Planing

Website Explanation

My website will be an index of amusement rides at theme parks or under construction which can then be accessed when looking at a park or a type of ride. Then a more complex feature is a park planner where you input the type of rides that you want to do and then my database will find the park with the most amount of the ride types you were looking for and if i have the time create an option to input your location and calculator which parks are the closest and sort it from there.

Website Layout

Colours:

location	option1	option2	option3	chosen
Background	#edd4b2	#fefae0	#f4f1bb	#fefae0
Secondary background				#f4f1bb
Nav bar colour	#222823	#08090A	#242325	#242325
Drop down	#929487	#acadbc	#dbdcda	#7f7f7f

colour				

Fonts:

NAVBAR - mediavail sharp

TEXT - lunsio

Footer - Shadows into light two

Header - quintessential

Layout

Page	description
Parks	alternating background with pictures and the name of a theme park, including location. When clicked on it will take you to a list of rides that are in that park. With a simple description included
Rides	alternating background with pictures and the name of the ride, including the manufacture and type. clicking on the ride will pull up more detail and then clicking on that will take you to the parks that have that exact ride or at least the ride type, it will also have a more detailed in depth look at that ride with a full ride layout with the list of elements included
Home	Just a simple page with a brief description on how to use the website with finding information methods and how to use selection boxes
Manufacturers	A simple page with the list of manufacturers with the same style as the other pages and clicking on a manufacturer will take you to the list

	of rides they have made sorted by the type
Rides elements	A page containing each ride element like a loop or a corkscrew with a description on the forces the element offers and if in is inverted or not and clicking on the page will take you to a list of rides with that element
Ride types	Found by using the dropdown menu the ride types page is just a list of the ride types using the same format as the manufacturers with a picture of that ride (both the track and train) and the name and a description.

Examples

Flask route /manufacturer

Home Parks Rides manufacturers

Name: Location:
description:

BOLLIGER & MABILLARD

Name: Location:
description:

VEKOMA

/parks

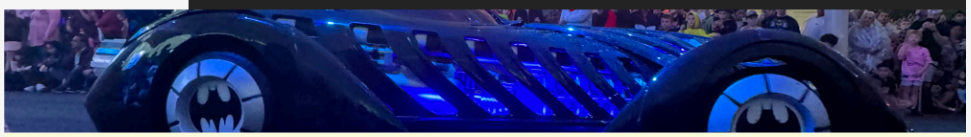
Home Parks Rides manufacturers

Name: Location:
description:

Name: Location:
description:

/

Home Parks Rides manufacturers



Description on how to use the website

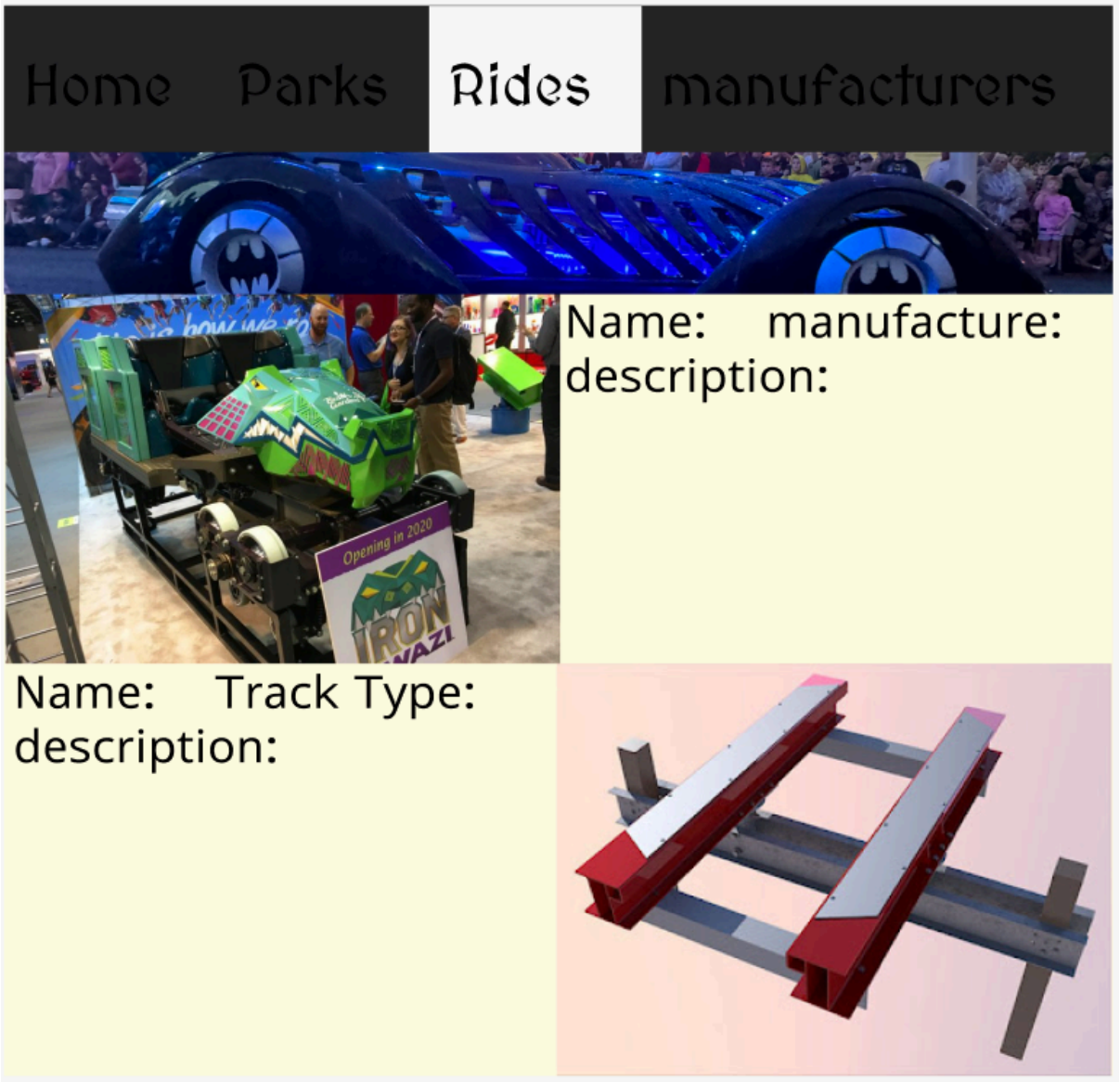
/rideelements

Home Parks Rides manufacturers

Name: inverted: description

Name: inverted: description

/ridetype



Justification:

My targeted end users of theme park enthusiasts more in terms of rides are a lot more interested in what the actual ride is over all the more atmospheric features of a park which is why the rides have the most detail as that is what ends up bringing everyone to those parks.

I am simply going to use the navigation by using the name of each page such as rides, parks etc. to make the website simpler to create when clicking on a park taking you to its ride list the url will be for rides but using the id of the park only its rides will be displayed.

Layout changes:

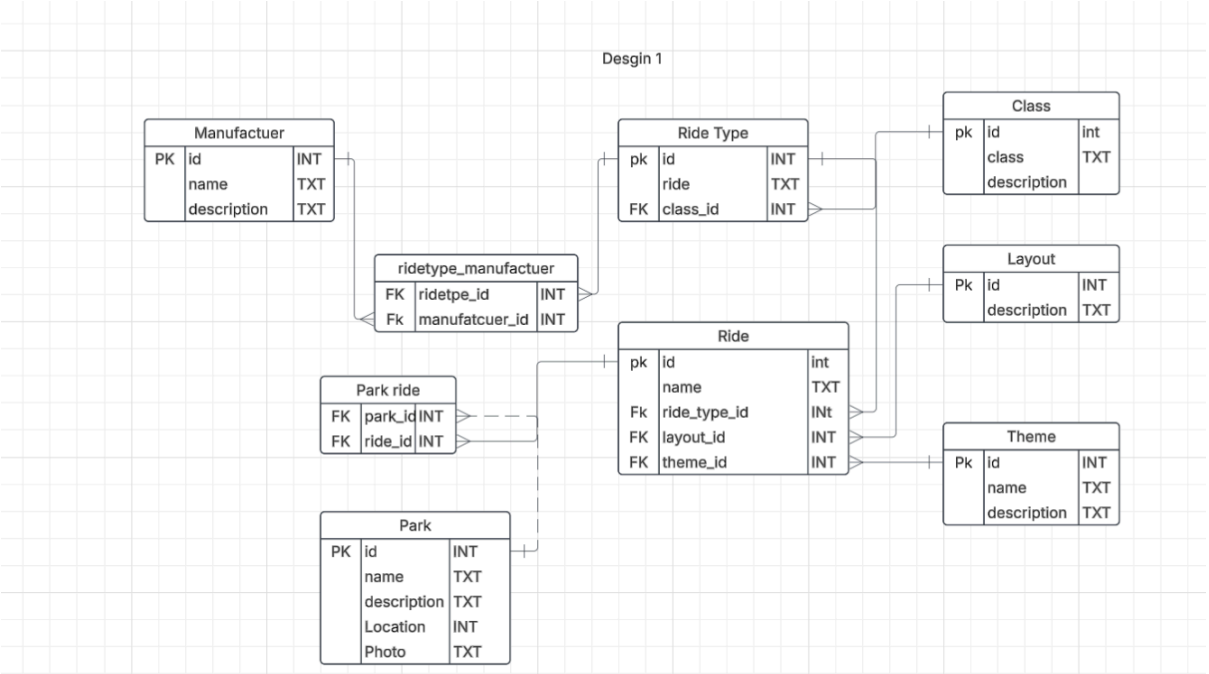
I have decided for the ride pages to sort the rides by height in order so by having the web page start at the bottom as you scroll up you get to taller and taller roller coasters, which should create a cool effect as you can see how high up you have climbed in the roller coaster ladder. This will mean that some java script will be required, to make the web page start at the bottom.

Data management

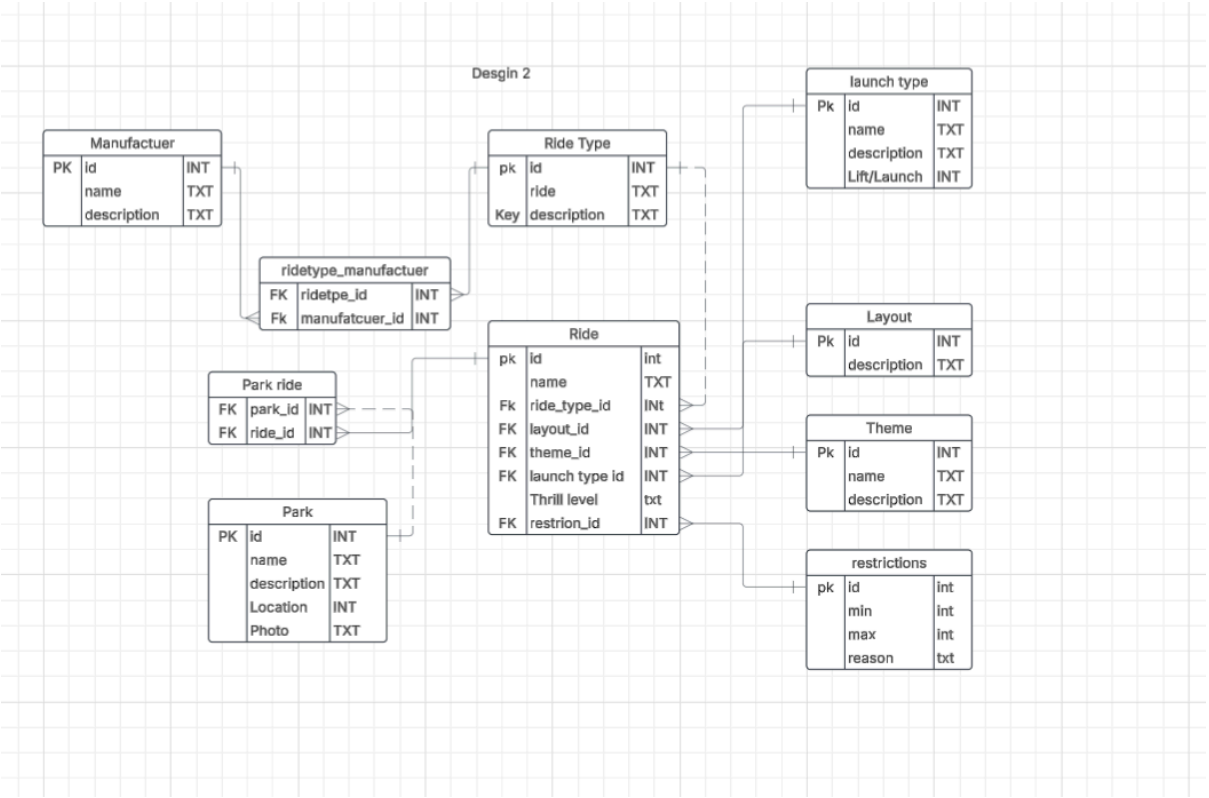
The website will be separated into the main components of the amusement industry being parks, rides and manufacturers. Then in each page it will go through the list of parks with their name and photo and to find out more clicking on the image will take you to that's comments table and give more information about that and links to where it appears elsewhere like when click on a park you can then click on a ride in that park to pull up more information on that because of the complexity of rides and parks without some separation the web page would get too crowded. To find out more on more exact things like just all the ride types or themes then a dropdown menu is used to keep the nav bar looking good without sacrificing detail accessibility, as most of my end users are unlikely to be very tech savvy.

ER Diagrams

Design 1

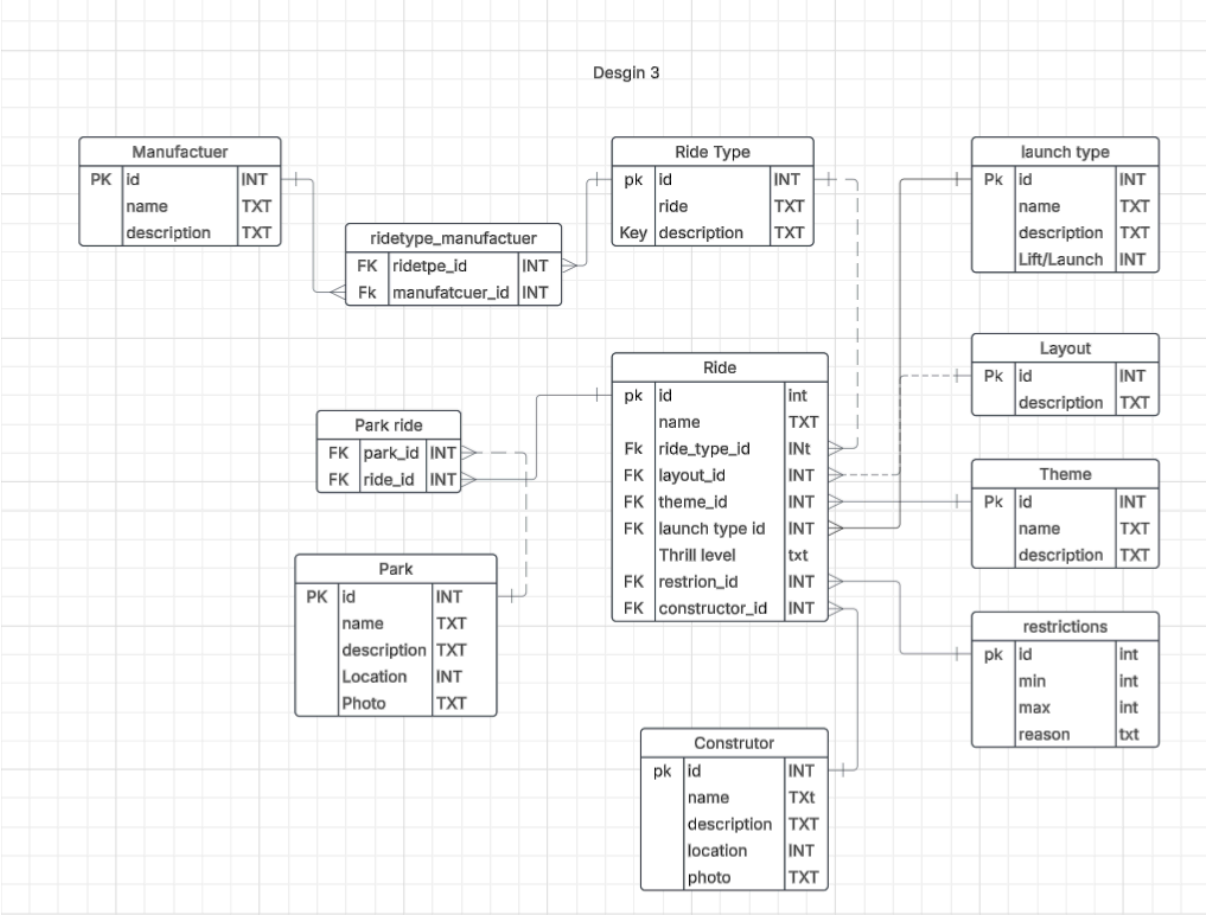


It Started quite complex as I already new most of the important things that had to be there and what i could separate due to ride layout or even full ride clones existing some even with duplicate names as in the disney parks there are multiple space mountains and big thunder railroads

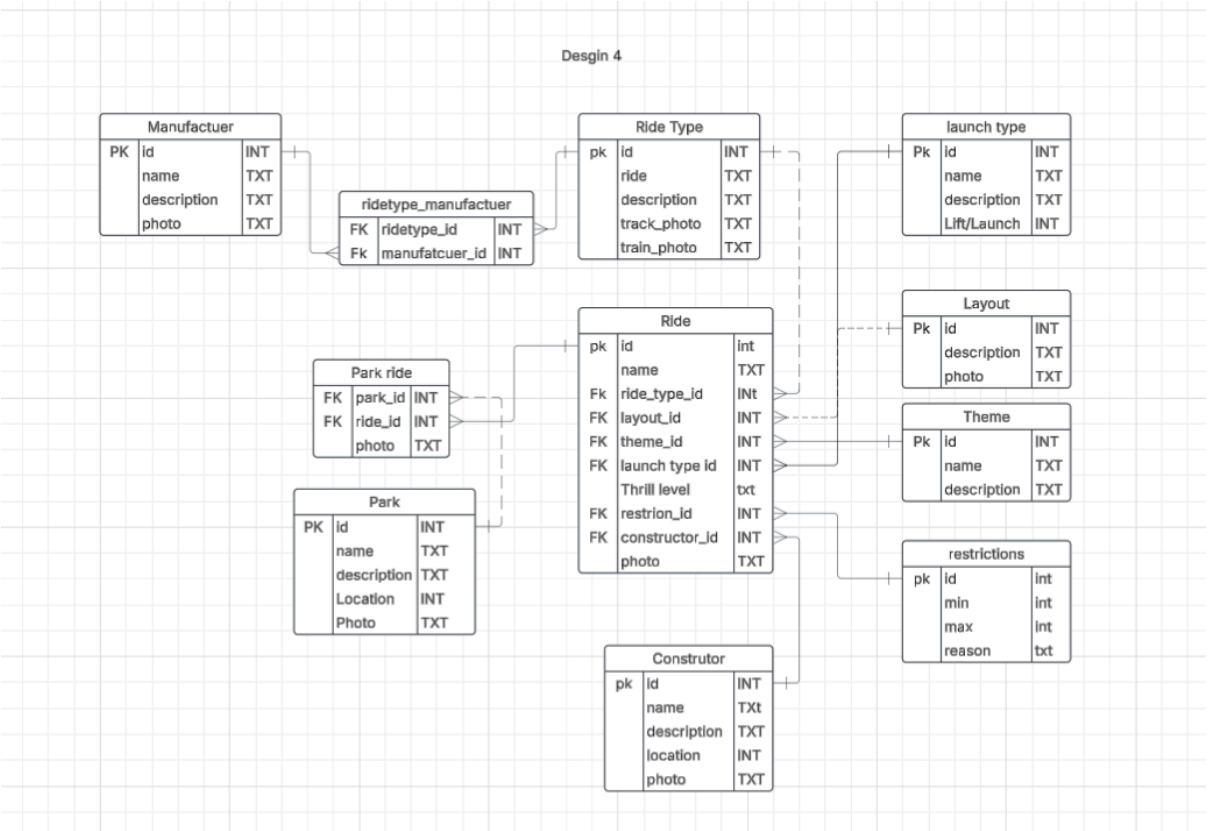


I had to change around some of the tables due to the fact that the main important part is the final ride itself as it can have many changes made from its base type to be more

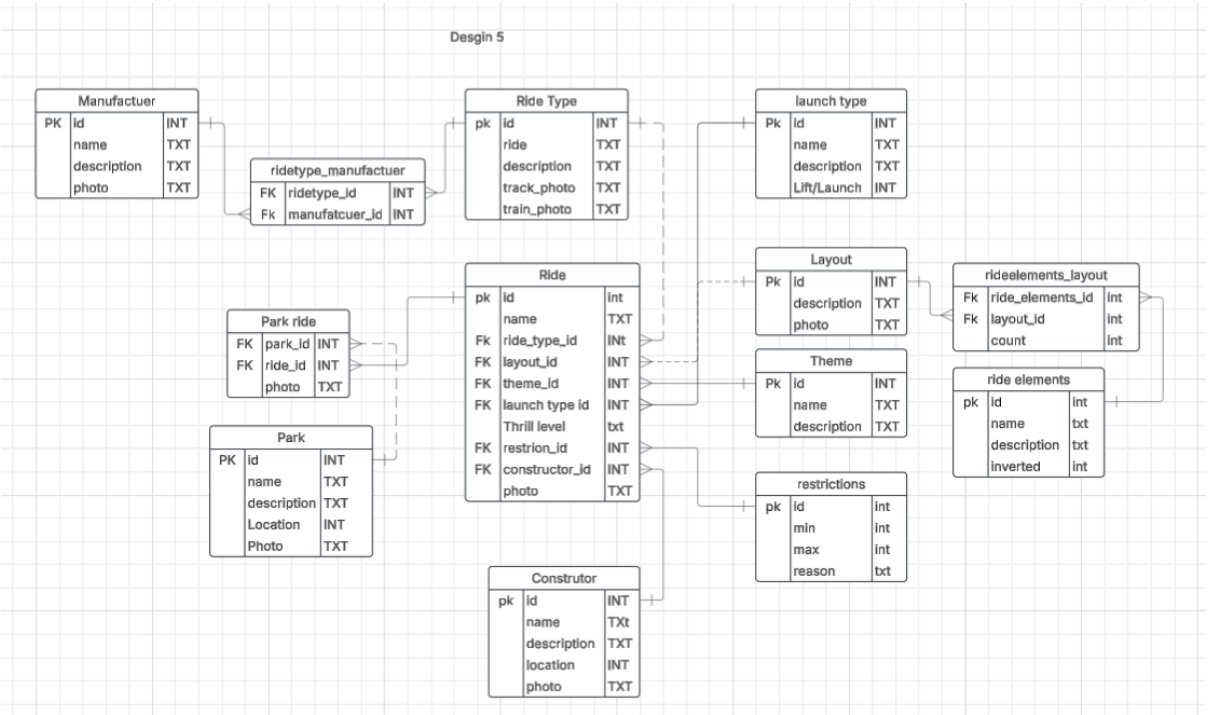
effective for that certain park. I then also added the thrill level which is given to most rides in the world to give a rough estimate to how extreme the ride is. Ride restrictions are also important as if you are too short or tall, or have a medical condition you may not be able to ride some rides.



The constructor table was added as the construction company used to build a ride is different even for the same manufacturer and they can have used different materials and some companies have bad reputations for cost cutting in particular using less bents of wooden roller coasters making them more bumping, jerky and overall a nightmare to maintain.



Extra photos were added where necessary like the photo of the ride in that park and photos for logos and the actual ride type for its track and trains.



To make the layout less exact for each individual ride I can instead give a generic layout description for each ride then have the counter for the more important ride elements like an outer banked turn, a loop or a corkscrew.

SQL/ sqlalchemy

Statements

display rides in a selected park:

```
SELECT
Ride.id, Ride.name, Ride.photo, Ride_Type.ride AS ride_type, Ride.thrill_level
FROM Ride
JOIN Park_Ride ON Ride.id = Park_Ride.ride_id
JOIN Park ON Park_Ride.park_id = Park.id
JOIN Ride_Type ON Ride.ride_type_id = Ride_Type.id
WHERE Park.id = ?
```

Get launched rides:

```
SELECT
Ride.name, Launch_Type.name AS launch_type
FROM Ride
JOIN Launch_Type ON Ride.launch_type_id = Launch_Type.id
WHERE Launch_Type.lift_launch = 1
```

Rides by manufacturer:

```
SELECT Ride.id, Ride.name, Ride.photo
FROM Ride
JOIN RideType_Manufacturer ON Ride.ride_type_id = RideType_Manufacturer.ride_type_id
JOIN Manufacturer ON RideType_Manufacturer.manufacturer_id = Manufacturer.id
WHERE Manufacturer.id = ?
```

Parks ride count:

```
SELECT Park.id, Park.name, Park.photo, COUNT(Park_Ride.ride_id) AS total_rides
FROM Park
LEFT JOIN Park_Ride ON Park.id = Park_Ride.park_id
GROUP BY Park.id, Park.name, Park.photo
ORDER BY total_rides DESC
```

Find all thrill levels for filtering

```
SELECT DISTINCT thrill_level FROM Ride ORDER BY thrill_level ASC
```

Inverting rides

```
SELECT
Ride.id, Ride.name, Ride.photo
FROM Ride
JOIN RideElements_Layout ON Ride.layout_id = RideElements_Layout.layout_id
JOIN Ride_Elements ON RideElements_Layout.ride_elements_id = Ride_Elements.id
WHERE Ride_Elements.inverted = 1
```

Ride details

```
SELECT
Ride.id, Ride.name, Ride.photo,
Ride_Type.ride AS ride_type,
Layout.description AS layout,
Launch_Type.name AS launch_type,
Ride.thrill_level,
Restrictions.min AS min_height, Restrictions.max AS max_height
FROM Ride
JOIN Ride_Type ON Ride.ride_type_id = Ride_Type.id
JOIN Layout ON Ride.layout_id = Layout.id
JOIN Launch_Type ON Ride.launch_type_id = Launch_Type.id
JOIN Restrictions ON Ride.restriction_id = Restrictions.id
WHERE Ride.id = ?
```

Number of rides for each type;

```
SELECT
Ride_Type.ride AS ride_type, COUNT(Ride.id) AS total_rides
FROM Ride
JOIN Ride_Type ON Ride.ride_type_id = Ride_Type.id
GROUP BY Ride_Type.ride
```

Get rides by a park

```
SELECT
Park.name AS park_name, Ride.name AS ride_name
FROM Park_Ride
JOIN Park ON Park_Ride.park_id = Park.id
JOIN Ride ON Park_Ride.ride_id = Ride.id
```

WHERE Park.name = ?

Filter rides by thrill level
SELECT id, name, photo FROM Ride
WHERE thrill_level = ?
ORDER BY name ASC

Sample outputs are in the views of my database.

models.py

```
# Association table for many-to-many relationship between RideType and Manufacturer
RideTypeManufacturer = db.Table('ridetype_manufacturer',
    db.Column('ridetype_id', db.Integer, db.ForeignKey('ride_type.id')),
    db.Column('manufacturer_id', db.Integer, db.ForeignKey('manufacturer.id'))
)

# Association table for ride elements and layouts
RideElementsLayout = db.Table('rideelements_layout',
    db.Column('ride_elements_id', db.Integer, db.ForeignKey('ride_elements.id')),
    db.Column('layout_id', db.Integer, db.ForeignKey('layout.id')),
    db.Column('count', db.Integer)
)

# Association table for many-to-many relationship between Park and Ride
ParkRide = db.Table('park_ride',
    db.Column('park_id', db.Integer, db.ForeignKey('park.id')),
    db.Column('ride_id', db.Integer, db.ForeignKey('ride.id')),
    db.Column('photo', db.Text)
)

class Manufacturer(db.Model):
    __tablename__ = 'manufacturer'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.Text)
```

```
    description = db.Column(db.Text)
    photo = db.Column(db.Text)

    ride_types = db.relationship('RideType', secondary=RideTypeManufacturer,
back_populates='manufacturers')


class RideType(db.Model):
    __tablename__ = 'ride_type'
    id = db.Column(db.Integer, primary_key=True)
    ride = db.Column(db.Text)
    description = db.Column(db.Text)
    track_photo = db.Column(db.Text)
    train_photo = db.Column(db.Text)

    manufacturers = db.relationship('Manufacturer', secondary=RideTypeManufacturer,
back_populates='ride_types')
    rides = db.relationship('Ride', backref='ride_type')


class RideElement(db.Model):
    __tablename__ = 'ride_elements'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.Text)
    description = db.Column(db.Text)
    inverted = db.Column(db.Integer)


class LaunchType(db.Model):
    __tablename__ = 'launch_type'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.Text)
    description = db.Column(db.Text)
    lift_launch = db.Column(db.Integer)  # 0 = Lift, 1 = Launch

    rides = db.relationship('Ride', backref='launch_type')


class Layout(db.Model):
    __tablename__ = 'layout'
```

```
    id = db.Column(db.Integer, primary_key=True)
    description = db.Column(db.Text)
    photo = db.Column(db.Text)

    rides = db.relationship('Ride', backref='layout')


class Restriction(db.Model):
    __tablename__ = 'restrictions'
    id = db.Column(db.Integer, primary_key=True)
    min = db.Column(db.Integer)
    max = db.Column(db.Integer)
    reason = db.Column(db.Text)

    rides = db.relationship('Ride', backref='restriction')


class Theme(db.Model):
    __tablename__ = 'theme'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.Text)
    description = db.Column(db.Text)

    rides = db.relationship('Ride', backref='theme')


class Constructor(db.Model):
    __tablename__ = 'constructor'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.Text)
    description = db.Column(db.Text)
    location = db.Column(db.Integer)
    photo = db.Column(db.Text)

    rides = db.relationship('Ride', backref='constructor')


class Ride(db.Model):
    __tablename__ = 'ride'
    id = db.Column(db.Integer, primary_key=True)
```

```
    name = db.Column(db.Text)
    ride_type_id = db.Column(db.Integer, db.ForeignKey('ride_type.id'))
    layout_id = db.Column(db.Integer, db.ForeignKey('layout.id'))
    theme_id = db.Column(db.Integer, db.ForeignKey('theme.id'))
    launch_type_id = db.Column(db.Integer, db.ForeignKey('launch_type.id'))
    thrill_level = db.Column(db.Text)
    restriction_id = db.Column(db.Integer, db.ForeignKey('restrictions.id'))
    constructor_id = db.Column(db.Integer, db.ForeignKey('constructor.id'))
    photo = db.Column(db.Text)

    parks = db.relationship('Park', secondary=ParkRide, back_populates='rides')


class Park(db.Model):
    __tablename__ = 'park'
    id = db.Column(db.Integer, primary_key=True)
    name = db.Column(db.Text)
    description = db.Column(db.Text)
    location = db.Column(db.Integer)
    photo = db.Column(db.Text)

    rides = db.relationship('Ride', secondary=ParkRide, back_populates='parks')
```

Sqlalchemy queries for the web pages

What it's for	SQLAlchemy Query	Sample Output
Home Page	*No query — static page*	Renders: home.html with page_title = "HOME"
Parks Page	parks = models.Park.query.all()	1. Cedar Point — cedar_point.jpg 2. Universal's Islands of Adventure — universal_ioa.jpg

Rides Page	rides = models.Ride.query.all()	1. Steel Vengeance — steel_vengeance.jpg — Extreme — Hybrid Coaster 2. VelociCoaster — velocicoaster.jpg — High — Giga Coaster
Manufacturers Page	manufacturers = models.Manufacturer.query.all()	1. Rocky Mountain Construction — rmc.jpg 2. Intamin — intamin.jpg 3. B&M — bm.jpg 4. Vekoma — vekoma.jpg
Ride Elements Page	elements = models.RideElement.query.all()	1. Vertical Loop — Inverted: Yes 2. Airtime Hill — Inverted: No
Ride Types Page	ride_types = models.RideType.query.all()	1. Hybrid Coaster — hybrid_track.jpg 2. Giga Coaster — giga_track.jpg
Park Details	park = models.Park.query.filter_by(id=park_id).first_or_404()	Cedar Point — Ohio, USA — cedar_point.jpg
Ride Details	ride = models.Ride.query.filter_by(id=ride_id).first_or_404()	Steel Vengeance — Extreme — Hybrid Coaster — steel_vengeance.jpg
Rides by Manufacturer	rides = models.Ride.query.join(models.RideTypeManufacturer).join(models.Manufacturer).filter(models.Manufacturer.id == manufacturer_id).all()	1. Steel Vengeance — Rocky Mountain Construction — steel_vengeance.jpg
Rides by Element	rides = models.Ride.query.join(models.RideElementsLayout).join(models.RideElement)	1. Steel Vengeance — Vertical Loop — steel_vengeance.jpg

	.filter(models.RideElement.id == element_id).all()	
Rides by Type	rides = models.Ride.query.filter_by(ride_type_id =type_id).all()	1. Steel Vengeance — Hybrid Coaster — steel_vengeance.jpg
Manufacturer Details	manufacturer = models.Manufacturer.query.filter_by(id= manufacturer_id).first_or_404()	Rocky Mountain Construction — Hybrid coaster specialists. — rmc.jpg
Ride Element Details	element = models.RideElement.query.filter_by(id= element_id).first_or_404()	Vertical Loop — A 360-degree vertical loop. — Inverted: Yes
Ride Type Details	ride_type = models.RideType.query.filter_by(id=type _id).first_or_404()	Hybrid Coaster — A mix of wood and steel. — hybrid_track.jpg — hybrid_train.jpg

Justifications

For the web pages I am going to use selection typing boxes where it will have the direct names of the items in the database sorted alphabetically but as you type it will only show the options that have the same set of starting letters. This approach is used to still give fast search but decrease the amount of spelling that can potentially cause errors. Therefore I do not need to have an id route at the top of the page preventing any errors for undesired id values like 0 or infinity. This benefit is made by the options appearing using id values so it is highly impossible for incorrect values and id errors because the names of the items used for searching come from the id values as this selection box is created by the table. Dropdown menus have a great level of detail whilst not taking up a lot of space combined with the searching options the website is robust having the ability to display all sets of the import and less important information without sacrificing aesthetics. This website is useful to my end users of the theme park enthusiast as there is such a wide variety of parks and rides,

with sheer number this website can become a one stop destination to finding the perfect theme park to go to for the rides you are looking for at the shortest possible distance.

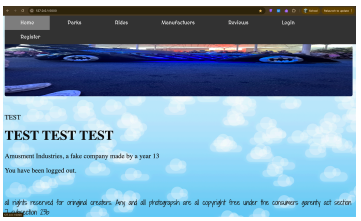
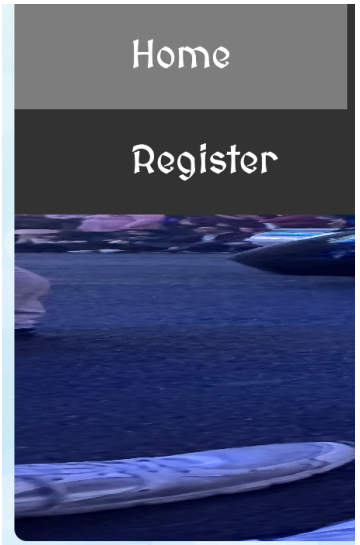
User Feedback

Person	Coding and theme park experience	Feedback given	Key take away
Adrian Sawyer	Little to no coding experience but enjoys theme parks	Lots of spelling and grammatical errors to fix along with some layout changes with the location of the register and login pages, and to fix the few errors in the website when reviewed at that time such as the currently broken add ride feature and the dismiss buttons not working.	Go over text to make sure it is grammatically correct and ensure that all other errors are fixed by doing rigorous tests.
Mr Dunford	Very well experienced with coding and not very experienced with theme parks	Most of the feedback was to do with meeting project requirements in order to get excellence, some of what I needed to do was add in error pages for the program to be robust, shorten overly long join	I have gone through and added an error 404 page to ensure even with buggy routes everything still will work as well as fixing the line to long and ambiguous variable name errors from flake 8. As well, I have completed the documentation

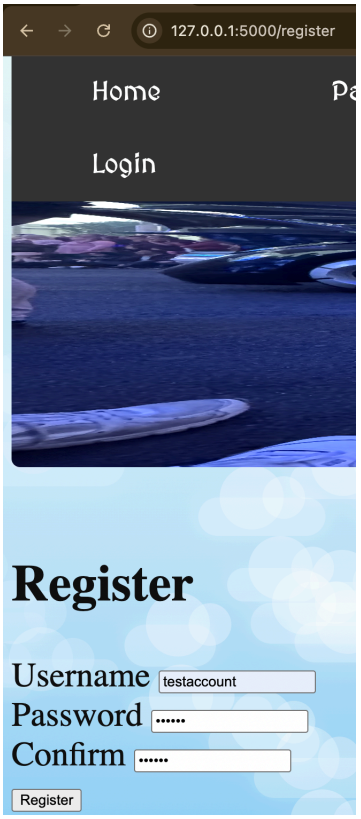
		statements with sqlalchemy, neilsons heuristics, data integrity and, website testing to be done thoroughly in order to get full excellence	steps that I needed to make with Nesilion's heuristics being covered and have gone through and completed the necessary testing after adding in each new feature.
William Cochrane	Well experienced with both	Centre the text in the selection boxes and have the star rating system go up to 7	Centring the selection boxes is a good idea however i will not change the rating system as 1-5 makes more sense and is more consistent with other sites than a 1-7
Ephrem chan	Not a programmer has been to one theme park before	Give a search function for the reviews that would be great and a confirmation message for deleting things, give a comma between thrill level and ride name	The changes that ephrem had suggested were very good and useful and weren't too complex to implement with the time I have so I managed to input all of them and they have made the website easier and better to understand and use.

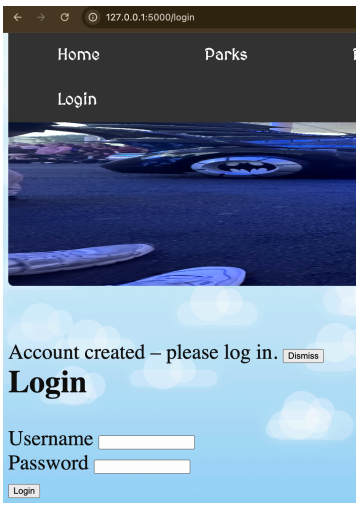
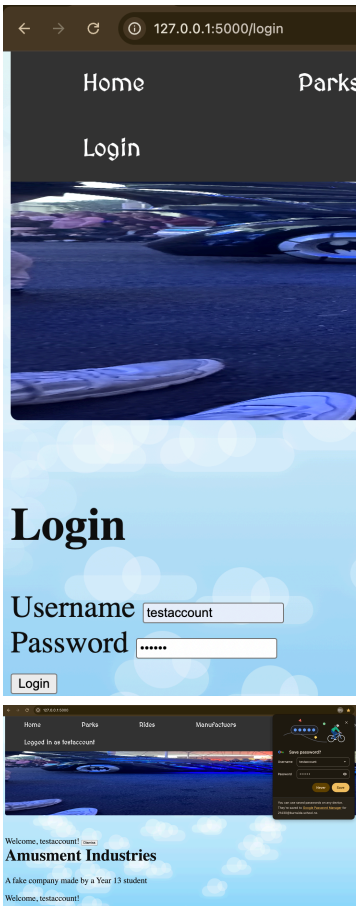
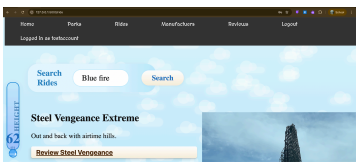
Testing:

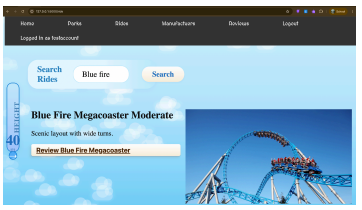
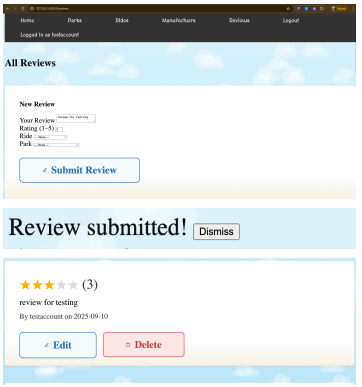
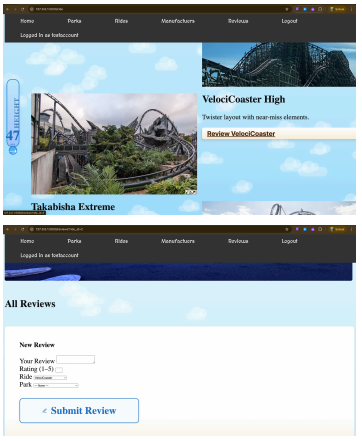
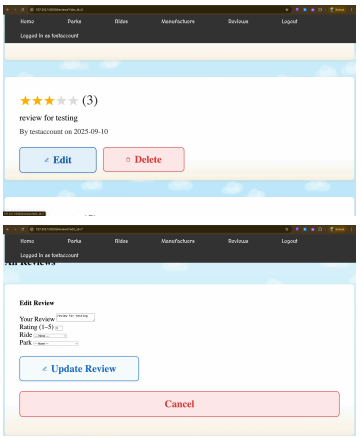
WHAT	HOW	EXPECT	GET	P/F/notes

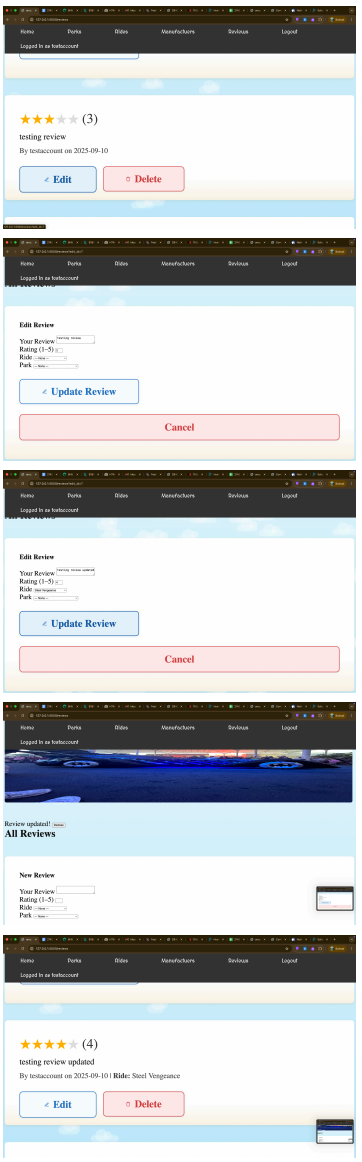
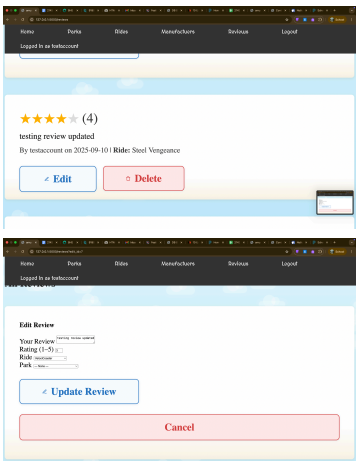
Home page	Click on the home button on the nav bar	Taken to the home page	  	P

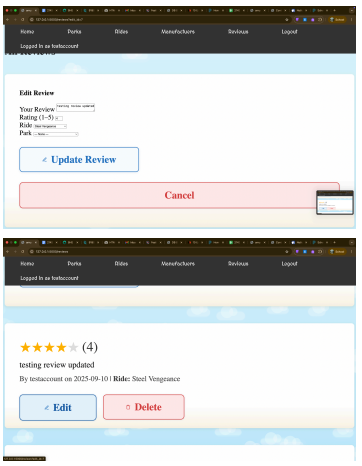
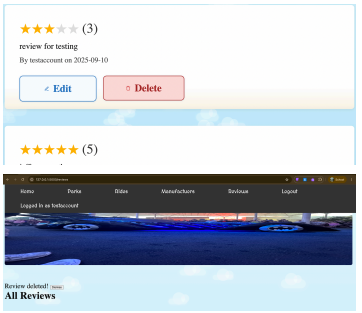
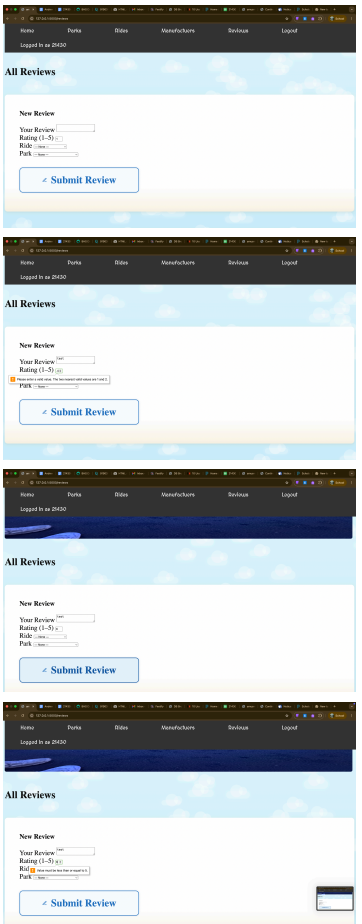
Text hover	Hover over text	Text should become bold		P
navbar	Click on a different web page on the top navbar	Taken to that page		P

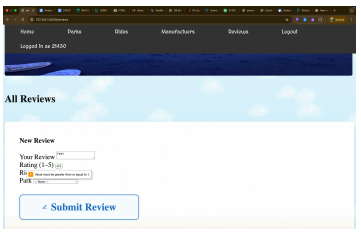
Dropdown menu	Click on a page inside of the dropdown menu	Taken to that page		P
Register page	Create a new account	Taken to the login page with a message saying registration excepted		P

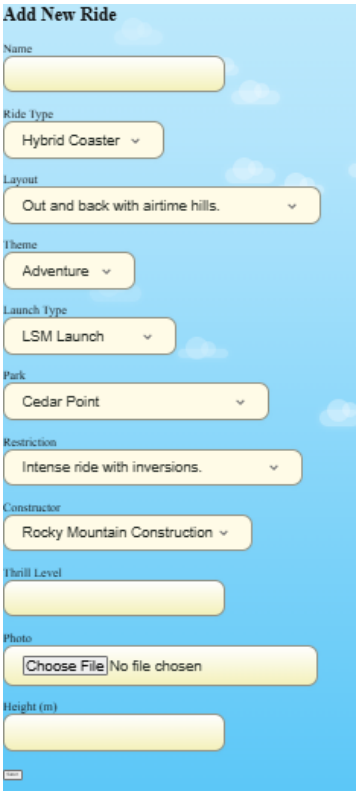
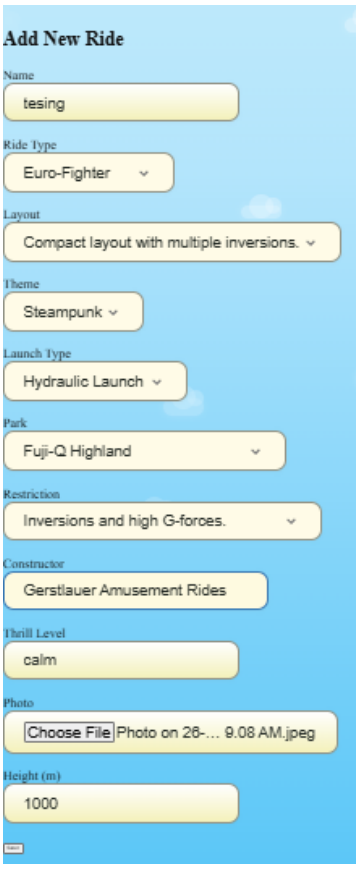
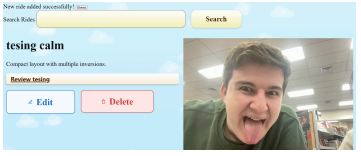
				
Login page	Login to the account just created	Logged in on the home page with the navbar updating to show logged in status		P
Ride search bar	Search for a ride	Taken to a page with that ride		P

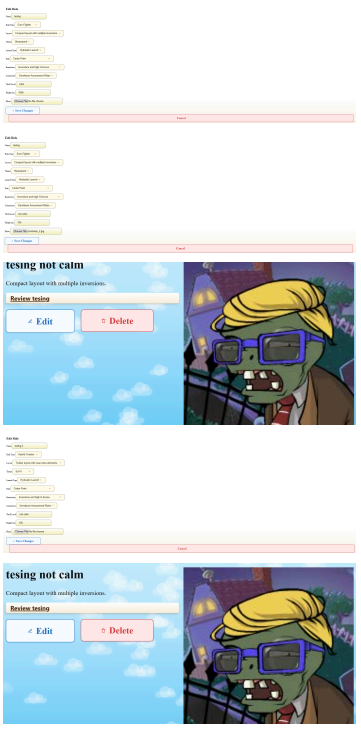
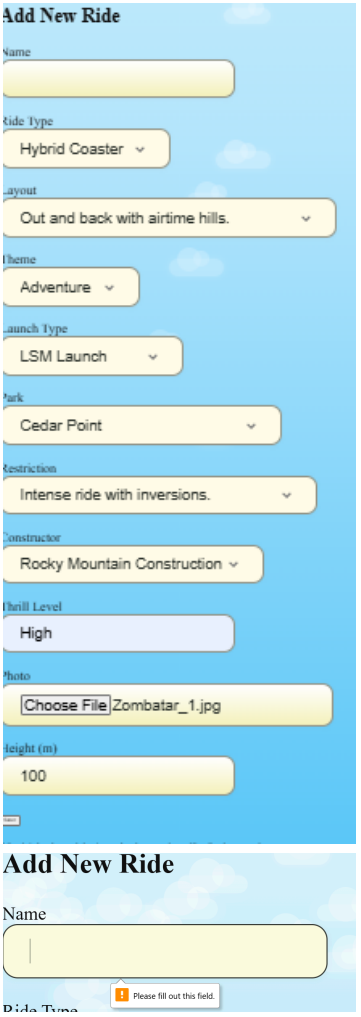
				
Review	Use the review system to review an item	Message saying review is successful and is present on the page		P
Review button	Click on it in the ride page	Taken to review with the ride already filled in on the review page		P
Edit review	Click on the edit button under one of my reviews	To be taken to the review which I can edit with a save and cancel button		P

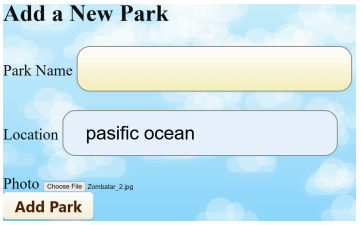
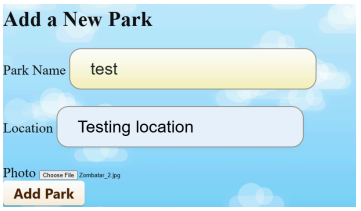
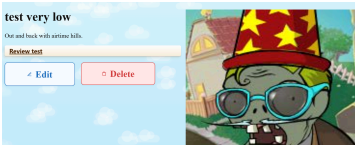
Editing the review	Make changes to a review and then save the changes	Should have a confirmation message that can be dismissed saying review updated		P
Cancel a review edit	Make some changes to a review then click on cancel	The review should not of changed		

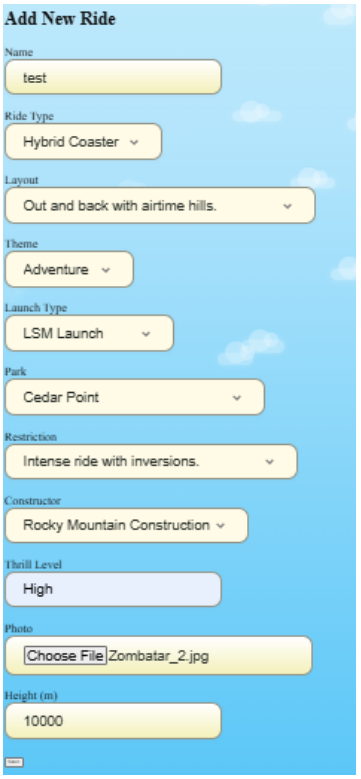
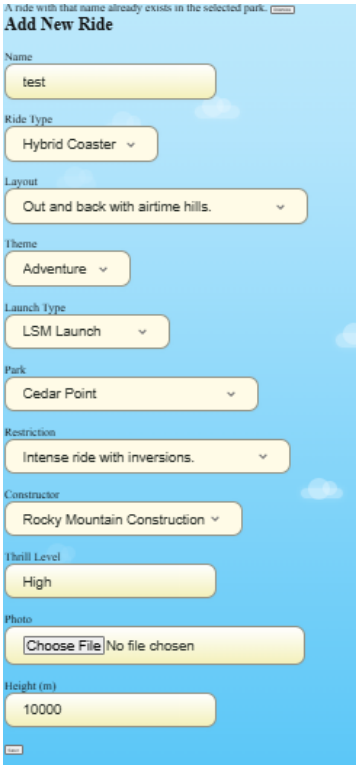
				
Delete review	Click on the delete button under the review	Review should be deleted with a confirmation before being processed		P
Stars system for reviews	Try to put in a value that is not an integer between 1-5 and test a number above 5 and below 1	For the first case it should say to pick a number that is biassed the one input i.e 2.4 should say only whole numbers values and suggest 2 or 3, and for the second case it shay pick a number less than/ greater than or equal that said number		P

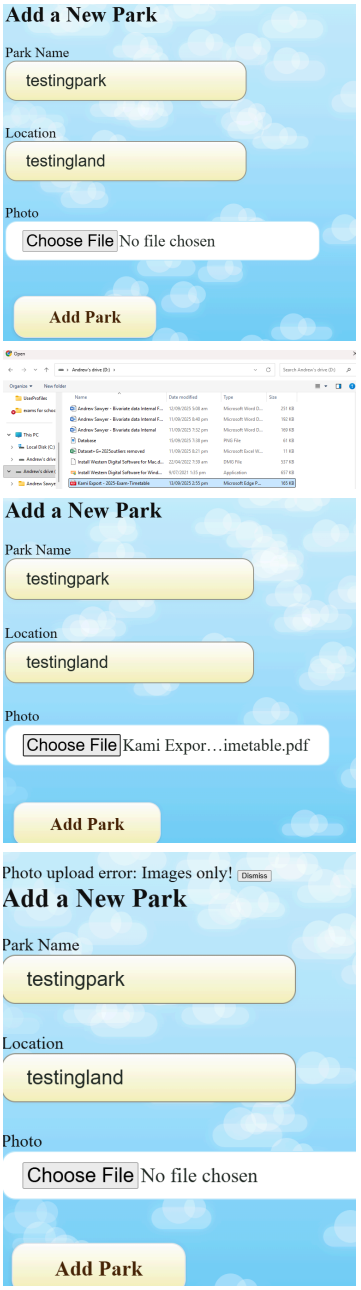
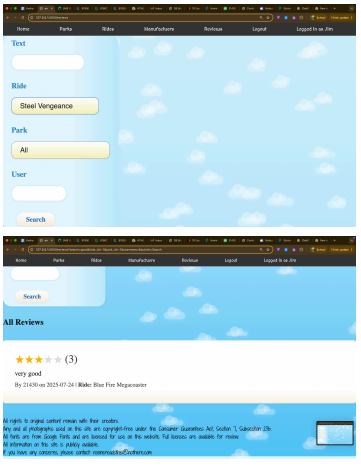
			 Home Parks Rides Manufacturers Reviews Logout Logged In as 25430 All Reviews New Review Your Review: Rating (1–5): Submit Review	

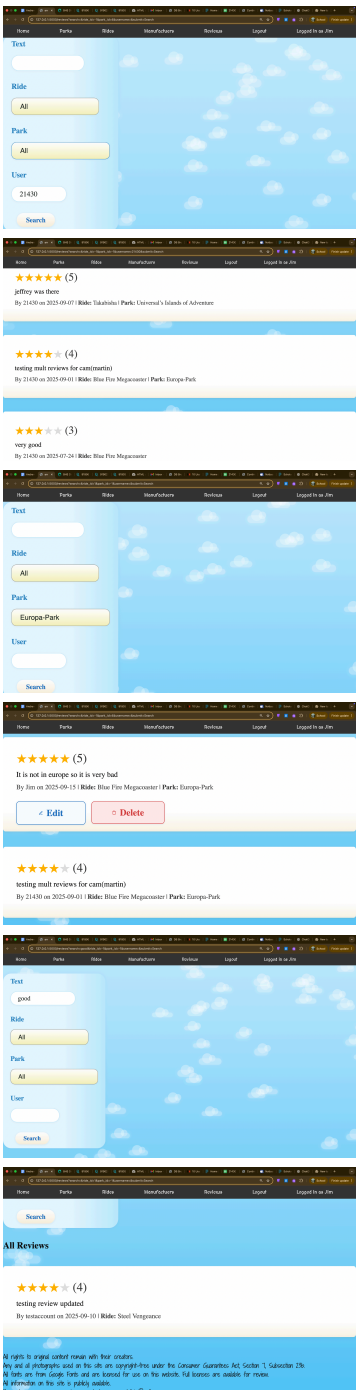
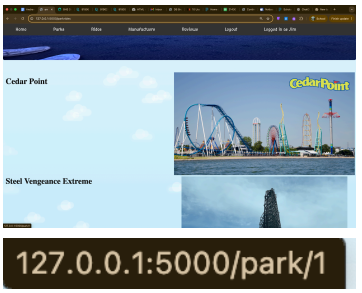
Add ride/park (same code idea for both)	Add a ride show that it is there then edit it and finally delete	It should all work using selection boxes for recognition over recall.	  	P

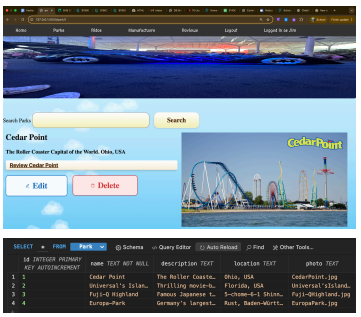
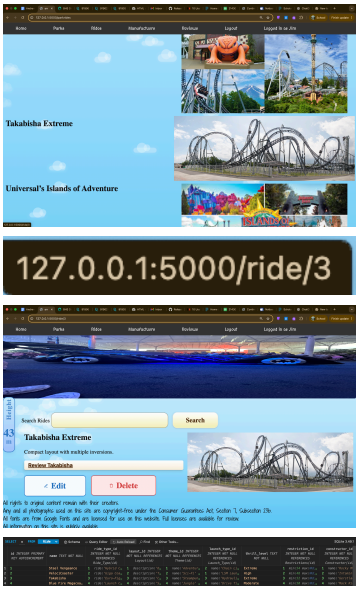
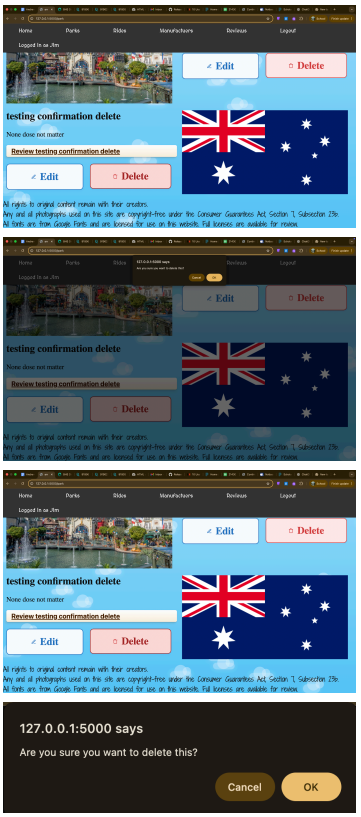
				
Blank park/ ride	Add a ride and park that are blank with no name	Should fail to make as a name is required		P

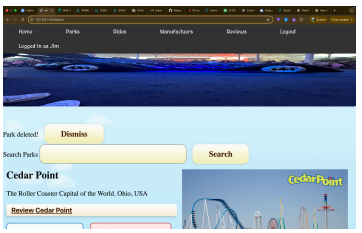
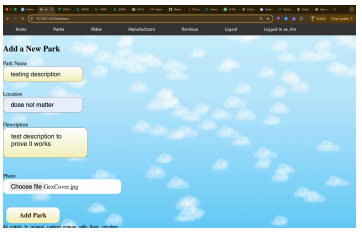
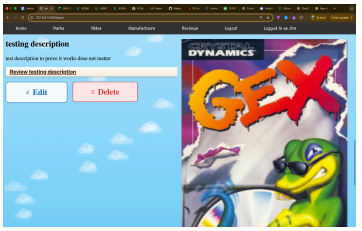
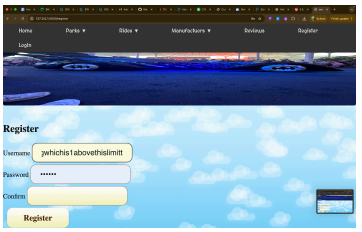
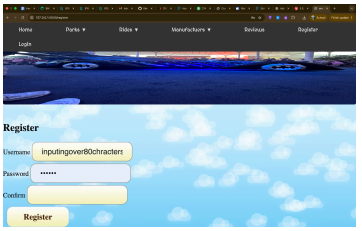
			 	
Add a ride/ park with a duplicate name	Add in a test ride/park then do it again	Should fail as two parks can not have the same name	   	P

			 Add New Ride Name test Ride Type Hybrid Coaster Layout Out and back with airtime hills. Theme Adventure Launch Type LSM Launch Park Cedar Point Restriction Intense ride with inversions. Constructor Rocky Mountain Construction Thrill Level High Photo Choose File Zombatar_2.jpg Height (m) 10000  A ride with that name already exists in the selected park. Add New Ride Name test Ride Type Hybrid Coaster Layout Out and back with airtime hills. Theme Adventure Launch Type LSM Launch Park Cedar Point Restriction Intense ride with inversions. Constructor Rocky Mountain Construction Thrill Level High Photo Choose File No file chosen Height (m) 10000	

Wrong file type upload	Try to upload a file that is not a png or jpg or jpeng	Should fail and say that only those file types are supported		P
Review search bar boxes	Select a ride or park and press search	Should be a list of reviews only for that ride or park		P

				
Park data integrity	Go to park listed as id 1	Should be taken to the park that is id 1 in the database		P

				
Ride Data integrity	Got to the ride listed as id 3	Should be taken to the ride that is id 3 in the database		p
Delete confirmation	Delete a park but then cancel the delete then confirm it latter	Should cancel the delete for the first one and just continue on the page, then when pressing ok the standard deletion process should happen		P

				
Testing parks add description	Type in a description for a ride	Should add the park with that description put in	 	P
Test max and min length	Type in 100 characters into a selection box	Should say that it is too large/stop at the 100 character	  	P

Relevant implications

Website:

Accessibility	My website needs to be accessible to all end users allowing them to access all information at a fast acceptable speed whilst being able to be read easily by all users. One way I have done this is by hovering over text it will become bolded making it easier to read against the background, I have also made the images enlarge when hovering over them as well. My website is runnable on all modern browsers with a standard font size and high contrasting colours (black, blue and white)
Aesthetics	My website needs to meet Aesthetics requirements as an appealing website can lead to end users leaving the website or not taking information seriously if the website looks silly. I have met these requirements by using a nice appealing gradient background of blue colours along with black text to have a modern colour pallet. I have also used modern website design techniques of a standardised grid layout across all pages.

Database:

Ethical	My website does not use any pirate or sensitive information about the theme park industry as all the information found is publicly available on each parks website with no sensitive numbers published by them on the site. My website is up to date with the start of 2025 and is listed as that on the home page with the ability for it to be updated with the adding features of the website, so the information is not misleading.
Intellectual property/ Legal	It is essential that all the images and fonts that I am using are obtained rightfully and

	respect the rights of the copyright holders. The images I am using all fall under fair trading and are for educational purposes as it is for people finding out and learning more about the amusement industry.
Sustainability and Future proofing	It is intended that my website can be updated biannually to go along with common park completing and building time in each hemisphere. This can be done by the add park and ride options on the website adding new thrills for everyone.

Development Process:

Stage 1; Database:

When I started my project I looked to start by creating the tables for my er diagram with all of the information I wanted to display, such as rides, parks and manufacturers. I then looked to expand each of these tables to fill in more relevant information which then expanded and grew into more tables. Once I had thought that I had completed my database to be able to hold and display all the desired information I wanted I then looked to make mock up pages to get an idea for what they would look like. In these designs I made sure to follow neilson consistency and standards heuristic making sure my website followed the same conventions as other websites to ensure my website would be easy to navigate for all users, such as having a navigation bar at the top of the screen with a dropdown menu contain all relevant web pages that relate to the previous topic. Then while adding in some test data into my database for view testing I made sure to follow Nielson's heuristic of match between the system and the real world, by ensuring to use the standard naming conventions for each ride park and elements. Instead of giving exact track specs for complex elements or describing them indepth, I have instead just gone with their simple "coaster" definition of just an overbanked turn or non-inverted loop, as some examples instead of 135° turn.

Stage 2; Functioning basic website:

Once I had finished my mock up designs and had a functioning database with views working correctly I started by implementing one of my first databases to be implemented into a flask

application. This was harder than usual as this year we are using SQLAlchemy which requires a models.py file and a new language model to be used. However after completing the first table and then a many to many it became pretty repetitive and self explanatory for the rest of the tables for the final er diagram to be added so thus I quickly added the full database to the flask web application. I then created a basic layout.html page as well as some basic html pages to start populating the website with data. When creating the review page after having basic data, I made sure to have error prevention when writing a review by adding a pop up message if your number is outside of the rating limit of 1-5 stars, this even includes if you put a decimal value of 3.4 for example it will ask you to change it to the nearest value in this case 3 or 4. This meets the error prevention nesilson-heuristic, the other heuristic I added in for reviews is recognition rather than recall, with the review system as well. Rather than having to remember a ride or park's name or id, I have added a selection box to then just select the name of the ride and or park without having to recall the name. This meets the heuristic and makes my website better and easier to use for my end users.

Stage 3; Testing:

During the testing phase I had to ensure that all of my pages were working and fully functional in order for it to be classified as a program, so I underwent numerous testing. After ensuring all basic functionality was working such as the navbar and search box, I looked at the three important and complex parts of my website that being the reviews login and add/edit rides. These pages are the most complex and since they deal with user input that gets added to the database which requires much more testing, to ensure there are no errors getting added and committed to the database. I then proceeded to get some extra feedback and realised that my website was missing some important features that had to be tested to ensure they worked such as the search form for reviews and a confirmation button for deleting to prevent unwanted and accidental deletes.

Stage 4; Final Outcome:

My final outcome is fully robust with error pages to handle any potential errors, I have also comprehensively tested my forms using flask WTF with validators to ensure no one could add in 10 peta bytes of data to my database and ruin it. Thus my website is robust against large or small amounts of data and by using flask WTF it is safe from inspect attacks of changing the limits as edge cases are accounted for, with an error page to boot. After completing all of the testing and stylised everything making the final project a modern robust web page. However if I had more time I would like to expand the website in order to fit the whole industry better then how it currently is focused on only the core theme park experience for rollercoaster enthusiasts.