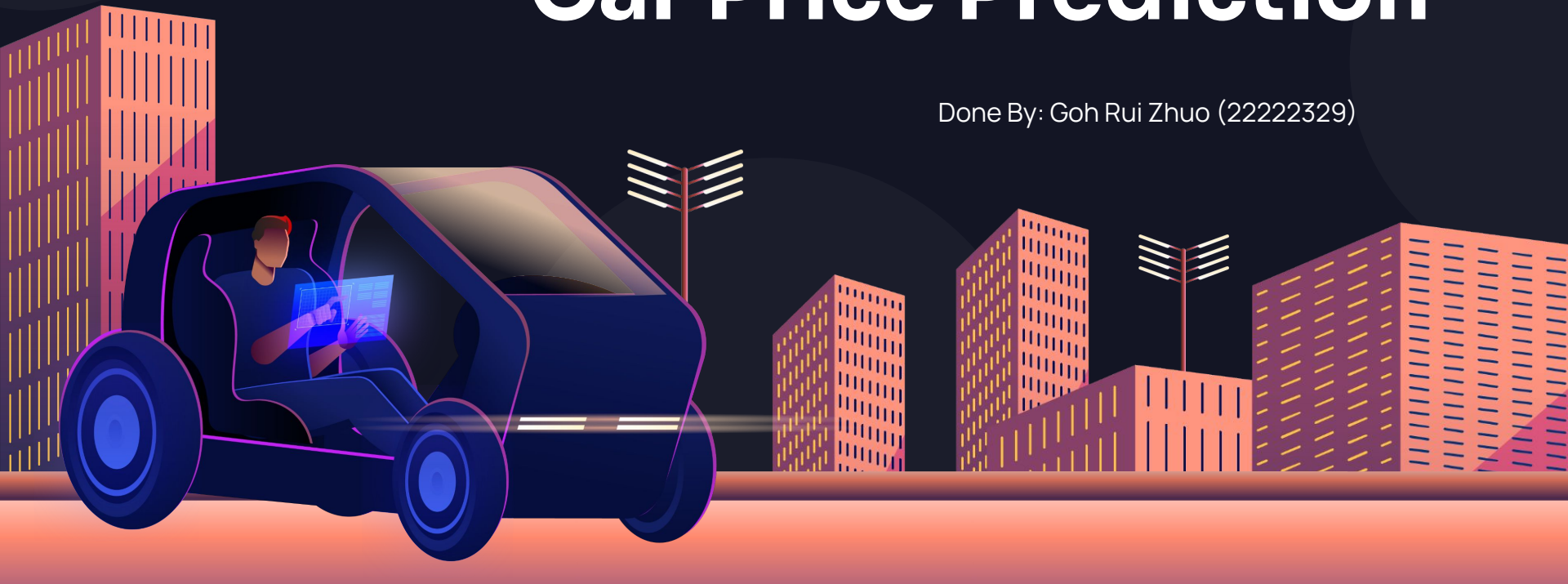
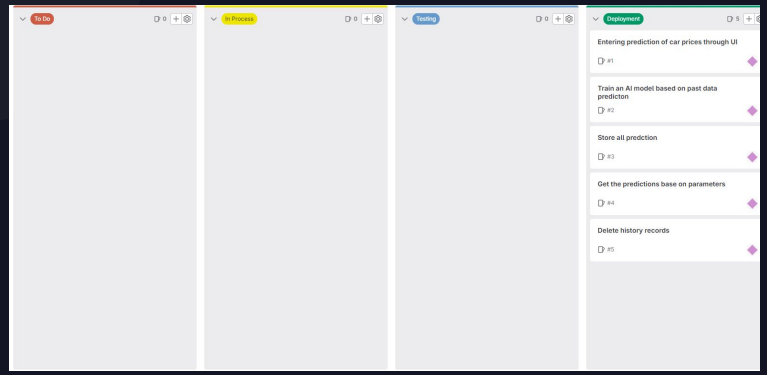
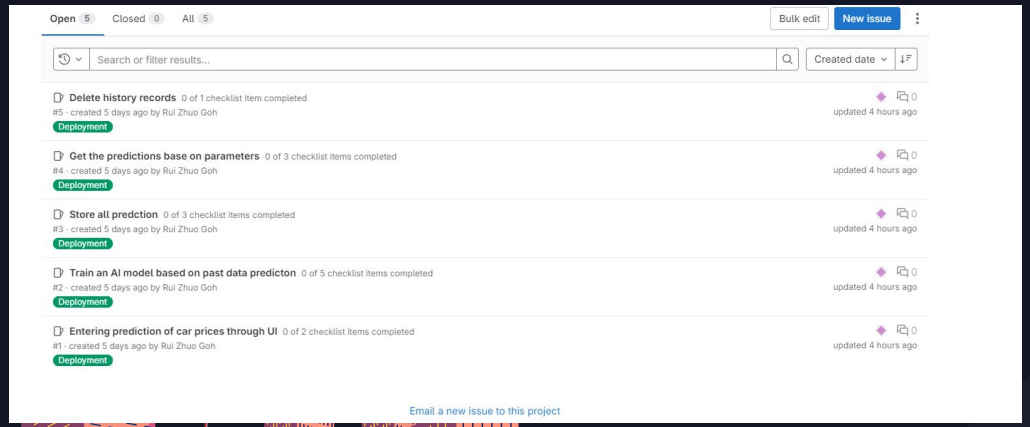
Car Price Prediction

Done By: Goh Rui Zhuo (22222329)



Scrum Board





1. Added four section which are
   a. To Do
   b. In Process
   c. Testing
   d. Deployment

2. Testing comes before deployment in internet services

3. 5 overall issue boards

Branch

1. Wireframe Branch
   a. Upload a wireframe for how the webpage will look like
2. Notebook Branch
   a. Upload the notebook branch which contains the finalised model
3. AppPredict_branch
   a. Integrate the model into the webpage
   b. Add login, register page
   c. Contains history database
4. AppImprove_branch
   a. Improve on UI
   b. Add more functions
5. Pytest_branch
   a. Adding the different test for the unit testing
6. Deploy_branch
   a. Prepare to deploy on render

Model Development

1. Dataset was selected with close to 20000 training rows with 36 columns
2. 15 columns was selected
3. Features are selected based on the importance of it

General Info

1. Null values can be seen
2. Distribution of data seems ok

brand	2
year	2
mileage	2
engine	47
engine_size	1249
transmission	105
fuel_type	2
drivetrain	2
min_mpg	3006
damaged	169
turbo	2
navigation_system	2
backup_camera	2
first_owner	306
price	2

	year	mileage	engine_size	min_mpg	damaged	
count	19107.000000	19107.000000	17860.000000	16103.000000	18940.000000	1
mean	2017.773120	47947.087403	2.973191	20.827796	0.217371	
std	4.996385	38285.061200	5.478008	6.017846	0.412468	
min	1962.000000	0.000000	0.000000	0.000000	0.000000	
25%	2016.000000	19307.000000	2.000000	17.000000	0.000000	
50%	2019.000000	39141.000000	2.500000	20.000000	0.000000	
75%	2021.000000	68431.500000	3.500000	24.000000	0.000000	
max	2024.000000	383614.000000	390.000000	89.000000	1.000000	

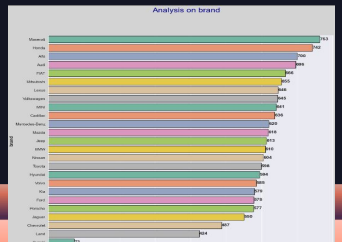
Model Development

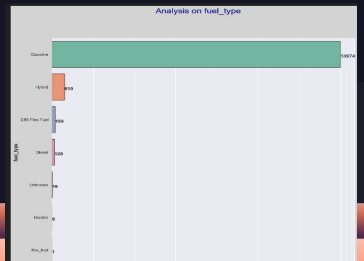
Data Preprocessing

1. Dropping of null values
   a. Imputing may be a viable option but it can result in inaccurate values produced , therefore the best option is to drop the rows
2. Change the dtype of the price column
   a. Datatype of the price is an object, hence it is required to change the dtype of the price

Exploratory Data Analysis

1. For brands, we can see that this dataset contains Maserati the most and Suzuki the least
2. For fuel type, we can see that this dataset contains Gasoline the most and Flex type the least
3. For fuel drivetrain, we can see that this dataset contains Four Wheel Drive the most and Unknown the least





Model Development

Exploratory Data Analysis

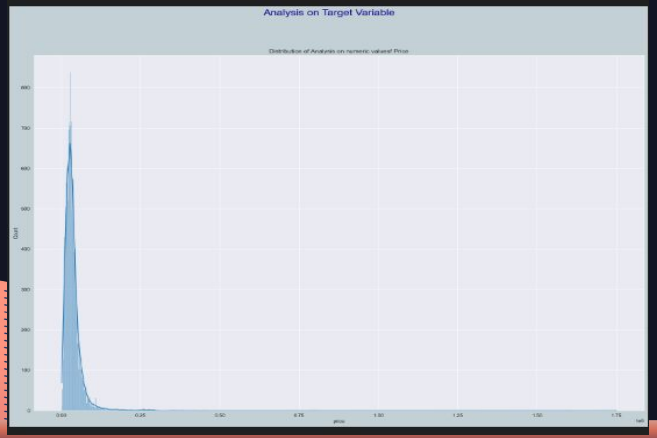
1. For mileage, we can see that this dataset is positively skewed
2. For engine size, we can see that this dataset pretty symmetrical distribution

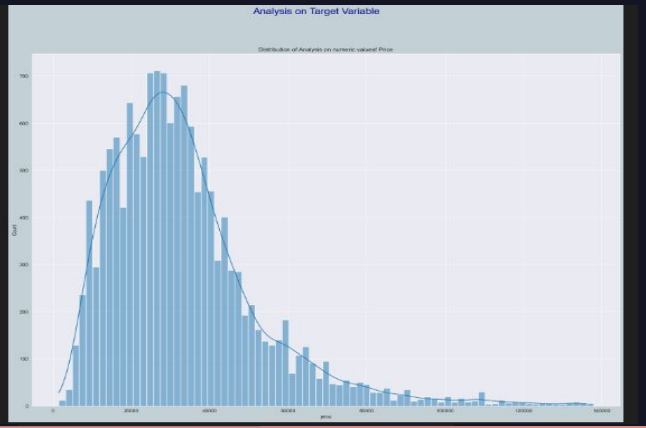
Analysis on Target Variable

1. For the target variable, we can see that the data contains extreme outliers

Data Preprocessing (2)

1. Extreme outlier was removed





Model Development

Feature Engineering

1. One Hot Encoder to encode
2. Age column was feature extracted
3. Standardisation was done

Model Development (Baseline)

1. Utilise multiple models
2. Best model are CatBoost with a r2 score of 0.91
3. Random forest was also chosen to hyperparameter tune

After Hyperparameter Tuning

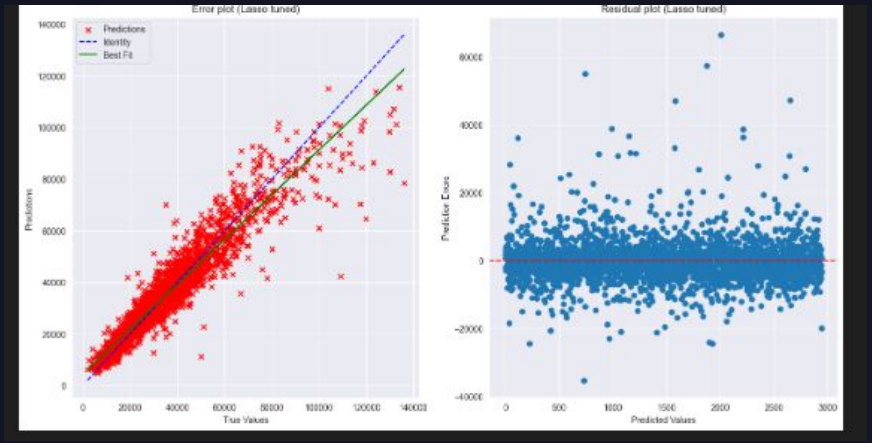
1. Fnal model is Cat Boost Tuned
2. Pipeline was utilized

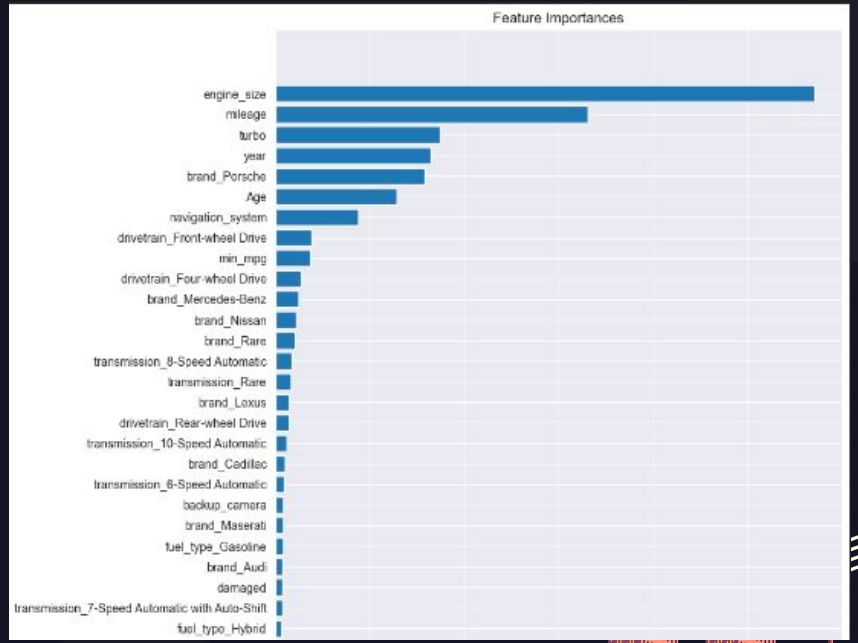
odel	fit_time	score_time	test_r2	train_r2	test_neg_root_mean_squared_error	train_neg_root_mean_squared_error	test_neg_mean_absolute_error	train_neg_mean_absolute_error	test_neg_mean_absolute_percentage_erro
CB	5.516140	0.053153	0.907988	0.948690	5853.851180	4379.544362	3691.023374	2998.195734	0.1198
XGB	0.398391	0.053254	0.896091	0.965067	6220.455916	3613.325774	3915.915031	2526.241080	0.1276
sting	2.483957	0.055267	0.890733	0.923071	6381.464614	5362.665842	4049.428998	3558.672704	0.1326
aTree	9.909779	0.072885	0.890136	1.000000	6395.633869	0.362273	3935.230406	0.006693	0.1301
stReg	11.701957	0.058584	0.884021	0.983668	6575.807892	2470.872781	4095.156074	1524.855741	0.1341
sting	2.036913	0.041187	0.851185	0.866834	7451.020565	7055.763792	4855.512567	4668.504426	0.1607
KNN	0.161788	0.122632	0.796741	0.868965	8700.185955	6998.969552	5506.882886	4369.596289	0.1886
nTree	0.275978	0.027358	0.783312	1.000000	8992.188964	0.000000	5537.747070	0.000000	0.1803
Ridge	0.115060	0.029323	0.727836	0.732284	10069.733527	10004.250625	6614.888194	6579.056158	0.2595
Lasso	4.916584	0.029928	0.727835	0.732284	10069.758864	10004.254723	6614.680679	6578.817915	0.2594
arReg	0.182802	0.033133	0.727803	0.732078	10070.478782	10008.094458	6614.855375	6578.247959	0.2593
Boost	1.004053	0.046301	0.350528	0.364056	15562.366951	15414.149581	13770.553754	13704.897736	0.6669

```python
preprocessor_pipe = Pipeline([
    ("num_var_imputer", MeanMedianImputer(imputation_method="median", v
    ("cat_var_imputer", CategoricalImputer(imputation_method="frequent"
    ("rare_label_encoder", RareLabelEncoder(
    tol=0.03, n_categories=2, variables=cat_cols)),
    ("one_hot_encoder", OneHotEncoder(
    variables=cat_cols, ignore_format=True)),
    ("scaling", StandardScaler()),
])
```

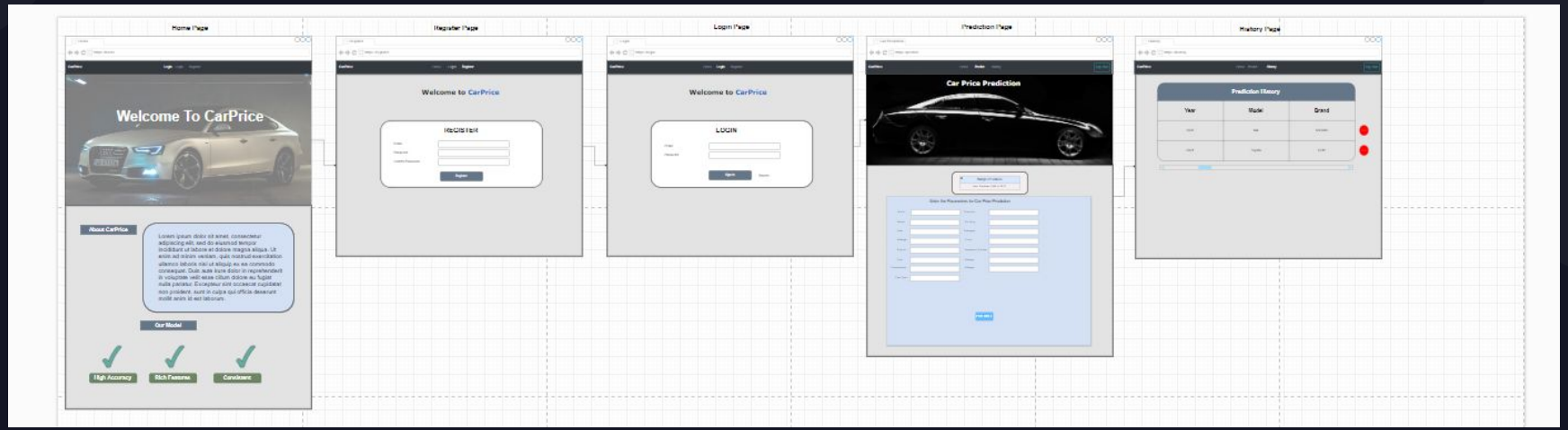
Model Development

1. Model Predict pretty accurately
2. Have Engine size as the most important feature





Wireframe


Home Page
Welcome To CarPrice
Register Page
Welcome to CarPrice
Login Page
Welcome to CarPrice
LOGIN
Prediction Page
Car Price Prediction
History Page
Prediction History
Year
Model
Brand

Wireframe

Idea Behind Wireframe

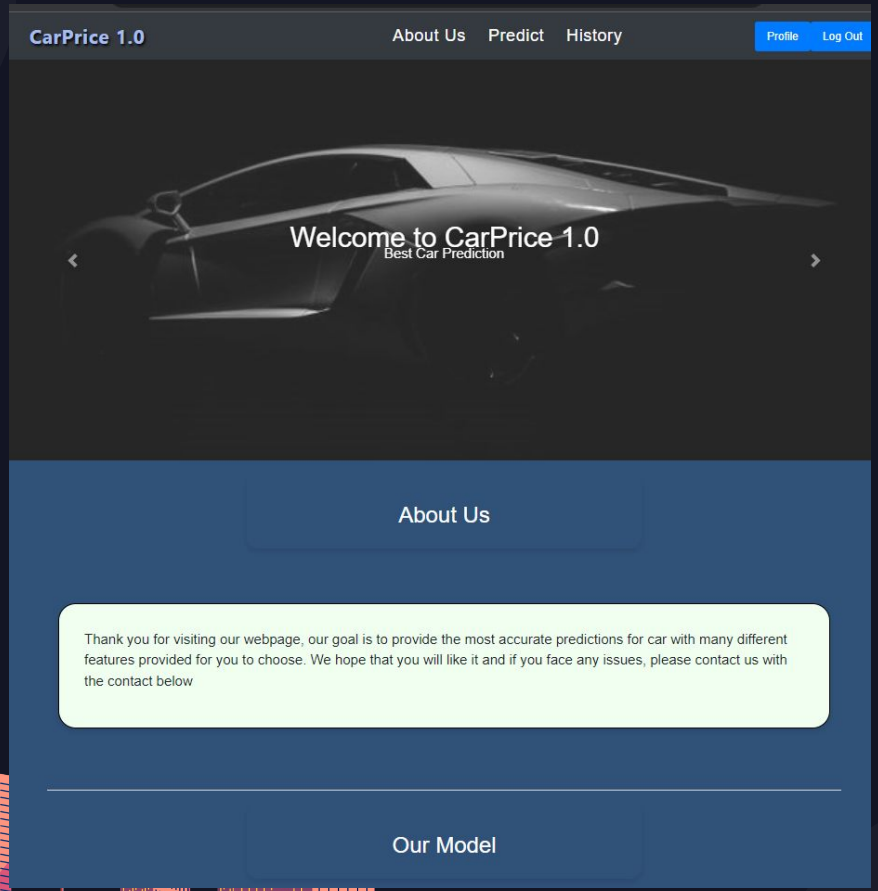
1. Clean and color on the darker theme
2. Added photos of cars to showcase the webpage
3. Having the 5 key pages
4. Model prediction contains drop down menu
5. Prediction history store in table form in a single webpage with ability to delete records

Deploying ML Model

1. Trained model was loaded
2. Flask app was utilised together with the trained model

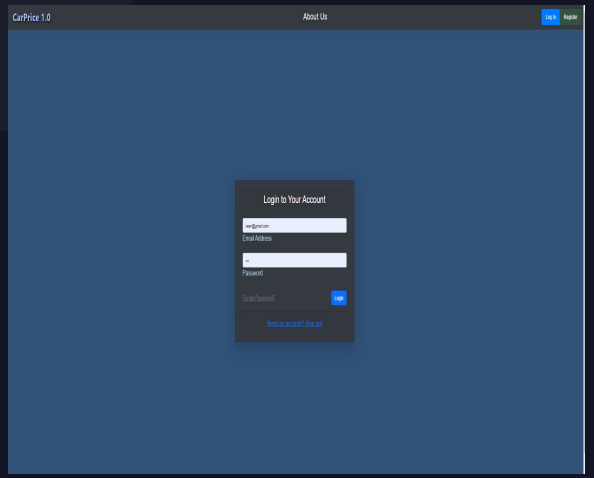
```
# Import the login manager
from flask_login import LoginManager
manager = LoginManager()
manager.init_app(app)

# Import the load from joblib to use ml file
from joblib import load

joblib_file = "./application/static/finalized_model.pkl"
ai_model = load(joblib_file)

# Import the routes
from application import routes
```

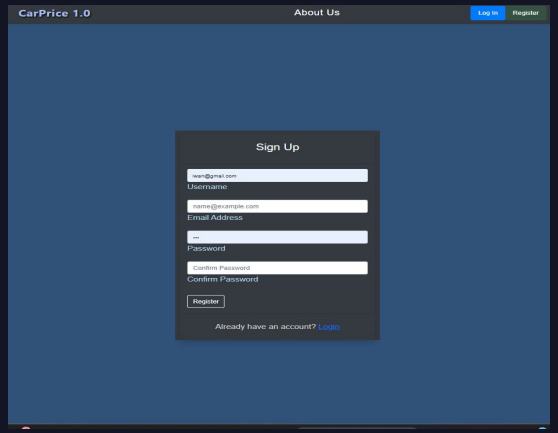
Web Page (Home)



1. Home page design with the use of cars images
2. Added an introduction
3. Provided idea of our model

Web Page (Log In and Sign Up)



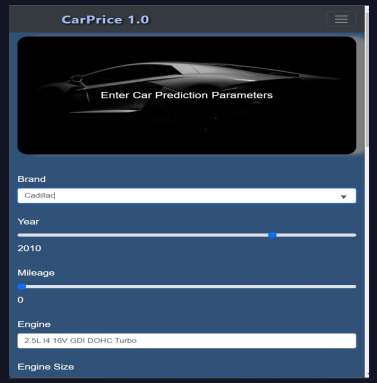


1. Login Page provides a clean and responsive look
2. It provides additional option for password resetting and all
3. Provides redirecting to register page if user has not log in before
4. Deployed with flask framework

```python
# Set the login route
@app.route('/login', methods=['GET','POST'] )
def login():

    # Set up the login form here
    form = Login()

    # If methods is equal to post
    if request.method == 'POST':

        # Validate on submit here
        if form.validate_on_submit():

            # Filter the uyser by their id
            user = User.query.filter_by(email = form.email.data ).first()
            # Debugging tools
            # print(user)

            # Check whether the password is correct
            if user:
                # Utilise password hash to hash the password so that the company is not able to see
                if check_password_hash(user.password_hash , form.password.data):
                    print('login')
                    login_user(user)

                    # Redirect to predict
                    return redirect('/predict')

                # If wrong password here
                else:
                    flash('Wrong password' , 'danger')
            # If user does not exist here
            else:
                flash('User does not exist' , 'danger')

        # Unexpected error
        else:
            flash('Error, Unable to login' , 'danger')

    # Return the rendered template here
    return render_template('login.html', form=form, title="Sign Up" , index = True )
```

```python
# Set the registration methods
@app.route('/register', methods=['GET', 'POST'])
def register():

    # Set up the form for registration
    form = RegistrationForm()
    # If it is a post method
    if request.method == 'POST':
        if form.validate_on_submit():

            # Check for existing user here

            existing_user = User.query.filter_by(email=form.email

            # If there is no existing user
            if existing_user is None:

                # It sets the password and
                user = User(username=form.username.data, email=fo

                # Set the password after validation of password
                user.set_password(form.password_1.data)

                # Commit to the database
                db.session.add(user)
                db.session.commit()

                # Redirect the user to login page here
                return redirect(url_for('login'))

            # If the user already exist, warmog will be shown
            else:
                flash('A user with that email already exists.')

    # Return the render template here
    return render_template('register.html', form=form)
```

Web Page (Predict)



1. Provides seamless autocomplete for user input text
2. Provide slider for numerical value
3. Provide dropdown menu for multiple options and binary options
4. Used flask framework

```python
# Creating the prediction form
class PredictionForm(FlaskForm):
    brand_choices = [(brand, brand) for brand in brand_names]
    engine_choices = [(engine, engine) for engine in engines]
    fuel_choices = [(fuel, fuel) for fuel in fuel_types]
    transmission_choices = [(transmission, transmission) for tran
    drivetrain_choices = [(drivetrain, drivetrain) for drivetrain
    brand = SelectField("Brand", choices=brand_choices, validator
    year = FloatField("Year", validators=[InputRequired(), Number
    mileage = FloatField("Mileage", validators=[InputRequired(),
    engine = SelectField("Engine", choices=engine_choices, valida
    engine_size = FloatField("Engine Size", validators=[InputRequ
    transmission = SelectField("Transmissions", choices=transmiss
    fuel_type = SelectField("Fuel Type", choices=fuel_choices, va
    drivetrain = SelectField("Drive Train", choices=drivetrain_ch
    min_mpg = FloatField("Minimum MPG", validators=[InputRequired
    damaged = SelectField('Damaged', choices=[('','') , (1, 'Yes'
    turbo = SelectField('Turbo', choices=[('','') , (1, 'Yes'),(0
    navigation_system = SelectField('Navigation System', choices=
    backup_camera = SelectField('Back Up Camera', choices=[('','
    first_owner = SelectField('New', choices=[('','') , (1, 'Yes'
    submit = SubmitField("Predict")
```

```python
# Set the predict methods
@app.route("/predict", methods=['GET','POST'])
@login_required #login is required for this
def predict():

    # If user is not authenticated here
    if not current_user.is_authenticated:
        flash('You have not logged in yet.', 'warning')

        # Redirect them back to login
        return redirect(url_for('login'))

    # Set up the prediction form here
    form = PredictionForm()

    # Retrieve the brands and models here
    brands = get_brands()
    prediction = None

    # If post request is requested
    if request.method == 'POST':

        if form.validate_on_submit():

            # Setting of the data here
            brand = form.brand.data
            year = int(form.year.data)
```

Database

```python
# Creating the user database to store info of user
class User(UserMixin, db.Model):
    __tablename__ = "User"
    id = db.Column(db.Integer, primary_key=True,autoincrement=True)
    username = db.Column(db.String(50), index=True, unique=True)
    email = db.Column(db.String(150), unique = True, index = True)
    password_hash = db.Column(db.String(150))
    joined_at = db.Column(db.DateTime(), default = datetime.utcnow, index = True)

    def set_password(self, password):
        self.password_hash = generate_password_hash(password)

    def check_password(self,password):
        return check_password_hash(self.password_hash,password)
```

1. Created the user database in sqlite
2. Store prediction inside

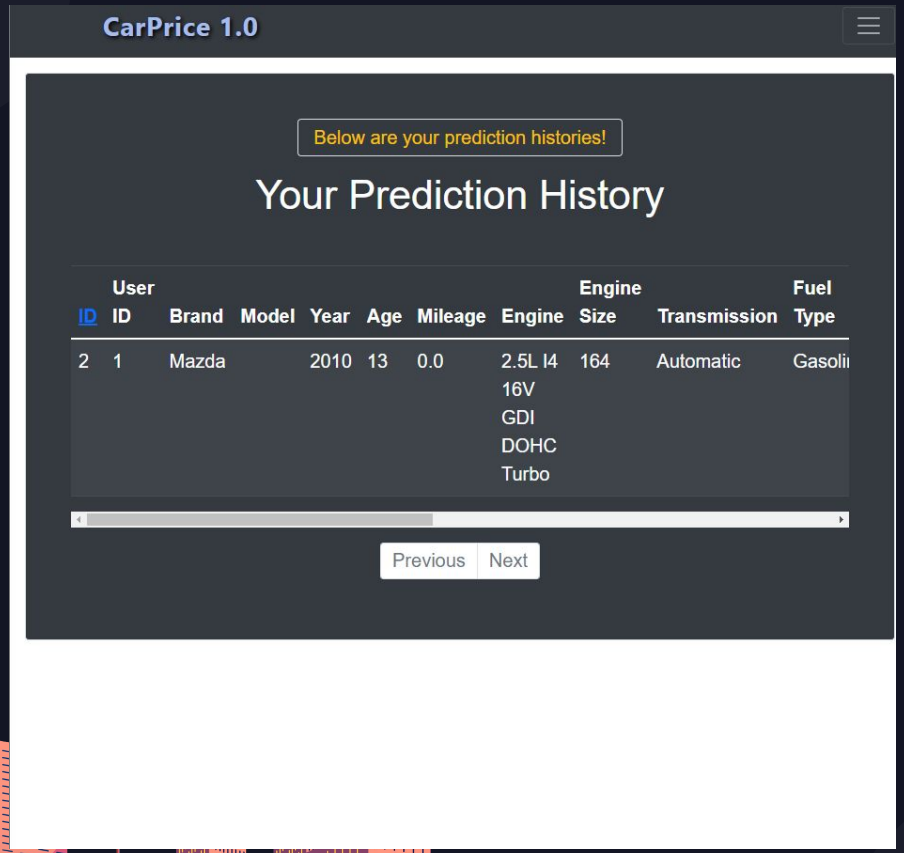
	id	userid	brand	year	age	mileage	engine	engine_size	transmission	fuel_type	drivetrain	min_mpg	damaged
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	2	Honda	1980	43	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
2	2	2	Toyota	2000	23	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
3	3	2	Honda	1980	43	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
4	4	2	Toyota	2000	23	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
5	5	2	Honda	1980	43	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
6	6	2	Toyota	2000	23	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
7	7	23	Toyota	1984	39	0.0	3.0L I6 24V GDI DOHC Turbo	0	Automatic	E85 Flex Fuel	Four-wheel Drive	0.0	No
8	8	2	Honda	1980	43	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
9	9	2	Toyota	2000	23	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
10	10	2	Honda	1980	43	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
11	11	2	Toyota	2000	23	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
12	12	2	Honda	1980	43	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
13	13	2	Toyota	2000	23	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
14	14	2	Honda	1980	43	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
15	15	2	Toyota	2000	23	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
16	16	2	Honda	1980	43	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
17	17	2	Toyota	2000	23	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
18	18	23	Honda	2010	13	0.0	Intercooled Supercharger ...	0	5-Speed Automatic	Gasoline	Front-wheel Drive	44.0	No
19	19	2	Honda	1980	43	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0
20	20	2	Toyota	2000	23	1.0	2.5L I4 16V GDI DOHC Turbo	2	6-Speed Automatic	Gasoline	Four-wheel Drive	3.0	0

Database

1. Creation of Prediction Database

```
# Creating the prediction database to store the data
class Prediction(db.Model):
    id = db.Column(db.Integer, primary_key=True, autoincrement=Tr
    userid = db.Column(db.Integer, db.ForeignKey('User.id') , nul
    brand = db.Column(db.String, nullable=False)
    year = db.Column(db.Integer,nullable=False)
    age = db.Column(db.Integer,nullable=False)
    mileage = db.Column(db.Float,nullable=False)
    engine = db.Column(db.String,nullable=False)
    engine_size = db.Column(db.Integer,nullable=False)
    transmission = db.Column(db.String,nullable=False)
    fuel_type = db.Column(db.String,nullable=False)
    drivetrain = db.Column(db.String,nullable=False)
    min_mpg = db.Column(db.Float,nullable=False)
    damaged = db.Column(db.String,nullable=False)
    turbo = db.Column(db.String,nullable=False)
    navigation_system = db.Column(db.String,nullable=False)
    backup_camera = db.Column(db.String,nullable=False)
    first_owner = db.Column(db.String,nullable=False)
    predicted_on = db.Column(db.DateTime, nullable=False)

    prediction = db.Column(db.Float,nullable=False)
```

Web Page (Database)



ID	User ID	Brand	Model	Year	Age	Mileage	Engine	Engine Size	Transmission	Fuel Type
2	1	Mazda		2010	13	0.0	2.5L I4 16V GDI DOHC Turbo	164	Automatic	Gasoli

1. Prediction History was stored
2. Provide next and back option where each page only show 5 predictions
3. Allow the deletion of records
4. Produced with flask

Validity Testing

1. Test Add Prediction
   a. Provide test for adding prediction
2. Test Entry
   a. Provide accurate in populating to database
3. Test Link
   a. Make sure that links are working
4. Test Client Login Link
   a. Make sure that only client can access this link
5. Test Prediction
   a. Test prediction with logged in client
6. Test Registration
   a. Test registration

Range Testing

1. Test out of range (year)
   a. Check whether it is still able to predict
2. Test out of range (engine size)
   a. Check if it still able to predict
3. Test Missing values
   a. Check whether missing or none input
4. Test Client Login Link
   a. Make sure that only client can access this link
5. Test negative user id
   a. Check whether negative id is able to login

Expected Failure Testing

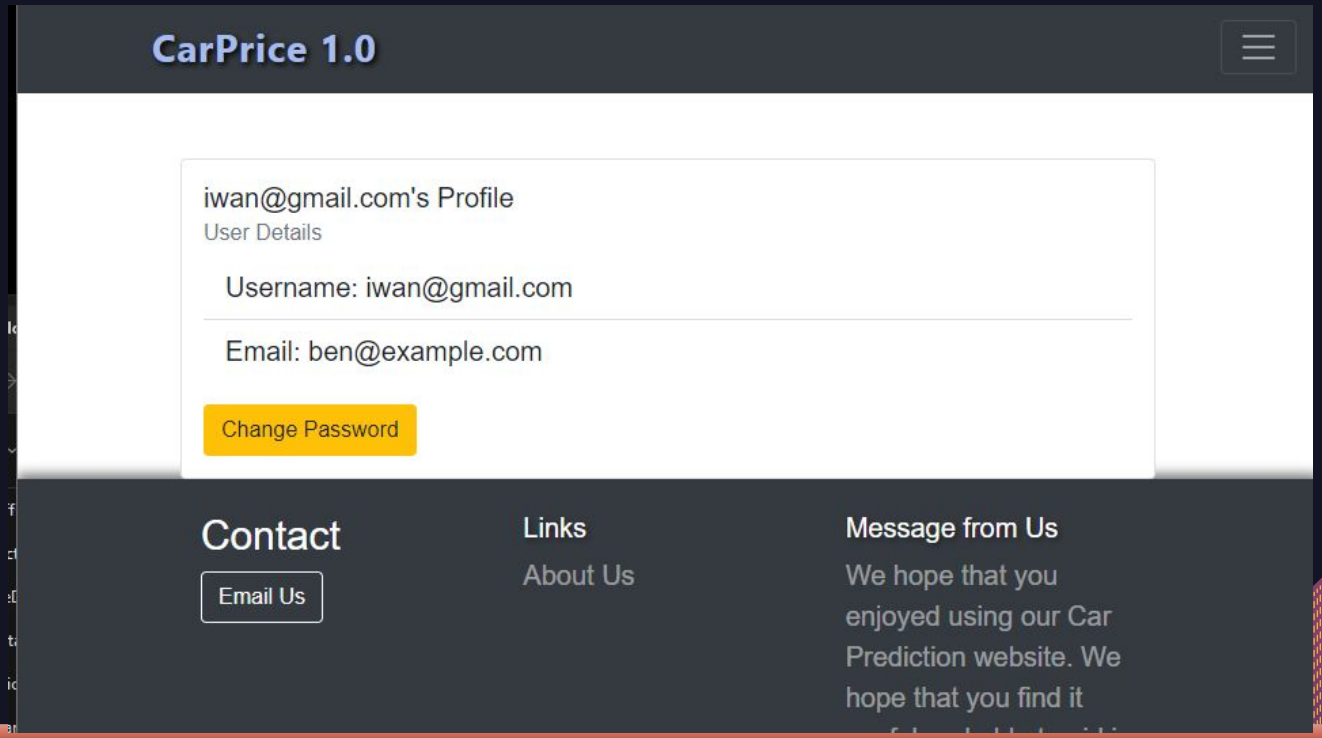
1. Test Missing values
   a. Where there is a missing input
2. Test invalid credentials
   a. Wrong password for login
3. Test duplicate records
   a. When user re register with the same email

Consistency Testing

1. Test Getting multiple prediction with the same perimeter
   a. Check if results are the same

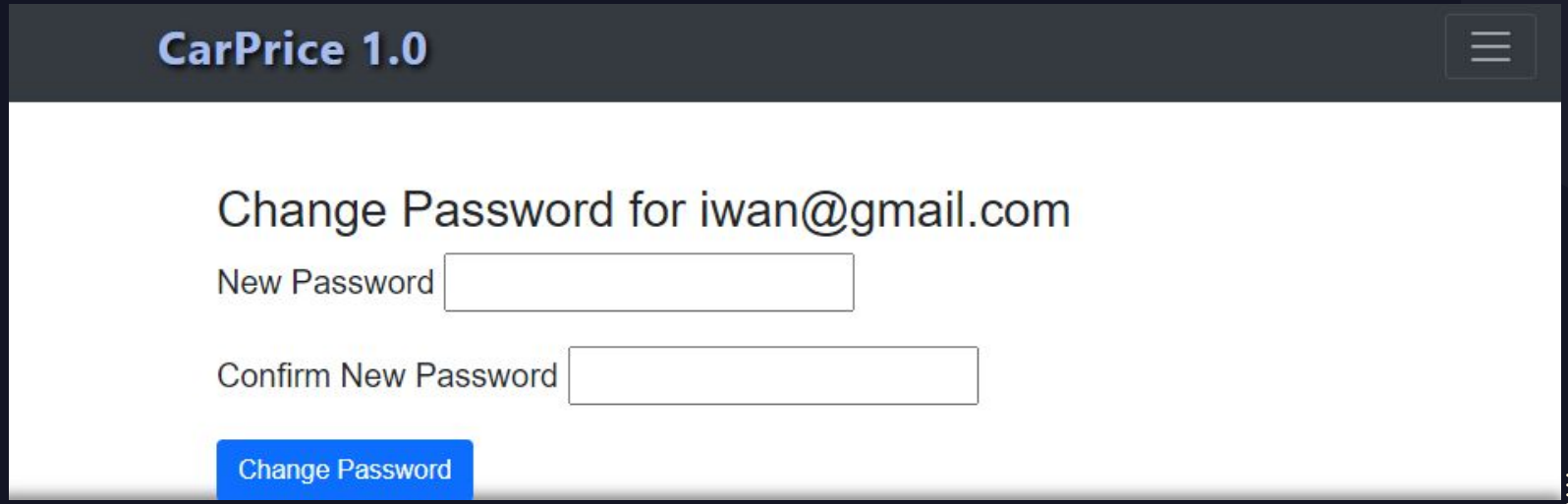
Additional

1. Created a user profile page that enables the change of password

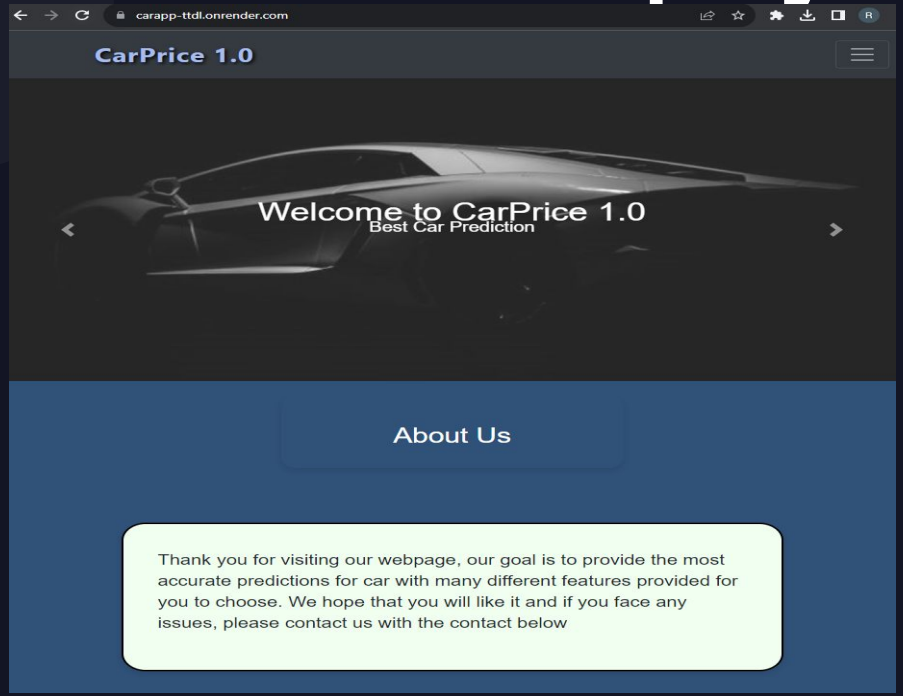


Additional

1. Provision of change password
2. Provision of reset password



Deployed on Render



https://carapp-ttdl.onrender.com/

Thank You