**BlockSQL**

A PROJECT REPORT SUBMITTED TO

**THE NATIONAL INSTITUTE OF ENGINEERING, MYSURU**

(An Autonomous College)



In partial fulfillment of the requirements for Project work (Database Laboratory), fifth semester

**Bachelor of Engineering**

**in**

**Computer Science and Engineering**

*Submitted by*

**Prathiksha M Urs (4NI16CS070)**

**Ram Prasad E Naik (4NI16CS075)**

**Sanchita Nandi (4NI16CS082)**

Under the Guidance of

**Mr.Yogesh M.J**Assistant Professor

**Ms. Ramya S**Assistant Professor

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**THE NATIONAL INSTITUTE OF ENGINEERING**

Mysore-570 008

2018-2019

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**THE NATIONAL INSTITUTE OF ENGINEERING**



***CERTIFICATE***

This is to certify that the project work entitled **BlockSQL** is a work carried out by **Prathiksha M Urs (4NI16CS070), Ram Prasad E Naik (4NI16CS075) and Sanchita Nandi (4NI16CS082)** in partial fulfillment for the project work (Database Laboratory), fifth semester, Computer Science & Engineering, The National Institute of Engineering **(**Autonomous Institution under Visvesvaraya Technological University, Belgaum) during the academic year 2018-2019. It is certified that all corrections and suggestions indicated for the Internal Assessment have been incorporated in the report deposited in the department library. The project work report has been approved in partial fulfillment as per academic regulations of The National Institute of Engineering, Mysuru.

**Internal Guides** **Professor & Head**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Ms.Ramya S  
Assistant Professor

Mr.Yogesh M J  
Assistant Professor

Dr. H D Phaneendra  
Professor and Head  
Dept. of CS&E  
NIE, Mysuru

**ACKNOWLEDGEMENT**

We would like to take this opportunity to express our profound gratitude to all those people who were directly or indirectly involved in the completion of this project. We thank each and everyone who encouraged us in every possible way.

We would like to thank our Principal Dr. G.Ravi for letting us be a part of this prestigious institution and letting us explore our abilities to the fullest.

We would also like to extend our sincere gratitude to Dr. H. D. Phaneendra, HOD of Computer Science Department, for being a source of inspiration and instilling an enthusiastic spirit in us throughout the process of project making.

We would also like to thank the Controller of Examinations, Dr. T N Shridhar and Mr. Ranganath for playing a crucial role in providing us with the required data.

We would like to express our heartfelt gratitude towards our project guides Mr. Yogesh M J and Ms Ramya S for their constant guidance, valuable knowledge and experience.

Thank you.

**-Prathiksha M Urs**

**-Ram Prasad E Naik**

**-Sanchita Nandi**

**Table of Contents**

**1.INTRODUCTION 1**

**2. SYSTEM ANALYSIS 3**

2.1 Existing system 3

2.2 Proposed system 4

**3. SYSTEM DESIGN 5**

3.1 Entity description and attribute details 5

3.2 ER Diagram 8

3.3 Schema of tables 9

**4. HARDWARE AND SOFTWARE REQUIREMENTS 12**

4.1 Software requirements 12

4.2 Hardware requirements 12

**5. SYSTEM IMPLEMENTATION WITH CODE SNIPPETS**

**AND SCREENSHOTS 13**

**6. TESTING 50**

6.1 System Testing 50

6.2 Unit Testing 51

6.3 Integration Testing 52

6.4 Validation Testing 52

6.5 Output Testing 52

6.6 Black-box Testing 52

6.7 User Acceptance Testing 53

**7. CONCLUSION AND FUTURE ENHANCEMENTS 54**

7.1 Conclusion 54

7.2 Future Enhancements 54

**8. REFERENCES 55**

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| Figure Number | Description | Page Number |
| 3.1 | Entity-Relationship diagram | 8 |
| 3.2 | Tables in Database | 9 |
| 3.3 | Course table description | 9 |
| 3.4 | Database\_admin table description | 9 |
| 3.5 | Login\_ledger table description | 9 |
| 3.6 | Marks table description | 10 |
| 3.7 | Student\_details table description | 10 |
| 3.8 | Teacher\_details table description | 10 |
| 5.1 | Function to get current database connection | 12 |
| 5.2 | Function to initialize database | 12 |
| 5.3 | Function to query database | 13 |
| 5.4 | Function to execute database | 13 |
| 5.5 | Function to make dictionaries | 13 |
| 5.6 | Use of indexes and views | 14 |
| 5.7 | Data provided by COE’s office | 15 |
| 5.8 | Adding missing columns in marks table | 15 |
| 5.9 | Providing unique USN | 16 |
| 5.10 | Update marks based on grade | 16 |
| 5.11 | Create student\_details table | 16 |
| 5.12 | Create teacher\_details table | 16 |
| 5.13 | Create database\_admin table | 17 |
| 5.14 | Create login\_ledger table | 17 |
| 5.15 | Create course table | 17 |

|  |  |  |
| --- | --- | --- |
| 5.16 | Set sem=8 in student\_details | 17 |
| 5.17 | Update student\_details section | 17 |
| 5.18 | Update student\_details email | 17 |
| 5.19 | Update student\_details phone\_no | 18 |
| 5.20 | Update student\_details year\_start | 18 |
| 5.21 | Update student\_details year\_end | 18 |
| 5.22 | Update student\_details fathers\_name | 18 |
| 5.23 | Update student\_details counselor\_teacher\_id | 18 |
| 5.24 | Update student\_details student\_name | 18 |
| 5.25 | Queries to compute sgpa and cgpa | 19 |
| 5.26 | Function to add blockchain information | 19 |
| 5.27 | Function to hash password | 20 |
| 5.28 | Function to wrangle course details | 20 |
| 5.29 | Course details obtained from COE’s office | 21 |
| 5.30 | Blockchain implementation | 22 |
| 5.31(a)(b) | Function to check integrity | 23-24 |
| 5.32 | Demonstration of integrity failure | 24 |
| 5.33 | Screenshot of user interface for integrity lost | 24 |
| 5.34 | Mail sent to database admin regarding data integrity lost | 25 |
| 5.35 | Function for previous block hash | 25 |
| 5.36 | Function for current block hash | 25 |
| 5.37 | Mail sent regarding failed attempts to login into student portal | 26 |

|  |  |  |
| --- | --- | --- |
| 5.38 | Screenshot of login ledger | 27 |
| 5.39 | Screenshot of successful request for transcript | 27 |
| 5.40 | Function for provisional marks card | 28 |
| 5.41(a)(b)(c) | Python file for provisional marks card | 29-30 |
| 5.42(a)(b)(c) | Python file for transcript | 30-31 |
| 5.43 | Makepdf function | 31 |
| 5.44 | Function to set password | 32 |
| 5.45 | Screenshot of provisional marks card | 33 |
| 5.46 | Screenshot of transcript | 34 |
| 5.47 | Screenshot of provisional marks card request generated mail | 35 |
| 5.48 | Screenshot of transcript request generated mail | 35 |
| 5.49 | Teachers Dashboard | 36 |
| 5.50 | Screenshot of insert marks of teachers dashboard | 36 |
| 5.51 | Screenshot of results affected after updation of marks by teachers | 37 |

|  |  |  |
| --- | --- | --- |
| 5.52 | Update student marks in teachers portal | 37 |
| 5.53 | Upload question papers and resources | 38 |
| 5.54 | Screenshot of login\_ledger | 38 |
| 5.55 | Screenshot of execute queries | 39 |
| 5.56 | Screenshot of check integrity | 39 |
| 5.57 | Screenshot of student admission | 40 |
| 5.58 | New blocks for student admission | 40 |
| 5.59 | Screenshot of current block hash=previous block hash | 41 |
| 5.60 | Screenshot of add teacher details | 41 |
| 5.61 | New teacher’s dashboard | 42 |
| 5.62 | Screenshot of student dashboard | 42 |
| 5.63 | Screenshot of previous results with no semester and grade specified | 43 |
| 5.64 | Screenshot of previous results with grade=PP | 44 |
| 5.65 | Screenshot of previous results with semester=4 | 45 |

|  |  |  |
| --- | --- | --- |
| 5.66 | Screenshot of previous results with sem=4 and grade=PP | 45 |
| 5.67 | Screenshot of request for provisional marks card | 46 |
| 5.68 | Update student details | 46 |
| 5.69 | Execute query for insert instead of update | 47 |
| 5.70 | Data reflected in student dashboard | 47 |
| 5.71 | Screenshot of sgpa analysis | 48 |
| 5.72 | Code snippet of sgpa analysis | 48 |
| 5.73 | Screenshot of internals analysis | 49 |
| 5.74 | Code snippet of internals analysis | 49 |

**Chapter 1**

**INTRODUCTION**

The BlockSQL has been designed to facilitate and help the Students, Teachers and Database Administrators of our college. It is a user-friendly, secure application implementing the concept of blockchain to provide students with ease of examination related facilities online and help smoothen the operations of the COE’s office.

We recognized that there were shortcomings in the current system of Examination such as requesting for provisional marks card, transcript and accessing the previous results, question papers and other resources, and hence, wanted to help by solving it.

We decided to choose this as our project because we wanted to solve an existing issue in the college. So, we approached our project guides, Mr. Yogesh M J and Ms. Ramya S, to seek their guidance and support. They encouraged us and helped us in approaching the HOD of Computer Science, Principal and the COE to obtain previous years data to solve the shortcomings in the system. They understood the problems faced by the students and the college, and gave us permission to obtain the meta data required to implement the features in our application.

**STUDENTS:**

The students can access and view their previous semester results and current semester results. Students can also view and download question papers of previous internals and examinations effortlessly. A request can also be made to obtain a copy of their provisional marks card and a copy of their transcript.

Data Analysis and visualization is used to generate useful graphs about student performance using matplotlib library from the provided data.

**TEACHERS:**

Teachers can insert as well as update student marks and upload question papers.

**DATABASE ADMINISTRATORS:**

Database administrator plays a crucial role in maintaining and ensuring integrity of database. The system provides complete control over the database system. They are allowed to execute required queries on the database. They also have information regarding the various logins by users and their time of logins. They can add details of new teachers and students during admission to the database.

**BLOCKCHAIN IMPLEMENTATION:**

The principles of Blockchain are used to ensure data security, integrity and consistency. It helps to see to it that the data is not tampered. This is done by adding the attributes of a block in a blockchain to the tables in the database(time\_stamp, prev\_block\_hash, block\_hash).

**Chapter 2**

**SYSTEM ANALYSIS**

**EXISTING SYSTEM**

The existing student portal does not have a provision for accessing previous results and current results. The students have to go to the college to check their results which is not feasible, time consuming and it is also a problem if they are out of town.

Presently, the students have to go to the college to request provisional marks card and transcript which is time intensive.

For obtaining provisional marks card, students have to initially collect the draft from the college wherein the students have to fill their personal details and marks. After that, students have to obtain a printout of the same and get it verified by the officials in their respective departments. Once the verification process is done, the Head of the Department has to sign the provisional marks card. Hence, this is a really long process.

For obtaining transcript, students have to download the form from the college’s website or obtain it at COE’s office, fill and submit the request form at the COE’s office and pay via challan. Requesting for the same cannot be done from anywhere else, but only by going to the college’s COE office. Students should also submit their provisional marks card of previous semesters for verifying the datawith students’ data present in the office**.** After the verification procedure, the transcript is signed by the COE and forwarded to the principal for signature.

Only after this whole procedure, the transcript is available for the student.

In the current system, students can access previous question papers and resources only via Intranet(only in campus) which is time consuming. It cannot be accessed anywhere outside the college limits.

The current database system does not use blockchain implementation for security and integrity purposes. Therefore, it may be tampered with, by updating student details or marks, that is if an intruder gains access to database administrator dashboard or the system’s root directory. Thus, there is a need for automating the system in few scenarios.

**PROPOSED SYSTEM**

With the proposed system we aim at solving many problems which the students and the COE’s office encounter by making use of the concepts of Blockchain and open-source libraries through a web application.

The students can access and view their previous and current results by logging in to the student portal anywhere and anytime and this tends to be more feasible than having to go to college to check their results as per the current system. This avoids the confusion and chaos caused by having to wait in queue to check the results on the notice board.

The students can request for provisional marks card and transcript using the student portal itself. They need not go the college and spend a lot of time for the same. Student details and marks will be auto-generated and the Head of the Department can sign the provisional marks card after which the students can go and collect it from the college.

The transcript also, like the provisional transcript can be requested from outside the college through the student portal. Hence the students need not wait in the college COE’s office for requesting the same. Once requested, payment can be done. The transcript is then automatically generated with accurate data of the respective student without any delay or need for verification. Since the blockchain is implemented in the marks table, the data related to marks is reliable and is sent to COE’s office for signature from COE and forwarded to principal’s office.Hence, the workload of the staff in the office is also reduced.

Students can also view and access previous question papers and resources in the student portal outside the college limits as well. They can access it from anywhere using the portal which is easier.

The proposed systemuses blockchain implementation for enhanced security and data integrity in the student\_details and marks table. This makes it hard for an intruder to compromise the database or modify any data without proper privileges or reason in the system, as a ledger of every action is maintained.

The proposed system helps in attaining an organized, structured way of handling certain glitches in the current system and hence, enhances it. This system not only benefits the students, but also the staff by helping them easen the procedure involved in the generation of provisional marks card and transcript.

**Chapter 3**

**SYSTEM DESIGN**

**ENTITY DESCRIPTION AND ATTRIBUTE DETAILS**

**student\_details:**

* sgpa1- float (4,2)
* sgpa2- float (4,2)
* sgpa3- float (4,2)
* sgpa4- float (4,2)
* sgpa5- float (4,2)
* sgpa6- float (4,2)
* sgpa7- float (4,2)
* sgpa8- float (4,2)
* cgpa- float (4,2)
* usn- varchar (10) not null
* student\_name- varchar (30) not null
* pwd- varchar (100) not null
* sem- int (1) not null
* section- varchar (1) not null
* branch- varchar (45) not null
* email- varchar (30) unique not null
* phone\_no- int (10) unique not null
* year\_start- date not null
* year\_end- date not null
* fathers\_name- varchar (50) not null
* counselor\_teacher\_id- varchar (10) foreign key
* time\_stamp- timestamp not null
* prev\_block\_hash- varchar (100) not null
* block\_hash- varchar (100) not null

**teacher\_details:**

* teacher\_id- varchar (10) primary key
* teacher\_name- varchar (30) not null
* pwd- varchar (100) not null
* designation- varchar (30) not null
* doj- date not null
* branch- varchar (45) not null
* phone\_no- int (10) unique
* email- varchar (30) unique

**database\_admin:**

* admin\_name- varchar (30) not null
* admin\_id- varchar (10) primary key
* pwd- varchar (100) not null
* email-varchar (30) unique not null

**marks:**

* result\_year- varchar (10) not null
* result\_type- varchar (7) not null
* branch- varchar (3)
* sem- int (1) not null
* section- varchar (1)
* usn- varchar (10) not null
* course\_code- varchar (6) not null
* grade- varchar (2)
* grade\_point- float (4,2)
* credits- float (2,1)
* int1- int (2)
* int2- int (2)
* int3- int (2)
* cie- int (2)
* see- int (3)
* total\_marks- int (3)
* time\_stamp- timestamp not null
* prev\_block\_hash- varchar (100) not null
* block\_hash- varchar (100) not null

**course:**

* course\_code- varchar (6) primary key
* course\_name- varchar (30) not null
* credits- float (2,1) not null

**login\_ledger:**

* usn- varchar (10) not null
* user\_type- varchar (10) not null
* login- timestamp not null
* logout- timestamp

**ENTITY RELATIONSHIP DIAGRAM**

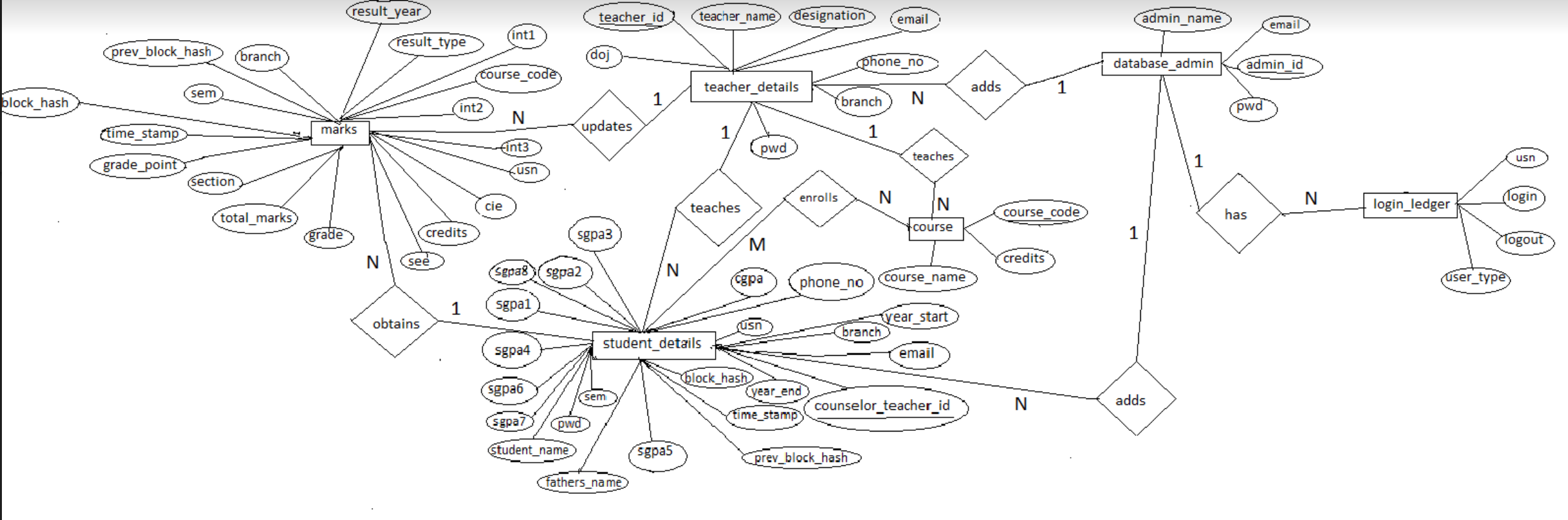


Figure 3.1

**Screenshots of Description Tables**

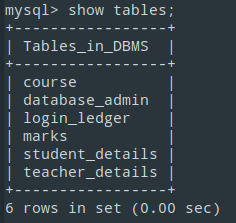
****

Figure 3.2

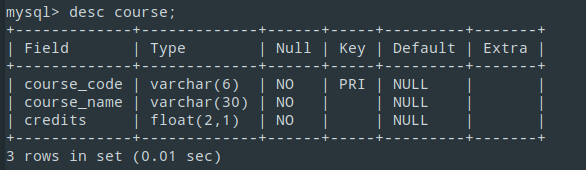
****

Figure 3.3

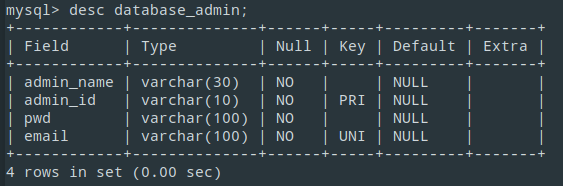
****

Figure 3.4

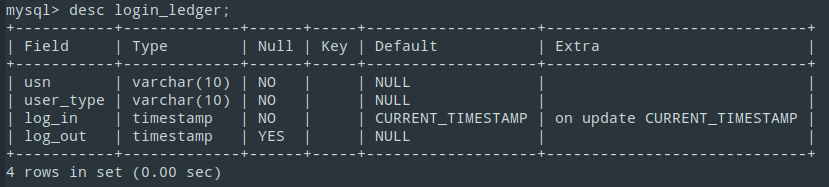
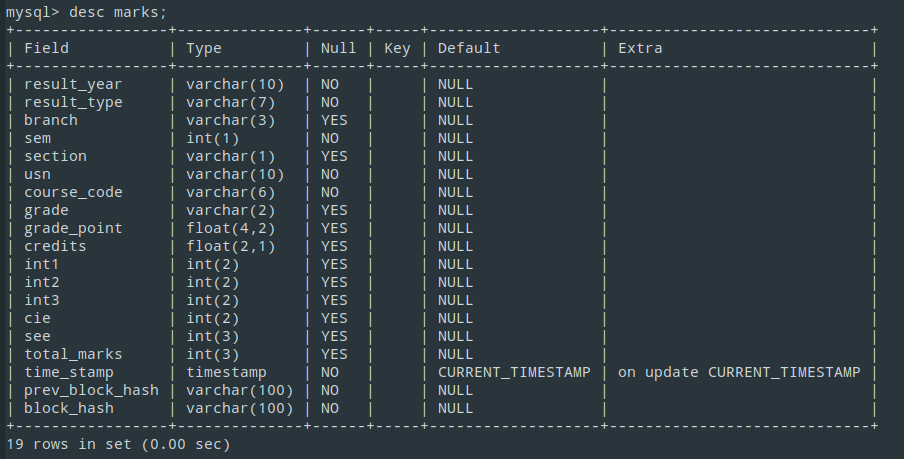
Figure 3.5

Figure 3.6

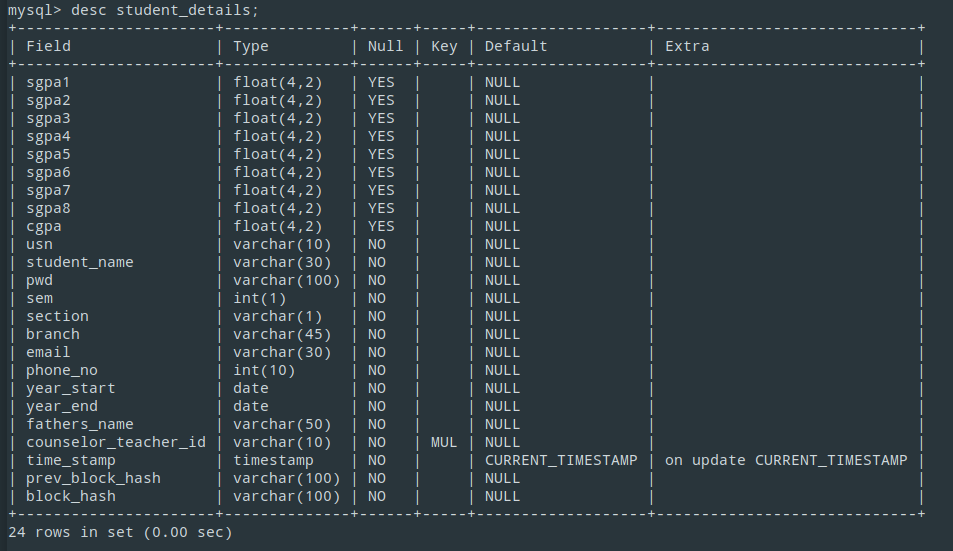
****

Figure 3.7

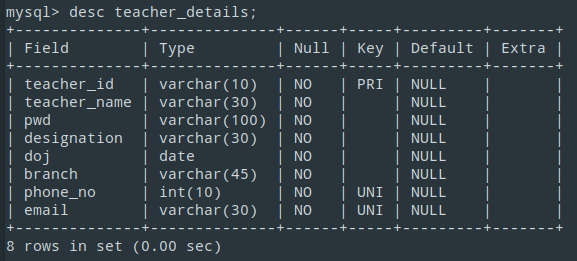


Figure 3.8

**Chapter 4**

**HARDWARE AND SOFTWARE REQUIREMENTS**

**SOFTWARE REQUIREMENTS**

* Client Side

Web Browser: Google Chrome, Firefox, Internet Explorer.

* Server Side:

Front end: HTML 5.0, CSS

Back end: SQLite3

Scripting Language: Python 3.x

Modules: Flask, werkzeug, tabula, pdfkit, Pypdf2, passlib, email, smtplib, pathlib, OS, shutil, time, itertools, datetime

**HARDWARE REQUIREMENTS**

OS: Windows / Linux /OSx

Memory: 1 GB RAM

Disk: 2 GB HDD Space

Network: 512 Kbps or faster

**Chapter 5**

**SYSTEM IMPLEMENTATION**

* **Server:**

Flask depends on two external libraries: the Jinja2 template engine and the Werkzeug WSGI toolkit.

Flask was picked as a server due to the following reasons:

* Flask has a lightweight and modular design, so it easy to transform any web framework.
* Built-in development server and fast debugger.
* Support for secure cookies (client-side sessions).
* Object-Relational Mapping agnostic: Ability to plug in ORM like SQLite database.
* Easy to deploy and scale Flask in production.
* Integrated support for unit testing.
* **The Database Management System**:

SQLite3 was picked due to its lightweight nature and support for concurrency for handling multiple request at the same instance and its support with flask.

Following routine functions are used to perform database connection and operations on the SQLite database.

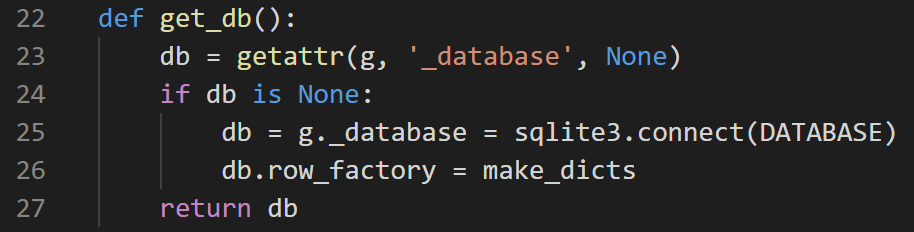


Figure 5.1

To use the database, the application must either have an active application context or create an application context itself. At that point the get\_db function can be used to get the current database connection.

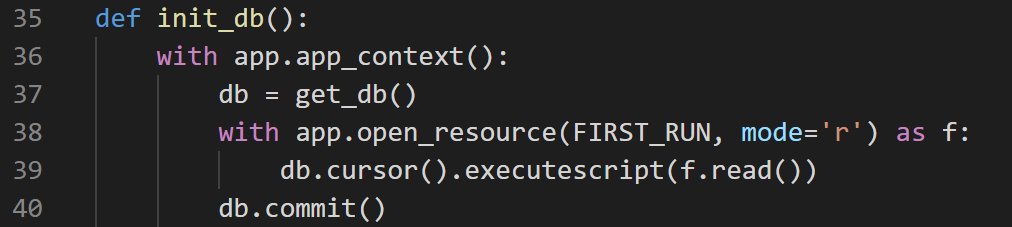


Figure 5.2

This function initializes the database by executing the SQL create script for various tables on the first run of the application this is called when route(/init) is loaded.

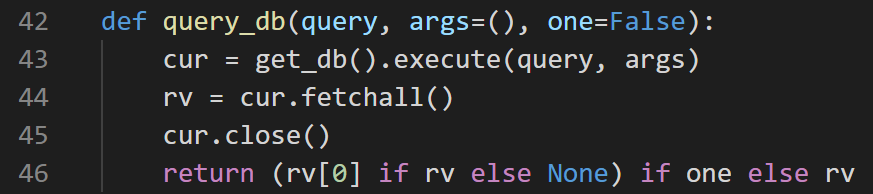


Figure 5.3

Handy function which queries the database and returns the rows in the form of list of dictionaries.

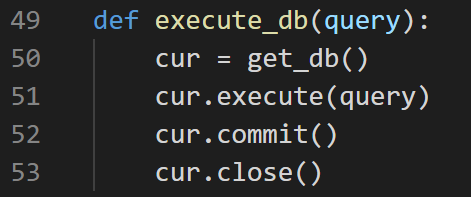


Figure 5.4

execute\_db function is used to execute an update or delete queries on the database.

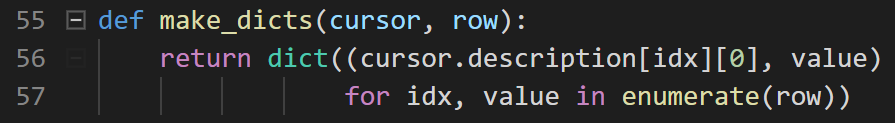


Figure 5.5

To simplify working with SQLite, a row factory function is useful. It is executed for every result returned from the database to convert the result into dicts for this database connection, which are much nicer and easier to deal with.

* **Tables**: Following tables form the base of the database and their integrity is maintained by implementation principles of blockchain in them.

student\_details: Information of students.

teacher\_detail: Information of teachers.

marks: Marks of all internals and exams.

database\_admin: Stores the information of administrators.

login\_ledger: Stores the information about login and logout history of the all types of users.

course: Stores information about the course codes their respective name and credits.

* **Indexes**: Special lookup tablesthat the database search engine can use to speed up data retrieval, slows down updates which isn't a problem since we insert instead of update due to implementation of blockchain to maintain a ledger of changes.

Index are implemented for student\_details,teacher\_detail,database\_admin tables to speedup lookup.

* **Views**: Summarizes data and restrict modifications from login\_ledger table to display all history.

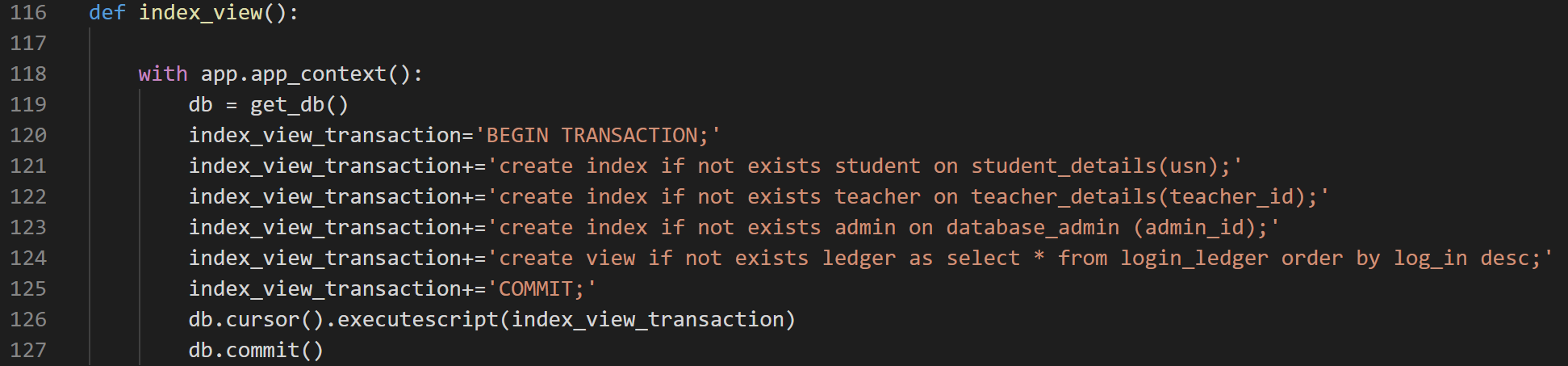


Figure 5.6

* **Python Modules**: Following modules were utilized in implementing features of application.

Flask: Microframework for Python based on Werkzeug and Jinja 2.

werkzeug: WSGI utility library for Python, base of frameworks for Flask.

pdfkit: Converts HTML to pdf file.

Pypdf2: Sets password to the pdf file output by pdfkit.

matplotlib: Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

It is used to generate performance reports of students.

tabula: Scrape course data such as course code, name and credits from table in timetable pdf file.

passlib: Hash/verify passwords and block chain hashes.

email: Sends mails from within the application, with attachment if required.

smtplib: Defines an SMTP client session object that can be used to send mail to any Internet machine with an SMTP or ESMTP listener daemon.

pathlib: Handles Operating System independent file paths.

os: Handles relative file Paths independent of Operating System.

shutil: Offers a number of high-level operations on files and collections of files. In particular, the application uses it for file copying and removal operations.

datetime: Supplies classes for manipulating dates and times. In particular, datetime.utcnow().strftime('%Y-%m-%d %H:%M:%S.%f') is used to obtain the current timestamp upto microsecond precision to log various events in the application, namely login, failed login attempt and block time(time\_stamp).

itertools: Standardizes a core set of fast, memory efficient tools. In particular, chain function is used to flatten list of lists during wrangling from timetable using tabula.

* **Database Preparation, Wrangling and Sanitation**:

After obtaining the requested data from the COE’s office, it had to be cleaned, post processed, prepared and missing data had to be logically inserted or wrangled to generate a database which is consistent with the application.

1.The provided data was in excel format(.xlsx), it was saved into a Comma Separated Value(.csv) and later imported to SQLite Database from the built-in import command.

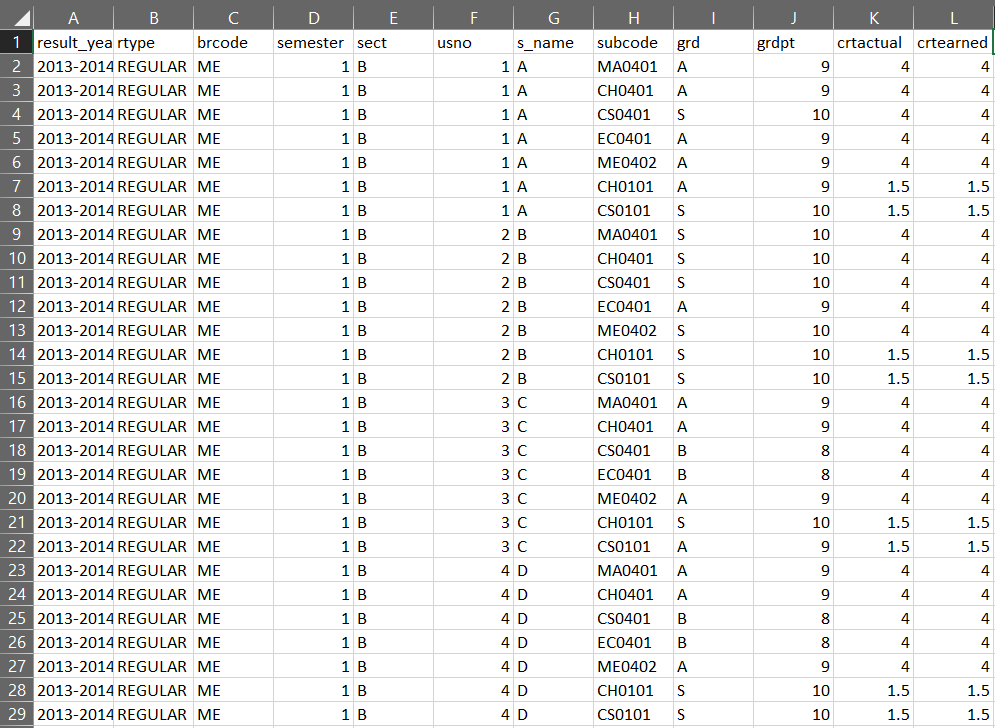


Figure 5.7

2.Certain missing columns were added such as int1, int2, int3, cie, see, total\_marks.



Figure 5.8

3.Column names were edited according to the database schema.

Edited columns from columns column\_number to: result\_year, result\_type, branch, sem, section, usn, course\_code, grade, grade\_point, credits.

4.Provided data had same usn (Username) for two sections of a department. Our system works on unique usn for each student.Query to assign unique usn to each student was executed.



Figure 5.9

5.student\_name column was removed, as student information would be stored in student\_details table and queried thereof.

6.Marks data was calculated based on grade obtained. Queries to update values of int1, int2, int3, cie, see, total\_marks based on grade were executed.

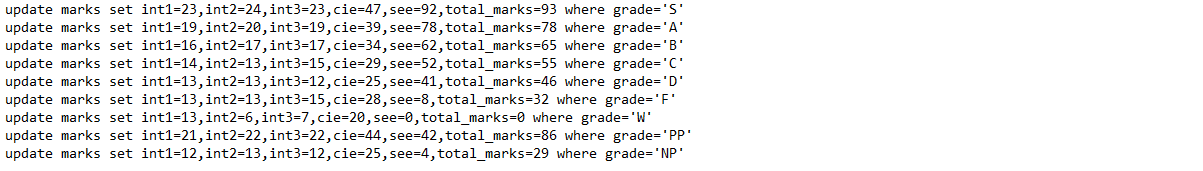


Figure 5.10

7.student\_details, teacher\_details, database\_admin, login\_ledger and course tables were created.

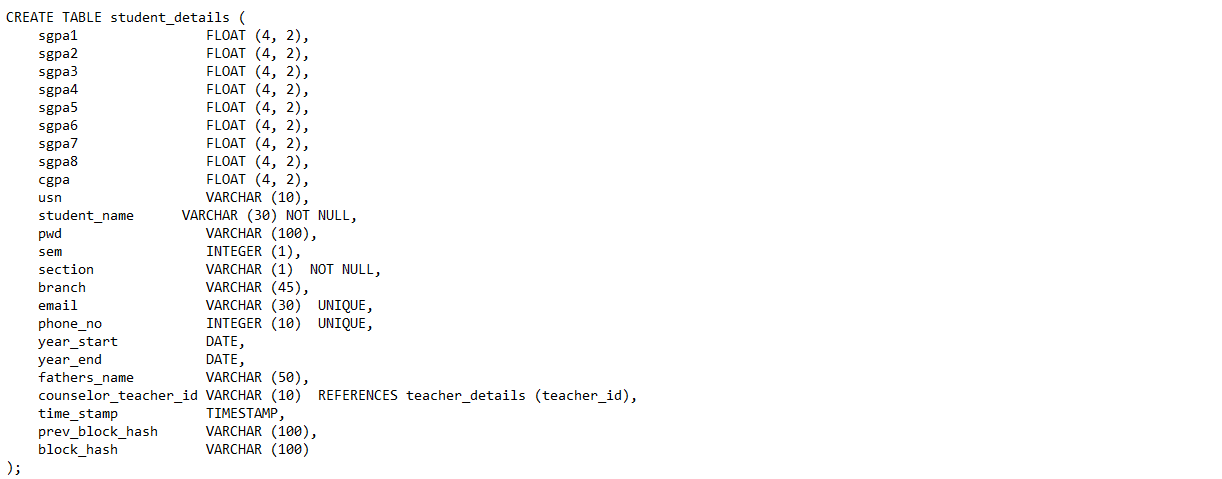


Figure 5.11



Figure 5.12



Figure 5.13



Figure 5.14



Figure 5.15

8.Updated sems column value to 8, denoting completion of all 8 semesters in student\_details table.



Figure 5.16

9.sections were updated to B for usn 1-5 and A for usn 6-10.



Figure 5.17

10. email, phone\_no, year\_start, year\_end, fathers\_name, counselor\_teacher\_id,student\_name were updated in student\_details table.student\_details table now holds complete student details.

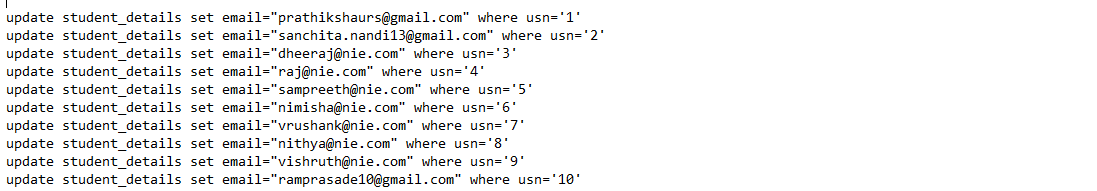


Figure 5.18

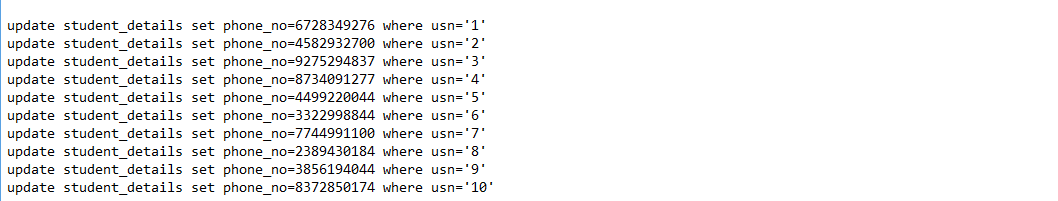


Figure 5.19



Figure 5.20



Figure 5.21

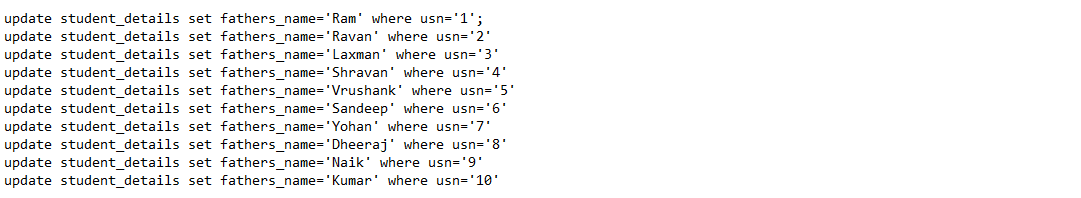


Figure 5.22

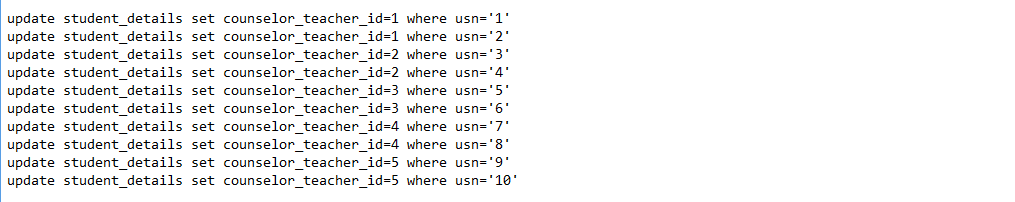


Figure 5.23

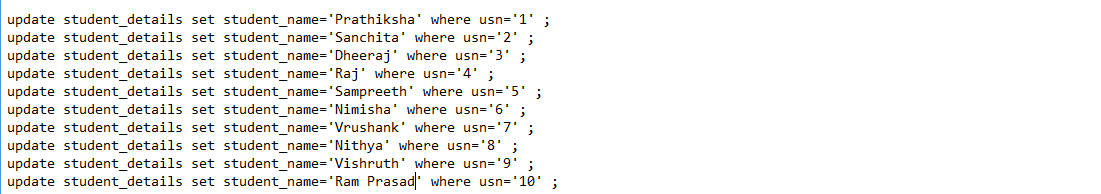


Figure 5.24

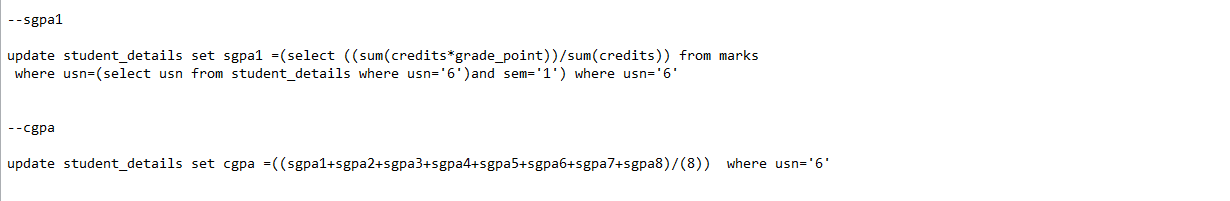
11. SGPA of all eight semesters and CGPA for all the students were computed with the help of queries. Below is a sample query for SGPA and CGPA computation. Similarly, SGPA and CGPA for all semesters and students were calculated.

Figure 5.25

12. Principles of blockchain were implemented in the database by adding Blockchain attributes as columns into student\_details and marks table.

time\_stamp,prev\_block\_hash,block\_hash

13.A separate python file data\_processing.py was created to wrangle data and prepare existing database.

14.Under *@app.route('/add\_blockchain\_info/<table>')* route a function *add\_blockchain\_info* was defined to add data into blockchain attributes of both marks table and student\_details including hashed password.

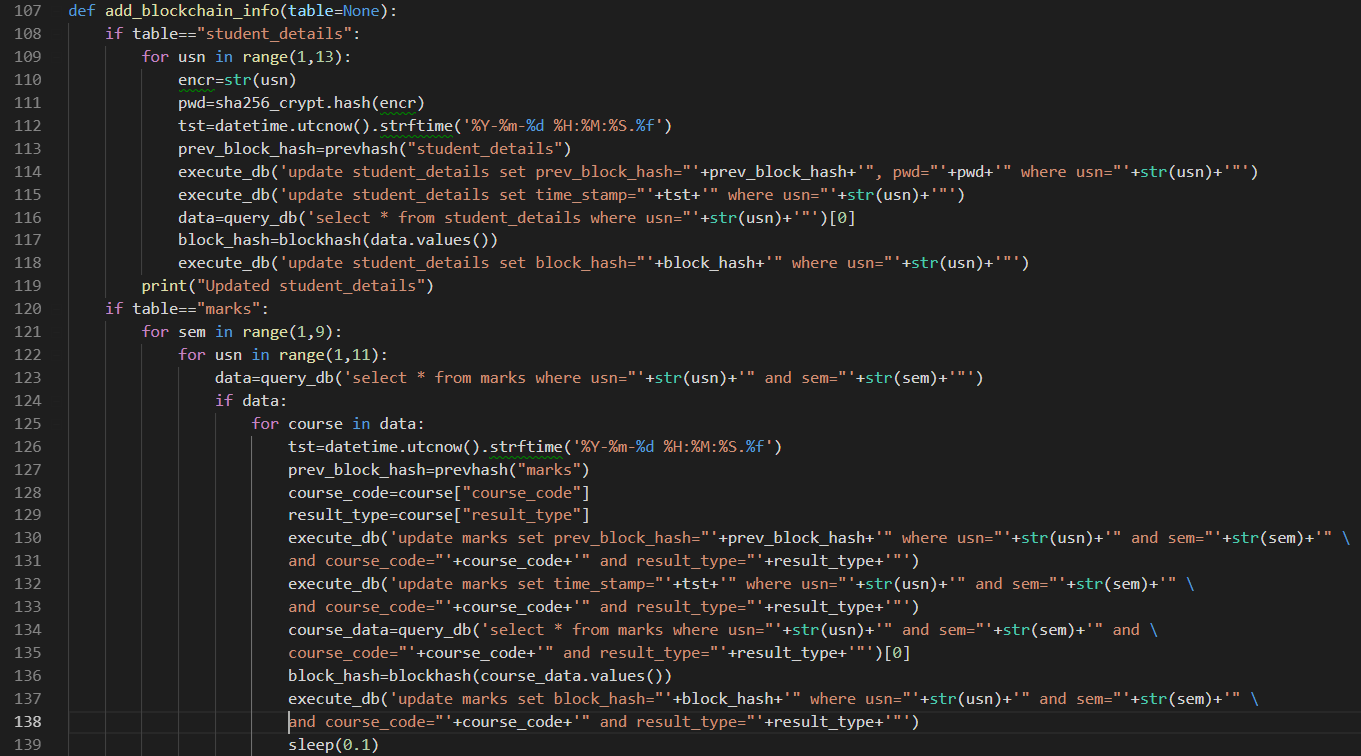


Figure 5.26

15.Under *@app.route('/pwd\_database')* route a function *pwd\_database* was defined to hash passwords to database\_admin and teacher\_details.

Figure 5.27

16.Course data such as course\_name, course\_name and credits were not requested initially but were needed for generation of transcript, provisional marks card and displaying results.

17.Time table of mechanical department was downloaded from college website, data from tables of timetable was *wrangled* using a python module *tabula* and inserted into course table.

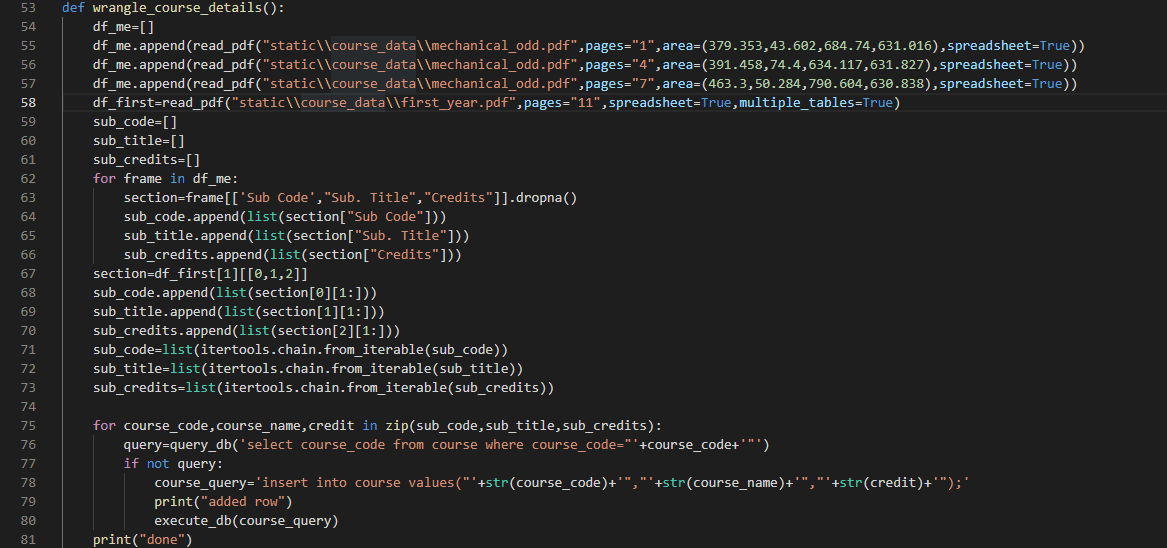


Figure 5.28

18.There were still some missing course details. On requesting and receiving complete course data, wrangled data from course table was deleted and complete course data was imported into course table from a Comma Separated Value (.csv) file.

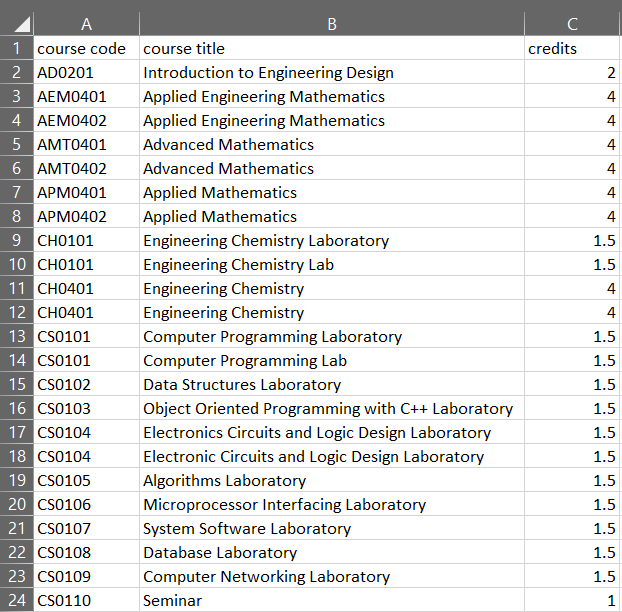


Figure 5.29

19.The database is now prepared to work with the application.

* **Implementation of Blockchain:**

The principles of blockchain state that by design:

* A blockchain is resistant to modification of the data. It is "an open, distributed ledger that can record transactions between two parties efficiently and in a verifiable and permanent way".
* Once recorded, the data in any given block cannot be altered retroactively without alteration of all subsequent blocks.



Figure 5.30

* Each block has a timestamp and a link to the previous block forming a chronological chain reinforced through cryptography ensuring the records cannot be altered by others. Due to the transactions needing multiple parties’ authorization before acceptance. To hack this system requires hackers to take control of all the nodes in the network and would need the enormous computation power to solve the sha256 hashing mechanism.

We took inspiration from these design principles and clubbed them into SQL to enhance security.

We do that by adding 3 attributes of a Blockchain to the database tables and treating each row as a block:

1.**time\_stamp:** Holds the timestamp of the creation of the block.

2.**prev\_block\_hash:** Holds the block\_hash of the previous block.

3.**block\_hash:** Holds the hash of the current block after hashing.

SQL is known for its functionality, reliability and scalability, but it is known that SQL databases have no mechanism to easily:

1. **Maintain a ledger of changes**: Since changes are to be allowed for student details and marks of the student, but by the principles of blockchain modifying an existing block is not permitted. The application allows updation of student details and marks through an Web User Interface. Hence a ledger of changes is maintained by inserting a new block with the changes and current timestamp instead of updating the existing block.
2. **Verify Data originality and integrity**: If an intruder is successful in gaining access to the administrator system and updates an existing block assuming it would go unnoticed.

In such situations, to verify data integrity, the database administrator has a check integrity feature which checks two integrity critical tables student\_details and marks for any integrity failure in two passes:

1.**Verifying block\_hash of each block(row):** At the time of creation of a block, the block\_hash is calculated by appending all the values and known salt, hence the passlib’s verify function can be used to verify the block by verifying the appended values of current state of block with the block\_hash value for the block.

2.**Verifying the blockchain:** After each of the block values are in integrity, next logical step is to check if the block of chains are in order. To do that we need to query the blockchain with time\_stamp descending and check if the current block hash is equal to the next block’s prev\_block\_hash.

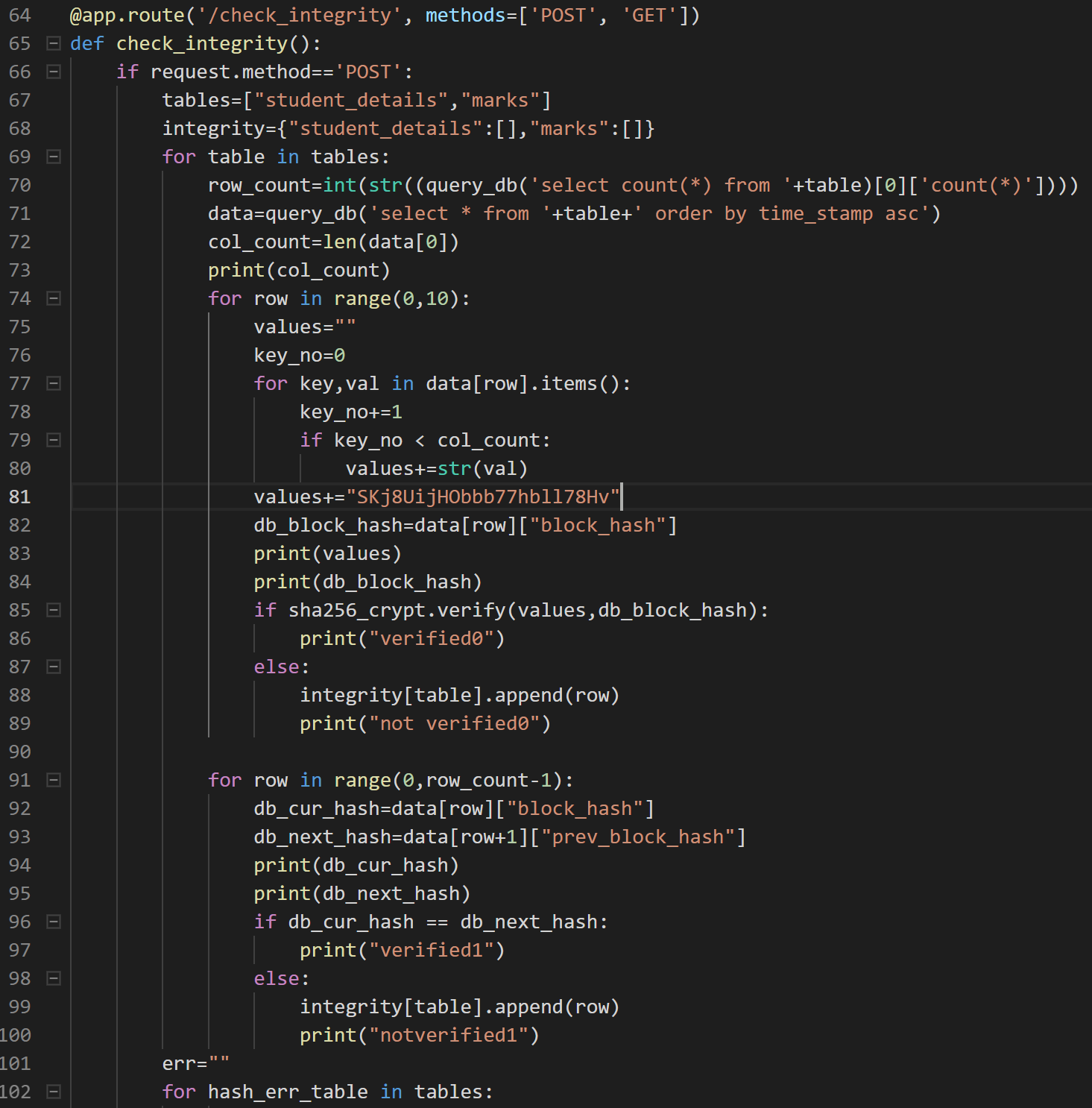


Figure 5.31(a)

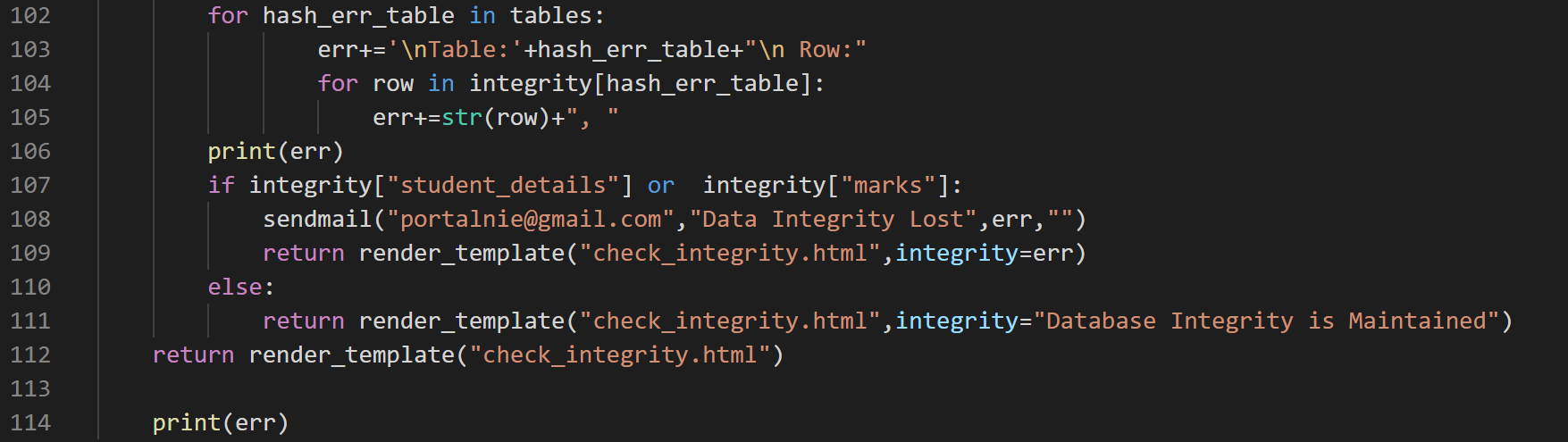


Figure 5.31(b)

The rows where the integrity fails are appended to a error(err) list and displayed onto the check integrity interface and also mailed to the Database Administrator.

**Demonstration of Integrity failure:**

Updating data of student\_details should cause integrity to fail at the respective rows.

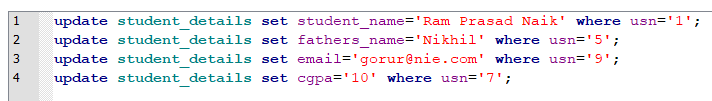


Figure 5.32

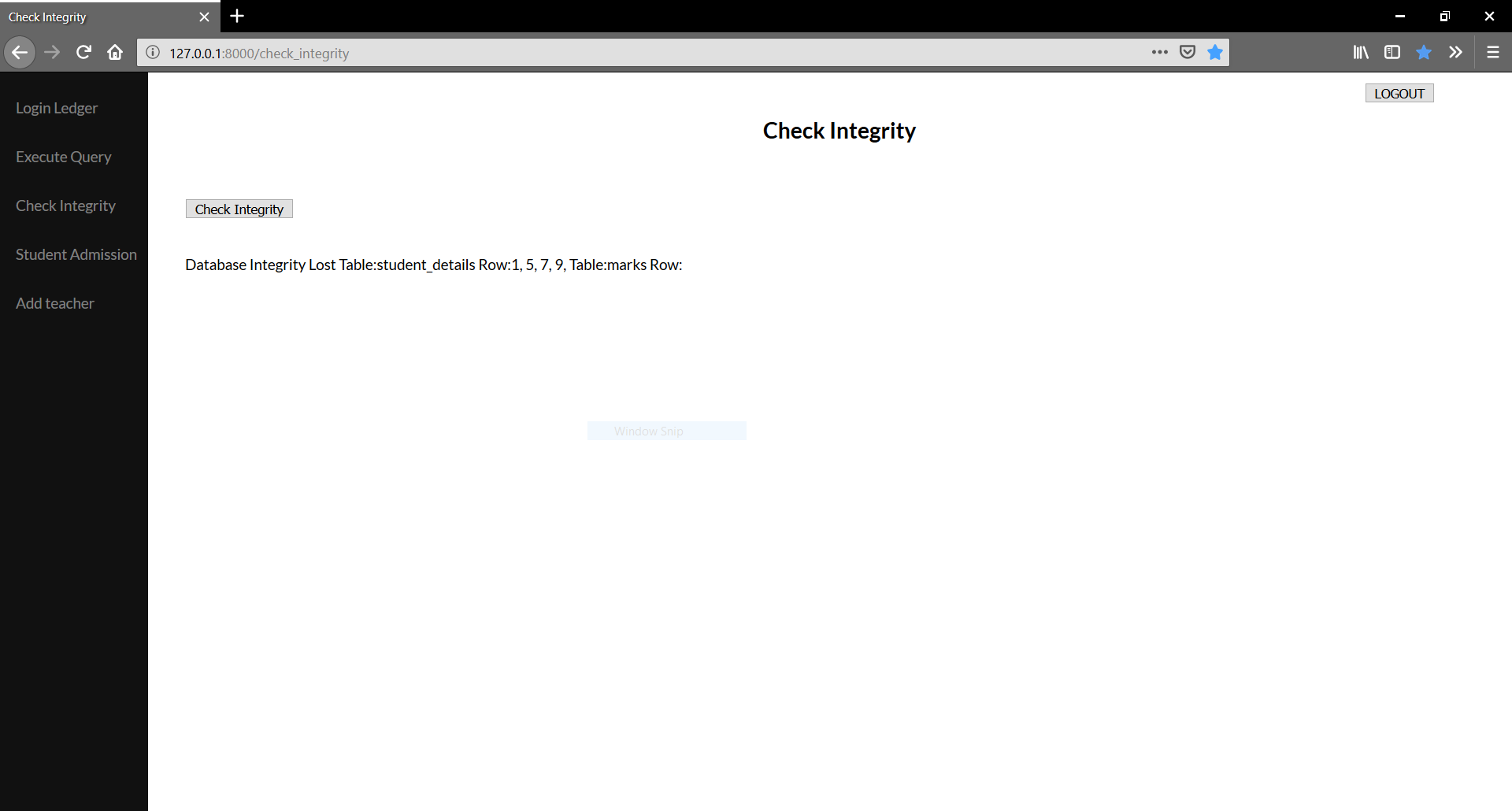


Figure 5.33

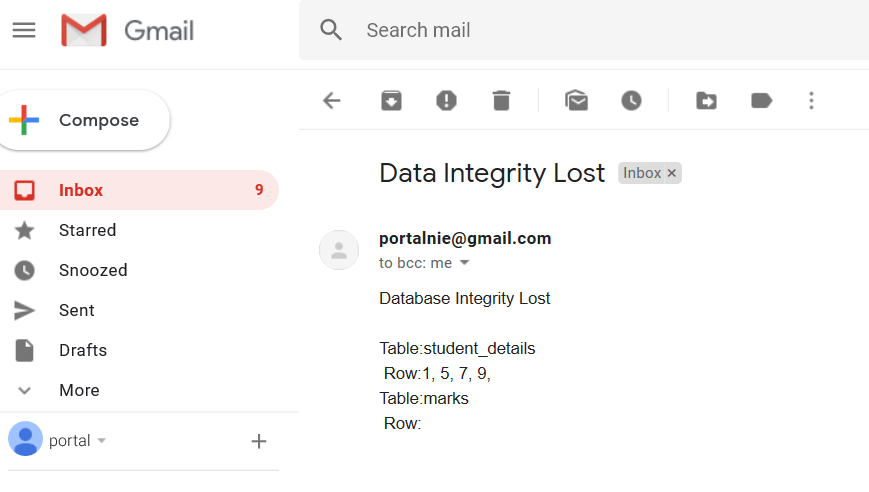


Figure 5.34

These are achieved by writing some routine blockchain methods:

1. **To generate previous block hash(prev\_block\_hash)**: return the block\_hash of the latest block(row) ordered by time\_stamp in the database.

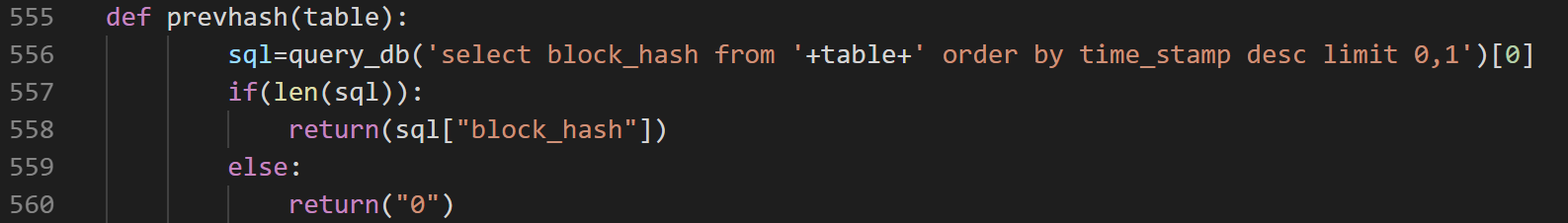


Figure 5.35

1. **To generate block hash(block\_hash):** return the hash of the block(row) after appending all its values and adding a fixed salt and passing it to passlib’s *sha256\_crypt.hash(concat)* function to calculate the blockhash.

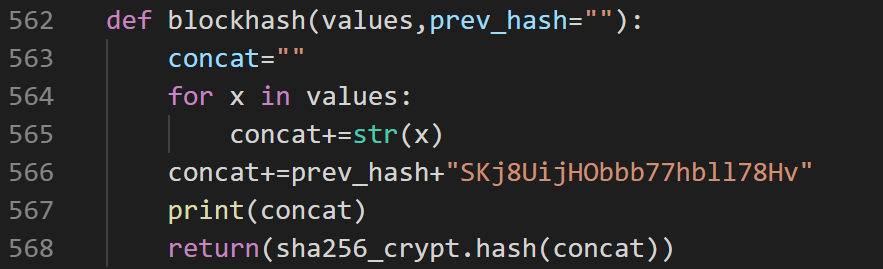


Figure 5.36

* **Security:**

An important aspect of modern applications is its security. With our application capable of handling sensitive and critical data about student details and marks, it raises the bar for security of user access and database. This forms the first layer of protection over user account access and maintain data integrity against potential intruders or hackers.

* **Hashing passwords**: This forms the first sublayer of Security. Passwords of Users (Admins,Teachers and Students) are hashed by sha256 hashing algorithm using passlib, a python hashing library. This ensures passwords are not stored in plain text, moreover the salt used in hashing is unknown to the programmer or database administrator even and is variable for each generation of password. Hence, a brute force attack has to be done for each row separately which alone is hard for a regular Personal Computer as the default rounds of hashing is suitable higher than the computing power of a average Personal Computer. Hence the password stored in the database is undecryptable, but can be verified with plain text entered by the user by a built in verify hash function.
* **Notification after failed login attempts**: This forms the second sublayer of Security. This sublayer prevents a brute force attack by limiting the attempts of failed login to three.

A mail is sent from the administrator to the mail account of the user, stating the failed attempts and the timestamp of the event.

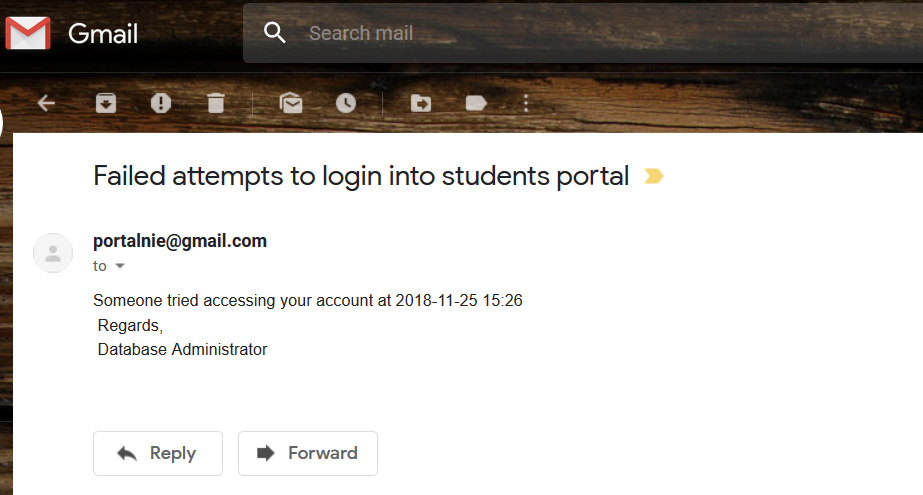


Figure 5.37

* **Login ledger**: This forms the third sublayer of Security. A ledger of all types of user login details is maintained and monitored by the database administrator for the purpose of tracing any suspicious or unusual activity from any user. Thus, contacting the user for further verification or security measures. This security feature implemented by the concept of views in database for the table login\_ledger thus restricting modification.

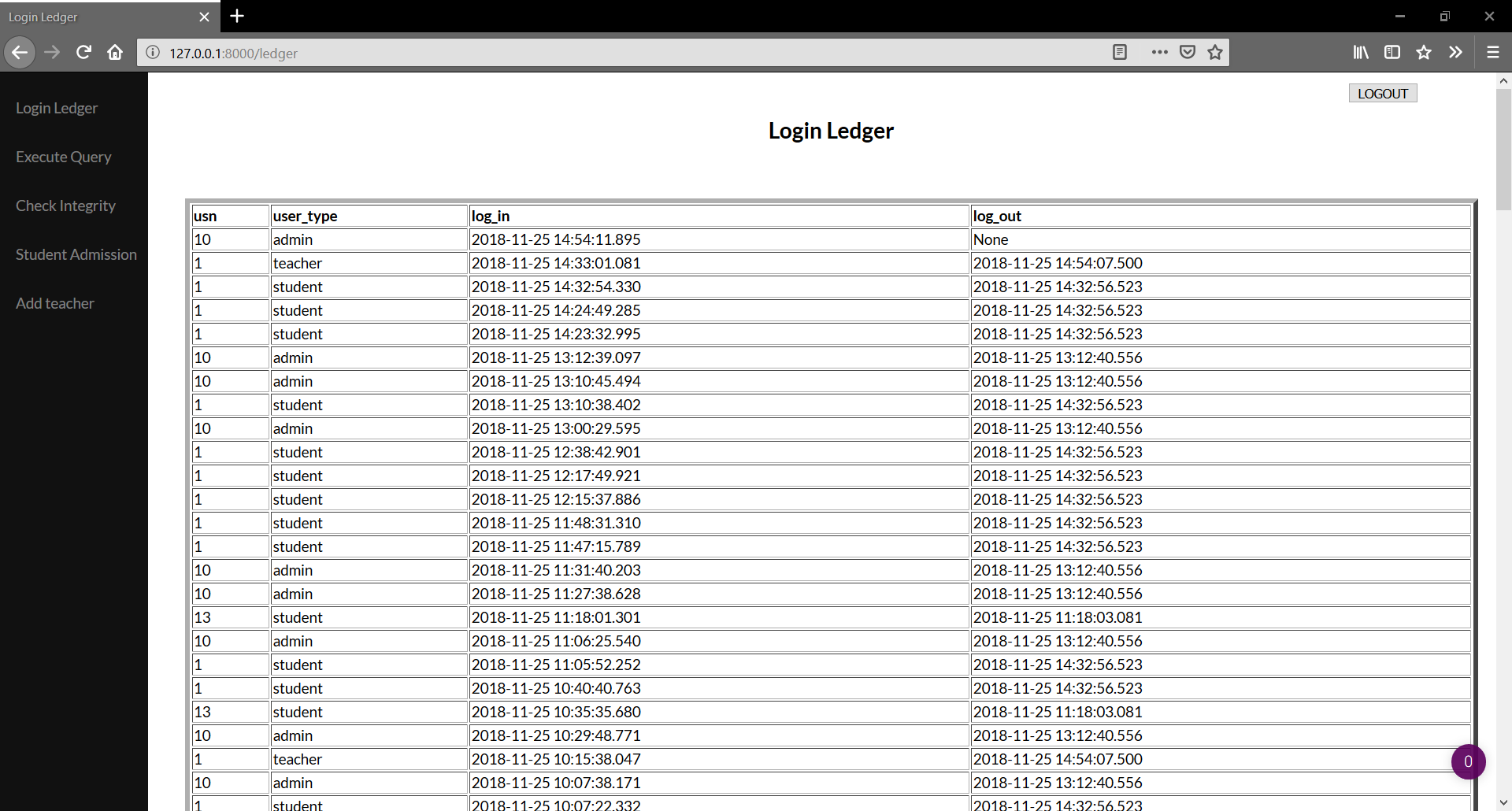


Figure 5.38

* **Generating the Transcript and Provisional marks cards**

1.After logging in, student can request for either Provisional marks cards and Transcript, by navigating to the respective page.

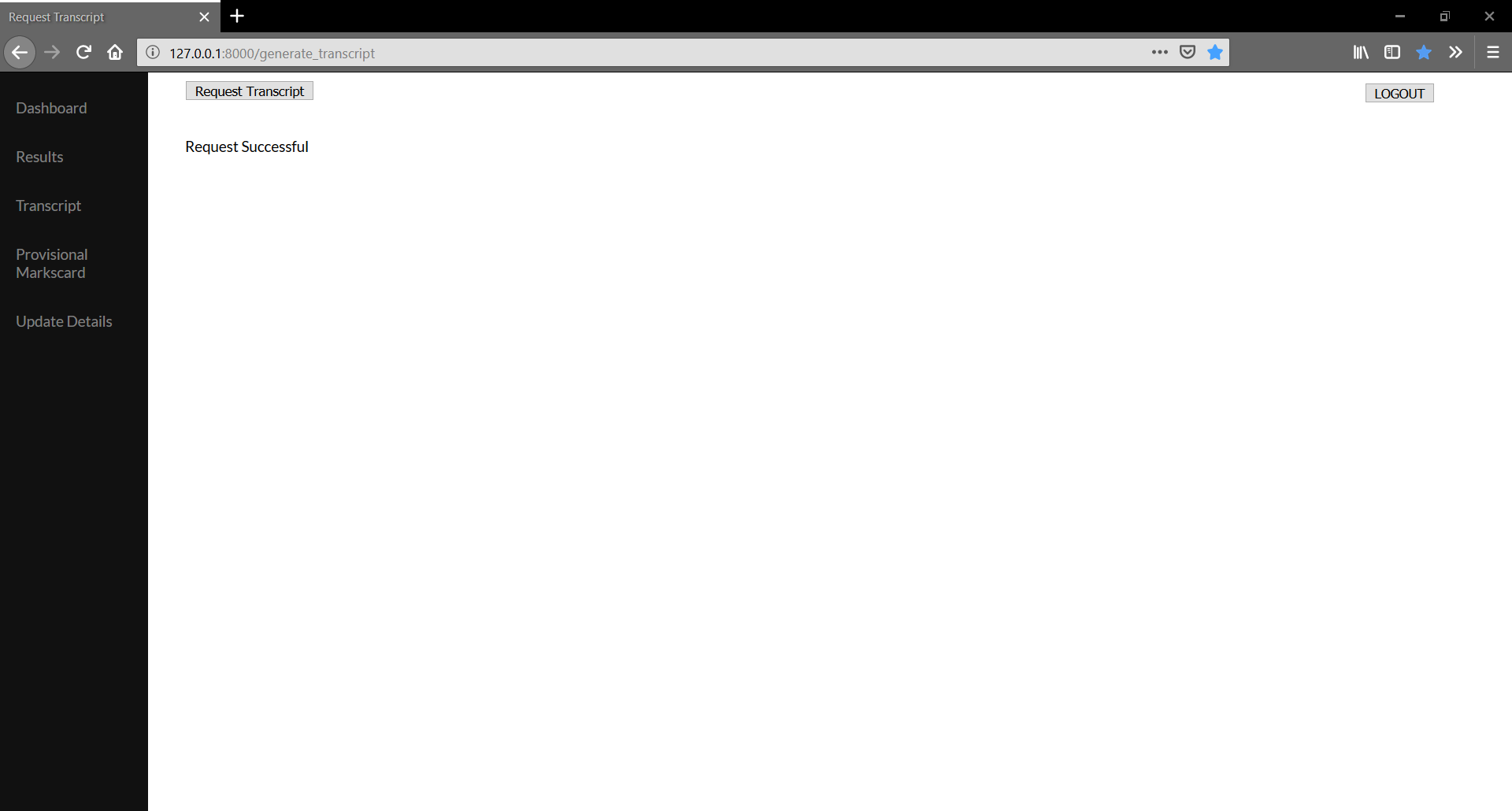


Figure 5.39

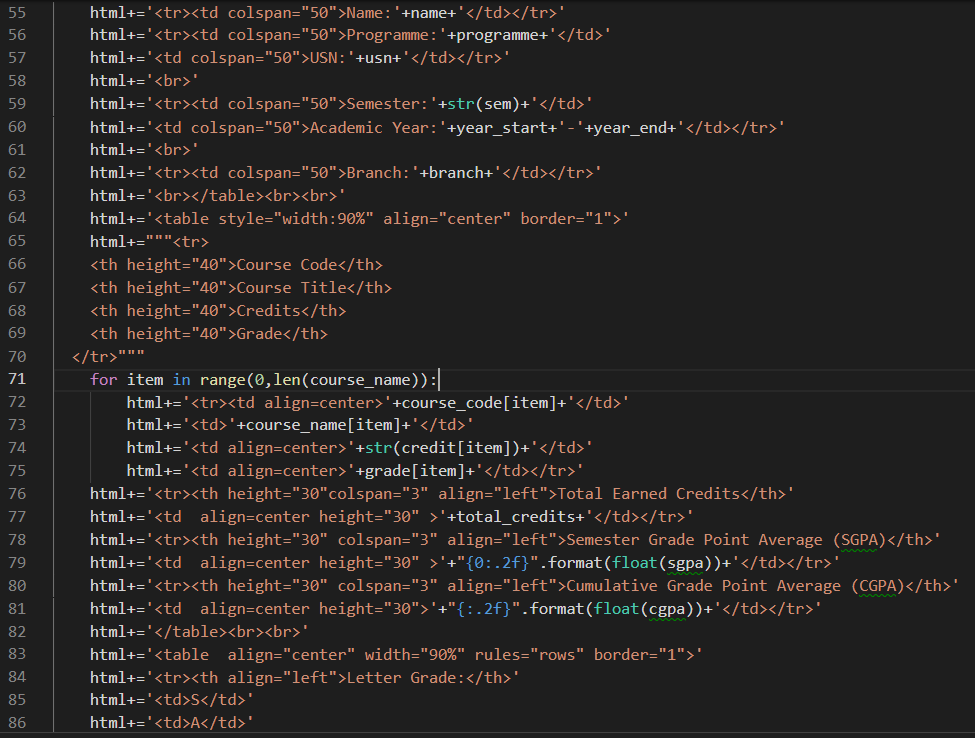
2. Once the request is received at the backend, the student’s details, course and its corresponding marks data is queried from the database and passed to makepdf function to process the complete request.



Figure 5.40

3. The queried data is sent to provisional\_html\_generate.py or transcript\_html\_generate.py to generate a html file of provisional marks card or transcript from the data received with python’s file handling function.

Figure 5.41(a)

Figure 5.41(b)

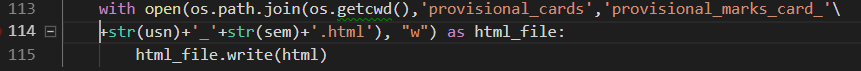


Figure 5.41(c)

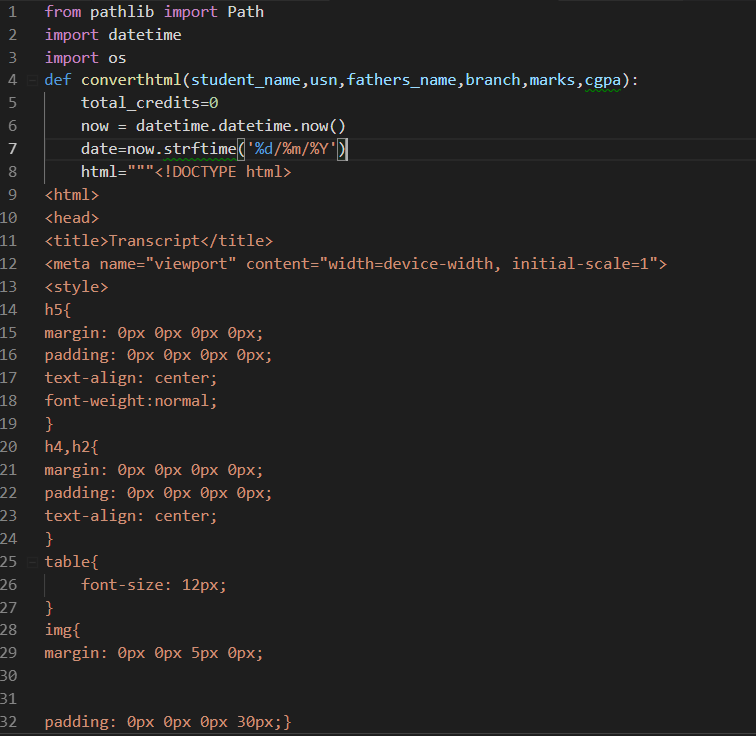


Figure 5.42(a)



Figure 5.42(b)

Figure 5.42(c)

4.pdfkit module’s pdfkit.from\_file(html\_path,pdf\_path) is called to generate pdf file from the html file generated.



Figure 5.43

5.PyPDF2 module’s set\_password(pdf\_path\_transcript,usn,usn) is called to set password for the generated pdf.

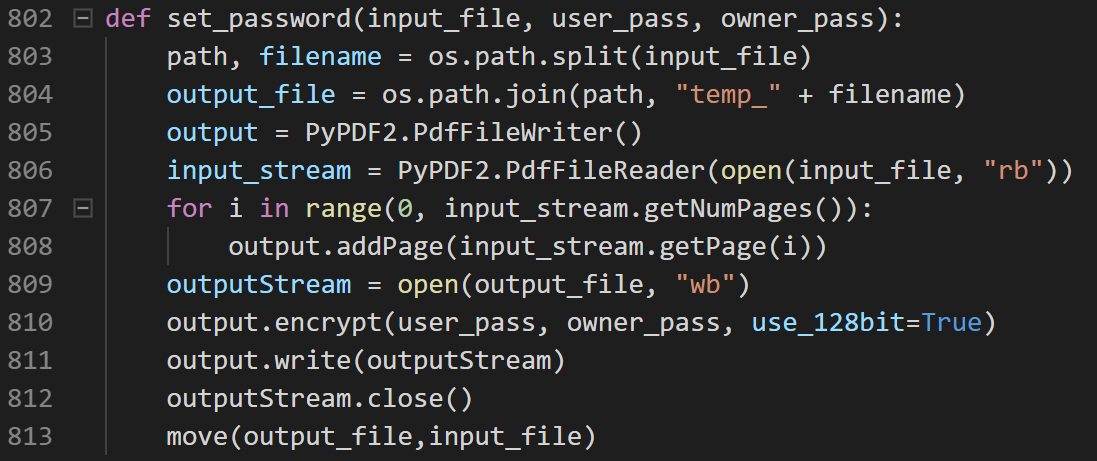
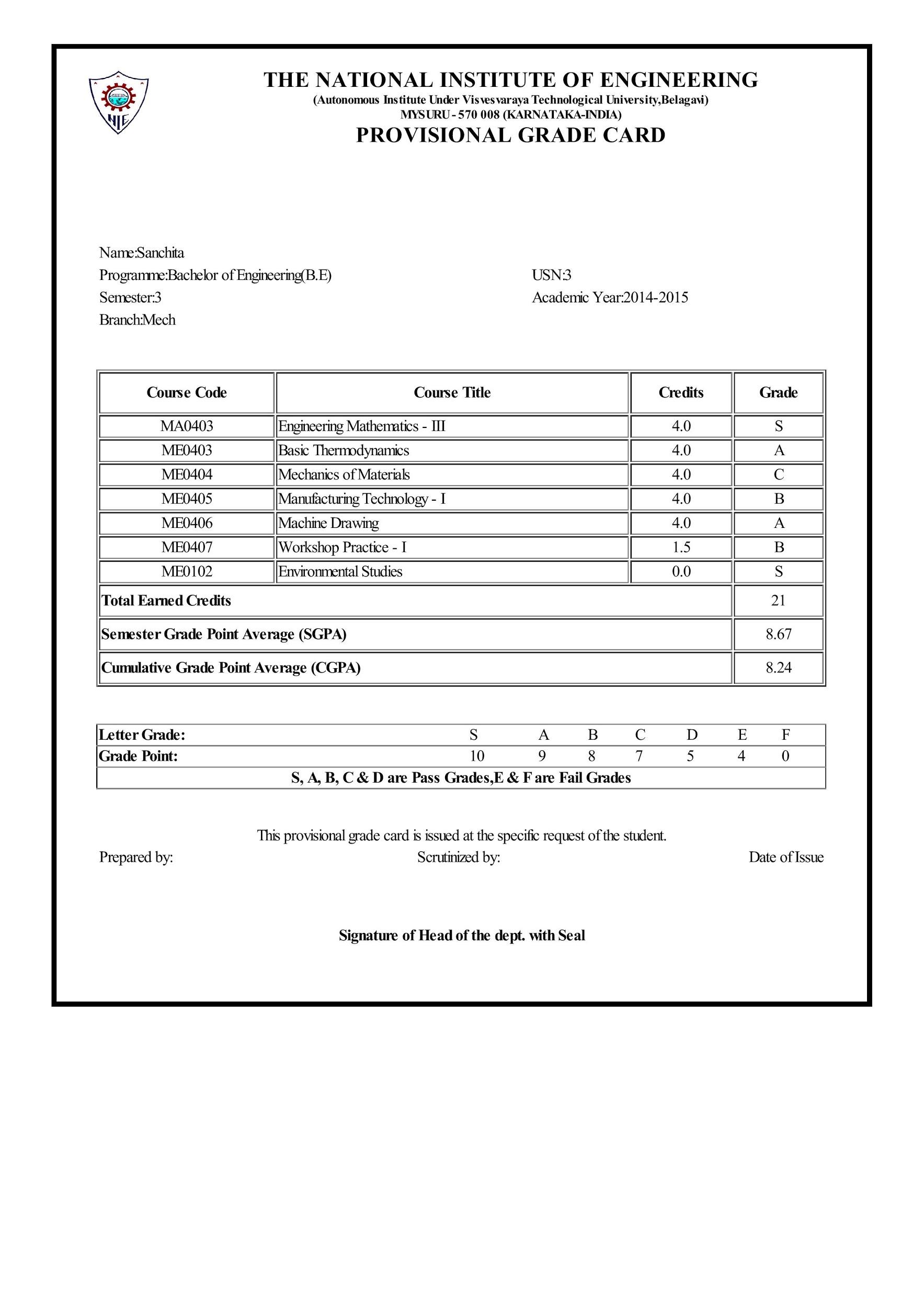


Figure 5.44

6.The provisional marks card or transcript is generated and mailed to COE’s office for further processing.

Figure 5.45

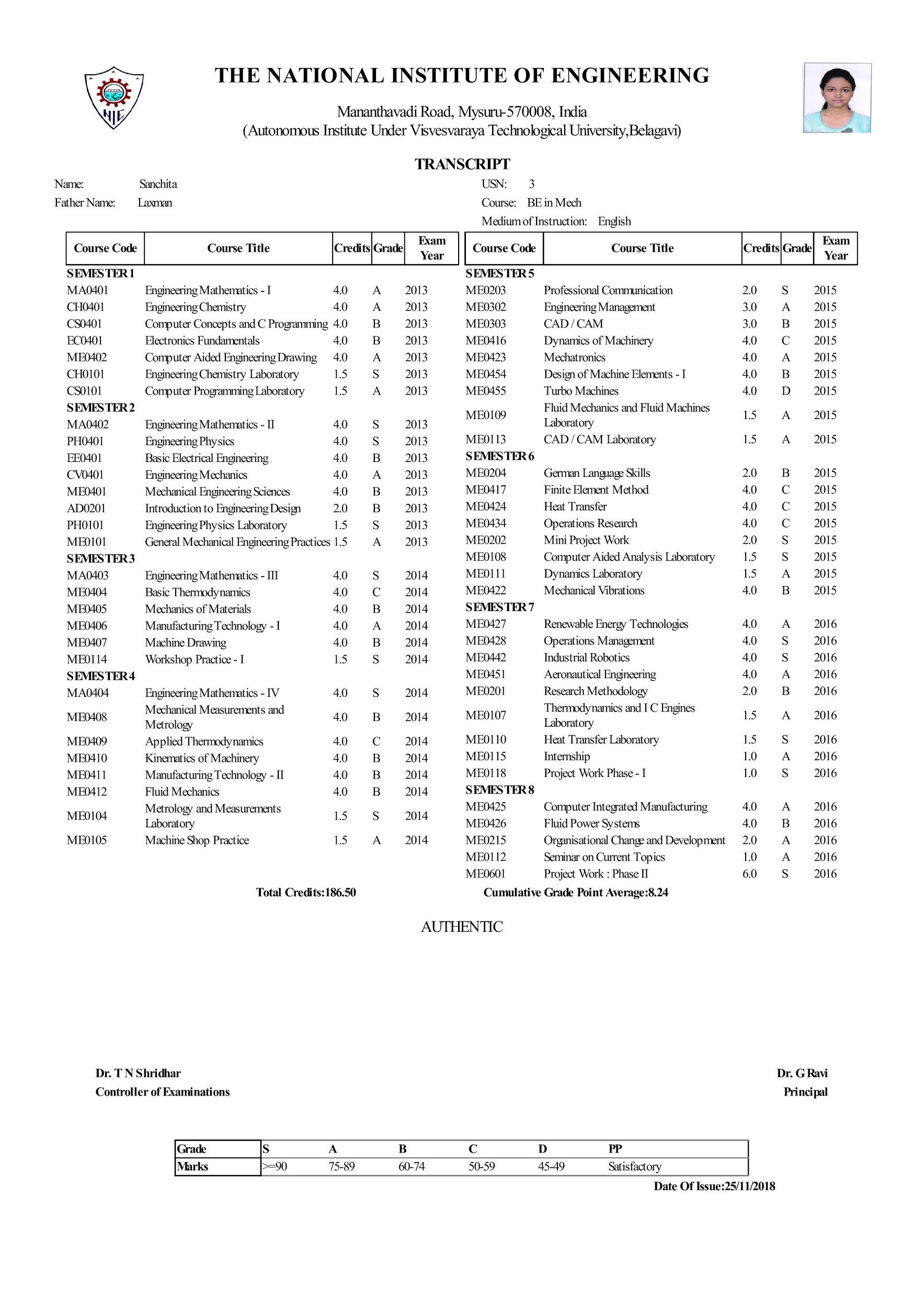


Figure 5.46

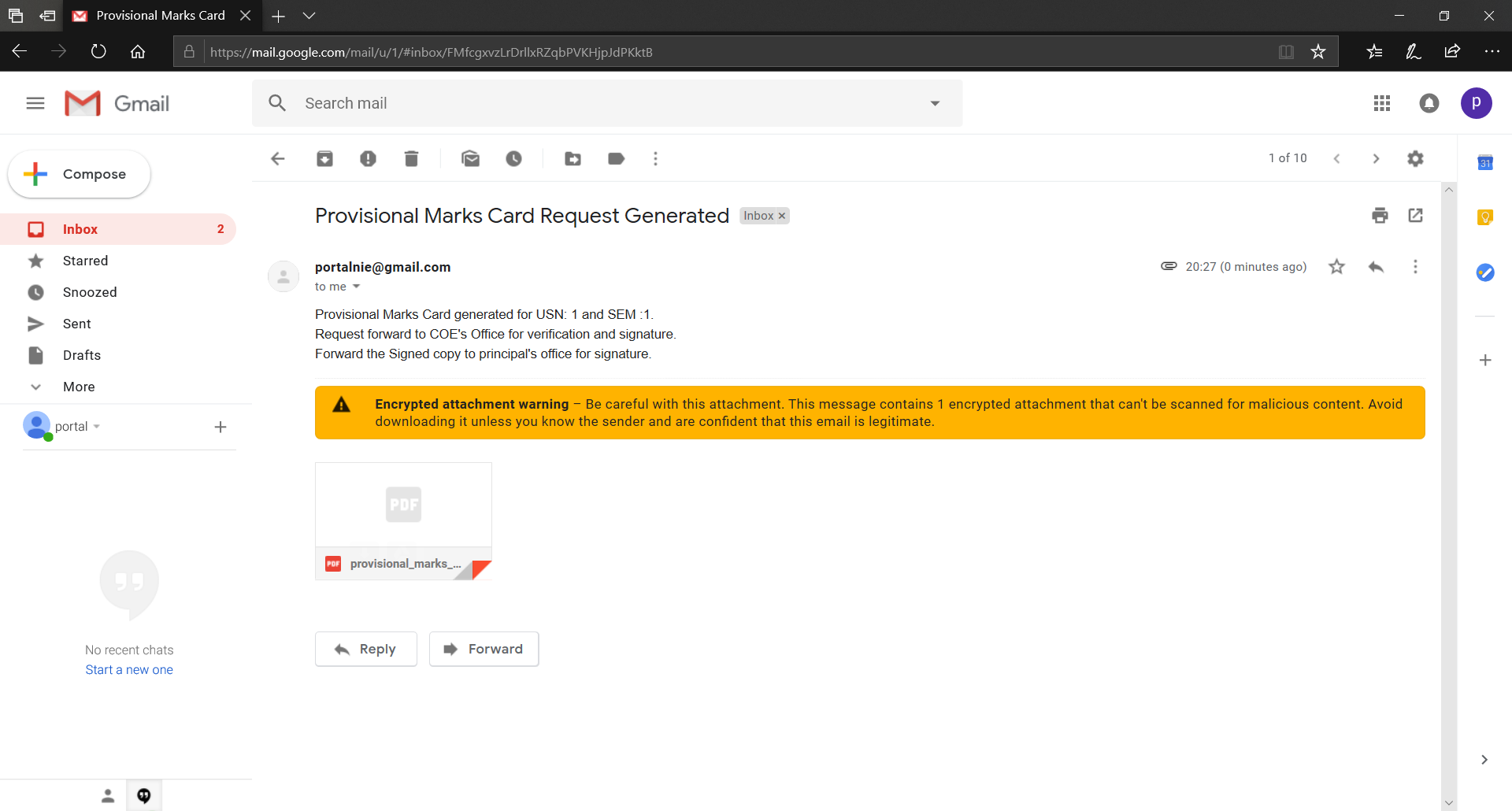


Figure 5.47

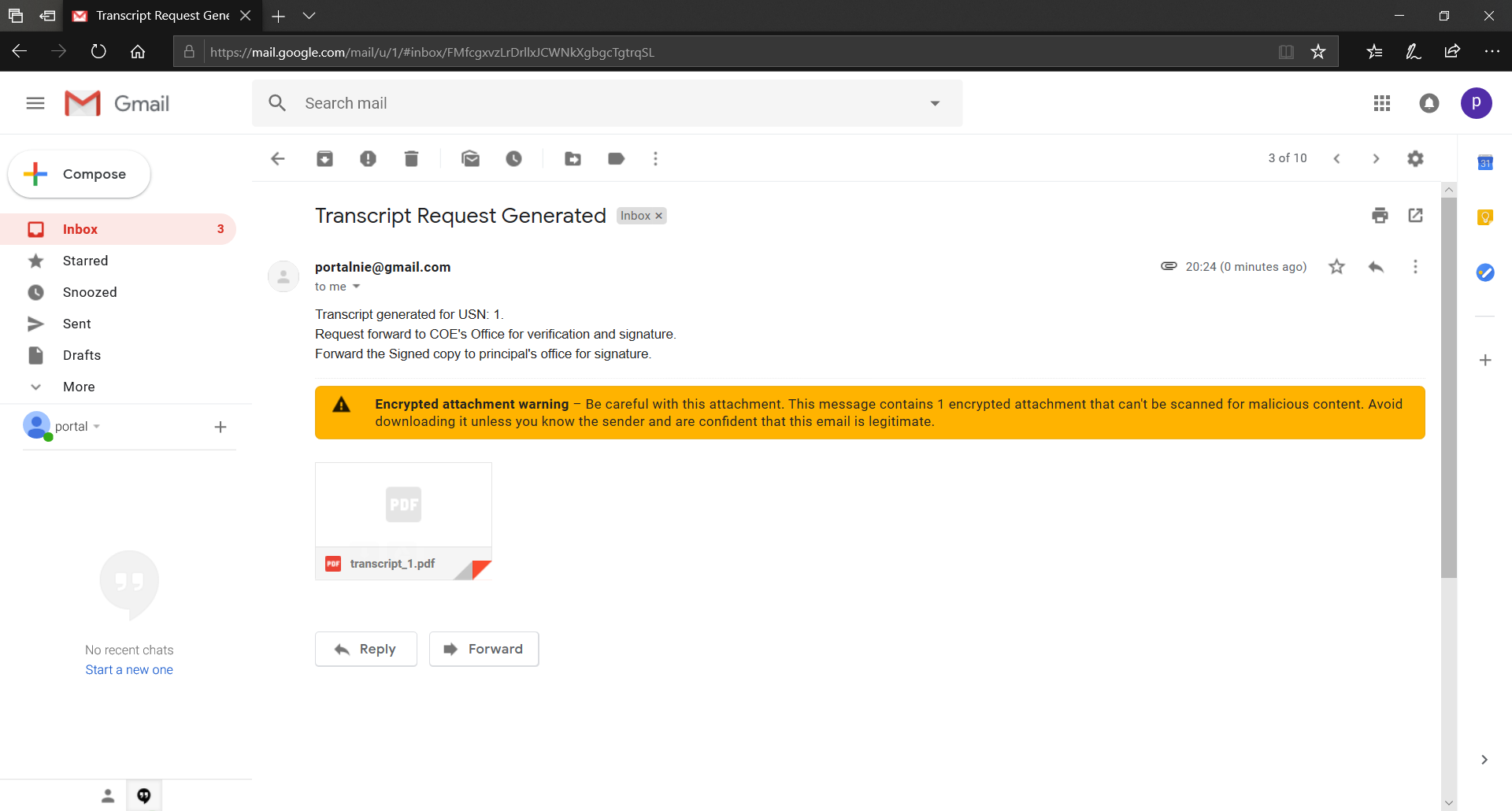


Figure 5.48

* **User Interface:** The user interface is given main focus in implementation of the application as the user interacts with the system through the User Interface.
* **Teachers Dashboard**

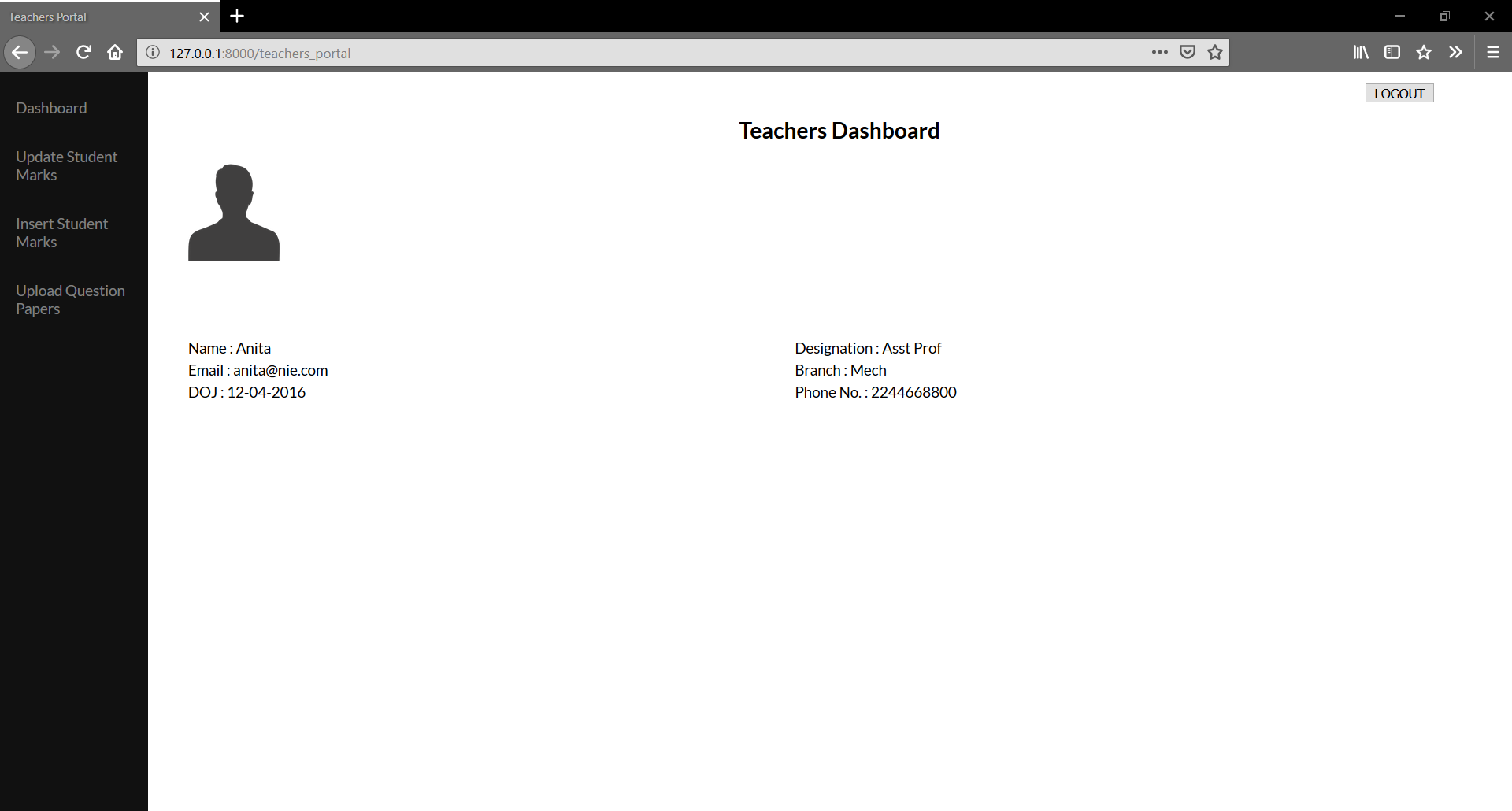
****

Figure 5.49

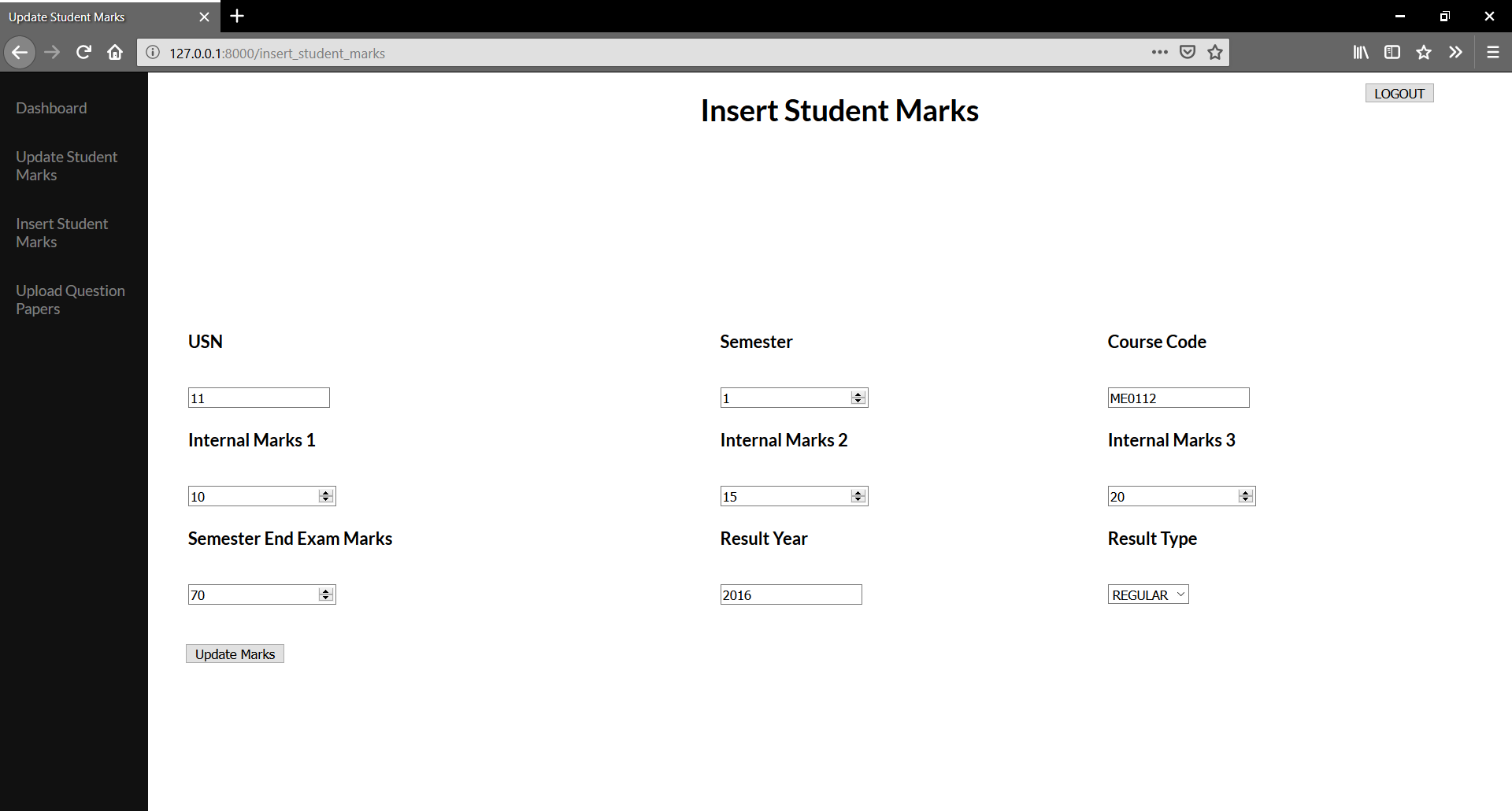
Teachers can login to insert and update student mark and upload question papers. 

Figure 5.50

Teacher can add marks of student specifying the usn, sem, course\_code and result type, the values for cie, see, total marks, grade is calculated and stored as a new block.

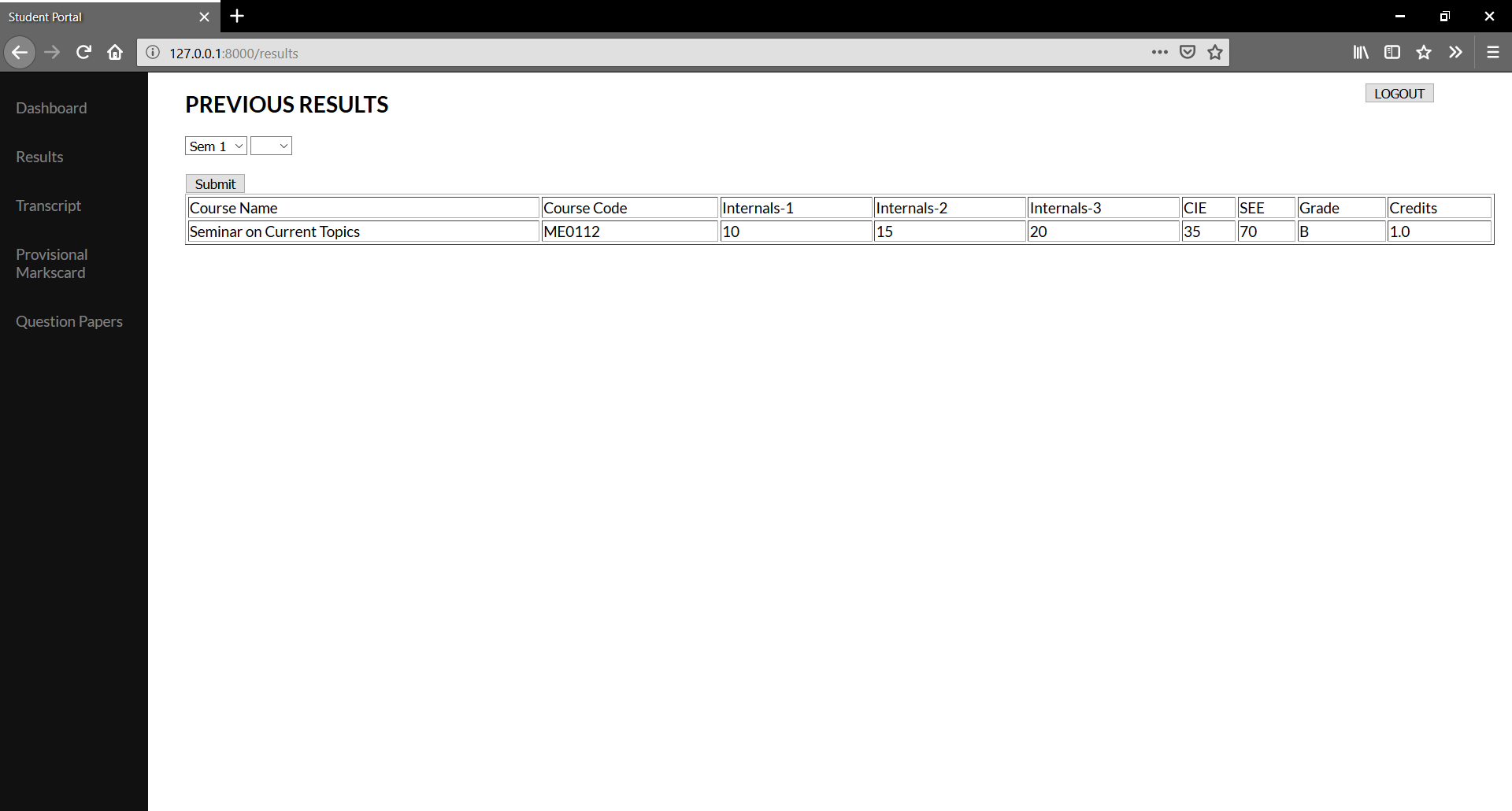
****

Figure 5.51

Student can now view the newly updated result in their portal.

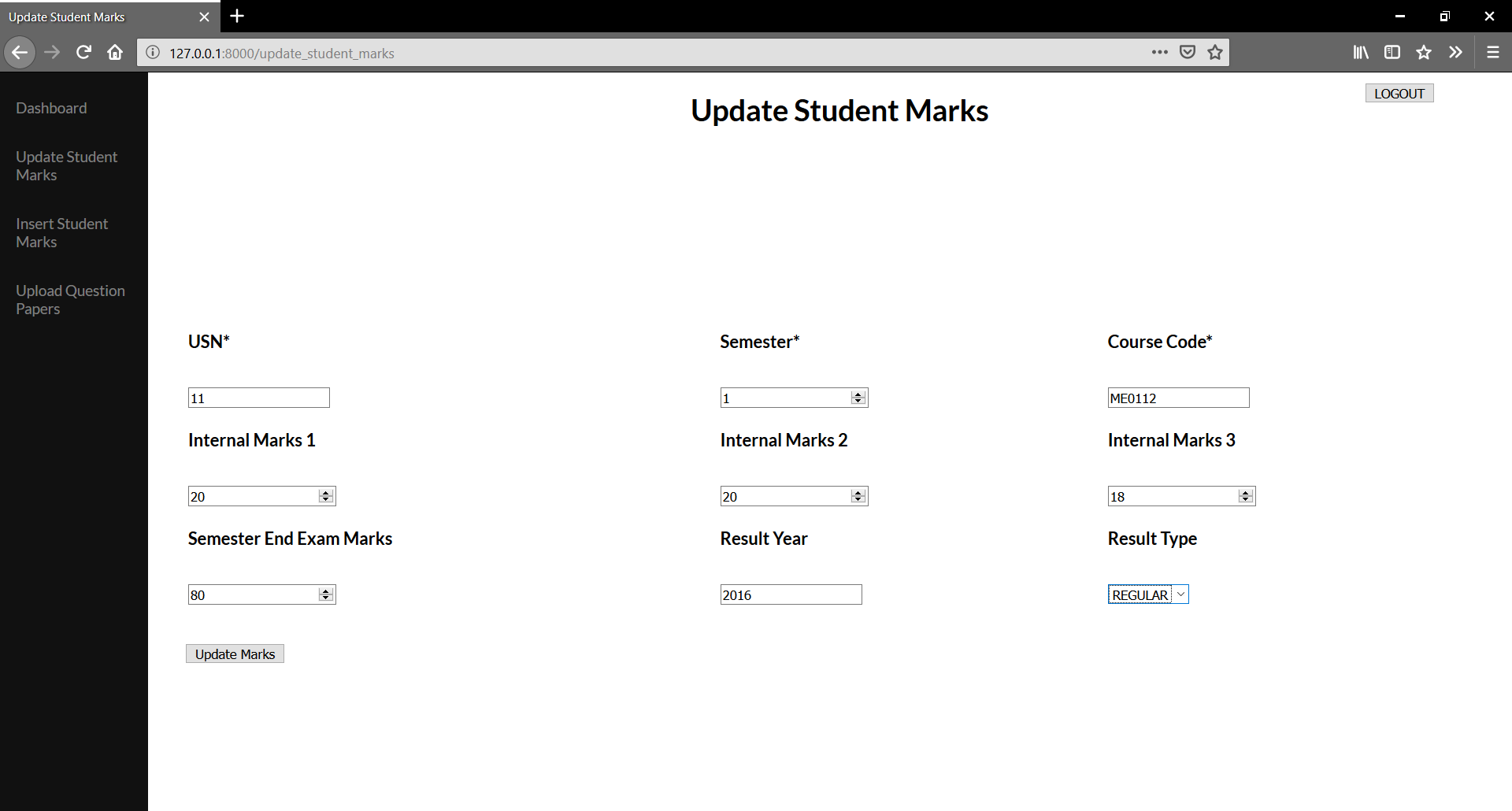


Figure 5.52

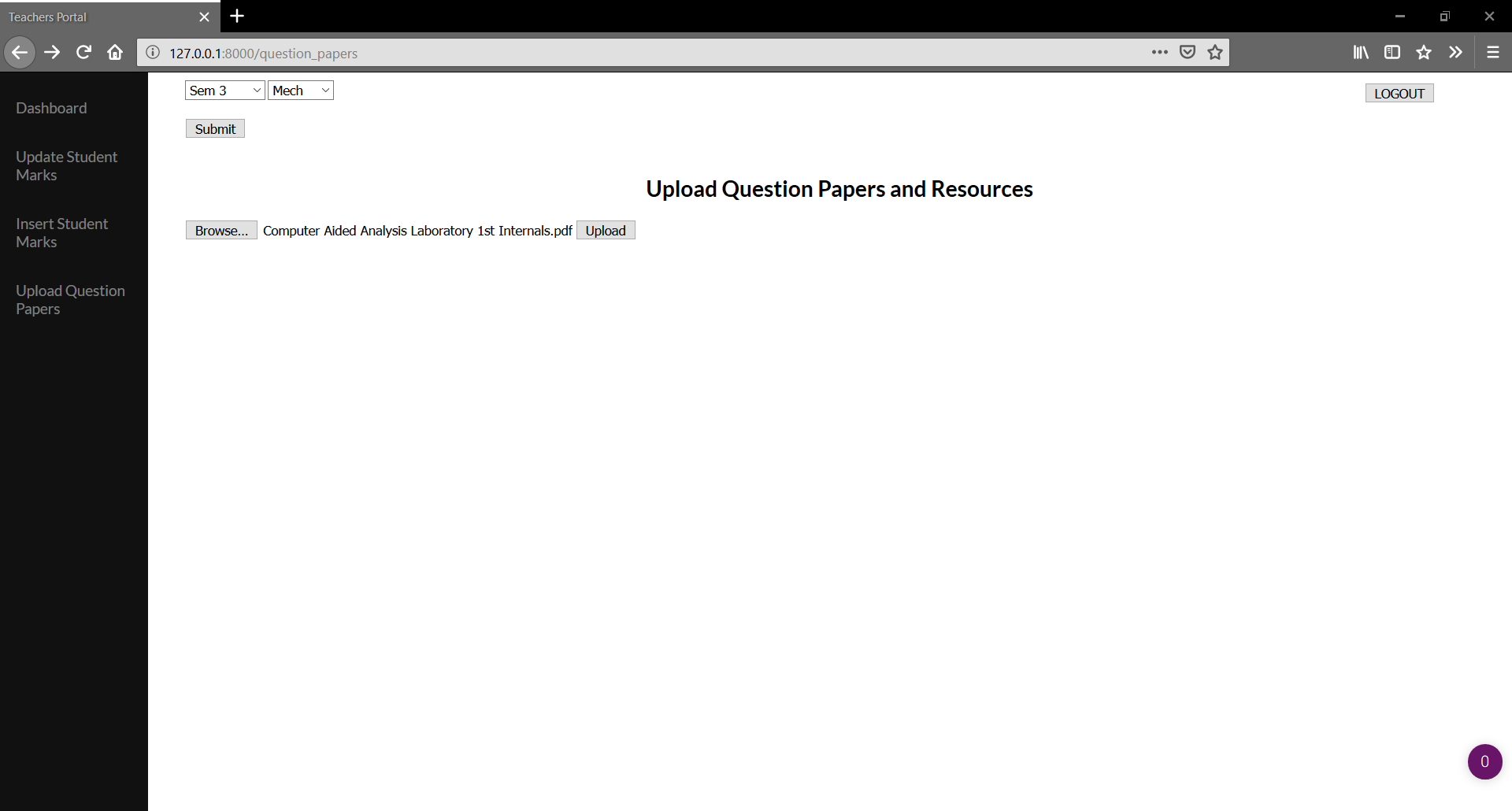
Teachers can update the marks, the updated marks is stored in a new block, student can view the updated marks.

Figure 5.53

Teachers can select the semester and branch, and upload previous question papers and resources which can be accessed by the students in the student portal.

* **Database Administrator Dashboard**

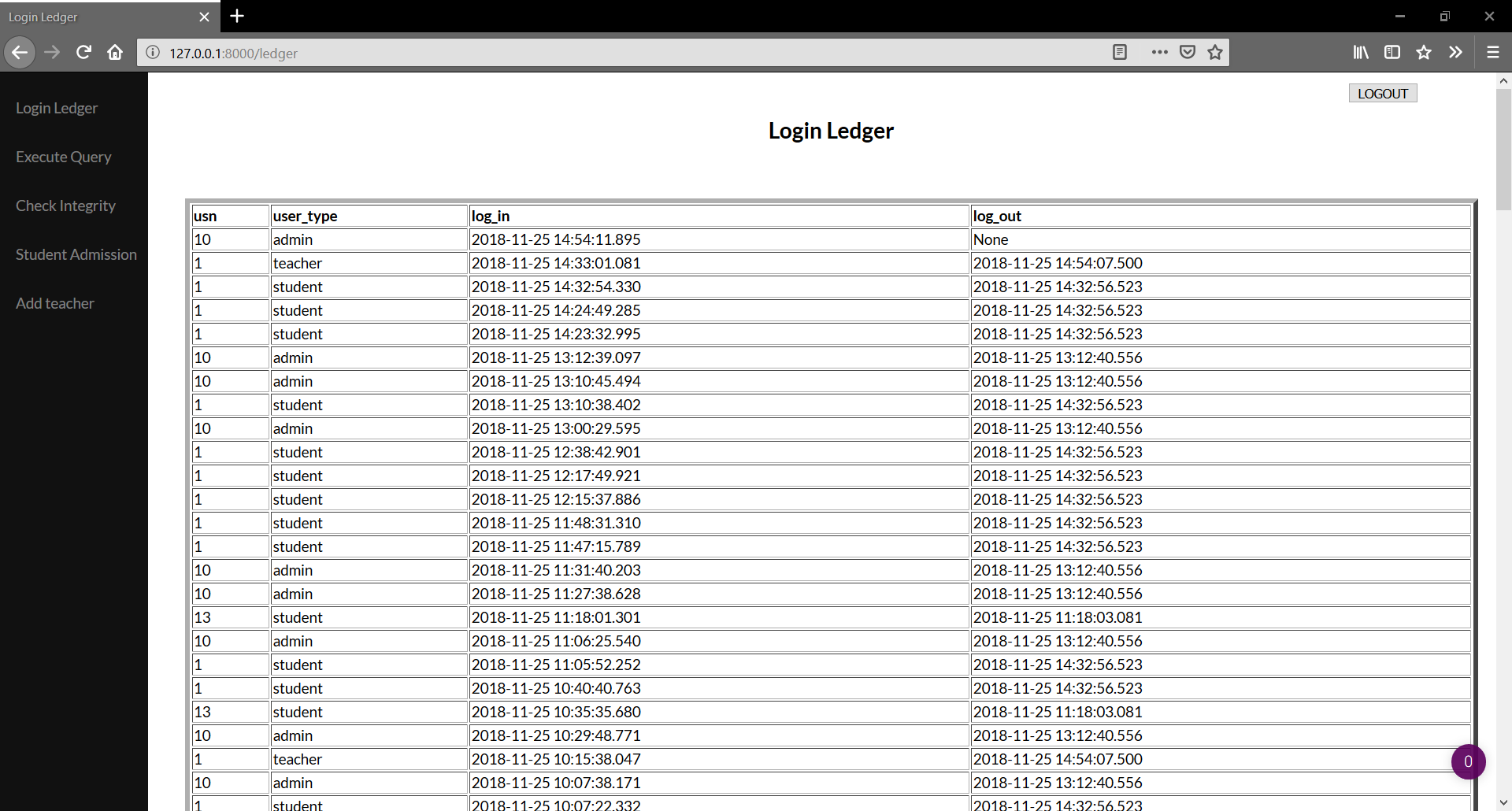
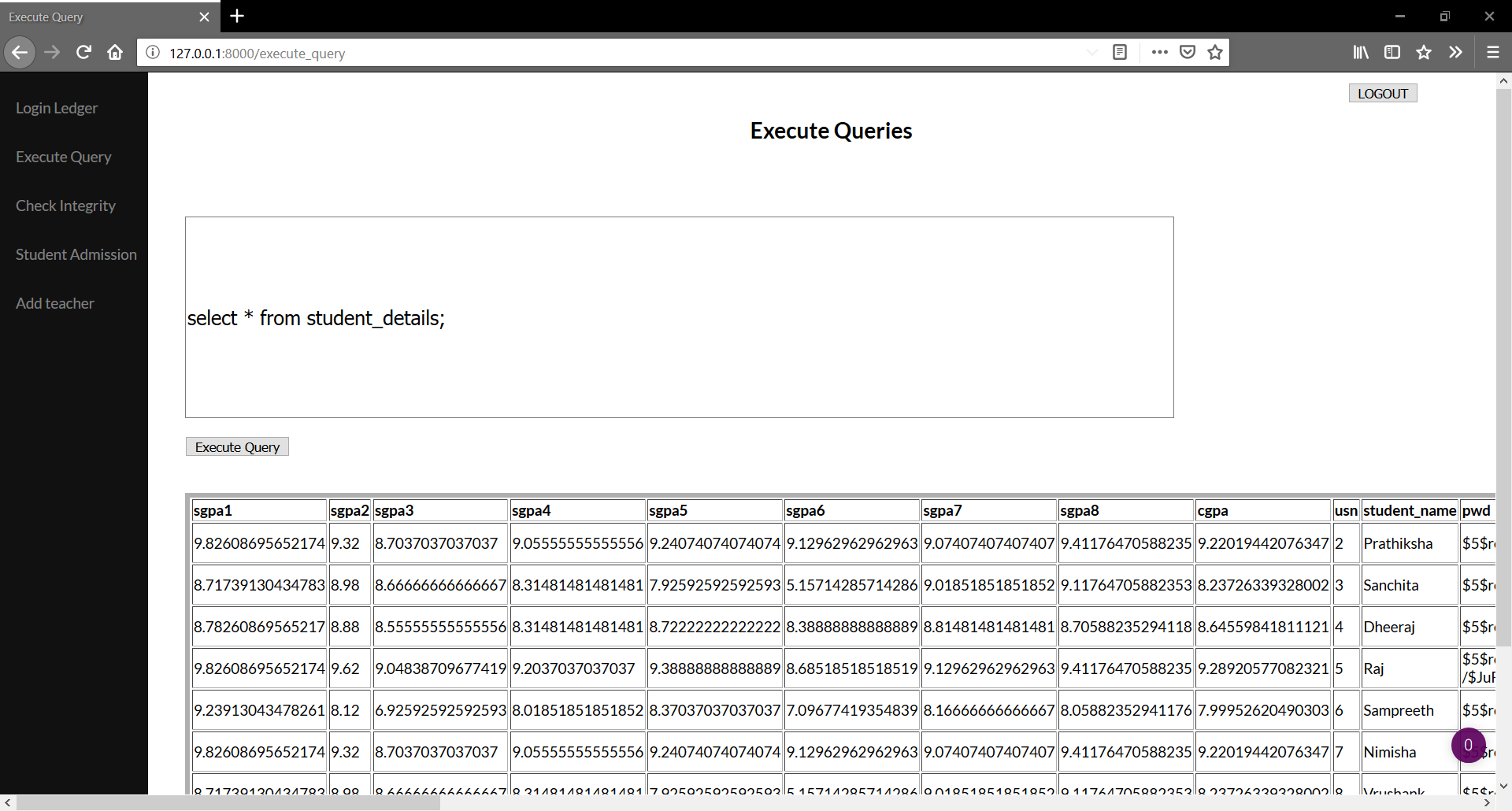
****

Figure 5.54

Database administrators can login to maintain the database, check integrity, execute queries and add details of new teachers and students during admission. All the records are maintained/edited by writing queries in SQLite3 through Flask.

Figure 5.55

Database administrators can execute queries to CRUD operations on the records.

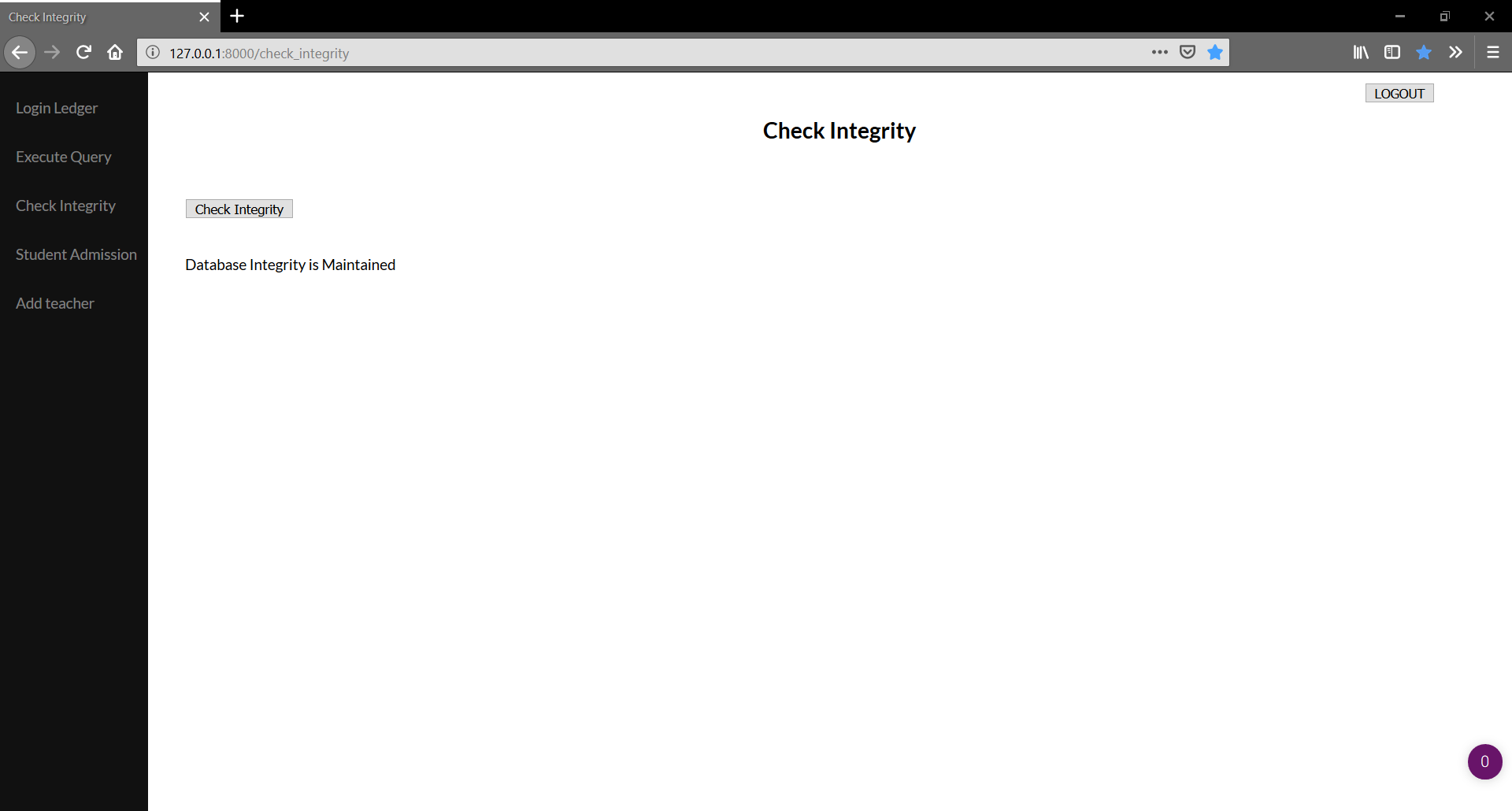
****

Figure 5.56

The database administrator is provided with a check integrity tool to check if the database integrity is maintained, since the process might take a while, if the integrity fails a mail is fired to the administrators mail address specifying the rows or blocks where the integrity failed.

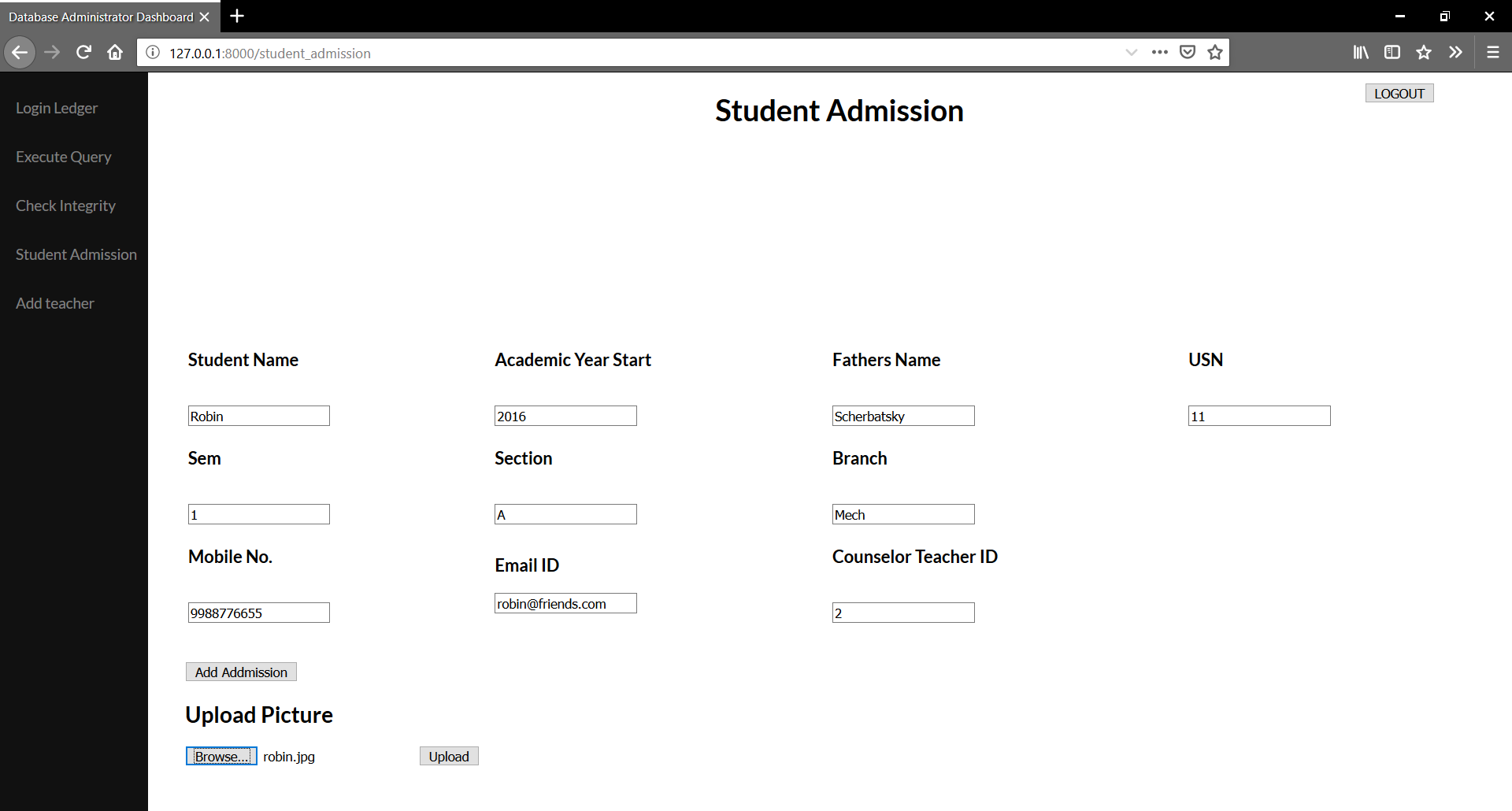


Figure 5.57

New students can be added with all the details and their picture.

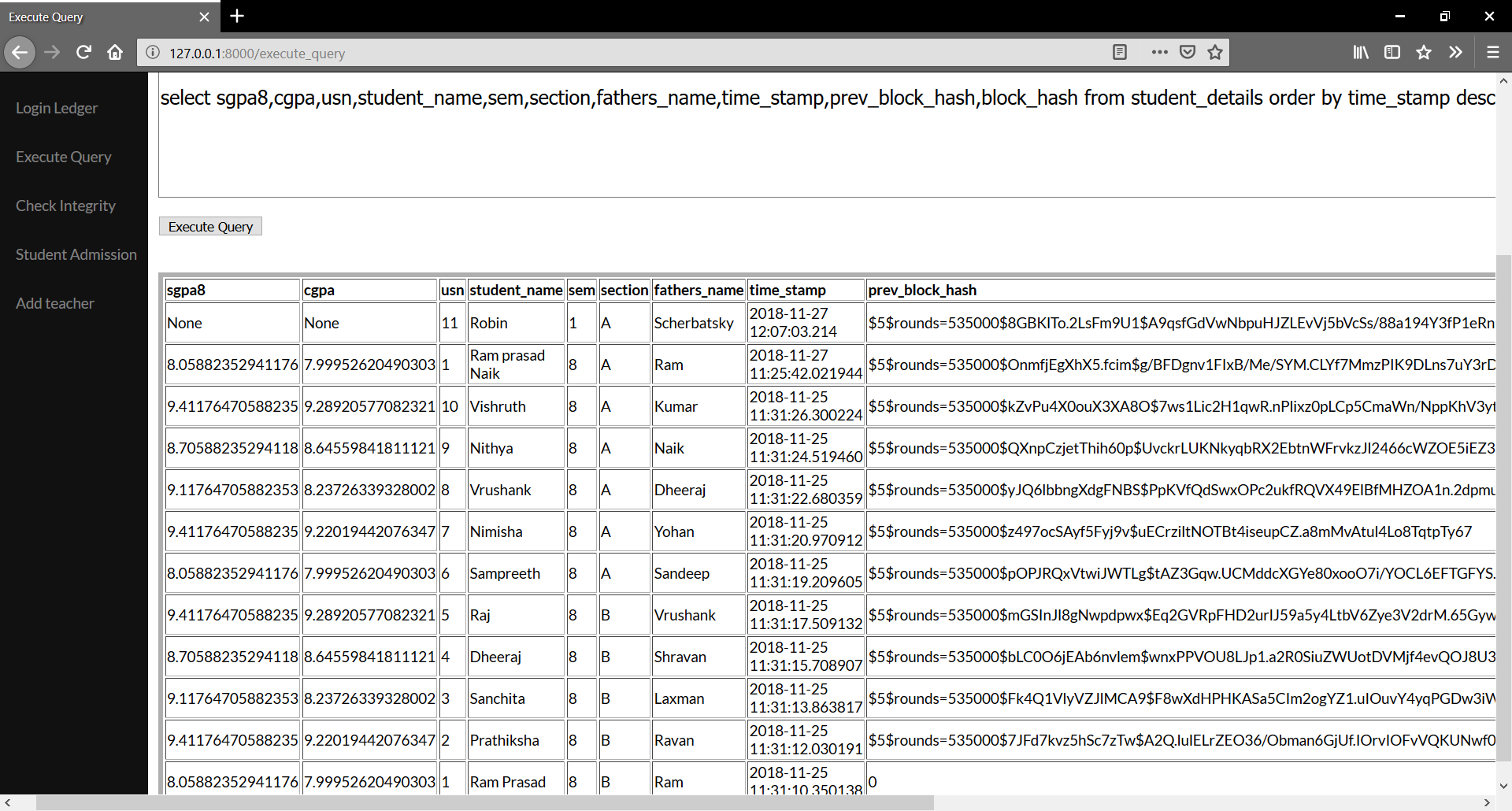
****

Figure 5.58

The new student added is added as a new block with the details as given in the admission form, rest of the values such as CGPA and SGPA remain Null.

****

Figure 5.59

The blockchain is maintained whenever a new student admission takes place, the current blocks’prev\_block\_hash is equal to the previous block’s block\_hash.

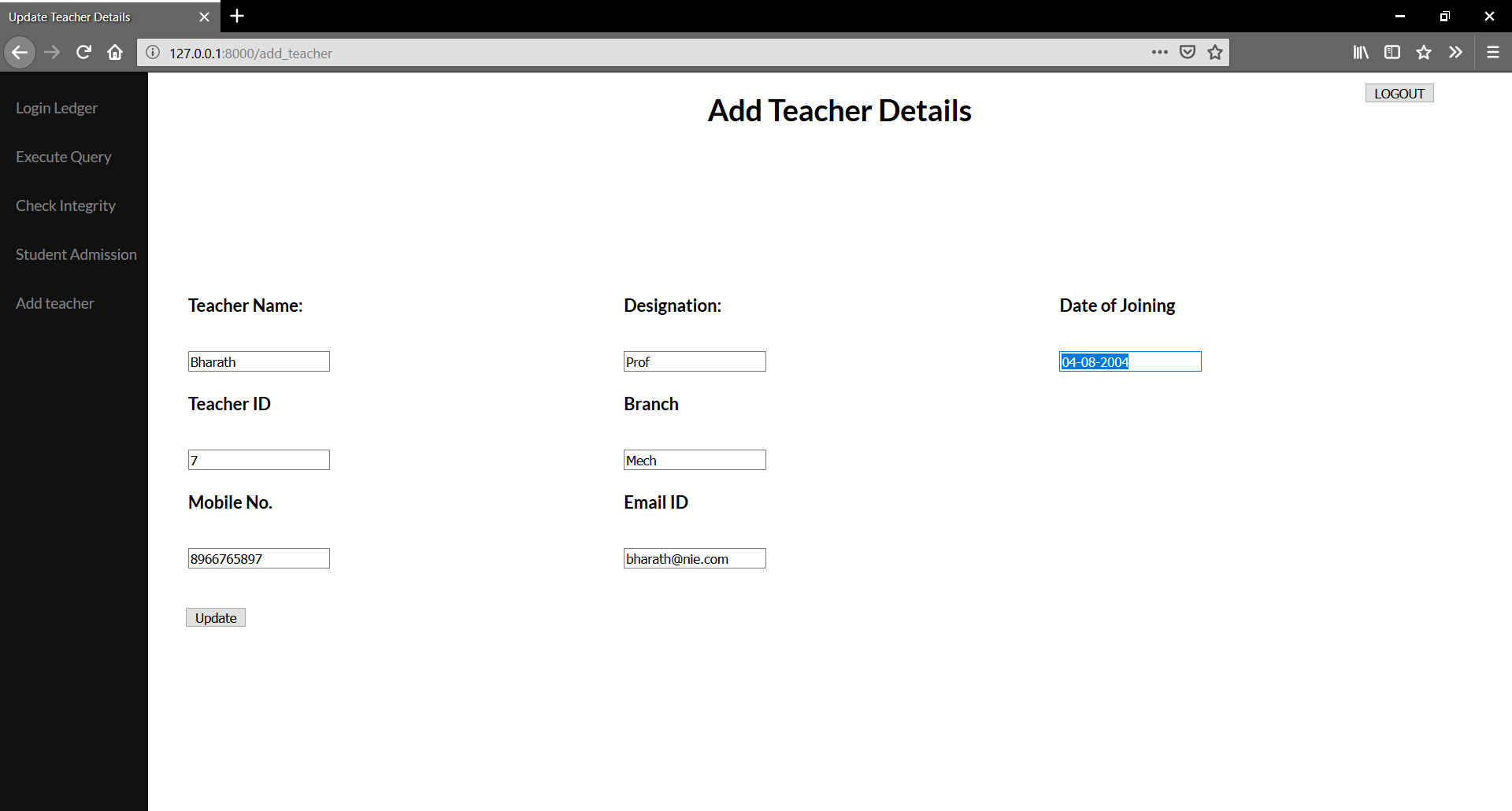


Figure 5.60

New teachers can be added along with their details.

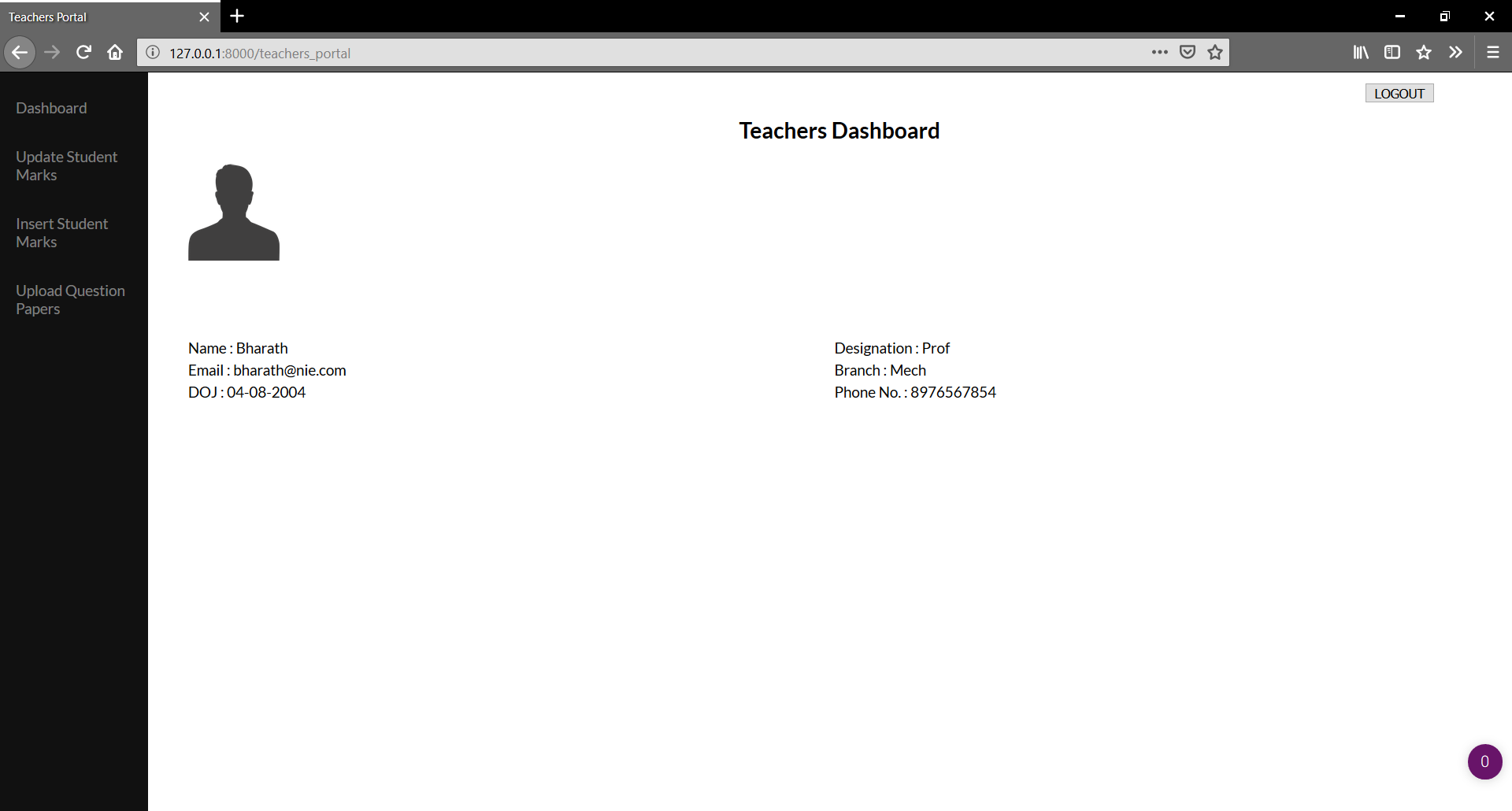


Figure 5.61

The teacher can now login and use the portal.

* **Student Dashboard:**

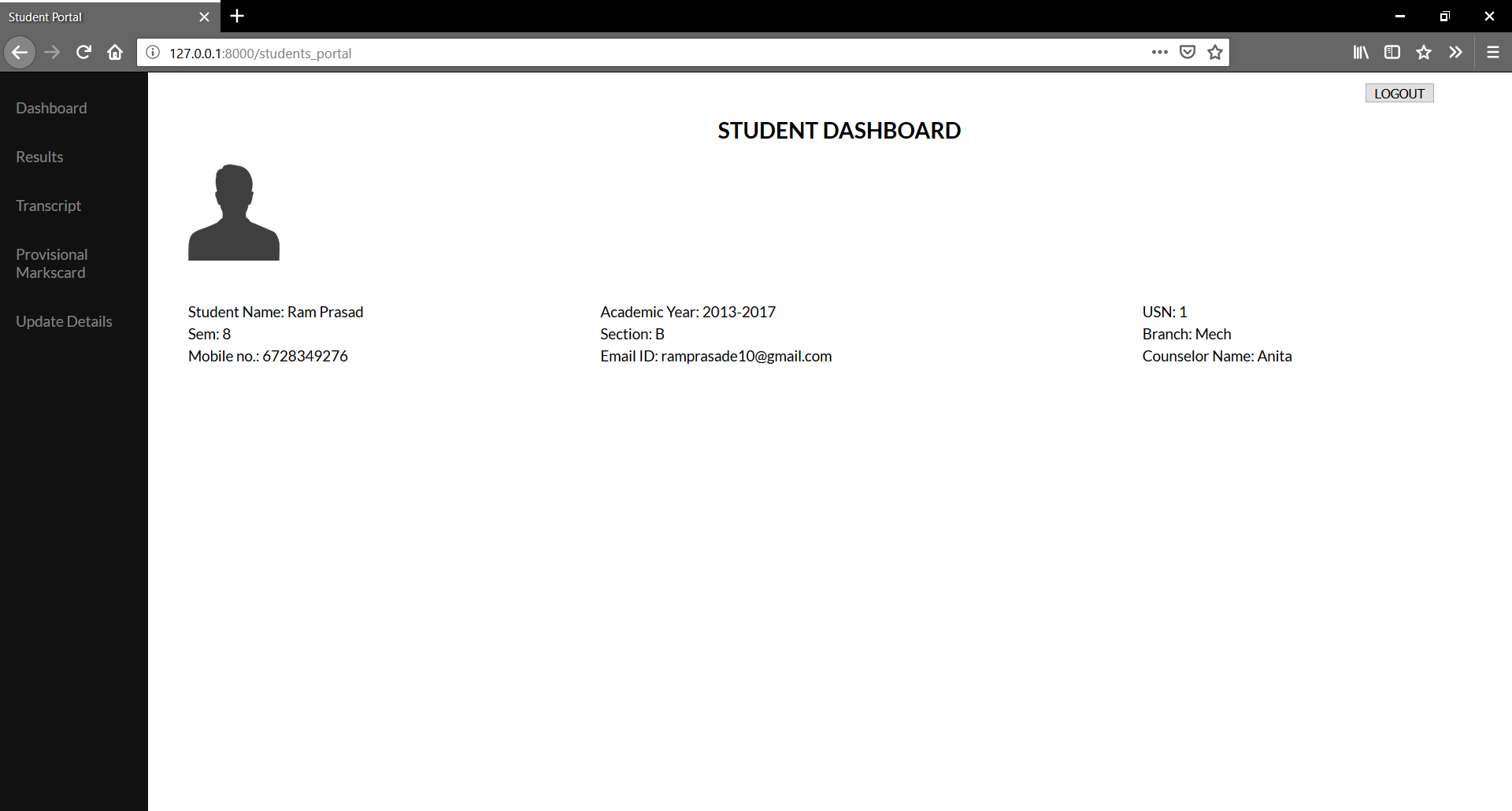
****

Figure 5.62

Students can login to view their current results and previous results, request for transcript and provisional marks card and also update their details.

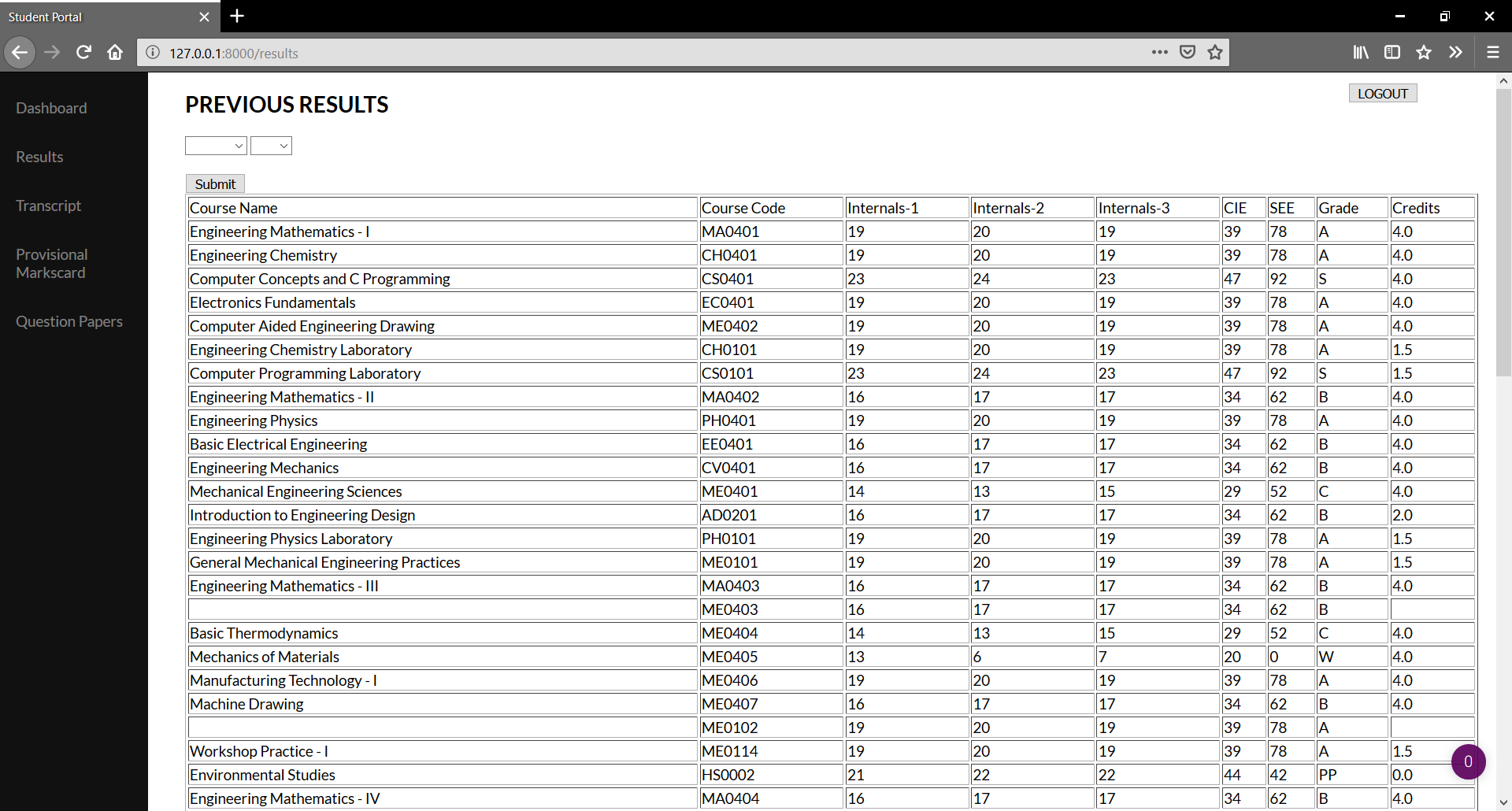
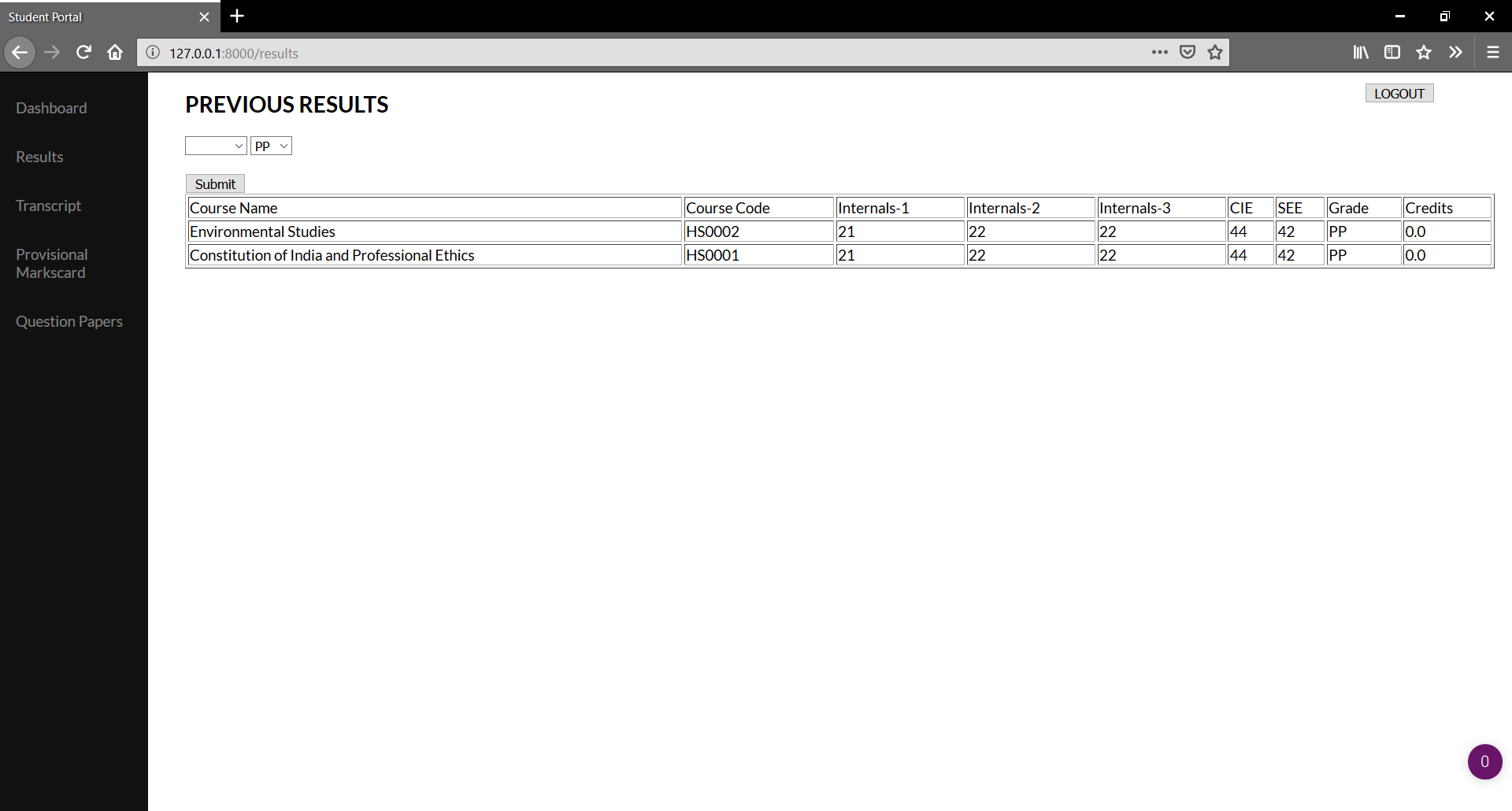
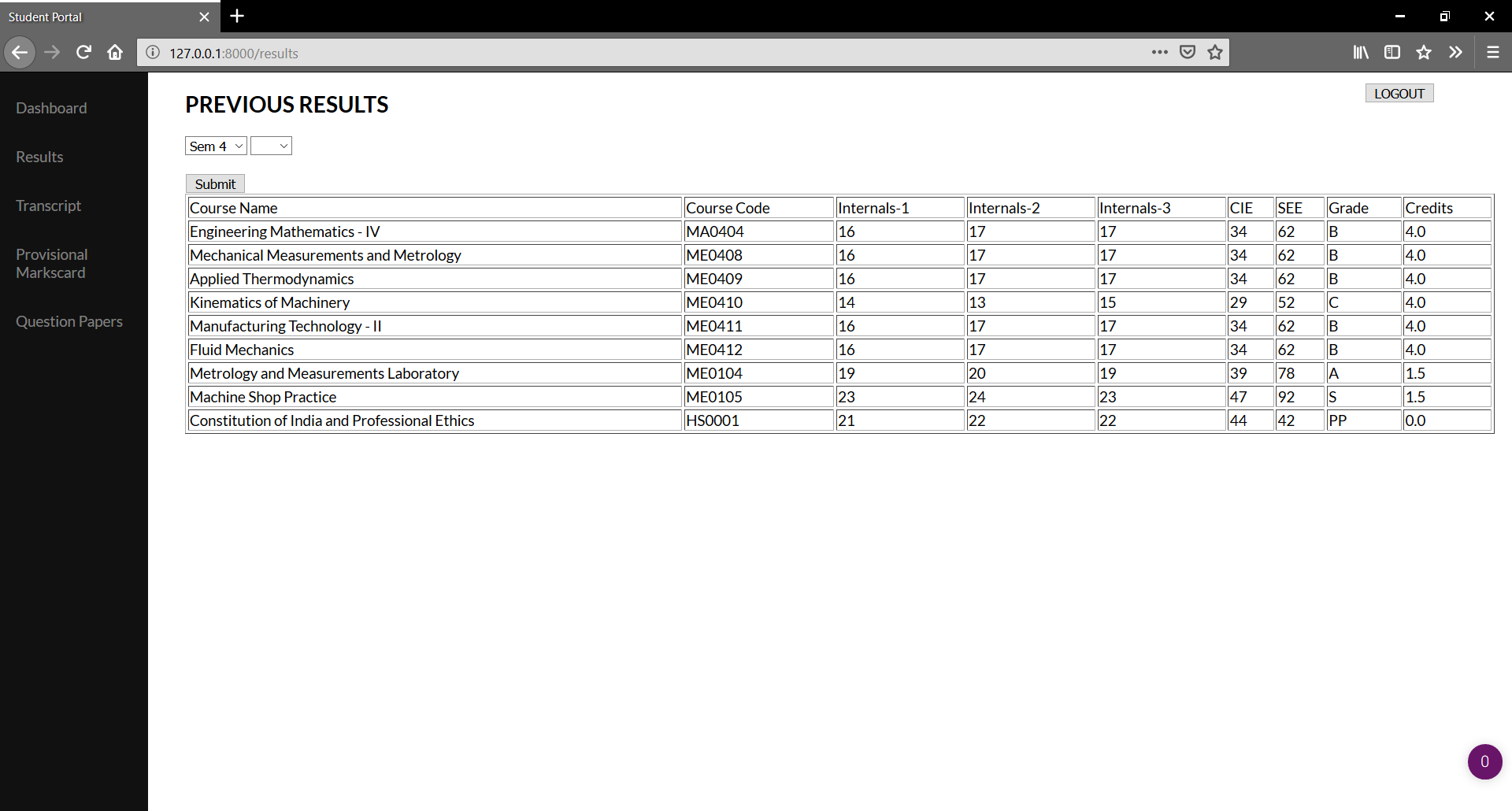


Figure 5.63

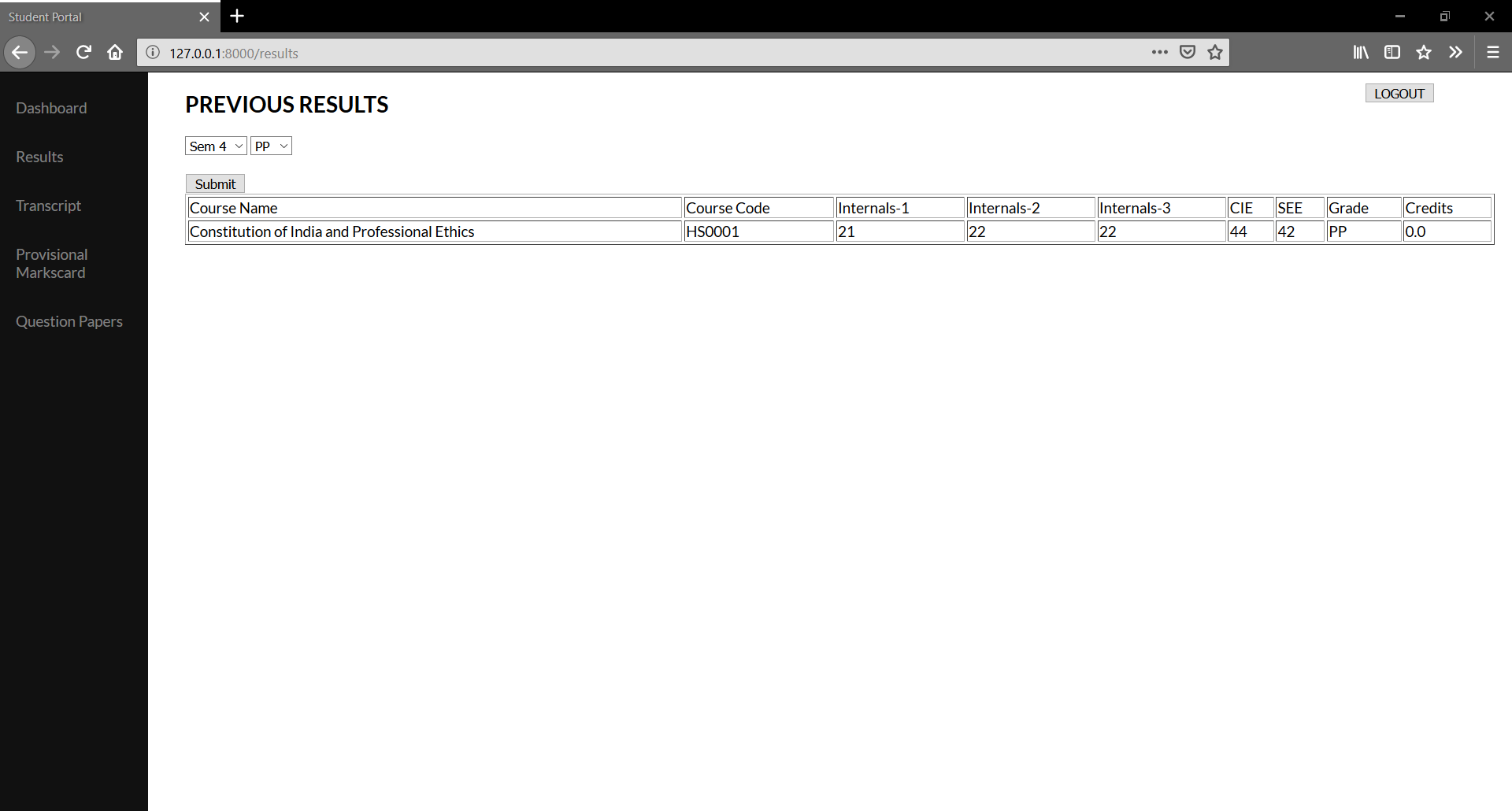
Students can view their previous results semester wise or based on grades. They can choose semester, grade, both semester and grade, or neither from the dropdown list. If neither is chosen, results of all previous semesters are displayed.

Figure 5.64

Here, the grade is selected as “PP” and hence, only the subjects where the student has scored “PP” grade is shown irrespective of the semester.

Figure 5.65

Since semester 4 is chosen, only results for that semester is displayed irrespective of the grades.

Figure 5.66

Results corresponding to the grade and semester selected are displayed.

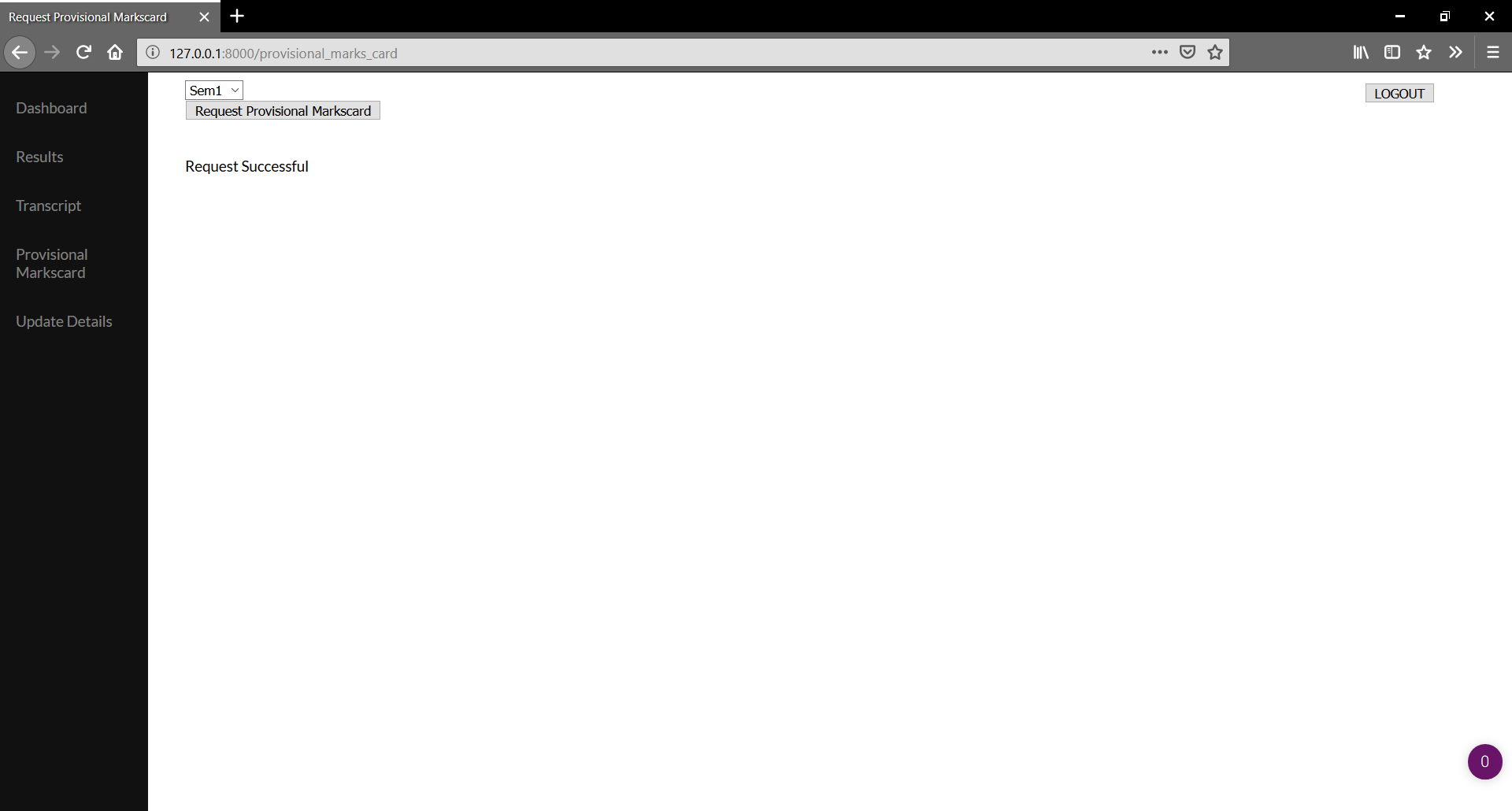


Figure 5.67

Provisional marks card can be requested from the student portal itself. Students can select the semester (from the dropdown list) for which the provisional marks card is required.

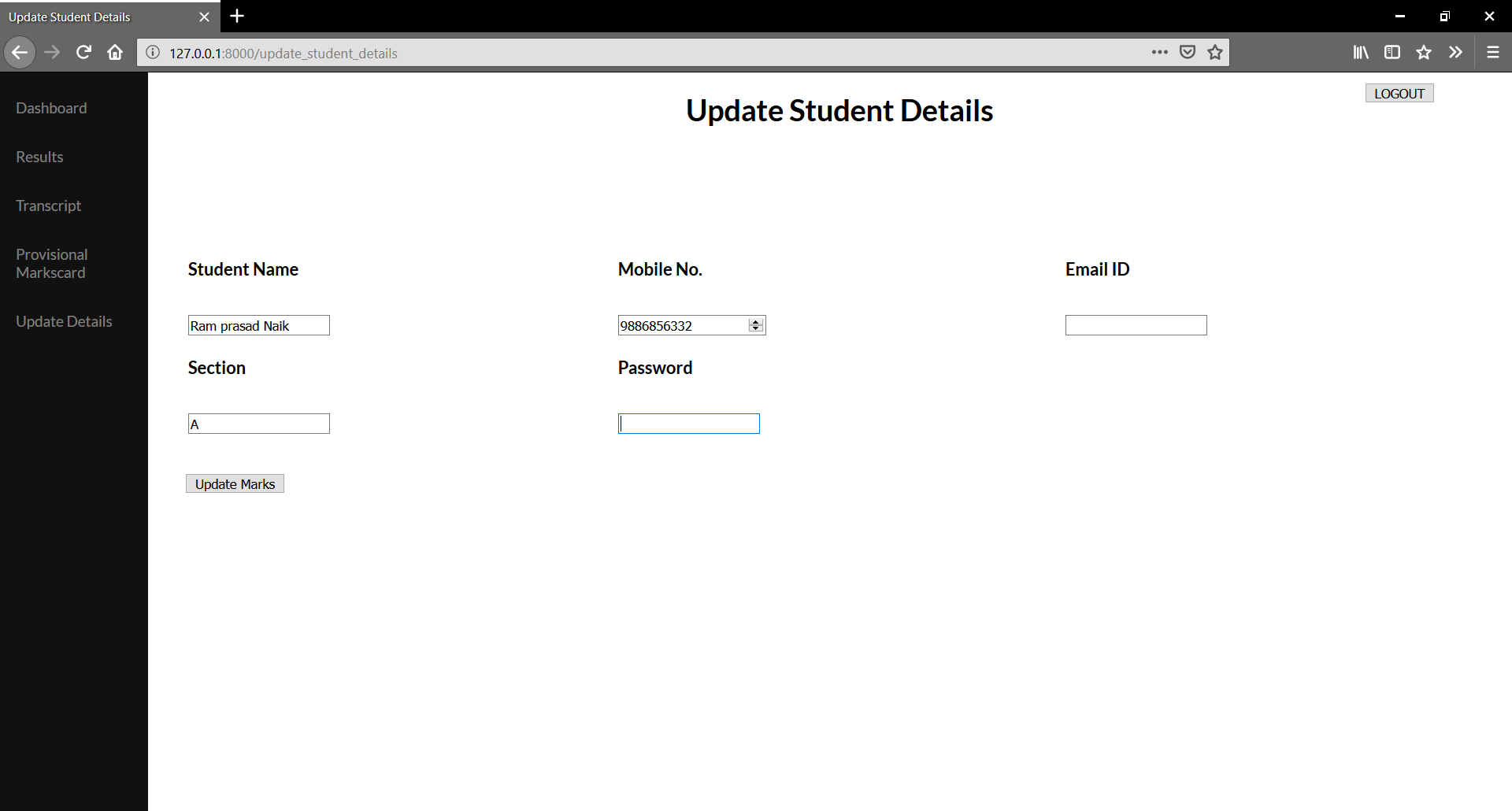
****

Figure 5.68

Student can update their details from the Update details page.

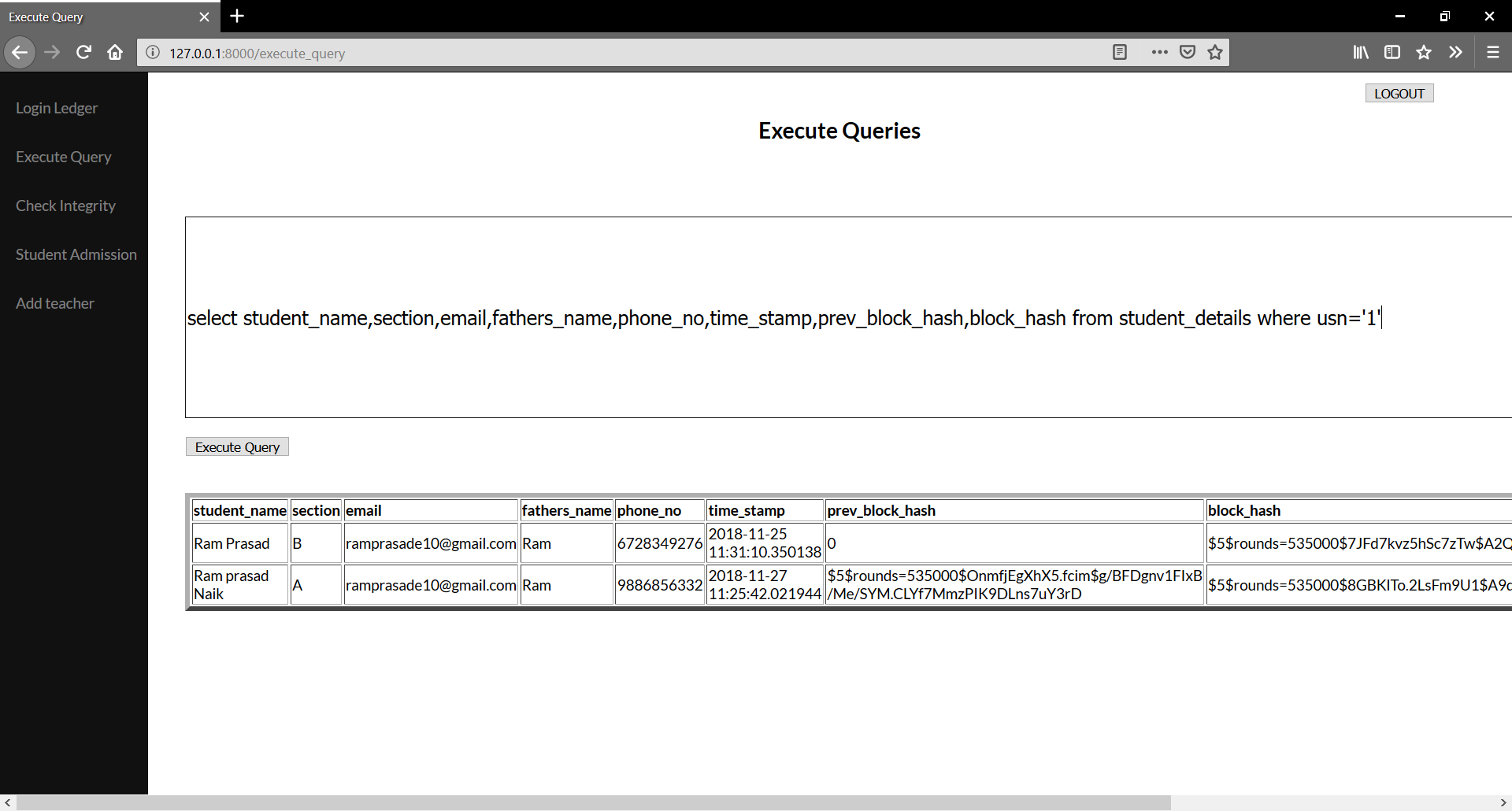
****

Figure 5.69

The updated details for a particular student are not updated on the same row corresponding to the student’s usn rather a new row is inserted with the updated values for the columns which are to be updated while rest of columns take the same values.This new block time\_stamp, prev\_block\_hash and block\_hash are queried,calculated and stored.

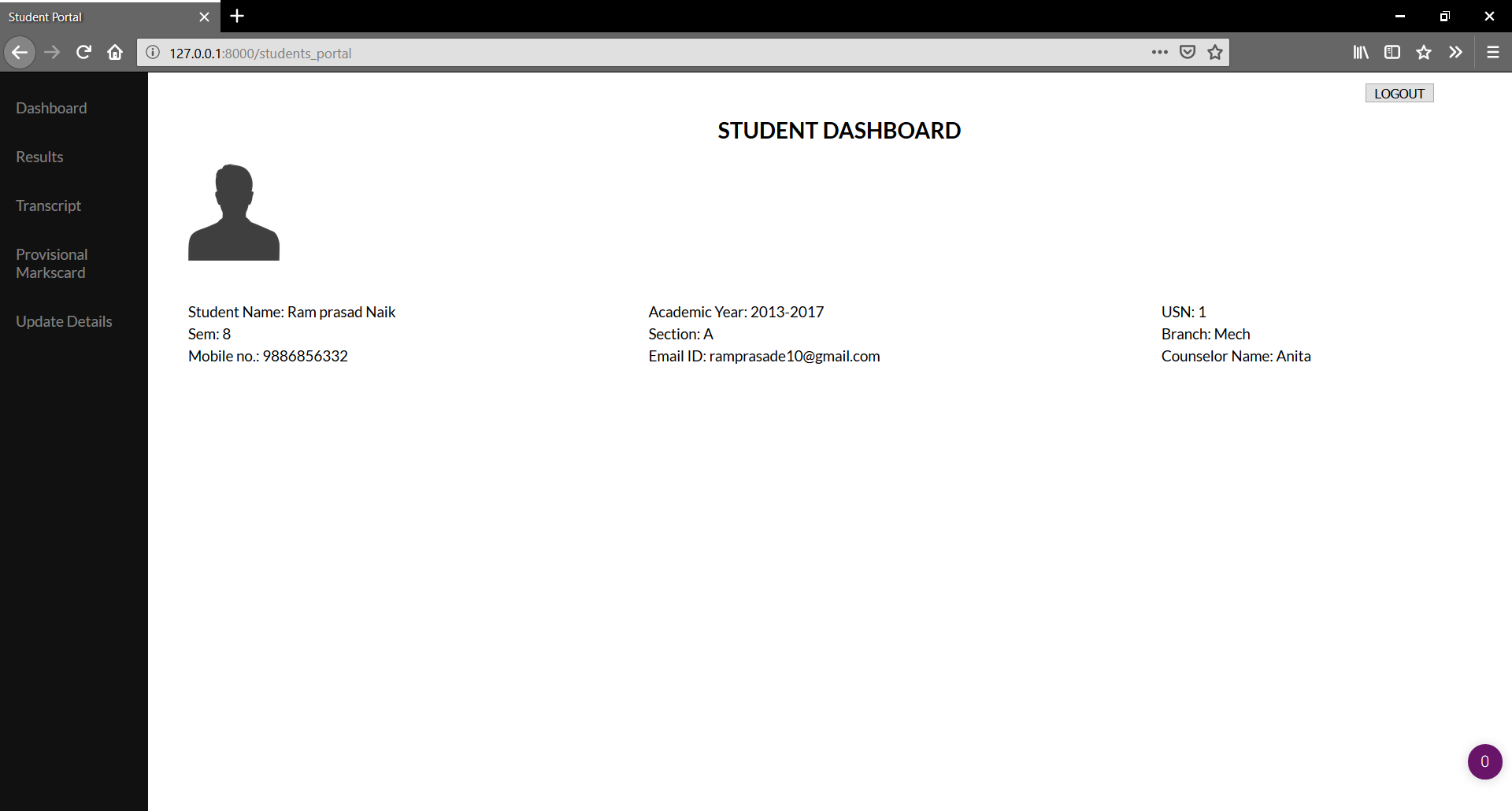
****

Figure 5.70

The updated student details are reflected onto the student dashboard.

**Student Performance Report:**

The provided data can be analysed to generate meaningful insights on student subject-wise and overall semester performance by utilizing techniques from data science to generate

graphs and charts.

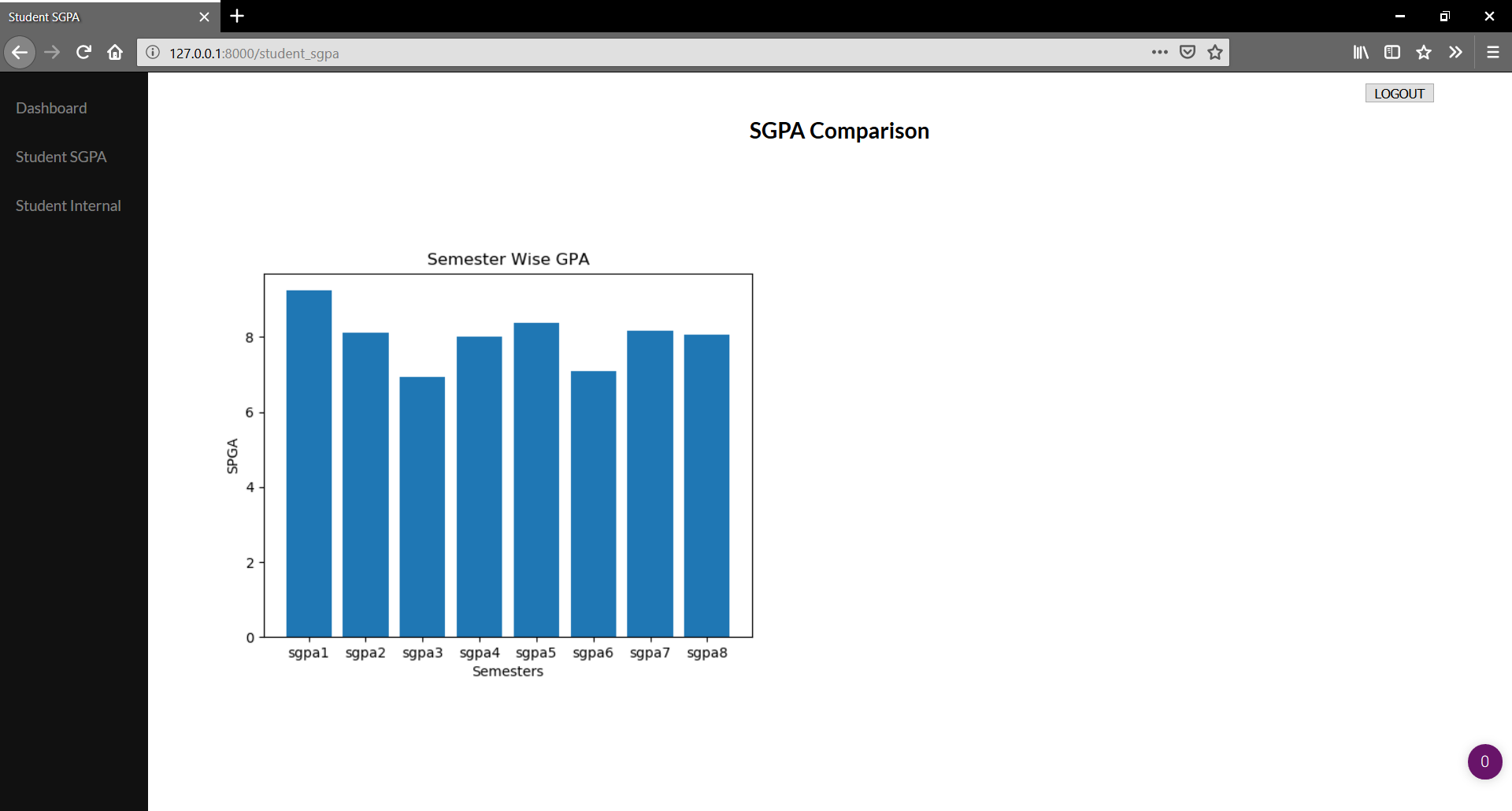
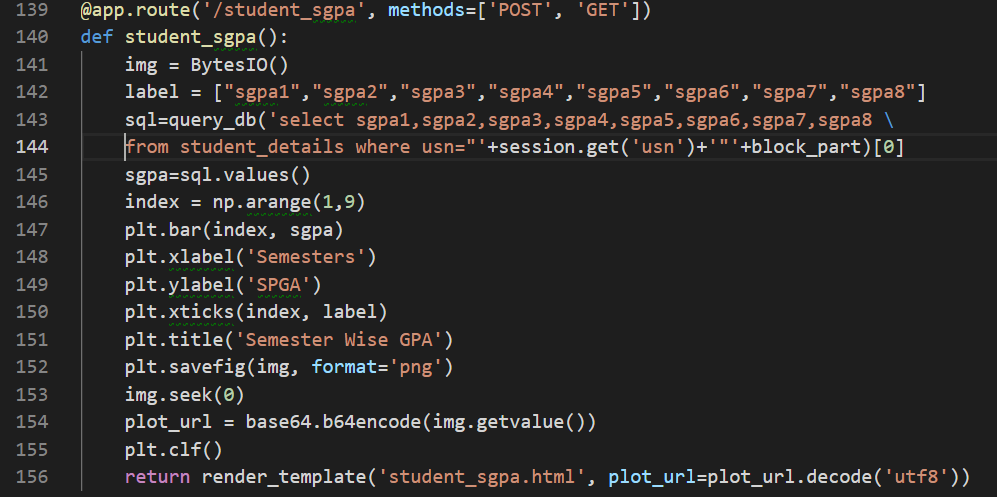
 Figure 5.71

Figure 5.72

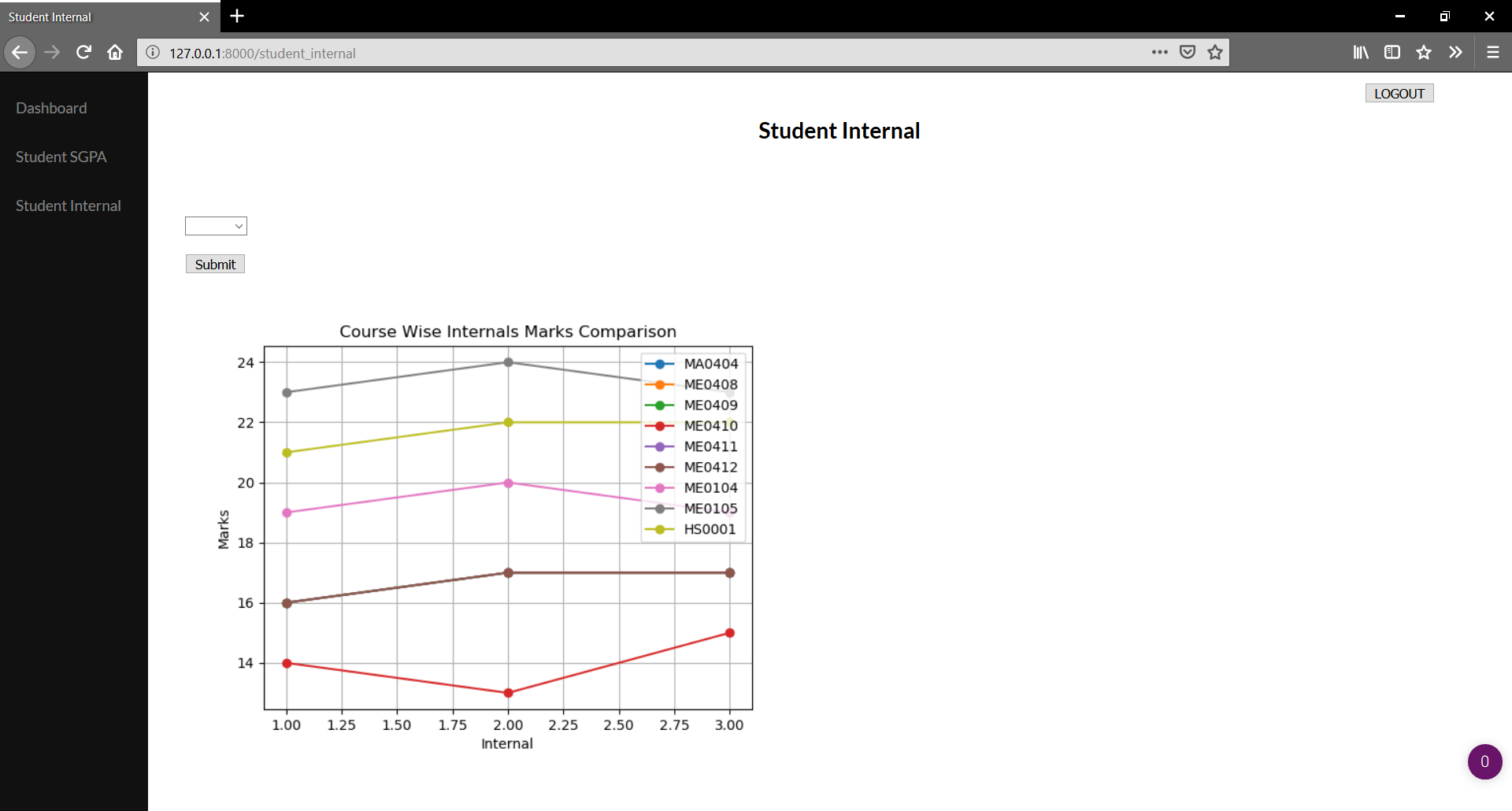
**SGPA Comparison:**A bar graph comparison of student's SGPA is generated by querying and plotting sgps1 to sgpa8 from student\_details table using matplotlib.

Figure 5.73

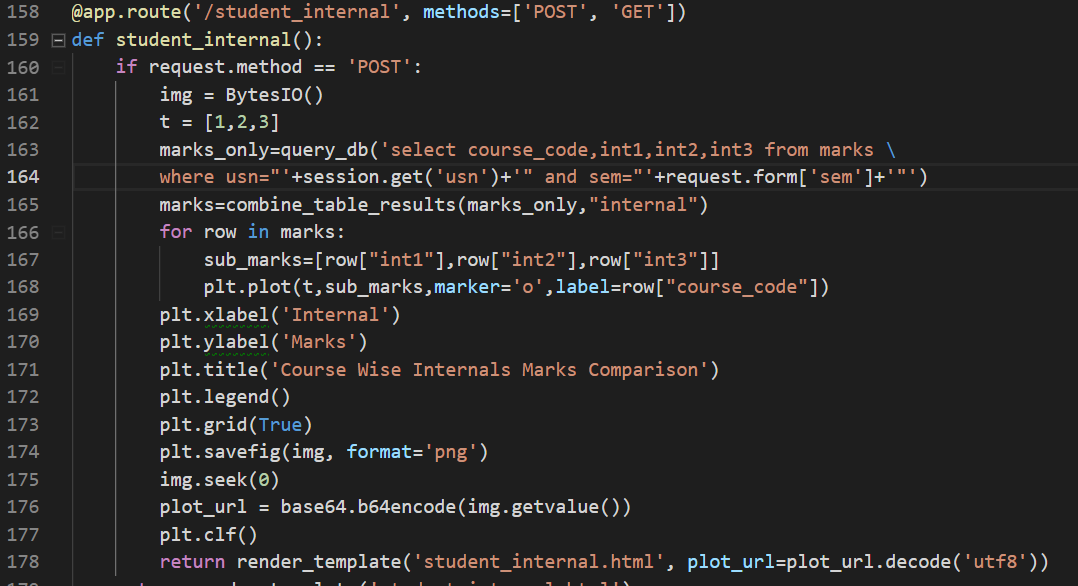


Figure 5.74

**Student Internal:** A line chart representing the marks of internals 1 to 3 for all courses in a Semesters selected by the student is generated by querying and plotting using matplotlib.

The legends represent the courses and their corresponding line color.

**Chapter 7**

**TESTING**

Testing is the set of activities that can be planned in advance and conducted systematically. Software testing is a critical element of software quality assurance and represents the ultimate review of specification,

design and coding.

Preparation of the test data plays a vital role in the system testing. After preparation of the test

data, the system under study was tested with the test data. Errors were found and corrected by using the following testing steps and corrections were recorded for future references. Thus, a series of testing is performed on the system before it is ready for implementation.

There are many strategies that can be used to test conventional software. Testing can be done once the entire software is complete. However, this results in buggy software and is simply not effective.

Alternate approach would be to test the software on a daily basis. Whenever a new part of software is developed, it is sent for testing first. This approach is effective as bugs are eliminated as the software is constructed and leads to more efficient software once all the modules are integrated.

Quality assurance is the review of software products and related documentation for completeness, correctness, reliability and maintainability. And of course it includes assurances that the system meets the specification and the requirements for its intended use and performance.

**The various types of testing on the system are:**

•Unit testing

•Integrated testing

•Validation testing

•Output testing

•User acceptance testing

**UNIT TESTING:**

It focuses on the smallest unit of the software design. Smallest unit include the

module which are integrated to produce the final project. The unit testing focuses on

the internal logic and data structures within the boundaries of the component. Test

considerations can be the data structures, boundary conditions, independent paths,

error handling paths, etc. Unit testing was done on verifying the email and password

for accessing the database. The following results were obtained:

**Test case for Login:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Cases** | **Username** | **Password** | **Test Result** |
| TC1 | Correct Username | Correct Password | Successful login,main page is displayed. |
| TC2 | Correct Username | Incorrect Password | Prompt saying,’Enter a valid username or password.’ |
| TC3 | Incorrect Username | Correct Password | Prompt saying,’Enter a valid username or password.’ |

**Test case for sem and grade:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test cases** | **Sem** | **Grade** | **Test Result** |
| TC1 | Sem not selected | Grade not selected | Displays result with all sems and all grades included. |
| TC2 | Sem selected | Grade not selected | Displays result for the selected sem and for all grades. |
| TC3 | Sem not selected | Grade selected | Displays result for all sems and the selected grade. |
| TC4 | Sem selected | Grade selected | Displays result for the selected sem and selected grade. |

**INTEGRATION TESTING:**

When all units are combined, we may not produce the desired major function; global data

structures can present problems. Integration testing is a symmetric technique for

constructing tests to uncover errors associated with the interface. All modules are

combined in this testing step. Then the entire program is tested as a whole.

**VALIDATION TESTING:**

At the culmination of integration testing, software is completely assembled as a package.

Interfacing errors are corrected and final series of software test-validation

testing begins. Validation testing can be defined in many ways, but a simple definition is that

validation succeeds when the software functions in a manner that is reasonably expected by the

consumer. Software validation is achieved through a series of black box tests that demonstrate

conformity with the requirement. After validation test has been conducted, one of two conditions

exists.

The function or performance characteristics confirm to specifications that are accepted.

A validation from specifications is uncovered and a deficiency created.

Deviation or errors discovered at this step in this project is corrected prior to completion of

the project with the help of the user by negotiating to establish a method for resolving deficiencies.

Thus the proposed system under consideration has been tested by using validation testing and

found to be working satisfactorily.

**OUTPUT TESTING:**

After performing the validation testing, the next step is output testing of the proposed

system. Here the output is considered in two ways: one is onscreen and the otheris printed format. The output format on the screen is found to be correct as the format is designed in the system design phase according to the user needs. As far as hard copies are considered it goes in terms with the user requirement. Hence output testing does not result in any correction in the system.

**BLACK-BOX TESTING:**

Black- Box testing refers to tests that are conducted at the software interface. A Black-box

test examines some fundamental aspects of a system with little regard for the internal logical

structure of the software.

The software developed was subjected to black-box testing to test the functionality of the

Graphical User Interface without checking the internal logic of the program. For example, the use

of back button resulting in the appearance of right frames etc.

**USER ACCEPTANCE TESTING:**

User acceptance of the system is a key factor for success of any system. The system under

consideration is tested for user acceptance by constantly keeping in touch with the prospective system

and user at the time of developing and making changes whenever required.

**Chapter 7**

**CONCLUSION AND FUTURE ENHANCEMENTS**

**CONCLUSION:**

Our project is an initiative towards easing the work of the college administration and also help the students with required examination related resources.

This application provides efficient way of generating and automating provisional marks card and transcript for students. The provisional marks card and transcript generated provides with accurate data.

The student portal provides with unambiguous and accurate information. They can check their CIE as well. Students need not go to college to view the results anymore.

Teachers can upload question papers, marks and resources easily for the benefit of students.

This portal can also be used by the administrator to add details of new teachers or students into the database.

Most importantly, it implements blockchain into SQL database and avoids hackers from compromising the database or system, as any changes made by the administrator or teacher in marks or student details would be kept as a ledger.

Hence it acts to demonstrate Proof Of Concept for implementing Blockchain into SQL.

**FUTURE ENHANCEMENTS:**

This application can be further extended by adding more features that will make it more scalable, portable, interactive and convenient for the users.

Some enhancements that can be introduced in the project are:

* More data analysis and visualization can be done for helping both teachers(for their respective subjects) and students, so that the students can work on their areas of weaknesses and teachers can mentor them.
* When the results of the sem-end examinations are out, students can receive a mail regarding the announcements of results. This will be a convenient methodology without students having to check their portal to see if results have been announced.
* During internals and examinations, students are expected to be seated in their respective allotted seats and classrooms. Students have to check their seating minutes before the internals and examinations. Hence, students can be mailed about the same so that confusion is avoided.

**Chapter 8**

**REFERENCES**

**WEB RESOURCES:**

http://stackoverflow.com/

https://www.tutorialspoint.com/flask/

https://www.tutorialspoint.com/flask/flask\_sqlite.htm

https://medium.com/@MicroPyramid/how-to-create-pdf-files-in-python-using-pdfkit-203c6464d92

https://passlib.readthedocs.io/en/stable/

http://gauravvichare.com/how-to-password-protect-pdf-file-using-python/

https://www.geeksforgeeks.org/send-mail-attachment-gmail-account-using-python/

**TEXTBOOKS:**

“DATABASE SYSTEMS” – Ramez Elmasri, Shamkant B. Navathe.

***“***Database System Concepts” - Henry Korth, S.Sudarshan and Abraham Silberschatz

“THE DATABASE BOOK – PRINCIPLE AND PRACTICE USING

“MYSQL” – Narain Gehani.