CSL7090 Assignment 2

Saurabh Shashikant Burewar (B18CSE050)

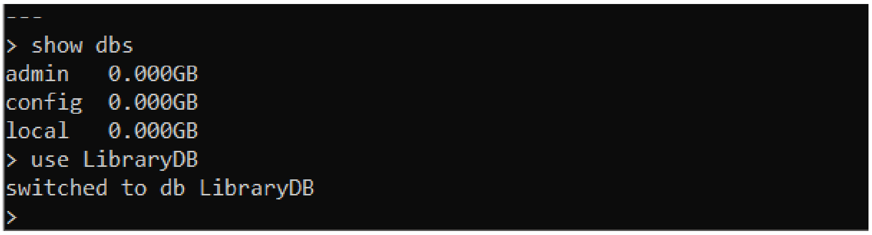
MongoDB and SQL

MongoDB setup

- We are running MongoDB on our system and not Atlas.
- Mongo server runs in the background in the system and all the operations are performed on the mongo shell.

1. **Create database**

   The “use” command creates a new database and switches to it.

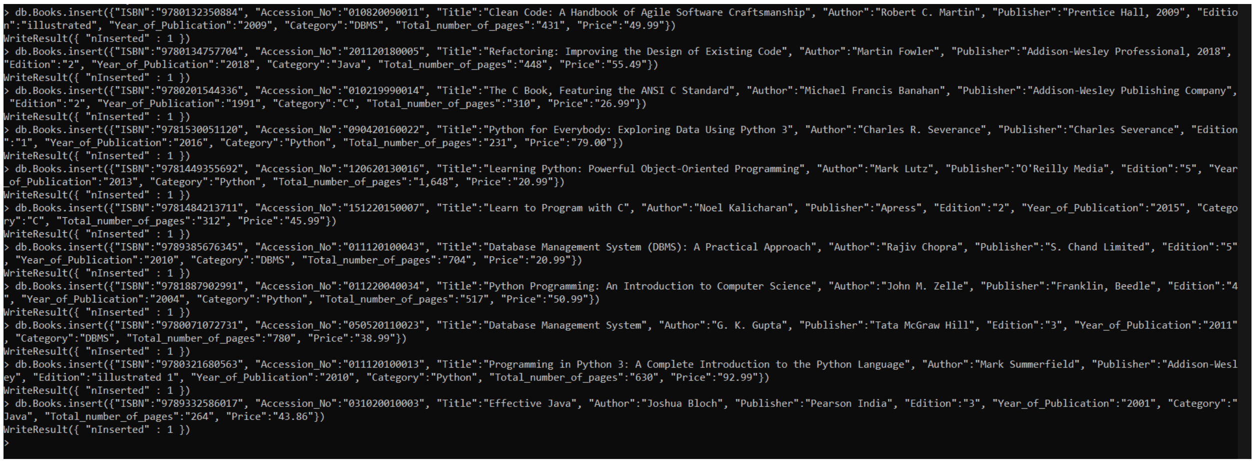


```
---
> show dbs
admin   0.000GB
config  0.000GB
local   0.000GB
> use LibraryDB
switched to db LibraryDB
>
```

   We can create a new collection using - *db.createCollection(“Books”).*

2. **Add documents**

   All entries are added manually in the database using *db.Books.insert().*



```
> db.Books.insert({"ISBN":"9780132350884", "Accession_No":"010820090011", "Title":"Clean Code: A Handbook of Agile Software Craftsmanship", "Author":"Robert C. Martin", "Publisher":"Prentice Hall, 2009", "Editio
n":"illustrated", "Year_of_Publication":"2009", "Category":"DBMS", "Total_number_of_pages":"431", "Price":"49.99"})
WriteResult({ "nInserted" : 1 })
> db.Books.insert({"ISBN":"9780134757704", "Accession_No":"201120180005", "Title":"Refactoring: Improving the Design of Existing Code", "Author":"Martin Fowler", "Publisher":"Addison-Wesley Professional, 2018",
"Edition":"2", "Year_of_Publication":"2018", "Category":"Java", "Total_number_of_pages":"448", "Price":"55.49"})
WriteResult({ "nInserted" : 1 })
> db.Books.insert({"ISBN":"9780201544336", "Accession_No":"010219990014", "Title":"The C Book, Featuring the ANSI C Standard", "Author":"Michael Francis Banahan", "Publisher":"Addison-Wesley Publishing Company",
 "Edition":"2", "Year_of_Publication":"1991", "Category":"C", "Total_number_of_pages":"310", "Price":"26.99"})
WriteResult({ "nInserted" : 1 })
> db.Books.insert({"ISBN":"9781530051120", "Accession_No":"090420160022", "Title":"Python for Everybody: Exploring Data Using Python 3", "Author":"Charles R. Severance", "Publisher":"Charles Severance", "Edition
":"1", "Year_of_Publication":"2016", "Category":"Python", "Total_number_of_pages":"231", "Price":"79.00"})
WriteResult({ "nInserted" : 1 })
> db.Books.insert({"ISBN":"9781449355692", "Accession_No":"120620130016", "Title":"Learning Python: Powerful Object-Oriented Programming", "Author":"Mark Lutz", "Publisher":"O'Reilly Media", "Edition":"5", "Year
_of_Publication":"2013", "Category":"Python", "Total_number_of_pages":"1,648", "Price":"20.99"})
WriteResult({ "nInserted" : 1 })
> db.Books.insert({"ISBN":"9781484213711", "Accession_No":"151220150007", "Title":"Learn to Program with C", "Author":"Noel Kalicharan", "Publisher":"Apress", "Edition":"2", "Year_of_Publication":"2015", "Catego
ry":"C", "Total_number_of_pages":"312", "Price":"45.99"})
WriteResult({ "nInserted" : 1 })
> db.Books.insert({"ISBN":"9789385676345", "Accession_No":"011120100043", "Title":"Database Management System (DBMS): A Practical Approach", "Author":"Rajiv Chopra", "Publisher":"S. Chand Limited", "Edition":"5"
, "Year_of_Publication":"2010", "Category":"DBMS", "Total_number_of_pages":"704", "Price":"20.99"})
WriteResult({ "nInserted" : 1 })
> db.Books.insert({"ISBN":"9781887902991", "Accession_No":"011220040034", "Title":"Python Programming: An Introduction to Computer Science", "Author":"John M. Zelle", "Publisher":"Franklin, Beedle", "Edition":"4
", "Year_of_Publication":"2004", "Category":"Python", "Total_number_of_pages":"517", "Price":"50.99"})
WriteResult({ "nInserted" : 1 })
> db.Books.insert({"ISBN":"9780071072731", "Accession_No":"050520110023", "Title":"Database Management System", "Author":"G. K. Gupta", "Publisher":"Tata McGraw Hill", "Edition":"3", "Year_of_Publication":"2011"
, "Category":"DBMS", "Total_number_of_pages":"780", "Price":"38.99"})
WriteResult({ "nInserted" : 1 })
> db.Books.insert({"ISBN":"9780321680563", "Accession_No":"011120100013", "Title":"Programming in Python 3: A Complete Introduction to the Python Language", "Author":"Mark Summerfield", "Publisher":"Addison-Wesl
ey", "Edition":"illustrated 1", "Year_of_Publication":"2010", "Category":"Python", "Total_number_of_pages":"630", "Price":"92.99"})
WriteResult({ "nInserted" : 1 })
> db.Books.insert({"ISBN":"9789332586017", "Accession_No":"031020010003", "Title":"Effective Java", "Author":"Joshua Bloch", "Publisher":"Pearson India", "Edition":"3", "Year_of_Publication":"2001", "Category":"
Java", "Total_number_of_pages":"264", "Price":"43.86"})
WriteResult({ "nInserted" : 1 })
>
```

3. **Create a replica set (important when writing script)**

   Next, we create a replica set which is needed for us to watch for changes in the database. To do this, we first make some changes to the mongo configuration file to enable replication and restart the server to incorporate the changes. Then, we open the mongo shell and start replication using *rs.initiate()*. This returns a log of successful start.

MySQL setup

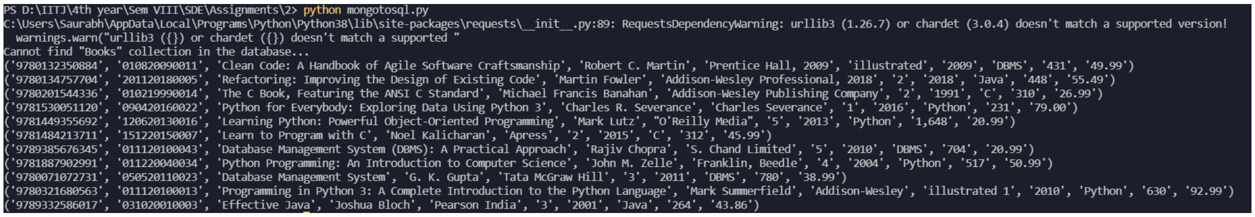
- MySQL installed on system
- Create database using shell

```
mysql -u “root” -p”<password>” -e “CREATE DATABASE librarydb”
```

MongoDB to MySQL script

1. **Libraries used**
   - Pymongo
   - Mysql-connector-python

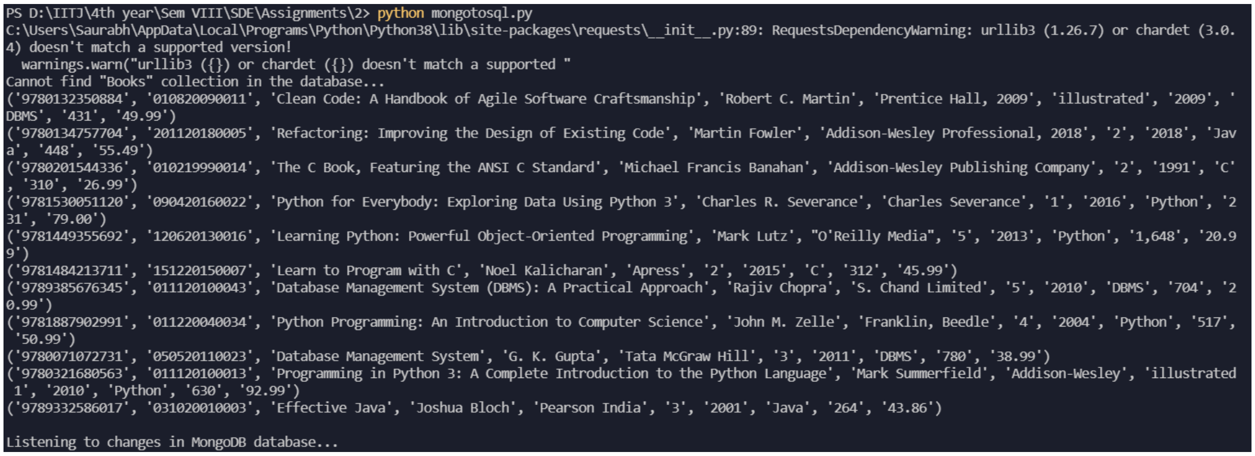
2. **Get MongoDB data and add to MySQL**
   - We create a client to access mongodb server. We use standard functions to access the collections and get all the documents in the collections.
   - We create a connection to MySQL database and create a table with all the required columns like ISBN, Accession No., etc.
   - We parse all the documents in MongoDB and insert them into the SQL table.
   - We run a select query to print all the entries in the SQL table.
   - This Script parses all the documents in the collection and adds them in the MySQL table, so if the run is interrupted, we won’t lose our progress.



```
PS D:\IITJ\4th year\Sem VIII\SDE\Assignments\2> python mongotosql.py
C:\Users\Saurabh\AppData\Local\Programs\Python\Python38\lib\site-packages\requests\__init__.py:89: RequestsDependencyWarning: urllib3 (1.26.7) or chardet (3.0.4) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({}) doesn't match a supported "
Cannot find "Books" collection in the database...
('9780132350884', '010820090011', 'Clean Code: A Handbook of Agile Software Craftsmanship', 'Robert C. Martin', 'Prentice Hall, 2009', 'illustrated', '2009', 'DBMS', '431', '49.99')
('9780134757704', '201120180005', 'Refactoring: Improving the Design of Existing Code', 'Martin Fowler', 'Addison-Wesley Professional, 2018', '2', '2018', 'Java', '448', '55.49')
('9780201544336', '010219990014', 'The C Book, Featuring the ANSI C Standard', 'Michael Francis Banahan', 'Addison-Wesley Publishing Company', '2', '1991', 'C', '310', '26.99')
('9781530051120', '090420160022', 'Python for Everybody: Exploring Data Using Python 3', 'Charles R. Severance', 'Charles Severance', '1', '2016', 'Python', '231', '79.00')
('9781449355692', '120620130016', 'Learning Python: Powerful Object-Oriented Programming', 'Mark Lutz', "O'Reilly Media", '5', '2013', 'Python', '1,648', '20.99')
('9781484213711', '151220150007', 'Learn to Program with C', 'Noel Kalicharan', 'Apress', '2', '2015', 'C', '312', '45.99')
('9789385676345', '011120100043', 'Database Management System (DBMS): A Practical Approach', 'Rajiv Chopra', 'S. Chand Limited', '5', '2010', 'DBMS', '704', '20.99')
('9781887902991', '011220040034', 'Python Programming: An Introduction to Computer Science', 'John M. Zelle', 'Franklin, Beedle', '4', '2004', 'Python', '517', '50.99')
('9780071072731', '050520110023', 'Database Management System', 'G. K. Gupta', 'Tata McGraw Hill', '3', '2011', 'DBMS', '780', '38.99')
('9780321680563', '011120100013', 'Programming in Python 3: A Complete Introduction to the Python Language', 'Mark Summerfield', 'Addison-Wesley', 'illustrated 1', '2010', 'Python', '630', '92.99')
('9789332586017', '031020010003', 'Effective Java', 'Joshua Bloch', 'Pearson India', '3', '2001', 'Java', '264', '43.86')
```

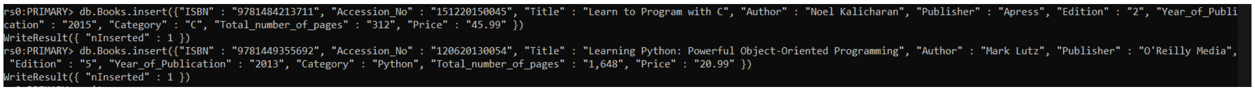
3. **Listening to MongoDB and updating MySQL**

   The script keeps running to listen to any changes in MongoDB using *db.collections.watch()*.



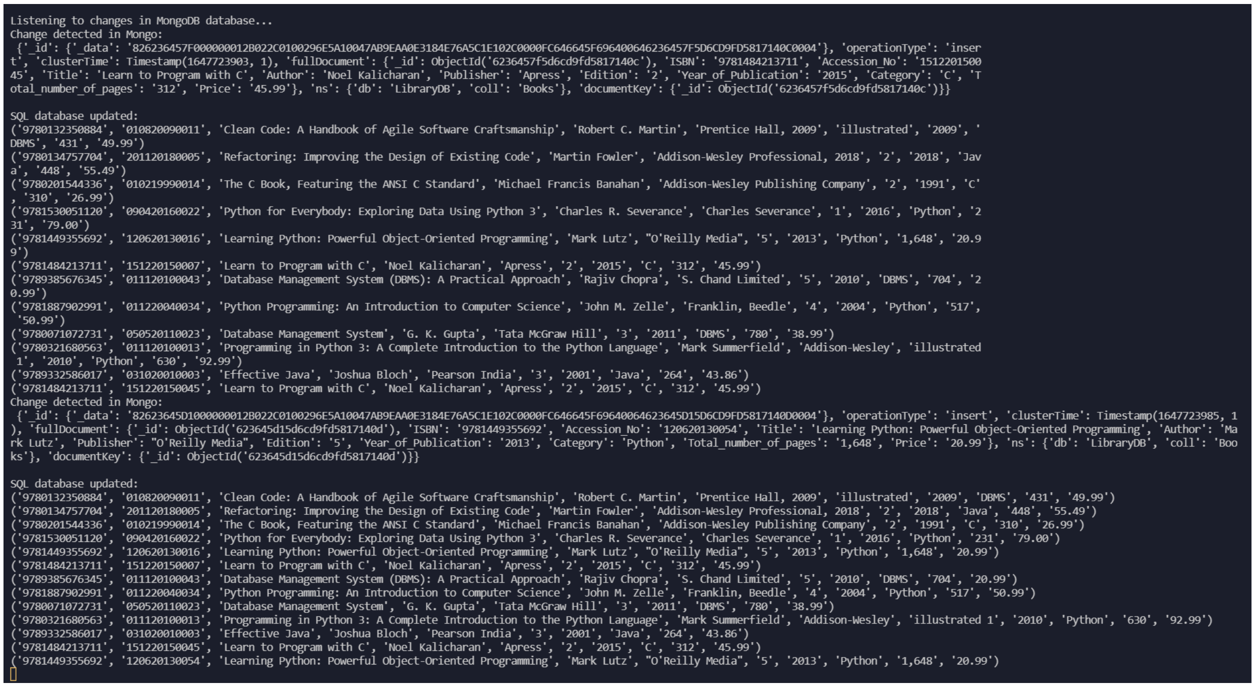
```
PS D:\IITJ\4th year\Sem VIII\SDE\Assignments\2> python mongotosql.py
C:\Users\Saurabh\AppData\Local\Programs\Python\Python38\lib\site-packages\requests\__init__.py:89: RequestsDependencyWarning: urllib3 (1.26.7) or chardet (3.0.
4) doesn't match a supported version!
  warnings.warn("urllib3 ({}) or chardet ({}) doesn't match a supported "
Cannot find "Books" collection in the database...
('9780132350884', '010820090011', 'Clean Code: A Handbook of Agile Software Craftsmanship', 'Robert C. Martin', 'Prentice Hall, 2009', 'illustrated', '2009', '
DBMS', '431', '49.99')
('9780134757704', '201120180005', 'Refactoring: Improving the Design of Existing Code', 'Martin Fowler', 'Addison-Wesley Professional, 2018', '2', '2018', 'Jav
a', '448', '55.49')
('9780201544336', '010219990014', 'The C Book, Featuring the ANSI C Standard', 'Michael Francis Banahan', 'Addison-Wesley Publishing Company', '2', '1991', 'C'
, '310', '26.99')
('9781530051120', '090420160022', 'Python for Everybody: Exploring Data Using Python 3', 'Charles R. Severance', 'Charles Severance', '1', '2016', 'Python', '2
31', '79.00')
('9781449355692', '120620130016', 'Learning Python: Powerful Object-Oriented Programming', 'Mark Lutz', "O'Reilly Media", '5', '2013', 'Python', '1,648', '20.9
9')
('9781484213711', '151220150007', 'Learn to Program with C', 'Noel Kalicharan', 'Apress', '2', '2015', 'C', '312', '45.99')
('9789385676345', '011120100043', 'Database Management System (DBMS): A Practical Approach', 'Rajiv Chopra', 'S. Chand Limited', '5', '2010', 'DBMS', '704', '2
0.99')
('9781887902991', '011220040034', 'Python Programming: An Introduction to Computer Science', 'John M. Zelle', 'Franklin, Beedle', '4', '2004', 'Python', '517',
 '50.99')
('9780071072731', '050520110023', 'Database Management System', 'G. K. Gupta', 'Tata McGraw Hill', '3', '2011', 'DBMS', '780', '38.99')
('9780321680563', '011120100013', 'Programming in Python 3: A Complete Introduction to the Python Language', 'Mark Summerfield', 'Addison-Wesley', 'illustrated
 1', '2010', 'Python', '630', '92.99')
('9789332586017', '031020010003', 'Effective Java', 'Joshua Bloch', 'Pearson India', '3', '2001', 'Java', '264', '43.86')

Listening to changes in MongoDB database...
```

Now, we open a mongo shell to make changes in our mongo database. We insert two documents in the database.



```
rs0:PRIMARY> db.Books.insert({"ISBN" : "9781484213711", "Accession_No" : "151220150045", "Title" : "Learn to Program with C", "Author" : "Noel Kalicharan", "Publisher" : "Apress", "Edition" : "2", "Year_of_Publi
cation" : "2015", "Category" : "C", "Total_number_of_pages" : "312", "Price" : "45.99" })
WriteResult({ "nInserted" : 1 })
rs0:PRIMARY> db.Books.insert({"ISBN" : "9781449355692", "Accession_No" : "120620130054", "Title" : "Learning Python: Powerful Object-Oriented Programming", "Author" : "Mark Lutz", "Publisher" : "O'Reilly Media",
 "Edition" : "5", "Year_of_Publication" : "2013", "Category" : "Python", "Total_number_of_pages" : "1,648", "Price" : "20.99" })
WriteResult({ "nInserted" : 1 })
```

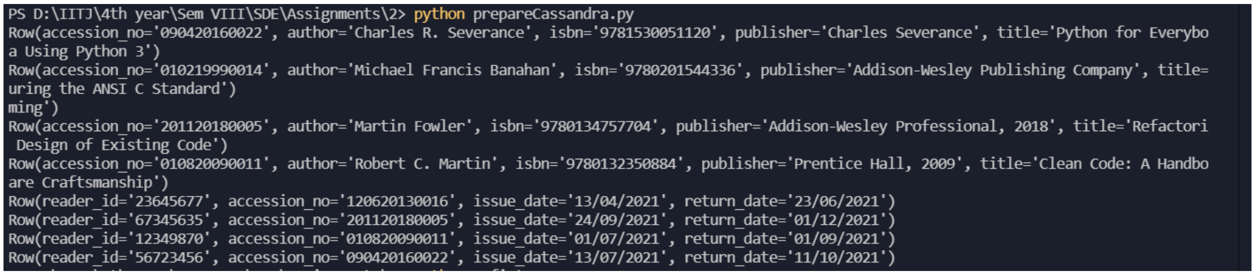
When a change is detected, the watch function returns an object with all details of the change.
We read if the change was insert type and get the key (id) of the document added.
Now, we can use the function collections.find_one({id}) to get the document that was added.
Once we have the document, we use the same method we used above to add this new entry into the SQL table.



```
Listening to changes in MongoDB database...
Change detected in Mongo:
 {'_id': {'_data': '826236457F000000012B022C0100296E5A10047AB9EAA0E3184E76A5C1E102C0000FC646645F696400646236457F5D6CD9FD5817140C0004'}, 'operationType': 'inser
t', 'clusterTime': Timestamp(1647723903, 1), 'fullDocument': {'_id': ObjectId('6236457f5d6cd9fd5817140c'), 'ISBN': '9781484213711', 'Accession_No': '1512201500
45', 'Title': 'Learn to Program with C', 'Author': 'Noel Kalicharan', 'Publisher': 'Apress', 'Edition': '2', 'Year_of_Publication': '2015', 'Category': 'C', 'T
otal_number_of_pages': '312', 'Price': '45.99'}, 'ns': {'db': 'LibraryDB', 'coll': 'Books'}, 'documentKey': {'_id': ObjectId('6236457f5d6cd9fd5817140c')}}

SQL database updated:
('9780132350884', '010820090011', 'Clean Code: A Handbook of Agile Software Craftsmanship', 'Robert C. Martin', 'Prentice Hall, 2009', 'illustrated', '2009', '
DBMS', '431', '49.99')
('9780134757704', '201120180005', 'Refactoring: Improving the Design of Existing Code', 'Martin Fowler', 'Addison-Wesley Professional, 2018', '2', '2018', 'Jav
a', '448', '55.49')
('9780201544336', '010219990014', 'The C Book, Featuring the ANSI C Standard', 'Michael Francis Banahan', 'Addison-Wesley Publishing Company', '2', '1991', 'C'
, '310', '26.99')
('9781530051120', '090420160022', 'Python for Everybody: Exploring Data Using Python 3', 'Charles R. Severance', 'Charles Severance', '1', '2016', 'Python', '2
31', '79.00')
('9781449355692', '120620130016', 'Learning Python: Powerful Object-Oriented Programming', 'Mark Lutz', "O'Reilly Media", '5', '2013', 'Python', '1,648', '20.9
9')
('9781484213711', '151220150007', 'Learn to Program with C', 'Noel Kalicharan', 'Apress', '2', '2015', 'C', '312', '45.99')
('9789385676345', '011120100043', 'Database Management System (DBMS): A Practical Approach', 'Rajiv Chopra', 'S. Chand Limited', '5', '2010', 'DBMS', '704', '2
0.99')
('9781887902991', '011220040034', 'Python Programming: An Introduction to Computer Science', 'John M. Zelle', 'Franklin, Beedle', '4', '2004', 'Python', '517',
 '50.99')
('9780071072731', '050520110023', 'Database Management System', 'G. K. Gupta', 'Tata McGraw Hill', '3', '2011', 'DBMS', '780', '38.99')
('9780321680563', '011120100013', 'Programming in Python 3: A Complete Introduction to the Python Language', 'Mark Summerfield', 'Addison-Wesley', 'illustrated
 1', '2010', 'Python', '630', '92.99')
('9789332586017', '031020010003', 'Effective Java', 'Joshua Bloch', 'Pearson India', '3', '2001', 'Java', '264', '43.86')
('9781484213711', '151220150045', 'Learn to Program with C', 'Noel Kalicharan', 'Apress', '2', '2015', 'C', '312', '45.99')
Change detected in Mongo:
 {'_id': {'_data': '82623645D1000000012B022C0100296E5A10047AB9EAA0E3184E76A5C1E102C0000FC646645F69640064623645D15D6CD9FD5817140D0004'}, 'operationType': 'insert', 'clusterTime': Timestamp(1647723985, 1
), 'fullDocument': {'_id': ObjectId('623645d15d6cd9fd5817140d'), 'ISBN': '9781449355692', 'Accession_No': '120620130054', 'Title': 'Learning Python: Powerful Object-Oriented Programming', 'Author': 'Ma
rk Lutz', 'Publisher': "O'Reilly Media", 'Edition': '5', 'Year_of_Publication': '2013', 'Category': 'Python', 'Total_number_of_pages': '1,648', 'Price': '20.99'}, 'ns': {'db': 'LibraryDB', 'coll': 'Boo
ks'}, 'documentKey': {'_id': ObjectId('623645d15d6cd9fd5817140d')}}

SQL database updated:
('9780132350884', '010820090011', 'Clean Code: A Handbook of Agile Software Craftsmanship', 'Robert C. Martin', 'Prentice Hall, 2009', 'illustrated', '2009', 'DBMS', '431', '49.99')
('9780134757704', '201120180005', 'Refactoring: Improving the Design of Existing Code', 'Martin Fowler', 'Addison-Wesley Professional, 2018', '2', '2018', 'Java', '448', '55.49')
('9780201544336', '010219990014', 'The C Book, Featuring the ANSI C Standard', 'Michael Francis Banahan', 'Addison-Wesley Publishing Company', '2', '1991', 'C', '310', '26.99')
('9781530051120', '090420160022', 'Python for Everybody: Exploring Data Using Python 3', 'Charles R. Severance', 'Charles Severance', '1', '2016', 'Python', '231', '79.00')
('9781449355692', '120620130016', 'Learning Python: Powerful Object-Oriented Programming', 'Mark Lutz', "O'Reilly Media", '5', '2013', 'Python', '1,648', '20.99')
('9781484213711', '151220150007', 'Learn to Program with C', 'Noel Kalicharan', 'Apress', '2', '2015', 'C', '312', '45.99')
('9789385676345', '011120100043', 'Database Management System (DBMS): A Practical Approach', 'Rajiv Chopra', 'S. Chand Limited', '5', '2010', 'DBMS', '704', '20.99')
('9781887902991', '011220040034', 'Python Programming: An Introduction to Computer Science', 'John M. Zelle', 'Franklin, Beedle', '4', '2004', 'Python', '517', '50.99')
('9780071072731', '050520110023', 'Database Management System', 'G. K. Gupta', 'Tata McGraw Hill', '3', '2011', 'DBMS', '780', '38.99')
('9780321680563', '011120100013', 'Programming in Python 3: A Complete Introduction to the Python Language', 'Mark Summerfield', 'Addison-Wesley', 'illustrated 1', '2010', 'Python', '630', '92.99')
('9789332586017', '031020010003', 'Effective Java', 'Joshua Bloch', 'Pearson India', '3', '2001', 'Java', '264', '43.86')
('9781484213711', '151220150045', 'Learn to Program with C', 'Noel Kalicharan', 'Apress', '2', '2015', 'C', '312', '45.99')
('9781449355692', '120620130054', 'Learning Python: Powerful Object-Oriented Programming', 'Mark Lutz', "O'Reilly Media", '5', '2013', 'Python', '1,648', '20.99')
```

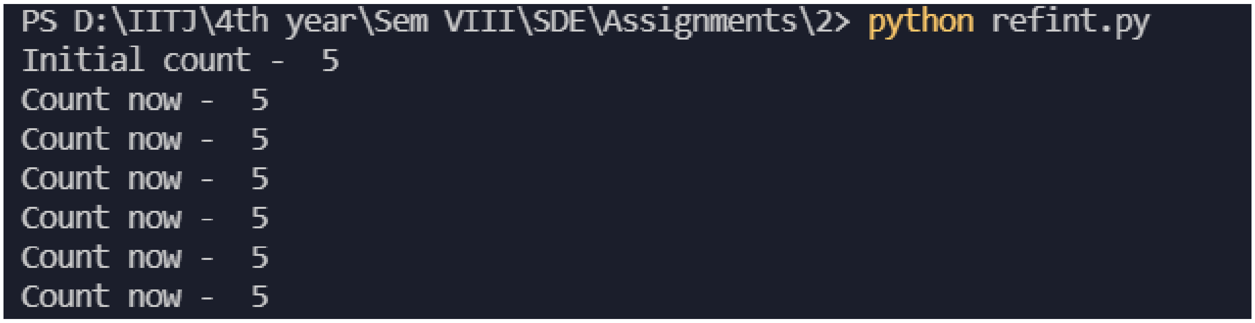
Wrapper Script

1. **Prepare database**
    - First, we need to prepare the database so we can write and test our script.
    - A python script is used to create two tables of Books and Readers and fill them with 5 entries each. The Books table has the Accession_no as the primary key which makes the Accession_no in Readers a foreign key.



```
PS D:\IITJ\4th year\Sem VIII\SDE\Assignments\2> python prepareCassandra.py
Row(accession_no='090420160022', author='Charles R. Severance', isbn='9781530051120', publisher='Charles Severance', title='Python for Everybo
a Using Python 3')
Row(accession_no='010219990014', author='Michael Francis Banahan', isbn='9780201544336', publisher='Addison-Wesley Publishing Company', title=
uring the ANSI C Standard')
ming')
Row(accession_no='201120180005', author='Martin Fowler', isbn='9780134757704', publisher='Addison-Wesley Professional, 2018', title='Refactori
 Design of Existing Code')
Row(accession_no='010820090011', author='Robert C. Martin', isbn='9780132350884', publisher='Prentice Hall, 2009', title='Clean Code: A Handbo
are Craftsmanship')
Row(reader_id='23645677', accession_no='120620130016', issue_date='13/04/2021', return_date='23/06/2021')
Row(reader_id='67345635', accession_no='201120180005', issue_date='24/09/2021', return_date='01/12/2021')
Row(reader_id='12349870', accession_no='010820090011', issue_date='01/07/2021', return_date='01/09/2021')
Row(reader_id='56723456', accession_no='090420160022', issue_date='13/07/2021', return_date='11/10/2021')
```

2. **Script**
    - The script keeps running in the background and keeps track of the number of entries in the table Books and also keeps a record of primary keys.
    - It checks for count every five seconds and if the count decreases, it means a row has been deleted. It checks which row is deleted using the keys and deletes the corresponding row in Readers. Then, it updates our initial count and keys record with these new values. Now, it again starts checking for counts.
    - This way referential integrity is maintained since the Readers will never have an entry which is not in Books.



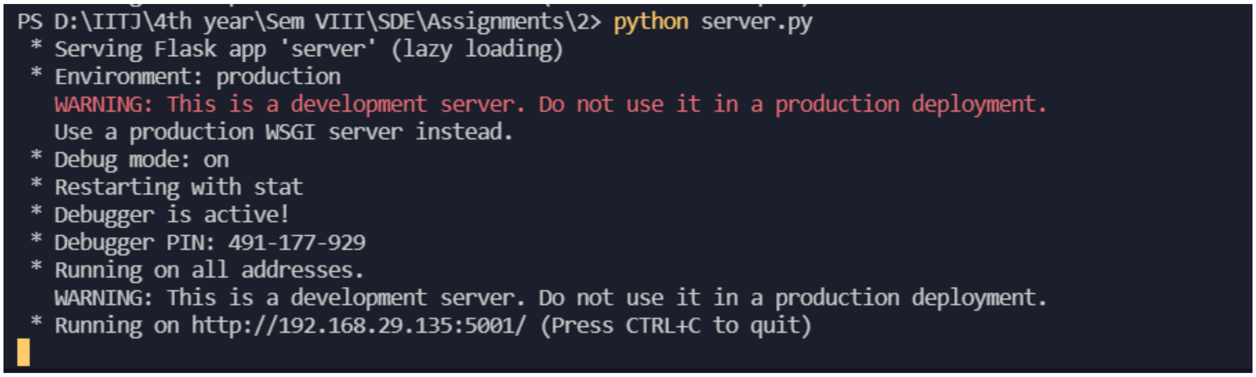
```
PS D:\IITJ\4th year\Sem VIII\SDE\Assignments\2> python refint.py
Initial count -  5
Count now -  5
Count now -  5
Count now -  5
Count now -  5
Count now -  5
Count now -  5
```

API

3. **Language and Libraries used**
    - Python
    - Flask
    - OpenCV

4. Server

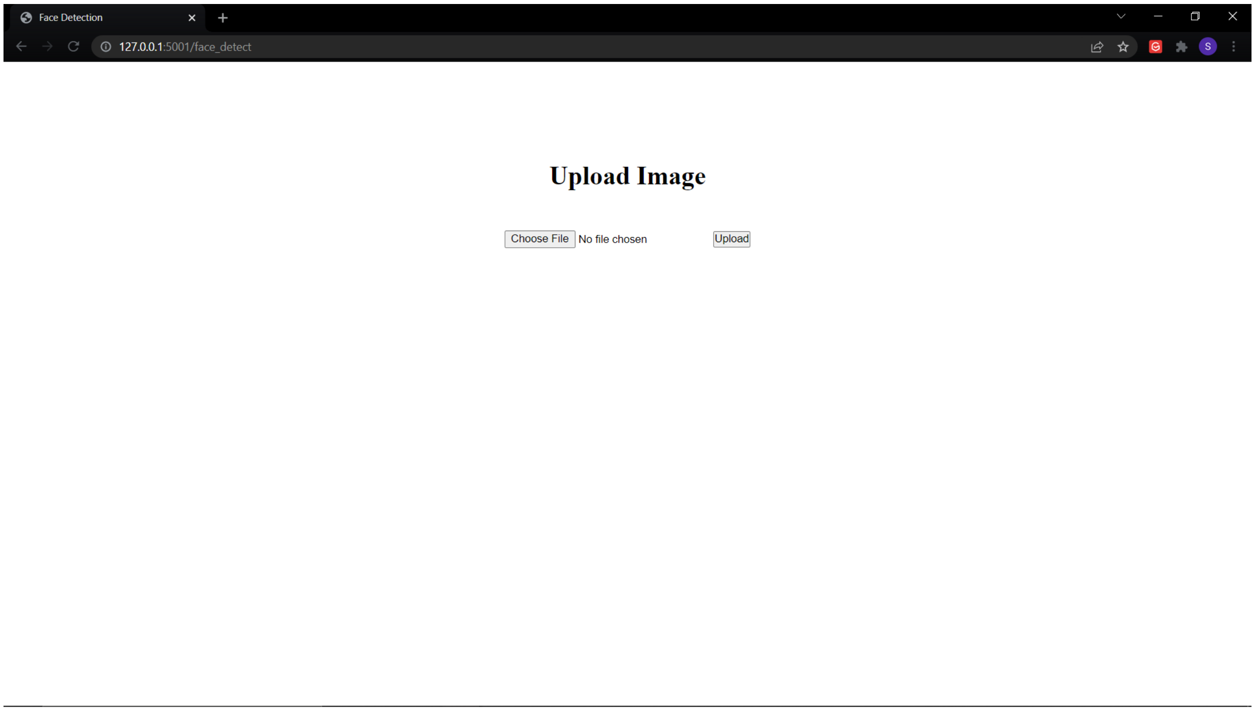
- We are using Flask to create our simple API. So, we create a Flask application and create a route '/face_detect' with POST and GET methods.
- It takes files from clients and calls our face detection function (discussed below) to detect faces. Then, it returns the result to the client.
- We have also defined a simple HTML page which is displayed at the client side.
- Then, we run the server and keep it running.



```
PS D:\IITJ\4th year\Sem VIII\SDE\Assignments\2> python server.py
 * Serving Flask app 'server' (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 491-177-929
 * Running on all addresses.
   WARNING: This is a development server. Do not use it in a production deployment.
 * Running on http://192.168.29.135:5001/ (Press CTRL+C to quit)
```

5. Client

- We define a localhost for the client and once the server is running, the url will show our simple HTML page.
- Then, we can upload our image for detection. The client sends a request to the server.
- The client receives a response from the server which is then displayed in the browser window.

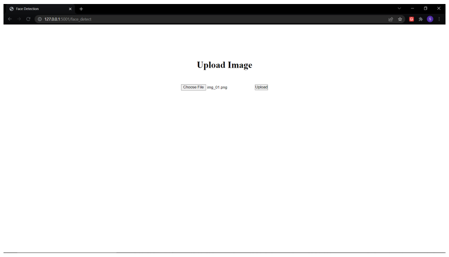
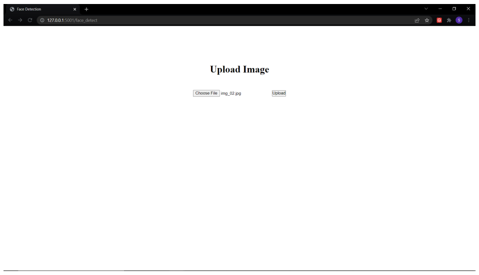
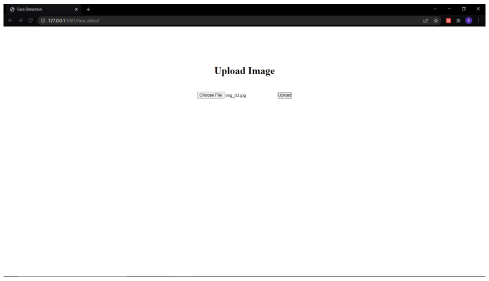
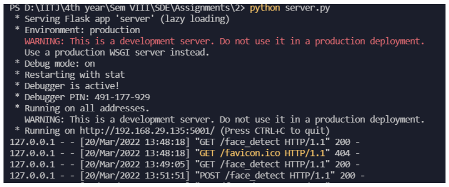
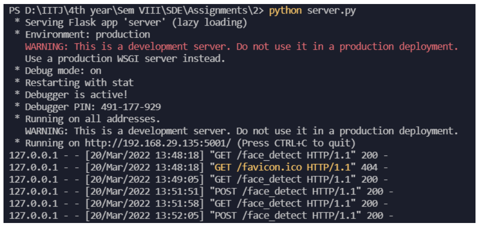
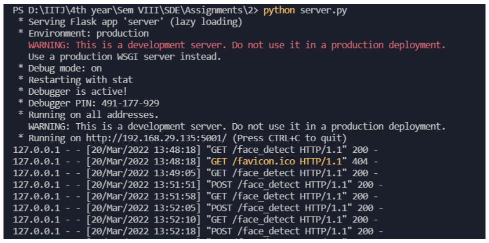
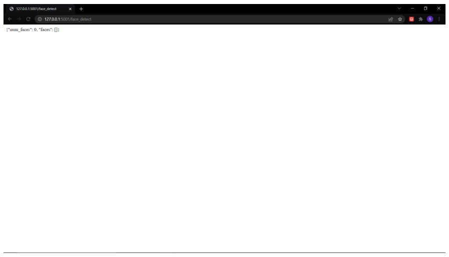
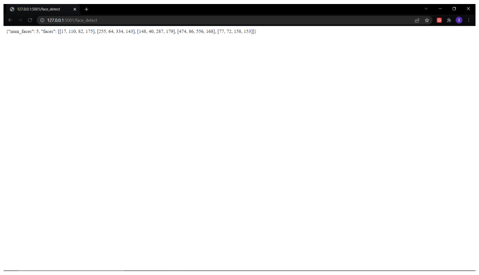
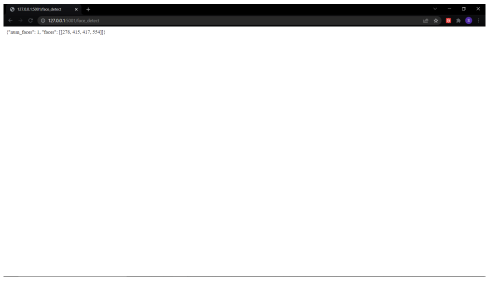


Upload Image

6. **Face detection**

   Our face detection utility works using OpenCV. We use the Haar Cascade classifier to detect faces and return the number of faces detected and the location of those faces. This detector is not completely correct which is why it is only able to detect one face in image 3.

7. **Results (zoom in to see images)**

	Image 1	Image 2	Image 3
Client input	 Face Detection 127.0.0.1:5001/face_detect Choose File Upload Upload Image img_01.png	 Face Detection 127.0.0.1:5001/face_detect Choose File Upload Upload Image img_02.jpg	 Face Detection 127.0.0.1:5001/face_detect Choose File Upload Upload Image img_03.jpg
Server	 PS D:\IITJ\4th year\Sem VIII\SDE\Assignments\2> python server.py * Serving Flask app 'server' (lazy loading) * Environment: production WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead. * Debug mode: on * Restarting with stat * Debugger is active! * Debugger PIN: 491-177-929 * Running on all addresses. WARNING: This is a development server. Do not use it in a production deployment. * Running on http://192.168.29.135:5001/ (Press CTRL+C to quit) 127.0.0.1 - - [20/Mar/2022 13:48:18] "GET /face_detect HTTP/1.1" 200 - 127.0.0.1 - - [20/Mar/2022 13:48:18] "GET /favicon.ico HTTP/1.1" 404 - 127.0.0.1 - - [20/Mar/2022 13:49:05] "GET /face_detect HTTP/1.1" 200 - 127.0.0.1 - - [20/Mar/2022 13:51:51] "POST /face_detect HTTP/1.1" 200 -	 PS D:\IITJ\4th year\Sem VIII\SDE\Assignments\2> python server.py * Serving Flask app 'server' (lazy loading) * Environment: production WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead. * Debug mode: on * Restarting with stat * Debugger is active! * Debugger PIN: 491-177-929 * Running on all addresses. WARNING: This is a development server. Do not use it in a production deployment. * Running on http://192.168.29.135:5001/ (Press CTRL+C to quit) 127.0.0.1 - - [20/Mar/2022 13:48:18] "GET /face_detect HTTP/1.1" 200 - 127.0.0.1 - - [20/Mar/2022 13:48:18] "GET /favicon.ico HTTP/1.1" 404 - 127.0.0.1 - - [20/Mar/2022 13:49:05] "GET /face_detect HTTP/1.1" 200 - 127.0.0.1 - - [20/Mar/2022 13:51:51] "POST /face_detect HTTP/1.1" 200 - 127.0.0.1 - - [20/Mar/2022 13:51:58] "GET /face_detect HTTP/1.1" 200 - 127.0.0.1 - - [20/Mar/2022 13:52:05] "POST /face_detect HTTP/1.1" 200 -	 PS D:\IITJ\4th year\Sem VIII\SDE\Assignments\2> python server.py * Serving Flask app 'server' (lazy loading) * Environment: production WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead. * Debug mode: on * Restarting with stat * Debugger is active! * Debugger PIN: 491-177-929 * Running on all addresses. WARNING: This is a development server. Do not use it in a production deployment. * Running on http://192.168.29.135:5001/ (Press CTRL+C to quit) 127.0.0.1 - - [20/Mar/2022 13:48:18] "GET /face_detect HTTP/1.1" 200 - 127.0.0.1 - - [20/Mar/2022 13:48:18] "GET /favicon.ico HTTP/1.1" 404 - 127.0.0.1 - - [20/Mar/2022 13:49:05] "GET /face_detect HTTP/1.1" 200 - 127.0.0.1 - - [20/Mar/2022 13:51:51] "POST /face_detect HTTP/1.1" 200 - 127.0.0.1 - - [20/Mar/2022 13:51:58] "GET /face_detect HTTP/1.1" 200 - 127.0.0.1 - - [20/Mar/2022 13:52:05] "POST /face_detect HTTP/1.1" 200 - 127.0.0.1 - - [20/Mar/2022 13:52:10] "GET /face_detect HTTP/1.1" 200 - 127.0.0.1 - - [20/Mar/2022 13:52:18] "POST /face_detect HTTP/1.1" 200 -
Client output	 {"num_faces": 0, "faces": []}	 {"num_faces": 5, "faces": [[17, 110, 82, 175], [255, 64, 334, 143], [148, 40, 287, 179], [474, 86, 556, 168], [77, 72, 158, 153]]}	 {"num_faces": 1, "faces": [[278, 415, 417, 554]]}

References

- MongoDB and SQL -
  - https://www.youtube.com/watch?v=FwMwO8pXfq0
  - https://www.bogotobogo.com/DevOps/MongoDB/MongoDB_Replication_Replica_Set.php
  - https://onecompiler.com/questions/3vchw5r82/unable-to-use-mongodb-change-streams-mongoerror-the-changestream-stage-is-only-supported-on-replica-sets
  - https://www.w3schools.com/python/python_mongodb_getstarted.asp

    - https://www.w3schools.com/python/python_mysql_getstarted.asp
- Wrapper Script -
    - https://www.youtube.com/watch?v=hJxlkHafYsQ
    - https://docs.datastax.com/en/developer/python-driver/3.15/getting_started/
    - https://cassandra.apache.org/doc/latest/cassandra/getting_started/drivers.html
    - https://towardsdatascience.com/getting-started-with-apache-cassandra-and-python-81e00ccf17c9
- Flask and OpenCV -
    - https://flask.palletsprojects.com/en/2.0.x/quickstart/
    - https://www.freecodecamp.org/news/how-to-build-a-web-application-using-flask-and-deploy-it-to-the-cloud-3551c985e492/
    - https://blog.stoplight.io/python-rest-api
    - https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html