STUDENT ENROLLMENT ANALYSIS SYSTEM (SEAS)

FINAL PROJECT REPORT
GROUP – 05

NAME	ID
MD ZAKARIA KABIR	1921371
SUMAIA ANJUM SHABA	1920932
AHAD AL TANVIR AHMED	1920468
QUAZI MUNZUR E ELLAHI RUHIN	1930274
FAYEEZA SUBAH ISLAM	1920044
NUSRAT JAHAN	1921793

CONTENTS

LIST OF FIGURES

CHAPTER 1- INTRODUCTION

A student is admitted to a university under a certain degree program. Each program is assigned to a department, which is listed under Schools. Students generally take courses according to their program's curriculum, which typically includes:

- Business & Entrepreneurship
- Engineering, Technology & Sciences
- Environment and Life Sciences
- Liberal Arts & Social Sciences
- Pharmacy and Public Health

The purpose of this report is to examine IUB's current student enrollment analysis system, conduct the necessary process analysis, and propose a new and improved system that reduces error, makes data analysis, report, and chart generation easier for all stakeholders, and produces/shows valuable information for IUB and its collaborators in order to make necessary improvements. The first section delves into the specifics of the organization in issue as well as the project we've undertaken for them. The second section focuses on the existing system and its flaws, as well as an introduction to the proposed system that would be used to replace it. The third and fourth sections will be highly technical, focusing on how we intend to implement the proposed system.

According to our investigation into the current system of student enrollment analysis, we've discovered places where some important modifications are necessary to make the system more efficient and facilitate communication among stakeholders. Furthermore, the modifications will eliminate the possibility of human mistake and data duplication, and stakeholders will be able to access a vast dataset and review valuable information through our system rather of reading through documents manually.

As we progress through the report, we'll look at how the present student enrollment analysis system works, the business processes involved, where there are concerns and difficulties with data management, and how we can create a better system to remedy and enhance these problems.

A. BACKGROUND OF THE ORGANIZATION

Established in 1993, Independent University, Bangladesh (IUB) is one of Bangladesh's oldest private institutions, with an estimated 8,423 undergraduate and graduate students, 13,745 alumni, and 401 faculty members. [1]

Our academic curriculum is based on a North American liberal arts model and the medium of instruction is English. We currently have six academic schools: School of Business, School of Engineering and Computer Science, School of Environmental Sciences and Management, School of Liberal Arts and Social Sciences, School of Life Sciences and School of Public Health. [1]

Only by being disciplined and up to date with the on-going curriculum and development system has IUB demonstrated outstanding results in creating graduates with marketable abilities over time. Investing in IUB's departments, particularly the Department of Computer Science and Electrical Science, in order to turn it into a well-funded research facility with several research programs. IUB is likewise dedicated to developing international-standard graduates who are primarily prepared to lead the national economy through skilled employment, entrepreneurship, and/or applied research.

B. BACKGROUND OF THE PROJECT SEAS

The project's goal is to develop a software system for stakeholders and institutional bodies to retrieve statistical data on school-level revenue analysis, classroom requirement analysis based on class slot, classroom allocation recommendations based on IUB best resource utilization, and IUB resource calculation.

C. OBJECTIVES OF THE PROJECT

To meet the purpose of the project several requirements must be met.

Our system SEAS will provide the users with the requested analysis and allow them to view it in tables and particular charts. The stakeholders – department/school/ High authority/Registrar's office will log into our system and select which analysis that is class requirement per slots, revenue analysis, resource utilization, available resource they want to see. The system will take the necessary information from SEAS database which is the extracted information from the provided tally sheet. SEAS will form a table on classroom requirements based on class size, total slots per day for any individual semester and if the user request to see a pie chart on distribution of the classrooms per class size, then the system based on the table formed will generate a pie chart. SEAS will generate table on based the number of sections offered per school for selected class size with respect to the total number of sections in every semester for entire university. The system will provide an analysis for the number of classrooms available per class size and number of classrooms required. It will also provide a revenue analysis based on each department with

respect to the school for every semester and show the percentage change in the revenues. Based on available resource our system will generate tables and charts to show the resource utilization, comparing enrollment per course and room capacity for every semester.

D. SCOPE OF THE PROJECT

We carried out a thorough evaluation of the existing system and identified areas in the business processes that might result in significant delays in time and communication, which we will cover in the following chapter.

Our solution is to build a Web application called SEAS (Student Enrollment Analysis System) that uses a Relational Database Management System (RDMS) to store, calculate, add, and update necessary data for student enrollment, as well as to generate the necessary data, reports, charts, and documents and store them in the database.

To ensure that a project is completed, the scope of the project must be defined. We must guarantee that the new system is more effective than the present one because we are modifying an existing system. Because maintaining thorough records of students' enrollment and documentation is inefficient, an improved and automated student enrollment analysis system is required (SEAS).

So, the goal of this project is to improve the present system by implementing our proposed solution, which will allow us to:

Create a system which takes input with an easy to go interface.

Allow only authorized users to access the system.

Insert, update, and delete data from the database in real time.

Calculate the revenue based on school , analysis the IUB resource , classroom allocation based on best resource utilization.

Generating table and chart.

CHAPTER 2-REQUIREMENT ANALYSIS

We will provide us with information on each stakeholder and their interactions. The Requirement Analysis entails researching and visualizing the present system and processes that go into a company's day-to-day operations, utilizing industry tools, methodologies, and standards. "The process of deciding what the database will be used for is referred to as requirements analysis." Interviews with user groups and other stakeholders are conducted to determine what database capabilities they want, the types of data they desire to handle, and the most frequently done activities." This analysis will provide us with information on each stakeholder and their interactions. We analyze business processes using simple notations and symbols to give everyone an understanding of how they function. First, we'll run into challenges with the present system's manual labor related student enrollment analysis, followed by third-party persons generating faults in the system.

A. RICH PICTURE – EXISTING BUSINESS SYSTEM

Rich pictures are modelled using diagrams and illustrated graphically which has the main elements and relationship they have between them according to the business system in place. A complete rich picture could be of significance to other stakeholders of the problems in an existing system and permit them to apprehend many different facets of the situation. Both the structure and the processes of a given situation are focused on Rich Picture. The Rich Picture Analysis also considers the following:

- Structures
- Processes
- Climate
- People
- Issues expressed by people
- Conflict

The following rich picture was created keeping that in mind.

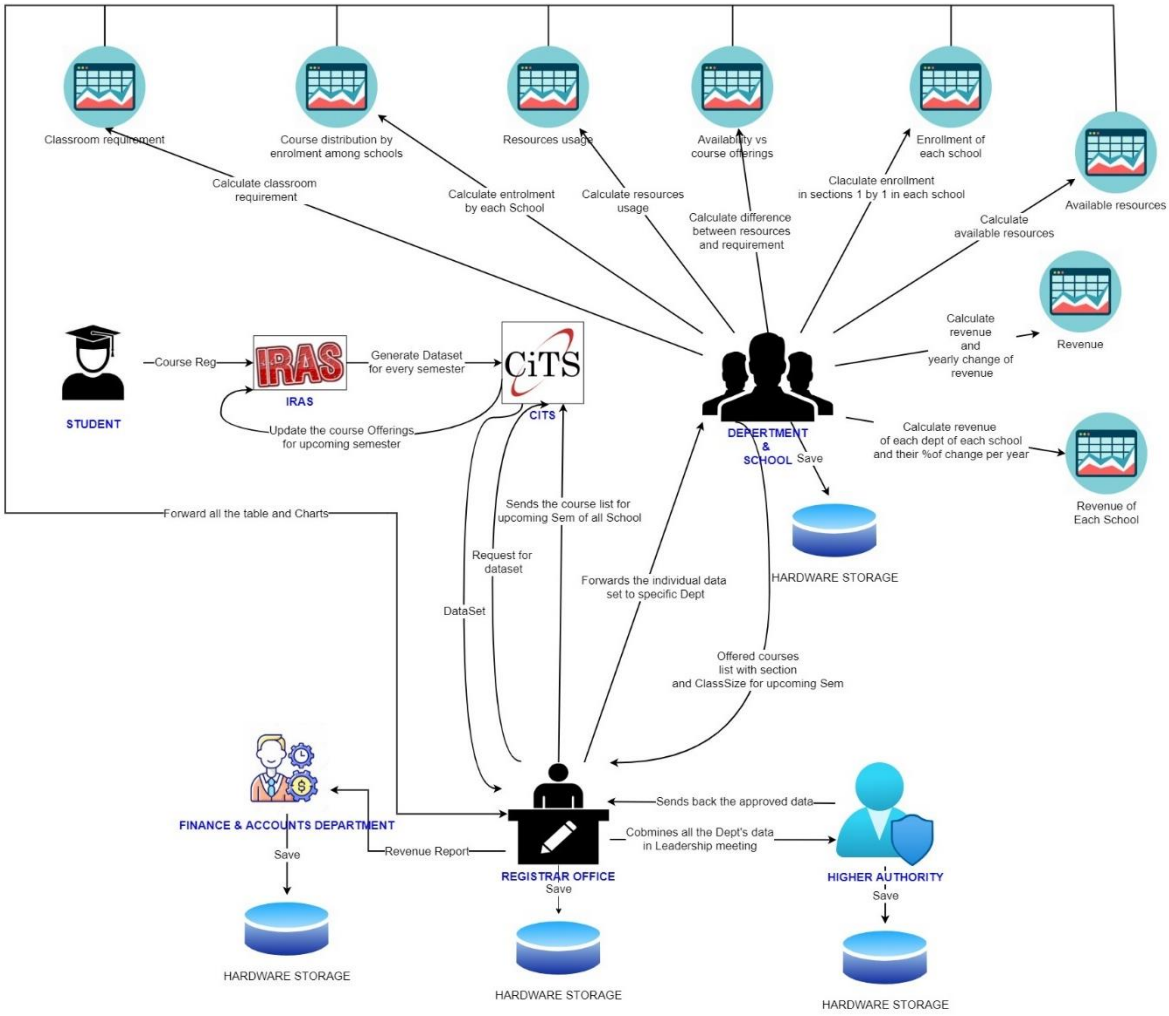


Figure 1: Rich Picture of Existing System

The Rich Picture Analysis shows us that we have the following types of stakeholders:

1. Department & School
2. Registrar's office
3. Higher Authority: VC/ Pro-VC/ Board of Trustees
4. CITS

We can also identify four separate storage systems or facilities, namely:

1. The Department and School Storage
2. The Registrar's Office Storage
3. Higher Authority's storage

B. SIX ELEMENTS ANALYSIS - EXISTING BUSINESS SYSTEM

The Six Elements Analysis provides a detailed description of the role of each element in each process. It is clear from the table below that Human entities dominate all key functions of this system. The existing system, for example, is substantially reliant on manually processed and managed hardcopy databases. As a result, before the Human parts can fulfill their end of the bargain in the process, there is a large amount of waiting between interdependent operations.

Process	System Rules					
	Human	**Non-Computing hardware**	**Computing hardware**	**Software**	**Database**	**Network and communication**
1. Datasheet Distribution	**CITS:** 1. Collects the generated dataset for every semester from IRAS. 2. Sends the generated dataset to the register's office 3. Updates the course offerings for upcoming semester according to the information provided by register's office **Registrar's office:** 1. Dataset for	**Pen and paper:** 1. Is used for writing down important notes specific to departments. 2. Intermediate ideas may be written down.	**Computer / Laptop:** 1. Is used to make soft copies of the dataset as well as extra copies for later use. **Printer:** 1. Used for printing out hardcopies of the dataset. **Networking Devices (Router, Switch, Bridge, Hub):** Used to access the Internet.	**Microsoft excel:** 1. It is used by the head of the departments as well as finance and accounts department to map all the calculations related to student enrollment and revenue. 2. To generate charts, MS excel is also used.	**Hardware storage:** 1. Is used for saving datasets in all the local computers associated with departments and higher authorities.	**Internet & Email:** 1. Internet and email is used to communicate between the register's office and all the departments along with CITS and higher authority. 2. All stakeholders may discuss important topics via email. **Others:** 1. Phone's or physical means may be used to

	specific departments are forwarded accordingly to the respected department heads and revenue report is sent to the finance and accounts department 2. The manually calculated charts and tables from the departments along with the offered courses list as well as section and class size offerings for upcoming semester are received by register's office 3. Combines all the received data then sends and waits for approval from higher authority. 4. Sends CITS the course list for					communicate about related topics and outcomes between the stakeholders.

	upcoming semester of all school based on the approved data from higher authority **Head of department:** Sends offered courses list with section and class size for upcoming semester to the register's office. **Higher Authority:** Verifies all the combined data and sends the approved data to the register's office					
2. Enrollment wise course distribution analysis	**School (Team from school):** 1.Each school takes the tally sheet. 2. Count the number of sections with respect to the number of students	**Pen and Paper:** 1. To note down the necessary information if needed. 2. Used for any hard copy of the charts and tables if needed.	**Computer / Laptop:** 1. Users will use the device to access and view the data. 2. Storing the necessary information of softcopies such as the	**Microsoft office:** 1.Generate tables and manage them in MS Excel. **Web Browser:** To send and receive mails about important details and	**Hardware Storage:** Stores the information of the table on enrollment wise course distribution.	**Internet:** Use the internet to access IRASv1 and Gmail.

	starting from 1 going to the largest with an increment of 1 3. Calculates the total number of sections with respect to the enrollment range. 4. Every school sends the total calculation with respect to the enrolled range to the registrar's office. **Registrar's office:** 1. Receives the calculated tables (in excel file) with respect to the enrolled range from individual schools. 2. Calculates the total sections of all the schools with respect to enrolled range.	**Calculator:** For calculating all the data.	excel file. **Printer:** Print the relevant documents. **Networking Devices (Router, Switch, Bridge, Hub):** Used to access the Internet.	documents.		

	3. Creates line charts for the total sections of all the schools with respect to enrolled range. 4. Creates bar charts based on the total enrolled range for every school.					
3. Revenue Analysis	**Department Team:** 1. Team from Each Dept. Collects the revenue Data from the Tally sheets 2. Starts to Calculate annual revenue data 3. Initially open the Excel software and Load the required revenue data 4. In order to calculate the annual % change in revenue , we need to set a formula to	**Pen and Paper:** 1. To note down the necessary information if needed. 2. Used for any hard copy of the charts and tables if needed. **Calculator:** For calculating all the data	**Computer:** 1. Department/ CITS/Higher authority use computers to make soft copies of calculation, charts, tables data 2. All related data is searched and stored using a computer. **Printer:** 1. To print out hard copies of the generated dataset. **Networking Devices (Router, Switch,**	**MS Excel:** All related information is stored and calculated and all charts and tables are generated.	**Hardware storage:** Used By Department/ CITS/Finance Accounts/Higher authority to store data. **File Cabinet:** Used by Department/ CITS/Finance & Accounts/Higher authority to maintain the necessary course related documents in hard copy.	**Internet & Email:** Used it to send mail to the Stakeholders to discuss information regarding courses list/capacity.

	find the total revenue of the departments. 5. Now set another formula to calculate the % change in revenue (e.g autumn 2009 to autumn 2010). 6. Now, the revenue data sheet prepared by each team of the departments is sent to the team of School **School (A Team from School):** 1. The Revenue data sheet prepared by each team of department is received by a team from School. 2. Now the team of school will combine the revenue data sheet into one using Excel.		**Bridge, Hub):** Used to access the Internet.			

	3. After combining all the revenue data semester wise . 4. Now, add all the semester wise revenue data to get the total sum of all departments (school e.g SETS). 5. Now set a formula to calculate the annual % change in revenue of the school (e.g autumn 2009 to autumn 2010). 6. The final revenue data that has been prepared will be sent to the higher authority for further analysis. **Registrar's office/ Higher Authority:** Receives the calculated excel sheets from each					

	school.					
4. Resources analysis	*a. Usage of the resources:* **School (A team from school):** 1. From the total number of enrolled students divide that value with the number of all sections for that school. This will generate Avg Enroll of a school in a semester. 2. Finding subtraction of Avg room and avg enrollment will give a difference of avg unused resources by each school in a semester. 3. To get %of unused resources divide the difference	**Pen and Paper:** 1. To note down the necessary information if needed. 2. Used for any hard copy of the data.	**Computer/ Laptop:** 1. Department / Registrar Office/ Higher authority use computers to make soft copies of data 2. All related data is searched and stored using a computer. **Printer:** To print out hard copies of the generated dataset. **Networking Devices (Router, Switch, Bridge, Hub):** Used to access the Internet. **Calculator:** For calculating all the data	**MS Excel:** All related information is stored and managed.	**Hardware storage:** Used by Department Heads /Registrar Office/Higher authority to store data. **File Cabinet:** By Department Heads /Registrar Office/Higher authority to maintain the necessary course related documents in hard copy.	**Internet & Email:** Used it to send mail to the stakeholders to discuss information regarding courses list.

	value with Avg room value then multiply by 100. 4. Send the calculated analysis of a school to the reg office. **Registrar Office:** 1. A central team combines all the school's resources usage analysis tables in Excel file in right format. 2. The team calculates overall semester's resources usage by all of the schools by using formulas in Excel file. *b. IUB available resources analysis:* **Registrar**					

	Office: 1. Calculate the available capacity of IUB based on class size and available class. 2.Using Class size* IUB resource formula a Central team finds the capacity table by class size as well as total available resource. 3. Then the team analyzes 6 slots or 7 slots per day capacity. 4. The team calculates the free% of capacity depending on 6/7 slots/day class allocation. *c. Availability and course offering comparison*					

	: **Registrar Office:** 1. The Central team calculates a comparison table based on class size, IUB resources vs offered courses in a semester. 2. The central team calculates the difference based on class size between available classroom and needed classroom in a semester. 3. Generate the charts for Resource Utilization in different semester(s) using Excel charts.					
5. Classroom requirement	**Registrar's office :** 1. A central	**Paper & pen:** 1. Need to	**Computer / Laptop:** 1. Users will	**Microsoft office:** 1.Make	**Hardware Storage:** Stores the	**Internet:** Use the internet to

analysis	team with respect to the total sections of all the schools based on the enrolled range, calculates the total number of slots required per day in excel sheet using the formula. 2. Sends the excel file to School for confirmation of distribution of classrooms per class size range on per slot. 3. Receives confirmation from each individual school. 3. Create pie charts showing percentage distribution of the classrooms per class size range based on per slot. **School(Team from School):**	take important notes for example during meetings.	use the device to access and view the data and use excel to design the table. 2.Storing the necessary information as softcopies such as charts and tables on classroom requirement summary. **Printer:** Print the relevant documents. **Networking Devices (Router, Switch, Bridge, Hub):** Used to access the Internet.	reports using MS Word. 2. Generate tables and manage them in MS Excel. **Web Browser:** To send mails about important details and documents.	information on classroom requirement summary.	access IRASv1 and Gmail.

	1. Receives the excel file containing slots per day based on class size range. 2. Checks if the slots calculated can be accepted or not. 3.After checking it sends the final excel sheet to the registrar's office.					
6. Accumulate school-based analysis	**School(Team from School):** 1. All the respective teams of each school sends the calculated table based on their school to the registrar office. The calculated tables include: i. Classroom requirement calculation	**Pen and Paper:** 1. To note down the necessary information if needed. 2. Used for any hard copy of the data.	**Computer/ Laptop:** 1. Department / Registrar Office/ Higher authority use computers to make soft copies of data 2. All related data is searched and stored using a computer. **Printer:** To print out hard copies	**MS Excel:** All related information is stored and managed.	**Hardware storage:** Used by Department Heads /Registrar Office/Higher authority to store data. **File Cabinet:** By Department Heads /Registrar Office/Higher authority to maintain the necessary	**Internet & Email:** Used it to send mail to the stakeholders to discuss information regarding courses list.

	ii. Calculation of enrollment by each school iii. Resource usage calculation iv. Calculation of differences between resources and requiremen ts. v. Calculation of enrollment in sections 1 by 1 in each school vi. Resource availability calculation. vii. Calculation of school wise revenue and yearly change of revenue. viii. Calculation of each department of each		of the generated dataset. **Networking Devices (Router, Switch, Bridge, Hub):** Used to access the Internet. **Calculator:** For calculating all the data		course related documents in hard copy.	

	school along with % change per year. **2. Central Team:** 1. Accumulates all the calculated tables of each school that was provided by the schools and places accordingly to the respected tables in an excel sheet. The tables here include: i. Revenue of IUB ii. Revenue in Engineering School. iii. Enrollment wise course distribution among the schools. iv. Usage of the resources v. Classroom requirement as per					

	course offering vi. Breakdown of number of courses of each school table. 2. Calculates some of the accumulated tables. The calculations include: i. The total revenue of all the schools in a certain semester. It is calculated by summing each of the schools revenue of a certain semester ii. The annual percentage change of revenue. iii. The total revenue of engineering schools along with the percentage					

	change annually. iv. The total enrollment among the schools is based on size. v. The total resource usage of a semester among all the schools including average enrollment, average room capacity, difference between them and unused resource percentage. vi. Total enrollment of all the schools of a semester based on incriminati on of 1 to the largest number of students. Also the end total. vii. Total sections needed for a certain range of					

	capacity of a semester also the total sections for all capacities. viii. Required classrooms for a given number of slots per day for a certain semester also the total number of classrooms for all capacities. 2. Generates charts on excel based on the accumulated data. The charts include: i. Class size requirement pie chart ii. Bar and line chart of class size distribution iii. Resource utilization bar chart . iv. Line					

	chart for Revenue Trend of the schools vii. Area chart on Revenue Distributio n among the schools viii. line and bar chart on IUB Revenue and Change % ix. Line chart on Department wise revenue in SETS. x. Area chart on Revenue of SETS.					

C. PROCESS MODEL – EXISTING BUSINESS SYSTEM

BPMN stands for Business Process Model and Notation, which is a graphical depiction of business processes in a business process model. Each of the business processes outlined in the preceding section is dissected using business process model diagrams. Each figure depicts the many stakeholders engaged in the processes, their interactions, and the decisions that each of them must make.

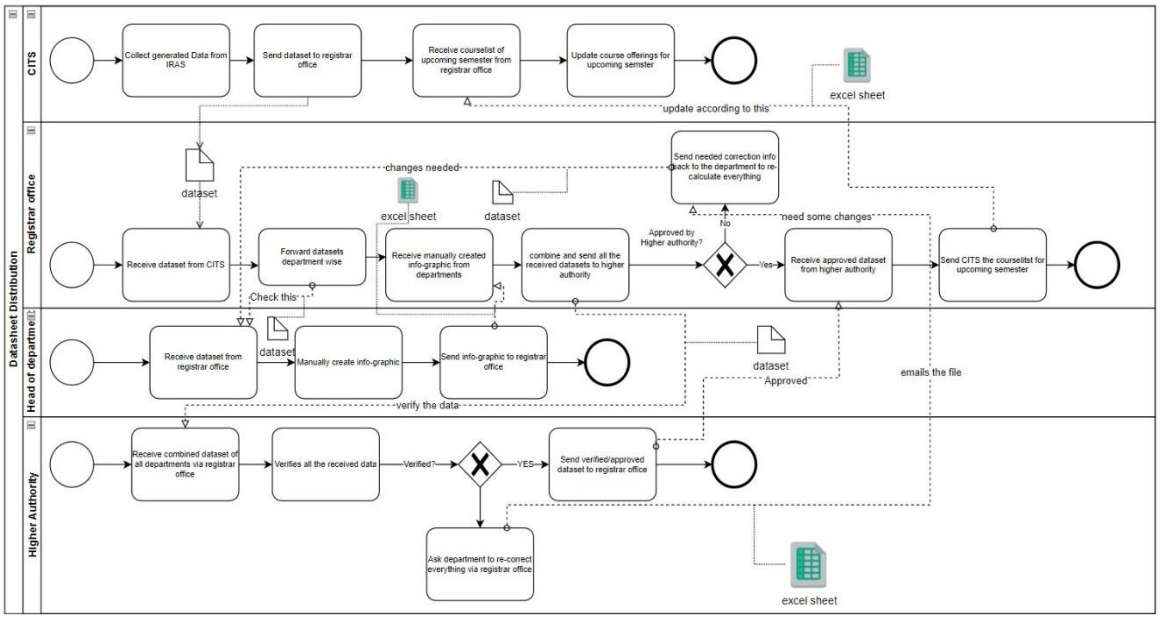


Figure 2: Datasheet Distribution

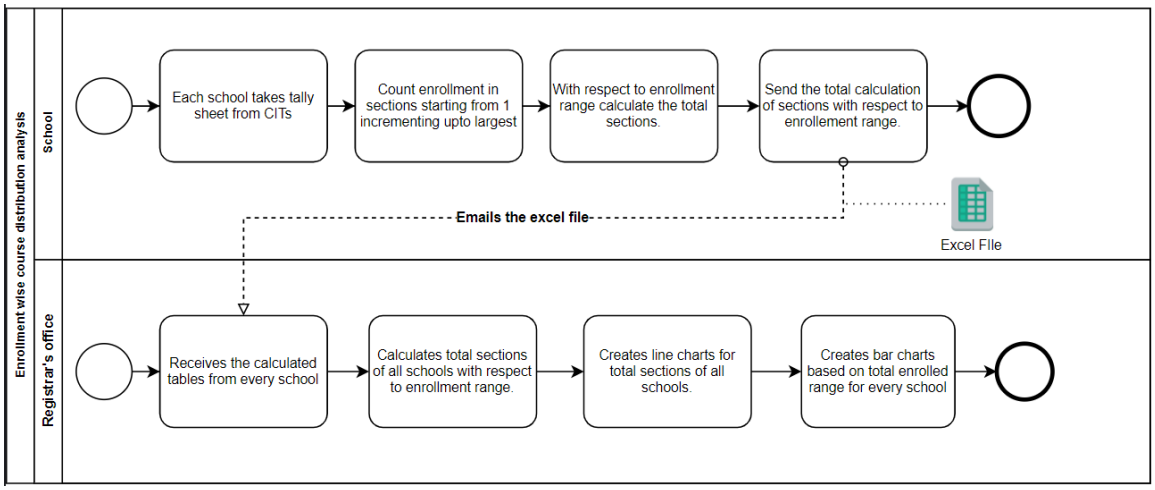


Figure 3: Enrollment wise course distribution analysis

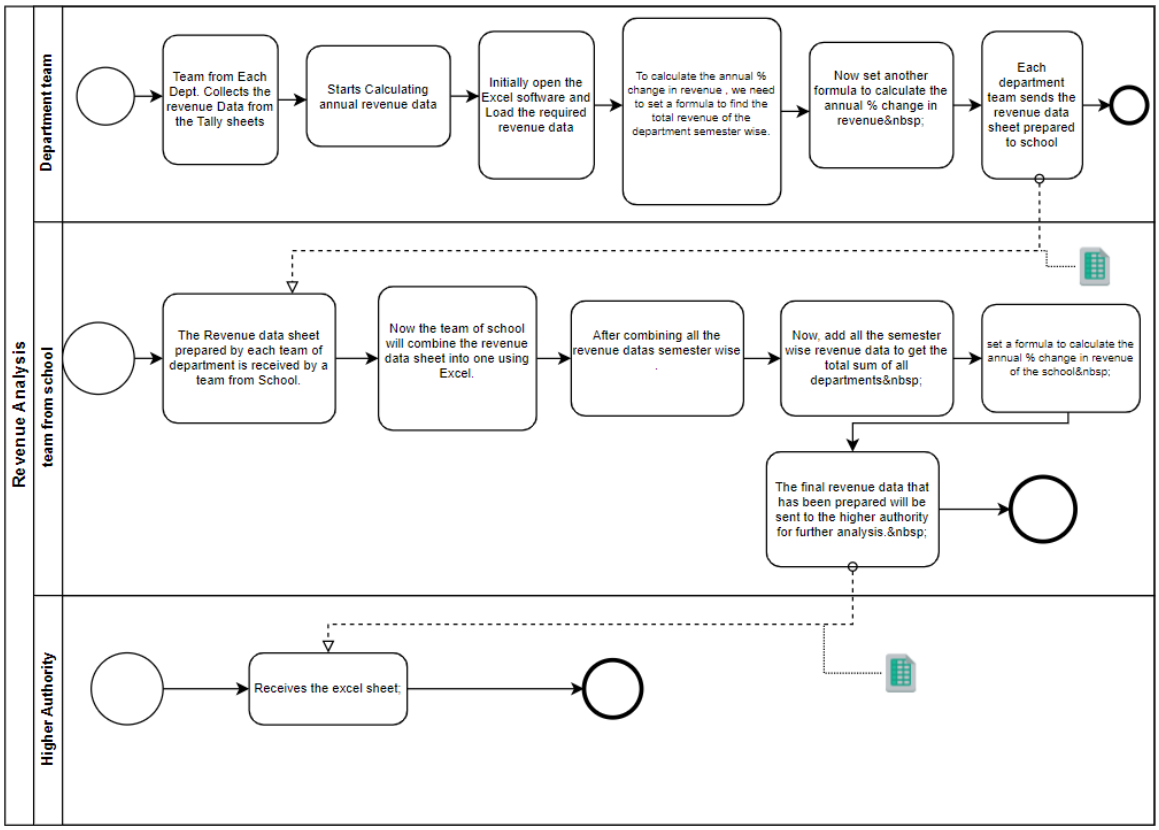


Figure 4: Revenue analysis.

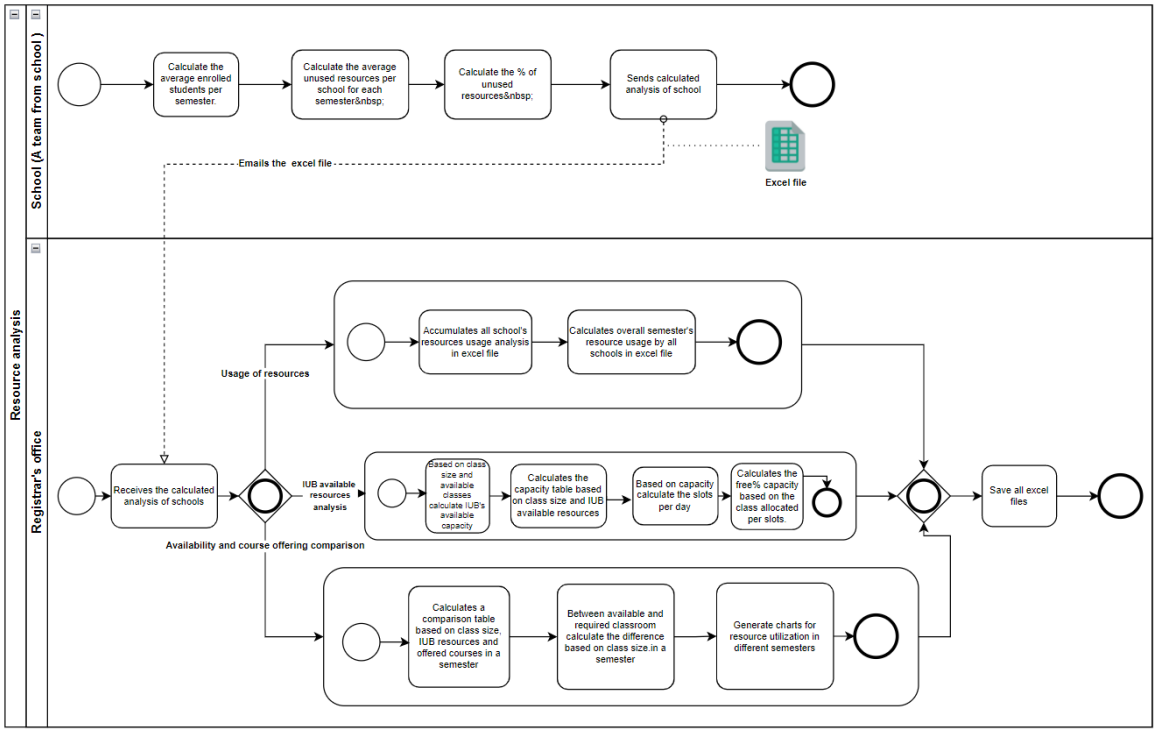


Figure 5: Resources analysis

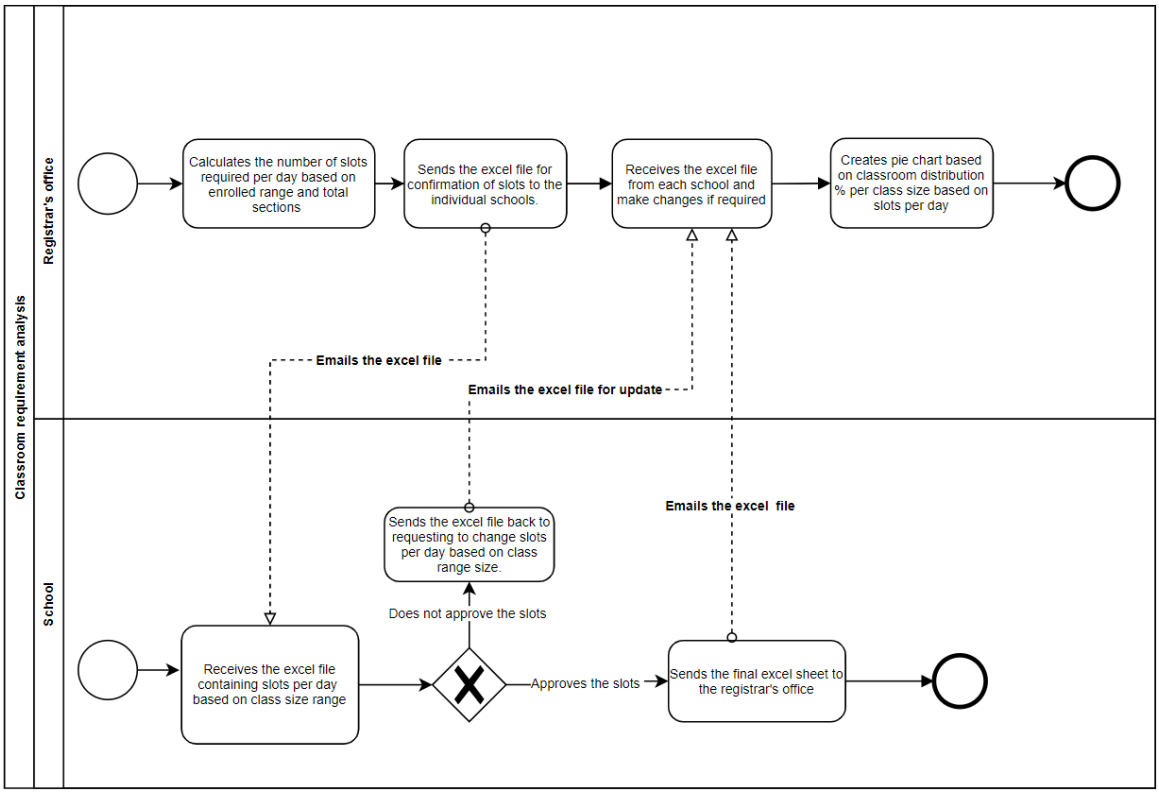


Figure 6: Classroom requirement analysis

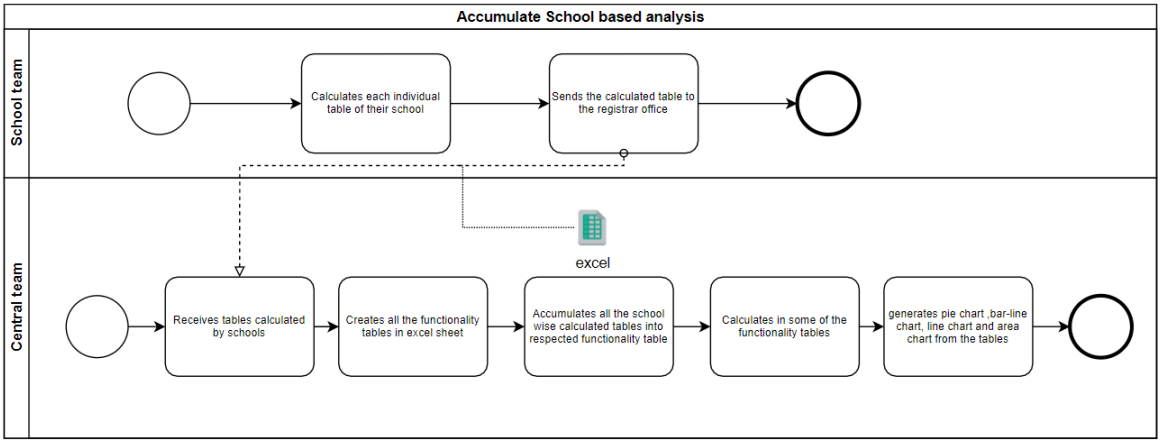


Figure 7: Accumulate School based analysis.

D. PROBLEM ANALYSIS – EXISTING BUSINESS SYSTEM

The deficiencies in each process were found using the Six Elements Analysis of the existing systems. The analysis column of this table contains a recurring pattern. The facilitation of a private online platform looks to benefit the system in a variety of ways.

Process Name	Stakeholders	Concerns(Problems)	Analysis (Reason of the Problems)	Proposed Solution
Datasheet distribution.	1. CITS 2. Registrar Office 3. Department 4. School 5. Higher Authority	1. Requires manual distribution of all the files 2. Data redundancy and variability 3. Time Consuming 4. Human error 5. Decentralized	The datasheet generated from IRAS needs to be distributed among the stakeholders to manually calculate their parts in different analyses. As there is no central system that will do all analysis irrespective of the department or school. In the current system each dept needs to do their part then with that each school needs to do their part finally a team needs to combine all, which is time consuming as well as there is huge possibility of data redundancy and inconsistency .	We proposed a system where there's no need for distribution. The system will work as a central system.

Enrollment wise course distribution analysis.	1. School 2. Registrar's Office	1. Time consuming 2. Human error 3. Data redundancy and variability 4. Involves manual calculations 5. Decentralized	Every school needs to calculate their course distribution part manually and then send them to the Registrar's office for combining all schools' analysis, which is a long process since one process needs to wait for another process to finish its task. As there is no central system that will do all the analysis irrespective of the schools the more human involvement the more human error can happen which is time consuming as well as there is huge possibility of data redundancy and inconsistency.	The system itself calculates and generates tables and charts according to the user request from the dataset it is provided. So no need for manual calculation or hassle for school or dept-wise data combination.
Revenue analysis.	1. Department 2. School 3. Registrar's office 4. Higher Authority	1. Involves manual calculations. 2. Human error 3. Time consuming 4. Data redundancy and variability 5. Incorrect data values	Every department and school needs to calculate their revenue analysis part manually and then send them to the Registrar's office for combining all schools' analysis sheets. As there is no central system that will do all the	The system itself calculates and generates tables and charts according to the user request from the dataset it is provided. So no need for manual calculation or hassle for school or dept data combined.

			analysis irrespective of the schools, this is time consuming as well as there is huge possibility of data redundancy and inconsistency. The datasheet also consist of some inexact values of data. The more human involvement the more human error can happen.	
Resources analysis.	1. School 2. Registrar's office	1. Involves huge data compilation 2. Time consuming 3. Data redundancy and variability 4. Human error 5. Decentralized 6. Incorrect data values	Analyze the resources usage by different schools individually. Overall usage needs to combine all school data which is a long process. As there is no central system that will do all the analysis irrespective of the schools, this is time consuming as well as there is huge possibility of data redundancy and inconsistency. The datasheet also consist of some invalid values of data. The more human involvement the	The system itself calculates and generates tables and charts according to the user request from the dataset it is provided. So no need for manual calculation or hassle for school or dept data combined.

			more human error can happen.	
Classroom requirement analysis.	1. Registrar's office	1. Involves manual calculations 2. Time consuming 3. Human error 4. Decentralized	A central team from all schools manually calculate the classroom analysis and then generate charts. For every semester doing this task manually takes a lot of time. As there is no central system that will do all the analysis irrespective of the schools. The more human involvement the more human error can happen.	The system itself calculates and generates tables and charts according to the user request from the dataset it is provided. Hence no need for manual calculation or hassle for school or dept data combined.
Accumulate school-based analysis.	1. School 2. Registrar's office	1. Involves huge data compilation 2. Data redundancy and variability 3. Human error 4. Time consuming 5. Decentralized	All the calculated excel sheets from all schools need to be combined and then overall university analysis is done. Since every dept, school does their calculation individually and saves that to their hardware storage a single data can be stored in multiple storage. Since there is no central system that will do all the analysis irrespective of the schools. The more human involvement the	The system itself calculates and generates tables and charts according to the user request from the dataset it is provided. Hence no need for manual calculation or hassle for school or dept data combined. Also all data is now in one place so no need to worry about data redundancy or inconsistency.

			more human error can happen. Different dept/ schools may save or generate data in different formats which may lead to difficulties while combining; moreover, if one calculation error is found anywhere or a data needs to be changed anywhere it needs to be changed or corrected everywhere which is a huge problem of the current system.	

E. RICH PICTURE - PROPOSED SYSTEM

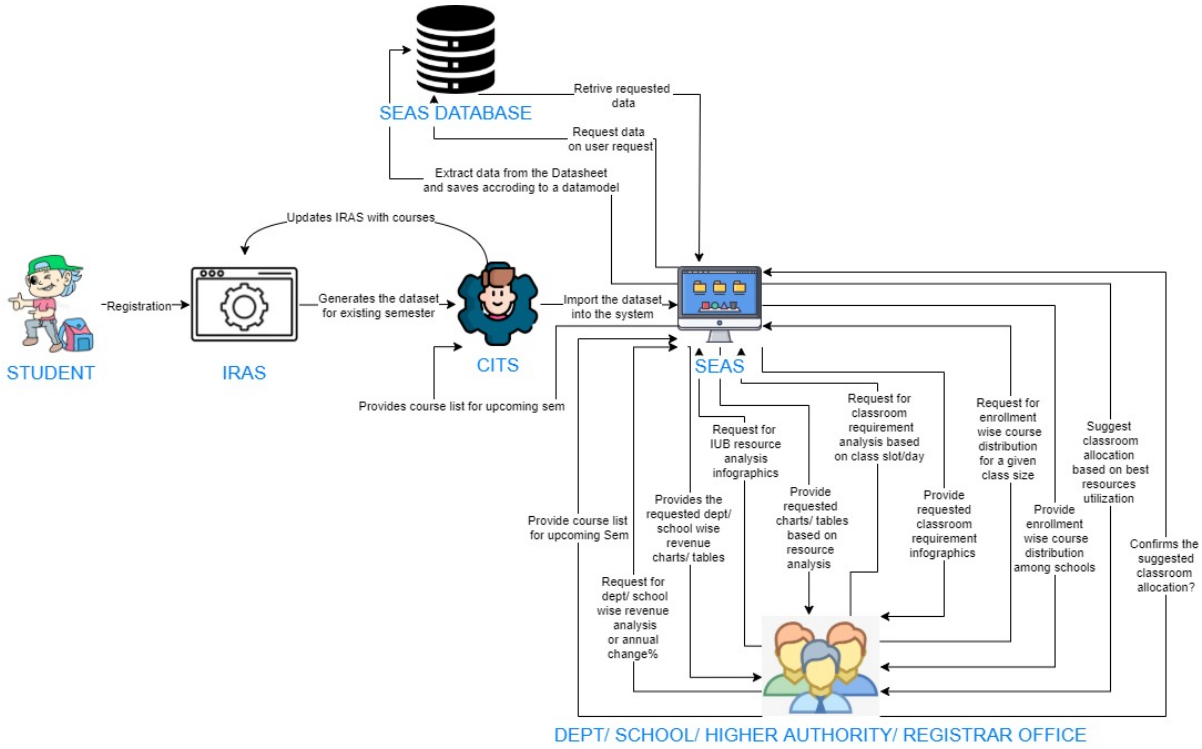


Figure 8: Rich Picture for Proposed System

F. SIX ELEMENTS ANALYSIS – PROPOSED SYSTEM

Process	System Rules					
	Human	**Non-Computing hardware**	**Computing hardware**	**Software**	**Database**	**Network and communication**
1. Import dataset into the system	**CITS:** 1. Retrieves the generated dataset from IRAS. 2. Upload the Dataset into the System.	**Pen and Paper:** 1. To note down the necessary information if needed. 2. Used for any hard copy of the data	**Computer:** 1. CITS uses computers to make soft copies of data. 2. All related data is searched and stored using a computer. **Printer:** To print out hard copies of the generated dataset. **Networking Devices (Router, Switch,**	**MS Excel:** All related information is stored. **IRAS:** Generates the dataset for the existing semester **SEAS:** Import the dataset into the system for further analysis.	**Hardware storage:** To store data. **SEAS Database:** The Imported Data will be stored into the System's Database.	**Internet :** Internet is used to import data from IRAS and store it into the System.

			Bridge, Hub): Used to access the Internet.			
2. View System generated Resources analysis.	**Department /School /High Authority/ Registrar Office:** 1. Logs into the System using UserID and password. 2. Select user type from User Type selection drop box. *(i) Usage of the resources:* 3. From the navigation menu select the option for Usage of the resources.		**Computer/ Laptop:** Users will need a computer to access SEAS. **Printer:** To print the infographic report if needed. **Networking Devices (Router, Switch, Bridge, Hub):** Used to access the Internet.	**SEAS:** The system will generate the requested infographic. **Web Browser:** To access SEAS.	**SEAS Database:** Contains all data. **Hardware storage:** To save the system generated infographic.	**Internet:** It is required to access the system.

	4. Select school from school selection checkbox, Semester from selection checkbox. 5. Click generate to view the usage table by selected school in the selected semester. 6. This will also show the overall usage by the selected semester(s). *(ii)IUB available resources:* 3. From the navigation menu select the option for IUB					

	available resources. 4. Selecting this will show a summary table of available resources based on the capacity of all university rooms and the quantity of such rooms. *(iii) Availability and course offering comparison:* 3. From the navigation menu select the option for Availability and course offering comparison. 4. Select type of infographics					

	(i.e. chart/ table), select semester from semester selection checkbox. 5. Clicking generate will show the selected chart/ table that is comparing the number of available classrooms per classroom size to the number necessary.					
3. View system generated Revenue Analysis.	**Department /School /High Authority/ Registrar Office:** 1. Logs into the System using UserID		**Computer/ Laptop:** User will need a computer to access SEAS. **Printer:**	**SEAS:** The system will generate the requested infographic. **Web Browser:**	**SEAS Database:** Contains all data. **Hardware storage:** To save the system	**Internet:** It is required to access the system.

	and password. 2. Select user type from User Type selection drop box. *(i)Revenue of IUB:* 3. From the navigation menu select the option for Revenue of IUB. 4. Select the type of infographic from the selection checkbox. 5. Select the school(s) from the school selection checkbox, Select Annual change% if the user		To print the infographic report if needed. **Networking Devices (Router, Switch, Bridge, Hub):** Used to access the Internet.	To access SEAS.	generated infographic.	

	wants to view the percentage of annual revenue change by a school. 6. Click generate to view the selected infographic. *(ii)Revenue Of a School:* 3. From the navigation menu select the option for Revenue Of a School. 4. Select the type of infographic from the selection checkbox. 5. Select the school from dropdown box.					

	6. Select the dept(s) from the available dept checkbox. 7.Click generate to view selected infographic of Department wise revenue in selected school. *(iii) Revenue by a Dept:* 3. From the navigation menu select the option for Revenue Of a School. 4. To view dept wise infographic Check Mark the Dept wise View option.					

	5. Select the Dept. 6.Clicking generate will show the Revenue trend as well as change% in revenue chart by that dept for all semester.					
4.View Classroom requirement analysis.	**Department /School /High Authority/ Registrar Office:** 1. Logs into the System using UserID and password. 2. Select user type from User Type selection drop box. 3. From the navigation menu select		**Computer/ Laptop:** Users will need a computer to access SEAS. **Printer:** To print the infographic report if needed. **Networking Devices (Router, Switch, Bridge, Hub):**	**SEAS:** The Software will propose course capacity based on previous semester data and resources. **Web Browser:** 1.Access important information from other sources such as the IUB	**SEAS Database:** 1. To provide required data based on user request. 2.SEAS retrieves data from the database, when needed. **Hardware storage:** To save the system generated infographic.	**Internet:** Use the internet to log in for accessing into SEAS.

	the option for Classroom requirement charts and tables. 4. Select the type of infographic from the selection checkbox. 5. Select the class size range from the dropdown box, select semester from semester selection checkbox, select class slot/day from selection checkbox. 6.Click generate to view the selected infographic.		Used to access the Internet.	webpage to view details about any courses. 2. To send mails about important details and documents. 3. To access SEAS.		

5.View enrollment wise course distribution analysis.	**Department /School /High Authority/ Registrar Office:** 1. Logs into the System using UserID and password. 2. Select user type from User Type selection drop box. *(i) Enrollment wise course distribution:* 3. From the navigation menu select the option for enrollment wise course distribution. 4. Select the type of infographic from the		**Computer/ Laptop:** Users will need a computer to access SEAS. **Printer:** To print the infographic report if needed. **Networking Devices (Router, Switch, Bridge, Hub):** Used to access the Internet.	**SEAS:** The Software will propose course capacity based on previous semester data and resources. **Web Browser:** 1.Access important information from other sources such as the IUB webpage to view details about any courses. 2. To send mails about important details and documents. 3. To access SEAS.	**SEAS Database:** 1. To provide required data based on user request. 2.SEAS retrieves data from the database, when needed. **Hardware storage:** To save the system generated infographic.	**Internet:** Use the internet to log in for accessing into SEAS.

	selection checkbox. 5. Select the class size range from the dropdown box, select school from school selection checkbox, select semester from semester selection checkbox. 6.Click generate to view the selected infographic. *(ii)View number of enrollments in sections 1 by 1:* 3. From the navigation menu select the option for					

	View number of enrollments in sections 1 by 1. 4. Select the school from the school selection checkbox, select semester from semester selection checkbox. 5.Click generate to view the table.					

G. PROCESS MODEL - PROPOSED SYSTEM

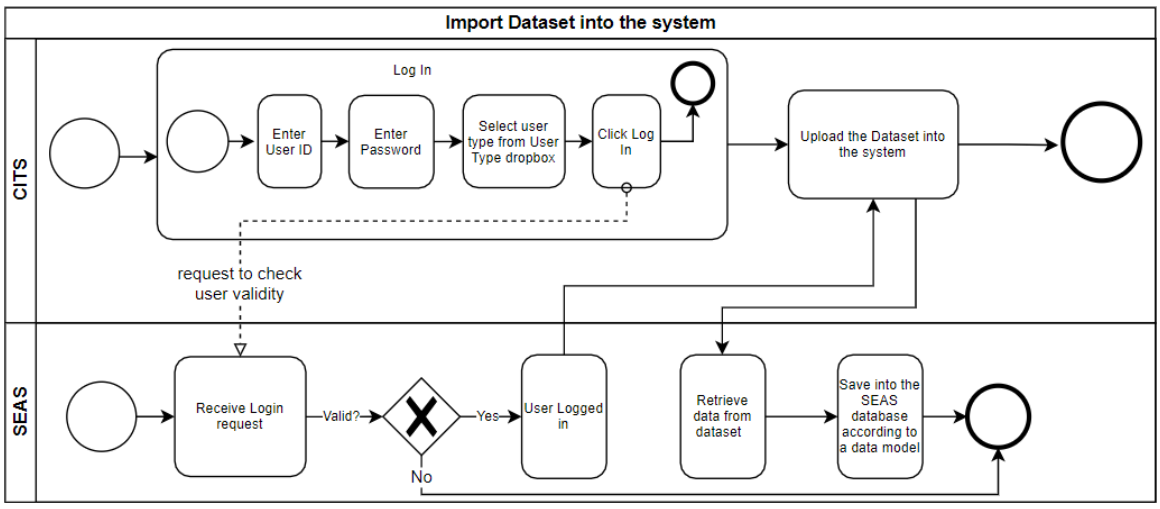


Figure 9: Import dataset into the system.

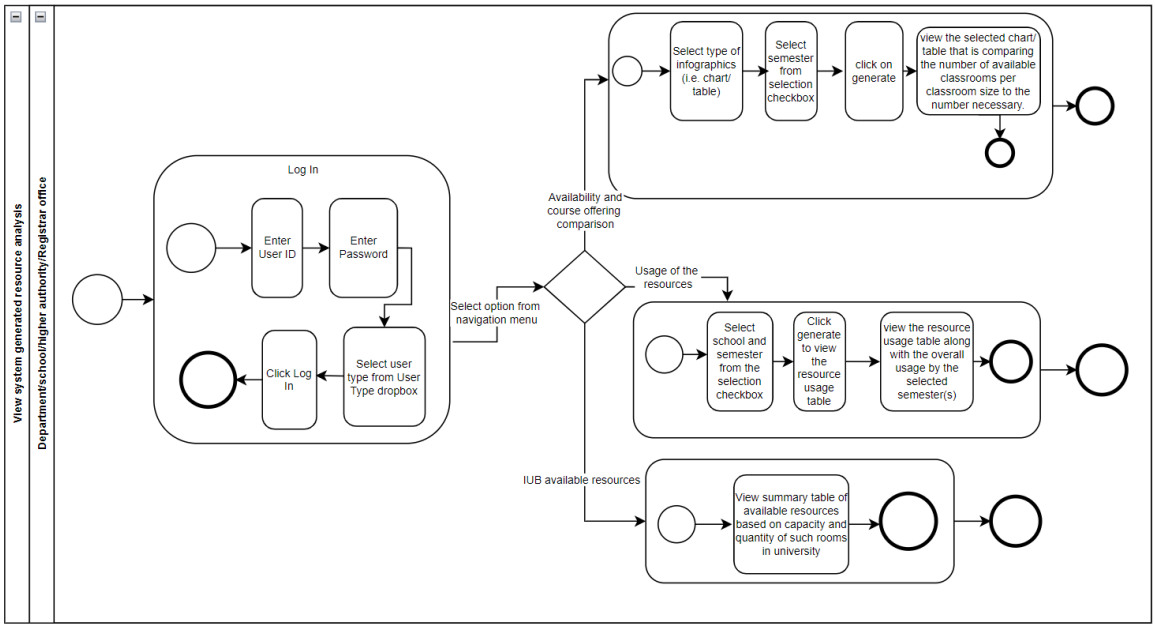


Figure 10: View System generated resource analysis.

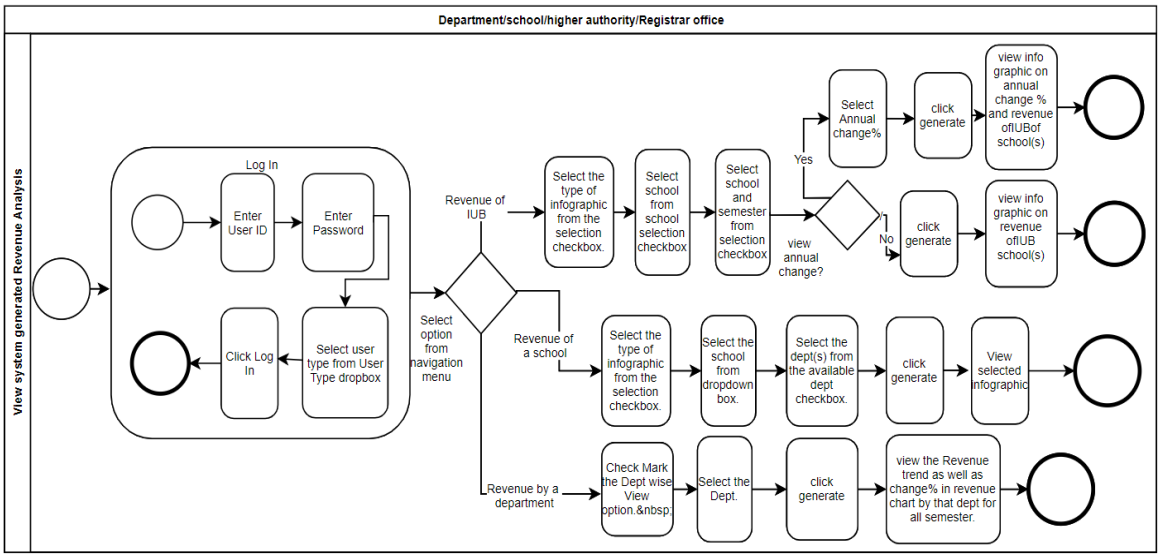


Figure 11: View system generated Revenue Analysis.

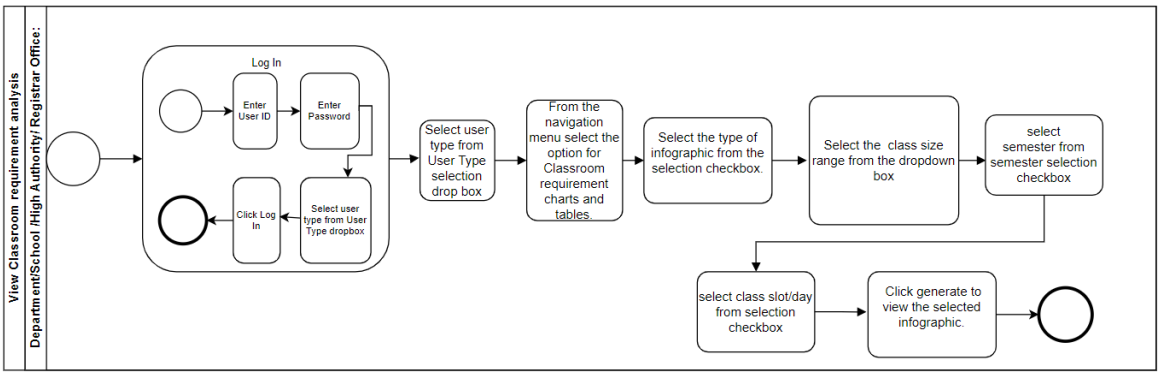


Figure 12: View Classroom requirement analysis.

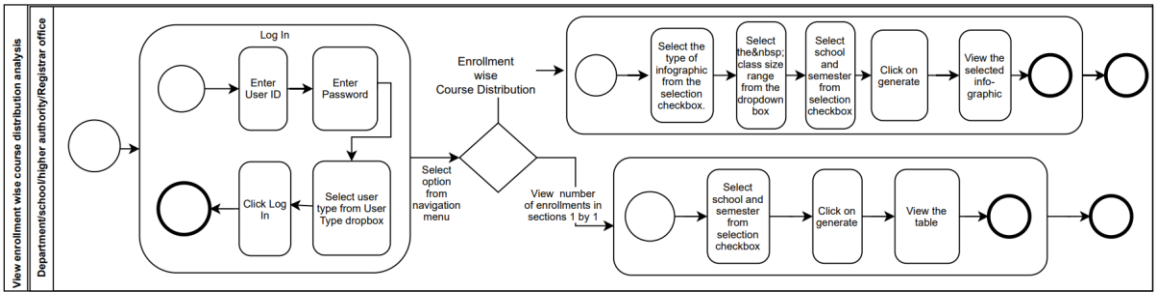


Figure 13: View enrollment wise course distribution analysis.

CHAPTER 3- LOGICAL SYSTEM DESIGN

A. BUSINESS RULE

1. A Faculty has a FacultyID, FacultyName.
2. A Department has a DeptID, DeptName. Each Faculty must belong to a Department. A Department must have one or more Faculties.
3. A School has a SchoolTitle, SchoolName. A Department must belong to exactly one School. A School must have one or more Departments. Each Faculty must belong to a School. A School must have one or more Faculties.
4. A Course has a CourseID, CourseName, CreditHour. Each Course must belong to exactly one Department. A Department must offer at least one Course.
5. A Course may have many co-offered courses as well as a course may be offered with multiple courses.
6. Each Course must have one or more Sections, where the existence of a Section depends on the existence of that Course. The SectionID can be uniquely identified using SectionNum, Semester and Year. A Section also includes SectionCapacity, SectionEnrolled, Blocked status of that section, Maximum Size of that section, FreeSpace which is always calculated based on SectionCapacity and SectionEnrolled, and TotalStudentCredit which is calculated by multiplying SectionEnrolled and CourseCredit. A Section must belong to exactly one course.
7. Each Section must be taken by a Faculty. A Faculty may take many sections.
8. A Room has a RoomID, RoomCapacity. Each Room may contain many Sections. A Section’s class must be held in one Room where the StartTime, EndTime and Day of the class must be recorded.

B. ENTITY RELATIONSHIP DIAGRAM

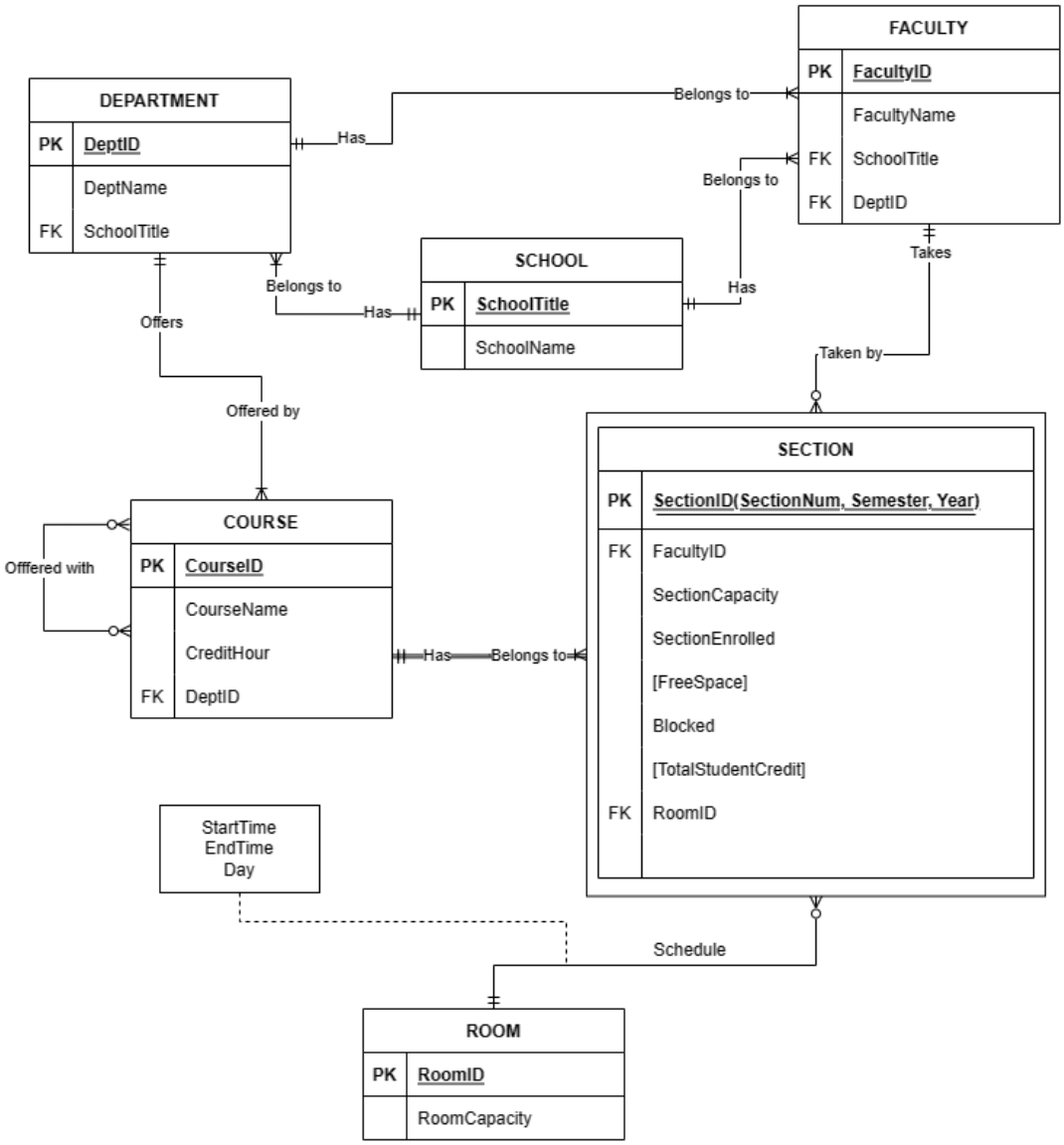


Figure 14: Entity Relationship Diagram

C. ENTITY RELATIONSHIP DIAGRAM TO RELATIONAL SCHEMA:

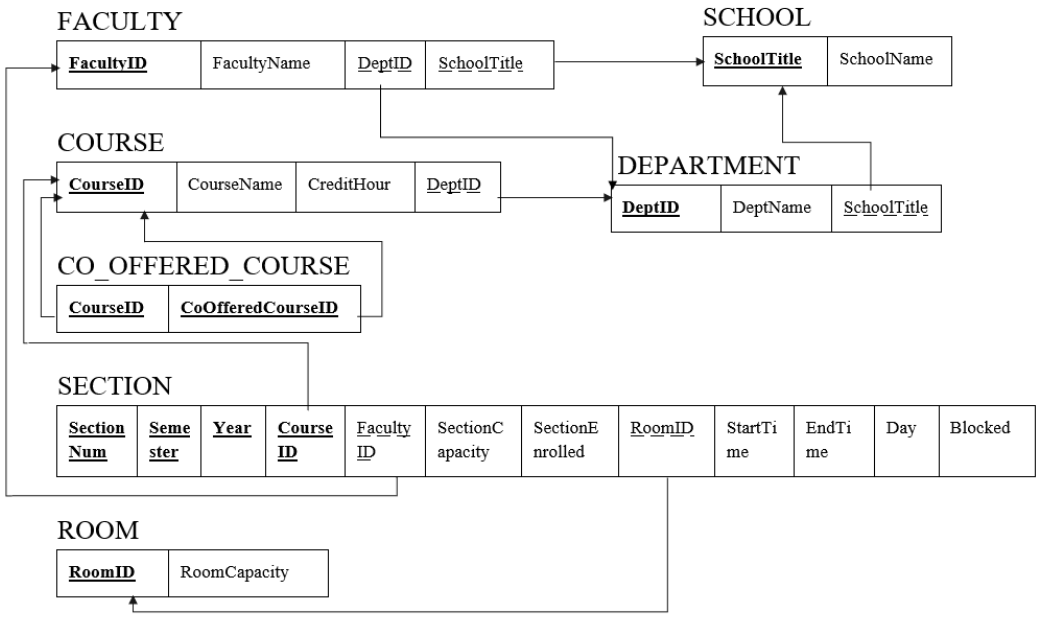


Figure 15: Relational Schema

D. NORMALIZATION

Functional Dependency:

FacultyID	→	FacultyName, DeptID, SchoolTitle
SchoolTitle	→	SchoolName
CourseID	→	CourseName, CreditHour, DeptID, CoOfferedCourseID
CoOfferedCourseID	→	CourseID
DeptID	→	DeptName, SchoolTitle
SectionNum, Semester, Year, CourseID	→	FacultyID, SectionCapacity, SectionEnrolled, RoomID, StartTime, EndTime, Day, Blocked
RoomID	→	RoomCapacity

1NF:

T1:

SectionNum	**Semester**	**Year**	**CourseID**	FacultyID	FacultyName	DeptID	DeptName	SchoolTitle	SchoolName

SectionCapacity	SectionEnrolled	RoomID	RoomCapacity	StartTime	EndTime	Day	Blocked	Course Name	CreditHour	CoOffered CourseID

2NF:

T11:

CourseID	CourseName	CreditHour	CoOfferdCourseID	DeptID	DeptName	SchoolTitle	SchoolName

T12:

SectionNum	**Semester**	**Year**	**CourseID**	FacultyID	FacultyName	SectionCapacity	SectionEnrolled	RoomID	RoomCapacity

StartTime	EndTime	Day	Blocked

3NF:

T113

FacultyID	FacultyName	DeptID

T112

DeptID	DeptName	SchoolTitle

T111

SchoolTitle	SchoolName

T114

CourseID	CourseName	CreditHour	DeptID	CoOfferedCourseID

T115

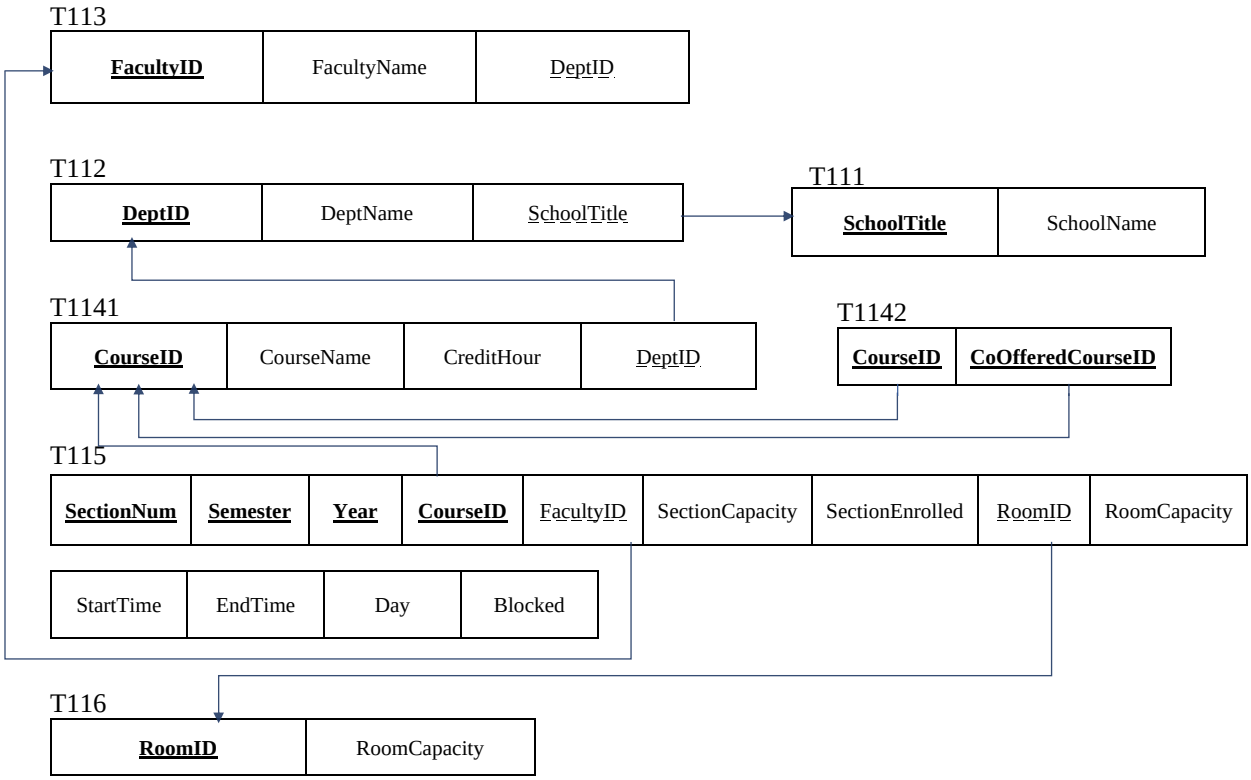
SectionNum	**Semester**	**Year**	**CourseID**	FacultyID	SectionCapacity	SectionEnrolled	RoomID	RoomCapacity

StartTime	EndTime	Day	Blocked

T116

RoomID	RoomCapacity

BCNF:



E. DATA DICTIONARY

tblschool:

Name	Data Type	Size	Remark
cSchoolTitle	VARCHAR	5	This is the primary key of School. E.g: “SETS”. This can’t be null.
cSchoolName	VARCHAR	50	This is the name of the school. E.g: “School of Engineering, Technology & Science”. This can’t be null.

tbldepartment:

Name	Data Type	Size	Remark
cDeptID	VARCHAR	6	This is the primary key for Department table E.g: “CSE”. This can’t be null.
cDeptName	VARCHAR	50	This is the name of the department.

			E.g: “Computer Science and Engineering”. This can’t be null.
cSchoolTitle	VARCHAR	5	This is the foreign key from School table. E.g: “SETS”. This can’t be null.

tblfaculty:

Name	**Data Type**	**Size**	**Remark**
nFacultyID	INTEGER		This is the primary key for Faculty table. E.g: “4434”. This can’t be null.
cFacultyName	VARCHAR	50	This is the name of the Faculty. E.g: “Sadita Ahmed”. This must not be nullable.
cDeptID	VARCHAR	6	This is the foreign key from Department table E.g: “CSE”. This can’t be null.

tblcourse:

Name	**Data Type**	**Size**	**Remark**
cCourseID	VARCHAR	7	This is the Primary Key for the Course table. E.g: “CSE303”. This can’t be null.
cCourseName	VARCHAR	100	This is the name of the Course. E.g: “Database Management”. This can’t be null.
nCreditHour	INTEGER		This is the number of credits for the Course. E.g: “3”. This can’t be null.
cDeptID	VARCHAR	6	This is the foreign key from Department table. E.g: “CSE”. This can’t be null.

tblcoofferedcourse:

Name	Data Type	Size	Remark
cCourseID	VARCHAR	7	This is the Primary Key for the coofferedcourse table and a foreign key from Course Table E.g: “CSE303”. This can’t be null.
cCoOfferdCourseID	VARCHAR	7	This is the Primary Key for the coofferedcourse table and a foreign key from Course Table. E.g: “CEN401”. This can’t be null.

tblroom:

Name	Data Type	Size	Remark
cRoomID	VARCHAR	9	This is the Primary Key for the Room table. E.g: “BC4017A-S”. This can’t be null.
nRoomCapacity	INTEGER		This is the capacity of the room. E.g: “50”. This can’t be null.

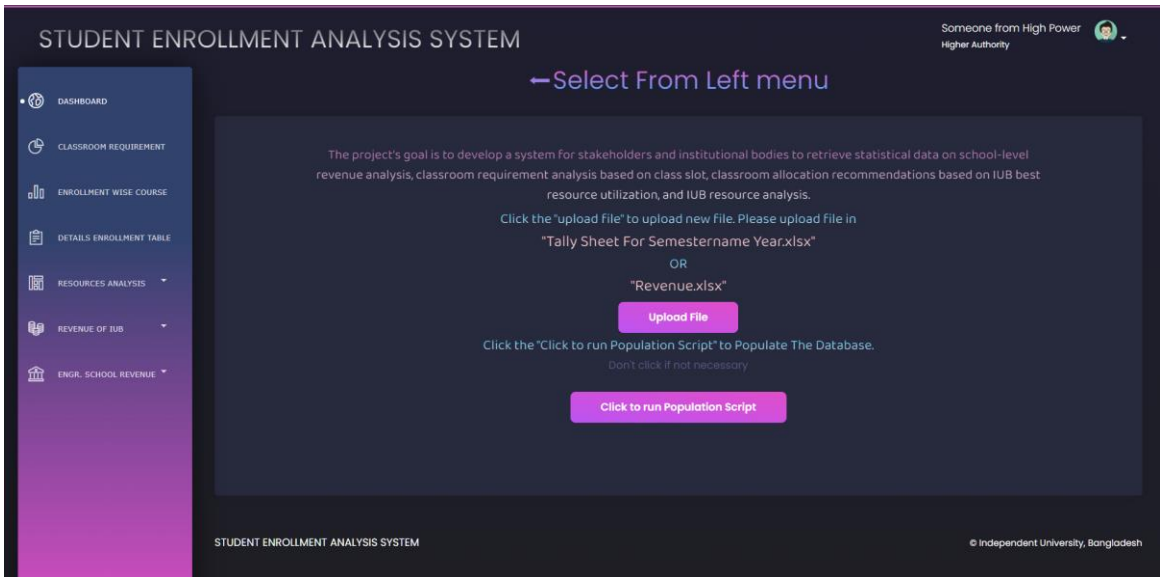
tblsection:

Name	Data Type	Size	Remark
nSectionNum	INTEGER		This is the primary key for Section table. E.g: “2”. This can’t be null.
cSemester	VARCHAR	6	This is the primary key for Section table. E.g: “Autumn”. This can’t be null.
dYear	YEAR	“YYYY”	This is the primary key for Section table. E.g: “2020”. This can’t be null.
cCourseID	VARCHAR	7	This is the part of the primary key for Section table and a foreign key from

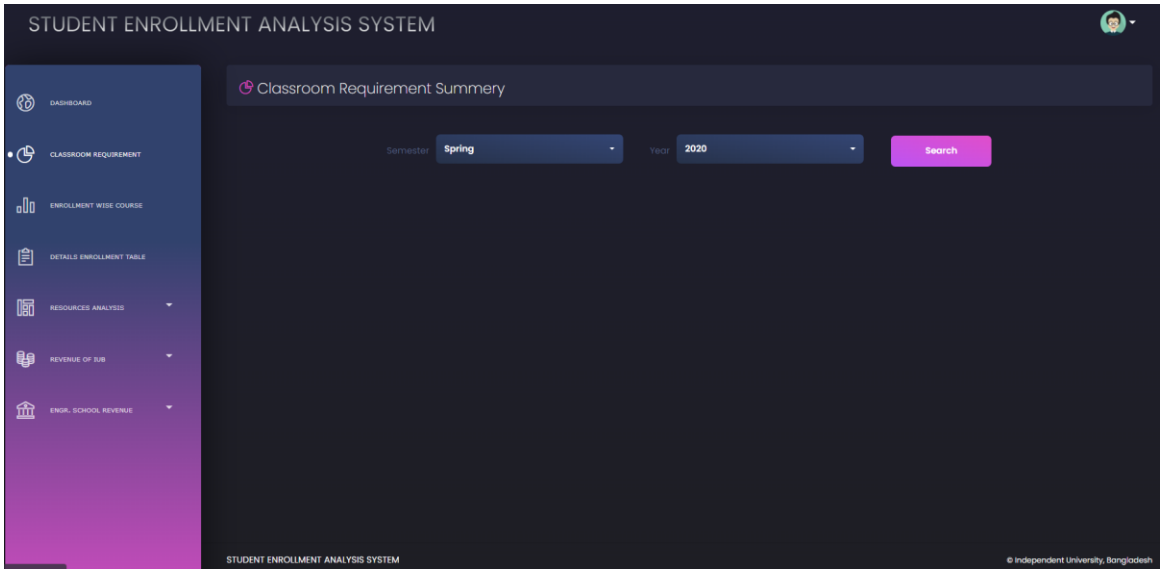
			Course table. E.g: “CSE303”. This can’t be null.
nFacultyID	INTEGER		This is the foreign key from Faculty table. E.g: “4434”
nSectionCapacity	INTEGER		This is the total capacity of the section. E.g: “30”
nSectionEnrolled	INTEGER		This is the number of students enrolled in a section. E.g: “29”
dStartTime	TIME	“hh:mm:ss”	This is the time when a section starts. E.g: “11:00”
dEndtime	TIME	“hh:mm:ss”	This is the time when a section ends. E.g: “17:00”
cDay	VARCHAR	10	This is the day of the Section’s class. E.g: “ST”
cRoomID	VARCHAR	9	This is the foreign key from Room table. E.g: “BC4017A-S”
cBlocked	VARCHAR	3	This shows if the section is blocked or not. E.g: “B-“
nMaxSize	INTEGER		This is the maximum capacity on the number of students allowed to enroll into a particular section. E.g: “250”

CHAPTER 4- PHYSICAL SYSTEM DESIGN:

A. INPUT FORM:



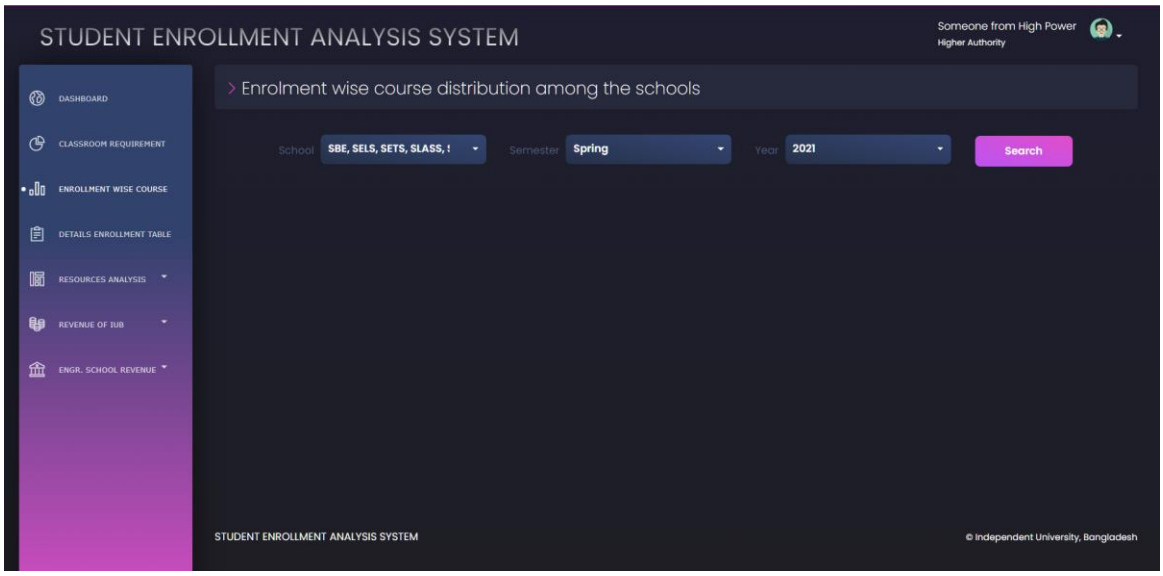

```python
def uploadfunc(request):
    if request.method == 'POST':
        form = uploadfileform(request.POST or None, request.FILES or None)
        if form.is_valid():
            form.save()
    else:
        form = uploadfileform()
    return render(request, 'home.html', {'form': form})

def runpopulationscript(request):
    path = str(BASE_DIR)+"//Scripts//PopulationScript.py"
    filepath = str(BASE_DIR)+"//media//Resources"
    filenameslist = next(walk(filepath), (None, None, []))[2]
    listToStr = ' '.join([str(elem) for elem in filenameslist])
    run([sys.executable, path, listToStr, filepath], shell=True)

    return render(request, 'home.html', {'data1': "DB updated"})
```




```
def view_classroom_requirement_course_offer(request):

    if request.method == 'POST':
                                        #semester        =
semesterlist[int(request.POST.get('sem'))]
        #year = yearlist[int(request.POST.get('year'))]
        lbl = ['1-10', '11-20', '21-30', '31-35',
               '36-40', '41-50', '51-55', '56-65']

        semester = request.POST.get('sem')
        year = request.POST.get('year')
                                          sections       =
classroom_requirement_course_offer(semester, year)
        selectedsem = semester+' '+year
        table = []
        class6 = []
        #print(sections)
        sumsections = sum(sections)
        for i in sections:
            class6.append("{:.2f}".format(i/12))
        #print(class6)
```

```
        sumcls6 = "{:.2f}".format(sum([float(i) for i
in class6]))
        class7 = []
        for i in sections:
            class7.append("{:.2f}".format(i/14))
        #print(class7)
        sumcls7 = "{:.2f}".format(sum([float(i) for i
in class7]))

        for i in range(len(lbl)):
            col1 = lbl[i]
            col2 = sections[i]
            col3 = class6[i]
            col4 = class7[i]
            table.append([col1, col2, col3, col4])
         table.append(['Total', sumsections, sumcls6,
sumcls7])
        print(class6)

        str1 = semester+" "+str(year)
        return render(request, 'classsize.html', {
            'semesters': semesterlist,
            'years': yearlist,
            'class6': class6,
            'class7': class7,
            'sections': sections,
            'seme': str1,
            'table': table,
            'selectedsem': selectedsem,

            'search': 0,
            'segment': 'cls_req',
        })
```

```
    else:
        return render(request, 'classsize.html', {
            'semesters': semesterlist,
            'years': yearlist,
            'segment': 'cls_req',
            'search': 1,
        })
```

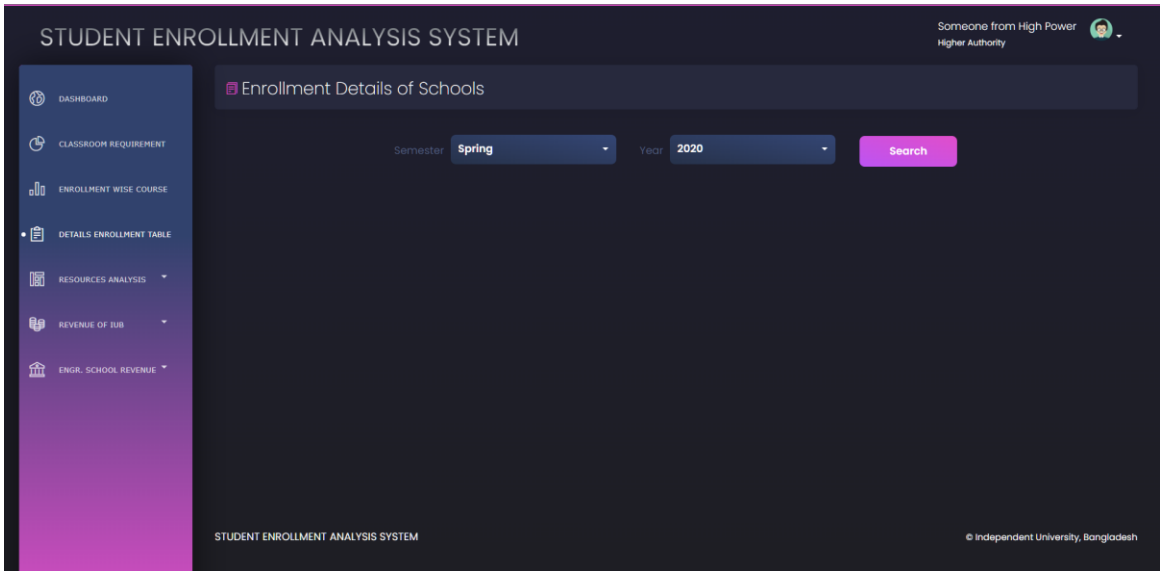



```
def view_enrolment_course_school(request):
    if request.method == 'POST':
        l = ['1-10', '11-20', '21-30', '31-35', '36-
40',
            '41-50', '51-55', '56-60', '60+']
        school = request.POST.getlist('scl')
        semester = request.POST.get('sem')
        year = request.POST.get('year')
        selectedsem = semester+' '+year
        rowsize = len(school)+2
        enrollment = []
        labels = []
```

```
        for i in school:

            enrollment.append(enrollment_wise_course_s
chool(i, semester, year))

        for j in l:
            for i in school:
                labels.append(i+':'+j)

       #enrollment = [item for t in enrollment for item
in t for item in item]

        list0 = []
        list1 = []
        list2 = []
        list3 = []
        list4 = []
        list5 = []
        list6 = []
        list7 = []
        list8 = []
        list9 = []
        list10 = []
        table = []
        for i in enrollment:
            e = [item for t in i for item in t]
            list0.append(e)
            list9.append(e[0])
            list1.append(e[1])
            list2.append(e[2])
            list3.append(e[3])
            list4.append(e[4])
```

```
            list5.append(e[5])
            list6.append(e[6])
            list7.append(e[7])
            list8.append(e[8])

        list10.append(sum(list9))
        list10.append(sum(list1))
        list10.append(sum(list2))
        list10.append(sum(list3))
        list10.append(sum(list4))
        list10.append(sum(list5))
        list10.append(sum(list6))
        list10.append(sum(list7))
        list10.append(sum(list8))
        temprow = []
        for i in range(len(l)):
            temprow = [row[i] for row in list0]
            temprow.append(list10[i])
            temprow.insert(0, l[i])
            table.append(temprow)  # make row wise

        list0.insert(0, list10)
               #   table.append([row[0]   for   row   in
list0].insert(0,'Total'))#add total to end of the list
        print(table)
        return render(request, 'enrollmentwise.html', {
            'schools': schoolList,
            'selectedschool': school,
            'rowsize': rowsize,
            'selectedsem': selectedsem,
            'semesters': semesterlist,
            'years': yearlist,
            'table': table,
```

```
            'enrollment': list0,
            'labels': labels,
            'search': 0,
            'segment': 'enroll',
        })

    else:
        return render(request, 'enrollmentwise.html', {
            'schools': schoolList,
            'semesters': semesterlist,
            'years': yearlist,
            'segment': 'enroll',
            'search': 1,
        })
```



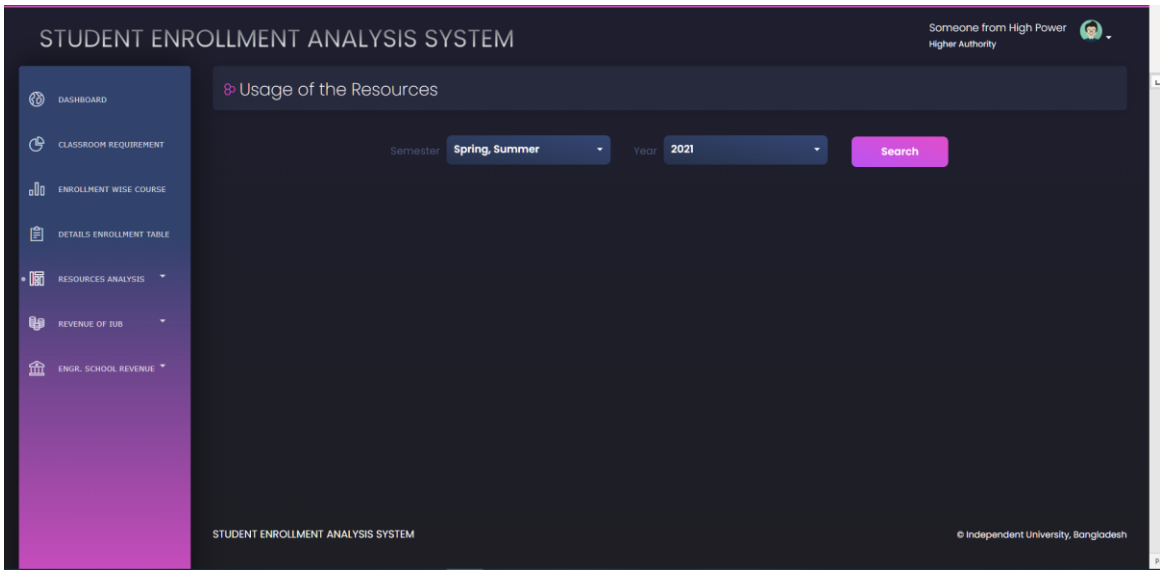

```
def view_enrolment_details(request):
    if request.method == 'POST':
        sem = request.POST.get('sem')
        year = request.POST.get('year')

        sbe = []
        sels = []
```

```
        sets = []
        slass = []
        spph = []

        sbe = details_enrollment('SBE', sem, year)
        sels = details_enrollment('SELS', sem, year)
        sets = details_enrollment('SETS', sem, year)
        slass = details_enrollment('SLASS', sem, year)
        spph = details_enrollment('SPPH', sem, year)

        selectedsem = sem+' '+year
        # print(sbe)
        # print(sels)

        m = []
        m.append(max(sbe, key=itemgetter(0))[0])
        m.append(max(sels, key=itemgetter(0))[0])
        m.append(max(sets, key=itemgetter(0))[0])
        m.append(max(slass, key=itemgetter(0))[0])
        m.append(max(spph, key=itemgetter(0))[0])

        maxenroll = max(m)+1

         sbe = [element for tupl in sbe for element in
tupl]
        sels = [element for tupl in sels for element in
tupl]
        sets = [element for tupl in sets for element in
tupl]
         slass = [element for tupl in slass for element
in tupl]
        spph = [element for tupl in spph for element in
tupl]
```

```
        # Elements from list1 starting from 1 iterating
by 2
        listOddsbe = sbe[1::2]
        # Elements from list1 starting from 0 iterating
by 2
        listEvensbe = sbe[::2]
        listOddsels = sels[1::2]
        listEvensels = sels[::2]
        listOddsets = sets[1::2]
        listEvensets = sets[::2]
        listOddslass = slass[1::2]
        listEvenslass = slass[::2]
        listOddspph = spph[1::2]
        listEvenspph = spph[::2]

        for i in range(maxenroll):
            if i not in listEvensbe:
                listEvensbe.insert(i, i)
                listOddsbe.insert(i, 0)
        sumsbe = sum(listOddsbe)
        for i in range(maxenroll):
            if i not in listEvensels:
                listOddsels.insert(i, 0)
        sumsels = sum(listOddsels)
        for i in range(maxenroll):
            if i not in listEvensets:
                listOddsets.insert(i, 0)
        sumsets = sum(listOddsets)
        for i in range(maxenroll):
            if i not in listEvenslass:
                listOddslass.insert(i, 0)
        sumslass = sum(listOddslass)
```

```
        for i in range(maxenroll):
            if i not in listEvenspph:
                listOddspph.insert(i, 0)
        sumspph = sum(listOddspph)
        tabledata = []
        totalsum = []
        tabledata2 = []
        tabledata3 = []
        tabledata4 = []
        tabledata5 = []

        for i in range(maxenroll):
            a0 = listEvensbe[i]
            a1 = listOddsbe[i]
            a2 = listOddsels[i]
            a3 = listOddsets[i]
            a4 = listOddslass[i]
            a5 = listOddspph[i]
            a6 = a1+a2+a3+a4+a5
            totalsum.append(a6)

             tabledata.append([a0, a1, a2, a3, a4, a5,
a6])
          tabledata.append(['Total', sumsbe, sumsels,
sumsets,
                                    sumslass, sumspph,
sum(totalsum)])
        # print(sum(tabledata))

       return render(request, 'enrolmentdetails.html',
{
            'semesters': semesterlist,
            'years': yearlist,
```

```
            'selectedsem': selectedsem,
            'selectedyear': year,
            'range': listEvensbe,
            'tabledata': tabledata,



            'search': 0,
            'segment': 'details',
        })

    else:
        return render(request, 'enrolmentdetails.html',
{
            'semesters': semesterlist,
            'years': yearlist,
            'search': 1,
            'segment': 'details',
        })
```

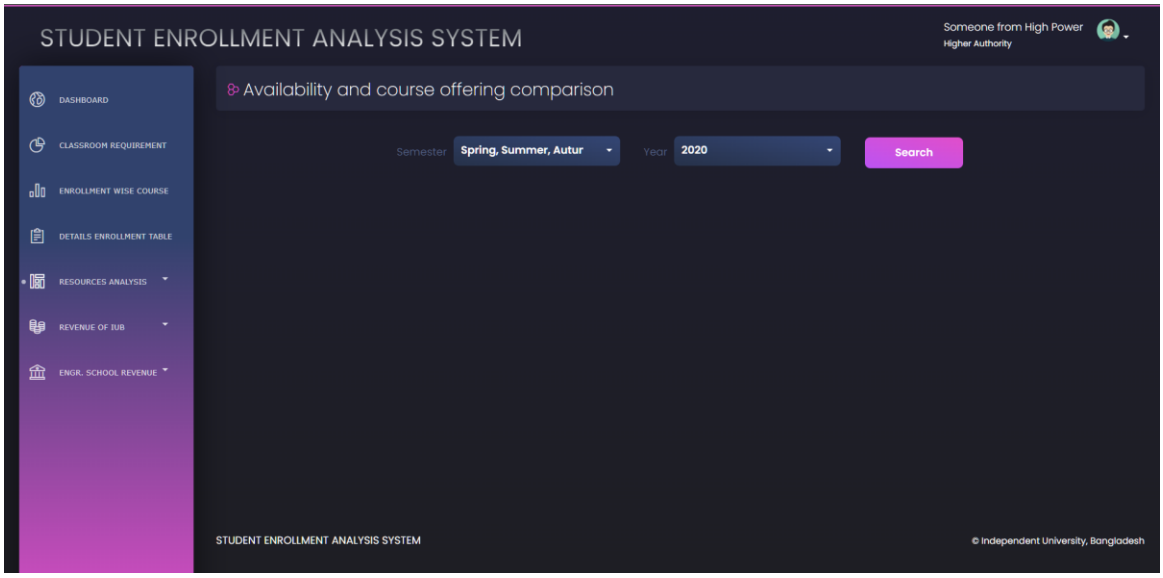



```
def view_usage_resource(request):
    if request.method == 'POST':
        sem = request.POST.getlist('sem')
```

```
        year = request.POST.get('year')
        table = []
        total = []
        table2=[]
        t2r1 = ['Average of ROOM_CAPACITY']
        t2r2 = ['Average of ENROLLED']
        t2r3 = ['Average of Unused Space']
        t2r4 = ['Unused Percent %']
        selectedsem='| '
        for i in sem:
            sbe = resources_usage('SBE', i, year)
            sels = resources_usage('SELS', i, year)
            sets = resources_usage('SETS', i, year)
            slass = resources_usage('SLASS', i, year)
            spph = resources_usage('SPPH', i, year)
            selectedsem+= i+' '+year+' | '

            # touple of touple to list
             sbe = [float("%.2f" % (element)) for tupl
in sbe for element in tupl]
                                                    #
https://stackoverflow.com/questions/3204245/how-do-i-
convert-a-tuple-of-tuples-to-a-one-dimensional-list-
using-list-comprehe
                                                    #
https://stackoverflow.com/questions/455612/limiting-
floats-to-two-decimal-points
            sels = [float("%.2f" % (element)) for tupl
in sels for element in tupl]
            sets = [float("%.2f" % (element)) for tupl
in sets for element in tupl]
            slass = [float("%.2f" % (element))for tupl
in slass for element in tupl]
```

```
            spph = [float("%.2f" % (element)) for tupl
in spph for element in tupl]



                                          totala      =
sbe[0]+sels[0]+sets[0]+slass[0]+spph[0]
                                          totalb      =
sbe[1]+sels[1]+sets[1]+slass[1]+spph[1]
                                          totalc      =
sbe[2]+sels[2]+sets[2]+slass[2]+spph[2]
                                          totald      =
sbe[3]+sels[3]+sets[3]+slass[3]+spph[3]
                                          totale      =
sbe[4]+sels[4]+sets[4]+slass[4]+spph[4]

            t2r1.append("%.2f" % (totalb/5))
            t2r2.append("%.2f" % (totalc/5))
            t2r3.append("%.2f" % (totald/5))
            t2r4.append("%.2f" % (totale/5))

                  total.append([i,  totala,  "%.2f"  %
(totalb/5),  "%.2f" %(totalc/5),  "%.2f" % (totald/5),
"%.2f" % (totale/5)])



            sbe.insert(0,'SBE')
            sels.insert(0, 'SELS')
            sets.insert(0,'SETS')
            slass.insert(0,'SLASS')
            spph.insert(0,'SPPH')

            total.append(sbe)
```

```
            total.append(sels)
            total.append(sets)
            total.append(slass)
            total.append(spph)
        print(total)
        table+=total
        table2=[t2r1]+[t2r2]+[t2r3]+[t2r4]
        total=[]
        print(table2)
        return render(request, 'resourceusage.html', {
            'semesters': semesterlist,
            'years': yearlist,
            'table': table,
            'table2':table2,
            'selectedsem': selectedsem,
            'selectedyear':year,
            'semfort2':sem,
            'rowsize':len(sem)+1,
            'search': 0,
            'segment': 'usage',

        })

    else:
        return render(request, 'resourceusage.html', {
            'semesters': semesterlist,
            'years': yearlist,
            'search': 1,
            'segment': 'usage',
        })
```




```
def view_availabilityvscourse_offer(request):
    if request.method == 'POST':

        semester = request.POST.getlist('sem')
        year = request.POST.get('year')

        roomsize = roomsizelist()
         roomsize = [element for tupl in roomsize for
element in tupl]
        class6 = []
        for i in semester:
                                        sections      =
classroom_requirement_course_offer(i, year)
            class6temp=[]
            for j in sections:
                class6temp.append(float("{:.2f}".forma
t(j/12)))
            class6.append(class6temp)
        # print(class6)
        # sumcls6 = "{:.2f}".format(sum([float(i) for i
in class6]))
        listOddsize = roomsize[::2]
```

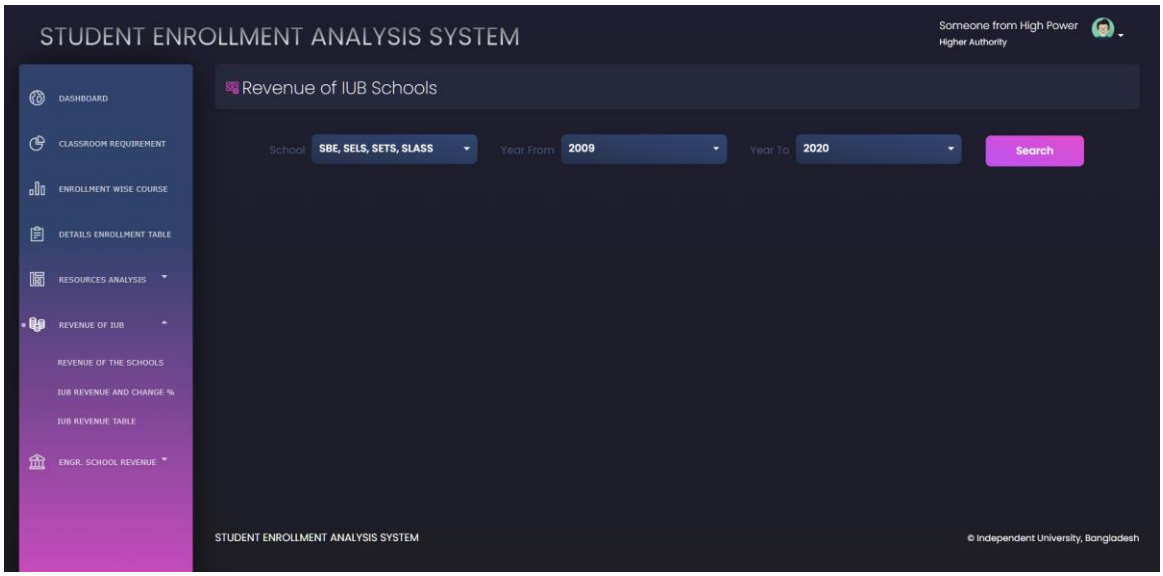
```
        listEvensize = roomsize[1::2] #ulta ache. odd -
>even, even-> odd e ase.
        class6offered=[]
        for j in range(0,len(semester)):
            a=[]
            b=[]
            c=[]
            d=[]
            e=[]
            f=[]
            g=[]
            c6 = []
            r=class6[j][0]
            s=class6[j][1]
            t=class6[j][2]
            u=class6[j][3]
            v=class6[j][4]
            w=class6[j][5]
            x=class6[j][6]
            y=class6[j][7]
            k=0
            l=0
            for i in range(0,len(listOddsize)):

                if listOddsize[i] in range(1, 21):
                    # print(listOddsize[i])
                    # print(r+s)
                    a.append(r+s)

                if listOddsize[i] in range(21, 31):
                    # print(listOddsize[i])
                    # print(t)
                    b.append(t)
```

```
            if listOddsize[i] in range(31, 36):
                # print(listOddsize[i])
                # print(u)
                c.append(u)

            if listOddsize[i] in range(36, 41):
                # print(listOddsize[i])
                # print(v)
                d.append(v)

            if listOddsize[i] in range(41, 51):
                # print(listOddsize[i])
                # print(w)
                e.append(w)

            if listOddsize[i] in range(51, 56):
                # print(listOddsize[i])
                # print(x)
                f.append(x)


            if listOddsize[i] in range(56, 65):
                # print(listOddsize[i])
                # print(y)
                g.append(y)

        c6=a+b+c+d+e+f+g
        if len(f)==0:
            listOddsize.append(54)
            listEvensize.append(0)
            c6.append(x)
        if len(g)==0:
```

```
                listOddsize.append(64)
                listEvensize.append(0)
                c6.append(y)


            #c6.insert(0,sum(c6))

            class6offered.append(c6)
            c6=[]
            a = []
            b = []
            c = []
            d = []
            e = []
            f = []
            g = []


        totalroom = listEvensize
        #totalroom.insert(0,sum(listEvensize))
        chartdata=[]
        chartlabel = listOddsize
        chartdata += class6offered #+[totalroom]
        print(class6offered)
        print(listEvensize)

        #table shuru->
        temptable=[]
        tablelabel=[]
        rowlentemp = len(listOddsize)
        tablelabel+=listOddsize
        tablelabel.append('Total')
        resource = listEvensize
```

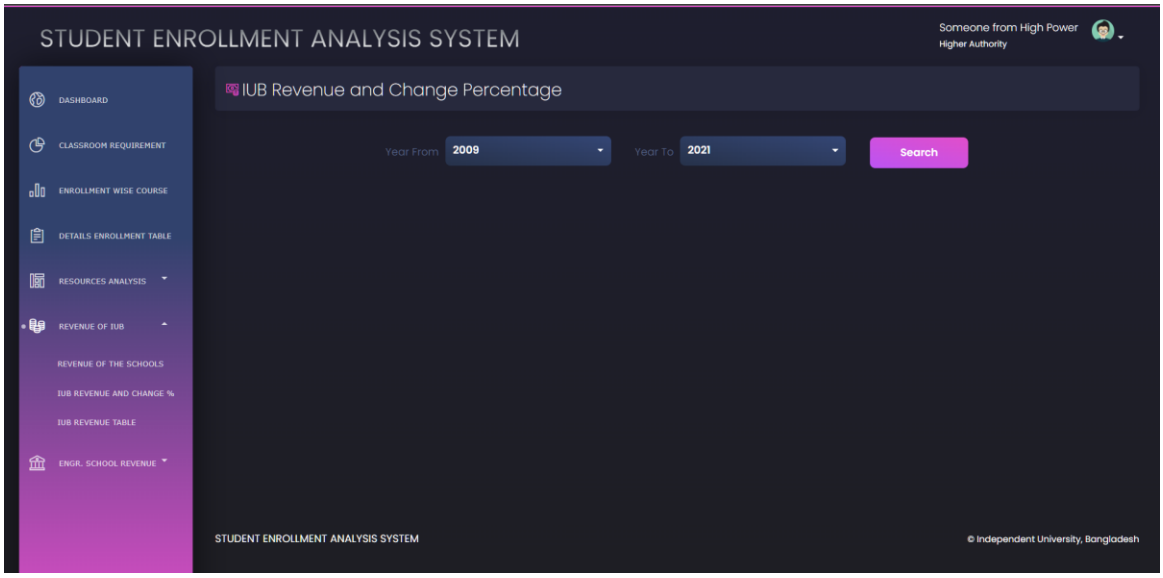
```
        difference = []
        for i in range(0,len(semester)):
            for j in range(len(listEvensize)):
                difference.append(float("{:.2f}".forma
t(listEvensize[j]-class6offered[i][j])))

        difference = [difference[i:i+rowlentemp]
                      for i in range(0, len(difference),
rowlentemp)]
        temptable.append(resource)
        for i in range(0, len(semester)):
            temptable.append(class6offered[i])
            temptable.append(difference[i])

                        table=np.transpose(temptable)#
https://numpy.org/doc/stable/reference/generated/numpy
.transpose.html
                              total=      np.sum(table,
axis=0)#https://www.delftstack.com/howto/numpy/sum-of-
columns-matrix-numpy/

           total = ["%.2f" % member for member in
total]#https://stackoverflow.com/questions/2762058/for
mat-all-elements-of-a-list
                table   =   np.append(table,   [total],
axis=0)#https://www.codegrepper.com/code-
examples/python/add+row+to+numpy+2+dimension+arrayhttp
s://www.codegrepper.com/code-
examples/python/add+row+to+numpy+2+dimension+array

        finaltable=np.c_[tablelabel,table]#https://moo
nbooks.org/Articles/How-to-add-a-new-column-in-a-
table-using-python-and-numpy-/
```

```
                                                #          #
#https://stackoverflow.com/questions/12575421/convert-
a-1d-array-to-a-2d-array-in-numpy
        # # differencematrix=np.reshape(difference,(-1,
len(semester)))

        #print(finaltable)

                            return      render(request,
'availabilityvscourse.html', {
            'schools': schoolList,
            'semesters': semesterlist,
            'years': yearlist,
            'chartdata':chartdata,
            'chartlabel':chartlabel,
            'selectedsem': semester,
            'noofcols': 2+len(semester)*2,
            'resource': listEvensize,
            'table': finaltable,
            'search': 0,
            'segment': 'usage',
        })

    else:
                            return      render(request,
'availabilityvscourse.html', {
            'schools': schoolList,
            'semesters': semesterlist,
            'years': yearlist,
            'search': 1,
            'segment': 'usage',
        })
```




```
def view_revenue_of_iub(request):
    if request.method == 'POST':
        school = request.POST.getlist('scl')
        yearf = request.POST.get('year1')
        yeart = request.POST.get('year2')
        print(school, yearf, yeart)
        revenue = []
        for i in school:
            revenue.append(iub_revenue(yearf, yeart,
i))
        # print(revenue)
        a = abs(int(yearf)-int(yeart))+1
        # print(type(a))

        list1 = []
        list2 = []
        list3 = []
        total = []

        for j in revenue:
            for i in j:
```

```
                list1.append(str(i[0])+i[1])
                list2.append(int(i[2]))
        list1 = list(dict.fromkeys(list1))
          list2 = [list2[i:i+a*3] for i in range(0,
len(list2), a*3)]

        # print(list1)
        # print(list2)
        return render(request, 'revenueofiub.html', {
            'schools': schoolList,
            'yearfrom': yearlist,
            'yearto': yearlist,
            'selectedschool': school,
            'revenuesemyear': list1,
            'revenue': list2,

            'search': 0,
            'segment': 'rev',
        })

    else:
        return render(request, 'revenueofiub.html', {
            'schools': schoolList,
            'yearfrom': yearlist,
            'yearto': yearlist,
            'search': 1,
            'segment': 'rev',
        })
```

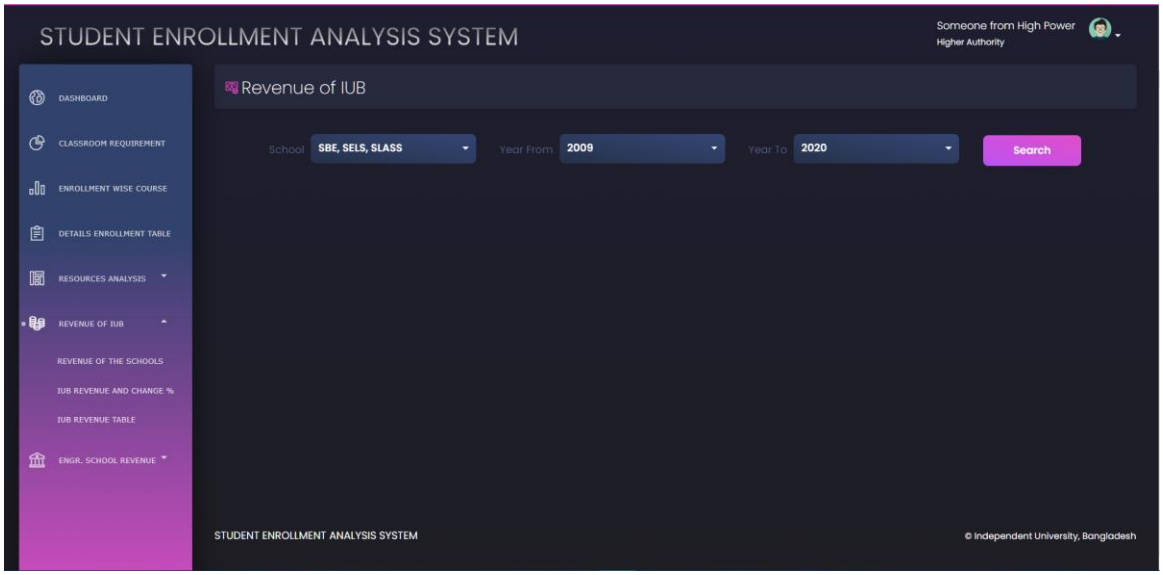



```python
def view_rev_change(request):
    if request.method == 'POST':
        yearf = request.POST.get('year1')
        yeart = request.POST.get('year2')
        revenue = []

        revenue.append(iub_revenue_total(yearf, yeart))
        # print(revenue)
        y = abs(int(yearf)-int(yeart))+1
        # print(type(a))

        list1 = []
        list2 = []
        list3 = []
        list4 = []
        list5 = []
        list6 = []
        list7 = []
        list8 = []
        list9 = []
        list10 = []
        total = []
```

```
        for j in revenue:
            for i in j:
                list1.append(str(i[0])+i[1])
                list2.append(int(i[2]))
        list1 = list(dict.fromkeys(list1))
        total = list2
           list2 = [list2[i:i+3] for i in range(0,
len(list2), 3)]
        print(total)
        for i in list2:
            list3.append(i[0])
            list4.append(i[1])
            list5.append(i[2])
        # print(list3)
        for i in list3:
            # if(list3.index(i)<len(list3)):
            if list3.index(i) != 0:
                a = list3[abs(list3.index(i)-1)]
                b = i
                c = int(((b-a)/b)*100)
            else:
                a = list3[abs(list3.index(i))]
                b = i
                c = int(((b-a)/b)*100)
            list6.append(c)
        for i in list4:
            # if(list3.index(i)<len(list3)):
            if list4.index(i) != 0:
                a = list4[abs(list4.index(i)-1)]
                b = i
                c = int(((b-a)/b)*100)
            else:
```

```
                a = list4[abs(list4.index(i))]
                b = i
                c = int(((b-a)/b)*100)
            list7.append(c)
        for i in list5:
            # if(list3.index(i)<len(list3)):
            if list5.index(i) != 0:
                a = list5[abs(list5.index(i)-1)]
                b = i
                c = int(((b-a)/b)*100)
            else:
                a = list5[abs(list5.index(i))]
                b = i
                c = int(((b-a)/b)*100)
            list8.append(c)
        # for i in list8:
        for j in range(y):
            list9.append(list6[j])
            list9.append(list7[j])
            list9.append(list8[j])

        # list9=[list6]+[list7]+[list8]

        # print(list1)
        print(list2)
        # print(list9)

        return render(request, 'revenuechange.html', {
            'yearfrom': yearlist,
            'yearto': yearlist,
            'revenuesemyear': list1,
            'revenueper': list9,
            'totalrev': total,
```

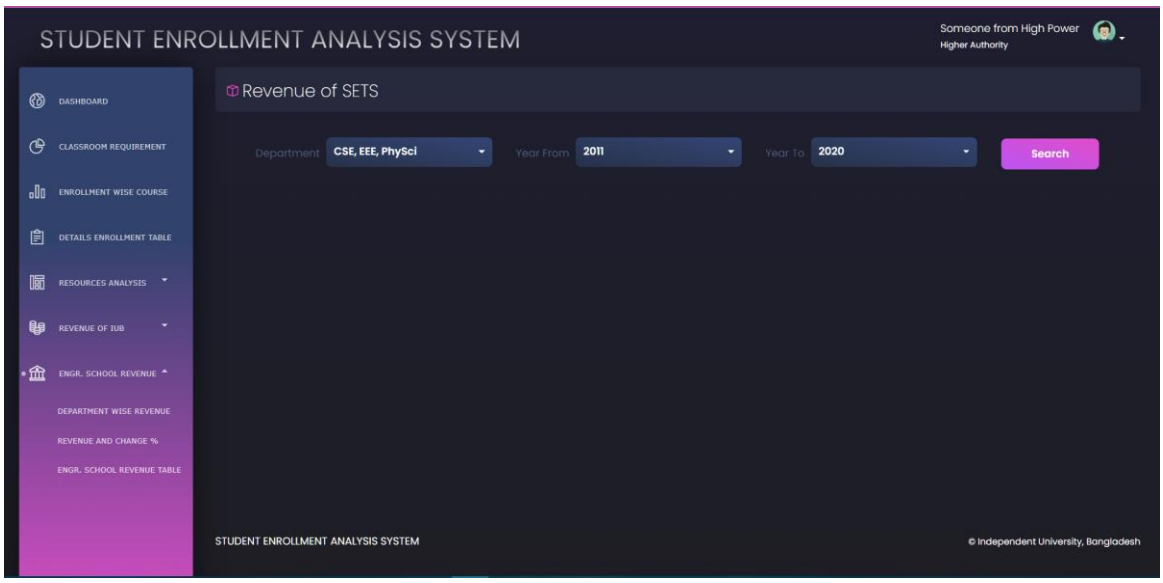
```
            'search': 0,
            'segment': 'rev',
        })

    else:
        return render(request, 'revenuechange.html', {
            'yearfrom': yearlist,
            'yearto': yearlist,
            'search': 1,
            'segment': 'rev',
        })
```




```
def view_revenue_table_of_iub(request):
    if request.method == 'POST':
        school = request.POST.getlist('scl')
        yearf = request.POST.get('year1')
        yeart = request.POST.get('year2')
        print(school, yearf, yeart)
        revenue = []
        rowsize=len(school)
        for i in school:
            revenue.append(iub_revenue(yearf, yeart, i))
        # print(revenue)
        a = abs(int(yearf)-int(yeart))+1
        # print(type(a))

        list1 = []
```

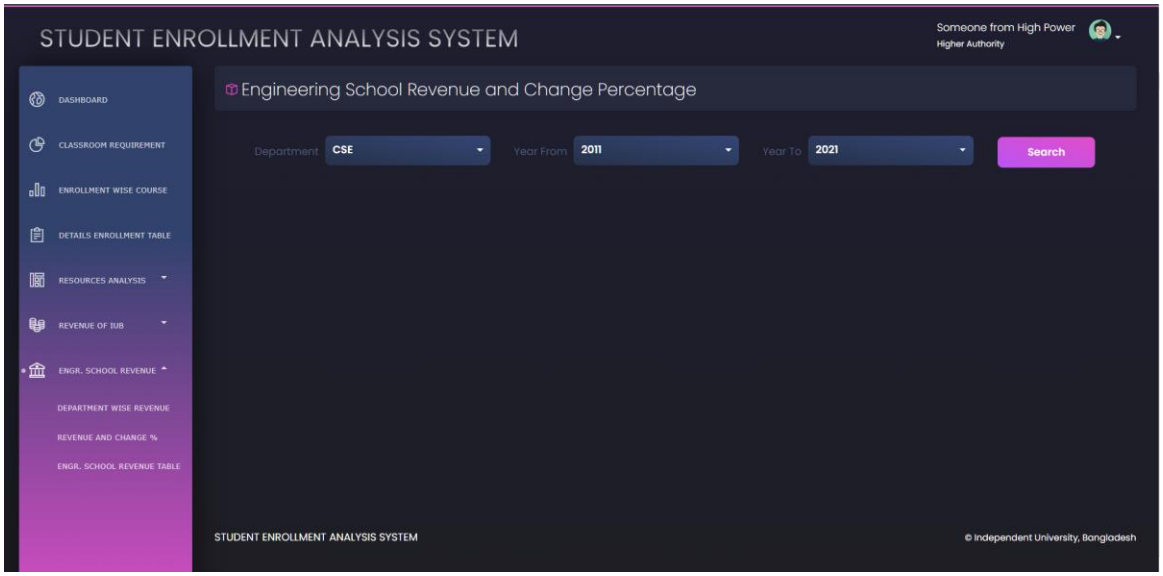
```
        list2 = []
        list3 = []
        total = []

        for j in revenue:
            for i in j:
                list1.append(str(i[0])+i[1])
                list2.append(int(i[2]))
        list1 = list(dict.fromkeys(list1))
        list2 = [list2[i:i+a*3] for i in range(0, len(list2), a*3)]
        #add missing spph to make col same for matrix
        if "SPPH" in school:
            index=school.index("SPPH")
            dif = len(list1) - len(list2[index])
            if dif==1:
                list2[index].insert(1,0)
            if dif==2:
                list2[index].insert(1,0)
                list2[index].insert(4, 0)
            if dif == 3:
                list2[index].insert(1, 0)
                list2[index].insert(4, 0)
                list2[index].insert(7, 0)
        list3=np.transpose(list2)
        total = list3.sum(axis=1).tolist()
        change=[0,0,0]
        for i in range(len(total)-3):
            change.append(int(((total[i+3]-total[i])/total[i+3])*100))
        list2.append(total)
        list2.append(change)
        list2 = np.transpose(list2)
        finaltable = np.c_[list1, list2]
        print(finaltable)
        return render(request, 'revenuetableiub.html', {
            'schools': schoolList,
            'yearfrom': yearlist,
            'yearto': yearlist,
            'selectedschool': school,
            'yearf':yearf,
            'yeart':yeart,
            'table': finaltable,
            'rowsize':rowsize,
            'search': 0,
            'segment': 'rev',
        })

    else:
        return render(request, 'revenuetableiub.html', {
```

```
            'schools': schoolList,
            'yearfrom': yearlist,
            'yearto': yearlist,
            'search': 1,
            'segment': 'rev',
        })
```




```
def view_sets_rev(request):
    if request.method == 'POST':
        dept = request.POST.getlist('dept')
        yearf = request.POST.get('year1')
        yeart = request.POST.get('year2')
        print(dept, yearf, yeart)
        revenue = []
        for i in dept:
            revenue.append(SETS_revenue(yearf, yeart,
i))
        # print(revenue)
        a = abs(int(yearf)-int(yeart))+1
        # print(type(a))

        list1 = []
        list2 = []
        list3 = []
```

```
        total = []

        for j in revenue:
            for i in j:
                list1.append(str(i[0])+i[1])
                list2.append(int(i[2]))
        list1 = list(dict.fromkeys(list1))
          list2 = [list2[i:i+a*3] for i in range(0,
len(list2), a*3)]

        print(list1)
        print(list2)
        return render(request, 'setsrev.html', {
            'dept': SETSdeptList,
            'yearfrom': yearlist,
            'yearto': yearlist,
            'selecteddept': dept,
            'revenuesemyear': list1,
            'revenue': list2,

            'search': 0,
            'segment': 'sets',
        })

    else:
        return render(request, 'setsrev.html', {
            'dept': SETSdeptList,
            'yearfrom': yearlist,
            'yearto': yearlist,
            'search': 1,
            'segment': 'sets',
        })
```

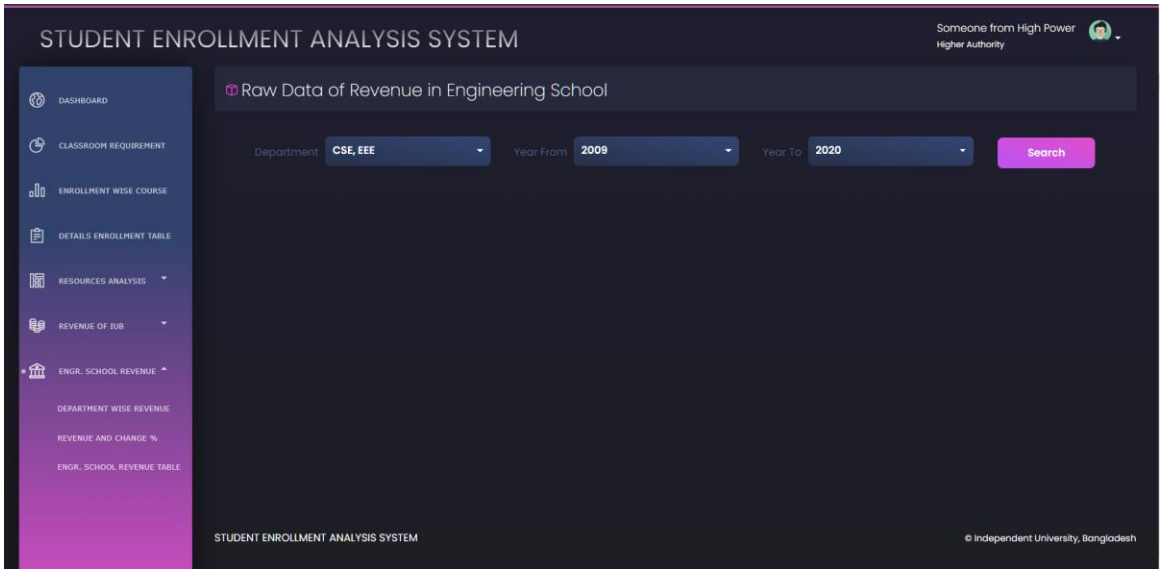



```
def view_deptwise_rev_per(request):
    if request.method == 'POST':
        dept = request.POST.get('dept')
        yearf = request.POST.get('year1')
        yeart = request.POST.get('year2')
        # print(dept, yearf, yeart)
        revenue = []
            revenue.append(SETS_revenue(yearf,  yeart,
dept))
        # print(revenue)
        y = abs(int(yearf)-int(yeart))+1
        # print(type(a))

        list1 = []
        list2 = []
        list3 = []
        list4 = []
        list5 = []
        list6 = []
        list7 = []
        list8 = []
        list9 = []
```

```
        list10 = []
        total = []

        for j in revenue:
            for i in j:
                list1.append(str(i[0])+i[1])
                list2.append(int(i[2]))
        list1 = list(dict.fromkeys(list1))
        total = list2
           list2 = [list2[i:i+3] for i in range(0,
len(list2), 3)]

        print(list1)
        for i in list2:
            list3.append(i[0])
            list4.append(i[1])
            list5.append(i[2])

        for i in list3:
            # if(list3.index(i)<len(list3)):
            if list3.index(i) != 0:
                a = list3[abs(list3.index(i)-1)]
                b = i
                c = int(((b-a)/b)*100)
            else:
                a = list3[abs(list3.index(i))]
                b = i
                c = int(((b-a)/b)*100)
            list6.append(c)
        for i in list4:
            # if(list3.index(i)<len(list3)):
            if list4.index(i) != 0:
                a = list4[abs(list4.index(i)-1)]
```

```
                b = i
                c = int(((b-a)/b)*100)
            else:
                a = list4[abs(list4.index(i))]
                b = i
                c = int(((b-a)/b)*100)
            list7.append(c)
        for i in list5:
            # if(list3.index(i)<len(list3)):
            if list5.index(i) != 0:
                a = list5[abs(list5.index(i)-1)]
                b = i
                c = int(((b-a)/b)*100)
            else:
                a = list5[abs(list5.index(i))]
                b = i
                c = int(((b-a)/b)*100)
            list8.append(c)
        # for i in list8:
        for j in range(y):
            list9.append(list6[j])
            list9.append(list7[j])
            list9.append(list8[j])

        print(list9)
        return render(request, 'setsdeptper.html', {
            'dept': SETSdeptList,
            'yearfrom': yearlist,
            'yearto': yearlist,
            'selecteddept': dept,
            'revenuesemyear': list1,
            'revenue': list9,
            'totalrev': total,
```

```
            'search': 0,
            'segment': 'sets',
        })

    else:
        return render(request, 'setsdeptper.html', {
            'dept': SETSdeptList,
            'yearfrom': yearlist,
            'yearto': yearlist,
            'search': 1,
            'segment': 'sets',
        })
```

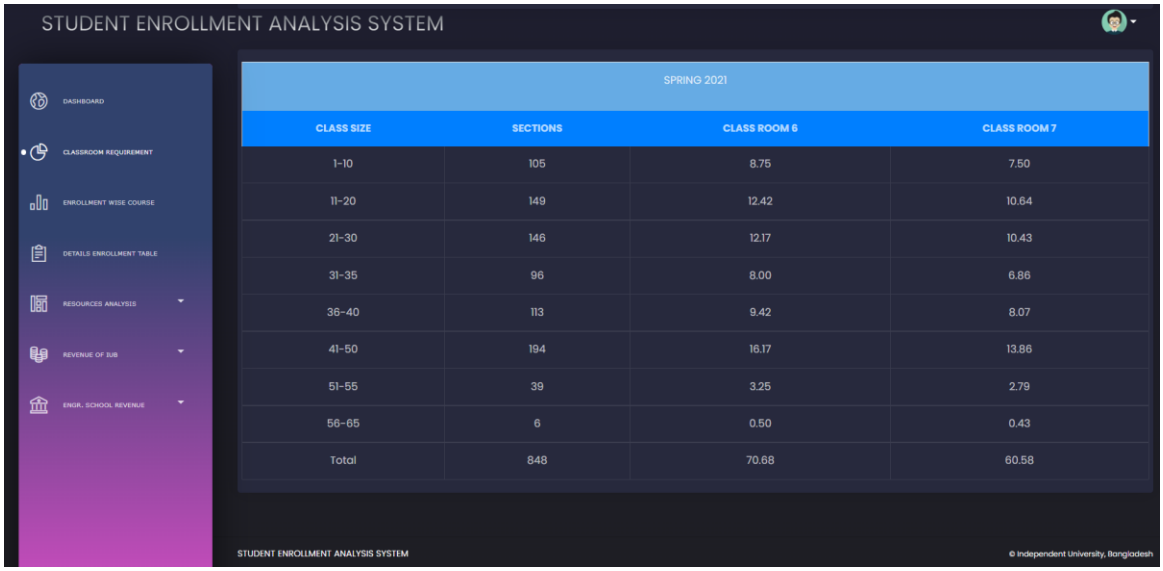


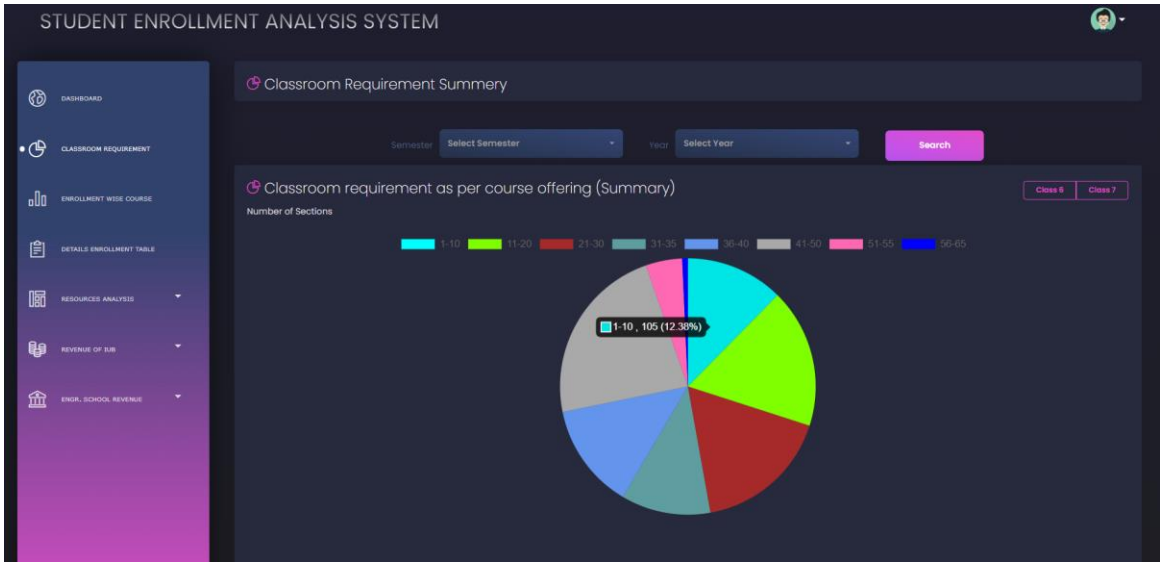

```
def view_engr_school_rev(request):
    if request.method == 'POST':
        dept = request.POST.getlist('dept')
        yearf = request.POST.get('year1')
        yeart = request.POST.get('year2')
        print(dept, yearf, yeart)
        revenue = []
        for i in dept:
            revenue.append(SETS_revenue(yearf,yeart,i))

        a = abs(int(yearf)-int(yeart))+1
        # print(type(a))
```

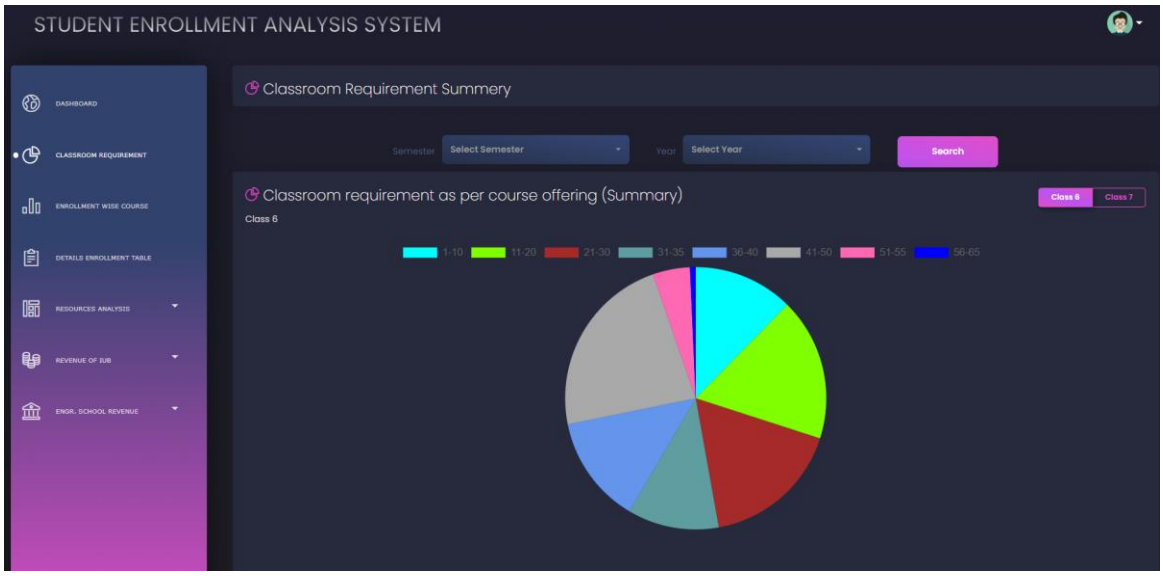
```
        list1 = []
        list2 = []
        list3 = []
        total = []

        for j in revenue:
            for i in j:
                list1.append(str(i[0])+i[1])
                list2.append(int(i[2]))
        list1 = list(dict.fromkeys(list1))
        list2 = [list2[i:i+a*3] for i in range(0, len(list2), a*3)]

        list3 = np.transpose(list2)
        total = list3.sum(axis=1).tolist()
        change = []
        for i in range(len(list2)):
            change1 = [0,0,0]
            for j in range(len(list2[i])-3):
                change1.append(int(((list2[i][j+3]-
list2[i][j])/list2[i][j+3])*100))
            change.append(change1)
        changet=[0,0,0]
        for i in range(len(total)-3):
            changet.append(int(((total[i+3]-
total[i])/total[i+3])*100))
        change.append(changet)

        finaltemp1=np.transpose(change)

        list2.append(total)
        finaltemp2=np.transpose(list2)

        finaltemp3=np.concatenate((finaltemp2,finaltemp1),axis=1)


        finaltable = np.c_[list1, finaltemp3]
        tablehead=[]

        for i in range(len(dept)):
            tablehead.append(dept[i])
        tablehead.append("SETS")
        for i in range(len(dept)):
            tablehead.append(dept[i]+"%")
        tablehead.append("SETS%")


        print(tablehead)
        return render(request, 'setsrevtable.html', {
```

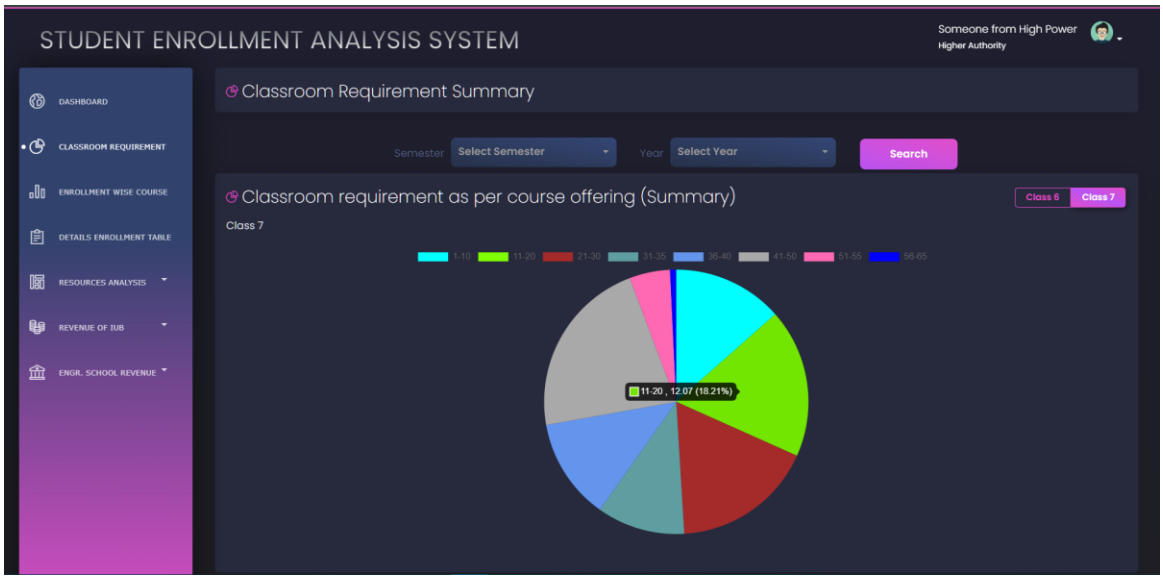
```
            'dept': SETSdeptList,
            'yearfrom': yearlist,
            'yearto': yearlist,
            'yearf': yearf,
            'yeart': yeart,
            'table': finaltable,
            'selecteddept':dept,
            'tablehead': tablehead,
            'search': 0,
            'segment': 'sets',
        })

    else:
        return render(request, 'setsrevtable.html', {
            'dept': SETSdeptList,
            'yearfrom': yearlist,
            'yearto': yearlist,
            'search': 1,
            'segment': 'sets',
        })
```

B. OUTPUT FORMS:




STUDENT ENROLLMENT ANALYSIS SYSTEM
DASHBOARD
CLASSROOM REQUIREMENT
ENROLLMENT WISE COURSE
DETAILS ENROLLMENT TABLE
RESOURCES ANALYSIS
REVENUE OF IUB
ENGR. SCHOOL REVENUE
Classroom Requirement Summery
Semester
Select Semester
Year
Select Year
Search
Classroom requirement as per course offering (Summary)
Number of Sections
Class 6
Class 7
1-10
11-20
21-30
31-35
36-40
41-50
51-55
56-65
1-10 , 105 (12.38%)

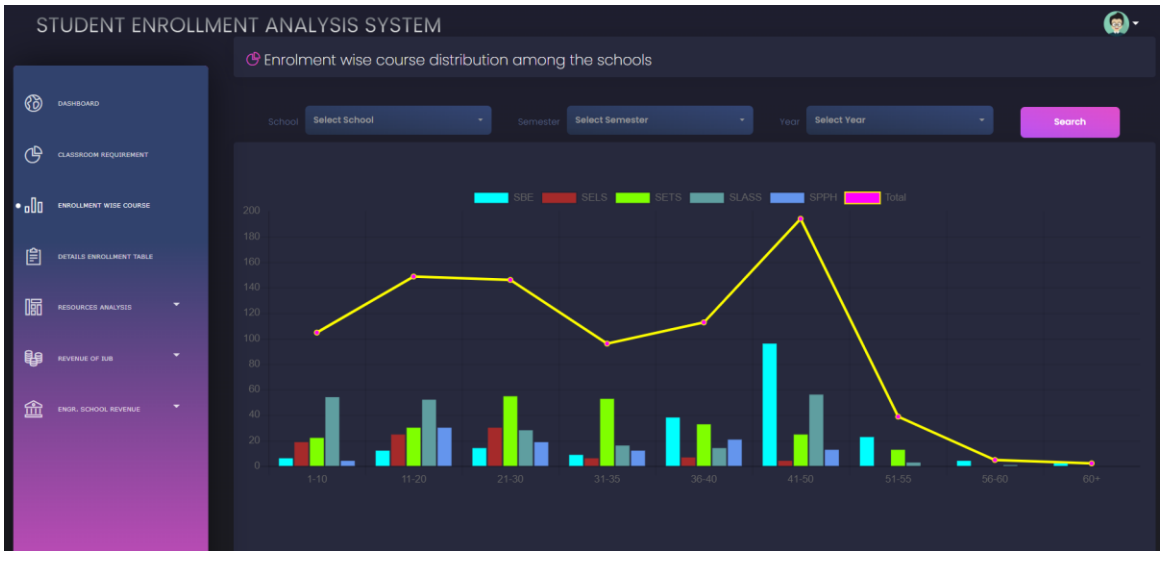

STUDENT ENROLLMENT ANALYSIS SYSTEM
DASHBOARD
CLASSROOM REQUIREMENT
ENROLLMENT WISE COURSE
DETAILS ENROLLMENT TABLE
RESOURCES ANALYSIS
REVENUE OF IUB
ENGR. SCHOOL REVENUE
Classroom Requirement Summery
Semester
Select Semester
Year
Select Year
Search
Classroom requirement as per course offering (Summary)
Class 6
Class 6
Class 7
1-10
11-20
21-30
31-35
36-40
41-50
51-55
56-65


STUDENT ENROLLMENT ANALYSIS SYSTEM
Someone from High Power
Higher Authority
DASHBOARD
CLASSROOM REQUIREMENT
ENROLLMENT WISE COURSE
DETAILS ENROLLMENT TABLE
RESOURCES ANALYSIS
REVENUE OF IUB
ENGR. SCHOOL REVENUE
Classroom Requirement Summary
Semester
Select Semester
Year
Select Year
Search
Classroom requirement as per course offering (Summary)
Class 7
Class 6
Class 7
1-10
11-20
21-30
31-35
36-40
41-50
51-55
56-65
11-20 , 12.07 (18.21%)

```
def classroom_requirement_course_offer(Sem, Year):
    with connection.cursor() as cursor:

        cursor.execute('''
        SELECT COUNT(*) AS Sections
        FROM joinedtable
        WHERE SectionEnrolled BETWEEN 1 AND 10
        AND semester = "{}"
        AND YEAR ={}

        UNION ALL

        SELECT COUNT(*)
        FROM joinedtable
        WHERE SectionEnrolled BETWEEN 11 AND 20
        AND semester = "{}"
        AND YEAR ={}

        UNION ALL

        SELECT COUNT(*)
        FROM joinedtable
        WHERE SectionEnrolled BETWEEN 21 AND 30
        AND semester = "{}"
        AND YEAR ={}

        UNION ALL

        SELECT COUNT(*)
        FROM joinedtable
        WHERE SectionEnrolled BETWEEN 31 AND 35
        AND semester = "{}"
        AND YEAR ={}
```

```
        UNION ALL

        SELECT COUNT(*)
        FROM joinedtable
        WHERE SectionEnrolled BETWEEN 36 AND 40
        AND semester = "{}"
        AND YEAR ={}

        UNION ALL

        SELECT COUNT(*)
        FROM joinedtable
        WHERE SectionEnrolled BETWEEN 41 AND 50
        AND semester = "{}"
        AND YEAR ={}

        UNION ALL

        SELECT COUNT(*)
        FROM joinedtable
        WHERE SectionEnrolled BETWEEN 51 AND 55
        AND semester = "{}"
        AND YEAR ={}

        UNION ALL

        SELECT COUNT(*)
        FROM joinedtable
        WHERE SectionEnrolled BETWEEN 56 AND 65
        AND semester = "{}"
        AND YEAR ={}
```

```
        '''.format(Sem, Year, Sem, Year, Sem, Year, Sem,
Year, Sem, Year, Sem, Year, Sem, Year, Sem, Year))
        sections=[]
        col = cursor.fetchall()
        for i in col:
            for j in i:
                sections.append(j)
    return (sections)
```

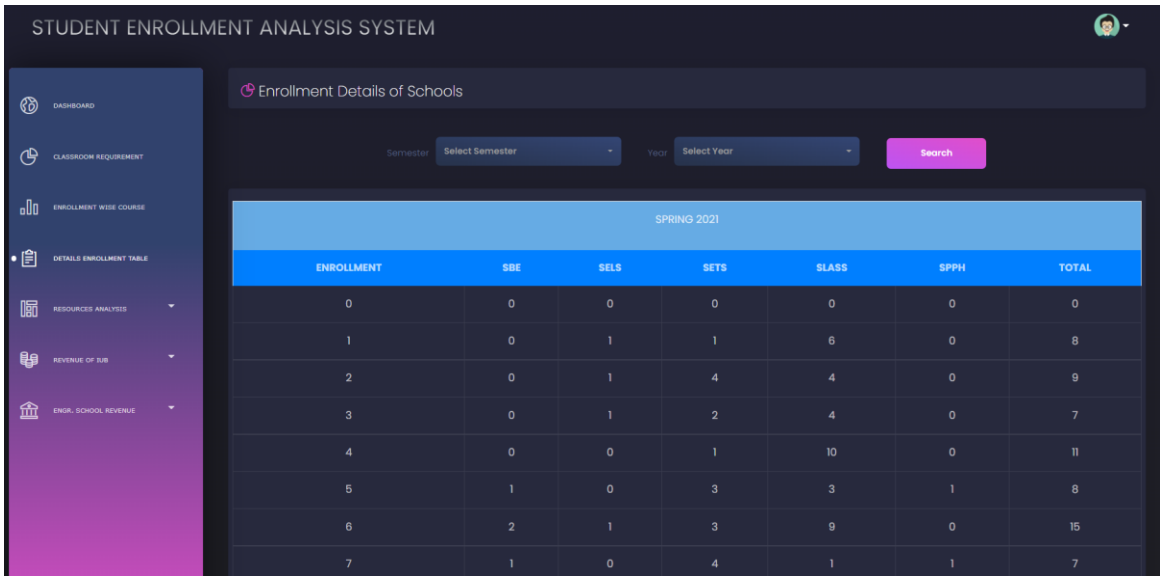



```
def enrollment_wise_course_school(School, Sem, Year):
    with connection.cursor() as cursor:
        cursor.execute('''
        SELECT COUNT(*)
```

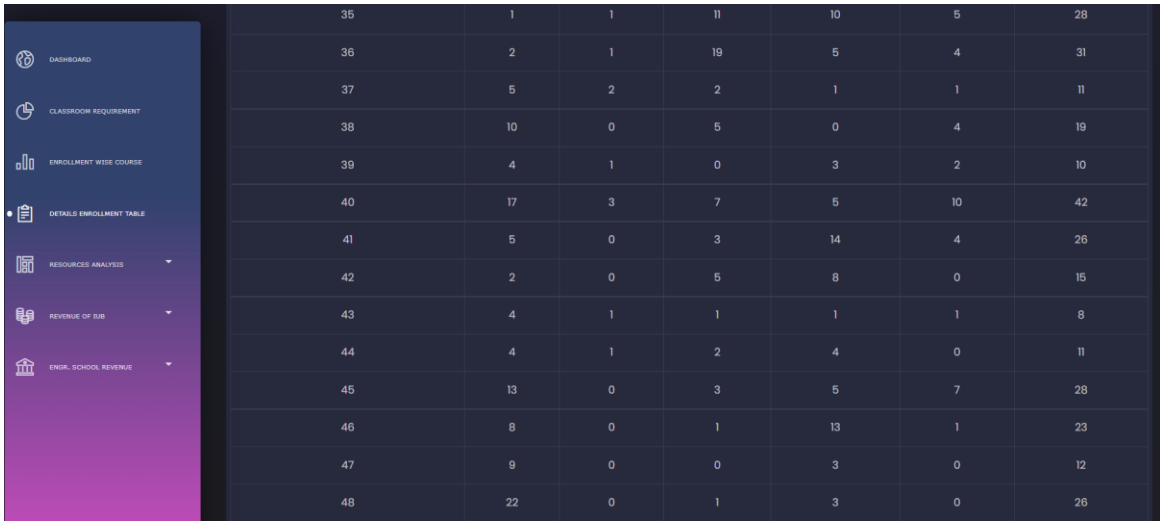
```
        FROM joinedtable
       WHERE SchoolTitle_id="{}" AND Semester="{}" AND
Year={} AND SectionEnrolled BETWEEN 1 AND 10

        UNION ALL

        SELECT COUNT(*)
        FROM joinedtable
       WHERE SchoolTitle_id="{}" AND Semester="{}" AND
Year={} AND SectionEnrolled BETWEEN 11 AND 20

        UNION ALL

        SELECT COUNT(*)
        FROM joinedtable
       WHERE SchoolTitle_id="{}" AND Semester="{}" AND
Year={} AND SectionEnrolled BETWEEN 21 AND 30

        UNION ALL

        SELECT COUNT(*)
        FROM joinedtable
       WHERE SchoolTitle_id="{}" AND Semester="{}" AND
Year={} AND SectionEnrolled BETWEEN 31 AND 35

        UNION ALL

        SELECT COUNT(*)
        FROM joinedtable
       WHERE SchoolTitle_id="{}" AND Semester="{}" AND
Year={} AND SectionEnrolled BETWEEN 36 AND 40

        UNION ALL
```

```
        SELECT COUNT(*)
        FROM joinedtable
        WHERE SchoolTitle_id="{}" AND Semester="{}" AND
Year={} AND SectionEnrolled BETWEEN 41 AND 50

        UNION ALL

        SELECT COUNT(*)
        FROM joinedtable
        WHERE SchoolTitle_id="{}" AND Semester="{}" AND
Year={} AND SectionEnrolled BETWEEN 51 AND 55

        UNION ALL

        SELECT COUNT(*)
        FROM joinedtable
        WHERE SchoolTitle_id="{}" AND Semester="{}" AND
Year={} AND SectionEnrolled BETWEEN 56 AND 60

        UNION ALL

        SELECT COUNT(*)
        FROM joinedtable
        WHERE SchoolTitle_id="{}" AND Semester="{}" AND
Year={} AND SectionEnrolled > 60

        '''.format(School, Sem, Year, School, Sem, Year,
School, Sem, Year, School, Sem, Year, School, Sem, Year,
School, Sem, Year, School, Sem, Year, School, Sem, Year,
School, Sem, Year))

        col = cursor.fetchall()
```

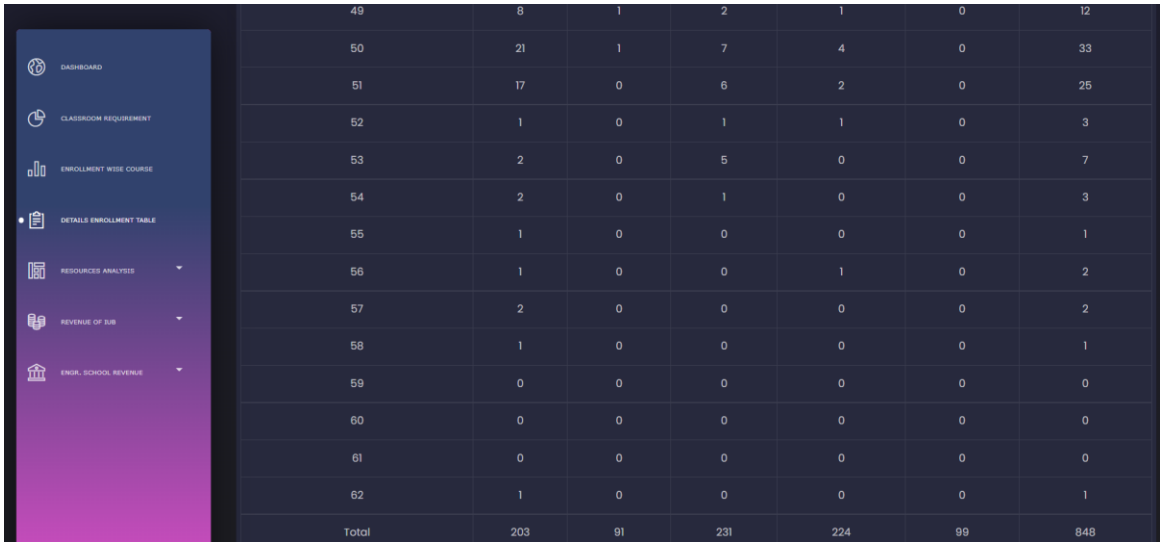
```
return col
```



ENROLLMENT	SBE	SELS	SETS	SLASS	SPPH	TOTAL
0	0	0	0	0	0	0
1	0	1	1	6	0	8
2	0	1	4	4	0	9
3	0	1	2	4	0	7
4	0	0	1	10	0	11
5	1	0	3	3	1	8
6	2	1	3	9	0	15
7	1	0	4	1	1	7
8	1	3	1	4	1	10
9	1	5	2	6	0	14
10	0	7	1	7	1	16
11	3	1	5	2	2	13
12	2	4	4	5	3	18
13	0	5	3	6	1	15
14	1	3	2	7	2	15
15	0	2	2	6	2	12
16	0	0	5	4	3	12
17	1	3	3	3	4	14
18	2	2	4	5	2	15
19	2	2	1	3	5	13
20	1	3	1	11	6	22
21	2	4	3	3	4	16
22	0	1	6	2	1	10
23	1	1	5	2	1	10
24	0	0	1	1	1	3
25	3	4	7	5	2	21
26	3	2	5	0	2	12
27	4	4	2	2	2	14
28	0	6	3	1	2	12
29	0	2	6	5	3	16
30	1	6	17	7	1	32
31	1	2	13	1	2	19
32	3	3	9	0	2	17
33	1	0	10	3	1	15
34	3	0	10	2	2	17



35	1	1	11	10	5	28
36	2	1	19	5	4	31
37	5	2	2	1	1	11
38	10	0	5	0	4	19
39	4	1	0	3	2	10
40	17	3	7	5	10	42
41	5	0	3	14	4	26
42	2	0	5	8	0	15
43	4	1	1	1	1	8
44	4	1	2	4	0	11
45	13	0	3	5	7	28
46	8	0	1	13	1	23
47	9	0	0	3	0	12
48	22	0	1	3	0	26



49	8	1	2	1	0	12
50	21	1	7	4	0	33
51	17	0	6	2	0	25
52	1	0	1	1	0	3
53	2	0	5	0	0	7
54	2	0	1	0	0	3
55	1	0	0	0	0	1
56	1	0	0	1	0	2
57	2	0	0	0	0	2
58	1	0	0	0	0	1
59	0	0	0	0	0	0
60	0	0	0	0	0	0
61	0	0	0	0	0	0
62	1	0	0	0	0	1
Total	203	91	231	224	99	848

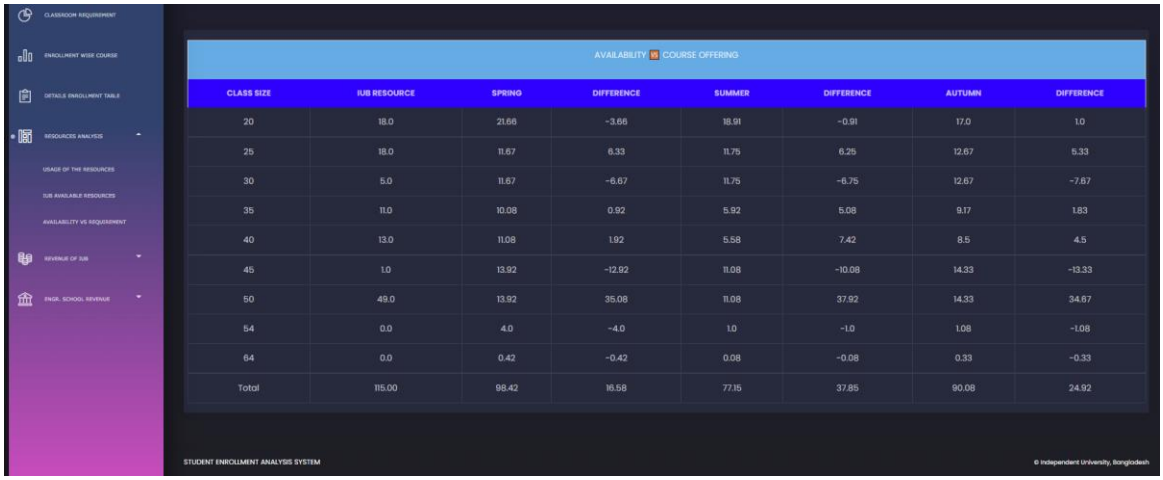
```
def details_enrollment(School, Sem, Year):
    with connection.cursor() as cursor:
        cursor.execute('''
        SELECT SectionEnrolled, COUNT(*) AS SBE
        FROM joinedtable
        WHERE SchoolTitle_id="{}" AND Semester="{}" AND
Year = {} AND SectionEnrolled BETWEEN 1 AND 100
        GROUP BY SectionEnrolled
        HAVING SectionEnrolled > 0
        ORDER BY SectionEnrolled ASC;
        '''.format(School, Sem, Year))

        col = cursor.fetchall()
```

```
return col
```

	SUM	AVG ENROLL	AVG ROOM	DIFFERENCE	UNUSED %
Spring	26733.0	26.22	38.86	12.65	33.45
SBE	8236.0	38.67	46.31	7.65	16.51
SELS	2354.0	20.12	37.09	16.97	45.76
SETS	8383.0	27.58	39.8	12.23	30.72
SLASS	6006.0	26.46	35.64	9.18	25.76
SPPH	1754.0	18.27	35.47	17.2	48.49

	SPRING 2020
Average of ROOM_CAPACITY	26.22
Average of ENROLLED	38.86
Average of Unused Space	12.65
Unused Percent %	33.45

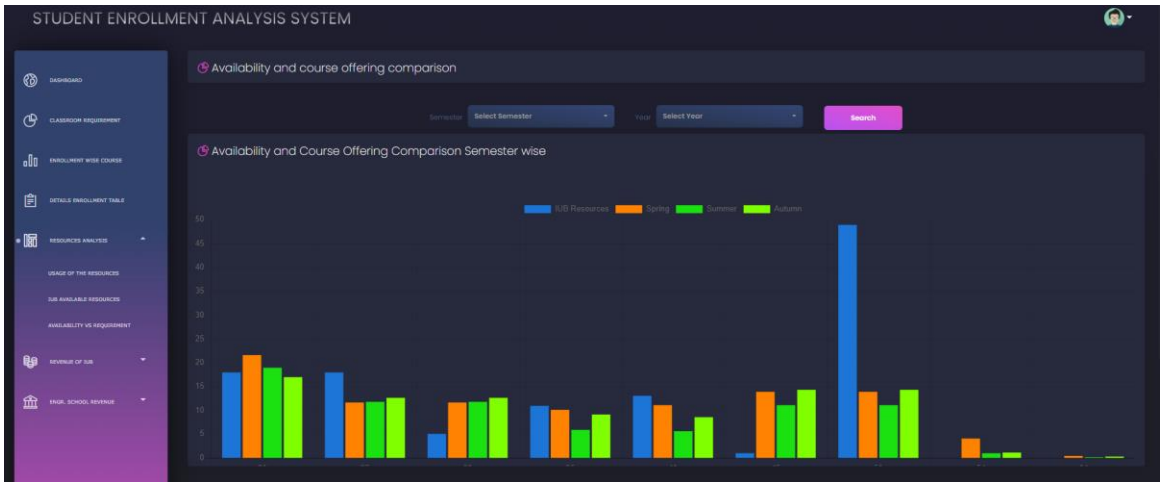
IUB AVAILABLE RESOURCES		
CLASS SIZE	IUB RESOURCE	CAPACITY
20	18	360
25	18	450
30	5	150
35	11	385
40	13	520
45	1	45
50	49	2450
Total	115	4360
Total Capacity with 6 slot 2 days		52320
Total Capacity with 7 slot 2 days		61040
Considering 3.5 average course load (6 slot)		14948
Considering 3.5 average course load (7 slot)		14948
free % for 6 slots capacity		71%
free % for 7 slots capacity		76%



AVAILABILITY VS COURSE OFFERING

CLASS SIZE	IUB RESOURCE	SPRING	DIFFERENCE	SUMMER	DIFFERENCE	AUTUMN	DIFFERENCE
20	18.0	21.66	-3.66	18.91	-0.91	17.0	1.0
25	18.0	11.67	6.33	11.75	6.25	12.67	5.33
30	5.0	11.67	-6.67	11.75	-6.75	12.67	-7.67
35	11.0	10.08	0.92	5.92	5.08	9.17	1.83
40	13.0	11.08	1.92	5.58	7.42	8.5	4.5
45	1.0	13.92	-12.92	11.08	-10.08	14.33	-13.33
50	49.0	13.92	35.08	11.08	37.92	14.33	34.67
54	0.0	4.0	-4.0	1.0	-1.0	1.08	-1.08
64	0.0	0.42	-0.42	0.08	-0.08	0.33	-0.33
Total	115.00	98.42	16.58	77.15	37.85	90.08	24.92

STUDENT ENROLLMENT ANALYSIS SYSTEM




```
def resources_usage(School, Sem, Year):
    with connection.cursor() as cursor:
        cursor.execute('''
        SELECT SUM(sectionEnrolled) AS Sum
        FROM joinedtable
        WHERE SchoolTitle_id="{}" AND Semester="{}" AND
Year={}

        UNION ALL

        SELECT AVG(sectionEnrolled)
        FROM joinedtable
        WHERE SchoolTitle_id="{}" AND Semester="{}" AND
Year={}
```

```
        UNION ALL

        SELECT AVG(roomCapacity)
        FROM joinedtable
        JOIN seasapp_room_t
        ON RoomID_id= RoomID
        WHERE SchoolTitle_id="{}" AND Semester="{}" AND
Year={}

        UNION ALL

         SELECT AVG(roomCapacity)-Avg(sectionEnrolled)
AS difference
        FROM joinedtable
        JOIN seasapp_room_t
        ON RoomID_id= RoomID
        WHERE SchoolTitle_id="{}" AND Semester="{}" AND
Year={}

        UNION ALL

                    SELECT    ((AVG(roomCapacity)-
Avg(sectionEnrolled))/AVG(RoomCapacity))*100        AS
percentage
        FROM joinedtable
        JOIN seasapp_room_t
        ON RoomID_id= RoomID
        WHERE SchoolTitle_id="{}" AND Semester="{}" AND
Year={}
```

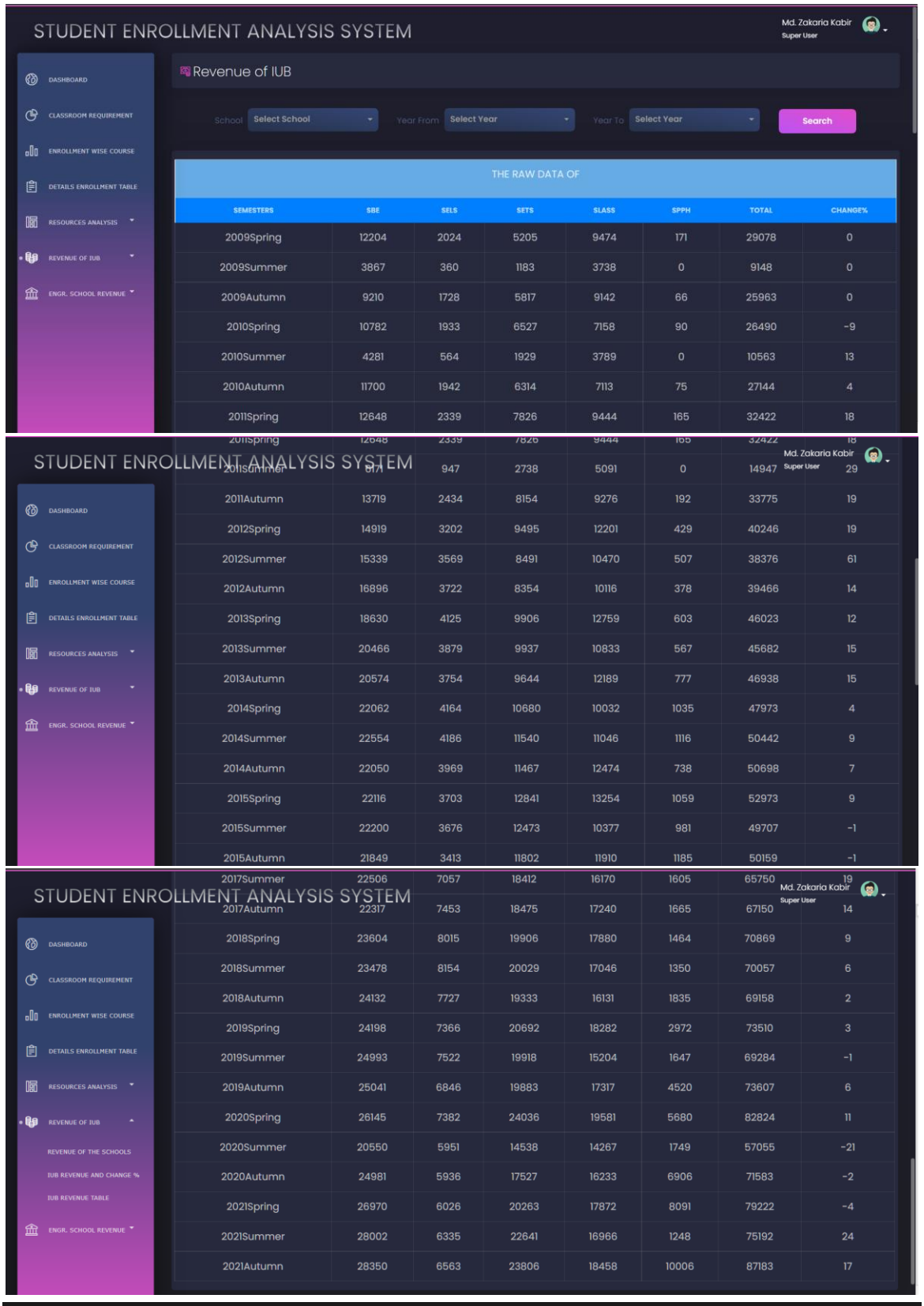
```
        '''.format(School, Sem, Year, School, Sem, Year, School, Sem, Year, School, Sem, Year, School, Sem, Year))

        row = cursor.fetchall()
    return row
```

```
def roomsizelist():
    with connection.cursor() as cursor:
        cursor.execute('''
        SELECT roomcapacity,COUNT(roomcapacity)
        FROM seasapp_room_t
        GROUP BY roomcapacity
        ORDER BY roomcapacity;
        ''')

        col = cursor.fetchall()

    return col
```


```
def iub_revenue(Yearfrom, Yearto, School):
    with connection.cursor() as cursor:
        cursor.execute('''
        SELECT Year,Semester,SUM(groupbycredit.sum)
        FROM
            (
                    SELECT  COUNT(*),credithour,  SUM(
SectionEnrolled),  credithour*SUM(  SectionEnrolled)  AS
sum, Semester,year
                FROM joinedtable
                                    WHERE   Semester   IN
("Spring","AUTUMN","SUMMER") AND Year BETWEEN {} AND {}
AND SchoolTitle_id = "{}"
                GROUP BY Year,Semester,Credithour
            ) AS groupbycredit
        GROUP BY Year,Semester
                        ORDER     BY     Year,     FIELD
(Semester,"Spring","Summer","Autumn");
        '''.format(Yearfrom, Yearto, School))

        col = cursor.fetchall()

    return col
```


```
def iub_revenue_total(Yearfrom, Yearto):
    with connection.cursor() as cursor:
        cursor.execute('''
        SELECT Year,Semester,SUM(groupbycredit.sum)
        FROM
            (
                    SELECT  COUNT(*),credithour,  SUM(
SectionEnrolled),  credithour*SUM(  SectionEnrolled)  AS
sum, Semester,year
                FROM joinedtable
                                    WHERE   Semester   IN
("Spring","AUTUMN","SUMMER") AND Year BETWEEN {} AND {}
AND SchoolTitle_id IN ('SBE', 'SELS', 'SETS', 'SLASS',
'SPPH')
                GROUP BY Year,Semester,Credithour
            ) AS groupbycredit
        GROUP BY Year,Semester
                        ORDER     BY     Year,     FIELD
(Semester,"Spring","Summer","Autumn");
        '''.format(Yearfrom, Yearto))

        row = cursor.fetchall()
```

```
    return row
```

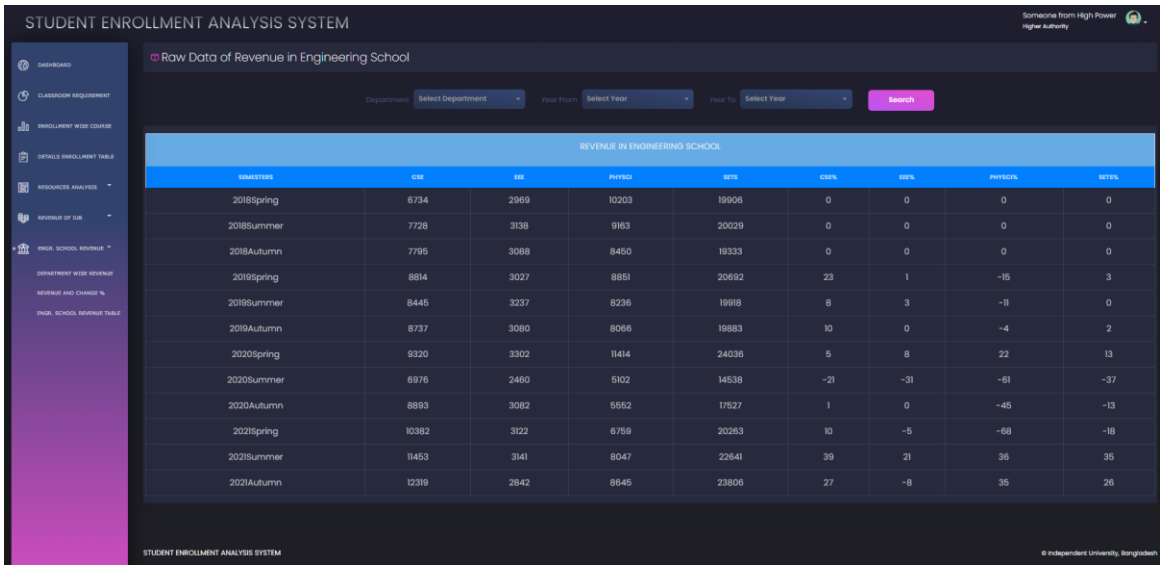



```
def iub_revenue(Yearfrom, Yearto, School):
    with connection.cursor() as cursor:
        cursor.execute('''
```

```
        SELECT Year,Semester,SUM(groupbycredit.sum)
        FROM
            (
                SELECT COUNT(*),credithour, SUM( SectionEnrolled),
credithour*SUM( SectionEnrolled) AS sum, Semester,year
                FROM joinedtable
                WHERE Semester IN ("Spring","AUTUMN","SUMMER") AND Year
BETWEEN {} AND {} AND SchoolTitle_id = "{}"
                GROUP BY Year,Semester,Credithour
            ) AS groupbycredit
        GROUP BY Year,Semester
        ORDER BY Year, FIELD (Semester,"Spring","Summer","Autumn");
        '''.format(Yearfrom, Yearto, School))

        col = cursor.fetchall()

    return col
```


STUDENT ENROLLMENT ANALYSIS SYSTEM
Revenue of SETS
Department
Select Department
Year From
Select Year
Year To
Select Year
Search
Revenue of SETS
Department wise revenue in SETS
Revenue of SETS
CSE
EEE
PhySci
DASHBOARD
CLASSROOM REQUIREMENT
ENROLLMENT WISE COURSE
DETAILS ENROLLMENT TABLE
RESOURCES ANALYSIS
REVENUE OF SUB
ENGR. SCHOOL REVENUE
STUDENT ENROLLMENT ANALYSIS SYSTEM
© Independent University, Bangladesh


STUDENT ENROLLMENT ANALYSIS SYSTEM
Engineering School Revenue and Change Percentage
Department
Select Department
Year From
Select Year
Year To
Select Year
Search
Change%
Total
DASHBOARD
CLASSROOM REQUIREMENT
ENROLLMENT WISE COURSE
DETAILS ENROLLMENT TABLE
RESOURCES ANALYSIS
REVENUE OF SUB
ENGR. SCHOOL REVENUE
DEPARTMENT WISE REVENUE
REVENUE AND CHANGE %
STUDENT ENROLLMENT ANALYSIS SYSTEM
© Independent University, Bangladesh


STUDENT ENROLLMENT ANALYSIS SYSTEM
Someone from High Power
Higher Authority
Raw Data of Revenue in Engineering School
Department
Select Department
Year From
Select Year
Year To
Select Year
Search
REVENUE IN ENGINEERING SCHOOL
SEMESTERS
CSE
EEE
PHYSCI
SETS
CSE%
EEE%
PHYSCI%
SETS%
20185pring 6734 2969 10203 19906 0 0 0 0
20185ummer 7728 3138 9163 20029 0 0 0 0
2018Autumn 7795 3088 8450 19333 0 0 0 0
20195pring 8814 3027 8851 20692 23 1 -15 3
20195ummer 8445 3237 8236 19918 8 3 -11 0
2019Autumn 8737 3080 8066 19883 10 0 -4 2
20205pring 9320 3302 11414 24036 5 8 22 13
20205ummer 6976 2460 5102 14538 -21 -31 -61 -37
2020Autumn 8893 3082 5552 17527 1 0 -45 -13
20215pring 10382 3122 6759 20263 10 -5 -68 -18
20215ummer 11453 3141 8047 22641 39 21 36 35
2021Autumn 12319 2842 8645 23806 27 -8 35 26
DASHBOARD
CLASSROOM REQUIREMENT
ENROLLMENT WISE COURSE
DETAILS ENROLLMENT TABLE
RESOURCES ANALYSIS
REVENUE OF SUB
ENGR. SCHOOL REVENUE
DEPARTMENT WISE REVENUE
REVENUE AND CHANGE %
ENGR. SCHOOL REVENUE TABLE
STUDENT ENROLLMENT ANALYSIS SYSTEM
© Independent University, Bangladesh

```
def SETS_revenue(Yearfrom, Yearto, Dept):
    with connection.cursor() as cursor:
        cursor.execute('''
        SELECT Year,Semester,SUM(groupbycredit.sum)
        FROM
            (
                    SELECT  COUNT(*),credithour,  SUM(
SectionEnrolled),  credithour*SUM(  SectionEnrolled)  AS
sum, Semester,year
                FROM joinedtable
                                    WHERE   Semester   IN
("Spring","AUTUMN","SUMMER") AND Year BETWEEN {} AND {}
AND SchoolTitle_id = "SETS" AND DeptID = "{}"
                GROUP BY Year,Semester,Credithour
            ) AS groupbycredit
        GROUP BY Year,Semester
                        ORDER     BY     Year,     FIELD
(Semester,"Spring","Summer","Autumn");
        '''.format(Yearfrom, Yearto, Dept))

        col = cursor.fetchall()

    return col
```

CHAPTER 5 - CONCLUSION:

A. PROBLEM AND SOLUTION:

I. ANALYSIS PHASE

Because there was no discrete data present, the majority of assumptions and queries were made while working on the rich picture and six element analysis of the organization's operations, which were based on project SEAS developers. To better understand the scenario and clear up any confusions, feedback sessions were held with respected faculty members.

II. DESIGNING PHASE

Based upon descriptive research created entities were kept at their Significant levels, which was also introduced in the Relational Schema schematic. The instructor's feedback was also very valid and crucial in this case.

III. IMPLEMENTATION PHASE

All the Software System Requirement's (SSR's) reached successfully!

Front-End Developing tools: HTML, CSS, Bootstrap JavaScript, Chart Js

Back End Developing tools: Python, Django

Database-integration: MySQL

B. ADDITIONAL FEATURE AND FUTURE DEVELOPMENT:

Future Developing Purposes:

- Plans for the project is, to add another feature which can predict classroom requirement size for the upcoming semester, providing better resource utilization.
- Deployment.

REFERENCES

[1] "IUB AT A GLANCE," Department of Computer Science and Engineering, [Online]. Available: http://www.iub.edu.bd/AboutIUB/ataglance.