INTRODUCTION

Scope

1) Add, view, delete student, faculty and accountant information
2) Do pre-advising
3) Administrator can add courses
4) Administrator organizes class schedules with details
5) Administrator manages student and faculty information
6) Faculty can take attendance
7) Faculty can update marks
8) Accountant can update payment information
9) Faculty can send mails to particular sections

Process Model

The process model we have used for our software project is Feature Driven Development.

Feature Driven Development (FDD) is an agile framework that, as its name suggests, organizes software development around making progress on features. It deals with working with different individual features and then integrating them together to form a cohesive project.

FDD was designed to follow a five-step development process:

1. Develop an overall model
2. Build a features list
3. Plan by feature
4. Design by feature
5. Build by feature

Here, we first brainstormed on what different features our project model would need, made a list, and divided our work accordingly. We each developed certain features within a limited time period, around 1-2 weeks for each feature.

We then slowly combined all the features together to form our project that is the University Student Information System (USIS).

Feasibility Analysis

(1) **Economical**: The cost of this project would be much lower than the benefits gained from it. That is because this is a highly useful project for any university, therefore, profits could be earned by selling this project.

(2) **Operational**: Operational checks if the software we are creating will be used or not. This has been satisfied here because USIS is highly used in education facilities around the world.

(3) **Technical**: With the rate technology is progressing, there is hardly any technology that has not yet been developed. Therefore, the technology for the software we are creating is also currently available. E.g. BracU USIS. However, out one is much better because more advanced technology and features have been added.

Requirements Engineering

I. **Inception**: We got the idea from clients who wanted a system where they could keep and manage student, faculty and accountant information.

II. **Elicitation**: We sat down for a meeting with our clients to find out their requirements and constraints for the software.

III. **Elaboration**: We listed down our features and made a use-case.

IV. **Negotiation:** Negotiation about what features they want and what we can provide has been done.

V. **Specification**: Functional and non-functional requirements were found out and a middle ground was reached.

VI. **Validation:** Problems with the code are found and checked here, since requirement error costs are high. Our team leader checked the validity of the system before handing it over to the clients.

VII. **Requirement Management:** During the development process, follow ups were done with the clients to find out their changing requirements which were then implemented accordingly. After delivering the software, we plan on keeping a close eye on the software in case any problem arises.

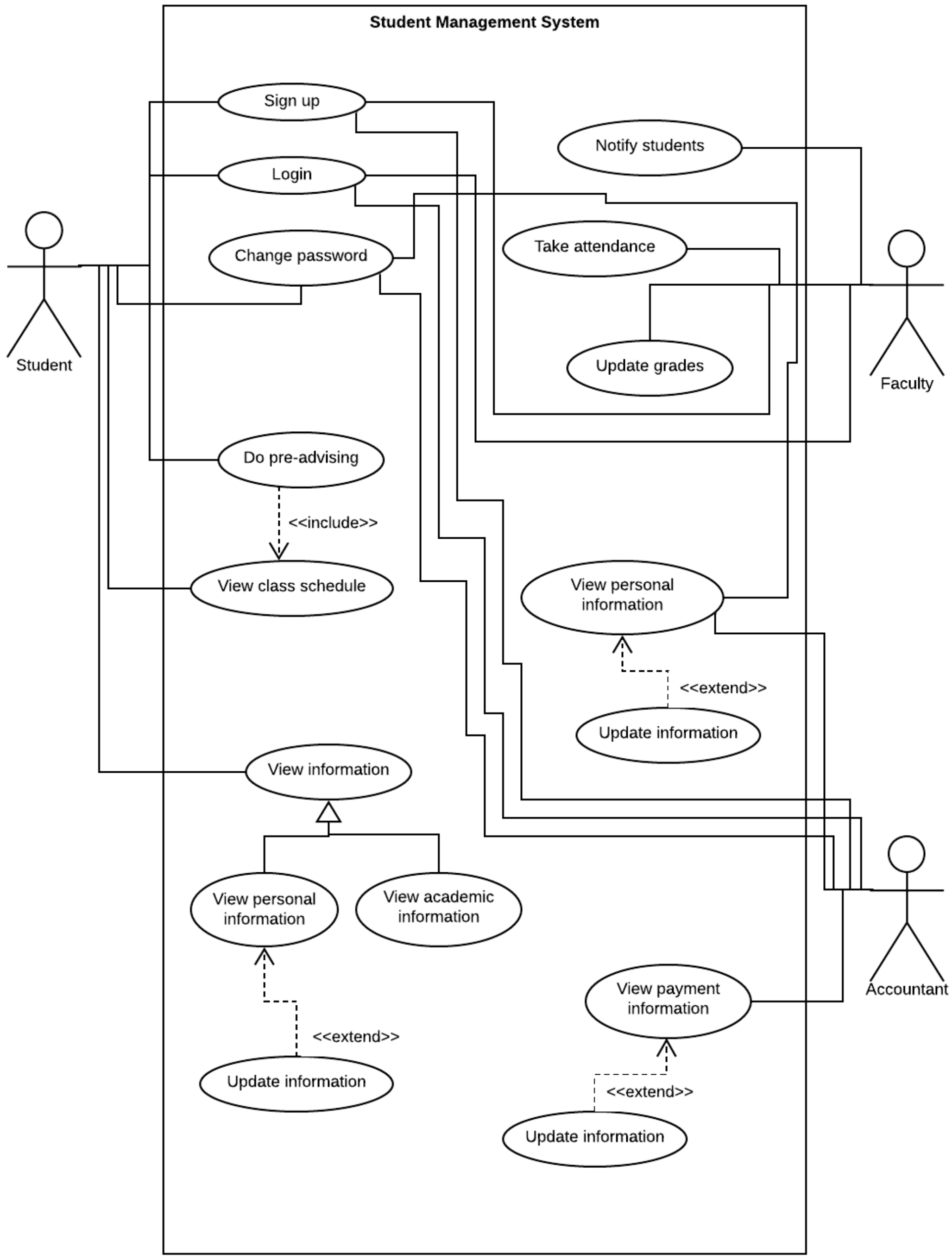
Functional Requirements:

1. Admin can do class scheduling
2. Students can do pre-advising
3. Students can view information, update personal info.
4. Faculty can update marks for students in his course
5. Faculty can take attendance for his courses
6. Admin can add courses
7. Accountant can update payment information
8. Faculty can view information and update personal information

Non-functional Requirements:

1. Server should always remain active
2. Good Internet connection should always be available
3. System should be capable of providing service to multiple user at the same time
4. Pre advising must work in real time
5. Database should always be up and working

Use-Case Diagram



Sign-up
Brief description: The actor signs up
Actors: Student, faculty, accountant
Preconditions: The actor is a registered student or employee
Basic flow of events: 1. The actor signs up to the system 2. The system authenticates the actor
Extensions: (null)
Post-conditions: The actor can login to the system
Special requirements: Actor is connected to internet and actor should have a valid email address

Login
Brief description: The actor logs in to the system
Actors: Student, faculty, accountant
Preconditions: The actor is a registered student or employee
Basic flow of events: 1. The actor logs in to the system 2. The system authenticates the actor and starts session
Extensions: (null)
Post-conditions: (null)
Special requirements: Actor is connected to internet and actor has an authenticated login information

Change password
Brief description: The actor changes password
Actors: Student, faculty, accountant
Preconditions: The actor is a registered student or employee
Basic flow of events: 1. The actor requests for changing password 2. The system approves the request
Extensions: (null)
Post-conditions: (null)
Special requirements: The answer of the security question should be matched

Do pre-advising
Brief description: The actor adds courses, drop courses
Actors: Student
Preconditions: The actor is a registered student and is entering during his/her time slot
Basic flow of events: 1. The actor logs in to the system 2. The actor selects courses 3. The actor add courses 4. The actor leaves the system
Extensions: (null)
Post-conditions: (null)
Special requirements: Actor is connected to internet

View information
Brief description: The actor logs in to the system
Actors: Student, faculty, accountant
Preconditions: The actor is a registered student or employee

Basic flow of events: 1. The actor logs in to the system 2. The actor requests to view information 3. The actor can view personal information 4. The actor can view academic information 5. The actor leaves the session
Extensions: 1a. The system fails to authenticate the actor - The system informs the actor and does not allow the actor to proceed 2a. The system fails to view information - The system informs the actor
Post-conditions: (null)
Special requirements: Actor is connected to internet

Notify students
Brief description: The actor notifies the student
Actors: Faculty
Preconditions: The actor is a registered employee
Basic flow of events: 1. The actor logs into the system 2. The actor selects courses and section of students to notify 3. The actor sends the emails or texts
Extensions: 1a. The system fails to authenticate the actor - The system informs the actor and does not allow the actor to proceed 2a. The system fails to notify students - The system informs the actor
Post-conditions: (null)
Special requirements: Actor is connected to internet

Take attendance
Brief description: The actor takes attendance of the student
Actors: Faculty
Preconditions: The actor is a registered employee
Basic flow of events: 1. The actor logs into the system 2. The actor selects courses and section of students to take attendance 3. The actor selects student to give attendance
Extensions: 1a. The system fails to authenticate the actor - The system informs the actor and does not allow the actor to proceed 2a. The system fails to select courses and section - The system informs the actor
Post-conditions: (null)
Special requirements: Actor is connected to internet

Update grades
Brief description: The actor updates grades for student
Actors: Faculty
Preconditions: The actor is a registered employee

Basic flow of events:

1. The actor logs into the system
2. The system authenticates the actor and starts session
3. The actor selects courses and section
4. The actor selects student id
5. The actor inserts marks
6. The actor leaves the system

Extensions:

1a. The system fails to authenticate the actor

- The system informs the actor and does not allow the actor to proceed

Post-conditions: (null)

Special requirements: Actor is connected to internet

View personal information

Brief description: The actor logs in to the system

Actors: Faculty, accountant

Preconditions: The actor is a registered employee

Basic flow of events:

1. The actor logs in to the system
2. The system authenticates the actor and start session
3. The actor requests to view personal information
4. The actor can update payment information if needed
5. The actor leaves the session

Extensions:

1a. The system fails to authenticate the actor

- The system informs the actor and does not allow the actor to proceed

2a. The system fails to view information

- The system informs the actor

Post-conditions: (null)

Special requirements: Actor is connected to internet

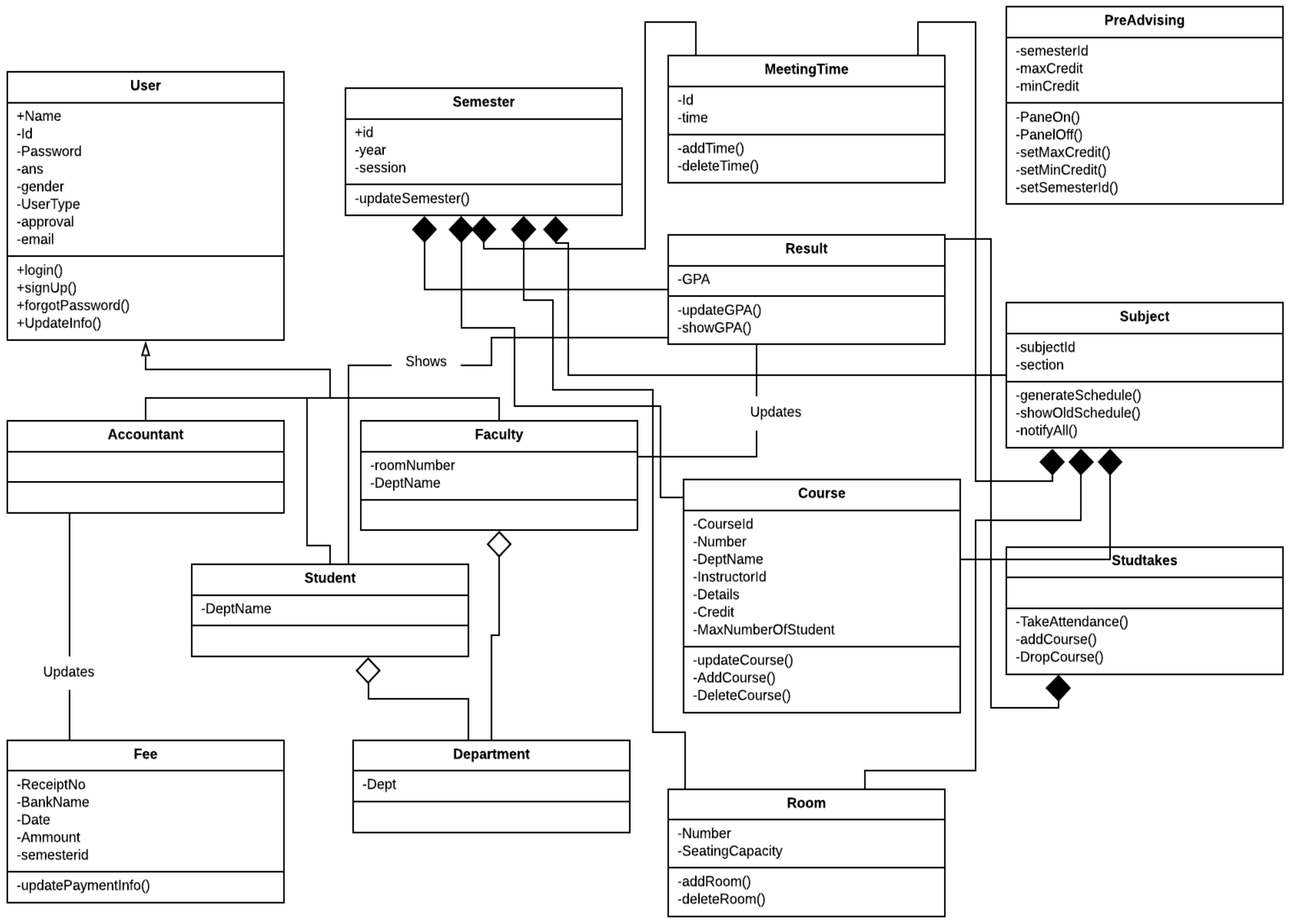
View payment information
Brief description: The actor logs in to the system
Actors: Accountant
Preconditions: The actor is a registered employee
Basic flow of events: 6. The actor logs in to the system 7. The system authenticates the actor and start session 8. The actor requests to view payment information 9. The actor selects id of the students to view specific payment information 10. The actor can update payment information if needed 11. The actor leaves the session
Extensions: 1a. The system fails to authenticate the actor - The system informs the actor and does not allow the actor to proceed 2a. The system fails to view information - The system informs the actor
Post-conditions: (null)
Special requirements: Actor is connected to internet

Analysis and Design

Architectural Design: Repository is used in a system in which large volumes of information are generated that has to be stored for a long time in a central server. Components do not interact directly only through repository. Therefore, to make our USIS system this architecture model has been used where all the information has been saved in a central server which is the core of our system.

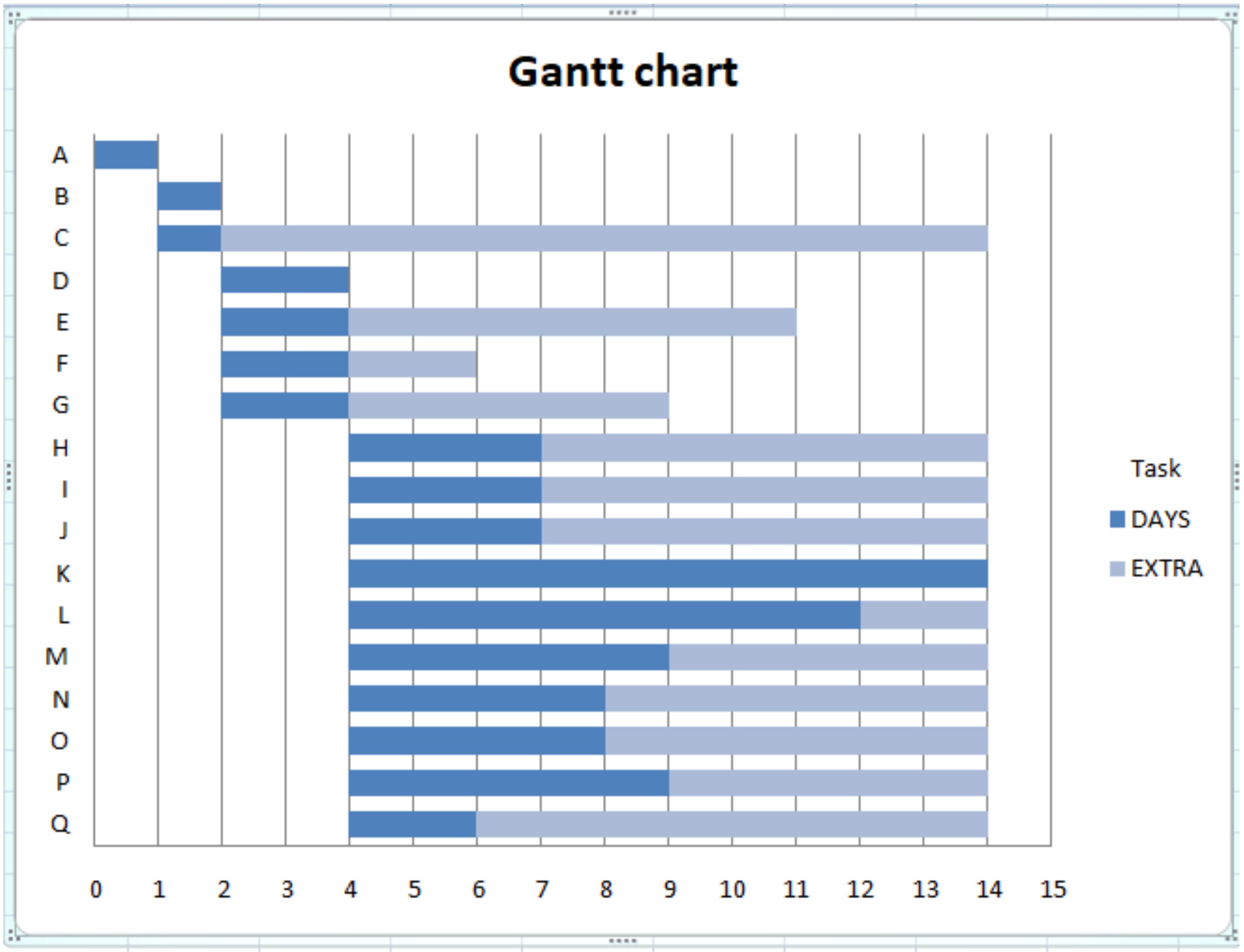
Risk	**Probability**	**Effect**
1.Server goes down	Low	Catastrophic
2.Internet connection goes down	Moderate	Catastrophic
3.Uncontrolled network traffic	Moderate	Serious
4.Chances of hacking and data theft	High	Serious
5.Faults in reusable software components have to be repaired before these components are reused	Moderate	Serious
6.Changes to requirements that require major design rework are proposed	Moderate	Serious
7.The rate of defect repair is underestimated	Moderate	Tolerable

Class Diagram



Critical Path Method (CPM)

Tasks	Predecessor	Duration
A. Sign up	A	1
B. Login	A	1
C. Forget password	B	1
D. Admin panel	B	2
E. Accountant panel	B	2
F. Student panel	B	2
G. Faculty panel	B	2
H. Show student info	F	3
I. Show accountant info	E	3
J. Show faculty info	G	3
K. Class scheduling	D	10
L. Pre-advising	F	8
M. Show result	F	5
N. Add marks	G	4
O. Attendance	G	4
P. Notify Students	G	5
Q. Payment	E	2



Complexity Adjustment Factor (C.A.F)

= [0.65 + 0.01 x _ (*Fi*)]

=0.65+ (0.01 x 41)

=1.06

Fi

= (1+4+3+2+5+1+3+5+4+2+3+2+1+5)

=41

FP Estimated

= count-total x C.A.F

=362 x 1.06

=383.72

COCOMO

Effort estimation:

E= ab*((KLOC)^ bb

=3.0*(5^1.12)

=18.20

=19 person-month

Duration estimation:

D= (cb *(E^db))

=2.5*(18.20^0.35)

=6.90

=7 months

Person estimation:

N= E/D

= 18.20/6.90

= 2.64

=3 persons approx.

Development and Testing

Test Cases

Test case ID	Test Title	Testing Steps	Test Data	Expected Results	Actual Results	Pass/Fail
TC-01	Check user login with invalid data	1.Enter username 2.Enter password 3.Click login	<<valid username>> <<invalid password>>	Fail to login	As expected	Pass

Test case ID	**Test Title**	**Testing Steps**	**Test Data**	**Expected Results**	**Actual Results**	**Pass/Fail**
TC-02	Check user signup with invalid data	1.Enter user name, ID, gender, dept, security question and answer, email, password 2.Click sign up button	<<valid information>> <<invalid email>>	Fail to sign up	Unexpected result	Fail

Test case ID	**Test Title**	**Testing Steps**	**Test Data**	**Expected Results**	**Actual Results**	**Pass/Fail**
TC-03	Check administrator approval after sign up	1.Login 2.Go to approve list 3.Click approve button	<<valid information>>	Administrator can approve sign up	As expected	Pass

Test case ID	**Test Title**	**Testing Steps**	**Test Data**	**Expected Results**	**Actual Results**	**Pass/Fail**
TC-04	Checking administrator blocking power	1.Login 2.Go to approve list 3.Click block button	<<valid information>>	Administrator can block students and faculties	As expected	Pass

Test case ID	**Test Title**	**Testing Steps**	**Test Data**	**Expected Results**	**Actual Results**	**Pass/Fail**
TC-5	Checking session validity	1.Login 2.Select pre-advising session 3.Click on button for pre-advising	<<valid response of button>>	Administrator starts and ends session for pre-advising	As expected	Pass

Test case ID	**Test Title**	**Testing Steps**	**Test Data**	**Expected Results**	**Actual Result**	**Pass/Fail**

TC-06	Check if faculty can take attendance	1.Login 2.click take attendance button 3.Select section and course 4.Take attendance	<<valid data>>	Attendance are being stored in database	As expected	Pass

Test case ID	**Test Title**	**Testing Steps**	**Test Data**	**Expected Results**	**Actual Result**	**Pass/Fail**
TC-07	Check whether faculty can send notification	1.Login 2.Select section and course 3.Click notification 4.send email	<<valid student email>> <<valid connection>>	Faculty can notify students	As expected	Pass

Test case ID	**Test Title**	**Testing Steps**	**Test Data**	**Expected Results**	**Actual Result**	**Pass/Fail**
TC-08	Check if student can drop courses	1.Login 2.Go to pre advising panel 3.Select course 4.Click drop course button	<<valid selected course>>	Student can drop course	As expected	Pass

Test case ID	**Test Title**	**Testing Steps**	**Test Data**	**Expected Results**	**Actual Result**	**Pass/Fail**
TC-09	Check whether accountant can update payment information	1.Login 2.Input student id 3.Update payment info	<<valid student info>>	Accountant cannot update payment info	Unexpected	Fail

Test case ID	**Test Title**	**Testing Steps**	**Test Data**	**Expected Results**	**Actual Result**	**Pass/Fail**
TC-10	Check to see if clashes in schedules exist	1.Login 2.Insert course id, time, date 3.Click generate class schedule	<<valid time>> <<valid date>> <<valid course id>>	Admin can make class schedule without clashes	As expected	Pass

Software Quality Assurance

SOFTWARE QUALITY ASSURANCE **(SQA)** is a set of activities for ensuring quality in software engineering processes that ultimately results, or at least gives confidence, in the quality of software products.

Software Quality Assurance encompasses the entire software development life cycle and the goal is to ensure that the development and maintenance processes are continuously improved to produce products that meet specifications. In our project, the features were continuously modified and tested in order to check whether they met the requirements or not. The scope of Quality is NOT limited to just Software Testing, how well the requirements are stated and managed matters a lot as well. We kept regular contact with our clients in order to find out their specific requirements so we could manage them accordingly.

Once the processes have been defined and implemented, we had a responsibility of identifying weaknesses such as bugs or errors in the code and correcting those weaknesses to continually improve the processes.

QA Metrics help us to measure the quality and make sure we meet our quality guidelines. Measuring quality helps us to make the right decisions at right times and identify actionable trends. Metrics play an important role in tracking the progress and quality. There are different aspects of quality that can be measured through the QA metrics.

Metrics can be categorized as below.

- **Process Metrics**: These can be used
  to improve the process efficiency of the Software Development Life Cycle, Testing process, etc. The debugging process took around 1 week for us to find and solve the bugs or defects in our code.
- **Product Metrics**: Product Metrics deal with the quality of the software such as product size and complexity. Our USIS had around 5k lines of code and its complexity is mediocre due to algorithms such as Genetic algorithm.
- **Project Metrics**: These can be used to measure the efficiency of a project team or any testing tools being used by the team members. Our team consisted of 5 members and the project took around 6 weeks to complete. There was no physical cost involved in making this project other than the time and effort given on it.

Implementation and Effort Distribution

Effort Distribution

The effort distribution of our project is as follows:

- Mahmudul Haque- 22%
- Md Rafid Mushfique- 20%
- Sadia Afrose- 20%
- Homaira Huda Shomee- 19%
- Alvina Awwal- 19%

The database design and implementation was done by Alvina and Homaira along with majority of the report work.

The UI interface was mainly designed and done by Sadia using JavaFX along with a lot of help in writing the report.

Mahmudul and Rafid did most of the critical coding parts along with the algorithms and debugging. They contributed in some parts of the report as well.

Implementation

We have manipulated only the Database throughout our project by storing, retrieving, updating data. Other than that for class scheduling, we have used Genetic Algorithm which gives an optimal solution so that there are no clashes between the classes.

Genetic Algorithms (GAs) are adaptive heuristic search algorithms that belong to the larger part of evolutionary algorithms. Genetic algorithms are based on the ideas of natural selection and genetics. These are intelligent exploitation of random search provided with historical data to direct the search into the region of better performance in solution space. They are commonly used to generate high-quality solutions for optimization problems and search problems.

The whole algorithm can be summarized as –

1) Randomly initialize population

2) Determine fitness of population

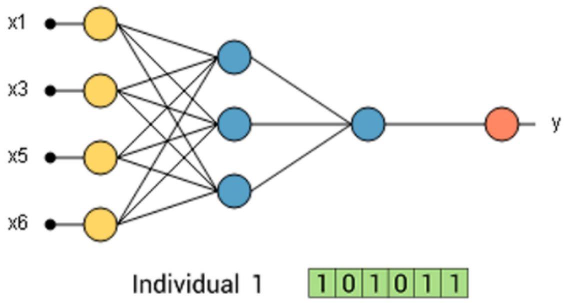
3) Until convergence repeat:

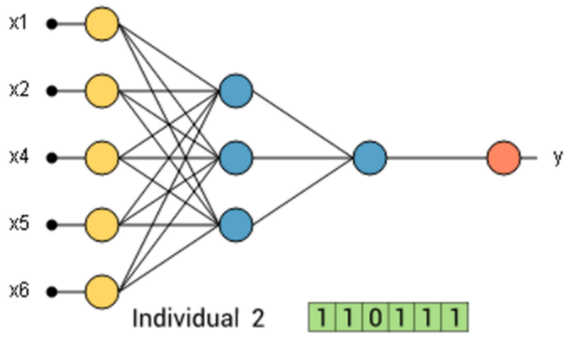
   a) Select parents from population

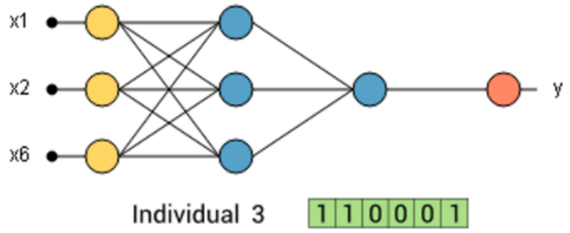
   b) Crossover and generate new population

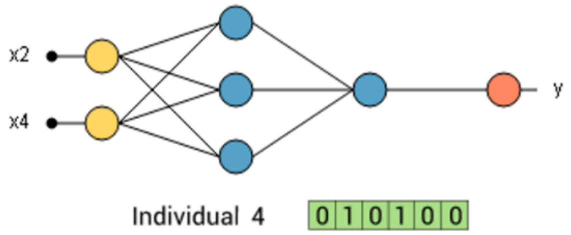
c) Perform mutation on new population

d) Calculate fitness for new population









<u>Conclusion</u>

Our project for USIS was a great learning experience for all of us since it helped us learn JavaFX, new algorithms such as Genetic algorithm, etc. It helped develop our teamwork skills, time management and punctuality, meeting deadlines and schedules as well as respecting our team members' opinions. We learned practical implementation of theory such as CPM, FP, COCOMO, Gantt chart, etc.

This course (CSE470) was a very valid teaching experience since Software Engineering is vital for all CSE students. Furthermore, we also implemented our learnings by creating a project with a report for University Student Information System which helped clear our concepts and make use of everything that we knew and had learned this semester.