missions(MissionID) ON DELETE CASCADE,  
    CONSTRAINT fk_schedules_astronaut FOREIGN KEY (AstronautID) REFERENCES  
    Astronauts(AstronautID) ON DELETE SET NULL  
 ) ENGINE=InnoDB;
```

```
CREATE TABLE MedicalRecords (  
    RecordID INT AUTO_INCREMENT PRIMARY KEY,
```

```
AstronautID INT NOT NULL,  
CheckupDate DATE,  
HealthCondition VARCHAR(200),  
Treatment TEXT,  
DoctorID INT,  
CONSTRAINT fk_med_astronaut FOREIGN KEY (AstronautID) REFERENCES  
Astronauts(AstronautID) ON DELETE CASCADE,  
CONSTRAINT fk_med_doctor FOREIGN KEY (DoctorID) REFERENCES  
Astronauts(AstronautID) ON DELETE SET NULL  
) ENGINE=InnoDB;
```

```
CREATE TABLE ResourceAllocations (  
AllocationID INT AUTO_INCREMENT PRIMARY KEY,  
MissionID INT NOT NULL,  
SupplyID INT NOT NULL,  
QuantityAllocated DECIMAL(18,3) NOT NULL,  
AllocationDate DATETIME DEFAULT CURRENT_TIMESTAMP,  
CONSTRAINT fk_alloc_mission FOREIGN KEY (MissionID) REFERENCES  
Missions(MissionID) ON DELETE CASCADE,  
CONSTRAINT fk_alloc_supply FOREIGN KEY (SupplyID) REFERENCES Supplies(SupplyID)  
ON DELETE RESTRICT  
) ENGINE=InnoDB;
```

```
CREATE TABLE Communications (  
CommID INT AUTO_INCREMENT PRIMARY KEY,  
MissionID INT NOT NULL,  
AstronautID INT,  
MessageType VARCHAR(80),
```

```

TimeStamp DATETIME DEFAULT CURRENT_TIMESTAMP,
MessageContent TEXT,
Recipient VARCHAR(200),
CONSTRAINT fk_comm_mission FOREIGN KEY (MissionID) REFERENCES
Missions(MissionID) ON DELETE CASCADE,
CONSTRAINT fk_comm_astronaut FOREIGN KEY (AstronautID) REFERENCES
Astronauts(AstronautID) ON DELETE SET NULL
) ENGINE=InnoDB;

```

```

CREATE TABLE Anomalies (
AnomalyID INT AUTO_INCREMENT PRIMARY KEY,
ModuleID INT NOT NULL,
DateDetected DATE DEFAULT (CURRENT_DATE),
Severity ENUM('Low','Medium','High','Critical') NOT NULL,
Description TEXT,
ResolvedByAstronautID INT,
ResolutionDate DATE,
CONSTRAINT fk_anom_module FOREIGN KEY (ModuleID) REFERENCES
StationModules(ModuleID) ON DELETE CASCADE,
CONSTRAINT fk_anom_resolver FOREIGN KEY (ResolvedByAstronautID) REFERENCES
Astronauts(AstronautID) ON DELETE SET NULL
) ENGINE=InnoDB;

```

```
-- =====
```

```
-- 3) mapping tables (composite PKs) - preserved
```

```
-- =====
```

```
CREATE TABLE Astronaut_Missions (
```

```
AstronautID INT NOT NULL,
```

```
MissionID INT NOT NULL,  
Role      VARCHAR(100),  
HoursWorked DECIMAL(10,2) DEFAULT 0,  
PRIMARY KEY (AstronautID, MissionID),  
CONSTRAINT fk_am_as FOREIGN KEY (AstronautID) REFERENCES  
Astronauts(AstronautID) ON DELETE CASCADE,  
CONSTRAINT fk_am_mi FOREIGN KEY (MissionID) REFERENCES Missions(MissionID) ON  
DELETE CASCADE  
) ENGINE=InnoDB;
```

```
CREATE TABLE Mission_Spacecraft (  
MissionID INT NOT NULL,  
SpacecraftID INT NOT NULL,  
AssignmentDate DATE DEFAULT (CURRENT_DATE),  
PRIMARY KEY (MissionID, SpacecraftID),  
CONSTRAINT fk_ms_mi FOREIGN KEY (MissionID) REFERENCES Missions(MissionID) ON  
DELETE CASCADE,  
CONSTRAINT fk_ms_sp FOREIGN KEY (SpacecraftID) REFERENCES  
Spacecrafts(SpacecraftID) ON DELETE RESTRICT  
) ENGINE=InnoDB;
```

```
CREATE TABLE Mission_Modules (  
MissionID INT NOT NULL,  
ModuleID INT NOT NULL,  
AssignmentDate DATE DEFAULT (CURRENT_DATE),  
PRIMARY KEY (MissionID, ModuleID),  
CONSTRAINT fk_mm_mi FOREIGN KEY (MissionID) REFERENCES Missions(MissionID) ON  
DELETE CASCADE,
```

```
CONSTRAINT fk_mm_mod FOREIGN KEY (ModuleID) REFERENCES
StationModules(ModuleID) ON DELETE CASCADE
) ENGINE=InnoDB;
```

```
CREATE TABLE Experiment_Astronauts (
ExperimentID INT NOT NULL,
AstronautID INT NOT NULL,
Role      VARCHAR(80),
PRIMARY KEY (ExperimentID, AstronautID),
CONSTRAINT fk_ea_exp FOREIGN KEY (ExperimentID) REFERENCES
Experiments(ExperimentID) ON DELETE CASCADE,
CONSTRAINT fk_ea_as FOREIGN KEY (AstronautID) REFERENCES
Astronauts(AstronautID) ON DELETE CASCADE
) ENGINE=InnoDB;
```

## CRUD Operation Screenshots

SRSMS - Space Research Station

### SRSMS — Role: admin

**Tables**

Astronauts

[Load Table](#)

[Refresh Tables List](#)

**CRUD**

[Insert](#) [Update](#) [Delete](#) [Refresh](#) [Export CSV](#)

**Procs/Funcs / Triggers**

[sp\\_allocate\\_supply](#) [sp\\_create\\_experiment](#)

[fn\\_mission\\_duration](#) [fn\\_remaining\\_supply](#)

[Audit Log \(refresh\)](#)

Search (simple substring filter):  [Apply](#) [Clear](#)

AstronautID	FirstName	LastName	DOB	Nationality	JobTitle	MedicalStatus
1	Anil	Sharma	1985-07-20	India	Flight Engineer	Fit
2	Sara	Lopez	1990-11-05	Spain	Lead Scientist	Fit
3	James	Owen	1982-03-12	USA	Commander	Fit
4	Mina	Khan	1992-09-25	Pakistan	Medical Officerfff	Fit
6	s	s	1980-11-11	s	s	s

**Audit / Details**

[AuditID](#) [TableName](#)

Join: Astronaut as

**Update Astronauts**

AstronautID	3
FirstName	James
LastName	Owen
DOB	1982-03-12
Nationality	USA
JobTitle	Commander
MedicalStatus	Fit

[Update](#)

**Insert into Astronauts**

AstronautID (auto)	(auto)
FirstName	surabhi
LastName	m
DOB	2000-02-22
Nationality	India
JobTitle	CEO
MedicalStatus	Fit

[Insert](#)

SRSMS - Space Research Station

## SRSMS — Role: admin

Tables

Astronauts

CRUD

Insert Update Delete Refresh Export CSV

Search (simple substring filter):  Apply Clear

Procs/Funcs / Triggers

sp\_allocate\_supply fn\_mission\_duration Audit Log (refresh)

AstronautID	FirstName	LastName	DOB	Nationality	JobTitle	MedicalStatus
1	Anil	Sharma	1985-07-20	India	Flight Engineer	Fit
2	Sara	Lopez	1990-11-05	Spain	Lead Scientist	Fit
3	Charles	Jacob	1982-03-12	USA	Commander In Chie	Fit
4	Mina	Khan	1992-09-25	Pakistan	Medical Officerfff	Fit
6	s	s	1980-11-11	s	s	s
7	surabhi	m	2000-02-22	India	CEO	Fit

Inserted Row inserted successfully. OK

SRSMS - Space Research Station

## SRSMS — Role: admin

Tables

Astronauts

CRUD

Insert Update Delete Refresh Export CSV

Search (simple substring filter):  Apply Clear

Procs/Funcs / Triggers

sp\_allocate\_supply fn\_mission\_duration Audit Log (refresh)

AstronautID	FirstName	LastName	DOB	Nationality	JobTitle	MedicalStatus
1	Anil	Sharma	1985-07-20	India	Flight Engineer	Fit
2	Sara	Lopez	1990-11-05	Spain	Lead Scientist	Fit
3	Charles	Owen	1982-03-12	USA	Commander In Chie	Fit
4	Mina	Khan	1992-09-25	Pakistan	Medical Officerfff	Fit
6	s	s	1980-11-11	s	s	s
7	surabhi	m	2000-02-22	India	CEO	Fit

Updated Row updated. OK

SRSMS - Space Research Station

### SRSMS — Role: admin

Tables: Astronauts

CRUD: Insert, Update, Delete, Refresh, Export CSV

Search (simple substring filter): Apply, Clear

Procs/Funcs / Triggers: sp\_allocate\_supply, fn\_mission\_duration, Audit Log (refresh)

AstronautID	FirstName	LastName	DOB	Nationality	JobTitle	MedicalStatus
1	Anil	Sharma	1985-07-20	India	Flight Engineer	Fit
2	Sara	Lopez	1990-11-05	Spain	Lead Scientist	Fit
3	Charles	Jacob	1982-03-12	USA	Commander In Chief	Fit
6	s	s	1980-11-11	s	s	s
7	surabhi	m	2000-02-22	India	CEO	Fit
8	a	a	2000-11-11	UK	Tech	Fit

Deleted: Row deleted. OK

## Functionalities / Features

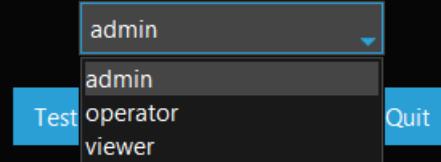
### Feature Description

### Screenshot

Login      Role-based authentication

## SRSMS - Login

Select role to login:



Insert /  
Update  
Astronaut      Operator access

## SRSMS — Role: operator

Tables: Astronauts

CRUD: Insert, Update, Delete, Refresh, Export CSV

Refresh Tables List

**Feature Description**

Delete Record Admin only

**Screenshot**

The screenshot shows the SRSMS application interface for the 'admin' role. At the top, there's a header bar with the SRSMS logo and the text 'SRSMS — Role: admin'. Below the header is a 'Tables' section with a dropdown menu set to 'anomalies', a 'Load Table' button, and a 'Refresh Tables List' button. To the right of the tables section is a 'CRUD' button group containing 'Insert', 'Update', 'Delete', 'Refresh', and 'Export' buttons. A search bar with placeholder text 'Search (simple substring filter)' and buttons for 'Apply' and 'Clear' is located below the tables section. The main area displays a table with the following data:

AnomalyID	ModuleID	DateDetected	Severity	Description
1	2	2025-11-10	Medium	Temperature spike i
2	3	2025-11-10	Low	Minor vibration dete
3	1	2025-11-10	High	Oxygen fluctuation
4	4	2025-11-10	Low	Storage compartment
5	2	2025-11-10	Medium	Humidity increase

A modal dialog box titled 'Confirm' is overlaid on the table, asking 'Delete selected row from anomalies?'. It has 'Yes' and 'No' buttons.

View only Viewer

The screenshot shows the SRSMS application interface for the 'viewer' role. At the top, there's a header bar with the SRSMS logo and the text 'SRSMS — Role: viewer'. Below the header is a 'Tables' section with a dropdown menu set to 'Astronauts', a 'Load Table' button, and a 'Refresh Tables List' button. To the right of the tables section is a 'CRUD' button group containing 'Insert', 'Update', 'Delete', 'Refresh', and 'Export' buttons. The main area displays a table with the following data:

AstronautID	Name	Role	Experience
1	John Doe	Crew Member	5 years
2	Jane Smith	Crew Member	3 years
3	Mike Johnson	Crew Member	7 years
4	Sarah Williams	Crew Member	4 years
5	David Miller	Crew Member	6 years

## Feature Description

Triggers

Astronaut  
must be  
above 22  
years of age

Export  
CSV

Export the  
table record  
as a CSV file

## Screenshot

The screenshot shows the SRSMS application interface. In the center, there is a table view for the 'Astronauts' table. The table has columns: AstronautID, FirstName, LastName, DOB, Nationality, JobTitle, and MedicalStatus. The data is as follows:

AstronautID	FirstName	LastName	DOB	Nationality	JobTitle	MedicalStatus
1	Anil	Sharma	1985-07-20	India	Flight Engineer	Fit
2	Sara	Lopez	1990-11-05	Spain	Lead Scientist	Fit
3	Charles	Jacob	1982-03-12	USA	Commander in Chief	Fit
6	s	s	1980-11-11	s	s	s
7	surabhi	m	2000-02-22	India	CEO	Fit

To the right of the table, a modal window titled 'Update Astronauts' is open, showing the details for AstronautID 7. The 'FirstName' field contains 'surabhi'. Below the table, a message box says 'Update failed' with the error message: '1644 (45000): Astronaut must be at least 22 years old.'.

The screenshot shows the SRSMS application interface. In the center, there is a table view for the 'Astronauts' table. The table has columns: AstronautID, FirstName, LastName, DOB, Nationality, JobTitle, and MedicalStatus. The data is as follows:

AstronautID	FirstName	LastName	DOB	Nationality	JobTitle
1	Anil	Sharma	1985-07-20	India	Flight Eng...
2	Sara	Lopez	1990-11-05	Spain	Lead Scien...
3	Charles	Jacob	1982-03-12	USA	Commander i...
7	surabhi	m	2000-02-22	India	CEO

To the right of the table, a modal window titled 'Save As' is open, allowing the user to save the data as a CSV file. The 'File name:' field is set to 'Astronauts\_20251112\_231105.csv' and the 'Save as type:' field is set to 'CSV files (\*.csv)'. There are 'Save' and 'Cancel' buttons at the bottom of the modal.

## SQL Components

- Triggers for insert/update astronauts under the age of 22

The screenshot shows the SRSMS application interface for the 'admin' role. On the left, the 'Tables' section displays the 'Astronauts' table with columns: AstronautID, FirstName, LastName, DOB, Nationality, JobTitle, and MedicalStatus. A modal window titled 'Update Astronauts' is open, showing fields for AstronautID (7), FirstName (surabhi), LastName (m), DOB (2005-02-22), Nationality (India), JobTitle (CEO), and MedicalStatus (Fit). Below the modal is a message box titled 'Update failed' containing the error message: '1644 (45000): Astronaut must be at least 22 years old.'.

- Stored Procedure: sp\_allocate\_supply

The screenshot shows the SRSMS application interface for the 'admin' role. On the left, the 'Tables' section displays the 'resourceallocations' table with columns: AllocationID, MissionID, SupplyID, QuantityAllocated, and AllocationDate. A modal window titled 'Call sp\_allocate\_supply' is open, showing fields for MissionID (2), SupplyID (2), and Qty (222). Below the modal is a message box titled 'Audit / Details' containing the AuditID (AuditID).

- Stored Procedure: sp\_create\_experiment

SRSMS - Space Research Station

### SRSMS — Role: admin

Tables

experiments

CRUD

Insert Update Delete Refresh Export CSV

Procs/Funcs / Triggers

sp\_allocate\_supply sp\_create\_experiment  
fn\_mission\_duration fn\_remaining\_supply  
Audit Log (refresh)

Search (simple substring filter):  Apply Clear

ExperimentID	MissionID	Title	Objective	Category	CurrentStatus	ModuleID	LeadAstronautID	Audit / Details
1	1	Microgravity Plan	Study plant growth	Biology	Planned	2	2	
2	5	Radiation Shield	Test composite materials	Materials	Planned	3	1	
3	2	Cargo Handling	Improve cargo handling	Engineering	Completed	3	1	
4	3	Thermal Test	System thermal validation	Engineering	Completed	3		
5	4	Lunar Regolith St.	Analyze regolith samples	Geology	Planned	2		
6	1	s	s	s	s	2	2	
7	2	ss	s	ss	s	3	1	
8	2	ss	s	ss	Planned	4	2	

Call sp\_create\_experiment

MissionID:

Title:

Objective:

Category:

ModuleID:

LeadAstronautID:

**Call sp\_create\_experiment**

#### 4. Function: fn\_mission\_duration

SRSMS - Space Research Station

### SRSMS — Role: admin

Tables

missions

CRUD

Insert Update Delete Refresh Export CSV

Procs/Funcs / Triggers

sp\_allocate\_supply sp\_create\_experiment  
fn\_mission\_duration fn\_remaining\_supply  
Audit Log (refresh)

Search (simple substring filter):  Apply Clear

MissionID	MissionName	LaunchDate	ReturnDate	MissionType	CurrentStatus	Audit / Details
1	SRS-Alpha	2026-01-10		Research	Planned	
2	SRS-Resupply-1	2025-11-20		Supply	Active	
3	SRS-Maint-1	2025-12-05	2025-12-20	Maintenance	Completed	
4	Lunar-Test	2027-02-10		Test	Planned	
5	Orbital-Physics	2026-06-01	2026-06-20	Research	Completed	

fn\_mission\_duration

MissionID:

**Call**

Duration: 15

#### 5. Function: fn\_remaining\_supply

SRSMS - Space Research Station

### SRSMS — Role: admin

**Tables**

CRUD

Procs/Funcs / Triggers

SupplyID	ResourceId	Quantity	Unit	ExpiryDate	SupplierName	StorageModule
1	1	9778.000	Liters	2030-01-01	SpaceSupplyCo	1
2	2	4556.000	Liters	2029-05-01	HydroSupplies	4
3	3	2000.000	Kg	2027-12-31	FoodForSpace	4
4	4	150.000	Units		OrbitalParts	3
5	5	50.000	Units		LabKitsInc	2

Search (simple substring filter):  Apply Clear

**fn\_remaining\_supply**

SupplyID: 1

Call

Remaining: 9778.000

## 6. Queries:

- Nested query: Above-average experiments

SRSMS - Space Research Station

### SRSMS — Role: admin

**Tables**

CRUD

Procs/Funcs / Triggers

MissionID	MissionName	expCount
1	SRS-Alpha	2
2	SRS-Resupply-1	3

Queries

Audit / Details

- Join: Astronaut assignments
- Aggregate: Avg Oxygen
- Nested: Above-average experiments

- Join: Astronaut assignments

SRSMS - Space Research Station

SRSMS — Role: admin

Tables

	AstronautID	Name	MissionName	Role
1	1	Anil Sharma	SRS-Alpha	Flight Engineer
1	1	Anil Sharma	SRS-Resupply-1	Engineer
2	2	Sara Lopez	SRS-Alpha	Lead Scientist
3	3	Charles Jacob	SRS-Alpha	Commander

CRUD

Procs/Funcs / Triggers

- sp\_allocate\_supply
- sp\_create\_experiment
- fn\_mission\_duration
- fn\_remaining\_supply
- Audit Log (refresh)

Queries

- Join: Astronaut assignments
- Aggregate: Avg Oxygen
- Nested: Above-average experiments

Search (simple substring filter):  Apply Clear

Audit / Details

AuditID	TableName	Operation	KeyData	NewRow	Change
---------	-----------	-----------	---------	--------	--------

- Aggregate: Average oxygen consumption

SRSMS - Space Research Station

SRSMS — Role: admin

Tables

	ModuleName	AvgOxygen
Habitat-1	20900.0000000	
Lab-Alpha	20850.0000000	
Engineering-1	20750.0000000	
Storage-1	20600.0000000	
Test-Module	20500.0000000	

CRUD

Procs/Funcs / Triggers

- sp\_allocate\_supply
- sp\_create\_experiment
- fn\_mission\_duration
- fn\_remaining\_supply
- Audit Log (refresh)

Queries

- Join: Astronaut assignments
- Aggregate: Avg Oxygen
- Nested: Above-average experiments

Search (simple substring filter):  Apply Clear

Audit / Details

AuditID	TableName	Operation	KeyData	NewRow	Change
---------	-----------	-----------	---------	--------	--------

## Code Snippets for Invoking Procedures/Functions

- Stored Procedure: sp\_allocate\_supply**  

```
DROP PROCEDURE IF EXISTS sp_allocate_supply;
DELIMITER $$

CREATE PROCEDURE sp_allocate_supply(
    IN p_missionid INT,
    IN p_supplyid INT,
    IN p_qty DECIMAL(18,3)
)
BEGIN
    DECLARE cur_qty DECIMAL(18,3);
    DECLARE v_err_msg VARCHAR(255);

    IF p_qty <= 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Quantity must be positive';
    END IF;
```

```

IF (SELECT COUNT(*) FROM Missions WHERE MissionID = p_missionid) = 0 THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Mission not found';
END IF;

START TRANSACTION;
    SELECT Quantity INTO cur_qty FROM Supplies WHERE SupplyID = p_supplyid
FOR UPDATE;
    IF cur_qty IS NULL THEN
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Supply not found';
    END IF;
    IF cur_qty < p_qty THEN
        SET v_err_msg = CONCAT('Insufficient stock. Available: ', cur_qty);
        ROLLBACK;
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = v_err_msg;
    END IF;

    UPDATE Supplies SET Quantity = Quantity - p_qty WHERE SupplyID = p_supplyid;
    INSERT INTO ResourceAllocations (MissionID, SupplyID, QuantityAllocated,
AllocationDate)
        VALUES (p_missionid, p_supplyid, p_qty, NOW());
    COMMIT;
END$$
DELIMITER ;

```

- **Stored Procedure: sp\_create\_experiments**

```

DROP PROCEDURE IF EXISTS sp_create_experiment;
DELIMITER $$
CREATE PROCEDURE sp_create_experiment(
    IN p_missionid INT,
    IN p_title VARCHAR(250),
    IN p_objective TEXT,
    IN p_category VARCHAR(80),
    IN p_moduleid INT,
    IN p_leadastronautid INT,
    OUT p_expid INT
)
BEGIN
    IF (SELECT COUNT(*) FROM Missions WHERE MissionID = p_missionid) = 0 THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Mission not found';
    END IF;

```

```

INSERT INTO Experiments (MissionID, Title, Objective, Category, CurrentStatus,
ModuleID, LeadAstronautID)
    VALUES (p_missionid, p_title, p_objective, p_category, 'Planned', p_moduleid,
p_leadastronautid);
    SET p_expid = LAST_INSERT_ID();
END$$
DELIMITER ;

```

- **Function: fn\_mission\_duration**

```

DROP FUNCTION IF EXISTS fn_mission_duration;
DELIMITER $$

CREATE FUNCTION fn_mission_duration(p_missionid INT)
RETURNS INT DETERMINISTIC
BEGIN
    DECLARE v_launch DATE;
    DECLARE v_return DATE;
    DECLARE v_days INT;
    SELECT LaunchDate, ReturnDate INTO v_launch, v_return FROM Missions WHERE
MissionID = p_missionid;
    IF v_launch IS NULL OR v_return IS NULL THEN
        RETURN NULL;
    END IF;
    SET v_days = DATEDIFF(v_return, v_launch);
    RETURN v_days;
END$$
DELIMITER ;

```

- **Function: fn\_remaining\_supply**

```

DROP FUNCTION IF EXISTS fn_remaining_supply;
DELIMITER $$

CREATE FUNCTION fn_remaining_supply(p_supplyid INT)
RETURNS DECIMAL(18,3) DETERMINISTIC
BEGIN
    DECLARE q DECIMAL(18,3);
    SELECT Quantity INTO q FROM Supplies WHERE SupplyID = p_supplyid;
    RETURN IFNULL(q, 0);
END$$
DELIMITER ;

```

- **Trigger: trg\_check\_astronaut\_dob\_update**

```

DROP TRIGGER IF EXISTS trg_check_astronaut_dob_update;
DELIMITER $$
```

```

CREATE TRIGGER trg_check_astronaut_dob_update
BEFORE UPDATE ON Astronauts
FOR EACH ROW
BEGIN
    IF TIMESTAMPDIFF(YEAR, NEW.DOB, CURDATE()) < 22 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Astronaut must be at least 22 years old.';
    END IF;
END$$

DELIMITER ;

```

- **Trigger: trg\_check\_astronaut\_dob\_insert**

```

DROP TRIGGER IF EXISTS trg_check_astronaut_dob_insert;
DELIMITER $$
```

```

CREATE TRIGGER trg_check_astronaut_dob_insert
BEFORE INSERT ON Astronauts
FOR EACH ROW
BEGIN
    IF TIMESTAMPDIFF(YEAR, NEW.DOB, CURDATE()) < 22 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'Astronaut must be at least 22 years old.';
    END IF;
END$$
```

```
DELIMITER ;
```

```

DROP TRIGGER IF EXISTS trg_check_astronaut_dob_update;
DELIMITER $$
```

## SQL File Reference

All SQL commands, triggers, and stored procedures are saved in srsdb\_script.sql.



## Github Repo Link

<https://github.com/surabhimuralidhar/Space-Research-Station-Management-System.git>

