SatelliteDB

Donald "Trey" Elkins & Adam Ruark
web.engr.oregonstate.edu/~ruarka/cs340/project

I. Introduction

SatelliteDB is a project aimed at creating an easy to access catalog of objects – primarily satellites - orbiting the Earth. The goal is to document and present relevant information (orbital data, ownership, etc.) in such a manner that users will be able to quickly reference the database for personal and educational purposes.

Our target users are individuals interested in satellites and other objects in personal, educational, or research projects. Desired functionalities for the project include a favorites list (on a per-user basis), category selection, and object entry. As a stretch goal, we would like to implement some kind of interactive widget (such as Google Maps) that will provide a more engaging experience for the user.

II. Requirements & Business Rules

Data validity is incredibly important for our system to be usable by the target audience. The information that's uploaded to the database can be cross-checked between several sources, including NASA, NORAD, Wikipedia, and other academic and government data repositories. Lack of data integrity renders the system and project pointless.

As for functionality requirements:

- Users must be able to create an account and login; this will allow for creation of a favorites list, as well as secure access to the form for inputting new objects into the project database.

- The web application for database access must be easily usable and navigable, preferably without looking hideous.

- Web application needs to be able to save and display images and text blurbs, preferably dynamically with JavaScript (or even PHP/SQL) to limit the amount of 'hard coded' web development.

- Ability to sort orbital objects by launch date, owner, orbital period, type, nationality, etc.

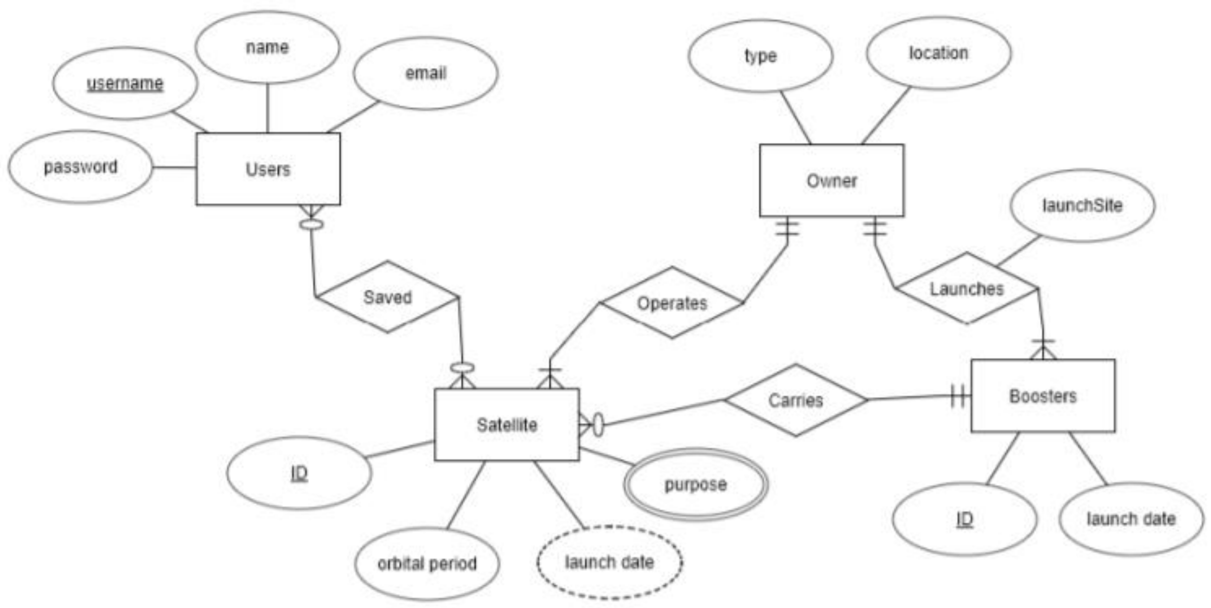
- Users must be able to add satellite objects to the database with relatively robust error catching.

Optional functionality goals:

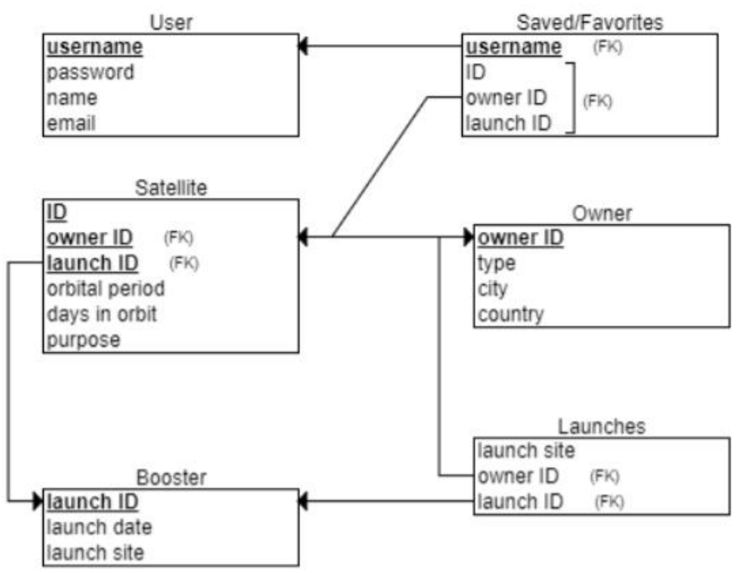
- Password hashing using PHP.
- Some kind of interactive widget – ex. Google maps.

III. Database Design

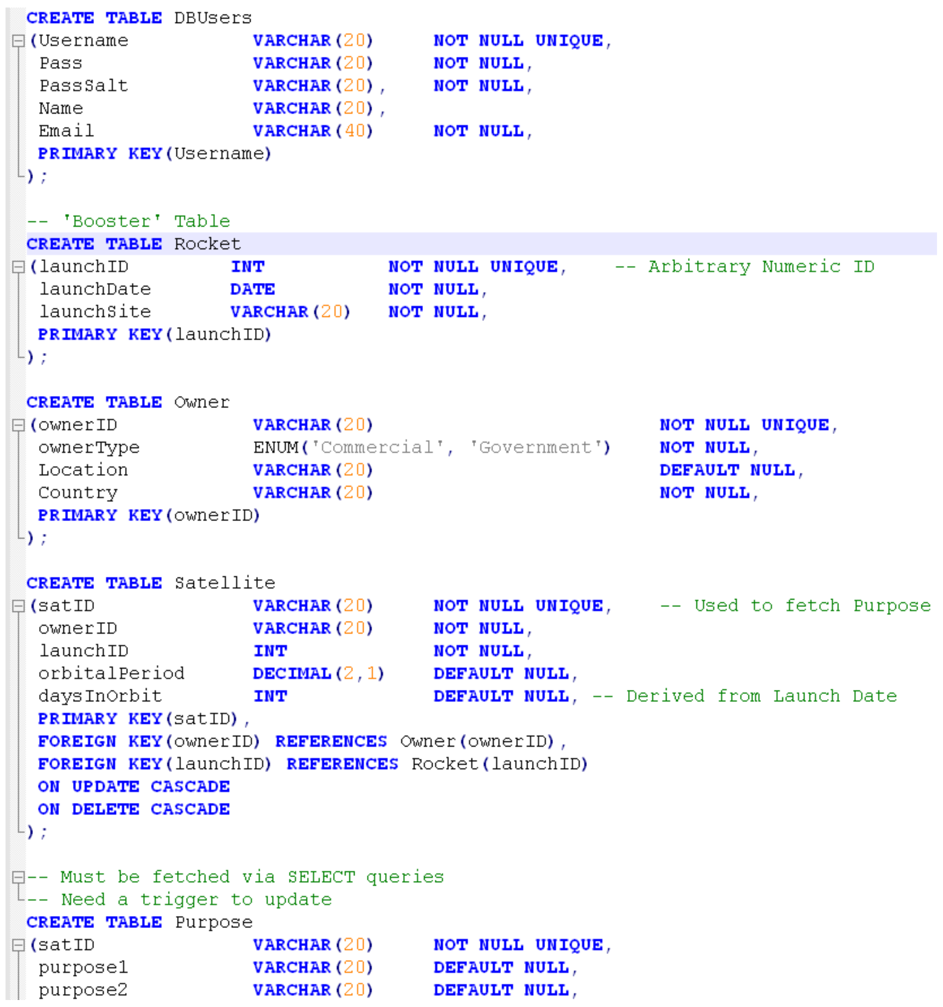
Project ER Diagram:



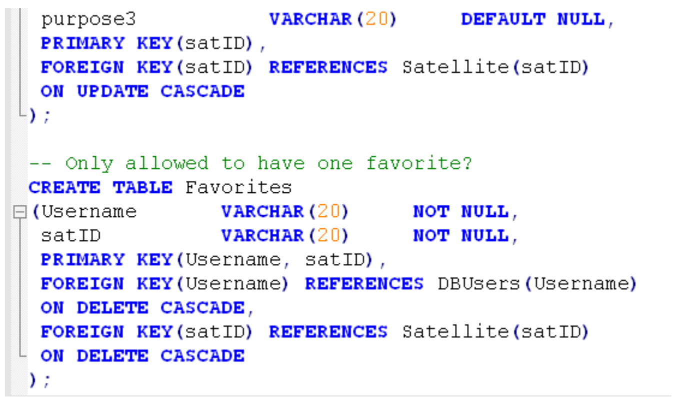
Relation Schema:



Setup Code:



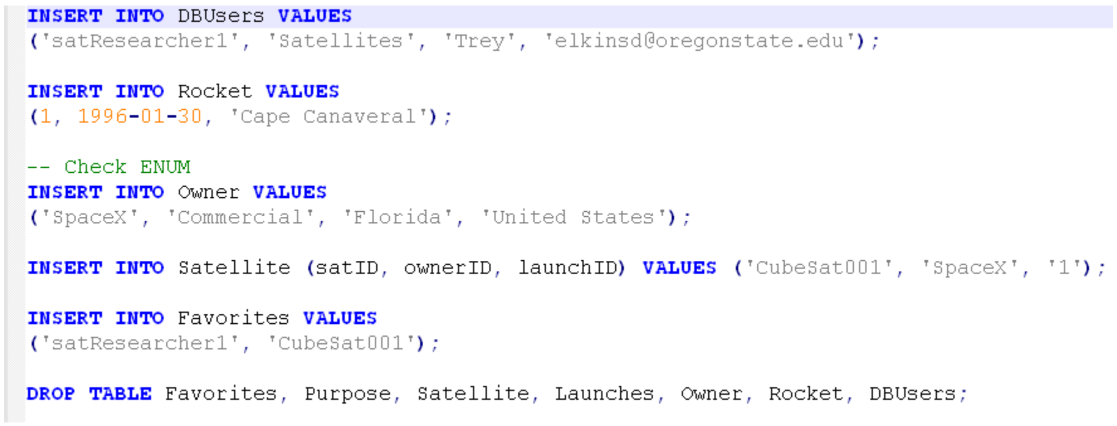
```sql
CREATE TABLE DBUsers
(Username             VARCHAR(20)     NOT NULL UNIQUE,
 Pass                 VARCHAR(20)     NOT NULL,
 PassSalt             VARCHAR(20),    NOT NULL,
 Name                 VARCHAR(20),
 Email                VARCHAR(40)     NOT NULL,
 PRIMARY KEY(Username)
);

-- 'Booster' Table
CREATE TABLE Rocket
(launchID           INT             NOT NULL UNIQUE,    -- Arbitrary Numeric ID
 launchDate         DATE            NOT NULL,
 launchSite         VARCHAR(20)     NOT NULL,
 PRIMARY KEY(launchID)
);

CREATE TABLE Owner
(ownerID              VARCHAR(20)                        NOT NULL UNIQUE,
 ownerType            ENUM('Commercial', 'Government')   NOT NULL,
 Location             VARCHAR(20)                        DEFAULT NULL,
 Country              VARCHAR(20)                        NOT NULL,
 PRIMARY KEY(ownerID)
);

CREATE TABLE Satellite
(satID                VARCHAR(20)     NOT NULL UNIQUE,    -- Used to fetch Purpose
 ownerID              VARCHAR(20)     NOT NULL,
 launchID             INT             NOT NULL,
 orbitalPeriod        DECIMAL(2,1)    DEFAULT NULL,
 daysInOrbit          INT             DEFAULT NULL, -- Derived from Launch Date
 PRIMARY KEY(satID),
 FOREIGN KEY(ownerID) REFERENCES Owner(ownerID),
 FOREIGN KEY(launchID) REFERENCES Rocket(launchID)
 ON UPDATE CASCADE
 ON DELETE CASCADE
);

-- Must be fetched via SELECT queries
-- Need a trigger to update
CREATE TABLE Purpose
(satID                VARCHAR(20)     NOT NULL UNIQUE,
 purpose1             VARCHAR(20)     DEFAULT NULL,
 purpose2             VARCHAR(20)     DEFAULT NULL,
```



```sql
 purpose3             VARCHAR(20)     DEFAULT NULL,
 PRIMARY KEY(satID),
 FOREIGN KEY(satID) REFERENCES Satellite(satID)
 ON UPDATE CASCADE
);

-- Only allowed to have one favorite?
CREATE TABLE Favorites
(Username         VARCHAR(20)     NOT NULL,
 satID            VARCHAR(20)     NOT NULL,
 PRIMARY KEY(Username, satID),
 FOREIGN KEY(Username) REFERENCES DBUsers(Username)
 ON DELETE CASCADE,
 FOREIGN KEY(satID) REFERENCES Satellite(satID)
 ON DELETE CASCADE
);
```

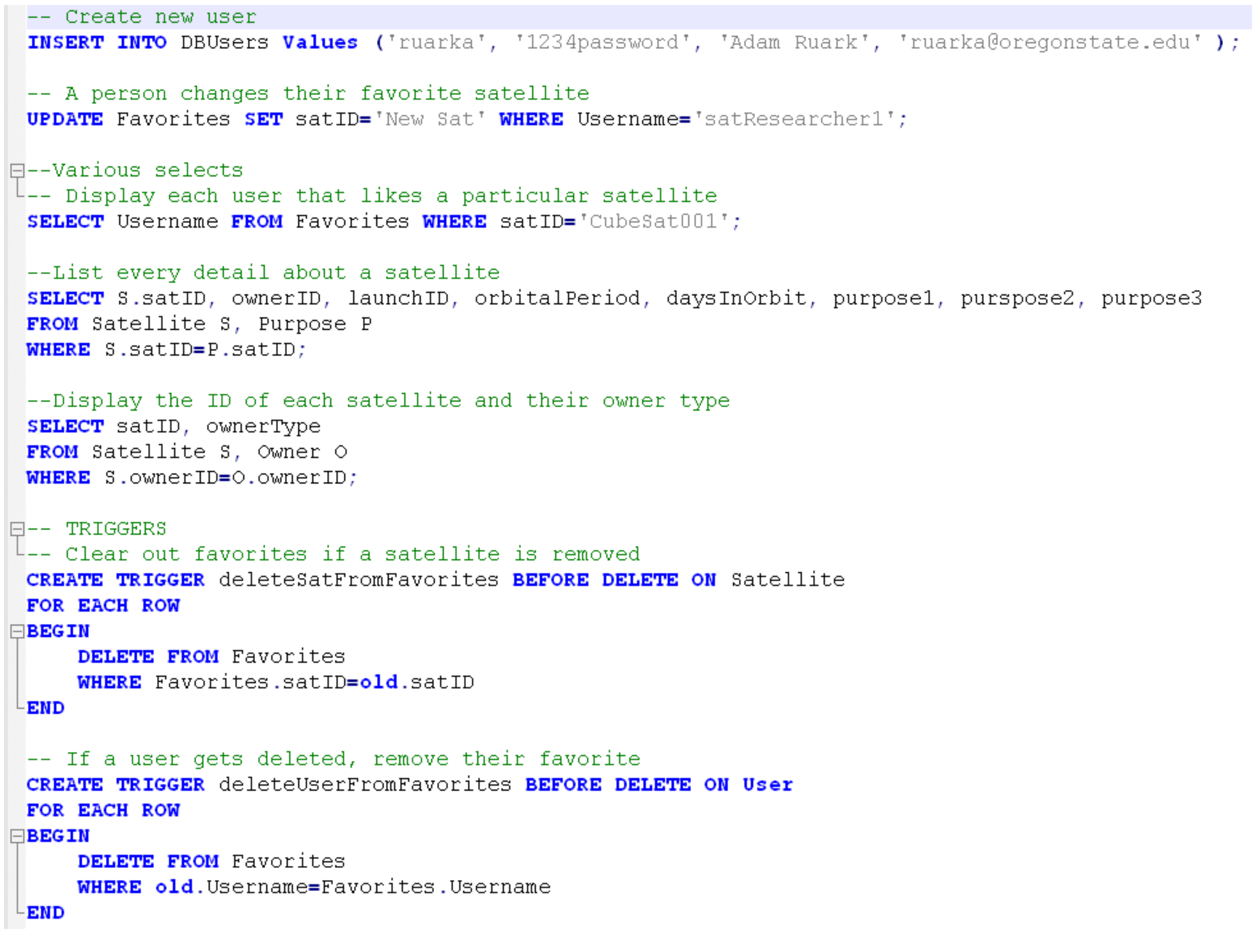
Note that code after this point has not been adjusted for some modifications and is reflective of initial documentation; final implementation will be cohesive, so treat this as prototype code.

Insertion testing code:



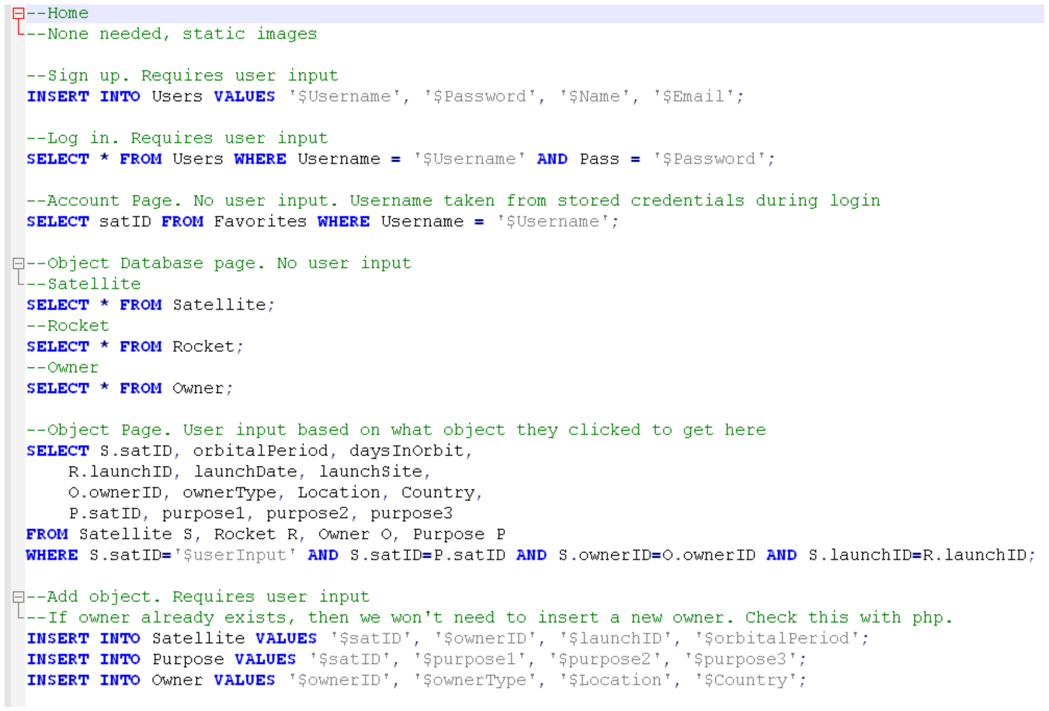
```sql
INSERT INTO DBUsers VALUES
('satResearcher1', 'Satellites', 'Trey', 'elkinsd@oregonstate.edu');

INSERT INTO Rocket VALUES
(1, 1996-01-30, 'Cape Canaveral');

-- Check ENUM
INSERT INTO Owner VALUES
('SpaceX', 'Commercial', 'Florida', 'United States');

INSERT INTO Satellite (satID, ownerID, launchID) VALUES ('CubeSat001', 'SpaceX', '1');

INSERT INTO Favorites VALUES
('satResearcher1', 'CubeSat001');

DROP TABLE Favorites, Purpose, Satellite, Launches, Owner, Rocket, DBUsers;
```

Sample Queries:



```sql
-- Create new user
INSERT INTO DBUsers Values ('ruarka', '1234password', 'Adam Ruark', 'ruarka@oregonstate.edu' );

-- A person changes their favorite satellite
UPDATE Favorites SET satID='New Sat' WHERE Username='satResearcher1';

--Various selects
-- Display each user that likes a particular satellite
SELECT Username FROM Favorites WHERE satID='CubeSat001';

--List every detail about a satellite
SELECT S.satID, ownerID, launchID, orbitalPeriod, daysInOrbit, purpose1, purspose2, purpose3
FROM Satellite S, Purpose P
WHERE S.satID=P.satID;

--Display the ID of each satellite and their owner type
SELECT satID, ownerType
FROM Satellite S, Owner O
WHERE S.ownerID=O.ownerID;

-- TRIGGERS
-- Clear out favorites if a satellite is removed
CREATE TRIGGER deleteSatFromFavorites BEFORE DELETE ON Satellite
FOR EACH ROW
BEGIN
    DELETE FROM Favorites
    WHERE Favorites.satID=old.satID
END

-- If a user gets deleted, remove their favorite
CREATE TRIGGER deleteUserFromFavorites BEFORE DELETE ON User
FOR EACH ROW
BEGIN
    DELETE FROM Favorites
    WHERE old.Username=Favorites.Username
END
```

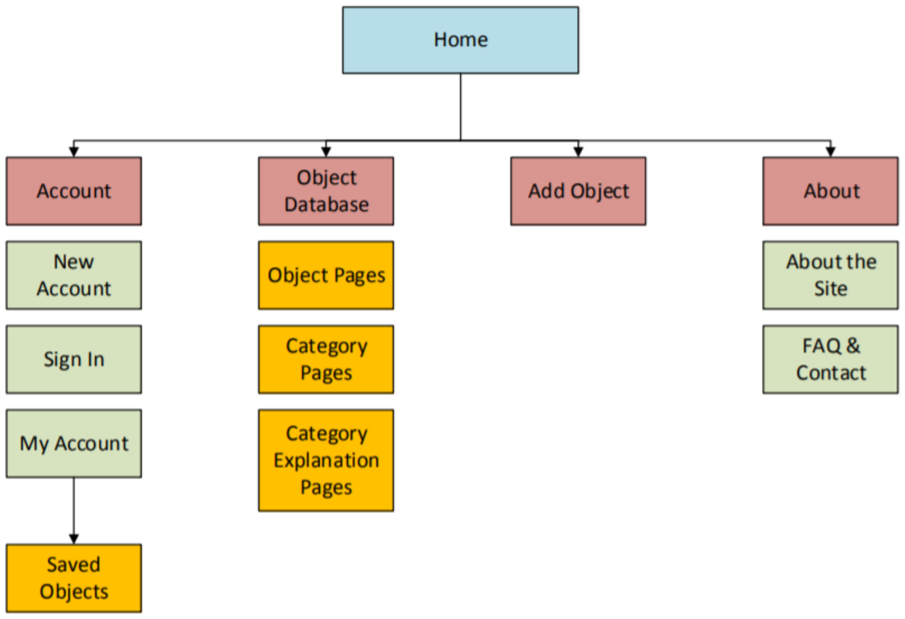
Webpage Queries:



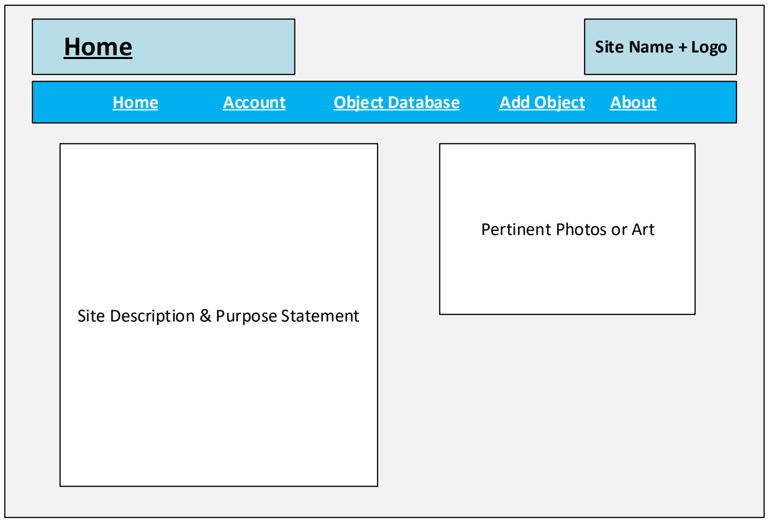
```
--Home
--None needed, static images

--Sign up. Requires user input
INSERT INTO Users VALUES '$Username', '$Password', '$Name', '$Email';

--Log in. Requires user input
SELECT * FROM Users WHERE Username = '$Username' AND Pass = '$Password';

--Account Page. No user input. Username taken from stored credentials during login
SELECT satID FROM Favorites WHERE Username = '$Username';

--Object Database page. No user input
--Satellite
SELECT * FROM Satellite;
--Rocket
SELECT * FROM Rocket;
--Owner
SELECT * FROM Owner;

--Object Page. User input based on what object they clicked to get here
SELECT S.satID, orbitalPeriod, daysInOrbit,
    R.launchID, launchDate, launchSite,
    O.ownerID, ownerType, Location, Country,
    P.satID, purpose1, purpose2, purpose3
FROM Satellite S, Rocket R, Owner O, Purpose P
WHERE S.satID='$userInput' AND S.satID=P.satID AND S.ownerID=O.ownerID AND S.launchID=R.launchID;

--Add object. Requires user input
--If owner already exists, then we won't need to insert a new owner. Check this with php.
INSERT INTO Satellite VALUES '$satID', '$ownerID', '$launchID', '$orbitalPeriod';
INSERT INTO Purpose VALUES '$satID', '$purpose1', '$purpose2', '$purpose3';
INSERT INTO Owner VALUES '$ownerID', '$ownerType', '$Location', '$Country';
```

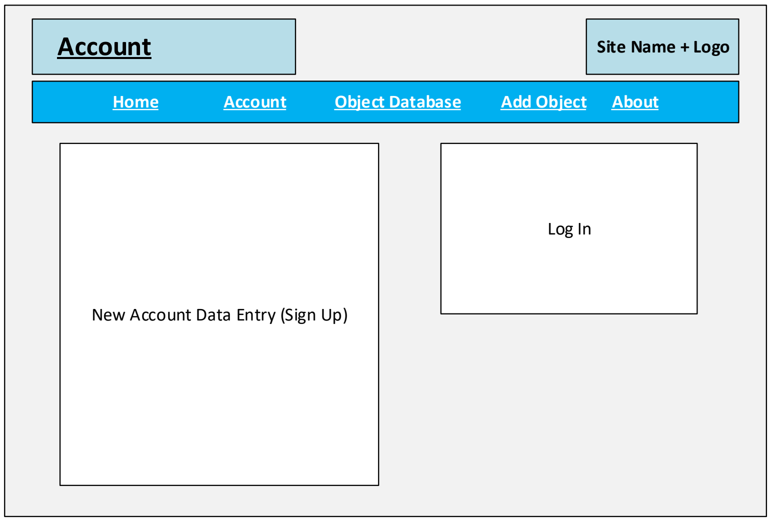
IV. Website Design

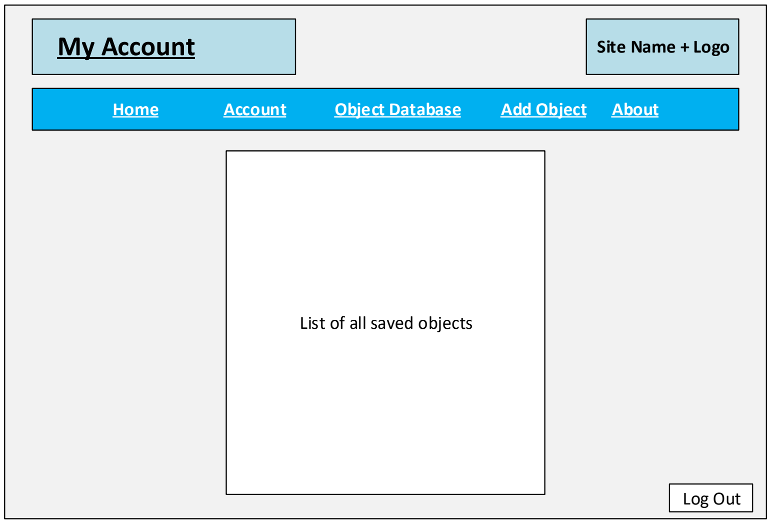
Organizational view of web application:

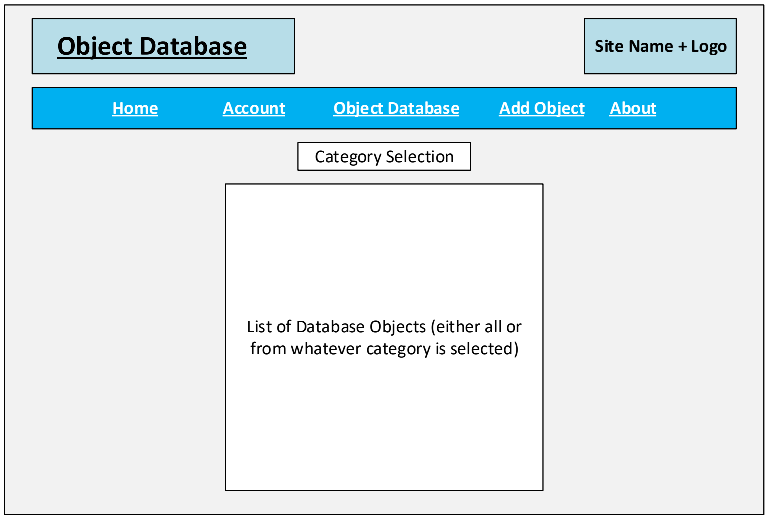


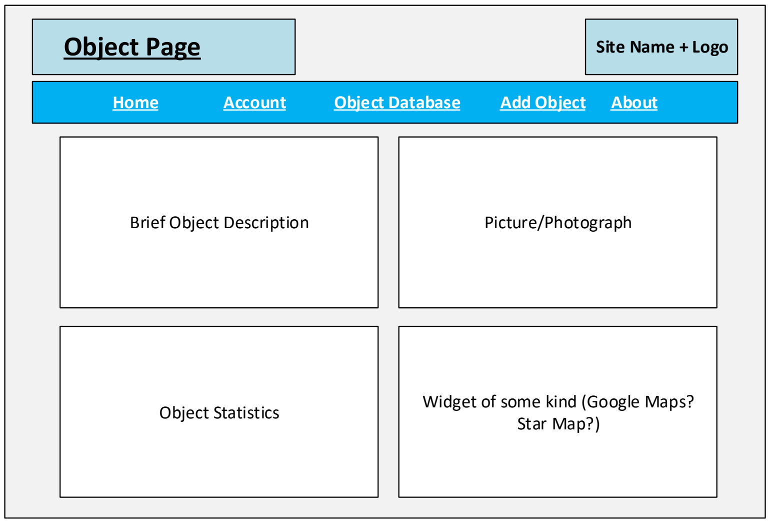
Website mocks:

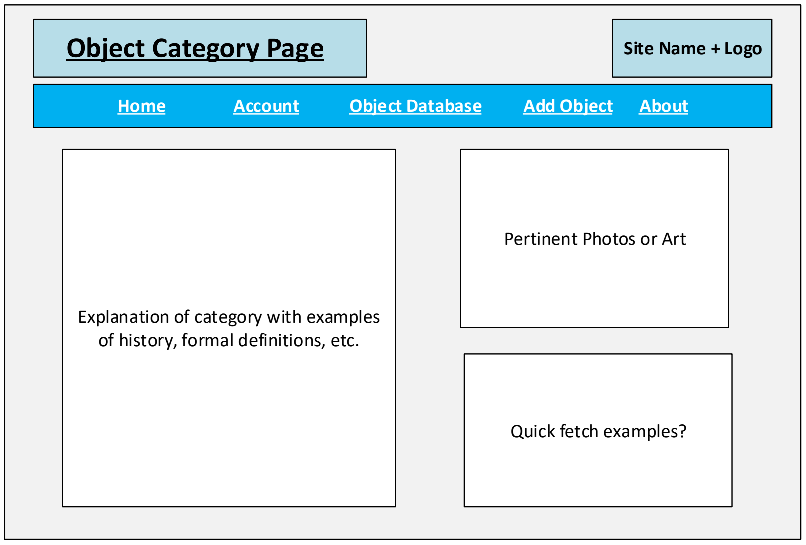


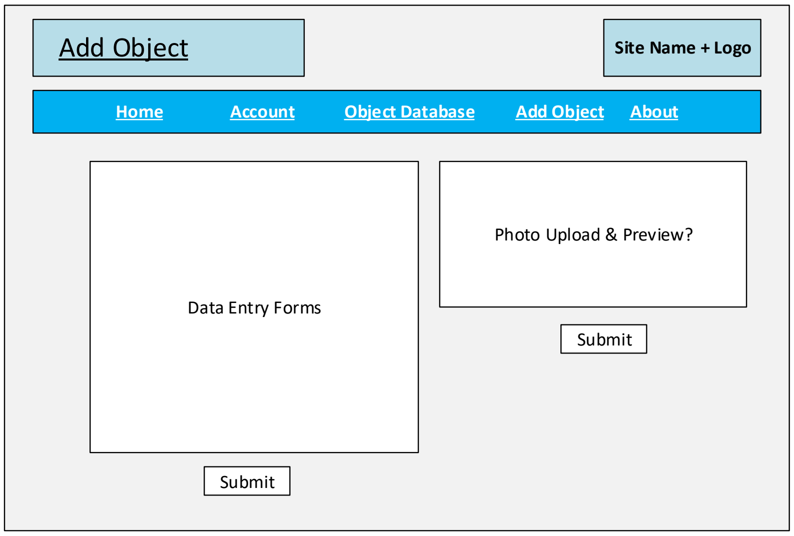





Object Database
Site Name + Logo
Home
Account
Object Database
Add Object
About
Category Selection
List of Database Objects (either all or from whatever category is selected)


Object Page
Site Name + Logo
Home
Account
Object Database
Add Object
About
Brief Object Description
Picture/Photograph
Object Statistics
Widget of some kind (Google Maps? Star Map?)


Object Category Page
Site Name + Logo
Home
Account
Object Database
Add Object
About
Explanation of category with examples of history, formal definitions, etc.
Pertinent Photos or Art
Quick fetch examples?



V. Application Implementation

Our web application uses standard HTML/CSS, but we're going to be using JavaScript to dynamically implement photos and potentially descriptive blurbs as well. We have a blueprint for this kind of dynamic page generation from CS 290, so it should be interesting to implement the same methodology on a more complicated scale and with PHP and SQL built into the application. We will be using standard PHP and SQL to do everything from salting user passwords to fetching object data for display.

Our SQL code (or at least the general syntax and style of it) can be viewed in the screenshots above. There is a significant amount of modification and deletion cascading within our database. Some examples: satellites need launches and vice versa, user favorites need to be capable of updates or deletion based on user actions, the purpose table needs to be updated every time a satellite is added since it functions as a multivariable attribute, etc. Our triggers and procedures are designed to automate the task of cross-relation updates, and should ensure correctness and cohesion across the entire database with minimal external involvement.