SkiOregon!

URL: http://flip2.engr.oregonstate.edu:8073/

CS340 Project Group 2
Project Step 6 (Portfolio Assignment)

Chris Kesting
Michael Chen

SkiOregon! Executive Summary

After Step 1 we began to better understand the purpose of the ERD (Entity Relationship Diagram). Attributes were removed from the ERD as they should only be included in the Schema. Many attributes were renamed to improve naming consistency. The M:M relationship between Customers/Passes was moved to Transactions/Passes. Many changes also occurred as we honed our understanding of data types and constraints. Costs were changed from Float to Decimal(19,2).

Changes for Step 2 were mainly associated with ERD structure and attribute naming as we continued to gain knowledge around these functions. The major change was the realization that our Resorts_has_Transactions Intersection Table was duplicative to the Passes Entity, so we eliminated Resorts_has_Transactions and replaced it with Passes. The sum_spend attribute in the Customers and Transactions entities were removed as this data could be obtained in a report query. We added ON DELETE CASCADE in appropriate locations in the DDL (Data Definition Language) queries. Attribute names were revised again to improve consistency. For example we removed all the prefixes on the primary key “id” attribute. We did not have any normalization steps based on the provided feedback.

Step 3 introduced the HTML for the basic App pages. We continued changing some attribute constraints like the FK (Foreign Key) on Resorts_name in the Passes entity became Nullable to fulfill a Nullable relationship criteria in the M:M relationship. This means that a Resort can be deleted, but the Transactions can still have a Pass record of it.

Big changes occurred on Step 4 where the game-changer was converting the front end to React and hosting the backend API on Express. We felt that this allowed a cleaner look and more intuitive coding while also using what is probably the most common front-end development tool. We finally gave in to all the advice to add an “id” INT as the PK for Resorts in lieu of the “name” attribute we used previously. While “name” should be unique, the id attribute with AUTO-INCREMENT resolved several issues where users would need to input the “name” attribute and could potentially change it. Several changes to the UI and aesthetics were made on suggestions as well as implementing the CRUD (Create, Read, Update and Delete) operations.

Step 5 involved just a few minor improvements like aliasing some attribute names to make them more readable and usable. We added some minor functionality like a “use 1” button for available passes in the Passes Table. We substantially completed the remaining CRUD operations required for the project.

At Step 6 we finished out the last remaining CRUD requirements for the project. The overall trend as the project progressed was actually to reduce features, attributes, and scope we originally imagined at the start of the project. We learned that the functions of a relational database already provided several of those features and attributes. There are still many improvements to the project which can be implemented such as data validation, but that has been left as an exercise beyond the scope of this class project.

Project Outline

Project Title
SkiOregon!: A one-stop shop for single-day lift tickets at any Oregon Ski Resort.

Team Members
Christian Kesting and Michael Chen

URL
http://flip2.engr.oregonstate.edu:8073/

Overview
SkiOregon! (SO!) partners with all 15 ski resorts in the great state of Oregon. The SO! website is a backend operated by the resorts which enables skiers and boarders to find information about and purchase day passes at any Oregon ski resort. Once a decision to purchase a pass has been made, the website can create a transaction for a specific customer. Passes can be added to the transaction. Each resort has its own day pass cost.

On the back end we collect all customer and transaction information in the database. The database also tracks passes held and used by customers. By providing more extensive Oregon resorts selection than most other websites, we hope to attract customers and generate a healthy sales volume. First year goal is 5,000 sales with a 10% increase per year going forward.

Database Outline

● Symbol represents an Entity
○ Symbol represents an Attribute to the entity listed above it
■ Symbol represents a Relationship

● **Resorts:** Contains all the Oregon ski resorts. Each row represents one Oregon ski resort.
- ○ id: INT, PK, AUTO_INCREMENT
- ○ name: VARCHAR(145), NOT NULL
- ○ vert: INT
- ○ avg_snowpack: INT
- ○ rate: DECIMAL(19,2), NOT NULL

■ 1:M relationship between Resorts and Runs. Resorts_id is a FK in Runs.
■ M:M relationship between Resorts and Transactions. Passes is the intersection table between Resorts and Transactions and includes the Transactions_id and Resorts_id as FK's.
■ 1:M relationship between Resorts and Passes as a product of the above M:M relationship. Resorts_name is a FK in Passes.

● **Customers:** Contains information about individual customers who book passes at Oregon ski resorts.
- ○ *id: INT, PK, Auto Increment, NOT NULL
- ○ fname: VARCHAR(45), NOT NULL
- ○ lname: VARCHAR(45), NOT NULL
- ○ age: INT, NOT NULL
- ○ phone: VARCHAR(45), NOT NULL
- ○ email: VARCHAR(145), NOT NULL

■ 1:M relationship between Customers and Transactions. Customer_id is FK in Transactions.

● **Transactions:** Contains customer orders for day passes at Oregon ski resorts. A single transaction may include multiple day passes from multiple resorts.
- ○ *id: INT, PK, auto_increment, NOT NULL
- ○ Customers_id: INT, FK
- ○ date: DATE, NOT NULL

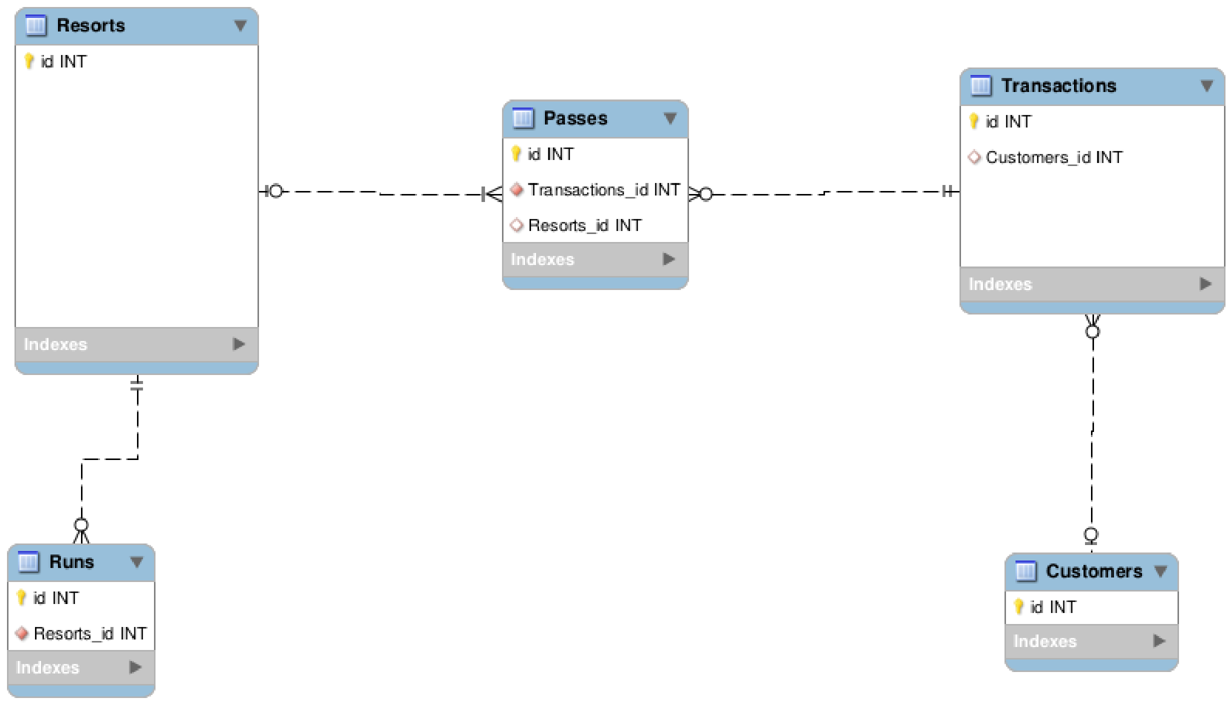
■ M:M relationship between Resorts and Transactions. See description of intersection table under Resorts.
■ 1:M relationship between Passes and Transactions as a result of the above M:M relationship. Transactions_trans_id is FK in Passes.
■ 1:M relationship between Customers and Transactions. Customer_id is FK in Transactions. A transaction may exist after a customer has been deleted; the relationship is 'nullable'.

● **Passes:** Intersection table between Resorts and Transactions. Also contains all the day passes currently held by customers at any resort and tracks their usage. Rows are populated after a transaction is finalized.
- ○ *id: INT, PK, auto_increment, NOT NULL

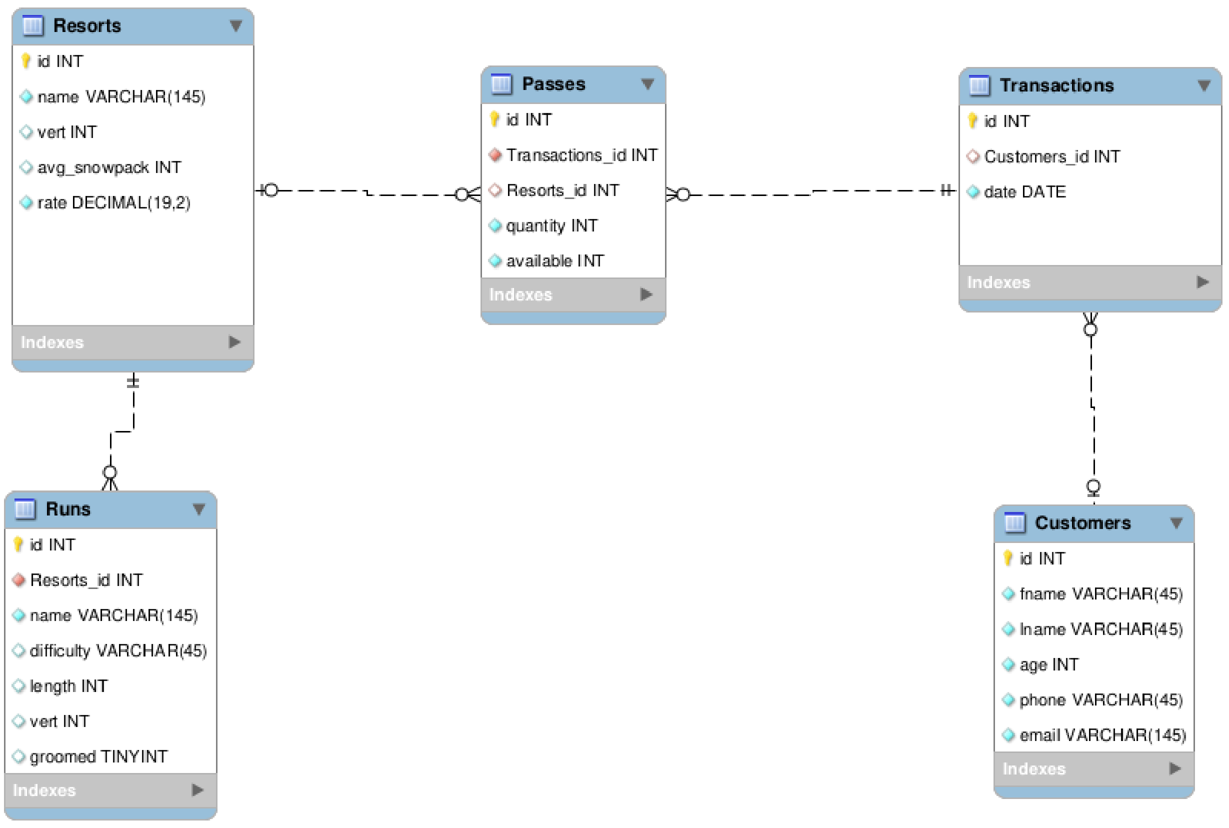
    - Transactions_id: INT, FK, NOT NULL
    - Resorts_id: INT, FK,
    - quantity: INT, NOT NULL
    - available: INT, NOT NULL
- ■ M:1 relationship between Passes and Resorts
- ■ M:1 relationship between Passes and Transactions

- **Runs:** Contains all the ski runs and salient information.
    - *id: INT, PK, NOT NULL, auto_increment
    - Resorts_id: INT, FK, NOT NULL
    - name: VARCHAR(145), NOT NULL
    - difficulty: VARCHAR(45)
    - length: INT
    - vert: INT
    - groomed: TINYINT (Bool)
- ■ M:1 relationship between Runs and Resorts

ERD


Resorts
id INT
Indexes
Passes
id INT
Transactions_id INT
Resorts_id INT
Indexes
Transactions
id INT
Customers_id INT
Indexes
Runs
id INT
Resorts_id INT
Indexes
Customers
id INT
Indexes

Schema


Resorts
id INT
name VARCHAR(145)
vert INT
avg_snowpack INT
rate DECIMAL(19,2)
Indexes
Passes
id INT
Transactions_id INT
Resorts_id INT
quantity INT
available INT
Indexes
Transactions
id INT
Customers_id INT
date DATE
Indexes
Runs
id INT
Resorts_id INT
name VARCHAR(145)
difficulty VARCHAR(45)
length INT
vert INT
groomed TINYINT
Indexes
Customers
id INT
fname VARCHAR(45)
lname VARCHAR(45)
age INT
phone VARCHAR(45)
email VARCHAR(145)
Indexes

Example Data:

RESORTS				
\`id\`	name\`	vert	avg_snowpack	rate
1	“Timberline”	3691	300	135.00
2	“Mt. Bachelor”	3366	280	195.00
3	“HooDoo”	1017	210	79.00
4	“Mt. Ashland”	1161	180	69.00

RUNS						
id	Resorts_name	name	difficulty	length	vert	groomed
1	“Timberline”	“Magic Mile”	“Intermediate”	5280	NULL	1
2	“Mt. Bachelor”	“The Cirque”	“Expert”	2100	880	0
3	“HooDoo”	“Grandstand”	“Advanced”	1850	560	0
4	“Mt. Ashland”	“Romeo”	“Intermediate”	3200	680	1

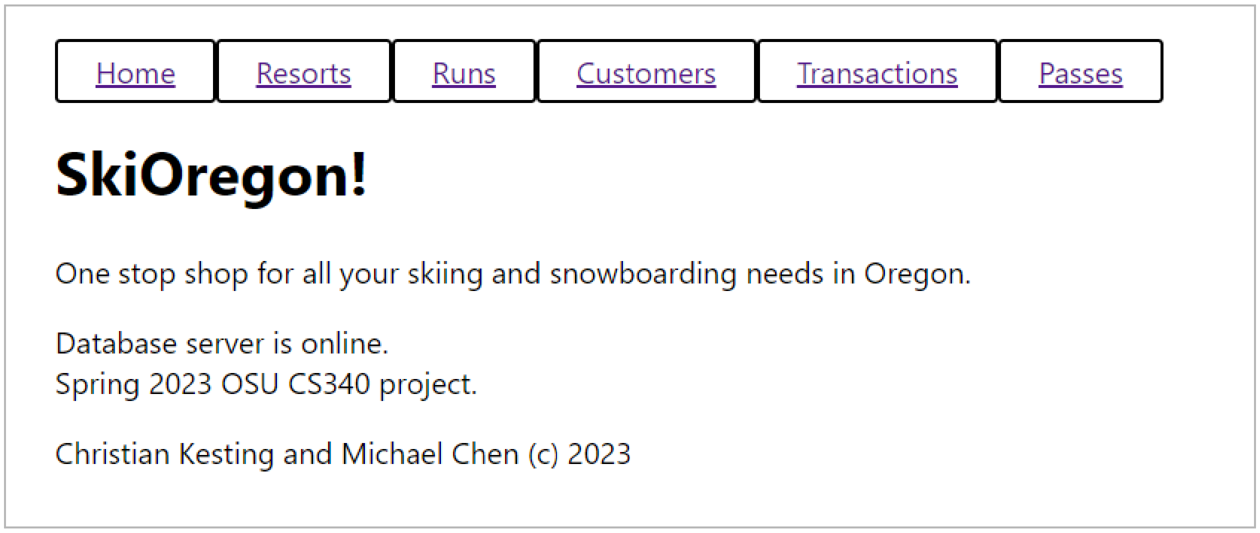
CUSTOMERS					
id	fname	lname	age	phone	email
1	"Chris"	"K"	40	"555-123-4567"	"chrisk@chris.com"
2	"Mike"	"C"	30	"555-123-4568"	"mike@mike.com"
3	"Three"	"Cust"	3	"555-123-3333"	"three@3.com"
4	"Four"	"Cust"	4	"555-123-4444"	"four@3.com"

TRANSACTIONS		
id	Customers_id	date
1	1	2022-11-02
2	1	2023-01-07
3	2	2023-02-12
4	3	2023-04-07

PASSES				
id	Resorts_name	Transactions_id	Quantity	available
1	“Timberline”	2	1	1
2	“HooDoo”	1	2	0
3	“HooDoo”	3	3	1
4	“HooDoo”	4	4	0
5	“Mt. Ashland”	4	5	1

Screenshots

Home Page



SkiOregon!

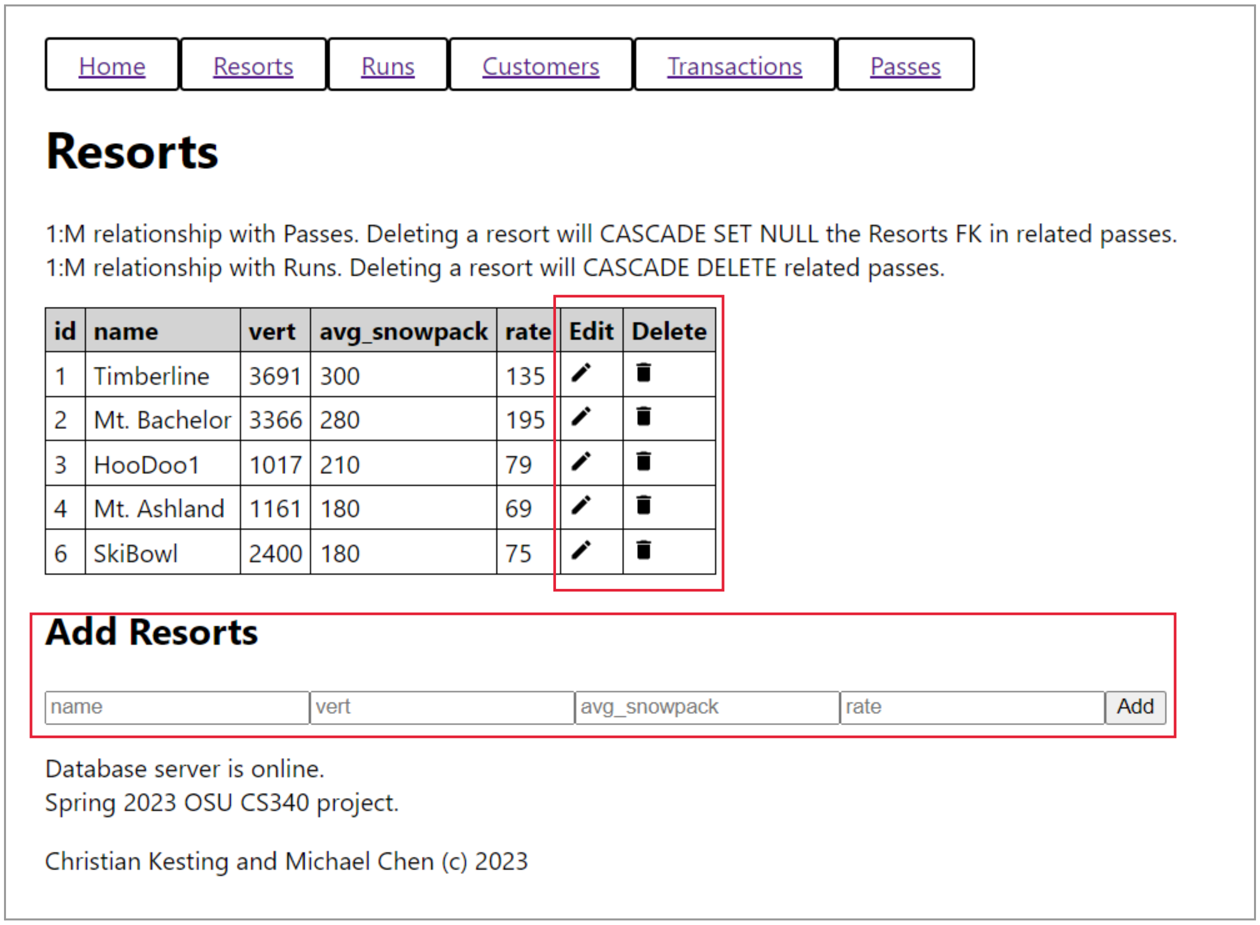
One stop shop for all your skiing and snowboarding needs in Oregon.

Database server is online.
Spring 2023 OSU CS340 project.

Christian Kesting and Michael Chen (c) 2023

No CRUD functions are on the home page. Click on any of the navigational links at the top of the page to access the functions for each entity in the the SkiOregon! database. The webpage displays a database server status message, which informs the user whether the express server

Resorts Page (Create, Read, Update, Delete)



Resorts

1:M relationship with Passes. Deleting a resort will CASCADE SET NULL the Resorts FK in related passes.
1:M relationship with Runs. Deleting a resort will CASCADE DELETE related passes.

id	name	vert	avg_snowpack	rate	Edit	Delete
1	Timberline	3691	300	135	✎	🗑
2	Mt. Bachelor	3366	280	195	✎	🗑
3	HooDoo1	1017	210	79	✎	🗑
4	Mt. Ashland	1161	180	69	✎	🗑
6	SkiBowl	2400	180	75	✎	🗑

Add Resorts

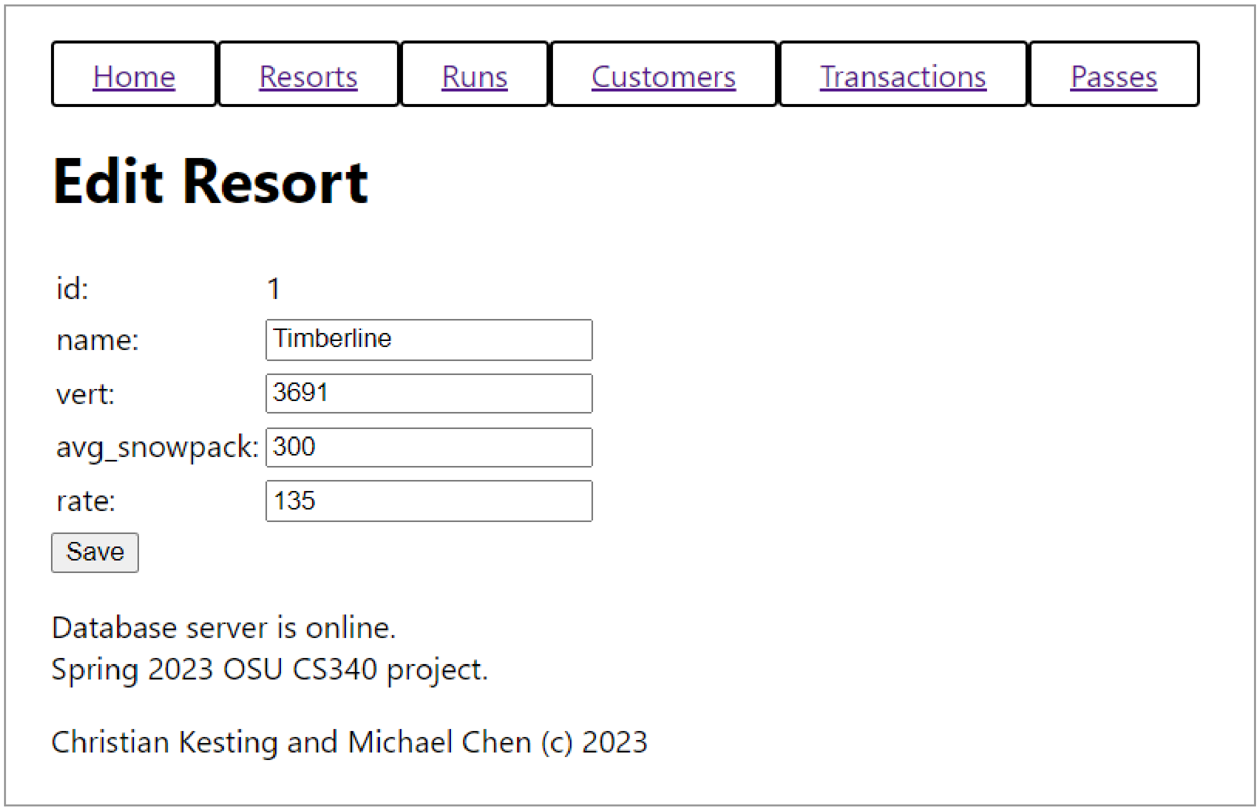
Database server is online.
Spring 2023 OSU CS340 project.

Christian Kesting and Michael Chen (c) 2023

The Resorts entity performs all CRUD functions:

- The “Add Resorts” form below the table enables the user to **create** a row for the Resorts table.
- The page populates by **read**ing from the Resorts table.
- Clicking the pencil icon under the Edit column in the displayed table takes the user to a page where the user can input data to **update** a particular Resort row. A screenshot for this page is shown on the next page.
- Clicking the trash can icon under the Delete column in the displayed table will **delete** a particular Resort row.
  - Deleting a resort will CASCADE DELETE all runs associated with the specific resort via the resort id FK.
  - Deleting a resort will CASCADE SET NULL all passes with resort id FK associated with the specific resort.

Edit Resort Page (Update)



Edit Resort

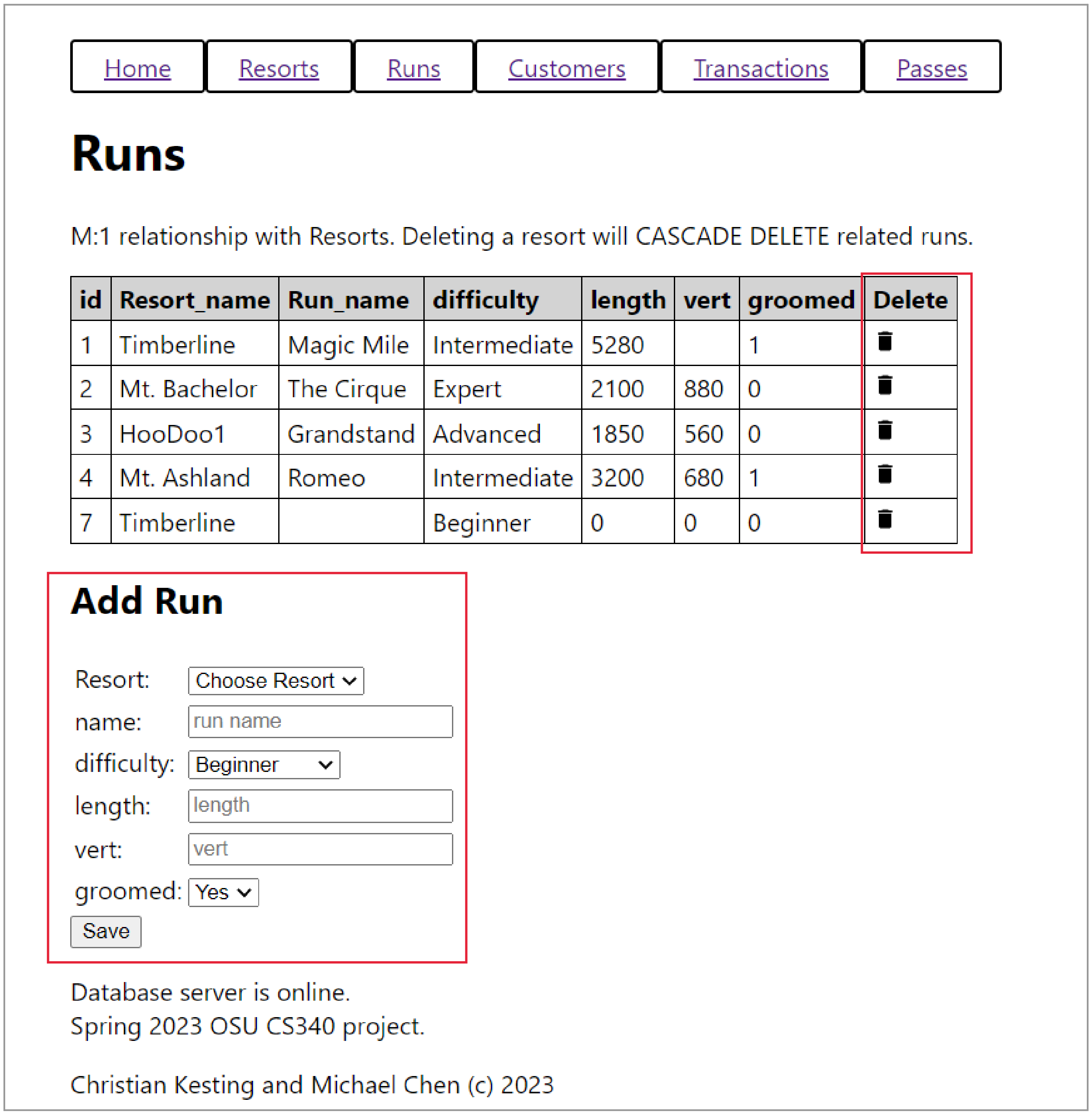
id:	1
name:	Timberline
vert:	3691
avg_snowpack:	300
rate:	135

Database server is online.
Spring 2023 OSU CS340 project.

Christian Kesting and Michael Chen (c) 2023

The Edit Resort page enables the user to **edit** a particular resort identified by the primary key “id”. The “id” is selected by clicking the pencil icon on the Resorts page for a particular resort row displayed in the table. The default entries in the table show the current attribute values before edit.

Runs Page (Create, Read, Delete)



Home | Resorts | Runs | Customers | Transactions | Passes

Runs

M:1 relationship with Resorts. Deleting a resort will CASCADE DELETE related runs.

id	Resort_name	Run_name	difficulty	length	vert	groomed	Delete
1	Timberline	Magic Mile	Intermediate	5280		1	🗑
2	Mt. Bachelor	The Cirque	Expert	2100	880	0	🗑
3	HooDoo1	Grandstand	Advanced	1850	560	0	🗑
4	Mt. Ashland	Romeo	Intermediate	3200	680	1	🗑
7	Timberline		Beginner	0	0	0	🗑

Add Run

Resort:	Choose Resort
name:	run name
difficulty:	Beginner
length:	length
vert:	vert
groomed:	Yes

Save

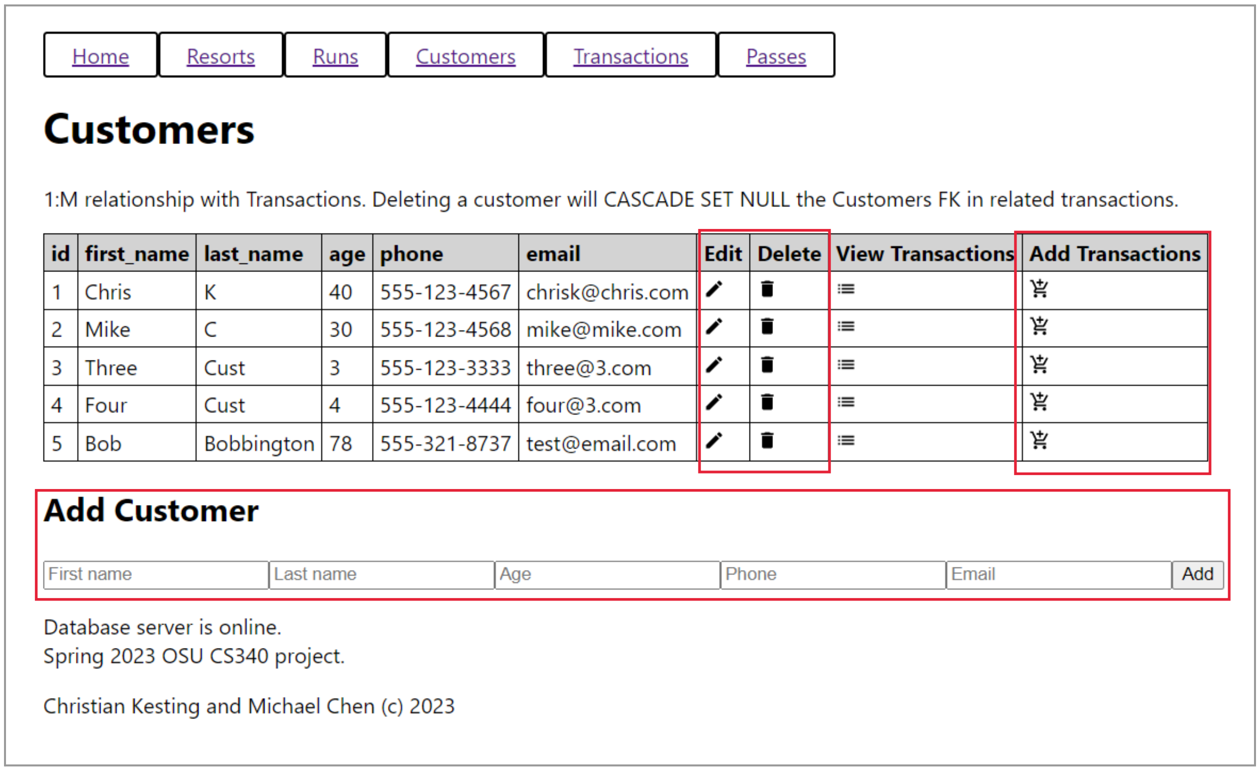
Database server is online.
Spring 2023 OSU CS340 project.

Christian Kesting and Michael Chen (c) 2023

The Runs page performs CRD functions:

- The “Add Run” form at the bottom of the page enables the user to **create** a run to insert into the Runs table. The Resort attribute populates a dropdown by querying the Resorts table in the database. The react webpage associates the resort names with the appropriate resort id behind the scenes.
- The page populates by **read**ing the Runs table.
- The trash can icon under the “Delete” column **delete**s a particular row in the Runs table.

Customers Page (Create, Read, Update, Delete)



Customers

1:M relationship with Transactions. Deleting a customer will CASCADE SET NULL the Customers FK in related transactions.

id	first_name	last_name	age	phone	email	Edit	Delete	View Transactions	Add Transactions
1	Chris	K	40	555-123-4567	chrisk@chris.com	✎	🗑	≡	🛒
2	Mike	C	30	555-123-4568	mike@mike.com	✎	🗑	≡	🛒
3	Three	Cust	3	555-123-3333	three@3.com	✎	🗑	≡	🛒
4	Four	Cust	4	555-123-4444	four@3.com	✎	🗑	≡	🛒
5	Bob	Bobbington	78	555-321-8737	test@email.com	✎	🗑	≡	🛒

Add Customer

First name	Last name	Age	Phone	Email	Add

Database server is online.
Spring 2023 OSU CS340 project.

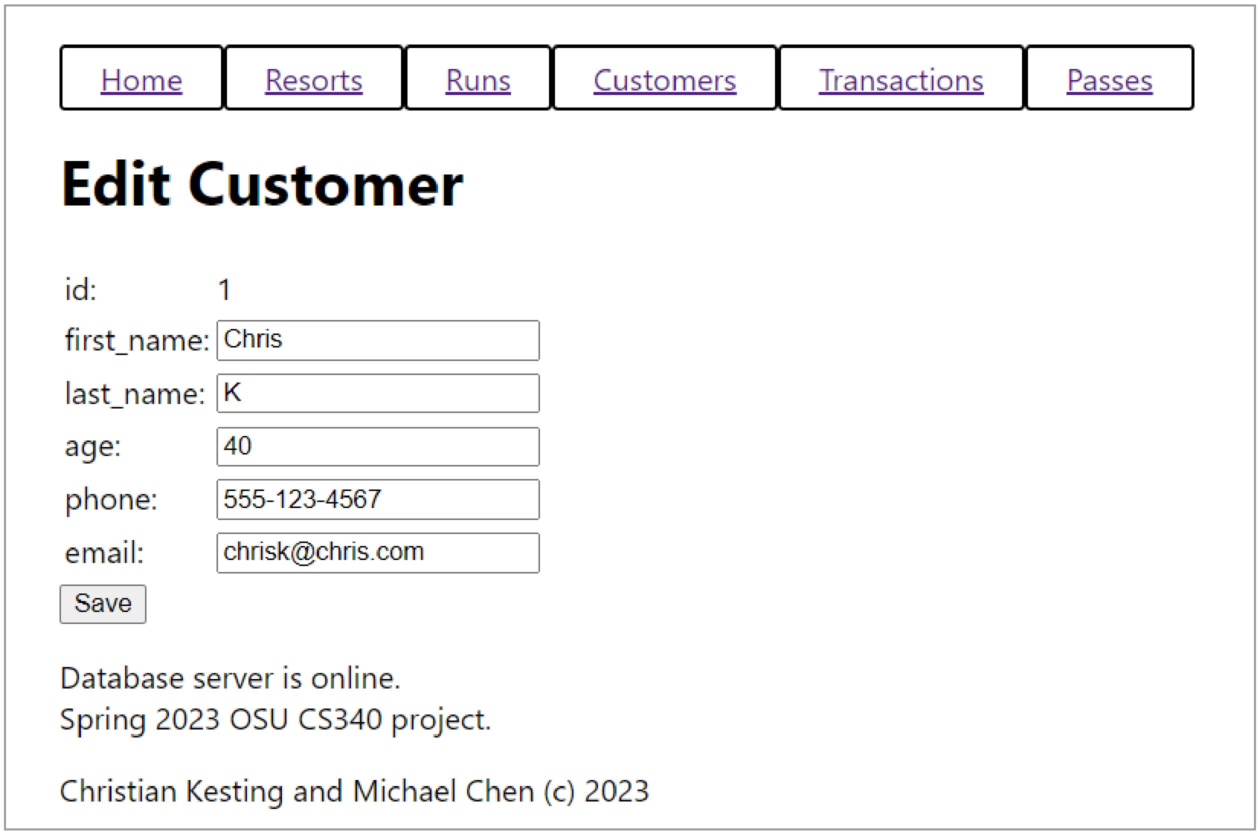
Christian Kesting and Michael Chen (c) 2023

The Customers page performs all CRUD functions and also hosts the Create function for the Transactions table:

- The “Add Customer” form below the displayed table enables the user to **create** a customer for the Customers table.
- The displayed table is populated by **read**ing from the Customers table in the database.
- Clicking the pencil icon under the “Edit” column in the displayed table takes the user to a page where the user can **update** the particular customer row in which the user clicked the pencil. A screenshot of the update page is shown on the next page.
- Clicking the trash can icon under the “Delete” column in the displayed table enables the user to **delete** a particular row in the Customers table.
    - Deleting a customer will cause the transactions associated with the customer to CASCADE SET NULL their Customers id FK to NULL.
- Clicking the shopping cart icon under the “Add Transactions” column takes the user to the Add Transactions page and allows the user to **create** a transaction associated with a particular customer row. The FK in the transaction is determined by which customer row shopping cart icon is clicked.
    - This design was chosen instead of a dropdown in an “Add Transaction” form on the Transactions page because if there are a lot of customers in the database, then the dropdown menu can become inconveniently long. The intuitive workflow is to find a customer first, then to create a transaction for that customer. This design was chosen after discussing with Dr. Michael Curry.

    - Unfortunately we did not complete a functionality to search for a particular customer, which would make the workflow better.
- Clicking the icon under “View Transactions” enables the user to view all transactions associated with the particular customer. This an extra functionality.

Edit Customer Page (Update)



Edit Customer

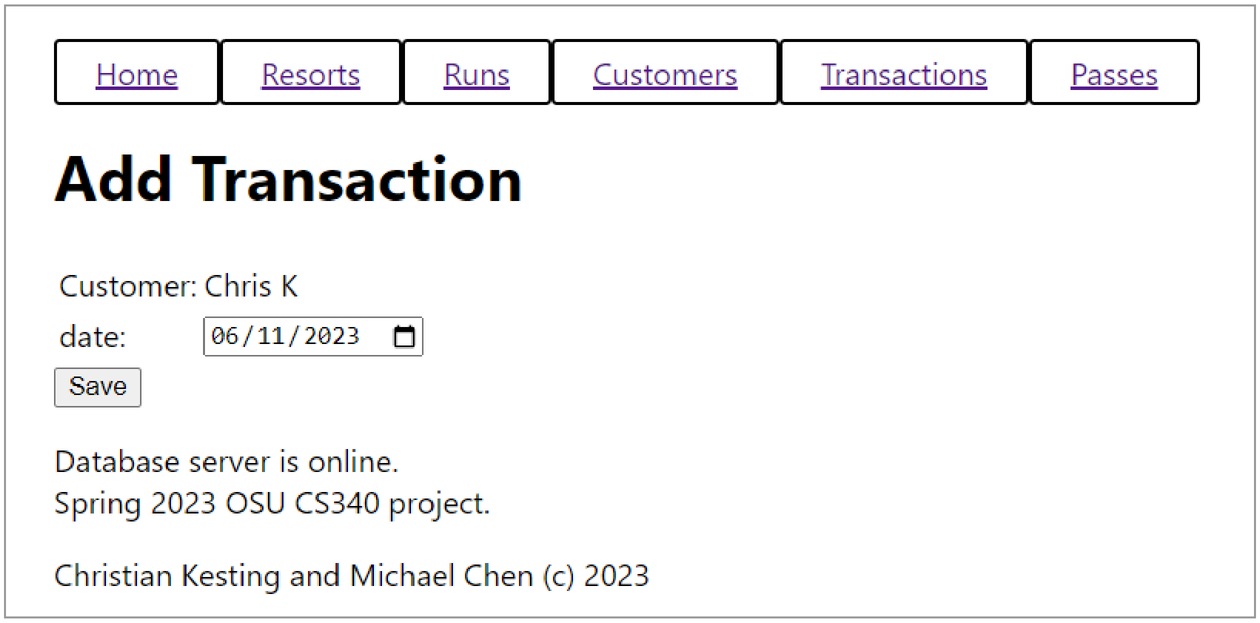
id:	1
first_name:	Chris
last_name:	K
age:	40
phone:	555-123-4567
email:	chrisk@chris.com

Database server is online.
Spring 2023 OSU CS340 project.

Christian Kesting and Michael Chen (c) 2023

The Edit Customer page allows the user to **update** a particular customer identified by the PK "id". The customer is chosen by which pencil icon is clicked on the Customers page. The default values are populated as the existing values for this customer before edit.

Add Transaction Page (Create)



Add Transaction

Customer: Chris K

date: 06/11/2023

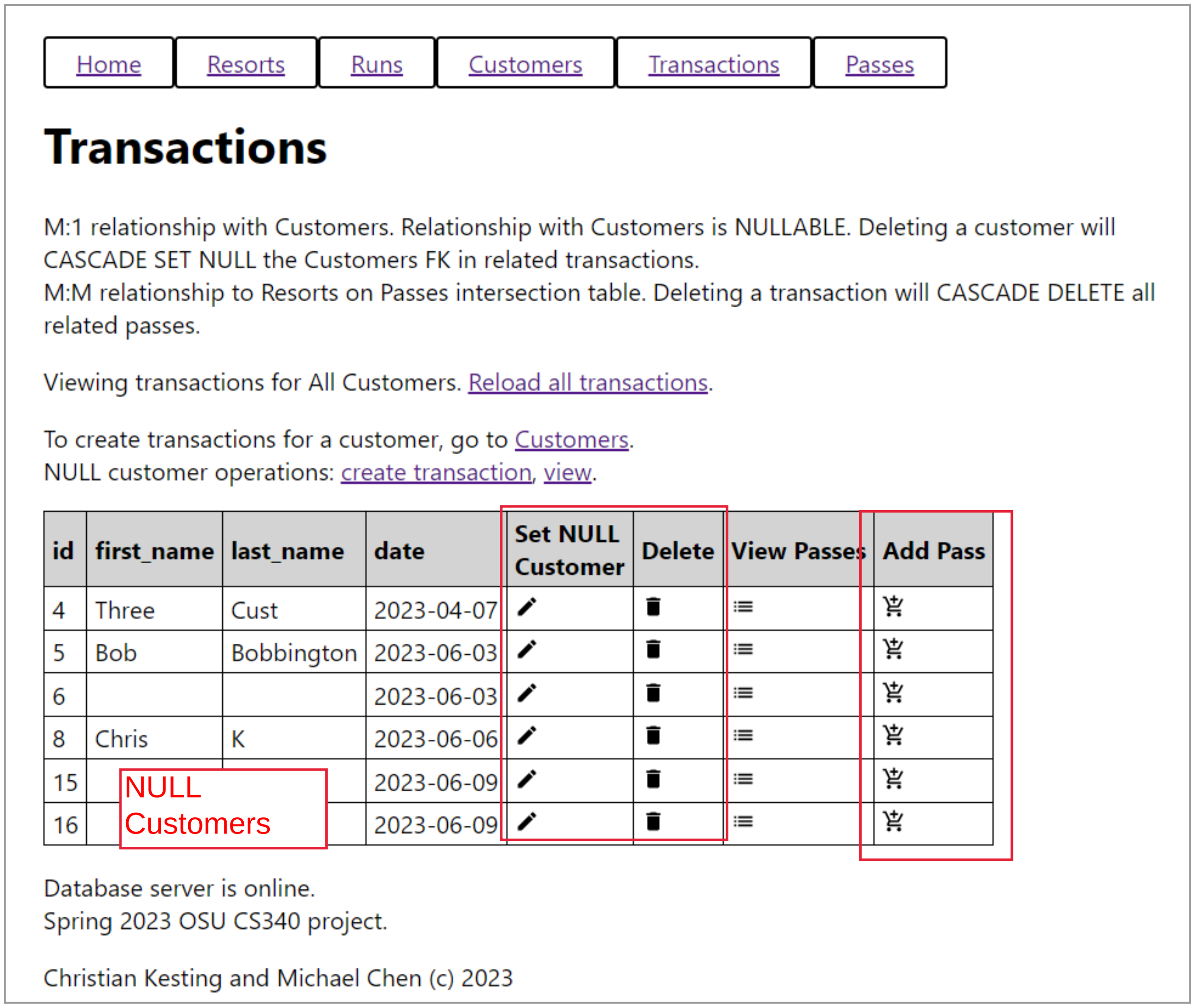
Database server is online.
Spring 2023 OSU CS340 project.

Christian Kesting and Michael Chen (c) 2023

The “Add Transaction” page is accessed from either the Customers page by clicked the shopping cart icon in a customer row, or by clicking the null customer “create transaction” link on the transaction page.

- The former will populate the customer field with the name for the customer row from which the shopping cart icon was clicked.
- The latter will populate the Customer field with NULL representing a null entry for the Customers_id FK for the new transaction.
- The default value in the date field is the current date when you access the webpage.
- After clicking “Save”, a transaction is **created** and the user is taken to the Transactions page where only the transactions associated with the particular customer is displayed.

Transactions Page (Create, Read, Update, Delete, Nullable)



Home | Resorts | Runs | Customers | Transactions | Passes

Transactions

M:1 relationship with Customers. Relationship with Customers is NULLABLE. Deleting a customer will CASCADE SET NULL the Customers FK in related transactions.
M:M relationship to Resorts on Passes intersection table. Deleting a transaction will CASCADE DELETE all related passes.

Viewing transactions for All Customers. Reload all transactions.

To create transactions for a customer, go to Customers.
NULL customer operations: create transaction, view.

id	first_name	last_name	date	Set NULL Customer	Delete	View Passes	Add Pass
4	Three	Cust	2023-04-07	✎	🗑	≡	🛒
5	Bob	Bobbington	2023-06-03	✎	🗑	≡	🛒
6			2023-06-03	✎	🗑	≡	🛒
8	Chris	K	2023-06-06	✎	🗑	≡	🛒
15			2023-06-09	✎	🗑	≡	🛒
16			2023-06-09	✎	🗑	≡	🛒

NULL Customers

Database server is online.
Spring 2023 OSU CS340 project.

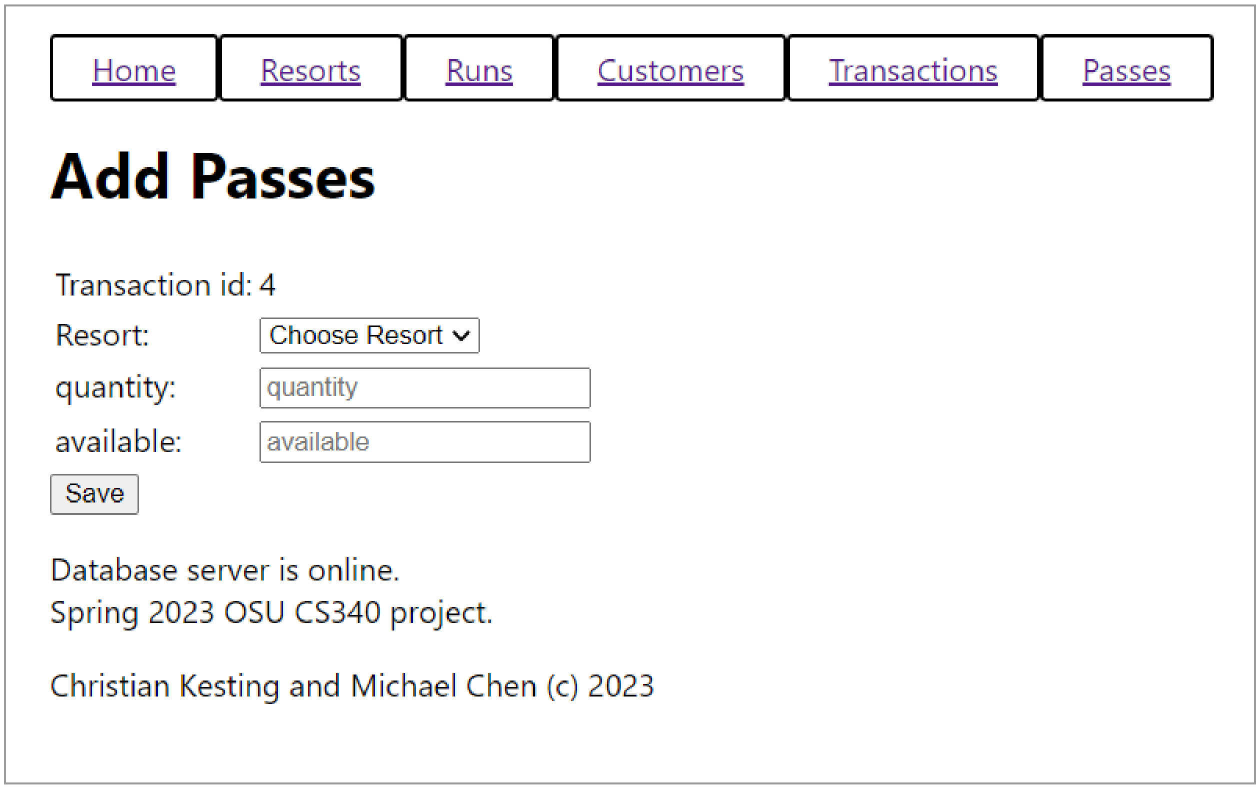
Christian Kesting and Michael Chen (c) 2023

The Transactions page performs or links to CRUD functions. Additionally this page hosts the beginning of the Create functionality for Passes.

- In order to **create** a transaction, the user either clicks on the link to the Customers page or the link to create a transaction for a NULL customer. On the customers page, the user then clicks on the shopping cart icon for a particular customer to create a transaction for that customer. See the Customers page screenshot and the Add Transactions page screenshot for more information.
- The displayed table populates by **read**ing the Transactions table.
- The user may **update** a particular transaction to change the customer to a NULL customer by clicking the pencil icon under the "Set NULL Customer" column in the displayed table. This is included to meet the **Nullable** foreign key and relationship requirement of the project.
- Clicking the trash can icon under the "Delete" column enables the user to **delete** a particular transaction.

    - Deleting a transaction will CASCADE DELETE all passes associated with the specific transaction via the transaction id as a FK.
- Clicking the shopping cart icon under the “Add Pass” column takes the user to the Add Passes page and enables the user to **create** a pass associated with the row in which the shopping cart was clicked.
    - This functionality is similar to the Add Transactions functionality from the Customers Page. The intended workflow is for the user to identify a transaction first then to add passes to that transaction.
- There is additional functionality in the website to display only the transactions associated with a specific customer or only the NULL customer transactions. The user may click the “Reload all transactions” link in order to refresh the page and display all the transactions again.

Add Passes Page (Create)



Add Passes

Transaction id: 4

Resort:	Choose Resort
quantity:	quantity
available:	available

Save

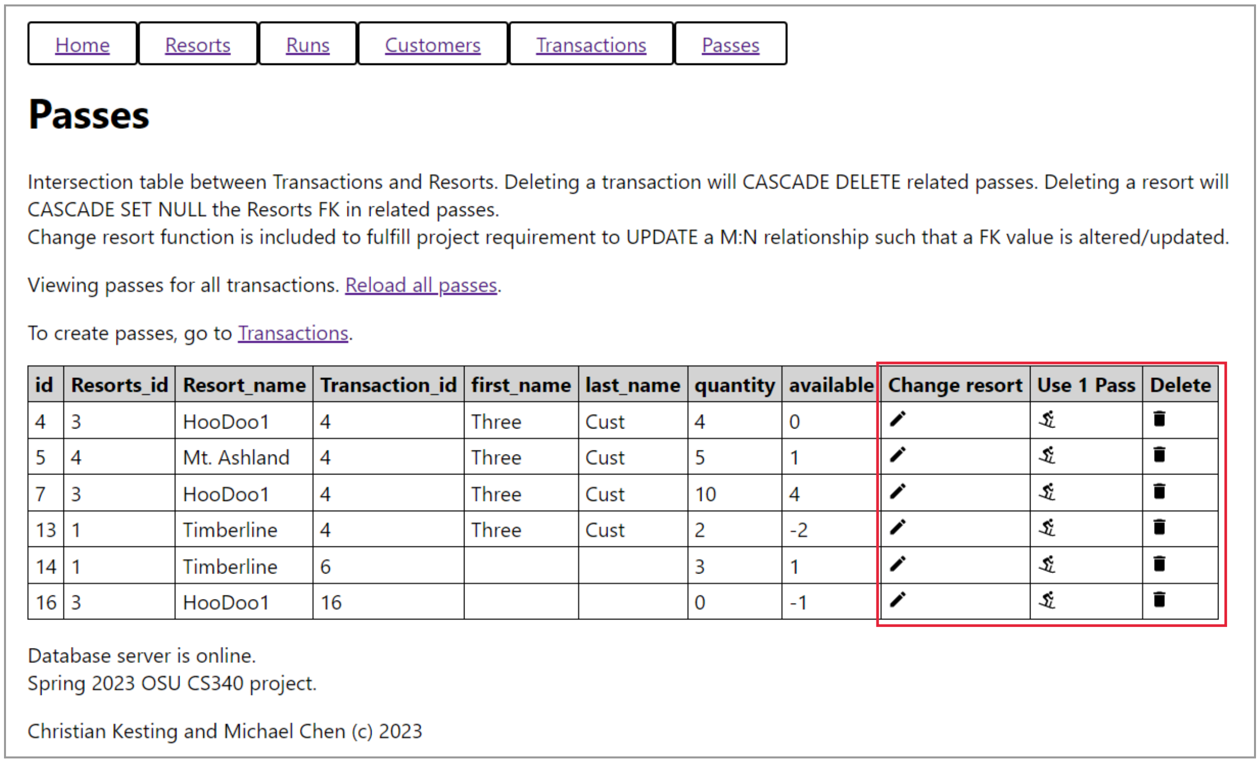
Database server is online.
Spring 2023 OSU CS340 project.

Christian Kesting and Michael Chen (c) 2023

The Add Passes page is accessed from the Transactions page by clicking on the shopping cart icon in a particular transaction row. The Transaction id FK for the new Passes row is taken from the transactions row in which the shopping cart icon was clicked.

- The resort field is a drop down menu of resort names populated via a query to the resorts table. The React framework javascript code associates the Resorts id FK with the selected resort name.
- The quantity field represents the number of passes purchased for the particular transaction and resort. The available field represents how many passes are left. For a new purchase, the available field should be the same as the quantity.
- Clicking “Save” will **create** a pass and take the user to the Passes page, where it displays the specific passes associated with this transaction. Clicking the reload all passes link on that page will refresh and display all passes for all transactions.

Passes Page (Create, Read, Update, Delete, M:N Update)



Home | Resorts | Runs | Customers | Transactions | Passes

Passes

Intersection table between Transactions and Resorts. Deleting a transaction will CASCADE DELETE related passes. Deleting a resort will CASCADE SET NULL the Resorts FK in related passes.
Change resort function is included to fulfill project requirement to UPDATE a M:N relationship such that a FK value is altered/updated.

Viewing passes for all transactions. Reload all passes.

To create passes, go to Transactions.

id	Resorts_id	Resort_name	Transaction_id	first_name	last_name	quantity	available	Change resort	Use 1 Pass	Delete
4	3	HooDoo1	4	Three	Cust	4	0	✎	⛷	🗑
5	4	Mt. Ashland	4	Three	Cust	5	1	✎	⛷	🗑
7	3	HooDoo1	4	Three	Cust	10	4	✎	⛷	🗑
13	1	Timberline	4	Three	Cust	2	-2	✎	⛷	🗑
14	1	Timberline	6			3	1	✎	⛷	🗑
16	3	HooDoo1	16			0	-1	✎	⛷	🗑

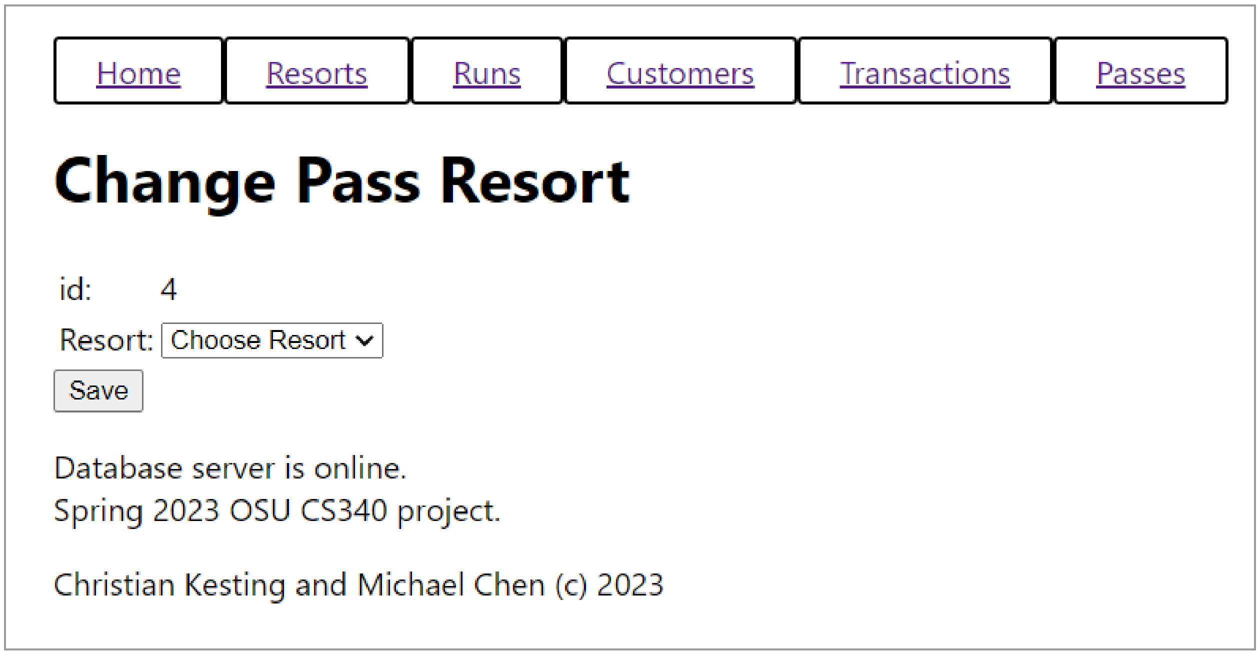
Database server is online.
Spring 2023 OSU CS340 project.

Christian Kesting and Michael Chen (c) 2023

The Passes page performs or links to CRUD functions.

- The **create** passes function is hosted on the Transactions page. The user is provided a helpful link the transactions page from which they can access the Add Passes page by clicking on a shopping cart icon in the Transactions table.
- The Passes page is populated by **read**ing from the Passes table in the database.
- There are two **update** queries available to the user:
    - The user may change the resort associated with a pass by clicking the pencil icon under the “Change resort” column, which fulfills the project requirement to update the FK in a M:N relationship. The user is taken to the Change Pass Resort page; a screenshot is shown on the next page.
    - The user may reduce the number of available passes by 1, which represents the customer using a day pass.
    - Data validation is not a completed feature, so notably it is possible to cause the number of available passes to reduce below 0.
- The user may click on the trash can icon under the “Delete” column, which **delete**s a pass row from the Passes table.
- There is an additional functionality on the website to display only the passes associated with a specific transaction. The user may click on the “Reload all passes” link to refresh the table and display all passes again.

Change Pass Resort Page (M:N Update)



Change Pass Resort

id: 4

Resort: Choose Resort

Save

Database server is online.
Spring 2023 OSU CS340 project.

Christian Kesting and Michael Chen (c) 2023

The Change Pass Resort page enables the user to update the Resorts_id FK for a particular pass. This fulfills the project requirement to allow the user to update the FK in a M:N relationship.

- The resort field is a drop down menu of resort names populated via a query to the resorts table. The React framework javascript code associates the Resorts id FK with the selected resort name.
- Clicking “Save” will send the request to perform the **update** query and take the user back to the Passes page.