Snake Cage Renting Application on MongoDB

A PROJECT REPORT

Submitted in partial fulfilment for the award of the degree of

M.Tech

in

Big Data Analytics

by

Bhargavi Sanadhya 18MCB1003
Shrikant Patro 18MCB1009
Divyansh Gupta 18MCB1015

Under the Guidance of

Dr. Sivagami M
Associate Professor

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING

VIT UNIVERSITY

Introduction

Mongodb is one of the most popular and existing database technology around. On other hand, python is also one of the most popular and fastest growing language. There are two technologies work together to get the application. While creating this application, we mainly keep our focus on three things that are how a document databases make working with the schemas easier and modelling the data easier and add performance to application and other is how to design things.

Here we used a Mongo ODM that is Object Document Mapper which is the core objects in mongoid any any object that is to be persisted to the database must include mongoid::Document. The representation of document in MongoDB is BSON object that is very similar to a JSON object. Documents can be stored in their own collections in thwe database or embedded in other document n level deep.

The application that we build a Snake business application this allowing the user when user is travelling with their snakes that are pet snakes. User need not to live out the snake in the car. The user wants a proper cage for snake to live in so they can feel happy. Now the aim of application is the owners of cages up for rent and pet owners, snake owners who are travelling with their pets can took it on rent for that particular days of travelling.

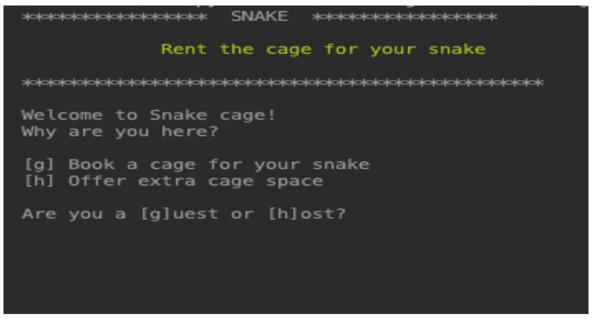
Technology Used

1. MongoDB
2. Python 3.6
   - pymongo
   - mongoengine
   - tqdm
   - colorama
   - python-dateutil
3. PyCharm Platform

Registering Connection with MongoDB

```
def global_init():
    mongoengine.register_connection(alias='core', name='snake_bnb')

def main():
    mongo_setup.global_init()
```

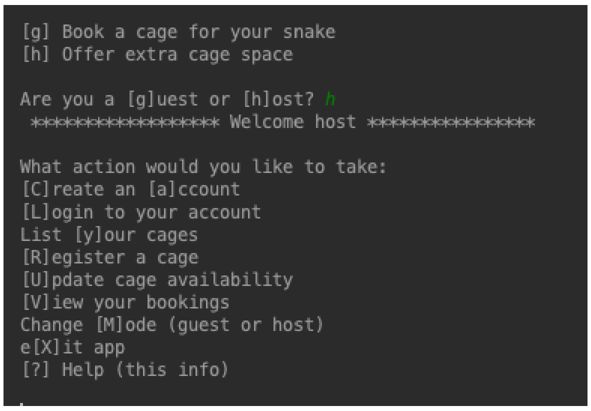
Different Modules in Application



```
****************  SNAKE  ****************

           Rent the cage for your snake

*********************************************

Welcome to Snake cage!
Why are you here?

[g] Book a cage for your snake
[h] Offer extra cage space

Are you a [g]uest or [h]ost?
```

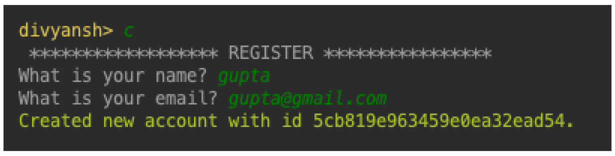
Guest access are for those who want to book the cage for their pet snakes while host access is for those who are willing to rent their cages.

Host Access Features



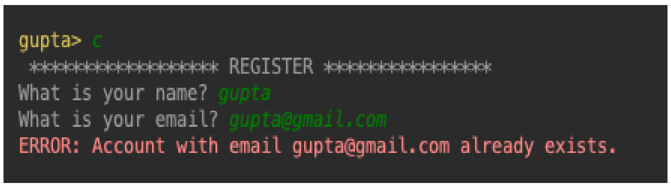
```
[g] Book a cage for your snake
[h] Offer extra cage space

Are you a [g]uest or [h]ost? h
 ****************** Welcome host ****************

What action would you like to take:
[C]reate an [a]ccount
[L]ogin to your account
List [y]our cages
[R]egister a cage
[U]pdate cage availability
[V]iew your bookings
Change [M]ode (guest or host)
e[X]it app
[?] Help (this info)
```

Create an account as seller



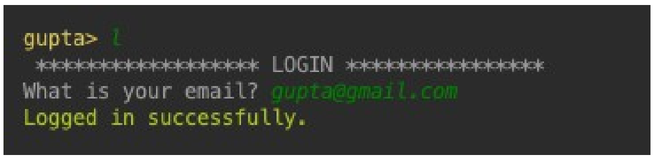
```
divyansh> c
 ****************** REGISTER ****************
What is your name? gupta
What is your email? gupta@gmail.com
Created new account with id 5cb819e963459e0ea32ead54.
```

Detect the duplicate account registering



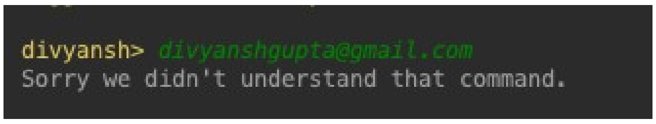
```
gupta> c
 ****************** REGISTER ****************
What is your name? gupta
What is your email? gupta@gmail.com
ERROR: Account with email gupta@gmail.com already exists.
```

Login as host



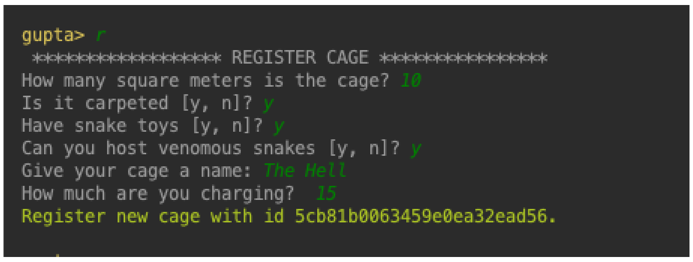
```
gupta> l
 ****************** LOGIN ****************
What is your email? gupta@gmail.com
Logged in successfully.
```

Restricting the unregister account

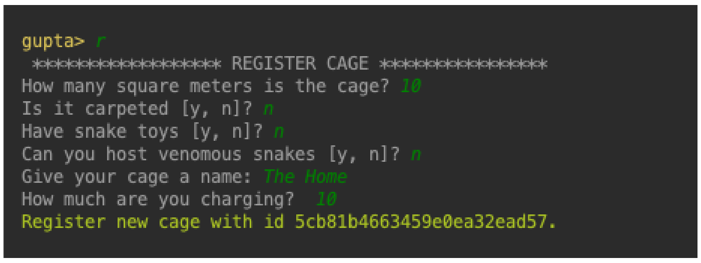


```
divyansh> divyanshgupta@gmail.com
Sorry we didn't understand that command.
```

Register the cage for renting

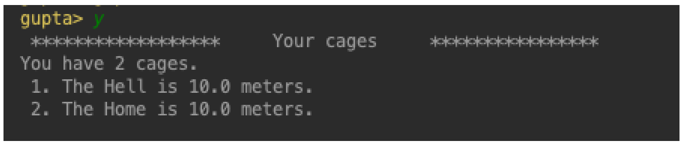


```
gupta> r
 ****************** REGISTER CAGE ****************
How many square meters is the cage? 10
Is it carpeted [y, n]? y
Have snake toys [y, n]? y
Can you host venomous snakes [y, n]? y
Give your cage a name: The Hell
How much are you charging?  15
Register new cage with id 5cb81b0063459e0ea32ead56.
```



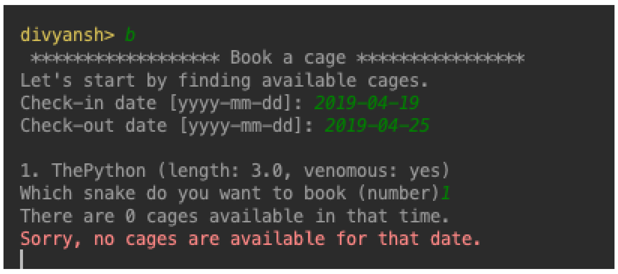
```
gupta> r
 ****************** REGISTER CAGE ****************
How many square meters is the cage? 10
Is it carpeted [y, n]? n
Have snake toys [y, n]? n
Can you host venomous snakes [y, n]? n
Give your cage a name: The Home
How much are you charging?  10
Register new cage with id 5cb81b4663459e0ea32ead57.
```

Listed cages



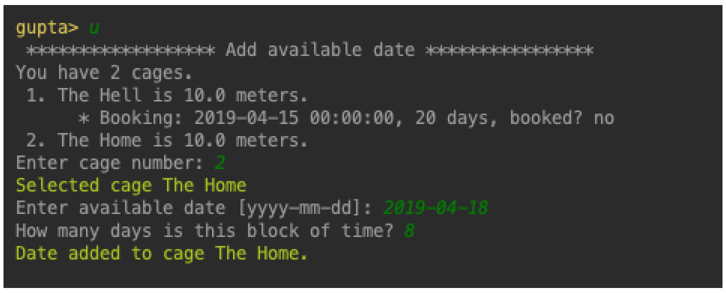
```
gupta> y
 ******************     Your cages     ****************
You have 2 cages.
 1. The Hell is 10.0 meters.
 2. The Home is 10.0 meters.
```

Booking of Cage



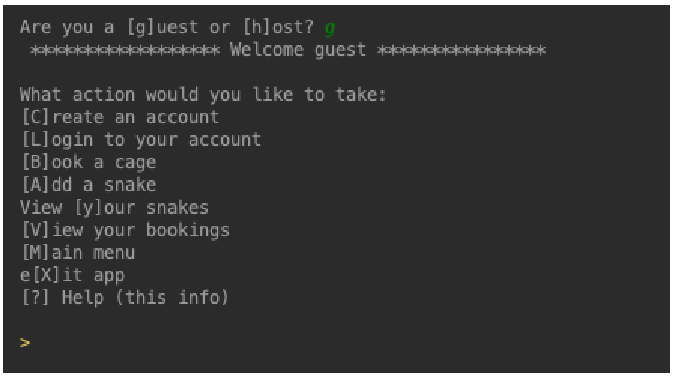
```
divyansh> b
 ****************** Book a cage ****************
Let's start by finding available cages.
Check-in date [yyyy-mm-dd]: 2019-04-19
Check-out date [yyyy-mm-dd]: 2019-04-25

1. ThePython (length: 3.0, venomous: yes)
Which snake do you want to book (number)1
There are 0 cages available in that time.
Sorry, no cages are available for that date.
```

Updating the cage availability



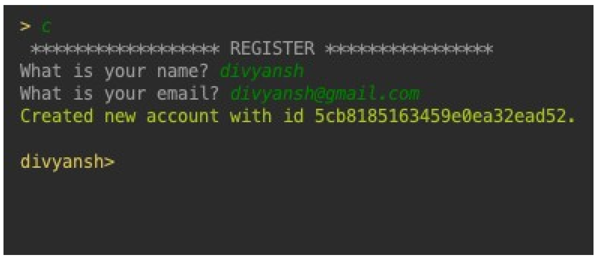
```
gupta> u
 ****************** Add available date ****************
You have 2 cages.
 1. The Hell is 10.0 meters.
      * Booking: 2019-04-15 00:00:00, 20 days, booked? no
 2. The Home is 10.0 meters.
Enter cage number: 2
Selected cage The Home
Enter available date [yyyy-mm-dd]: 2019-04-18
How many days is this block of time? 8
Date added to cage The Home.
```

Guest Access Features



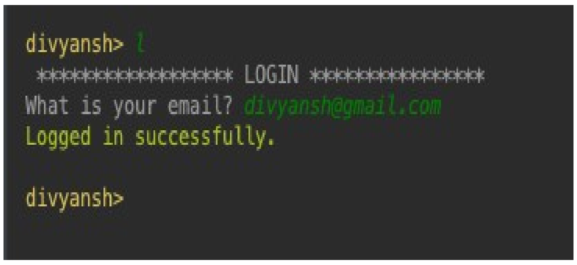
```
Are you a [g]uest or [h]ost? g
 ****************** Welcome guest ****************

What action would you like to take:
[C]reate an account
[L]ogin to your account
[B]ook a cage
[A]dd a snake
View [y]our snakes
[V]iew your bookings
[M]ain menu
e[X]it app
[?] Help (this info)

>
```

Creating an account as Customer



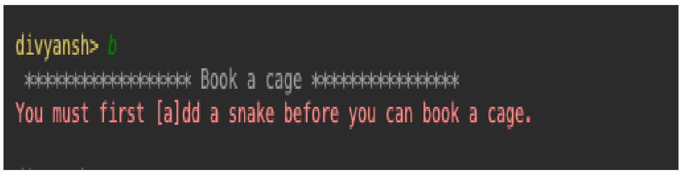
```
> c
 ****************** REGISTER ****************
What is your name? divyansh
What is your email? divyansh@gmail.com
Created new account with id 5cb8185163459e0ea32ead52.

divyansh>
```

Login to the created account



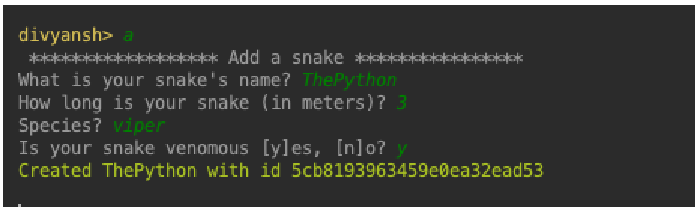
```
divyansh> l
 ****************** LOGIN ****************
What is your email? divyansh@gmail.com
Logged in successfully.

divyansh>
```

Book a cage



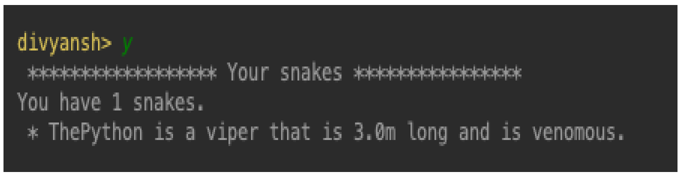
```
divyansh> b
 ***************** Book a cage ***************
You must first [a]dd a snake before you can book a cage.
```

Add a snake details



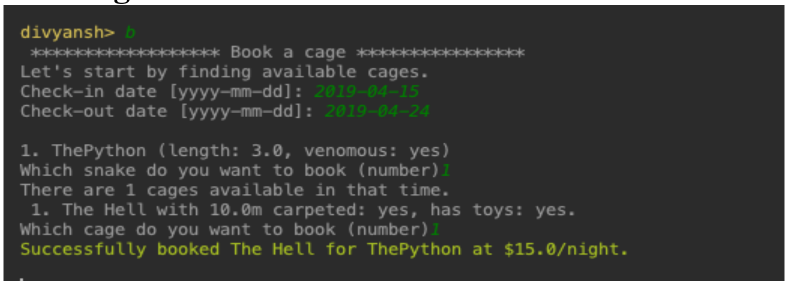
```
divyansh> a
 ****************** Add a snake ****************
What is your snake's name? ThePython
How long is your snake (in meters)? 3
Species? viper
Is your snake venomous [y]es, [n]o? y
Created ThePython with id 5cb8193963459e0ea32ead53
```

View the details of Snakes added by customer



```
divyansh> y
 ****************** Your snakes ****************
You have 1 snakes.
 * ThePython is a viper that is 3.0m long and is venomous.
```

Booking done for the Cages



```
divyansh> b
 ****************** Book a cage ****************
Let's start by finding available cages.
Check-in date [yyyy-mm-dd]: 2019-04-15
Check-out date [yyyy-mm-dd]: 2019-04-24

1. ThePython (length: 3.0, venomous: yes)
Which snake do you want to book (number)1
There are 1 cages available in that time.
 1. The Hell with 10.0m carpeted: yes, has toys: yes.
Which cage do you want to book (number)1
Successfully booked The Hell for ThePython at $15.0/night.
```

Code:

MongoDB Collection for booking module.

```
import mongoengine
class Booking(mongoengine.EmbeddedDocument):
    guest_owner_id = mongoengine.ObjectIdField()
    guest_snake_id = mongoengine.ObjectIdField()

    booked_date = mongoengine.DateTimeField()
    check_in_date = mongoengine.DateTimeField(required=True)
    check_out_date = mongoengine.DateTimeField(required=True)

    review = mongoengine.StringField()
    rating = mongoengine.IntField(default=0)

    @property
    def duration_in_days(self):
        dt = self.check_out_date - self.check_in_date
        return dt.days
```

MongoDB Collection for Cage module.

```
import datetime
import mongoengine

from data.bookings import Booking


class Cage(mongoengine.Document):
    registered_date = mongoengine.DateTimeField(default=datetime.datetime.now)

    name = mongoengine.StringField(required=True)
    price = mongoengine.FloatField(required=True)
    square_meters = mongoengine.FloatField(required=True)
    is_carpeted = mongoengine.BooleanField(required=True)
    has_toys = mongoengine.BooleanField(required=True)
    allow_dangerous_snakes = mongoengine.BooleanField(default=False)

    bookings = mongoengine.EmbeddedDocumentListField(Booking)

    meta = {
        'db_alias': 'core',
        'collection': 'cages'
    }
```

MongoDB setup connection

```
import mongoengine
```

```
def global_init():
    mongoengine.register_connection(alias='core', name='snake_bnb')
```

MongoDB collection for owners details

```
import datetime
import mongoengine

class Owner(mongoengine.Document):
    registered_date = mongoengine.DateTimeField(default=datetime.datetime.now)
    name = mongoengine.StringField(required=True)
    email = mongoengine.StringField(required=True)

    snake_ids = mongoengine.ListField()
    cage_ids = mongoengine.ListField()

    meta = {
        'db_alias': 'core',
        'collection': 'owners'
    }
```

MongoDB collection for Snakes details

```
import datetime
import mongoengine


class Snake(mongoengine.Document):
    registered_date = mongoengine.DateTimeField(default=datetime.datetime.now)
    species = mongoengine.StringField(required=True)

    length = mongoengine.FloatField(required=True)
    name = mongoengine.StringField(required=True)
    is_venomous = mongoengine.BooleanField(required=True)

    meta = {
        'db_alias': 'core',
        'collection': 'snakes'
    }
```

Main Function

```
from colorama import Fore
import program_guests
import program_hosts
import data.mongo_setup as mongo_setup

def main():
   mongo_setup.global_init()

   print_header()

   try:
      while True:
         if find_user_intent() == 'book':
            program_guests.run()
         else:
            program_hosts.run()
   except KeyboardInterrupt:
      return


def print_header():
   snake = \
      """
         Rent the cage for your snake
            """

   print(Fore.WHITE + '****************  SNAKE  ****************')
   print(Fore.GREEN + snake)
   print(Fore.WHITE + '*********************************************')
   print()
   print("Welcome to Snake cage!")
   print("Why are you here?")
   print()


def find_user_intent():
   print("[g] Book a cage for your snake")
   print("[h] Offer extra cage space")
   print()
   choice = input("Are you a [g]uest or [h]ost? ")
   if choice == 'h':
      return 'offer'

   return 'book'
if __name__ == '__main__':
   main()
```

For Guest Access

```
import datetime
```

```
from dateutil import parser

from infrastructure.switchlang import switch
import program_hosts as hosts
import services.data_service as svc
from program_hosts import success_msg, error_msg
import infrastructure.state as state


def run():
    print(' ****************** Welcome guest **************** ')
    print()

    show_commands()

    while True:
        action = hosts.get_action()

        with switch(action) as s:
            s.case('c', hosts.create_account)
            s.case('l', hosts.log_into_account)

            s.case('a', add_a_snake)
            s.case('y', view_your_snakes)
            s.case('b', book_a_cage)
            s.case('v', view_bookings)
            s.case('m', lambda: 'change_mode')

            s.case('?', show_commands)
            s.case('', lambda: None)
            s.case(['x', 'bye', 'exit', 'exit()'], hosts.exit_app)

            s.default(hosts.unknown_command)

        state.reload_account()

        if action:
            print()

        if s.result == 'change_mode':
            return


def show_commands():
    print('What action would you like to take:')
    print('[C]reate an account')
    print('[L]ogin to your account')
    print('[B]ook a cage')
    print('[A]dd a snake')
    print('View [y]our snakes')
    print('[V]iew your bookings')
    print('[M]ain menu')
    print('e[X]it app')
    print('[?] Help (this info)')
    print()


def add_a_snake():
    print(' ****************** Add a snake **************** ')
    if not state.active_account:
        error_msg("You must log in first to add a snake")
        return

    name = input("What is your snake's name? ")
    if not name:
        error_msg('cancelled')
        return

    length = float(input('How long is your snake (in meters)? '))
    species = input("Species? ")
    is_venomous = input("Is your snake venomous [y]es, [n]o? ").lower().startswith('y')

    snake = svc.add_snake(state.active_account, name, length, species, is_venomous)
    state.reload_account()
    success_msg('Created {} with id {}'.format(snake.name, snake.id))


def view_your_snakes():
    print(' ****************** Your snakes **************** ')
    if not state.active_account:
        error_msg("You must log in first to view your snakes")
        return

    snakes = svc.get_snakes_for_user(state.active_account.id)
    print("You have {} snakes.".format(len(snakes)))
    for s in snakes:
        print(" * {} is a {} that is {}m long and is {}venomous.".format(
            s.name,
            s.species,
            s.length,
            '' if s.is_venomous else 'not '
        ))


def book_a_cage():
    print(' ****************** Book a cage **************** ')
```

```
if not state.active_account:
    error_msg("You must log in first to book a cage")
    return

snakes = svc.get_snakes_for_user(state.active_account.id)
if not snakes:
    error_msg('You must first [a]dd a snake before you can book a cage.')
    return

print("Let's start by finding available cages.")
start_text = input("Check-in date [yyyy-mm-dd]: ")
if not start_text:
    error_msg('cancelled')
    return

checkin = parser.parse(
    start_text
)
checkout = parser.parse(
    input("Check-out date [yyyy-mm-dd]: ")
)
if checkin >= checkout:
    error_msg('Check in must be before check out')
    return

print()
for idx, s in enumerate(snakes):
    print('{}. {} (length: {}, venomous: {})'.format(
        idx + 1,
        s.name,
        s.length,
        'yes' if s.is_venomous else 'no'
    ))

snake = snakes[int(input('Which snake do you want to book (number)')) - 1]

cages = svc.get_available_cages(checkin, checkout, snake)

print("There are {} cages available in that time.".format(len(cages)))
for idx, c in enumerate(cages):
    print(" {}. {} with {}m carpeted: {}, has toys: {}.".format(
        idx + 1,
        c.name,
        c.square_meters,
        'yes' if c.is_carpeted else 'no',
        'yes' if c.has_toys else 'no'))

if not cages:
        error_msg("Sorry, no cages are available for that date.")
        return

    cage = cages[int(input('Which cage do you want to book (number)')) - 1]
    svc.book_cage(state.active_account, snake, cage, checkin, checkout)

    success_msg('Successfully booked {} for {} at ${}/night.'.format(cage.name, 
snake.name, cage.price))


def view_bookings():
    print(' ****************** Your bookings **************** ')
    if not state.active_account:
        error_msg("You must log in first to register a cage")
        return

    snakes = {s.id: s for s in svc.get_snakes_for_user(state.active_account.id)}
    bookings = svc.get_bookings_for_user(state.active_account.email)

    print("You have {} bookings.".format(len(bookings)))
    for b in bookings:
        print(' * Snake: {} is booked at {} from {} for {} days.'.format(
            snakes.get(b.guest_snake_id).name,
            b.cage.name,
            datetime.date(b.check_in_date.year, b.check_in_date.month, b.check_in_date.day),
            (b.check_out_date - b.check_in_date).days
        ))
```

For Host Access

```
import datetime
from colorama import Fore
from dateutil import parser

from infrastructure.switchlang import switch
import infrastructure.state as state
import services.data_service as svc


def run():
    print(' ****************** Welcome host **************** ')
    print()

    show_commands()

    while True:
        action = get_action()
        with switch(action) as s:
            s.case('c', create_account)
            s.case('a', create_account)
            s.case('l', log_into_account)
            s.case('y', list_cages)
            s.case('r', register_cage)
            s.case('u', update_availability)
            s.case('v', view_bookings)
            s.case('m', lambda: 'change_mode')
            s.case(['x', 'bye', 'exit', 'exit()'], exit_app)
            s.case('?', show_commands)
            s.case('', lambda: None)
            s.default(unknown_command)

        if action:
            print()

        if s.result == 'change_mode':
            return


def show_commands():
    print('What action would you like to take:')
    print('[C]reate an [a]ccount')
    print('[L]ogin to your account')
    print('List [y]our cages')
    print('[R]egister a cage')
    print('[U]pdate cage availability')
    print('[V]iew your bookings')
    print('Change [M]ode (guest or host)')
    print('e[X]it app')
    print('[?] Help (this info)')
    print()


def create_account():
    print(' ****************** REGISTER **************** ')

    name = input('What is your name? ')
    email = input('What is your email? ').strip().lower()

    old_account = svc.find_account_by_email(email)
    if old_account:
        error_msg(f"ERROR: Account with email {email} already exists.")
        return

    state.active_account = svc.create_account(name, email)
    success_msg(f"Created new account with id {state.active_account.id}.")
```

```
def log_into_account():
   print(' ****************** LOGIN **************** ')

   email = input('What is your email? ').strip().lower()
   account = svc.find_account_by_email(email)

   if not account:
      error_msg(f'Could not find account with email {email}.')
      return

   state.active_account = account
   success_msg('Logged in successfully.')


def register_cage():
   print(' ****************** REGISTER CAGE **************** ')

   if not state.active_account:
      error_msg('You must login first to register a cage.')
      return

   meters = input('How many square meters is the cage? ')
   if not meters:
      error_msg('Cancelled')
      return

   meters = float(meters)
   carpeted = input("Is it carpeted [y, n]? ").lower().startswith('y')
   has_toys = input("Have snake toys [y, n]? ").lower().startswith('y')
   allow_dangerous = input("Can you host venomous snakes [y, n]? ").lower().startswith('y')
   name = input("Give your cage a name: ")
   price = float(input("How much are you charging?  "))

   cage = svc.register_cage(
      state.active_account, name,
      allow_dangerous, has_toys, carpeted, meters, price
   )

   state.reload_account()
   success_msg(f'Register new cage with id {cage.id}.')


def list_cages(suppress_header=False):
   if not suppress_header:
      print(' ******************     Your cages     **************** ')
```

```
    if not state.active_account:
        error_msg('You must login first to register a cage.')
        return

    cages = svc.find_cages_for_user(state.active_account)
    print(f"You have {len(cages)} cages.")
    for idx, c in enumerate(cages):
        print(f' {idx+1}. {c.name} is {c.square_meters} meters.')
        for b in c.bookings:
            print('      * Booking: {}, {} days, booked? {}'.format(
                b.check_in_date,
                (b.check_out_date - b.check_in_date).days,
                'YES' if b.booked_date is not None else 'no'
            ))


def update_availability():
    print(' ****************** Add available date **************** ')

    if not state.active_account:
        error_msg("You must log in first to register a cage")
        return

    list_cages(suppress_header=True)

    cage_number = input("Enter cage number: ")
    if not cage_number.strip():
        error_msg('Cancelled')
        print()
        return

    cage_number = int(cage_number)

    cages = svc.find_cages_for_user(state.active_account)
    selected_cage = cages[cage_number - 1]

    success_msg("Selected cage {}".format(selected_cage.name))

    start_date = parser.parse(
        input("Enter available date [yyyy-mm-dd]: ")
    )
    days = int(input("How many days is this block of time? "))

    svc.add_available_date(
        selected_cage,
        start_date,
        days
    )
    success_msg(f'Date added to cage {selected_cage.name}.')


def view_bookings():
    print(' ***************** Your bookings *************** ')

    if not state.active_account:
        error_msg("You must log in first to register a cage")
        return

    cages = svc.find_cages_for_user(state.active_account)

    bookings = [
        (c, b)
        for c in cages
        for b in c.bookings
        if b.booked_date is not None
    ]

    print("You have {} bookings.".format(len(bookings)))
    for c, b in bookings:
        print(' * Cage: {}, booked date: {}, from {} for {} days.'.format(
            c.name,
            datetime.date(b.booked_date.year, b.booked_date.month, b.booked_date.day),
            datetime.date(b.check_in_date.year, b.check_in_date.month, b.check_in_date.day),
            b.duration_in_days
        ))


def exit_app():
    print()
    print('bye')
    raise KeyboardInterrupt()


def get_action():
    text = '> '
    if state.active_account:
        text = f'{state.active_account.name}> '

    action = input(Fore.YELLOW + text + Fore.WHITE)
    return action.strip().lower()


def unknown_command():
    print("Sorry we didn't understand that command.")
```

```
def success_msg(text):
    print(Fore.LIGHTGREEN_EX + text + Fore.WHITE)


def error_msg(text):
    print(Fore.LIGHTRED_EX + text + Fore.WHITE)
```

Conclusion

We builded a business application on renting the cages based on MongoDB and python. We learnt document design and data modelling with document databases and use of mongoengine ODM to map classes to MongoDB as well.