T1A3 – Terminal Application
Horse Stable Management App

Purpose

- [illegible]
- [illegible]
- [illegible]

Feature 1: Main Menu

src > main.py > ...

```python
import os
import horse_manager
import race_logs_manager

# Main menu for application
def main():
    os.system('clear')  # Clears console when called
    while True:
        print("-- Horse Stable Management App --")
        print("1. Add horse")
        print("2. Remove horse")
        print("3. Update horse details")
        print("4. View horse details")
        print("5. Add race results")
        print("6. View race logs")
        print("7. Exit app")

        choice = input("Choose an option: ")

        if choice == '1':
            horse_manager.add_horse()
        elif choice == '2':
            horse_manager.remove_horse()
        elif choice == '3':
            horse_manager.update_horse_details()
        elif choice == '4':
            horse_manager.view_all_horses()
        elif choice == '5':
            race_logs_manager.add_race_log()
        elif choice == '6':
            race_logs_manager.view_race_logs()
        elif choice.lower() == '7':
            print("Goodbye!")
            exit()
        else:
            print("Invalid choice, please try again!")

if __name__ == "__main__":
    main()
```

- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]

Feature 2: Add a horse

src > horse_manager.py > ...

```python
import json
from utility import input_int
from validator import validate_horse

# Add new horse to the database
def add_horse():
    name = input("Enter horse name: ")
    age = input_int("Enter horse age: ")
    health = input("Enter horse health status: ")
    diet = input("Enter horse diet: ")
    new_horse = {"Name": name, "Age": age, "Health": health, "Diet": diet}

# Validate the data entry from the user to ensure that it matches
# against the specified schema, else return error message
    if validate_horse(new_horse):
        with open('data.json', 'r+') as file:
            data = json.load(file)
            data['horses'].append(new_horse)
            file.seek(0)
            file.truncate()
            json.dump(data, file)
    else:
        print("Invalid horse data")
```

- [illegible]
- [illegible]
- [illegible]

Feature 3: Remove a horse

```python
# Remove a horse from the database after checking the database
# for a matching horse name, else return error message
def remove_horse():
    horse_name = input("Enter the name of the horse to remove: ")
    with open('data.json', 'r+') as file:
        data = json.load(file)
        horses = data['horses']
        for horse in horses:
            if horse['Name'] == horse_name:
                horses.remove(horse)
                file.seek(0)
                file.truncate()
                json.dump(data, file)
                print(f"Horse {horse_name} removed successfully!")
                return
        print(f"No horse found with the name {horse_name}.")
```

- [illegible]
- [illegible]
- [illegible]
- [illegible]

Feature 4: Update horse details

```python
# Provide option to update horse details in the database
# which will overwrite existing informatiom if match found
def update_horse_details():
    horse_name = input("Enter the name of the horse to update: ")
    with open('data.json', 'r+') as file:
        data = json.load(file)
        horses = data['horses']
        for horse in horses:
            if horse['Name'] == horse_name:
                print("Enter new details (or press Enter to skip updating):")
                name = input(f"Name (current: {horse['Name']}): ")
                age = input_int(f"Age (current: {horse['Age']}): ")
                health = input(f"Health (current: {horse['Health']}): ")
                diet = input(f"Diet (current: {horse['Diet']}): ")

                if name: horse['Name'] = name
                if age: horse['Age'] = int(age)
                if health: horse['Health'] = health
                if diet: horse['Diet'] = diet

                file.seek(0)
                file.truncate()
                json.dump(data, file)
                print(f"Horse {horse_name} details updated successfully!")
                return
        print(f"No horse found with the name {horse_name}.")
```

- [illegible]
- [illegible]
- [illegible]
- [illegible]

Feature 5: View horse details

```python
# Option for user to view all horses stored in the database
def view_all_horses():
    with open('data.json', 'r') as file:
        data = json.load(file)
        for horse in data['horses']:
            print(horse)
```

- [illegible]
- [illegible]
- [illegible]

Feature 6: Add race results

src › race_logs_manager.py › ...

```python
import json
from utility import input_date
from validator import validate_race_log

# Allow user to add race results which will be stored in json file
def add_race_log():
    horse_name = input("Enter horse name: ")
    race_date_obj = input_date("Enter race date (DD-MM-YYYY): ")  # Returns a datetime.date object
    race_date = race_date_obj.strftime('%d-%m-%Y')  # Converts the datetime.date object to a string
    race_result = input("Enter race result: ")

    new_log = {"HorseName": horse_name, "RaceDate": race_date, "RaceResult": race_result}

# Validate the data entry from the user to ensure that it matches
# against the specified schema, else return error message
    if validate_race_log(new_log):
        with open('data.json', 'r+') as file:
            data = json.load(file)
            data['race_logs'].append(new_log)
            file.seek(0)
            file.truncate()
            json.dump(data, file)
    else:
        print("Invalid race log")
```

- [redacted]
- [redacted]
- [redacted]
- [redacted]
- [redacted]

Feature 7: View race logs

```python
# Display race logs to user from the json file database
def view_race_logs():
    with open('data.json', 'r') as file:
        data = json.load(file)
        for log in data['race_logs']:
            print(log)
```

- [illegible]
- [illegible]
- [illegible]

Development/ Implementation

For project management I utilised the Trello's online platform. I created a workspace to input all the tasks that I needed to do to complete the assignment. Tasks were placed in either the To-Do, doing, next up, or end tasks section and was updated regularly. Under the define features section of the trello board I worked through each feature by creating a checklist of tasks required to be completed to have the feature implemented. Once all features were completed, I was able to move the status of feature implementation into the 'Done' section of the Trello board.

Challenges

- [illegible]
  - [illegible]
    - [illegible]
  - [illegible]

Favorite parts

- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]

Ethical Issues

Thank you