5 SQL Commands

We're going to have an account system. The accounts have subtypes (student and owner), but share the username and password type.

```
INSERT INTO Account VALUES ('$username', '$password')
```

"Username", "password" arephp variables. Username and password will be user created, but the sid (student id) should be autocreated. Inserts into the Account table

```
SELECT restaurant_name FROM Restaurant
```

Selects all restaurant names. The names should not be unique, so we should expect duplicates.

```
UPDATE Review SET description = '$content' WHERE description =
'$old_content'
```

Similar to the provided examples, I want the user to be able to edit reviews. This allows them to do so.

```
SELECT review_id, description FROM Reviews WHERE sid = '$provided';
```

This grabs all reviews from a specific student. This way the students can be moderated, we may add a delete feature in the near future or have "rated" reviewers.

```
SELECT MAX(rating) as `Top Rating`, restaurant_name FROM Restaurant R,
Review R2 Where R2.restaurant_ID = R.restaurant_ID GROUP BY
R.restaurant_ID
```

The following command should get the max rated review from each restaurant and group them by restaurant_id. Note that if multiple reviews have given the top score, it will simply grab one and not choose between them.

Triggers/Functions

```
CREATE TRIGGER CheckRating BEFORE INSERT ON Review
BEGIN
      IF new.rating < 0 OR new.rating > 5 THEN
            SIGNAL SQLSTATE "45000"
            SET MESSAGE_TEXT = "Rating must be above 0 and less than
5";
      END IF;
END
```

Basically we want to make sure that all review scores are between 0 and 5. Because they can edit their review scores, it may be best to create a trigger that has the same condition. Alternatively, we could restrict options via some html/php, but this is just some basic error-checking.

```
CREATE TRIGGER CheckRatingUpdate BEFORE UPDATE ON Review
BEGIN
      IF new.rating < 0 OR new.rating > 5 THEN
            SIGNAL SQLSTATE “45000”
            SET MESSAGE_TEXT = “Rating must be above 0 and less than
5”;
      END IF;
END
```

Here’s the other version of the trigger on update above.

This function will rank restaurants based on their reviews. This way people can see the general consensus without going through too many reviews. It has a case where there are none, so fresh restaurants should show up with a “No Reviews” marker.

```
DELIMITER $$
CREATE DEFINER=`cs340_huangmic`@`%` FUNCTION `restaurantRank`(`x`
VARCHAR(6)) RETURNS varchar(10) CHARSET utf8
    NO SQL
BEGIN
      declare average decimal(3,2);

    SELECT AVG(rating) INTO average FROM Review R, Restaurant R1 WHERE
    R1.restaurant_ID = R.restaurant_ID AND X = R1.restaurant.ID GROUP
BY R1.restaurant_ID;

    IF average IS NULL THEN
      RETURN 'No Reviews';
    END IF;

    IF average > 4.0 THEN
      RETURN "GREAT";
    END IF;

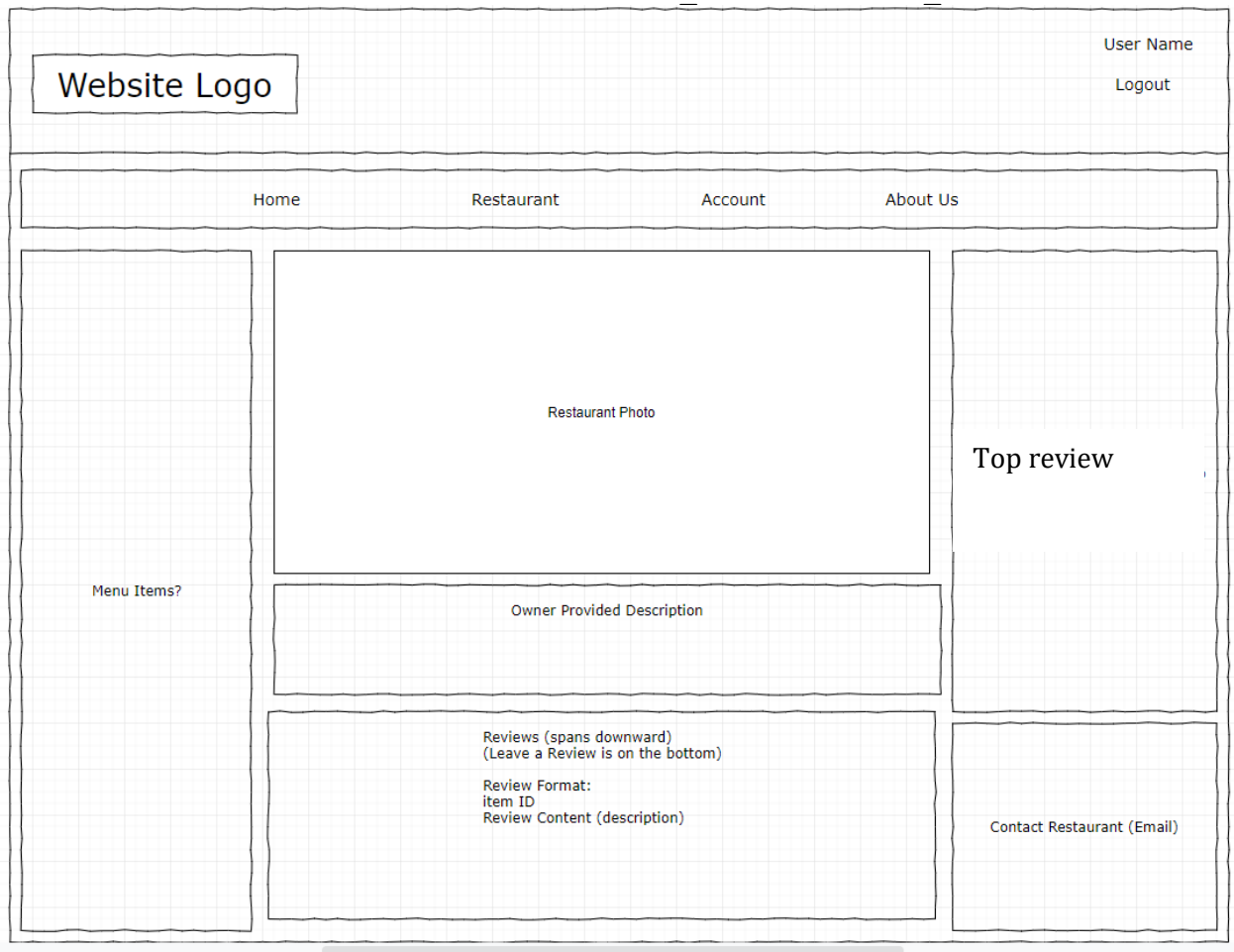
    IF average > 2.5 THEN
      RETURN "GOOD";
    END IF;
    RETURN 'BAD';

END$$
DELIMITER ;
```

Pages

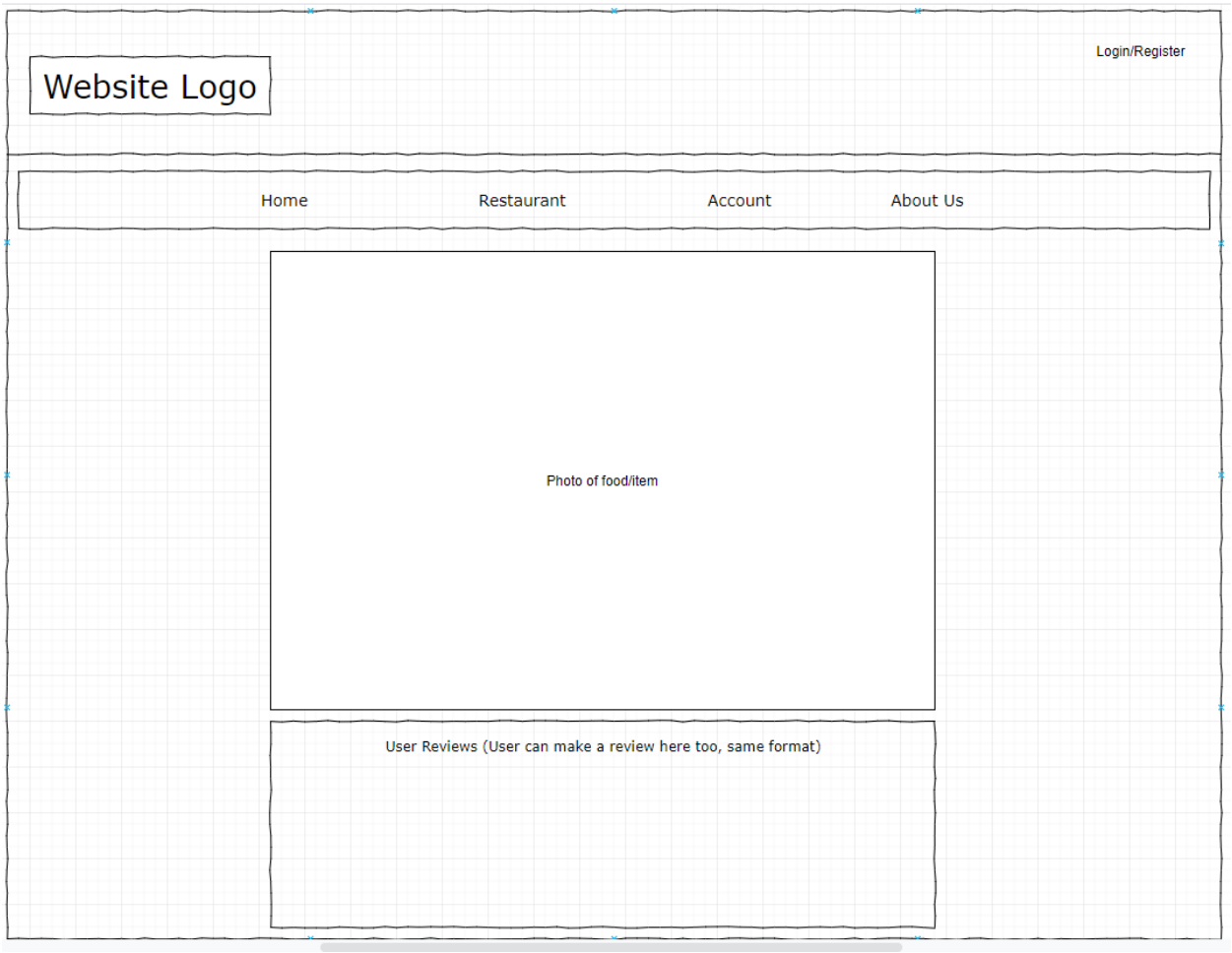
I just want to cover one thing overall with each page. Once login occurs, we want to store the related student name from that username.

```
Select sname FROM Student where Account_ID = $account_id
```

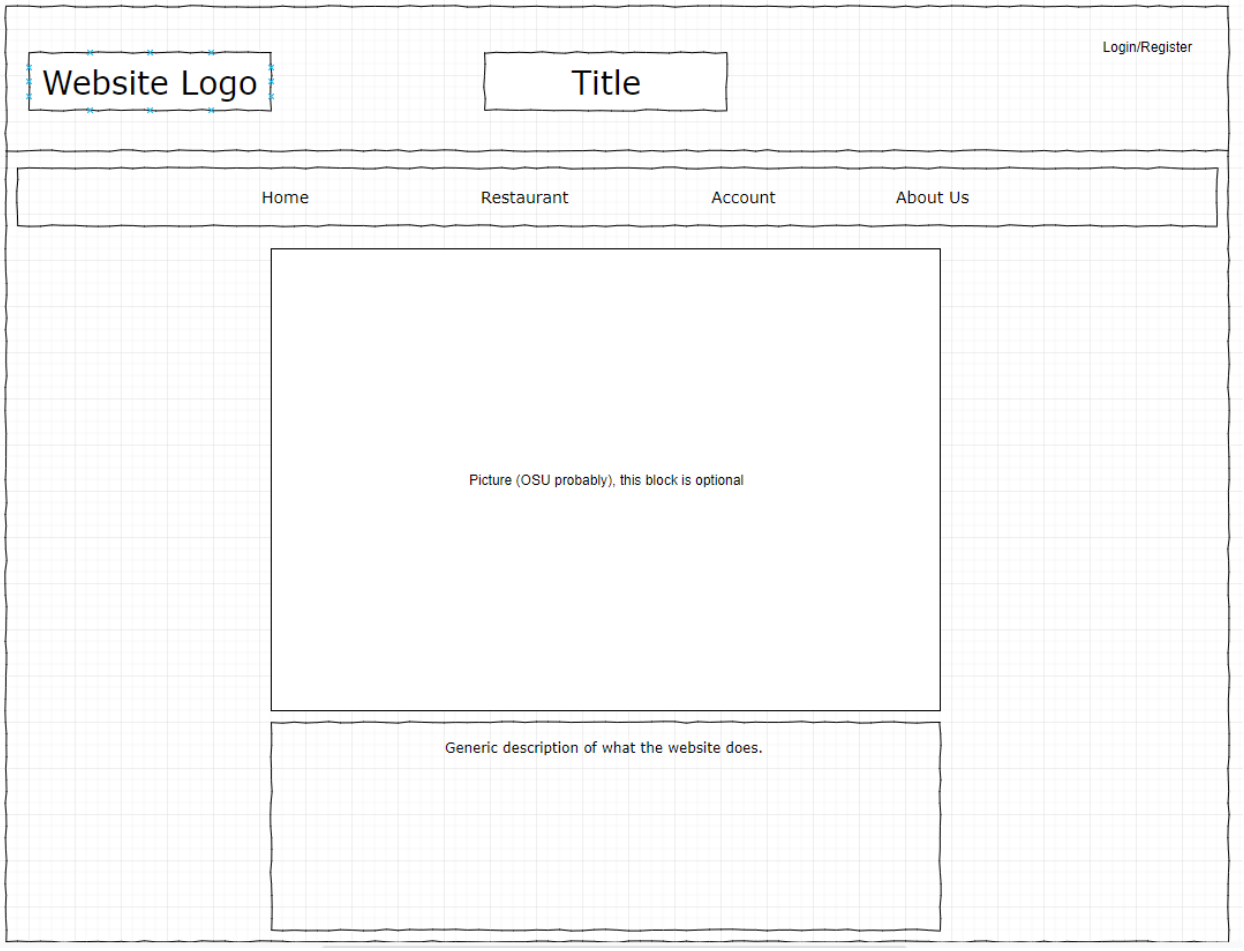


- Menu items and reviews should be a simple query.
    - `Select item_name from Restaurant Natural Join Item Where restaurant_id = $specficpage`
    - ```
      Select rating, description, sname from Restaurant
                                        NATURAL JOIN Review
                                        NATURAL JOIN Student_Data
      Where restaurant_id = $specificpage
      ```
    - Our where clause should be focused on the page we're on.
- We have not added an attribute for the contact info, however it'd just be a simple query. More than likely, the final draft will NOT include that block.
- However, we do intend default to have the max review, we can reference the select statement below for this.
      ```

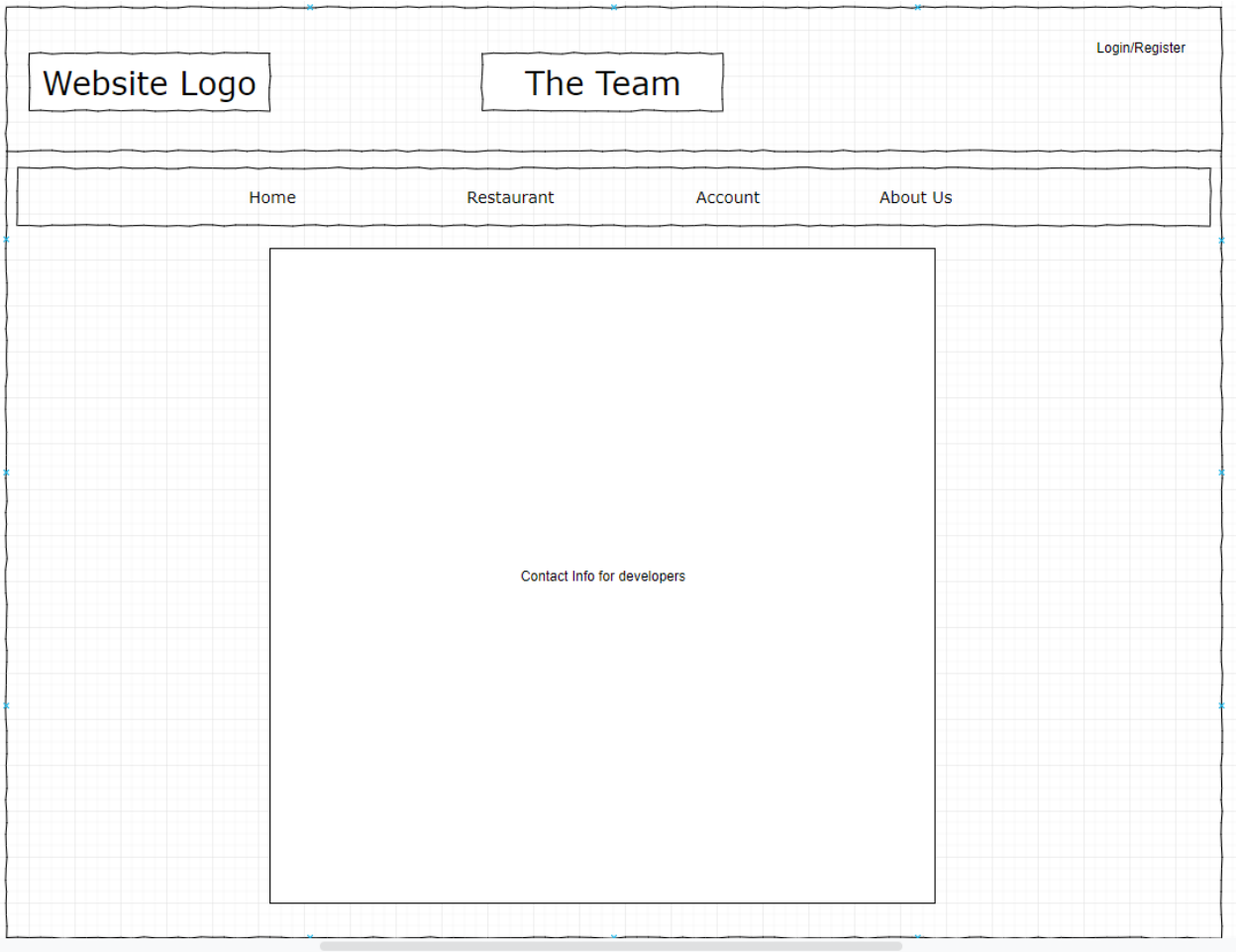
- `SELECT MAX(rating) as `Top Rating`, restaurant_name FROM Restaurant R, Review R2 Where R2.restaurant_ID = R.restaurant_ID GROUP BY R.restaurant_ID`
    - Gives us the max reviews for all pages. Like before, we can divide this to a specific page OR we can store all the results and play with them beforehand.



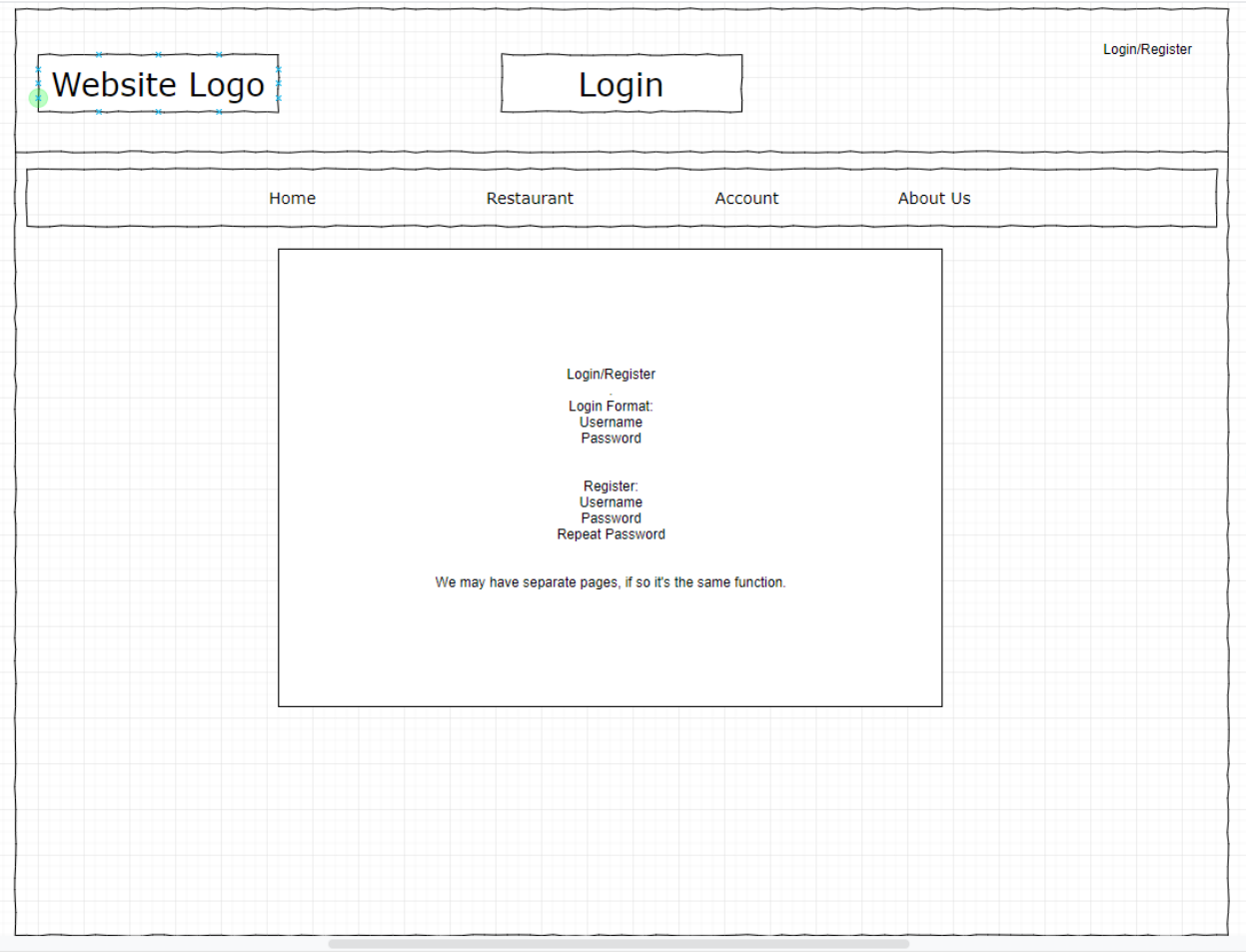
- This is the item page.
    - We are optionally supporting a picture, but if we were to do that, we'd need to setup a php variable that holds a query result. In this case, just getting the related ID is fine.
    - `Select item_id FROM Item where item_id = $id`
    - We could also try uploading a photo to the db, although I'm not sure if myphpadmin supports that. We could alternatively try using html mixups to find a way to reference the picture using the result we receive from the select statement.
    - The list of item reviews
        - `Select sname, item_name, rating FROM Review NATURAL JOIN Item_ID NATURAL JOIN Student`
    - `Quick reminder that you might have the "student" db from an earlier assignment, delete that or rename it.`


Website Logo
Title
Login/Register
Home
Restaurant
Account
About Us
Picture (OSU probably), this block is optional
Generic description of what the website does.

No SQL


Website Logo
The Team
Login/Register
Home
Restaurant
Account
About Us
Contact Info for developers

No SQL



This is the Register portion.

```
INSERT INTO Account VALUES ('$username','$password')
```

This is the login portion

```
Select username, password FROM Account where username =
$username AND password = $password
```

If a result is returned, then we are good. Otherwise the login should fail. The result returned is either the username OR the account_ID, we haven't quite decided yet.