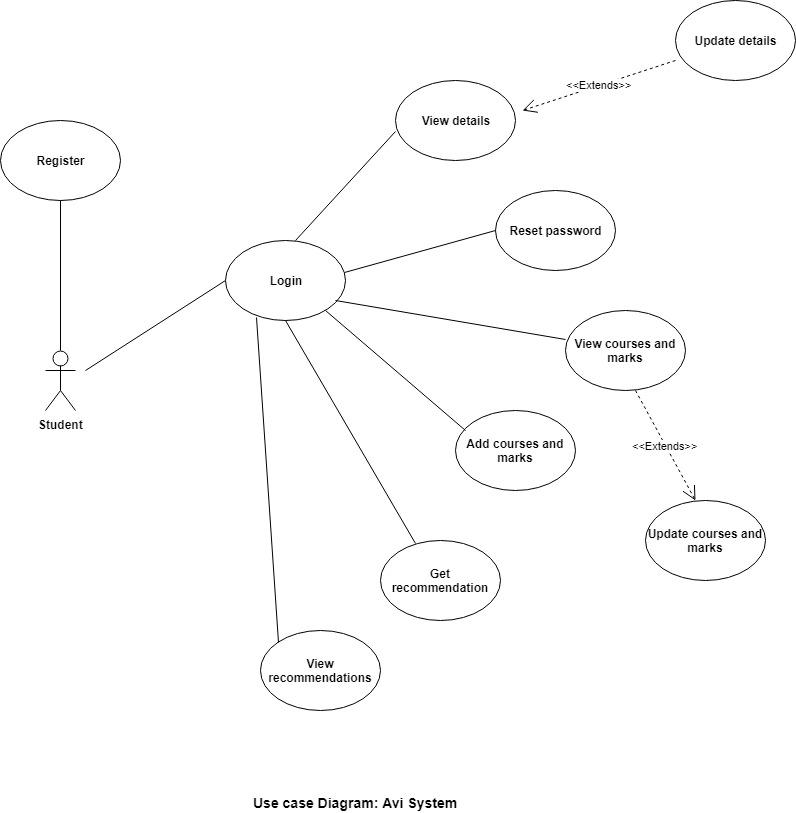
**3.Specific Requirements**

3.1 Functional requirements

     Students class

* Registration - unregistered users of the system will be able to register so that they can access to the system
* Login - after successful registration users a required to login using their created username and password
* A registered user/student is then able to insert the courses they are currently taking or have taken together with their mark to be captured by the system
* Get/view recommendation - a student
* A registered user should also be able to reset password

|  |  |
| --- | --- |
| Use Case | Description |
| Register | Unregistered students have to register |
| login | The student has to login to get access into the system |
| Update details | The student will be able to edit his/her details |
| View details | A student will be able to view their details |
| Add courses and marks | A registered student has to be able to add courses and along with their marks |
| View courses and marks | The student will be able to see the added courses with their marks |
| Update courses and marks | The student will be able to make changes to the added courses with their marks |
| Reset password | A student will be able to reset his/her password if forgotten |
| Get recommendation | Based on the marks entered the students can get a recommendation of courses to take from the system |
| View recommendations | The student will be able to see previously made recommendations |



3.2 External Interfaces.

Possible Inputs:

1. Undergraduate Results.
2. Demographic Information.
3. Student Profile.

Possible outputs:

1. Recommended Course.
2. Expected Grades.

The Student Profile consists of the student’s traits such as hobbies, interests, leadership qualities, problem solving skills, social interests and more. It could furthermore prompt the student to select a rating for certain attributes such as commitment, creativity, attention to detail, ability to work independently, etc.

The undergraduate results should be integers ranging from 0 – 100 and will be directly linked/ related to output i. Computer Science undergraduate courses are higher ranked in the decision-making process, the supplementary courses are considered after and then final decision is made from a calculated weighing of the main and supplementary courses of the results.

3.3 Functions

The system shall check the validity of data.

The system shall handle errors and invalid data.

The system shall check for data overflow, check that the number of simultaneous user access request can be handled to avoid malfunction, bugs and system crashes.

The system shall be logged to monitor admin access and security breaches.

The system shall use data mining techniques to properly group students into clusters accordingly.

The system shall use formulas for calculating expected grades derived from previous similar students using techniques such as statistical correlation.

The system shall then recommend courses which the student may consider choosing.

The system shall return expected grades for those recommended courses.

3.4 Performance Requirements

The system is not expected to handle unreasonably large amounts of data.

The system will expect integer values for academic results and will avoid string inputs using drop down selection and similar implementations.

The system will be able to process student input - clusters comparisons within two minutes from run time

The system will be able to accommodate up to 1000 simultaneous users.

3.5 Logical Database Requirements

Type of variables to be used in database:

1. Integers.
2. Converted Strings.
3. Protected Strings.
4. Floats

Variables will be checked for validity and range correspondence.

Variables such as integers will have a higher frequency retrieval from the database.

Sensitive information already in the database such as identity and student numbers should be locked from retrieval and have limited access – such as admin access only.

3.6 System Software Requirements.

The system should keep checkpoints every 6 hours and restart daily at an allocated time -preferably at night- to avoid malfunctions.

The system should check for errors on a regular interval and use checkpoints to recover from any error if needed.

The system will keep security logs to verify admin access, user access attempts, and privilege information access.

The system should check for data integrity to avoid corrupted and malicious data for critical variables.

The system should utilize cryptographic techniques for database encryption such as RSA encryption or Prime Factorization.

3.7 Design Constraints

-The system will be design using python on the django framework

-The system must comply with the IEEE standards

3.8 Other Requirements

Data retention and other user specific data related storage will not be implemented and user information will be deleted as soon as they deactivate or delete their account off the server.