

Graphical user interface, text

Description automatically generated

**A MINI PROJECT REPORT ON**

**Online Course Management System**

*Submitted to Visvesvaraya Technological University in partial fulfillment of the requirement for the award of degree of*

***Bachelor of Engineering***

***in***

***Computer Science and Engineering.***

**Submitted by:**

Mr.Manjush V G 4JN19CS050

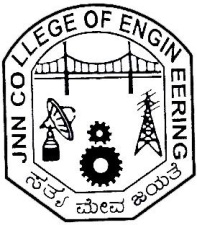
Mr.Murali venkat J 4JN19CS057

Mr.Ganesh prasad M 4JN19CS033

Mr.Mahendra S M 4JN19CS048

**Under the Guidance of:**

|  |  |
| --- | --- |
| **Mr. Sathyanarayana S**  B.E, M.Sc(IT), M.Tech  Assistant Prof., CSE | **Mrs. Namitha M V**  B.E  Assistant Prof., CSE |



**January 2022**

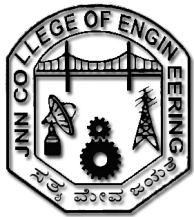
Department of Computer Science & Engineering

**Jawaharlal Nehru New College of Engineering**

Shivamogga - 577 204

**National Education Society ®**





Jawaharlal Nehru New College of Engineering

Department of Computer Science & Engineering

**CERTIFICATE**

*This is to certify that the Mini-project Report entitled*

**Online Course Management System**

**Submitted by:**

|  |  |
| --- | --- |
| Manjush V G | 4JN19CS050 |
| Murali venkat J | 4JN19CS057 |
| Ganesh prasad M | 4JN19CS033 |
| Mahendra S M | 4JN19CS048 |

*Students of 5th semester B.E under the supervision and guidance towards the partial fulfillment of the requirement for award of degree of the Bachelor of Engineering in Computer Science and Engineering of Visvesvaraya Technological University, Belgaum during the year 2021 – 2022.*

**Signature of Guides and HOD:**

|  |  |  |
| --- | --- | --- |
| **Mr. Sathyanarayana S**  B.E, M.Sc(IT), M.Tech  Assistant Prof., CSE | **Mrs. Namitha M V**  B.E  Assistant Prof., CSE | **Dr. Poornima K M**  Ph.D.,  Professor & Head, CSE |

**Name of Examiners Signature with Date**

1.

2.



**Abstract**



Acknowledgement

On presenting the Database Management Systems Mini Project report on "**Online Course Management System** ", I feel great to express my humble feelings of thanks to all those who have helped me directly or indirectly in the successful completion of the project work.

I would like to thank our respected guides **Mr.Sathyanarayan S**, Assistant Professor, Dept. of CS & E, and **Mrs Namitha Venkateshaiah**, Assistant Professor, Dept. of CS &E, who has helped us a lot in completing this task, for their continuous encouragement and guidance throughout the project work.

I would like to thank **Dr. Nirmala Shivanand**, Professor and Head, Dept of CSE, JNNCE, shivamogga and **Dr. Shashidhar K Kudari**, The Principal, JNNCE, Shivamogga for all their support and encouragement.

I am grateful to Department of Computer Science and Engineering and Our institution Jawaharlal Nehru National College of Engineering and for imparting me the knowledge with which I can do my best.

Finally, I also would like to thank the whole teaching and nonteaching staff of Computer Science and Engineering Dept.

**Thank You All,**

**Name And USN**

Manjush V G 4JN19CS050

Murali venkat J 4JN19CS058

Ganesh prasad M 4JN19CS033

Mahendra S M 4JN19CS048



**Table of Content**

1. **Chapter-1: Introduction**
   1. Overview of Online Courses 1-2
   2. Applications 2-4
   3. Problem Statement 5-6
   4. Objectives
   5. Overview of Django
   6. Overview of MySQL
2. **Chapter-2: Design and Implementation**
   1. Functional Requirements
   2. Design
   3. HTML
   4. Django MVT System
   5. Django Admin Interfaces
   6. Implementation
   7. APIs Used
3. **Chapter-3: Results**
   1. Explore
   2. Search
   3. My courses of student
   4. Wishlist
   5. Course
   6. My courses of Faculty
4. **Chapter-4: Conclusions and Future Scope**



**Chapter**-1

**Introduction**

* 1. **Overview Of Online Courses**

**Online course** is education that takes place over the Internet. They are generally conducted through a learning management system as our project, in which students can view their course syllabus and academic progress. **Online courses** are generally self-paced, allowing for greater flexibility in completing coursework.

* 1. **Applications**
  2. **Problem Statement**

Nowadays’ the world of education and learning is moving towards online. Every student may not have a good bandwidth and they may face network issues if the teaching is live. So, for those students who do not have a good bandwidth, concepts will not be clear.



* 1. **Objectives**
* Main objective of Online Course management system is to provide simple, universal access to information and services for students, faculty, and universities.
* Our aim is to give best service to the users who wants to get knowledge from our online platform.
* Enhance the quality of learning and teaching.
* Meet the learning style or needs of students.
* Improve the efficiency and effectiveness.
* Improve user-accessibility and time flexibility to engage learners in the learning process.
  1. **Overview of Django**

Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. Django is based on MVT (Model-View-Template) architecture. MVT is a software design pattern for developing a web application. MVT Structure has the following three parts – Model: The model is going to act as the interface of your data. It is responsible for maintaining data. It is the logical data structure behind the entire application and is represented by a database (generally relational databases such as MySQL, Postgres). View: The View is the user interface — what you see in your browser when you render a website. It is represented by HTML/CSS/JavaScript and Jinja files. Template: A template consists of static parts of the desired HTML output as well as some special syntax describing how dynamic content will be inserted.

**Features of Django**: -

* **Rapid Development**: - Django was designed with the intention to make a framework which takes less time to build web application. The project implementation phase is a very time taken but Django creates it rapidly.
* **Secure**: - Django takes security seriously and helps developers to avoid many common security mistakes, such as SQL injection, cross-site scripting, cross-site request forgery etc. Its user authentication system provides a secure way to manage user accounts and passwords.
* **Scalable**: - Django is scalable in nature and has ability to switch from small quickly and flexibly to large scale application project
* **Fully loaded**: - Django includes various helping task modules and libraries which can be used to handle common Web development tasks. Django takes care of user authentication, content administration, site maps, RSS feeds etc.



* **Versatile**: - Django is versatile in nature which allows it to build applications for different-different domains. Now a days, Companies are using Django to build various types of applications like content management systems, social networks sites or scientific computing platforms etc.
* **Open Source**: - Django is an open-source web application framework. It is publicly available without cost. It can be downloaded with source code from the public repository. Open source reduces the total cost of the application development.
* **Vast and Supported Community**: - Django is a one of the most popular web frameworks. It has widely supportive community and channels to share and connect.
  1. **Overview of MySQL**

SQL (Structured Query Language) queries. Architecture of MYSQL describes the relation among the different components of MYSQL System. MYSQL follow Client-Server Architecture. It is designed so that end user that is Clients can access the resources from Computer that is server using various networking services. The Architecture of MYSQL contain following major layer’s:

• Client

• Server

• Storage Layer

**Features of MySQL: -**

* MySQL is a relational database management system. This database language is based on the SQL queries to access and manage the records of the table.
* MySQL is easy to use. We must get only the basic knowledge of SQL. We can build and interact with MySQL by using only a few simple SQL statements.
* MySQL consists of a solid data security layer that protects sensitive data from intruders. Also, passwords are encrypted in MySQL.
* MySQL follows the working of a client/server architecture. There is a database server (MySQL) and arbitrarily many clients (application programs), which communicate with the server; that is, they can query data, save changes, etc.
* MySQL supports multi-threading that makes it easily scalable. It can handle almost any amount of data, up to as much as 50 million rows or more. The default file size limit is about 4 GB. However, we can increase this number to a theoretical limit of 8 TB of data.
* MySQL is considered one of the very fast database languages, backed by many the benchmark test.
* MySQL supports many embedded applications, which makes MySQL very flexible.
* MySQL is faster, more reliable, and cheaper



**Chapter-2**

**Design and Implementation**

**2.1. Functional Requirements**

* **Search**

Search feature will allow a user to find the courses based on the topic, specialization, skills that he/she wants to acquire, instructor who taught the course or the courses which belong to an institute.

* **Wishlist**

A student can add courses to Wishlist. Later he/she can enroll that course.

* **Enroll**

A student could enroll to any course, if the course provided by the same institute to which the student belongs then the course will be enrolled for free else, they must pay the fees for the course.

* **Progress**

This website will also keep track of students' progress in the course. If the student watches up to 95% of the video, then that video will be marked as completed.

* **Upload**

A faculty can upload courses to the website. After uploading he/she can see the number of views for his/her course, number of students who have enrolled for the course and the rating of the course.

* **Approve**

An Institute must verify whether the student studies in their institute or not and the institute has to approve the enrollment of the course for a student after the payment.



**ER Diagram**

The below figure shows the ER diagram and ERD is a model that identifies the concepts or entities that exist in a system and the relationships between those entities.

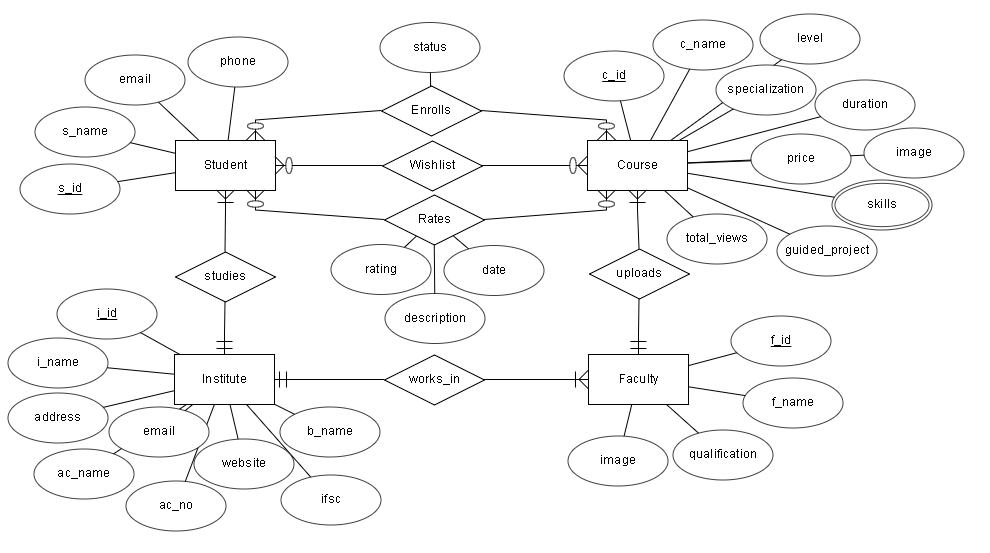


Figure 2.1

**Entities**

* **Institute: -**

This entity stores Institute details such as name, address, email, image, institute website link, account number, IFSC, account holder name. i\_id is the primary key. email is used for login. Bank account details is needed because the students must pay the fees for the institute to get access for a course.

* **Faculty: -**

This entity stores the faculty details such as name, qualification, image, i\_id. Here f\_id is the primary key. i\_id is the foreign key which refers to the institute to which the faculty works for. Faculty will upload the courses.

* **Student: -**

This entity stores Student details such as name, email, phone number. s\_id is the primary key. i\_id is the foreign key which refers to the institute in which the student studies. A student can add a course to Wishlist, and he/she can enroll for the course by paying the fees. After completing the course, he/she can give the rating for that course which would help other students.



* **Course: -**

This entity stores many attributes related to a course such as name, description, playlist\_id, specialization, level, skills, duration, price, image, total\_views, guided\_project, date, no\_videos, rating. c\_id is the primary key. f\_id is the foreign key which refers to the faculty who has uploaded the course. playlist\_id stores the playlist id of the YouTube playlist which is unlisted (playlist can be accessed only through link). skills attribute is multivalued attribute. no\_videos will store the number of videos in the playlist, duration will store the total duration of the playlist and values of both attributes are calculated before inserting a course using the YouTube data API v3. guided\_project is Boolean attribute which will store 1 if the course is guided project else 0. Initially rating will be zero, when a student gives a review then the trigger which we have created will be fired and the rating will be updated.

**Relationship between Entities**

* Student—Enrolls—Course

**Constraint** **Cardinality ratio**

**Student** Optional Many

**Course** Optional Many

* Student—Wishes—Course

**Constraint** **Cardinality ratio**

**Student** Optional Many

**Course** Optional Many

* Student—Rates—Course

**Constraint** **Cardinality ratio**

**Student** Optional Many

**Course** Optional Many

* Student—Studies in—Institute

**Constraint** **Cardinality ratio**

**Student** Mandatory Many

**Institute** Mandatory One

* Faculty—Works in—Institute

**Constraint** **Cardinality ratio**

**Faculty** Mandatory Many

**Institute** Mandatory One



* Faculty—Uploads—Course

**Constraint** **Cardinality ratio**

**Faculty** Mandatory One

**Course** Mandatory Many

**2.2. Design**

**Schema diagram:**

* Schema diagram is a skeletal structure that represents the local view of the entire database.
* Pointed arrow is used here to represent the foreign keys of tables (attribute at the head of the arrow represents parent key and attributes at the tail represents child key).

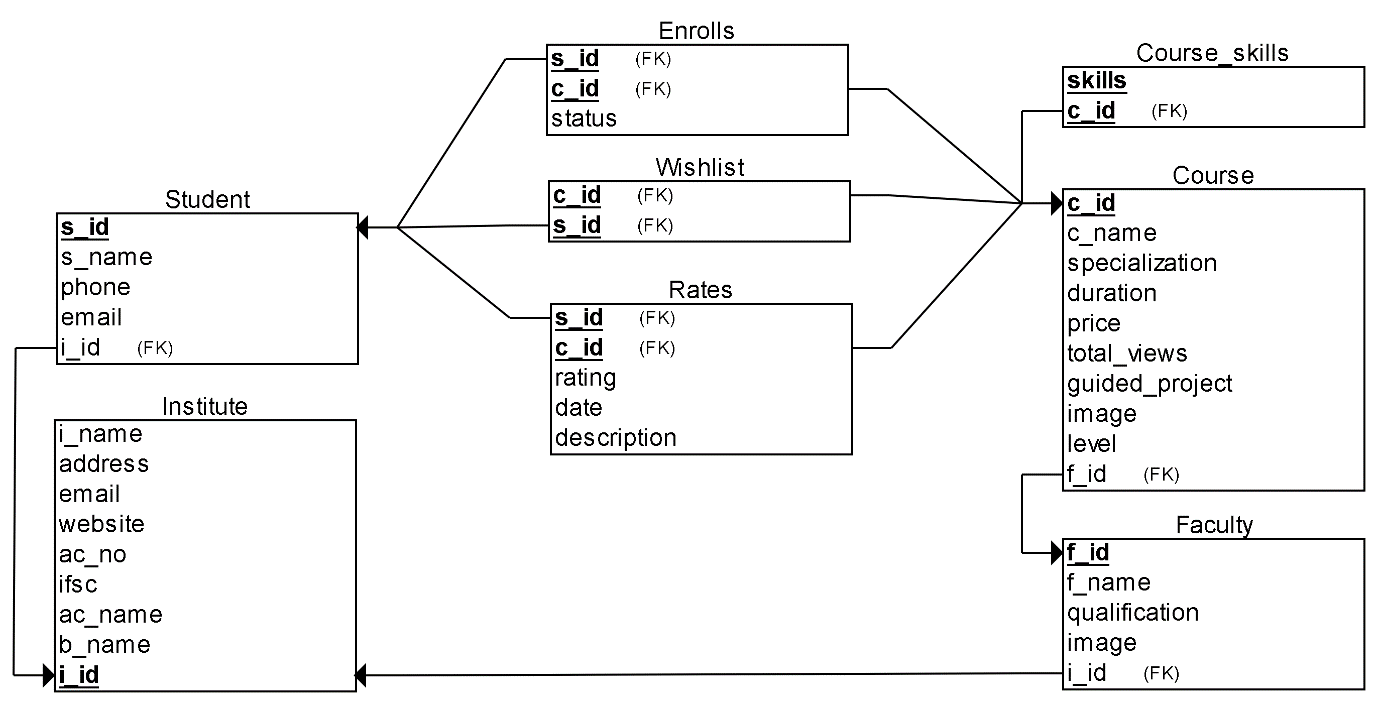


Figure 2.2



**2.3. HTML**

* HTML stands for Hyper Text Markup Language.
* HTML is the standard markup language for creating Web pages.
* HTML describes the structure of a Web page.
* HTML consists of a series of elements.
* HTML elements tell the browser how to display the content.

**HTML TAGS**

* The <!DOCTYPE html> declaration defines that this document is an HTML5 document.
* The <html> element is the root element of an HTML page.
* The <head> element contains meta information about the HTML page.
* The <title> element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab).
* The <body> element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.,
* The <h1> element defines a large heading.
* <p> element defines a paragraph.

**Examples: -**

**<p>**Paragraph**</p>**

**<h2>**Heading tag**</h2>**

**<b>**Bold**</b>**

**<i>**Italic**</i>**

**<u>**Underline**</u>**

**<br>**Tag: br stands for break line, it breaks the line of the code.

**<hr>** Tag: hr stands for Horizontal Rule. This tag is used to put a line across the webpage

**W3.CSS**

* W3.CSS is a modern framework with built-in responsiveness and easy to learn and use compared to other CSS frameworks.
* It aims to speed up and simplify web development and support modern responsive devices like Mobile, Laptop, Tablet and Desktop.
* W3.CSS was designed to be a high-quality alternative to Bootstrap.



* W3.CSS is free for everyone to use. Anyone can use it by simply linking the style in their webpage.

**2.4. Django MVT System**

**A picture containing text

Description automatically generated**

**Figure 2.3**

The MVT (Model View Template) is a software design pattern. It is a collection of three important components Model, View and Template.

**Model:** This is an abstraction layer for structuring and manipulating the data of the Web Application. It acts as an interface for maintaining data. This is a logical data structure behind the entire application and helps to handle the database.

**View:** This layer encapsulates the logic responsible for processing a user’s request and returns a response. It is a user interface to execute the logic and interact with the models. It is responsible for displaying all or a portion of data to the user.

**Template:** The template layer provides a designer-friendly syntax for rendering the information to be presented to the user. It contains the static parts of the desired HTML output 16 along with some special syntax, also known as Django Template Language (DTL), describing how dynamic content will be inserted.



**2.5. Django Admin Interface**

Django provides an admin site to allow CRUD (Create Read Update Delete) operations on registered app model. It is a built-in feature of Django that automatically generates interface for models. We can see the URL entry for admin in urls.py file, it is implicit and generated while creating a new project.

1. urlpatterns = [
2. path('admin/', admin.site.urls),
3. ]

It can be easily accessed by after login from the admin panel, lets run the server python3 manage.py runserver and access it through the localhost:8000/admin.

A login form will be displayed,

Graphical user interface, application, website

Description automatically generated

Figure 2.4

To login, first create admin (super user) user and provide password as we did here:

Text

Description automatically generated

Figure 2.5



Super user is created successfully, now you can login with the username and password that you have given while creating the superuser. After login you will see a page as shown below:

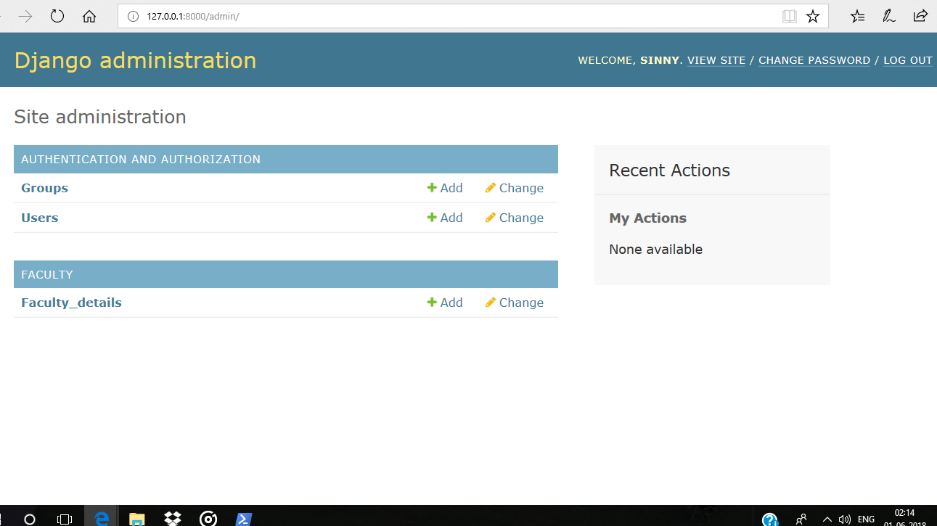


Figure 2.6

It is an admin dashboard that provides facilities like creating groups and users. It also used to manage the models.



**2.6. Implementation**

**2.6.1 Flowchart for Student, Faculty, Institute registration:**

Start

Collect faculty name, email id, phone no , qualification, institute, image, password

Collect student name, email id, phone no, institute, password

Collect institute, Address, image, email, website

Add collected data

to backend

Stop



**2.6.2 Flowchart for user login**

Start

Collect email id, Password

False

Check whether email name & Password are present in backend

True

Display user part page

Stop



**2.6.3 Flowchart for user part page:**

Start

Collect email id, Password

False

Check whether email name & Password are present in backend

True

Display user part page

Stop



**2.6.4 Flowchart for user part page:**

Start

If user click explore button

True

Display explore page

False

If user click my course button

True

Display mycourse page

True

False

If user click wishlist button

True

Display wishlist page

Stop



**2.7. APIs used**

|  |  |
| --- | --- |
| **YouTube Data API V3** | |
| **APIs used** | **Explanation** |
| playlistItems().list() | In this function we have passed four parameters: part, playlistId, maxResults and pageToken.  part is string which contains comma separated playlistItem resource properties. We have used contentDetails (to get video ids of all videos in the playlist) and snippet (to get title, description, and thumbnail of each video of the playlist) properties in part.  With a single request we can fetch maximum of 50 objects of the given properties. If the playlist has more than 50 videos, then we will get nextPageToken (page token of next page) in the response. Now we can use this token as a parameter to fetch next 50 videos. |
| Videos().list() | In this function we have passed two parameters: part, id.  Here part contains videos resource properties. We have used only one property that is contentDetails (to get duration of video whose id is given).  id is string which contains comma separated video ids. We can fetch duration of multiple videos (maximum of 50) at a single time. |

Table 2.1

|  |  |
| --- | --- |
| **YouTube Player API** | |
| **APIs used** | **Explanation** |
| YT.Player() | This function is used to create a Player object. This function will take two parameters: first parameter is DOM element or id of HTML element where the API will insert <iframe> tag containing the player, and second parameter is a JavaScript object which specifies the player options.  The object contains the properties such as width, height, videoId, playerVars, events.  playerVars is an object which contains player parameters such as rel (related videos), autoplay, controls etc. These parameters are either set to 0 or 1.  Events is an object which contains events and their corresponding handler functions. |
| playerObj.getDuration() | Here playerObj is the object that we created using above function. The function getDuration will return the duration of the playerObj video in seconds. |
| playerObj.getCurrentTime() | This function will return the current playing time of the playerObj video in seconds. |

Table 2.2



|  |  |
| --- | --- |
| **Django APIs** | |
| **APIs used** | **Explanation** |
| request.GET.get | request.GET contains the GET variables. These are what you see in your browser's address bar. The. get() method is a method used for dictionaries . |
| Request.POST.get | request.POST contains the POST variables. This function is used to get the variables that are sent with http post request. |
| objects.filter | Objects.filter is to get a matching result from the database, return a list of objects. |
| objects.get | It is a method of accessing objects in python. It returns the value associated with that object. |
| objects.raw | This method takes a raw SQL query, executes it, and returns a django.db.models.query.RawQuerySet instance. This RawQuerySet instance can be iterated over like a normal [QuerySet](https://docs.djangoproject.com/en/4.0/ref/models/querysets/" \l "django.db.models.query.QuerySet" \o "django.db.models.query.QuerySet) to provide object instances. |
| objects.set\_cookie | The set\_cookie() method in DjangoHttpResponse has a name: Name of the cookie.value: Value you want to store , int or string but it will return string. |
| request.COOKIES.get | Using request.COOKIES.get () Django also provides a method to get the desired value from the cookie. You can directly access that value using get method over request object. |
| objects.latest | Returns the latest object in the table based on the given field(s). |
| objects.create | To create and save an object in a single step, we can use the [create()](https://docs.djangoproject.com/en/4.0/ref/models/querysets/#django.db.models.query.QuerySet.create) method. |

Table 2.3



**Chapter 3**

**Results**

**3.1 Explore**