KnowCars – A Car Encyclopedia

Revanth Srinivas Kovuru (22STUCHH010199)

Srujan Jaini (22STUCHH010234)

Easwar Chandra Vidya Sagar Mahanthi (22STUCHH010331)

Department of Computer Science & Engineering

IcfaiTech

Hyderabad

May, 2024

KnowCars – A Car Encyclopedia

Report submitted for Database Management Systems course,

Dr. Rohini Pinapatruni

Revanth Srinivas Kovuru (22STUCHH010199)

Srujan Jaini (22STUCHH010234)

Easwar Chandra Vidya Sagar Mahanthi (22STUCHH010331)

Department of Computer Science & Engineering

IcfaiTech

Hyderabad

May, 2024

Table of Contents

Table of Figures

INTRODUCTION

This report is a documentation of the details of the project KnowCars, a web application built to function as a go to place for any and all information regarding cars. Built by three 2nd year students as a project work to demonstrate the knowledge of database integration to a webapp, every information displayed by KnowCars is fetched from MySQL Database (even the image path).

There are a few websites that display car information, but not all can be found at one place. Our goal is to make most of the information of most car models available at one place. KnowCars currently has information of over 330 models from 40 different manufacturers. All the data is stored in local MySQL server and is fetched to the webapp by Flask-SQLAlchemy. More details about technical dependencies are detailed later in this report.

KnowCars started as a small idea (or a wish) to ease the pain when searching online about cars. One and a half month later, the same idea is a reality with a fully functional, ready to deploy webapp with all the information is built. Let us dive more deeper into the application.

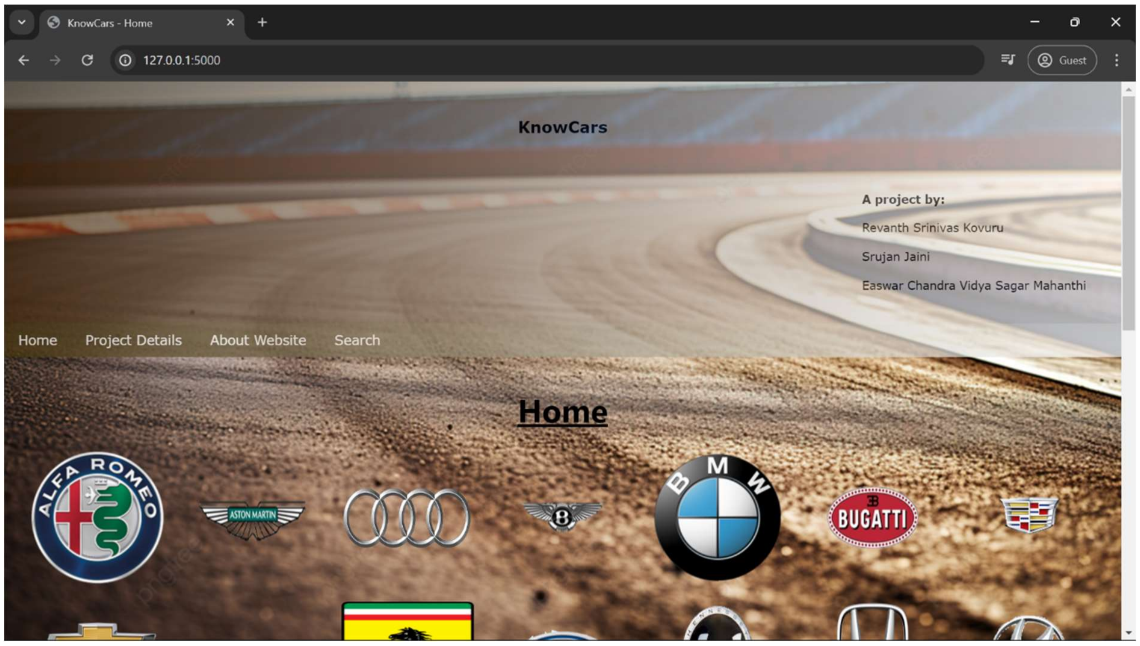


Figure 1: Home page of KnowCars

PROJECT OVERVIEW

Technologies used

1. Flask framework
2. MySQL
3. SQLAlchemy ORM
4. Jinja 2.0
5. W3.CSS
6. HTML
7. CSS (vanilla)
8. Werkzeug for HTTP connections (a module of flask)
9. Flask-WTF

KnowCars webapp is built on Flask. Flask is a lightweight web framework for building web applications in Python. The Flask framework has various components that combine to run our web application. Routing is done through python decorative functions (@app.route('/route')) inside flask.

MySQL database is linked to the Flask application through the SQLAlchemy ORM. ORM is an Object Relational Mapper. It relates python database model classes to the relations (tables) of any SQL database.

Front-end development is done using Jinja Templating on HTML files. Jinja is a sub-package of Flask that adds python programming functionalities to HTML pages. Styling is done through vanilla CSS alongside W3.CSS. W3.CSS is a free, open-source CSS library and is a best alternative to CSS Bootstrap.

The search bar is created using Flask-wtf module from which forms are created. The data is transported between the app and server through flask's internal sub-packages.

Functionalities and Features

KnowCars has a wide range of functionalities and features. The website is easy to navigate through by any person. The Home Page features a header, a navigation bar and clickable image hyperlinks of all the 40 car manufacturing companies (Figure 1). Each image directs the user to a new page where all the models manufactured by the company are displayed. This page also has clickable image hyperlinks that display all the information about that car model.

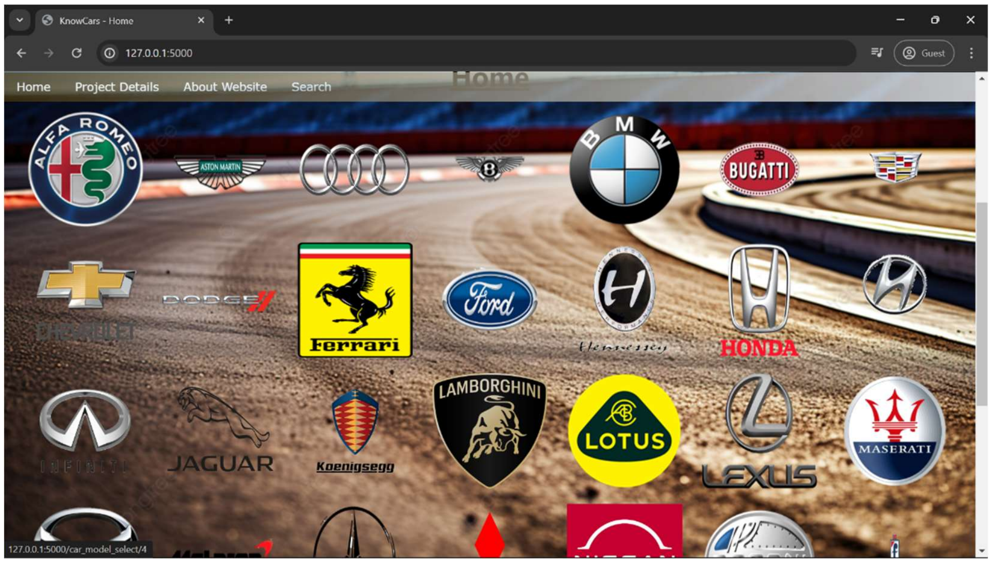


Figure 2: View of clickable images in Home Page

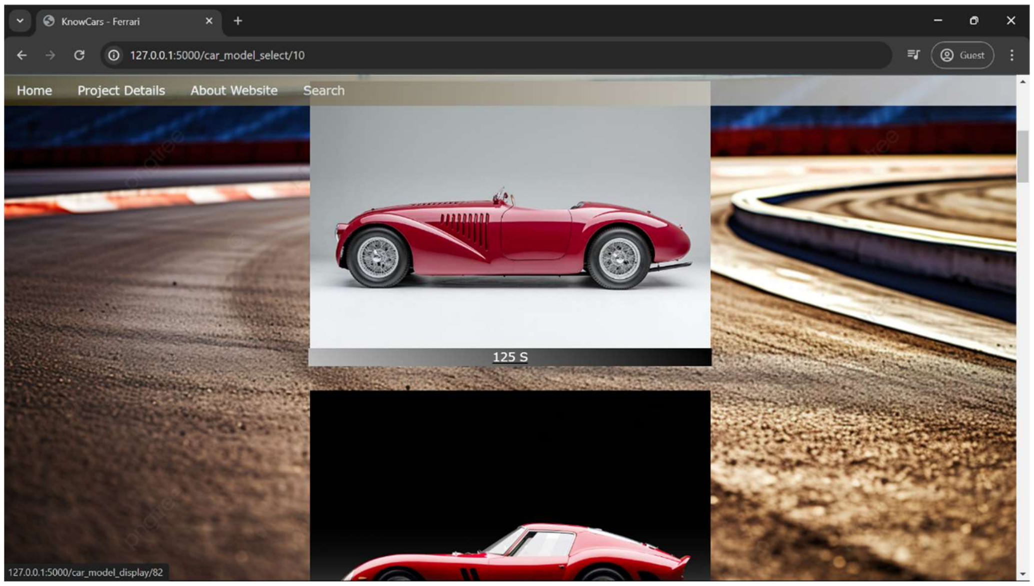

KnowCars - Ferrari
127.0.0.1:5000/car_model_select/10
Home
Project Details
About Website
Search
125 S
127.0.0.1:5000/car_model_display/82

Figure 3: View of models of a manufacturer

The Navigation bar contains link to home page, project details page, about website page and search page. The search page is a great feature of the website. It takes characters as input and searches for matching data in the database. The results are displayed as clickable hyperlinks that redirect to the details of the page.

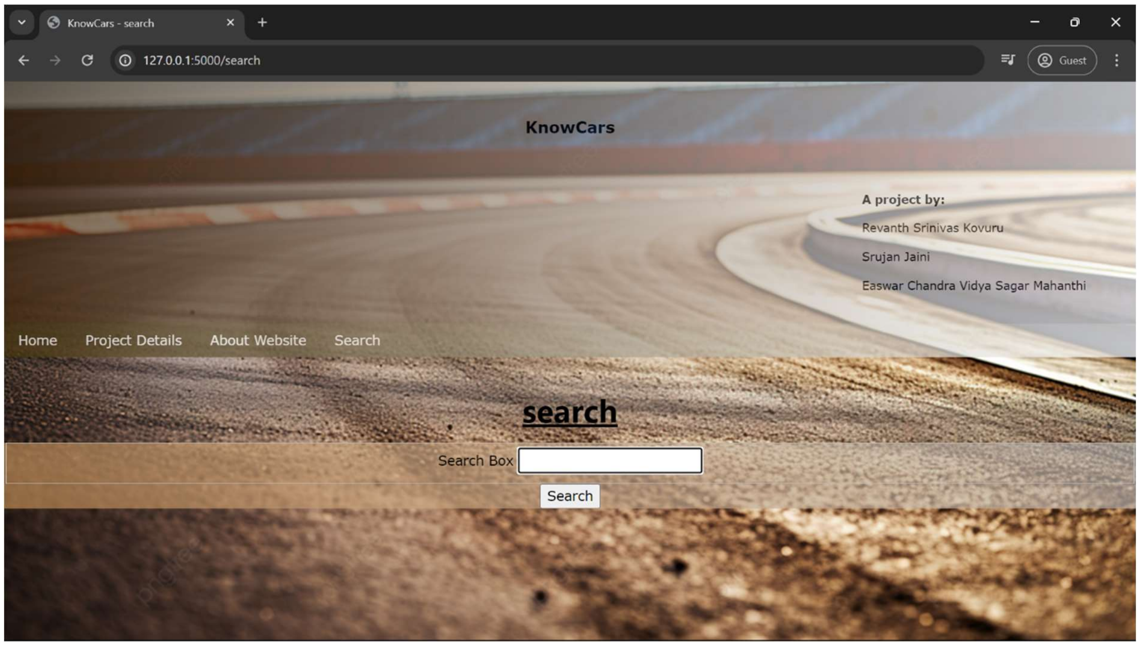
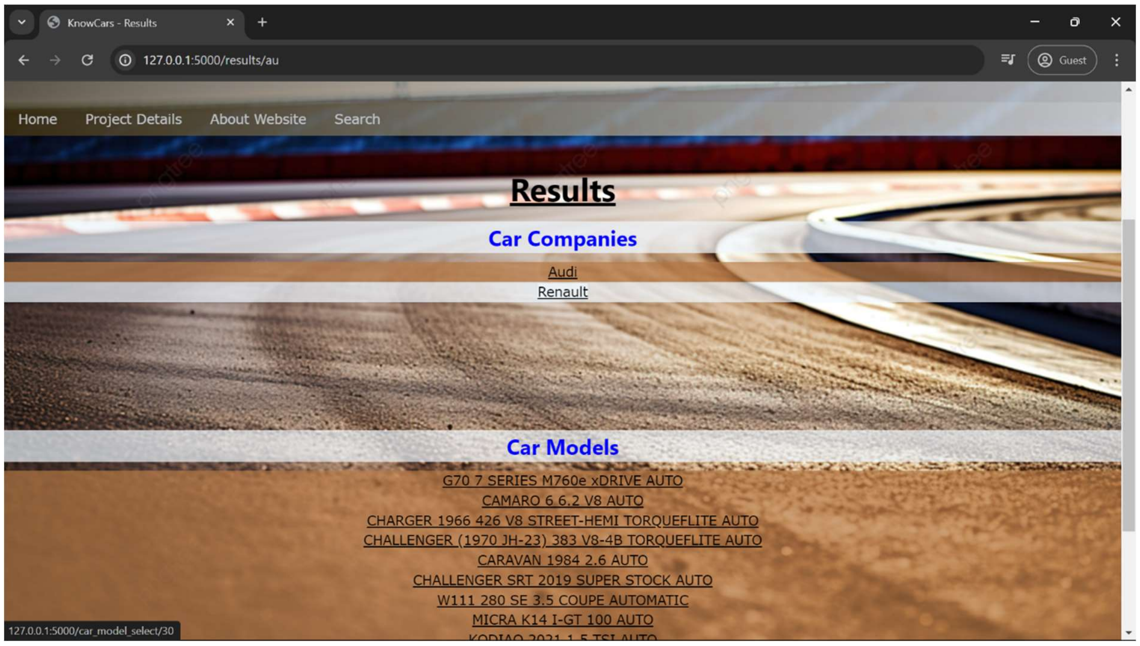


Figure 4: The search bar



Results

Car Companies

Audi
Renault

Car Models

G70 7 SERIES M760e xDRIVE AUTO
CAMARO 6.6.2 V8 AUTO
CHARGER 1966 426 V8 STREET-HEMI TORQUEFLITE AUTO
CHALLENGER (1970 JH-23) 383 V8-4B TORQUEFLITE AUTO
CARAVAN 1984 2.6 AUTO
CHALLENGER SRT 2019 SUPER STOCK AUTO
W111 280 SE 3.5 COUPE AUTOMATIC
MICRA K14 I-GT 100 AUTO
KODIAQ 2021 1.5 TSI AUTO

Figure 5: The search results for 'au' string

These are all the basic functionalities and features of the project. Next chapter will provide details about the system architecture and designs.

SYSTEM ARCHITECTURE

This chapter explains the technology stack used to build the project. The stack is mostly Flask and its sub-modules. A more detailed diagram is illustrated below.

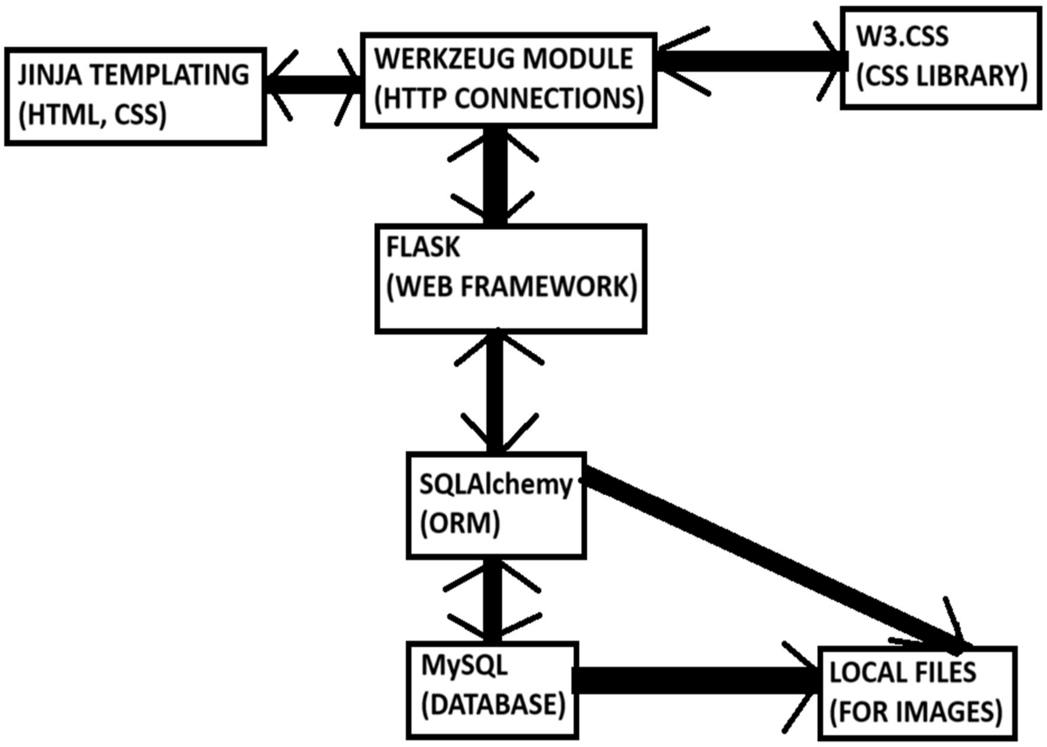


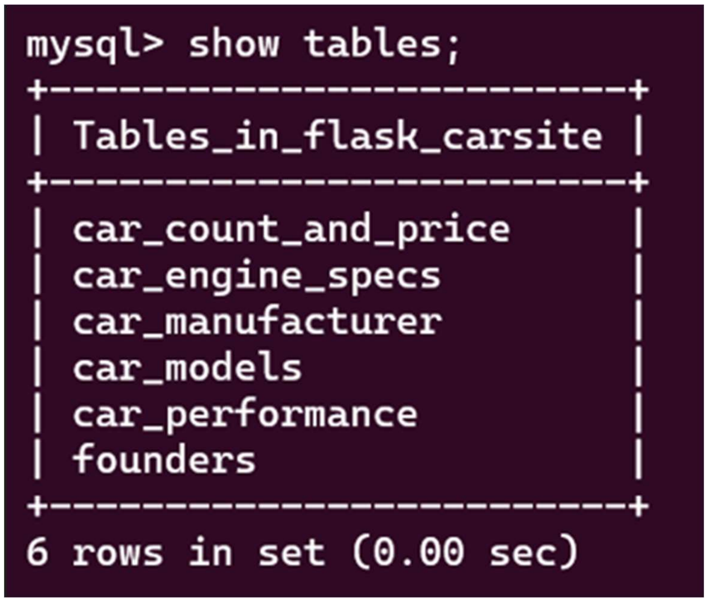
Figure 6: Illustration of Project Architecture

Flask is the major module that controls all the elements of the webapp. The application is not hosted to any server, so every component is locally stored on the device. Images are accessed by storing the path to it in the relations (MySQL tables). These are then fetched by the Werkzeug module through local HTTP requests.

All the webapp's routes are managed directly by Flask and are initiated through HTTP requests. Jinja helps build the skeletal HTML pages for each route which are then populated by data sent in by flask. The data sent by flask is taken in multiple ways, primarily from MySQL database through SQLAlchemy ORM, from previous page via route specification, and from search bar as well.

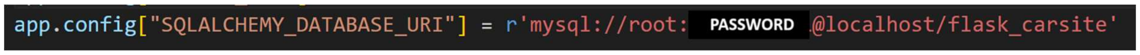
DATABASE DESCRIPTIONS

This chapter describes all the relational models used in the project. Due to the involvement of an ORM, an equivalent class in python is created for every relation in MySQL database. The database is named “flask_carsite” and there are a total of 6 relations in the database Below are the images describing the relations.



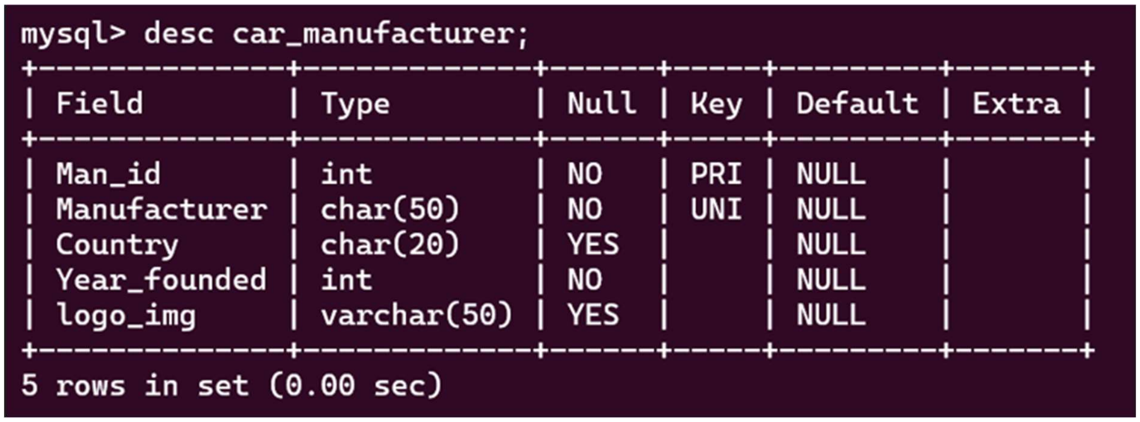
```
mysql> show tables;
+-------------------------+
| Tables_in_flask_carsite |
+-------------------------+
| car_count_and_price     |
| car_engine_specs        |
| car_manufacturer        |
| car_models              |
| car_performance         |
| founders                |
+-------------------------+
6 rows in set (0.00 sec)
```

Figure 7: Tables in MySQL database



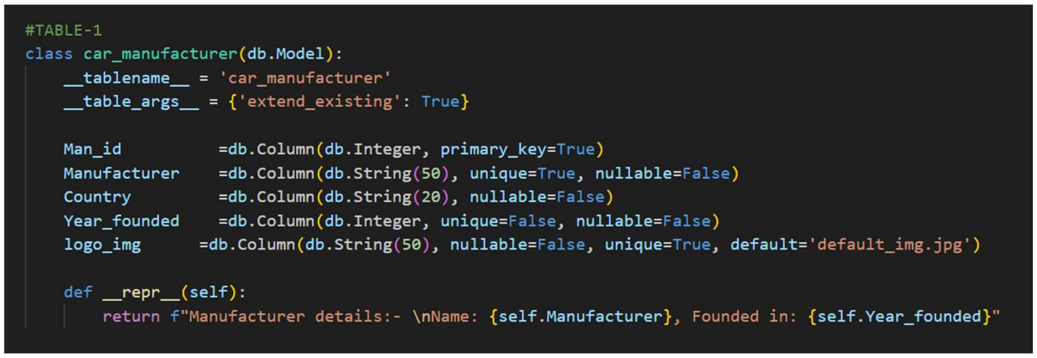
```
app.config["SQLALCHEMY_DATABASE_URI"] = r'mysql://root: PASSWORD @localhost/flask_carsite'
```

Figure 8: Connecting database to the application



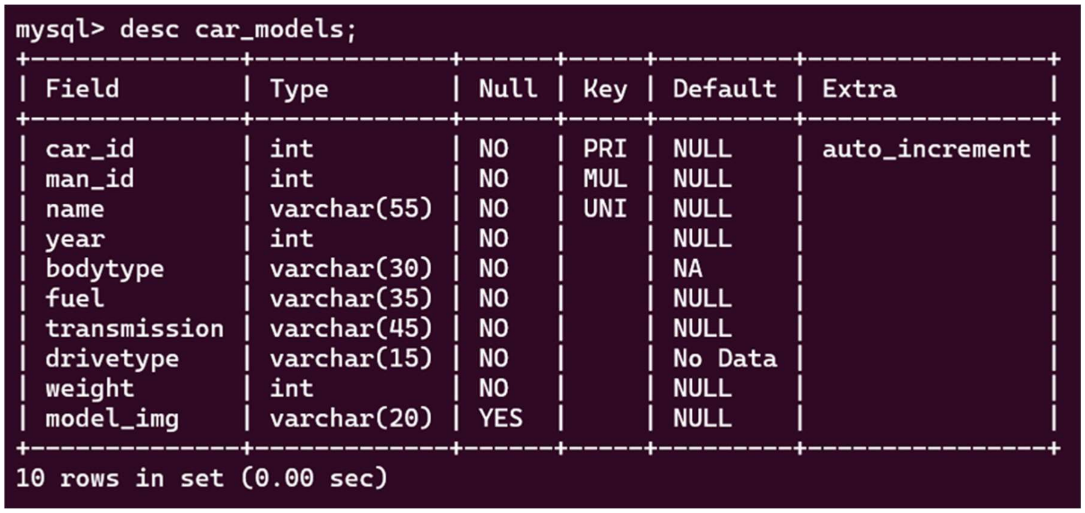
```
mysql> desc car_manufacturer;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Man_id       | int         | NO   | PRI | NULL    |       |
| Manufacturer | char(50)    | NO   | UNI | NULL    |       |
| Country      | char(20)    | YES  |     | NULL    |       |
| Year_founded | int         | NO   |     | NULL    |       |
| logo_img     | varchar(50) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
5 rows in set (0.00 sec)
```

Figure 9: Description of car_manufacturer



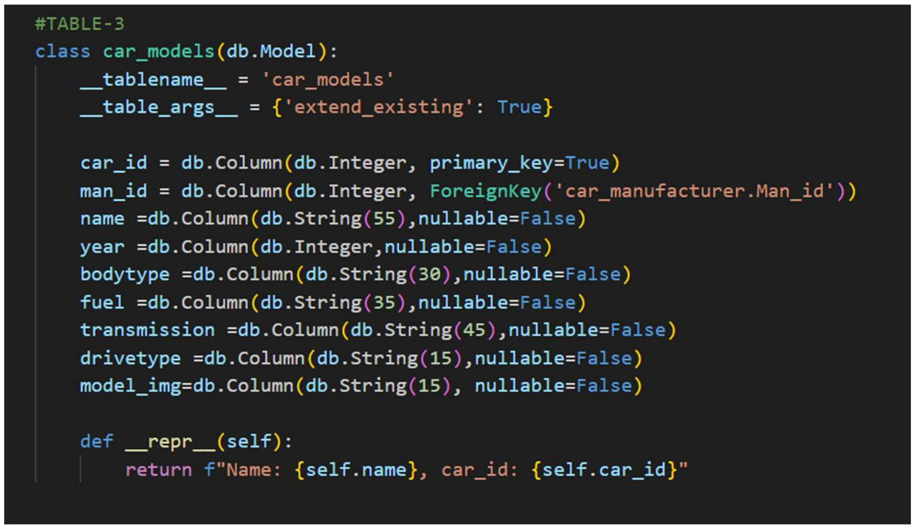
```python
#TABLE-1
class car_manufacturer(db.Model):
    __tablename__ = 'car_manufacturer'
    __table_args__ = {'extend_existing': True}

    Man_id          =db.Column(db.Integer, primary_key=True)
    Manufacturer    =db.Column(db.String(50), unique=True, nullable=False)
    Country         =db.Column(db.String(20), nullable=False)
    Year_founded    =db.Column(db.Integer, unique=False, nullable=False)
    logo_img      =db.Column(db.String(50), nullable=False, unique=True, default='default_img.jpg')

    def __repr__(self):
        return f"Manufacturer details:- \nName: {self.Manufacturer}, Founded in: {self.Year_founded}"
```

Figure 10: Object Representation of Table-1



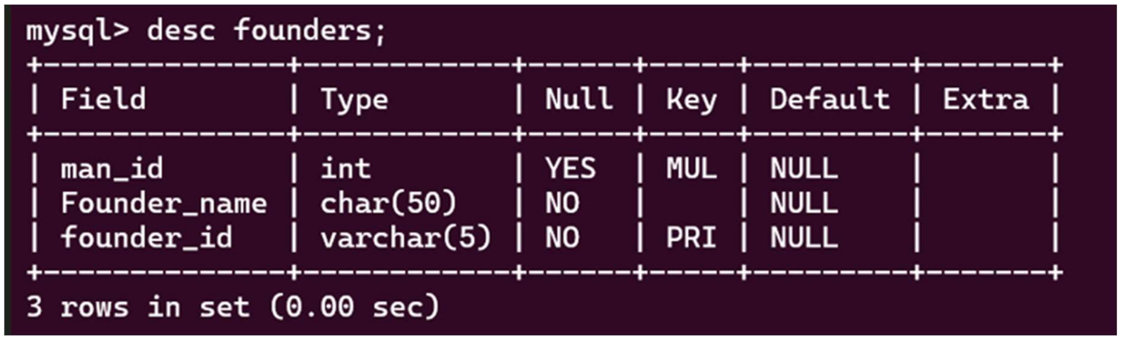
```
mysql> desc car_models;
+--------------+-------------+------+-----+---------+----------------+
| Field        | Type        | Null | Key | Default | Extra          |
+--------------+-------------+------+-----+---------+----------------+
| car_id       | int         | NO   | PRI | NULL    | auto_increment |
| man_id       | int         | NO   | MUL | NULL    |                |
| name         | varchar(55) | NO   | UNI | NULL    |                |
| year         | int         | NO   |     | NULL    |                |
| bodytype     | varchar(30) | NO   |     | NA      |                |
| fuel         | varchar(35) | NO   |     | NULL    |                |
| transmission | varchar(45) | NO   |     | NULL    |                |
| drivetype    | varchar(15) | NO   |     | No Data |                |
| weight       | int         | NO   |     | NULL    |                |
| model_img    | varchar(20) | YES  |     | NULL    |                |
+--------------+-------------+------+-----+---------+----------------+
10 rows in set (0.00 sec)
```

Figure 11: Description of car_models



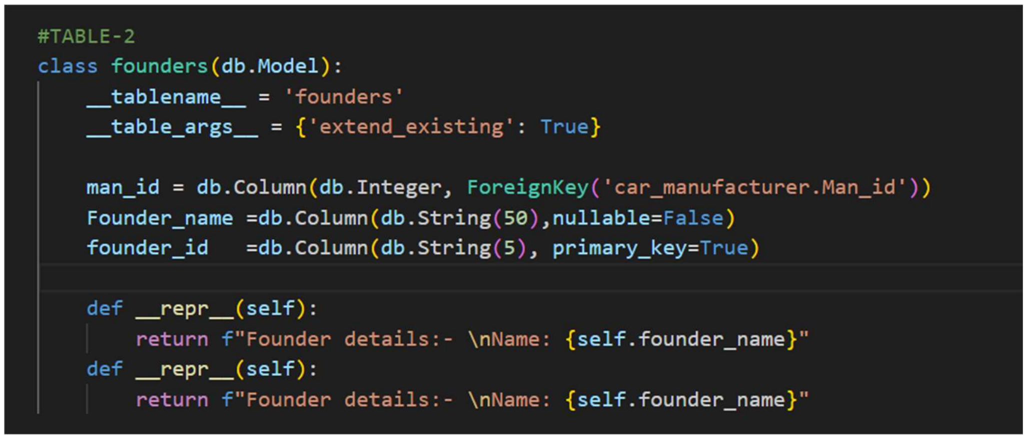
```python
#TABLE-3
class car_models(db.Model):
    __tablename__ = 'car_models'
    __table_args__ = {'extend_existing': True}

    car_id = db.Column(db.Integer, primary_key=True)
    man_id = db.Column(db.Integer, ForeignKey('car_manufacturer.Man_id'))
    name =db.Column(db.String(55),nullable=False)
    year =db.Column(db.Integer,nullable=False)
    bodytype =db.Column(db.String(30),nullable=False)
    fuel =db.Column(db.String(35),nullable=False)
    transmission =db.Column(db.String(45),nullable=False)
    drivetype =db.Column(db.String(15),nullable=False)
    model_img=db.Column(db.String(15), nullable=False)

    def __repr__(self):
        return f"Name: {self.name}, car_id: {self.car_id}"
```

Figure 12: Object Representation of Table-3



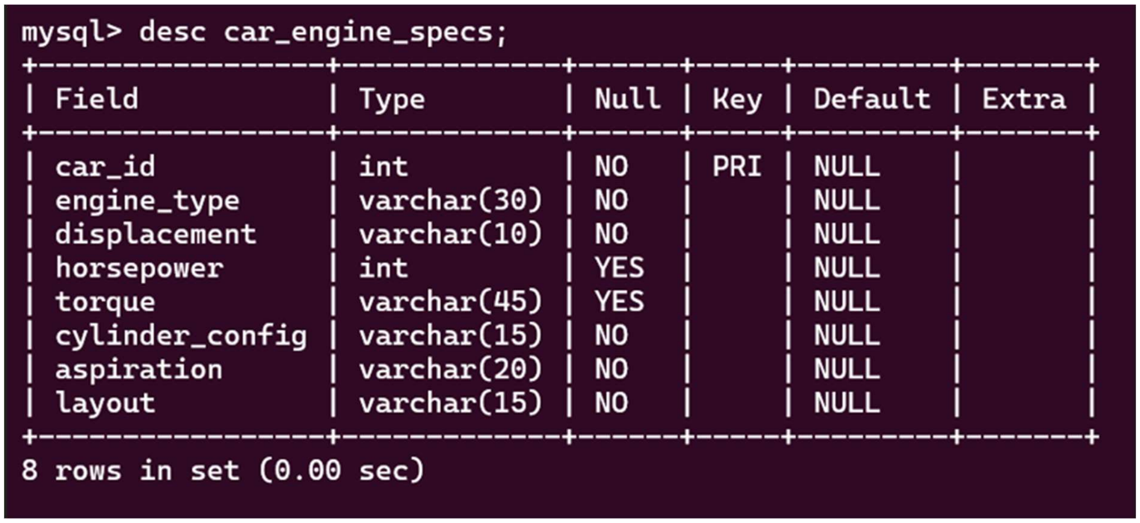
```
mysql> desc founders;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| man_id       | int         | YES  | MUL | NULL    |       |
| Founder_name | char(50)    | NO   |     | NULL    |       |
| founder_id   | varchar(5)  | NO   | PRI | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

Figure 13: Description of founders



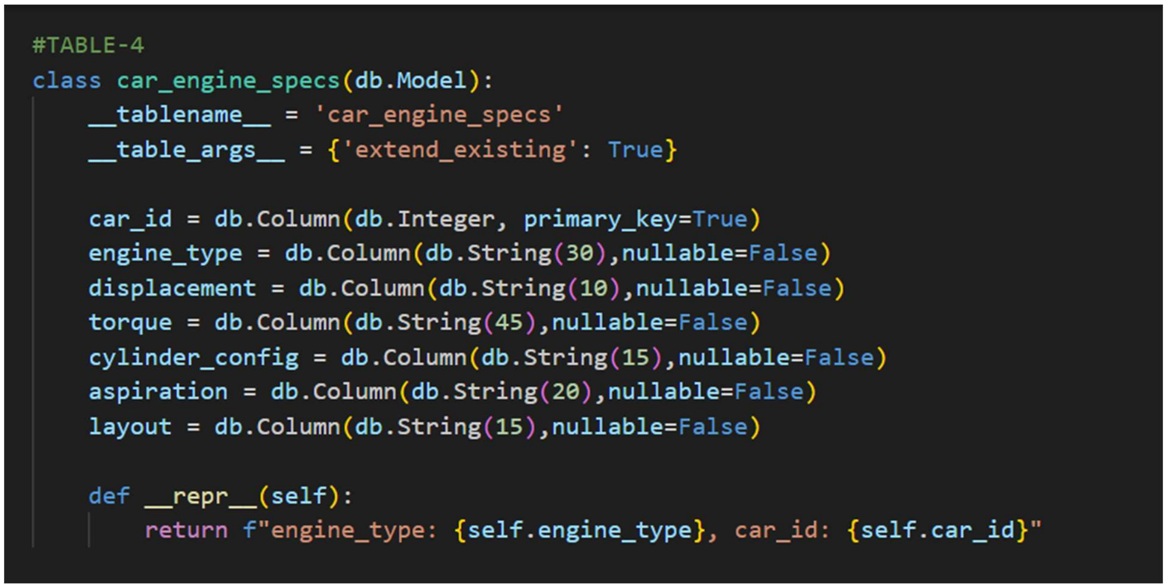
```
#TABLE-2
class founders(db.Model):
    __tablename__ = 'founders'
    __table_args__ = {'extend_existing': True}

    man_id = db.Column(db.Integer, ForeignKey('car_manufacturer.Man_id'))
    Founder_name =db.Column(db.String(50),nullable=False)
    founder_id   =db.Column(db.String(5), primary_key=True)

    def __repr__(self):
        return f"Founder details:- \nName: {self.founder_name}"
    def __repr__(self):
        return f"Founder details:- \nName: {self.founder_name}"
```

Figure 14: Object Representation of Table-2



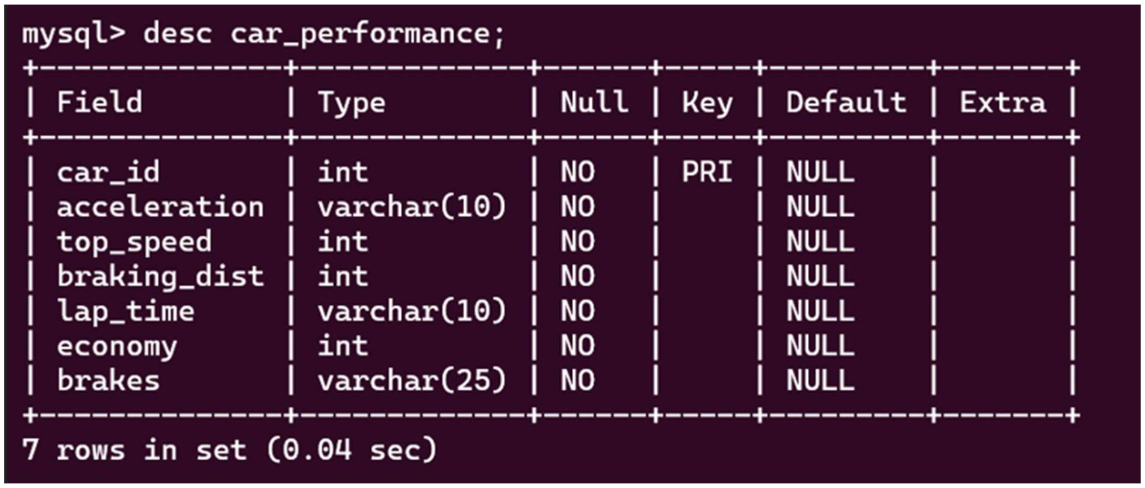
```
mysql> desc car_engine_specs;
+-----------------+-------------+------+-----+---------+-------+
| Field           | Type        | Null | Key | Default | Extra |
+-----------------+-------------+------+-----+---------+-------+
| car_id          | int         | NO   | PRI | NULL    |       |
| engine_type     | varchar(30) | NO   |     | NULL    |       |
| displacement    | varchar(10) | NO   |     | NULL    |       |
| horsepower      | int         | YES  |     | NULL    |       |
| torque          | varchar(45) | YES  |     | NULL    |       |
| cylinder_config | varchar(15) | NO   |     | NULL    |       |
| aspiration      | varchar(20) | NO   |     | NULL    |       |
| layout          | varchar(15) | NO   |     | NULL    |       |
+-----------------+-------------+------+-----+---------+-------+
8 rows in set (0.00 sec)
```

Figure 15: Description of car_engine_specs



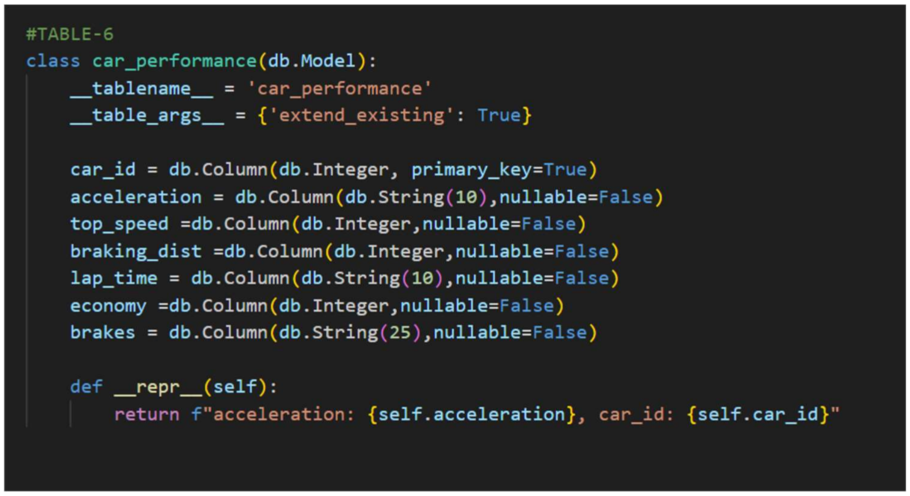
```python
#TABLE-4
class car_engine_specs(db.Model):
    __tablename__ = 'car_engine_specs'
    __table_args__ = {'extend_existing': True}

    car_id = db.Column(db.Integer, primary_key=True)
    engine_type = db.Column(db.String(30),nullable=False)
    displacement = db.Column(db.String(10),nullable=False)
    torque = db.Column(db.String(45),nullable=False)
    cylinder_config = db.Column(db.String(15),nullable=False)
    aspiration = db.Column(db.String(20),nullable=False)
    layout = db.Column(db.String(15),nullable=False)

    def __repr__(self):
        return f"engine_type: {self.engine_type}, car_id: {self.car_id}"
```

Figure 16: Object Representation of Table-4



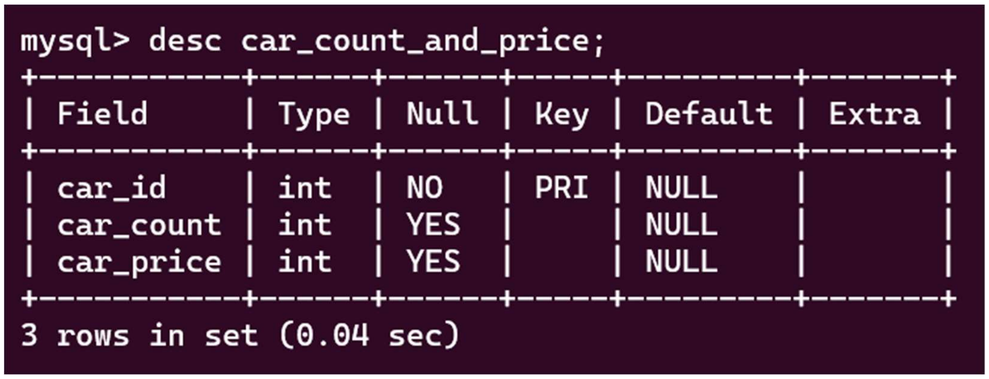
```
mysql> desc car_performance;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| car_id       | int         | NO   | PRI | NULL    |       |
| acceleration | varchar(10) | NO   |     | NULL    |       |
| top_speed    | int         | NO   |     | NULL    |       |
| braking_dist | int         | NO   |     | NULL    |       |
| lap_time     | varchar(10) | NO   |     | NULL    |       |
| economy      | int         | NO   |     | NULL    |       |
| brakes       | varchar(25) | NO   |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
7 rows in set (0.04 sec)
```

Figure 17: Description of car_performance



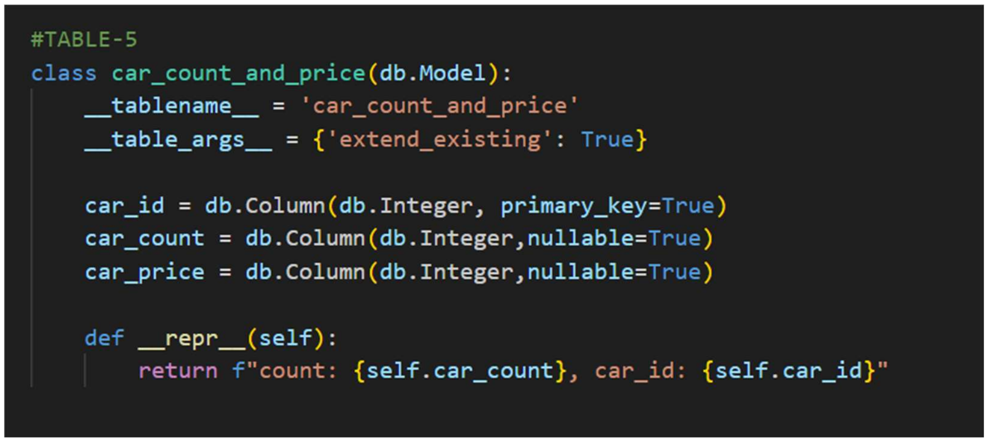
```python
#TABLE-6
class car_performance(db.Model):
    __tablename__ = 'car_performance'
    __table_args__ = {'extend_existing': True}

    car_id = db.Column(db.Integer, primary_key=True)
    acceleration = db.Column(db.String(10),nullable=False)
    top_speed =db.Column(db.Integer,nullable=False)
    braking_dist =db.Column(db.Integer,nullable=False)
    lap_time = db.Column(db.String(10),nullable=False)
    economy =db.Column(db.Integer,nullable=False)
    brakes = db.Column(db.String(25),nullable=False)

    def __repr__(self):
        return f"acceleration: {self.acceleration}, car_id: {self.car_id}"
```

Figure 18: Object Representation of Table-6



```
mysql> desc car_count_and_price;
+-----------+------+------+-----+---------+-------+
| Field     | Type | Null | Key | Default | Extra |
+-----------+------+------+-----+---------+-------+
| car_id    | int  | NO   | PRI | NULL    |       |
| car_count | int  | YES  |     | NULL    |       |
| car_price | int  | YES  |     | NULL    |       |
+-----------+------+------+-----+---------+-------+
3 rows in set (0.04 sec)
```

Figure 19: Description of car_count_and_price



```python
#TABLE-5
class car_count_and_price(db.Model):
    __tablename__ = 'car_count_and_price'
    __table_args__ = {'extend_existing': True}

    car_id = db.Column(db.Integer, primary_key=True)
    car_count = db.Column(db.Integer,nullable=True)
    car_price = db.Column(db.Integer,nullable=True)

    def __repr__(self):
        return f"count: {self.car_count}, car_id: {self.car_id}"
```

Figure 20: Object Representation of Table-5

CONCLUSION

In conclusion, the KnowCars project has successfully demonstrated the integration of a comprehensive database with a user-friendly web application. Through the diligent efforts of the team, we have created a centralized repository of car information that is both accessible and informative. The application's seamless functionality, powered by Flask and MySQL, showcases the practical application of database management systems in real-world projects.

The journey from a simple idea to a fully functional web application has been both challenging and rewarding. The KnowCars platform now hosts detailed information on over 330 car models from 40 different manufacturers, making it a valuable resource for car enthusiasts and potential buyers alike.

We extend our gratitude to Dr. Rohini Pinapatruni for her guidance throughout this project and to IcfaiTech for providing us with the opportunity to apply our knowledge in a meaningful way. This project has been a testament to the power of collaboration, dedication, and the pursuit of knowledge.