BSC. HONS IN BUSINESS INFORMATION TECHNOLOGY

Course Name

COMP1640

Course Title

Enterprise Web Software Development

Submitted to:

Dr Ray Stoneham

Course Leader,

University of Greenwich, UK

Submitted by:

Name: S. M. Abdul Wassae

ID: 000990169

Date of Submission: 13th April 2017

Team Name: The Pillar of Success

Team Member:

ID	Name	Role
000990134	Tabana Islam Trisha	Information Architect, Scrum Master
000990133	Mahfuzur Rahman Bhuiyan	Web Designer
000990169	S. M. Abdul Wassae	Database Designer, Programmer
000914249	Ankhi Akter	Tester

Table 1: team member list and their role

Application Liv

Screencast UR

Repository UR

https://drive.goo

p=sharing

Credential:

Role	Username	Password
Administrator	Admin	1234
EC Manager(ECM)		
Coordinator		
Coordinator		
Student		
Student		

Table of Contents

Introduction

Extenuating Circumstances is a role based system for a large university which is considered as a subsystem of a large system. There are different roles that have different level of work. It actually used for submit and accept claim under department and monitored by higher level of user. There are four roles to handle the entire claim maintenance and they have different type of permissions to do different type of works. The administrator is the superior among them. He/she can add complain subject under any department. He/she can also manage any system data including with change passwords. After admin basically next powerful user is manager but he only can view related reports. The actual claim handler is coordinator who has only access to accept or deny claims and provide their solution to students. Perhaps the student can see complain subject but no claim will be added until he/she submit a claim. The system is highly secured to maintain the roles that no one can access without login and no role can view other role data.

As a solution of this system it is defined as a group work with limited time. So the group work is following agile methodology especially go through scrum. The total work is divided in chunks and group members are playing different roles to solve each chunk. After solving all the chunks it is merged as a total solution. The roles to solve the group work the team need a database designer, an information architect, a programmer, a web designer and a tester, as well as a scrum master and product owner, team leader if needed. One can play multiple roles as number of member in our team is four. Contribution of all team members makes a successful solution.

The report contains a brief description of the system scenario, assumptions, functionality, future developments, evaluation of the group work, own contribution in group as well as important diagrams etc.

Evaluation of product and Process

Scenario

Extenuating Circumstances is a web solution which can be counted as a sub system of any management system of a large university. Basically it is designed and developed to accept claims related to their faculties. It will also manage solutions depend on the evidences provided by student for each claim. Besides being much functionality there are some data assumed to complete the development of the system. The assumptions are given below:

Assumptions:

The assumed data in the role based system are:

- All users are predefined as the system is considered as a part of a large system. All user data is based on assumption.
- Administrator can change any user role
- The departments are predefined in the system and only admin can see the list of the departments. Defined departments are: BBA, CSE, Information Technology, Marketing, MBA and MIS
- Each department has many courses that can be added by administrator.
- Complains that are set by admin will be arranged under courses in each department.
- Coordinator cannot process claims without evidence but can see their details.
- Coordinator can see claims that crossed date for reply but don't have any solution.
- Denied claims won't be shown but will count as a claim and administrator can see them.
- Students can edit the claim details but cannot edit files uploaded or change claim name.
- Student can view solution in their site page.

Roles and their screenshots

It is a role based system where different levels of user have different type of task. The users are defined as four categories as like as Administrator, Manager, Coordinator and Student. The entire roles have some limited boundary of features that is described below:

Administrator:

The EC administrator is called admin as a short form. To access as administrator user have to just enter his/her username or email and password and the system automatically detect his/her user role based on username. If his user role is administrator then he redirected to admin home page.

- Only he can access to view all departments,
- add course under departments,
- set complain subject and details,
- view all claims even denied claims,
- change any user password and define roles.

Admin functionalities and their screenshots are given below:

Login: login form is necessary to enter in admin panel. Without login with admin username or email and password no one can access admin pages.

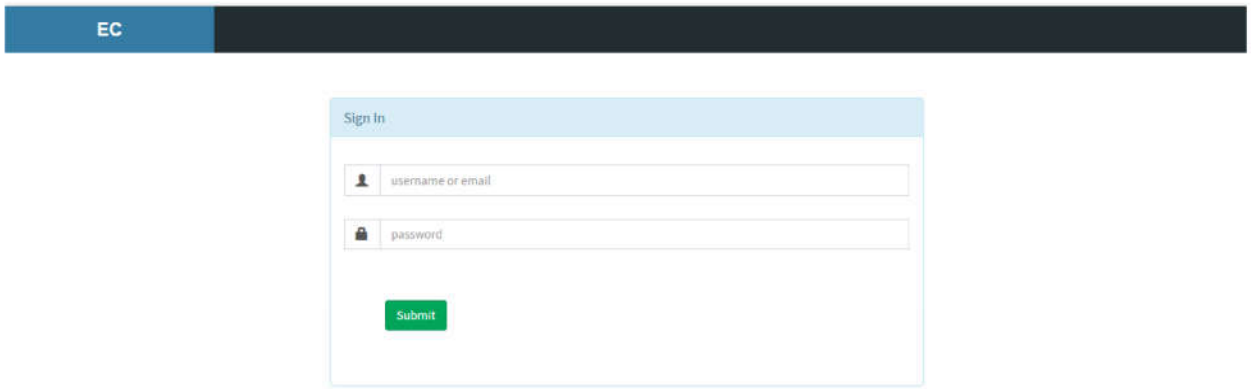


Figure 1: Login Page

Admin home: After login as admin this page appears. It is a dashboard page and admin need to visit this page to find his functionalities.

Figure 2: Admin home page

Add Course: In here admin can add course from predefined departments.

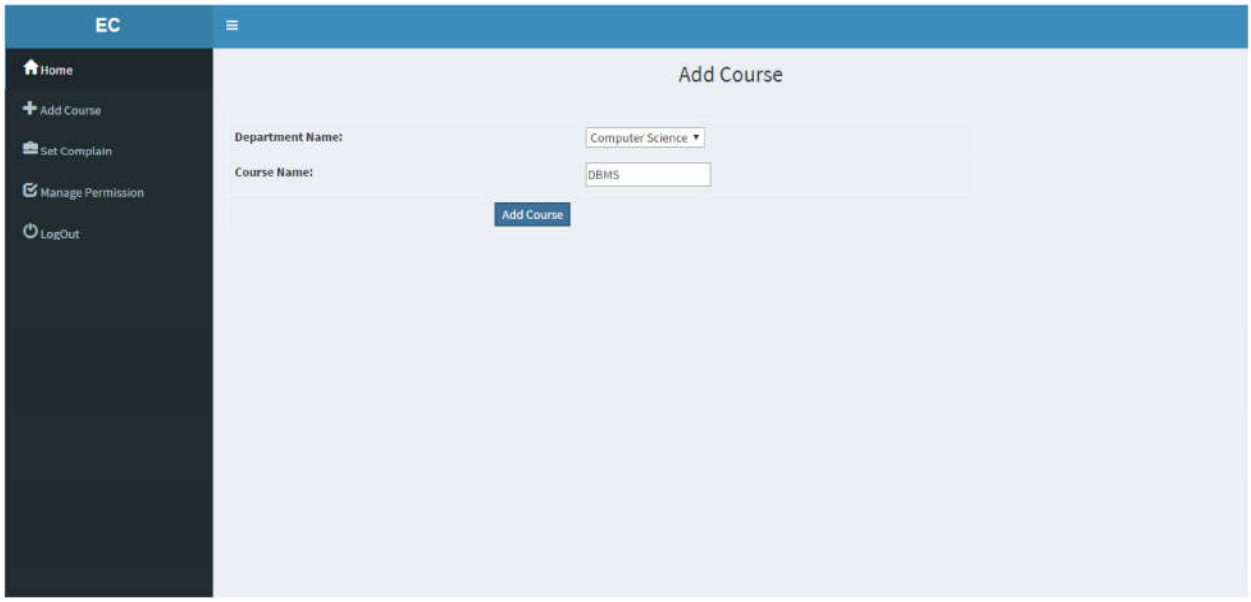


Figure 3: add course

If course added success massage will show to admin

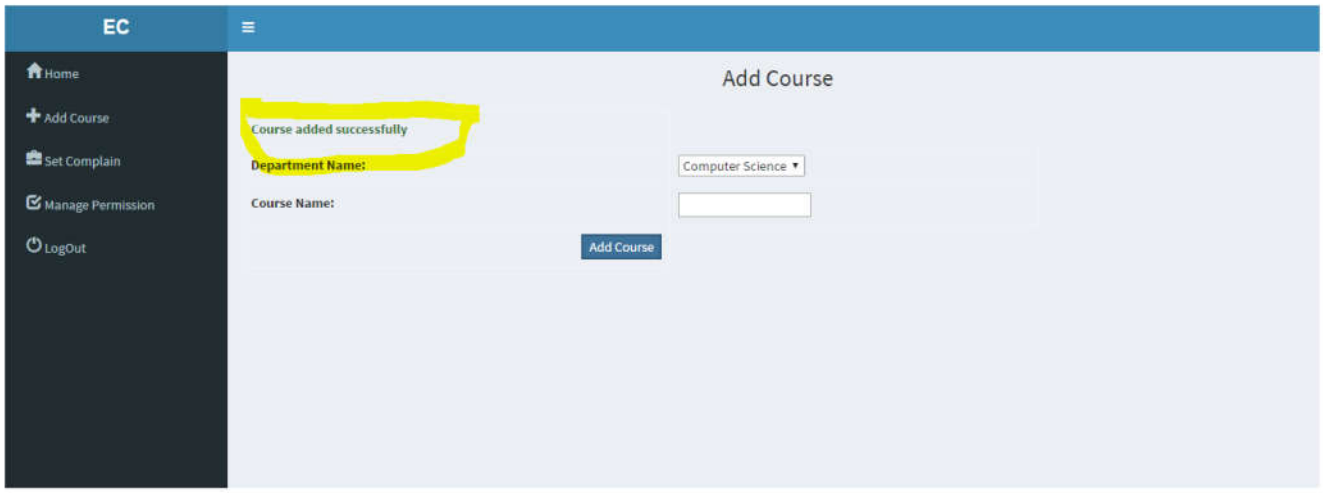


Figure 4: add course success report

Set complains: In this page admin can add complain subject under any course.

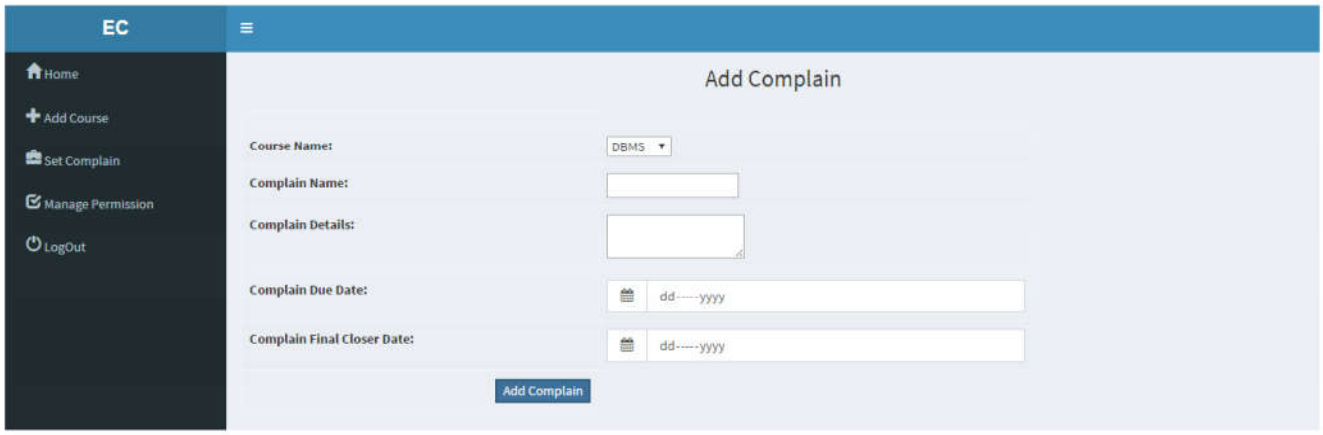


Figure 5: set complain page

Admin is filling his data to set complain

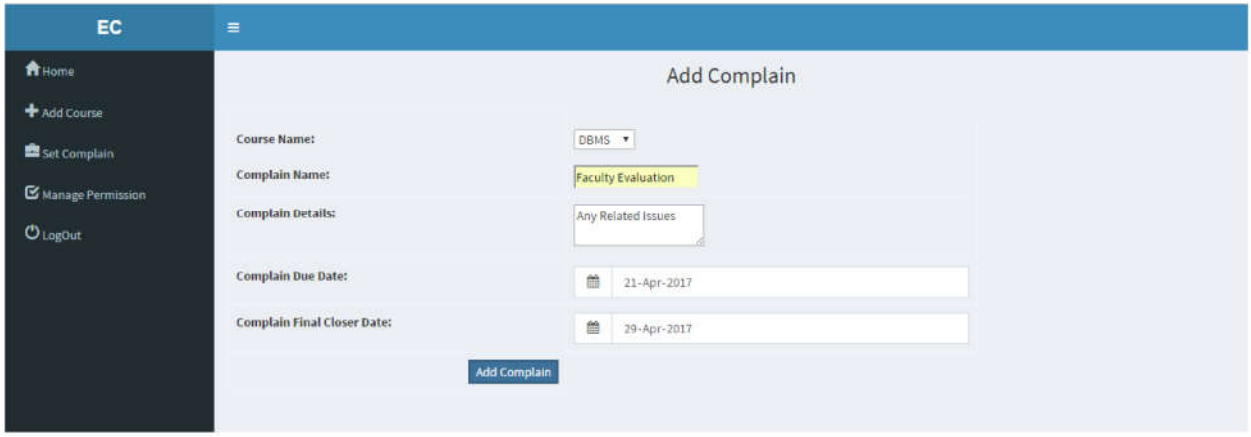


Figure 6: filling set complain data

After add a complain subject success message will shown

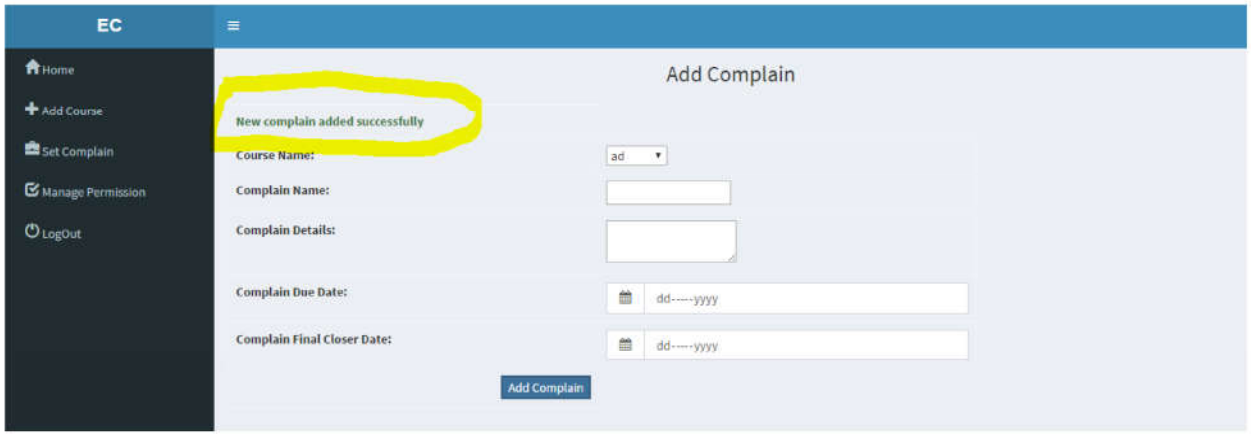


Figure 7: success report of set complain

Set permission: Admin can define roles in this page.

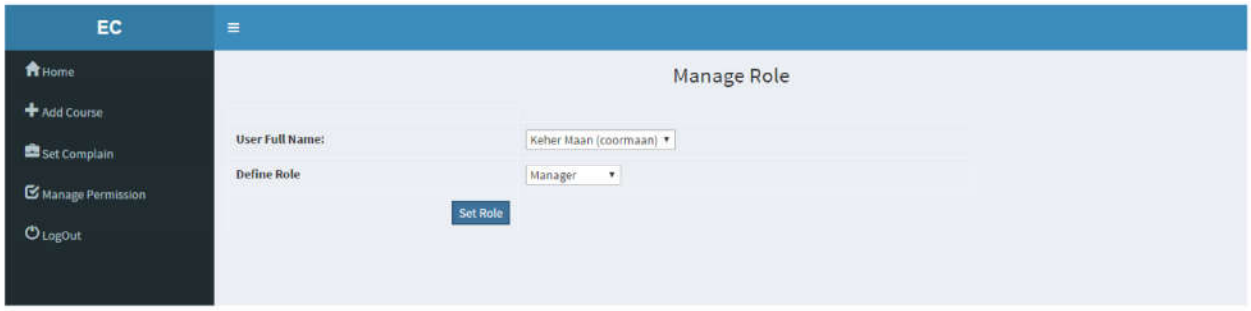


Figure 8: role defining page

View Department List: Department list is shown in this page if admin wish to see.

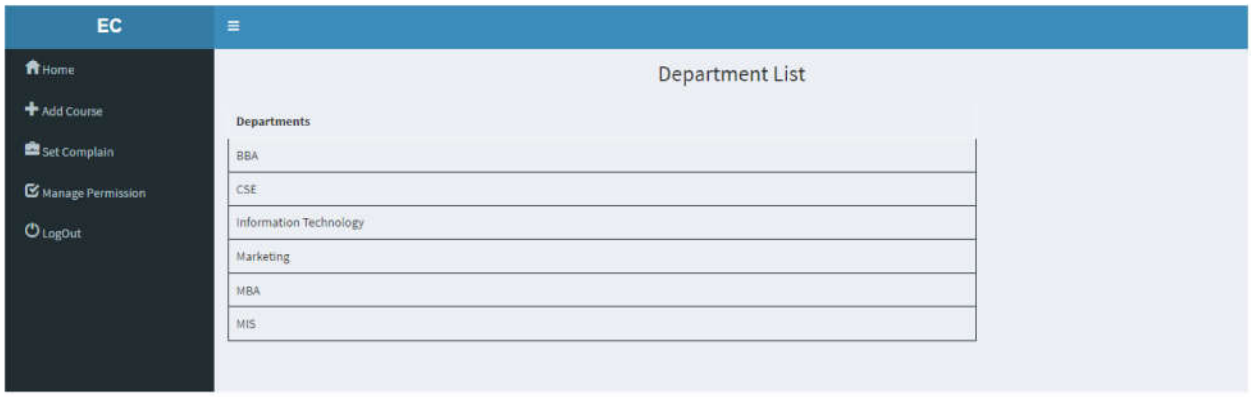
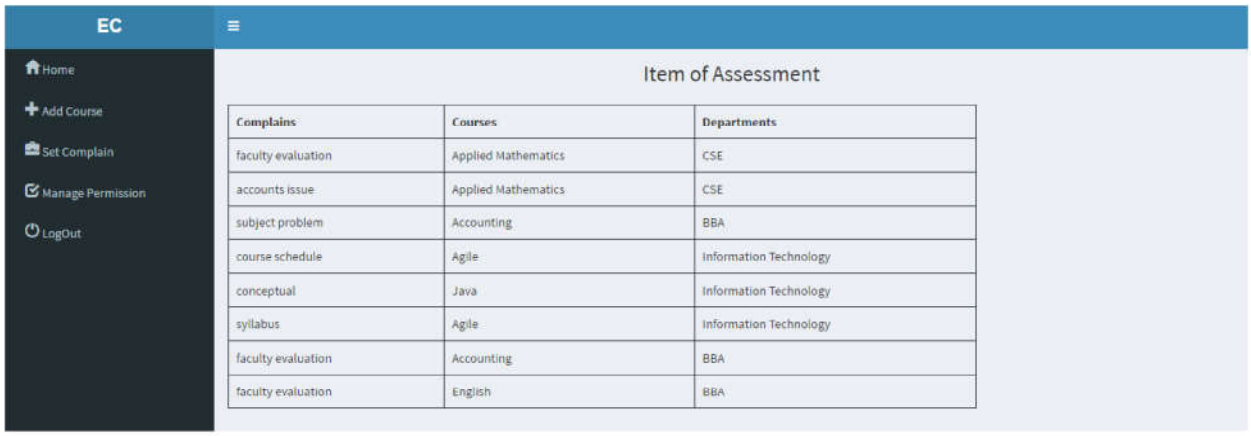


Figure 9: Department list page

View all claims: All claim list and their status are shown in this section for admin.

Figure 10: all claim view page

View items of assessments: Show department and course wise all complain subjects.



View Student List: Show department wise all student list to admin

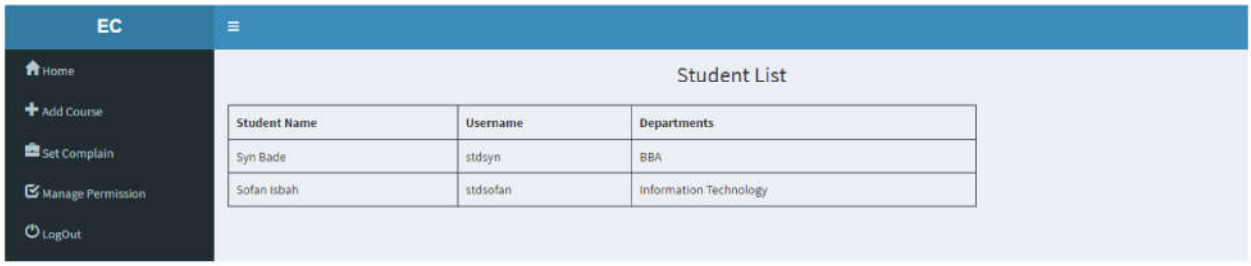


Figure 11: student list page

Change Password: In here admin can change password for any user.

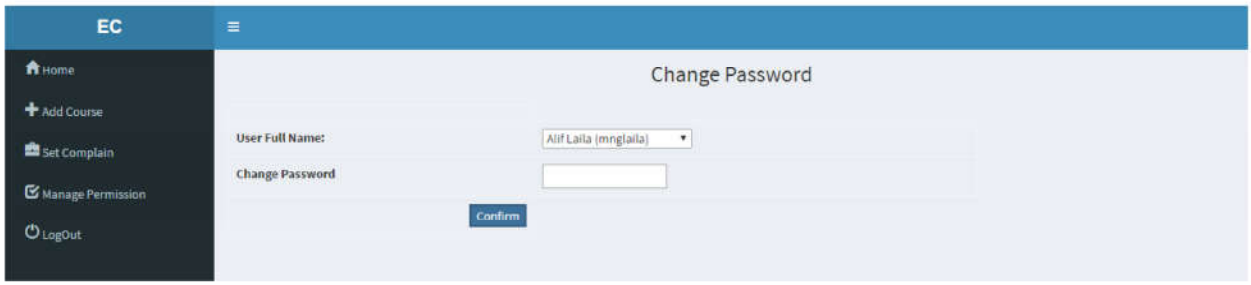


Figure 12: change password page

EC Manager:

The EC Manager is also called manager. As like as administrator he/she simply enter his/her username or email and passwordin login form and the system automatically redirect him/her to manager home page according his role if his/her username and password is considered as correct. Manger of this system will be able

- to see Number of Claims in each Department,
- Yearly Claim report for each faculty in percentage,
- Number of Student complained in Department.

Manager functionalities and their screenshots are given below:

Login: login form is necessary to enter in Manager home. Without login with admin username or email and password no one can access manager pages.

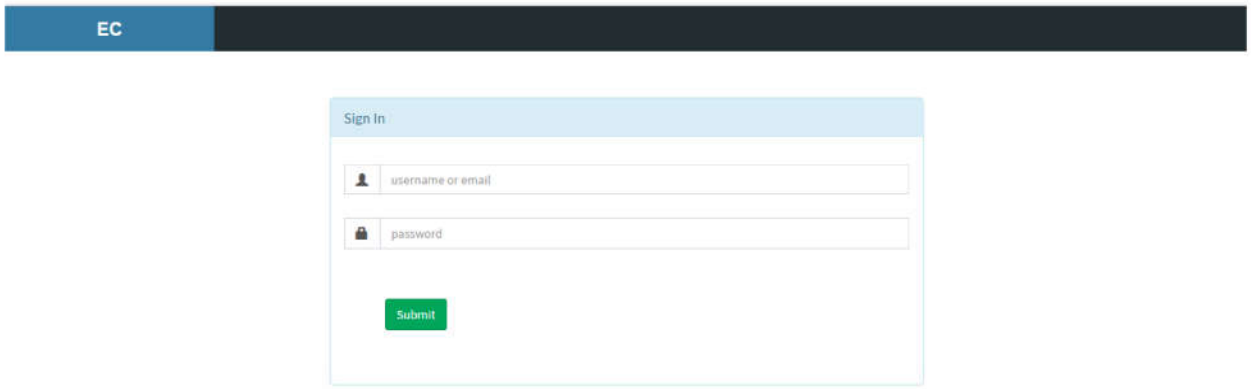


Figure 13: login page

Manager home: After login as manager this page appears. It is a dashboard page and manager needs to visit this page to see his functionalities.

Figure 14: manager home page

Number of Claims in each Department: This page shows manager how many claims come from each department. Here is example of two departments.

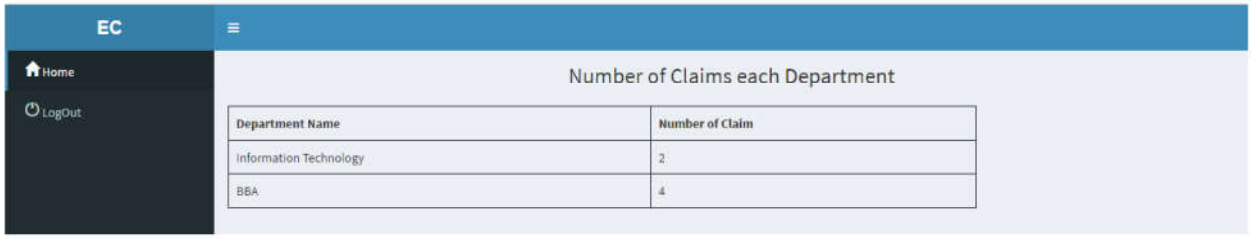


Figure 15: Page shows number of claims for each department

Yearly Claim report for each faculty in percentage: This page shows manager the percentage report of overall claims per year in each department and their comparison.

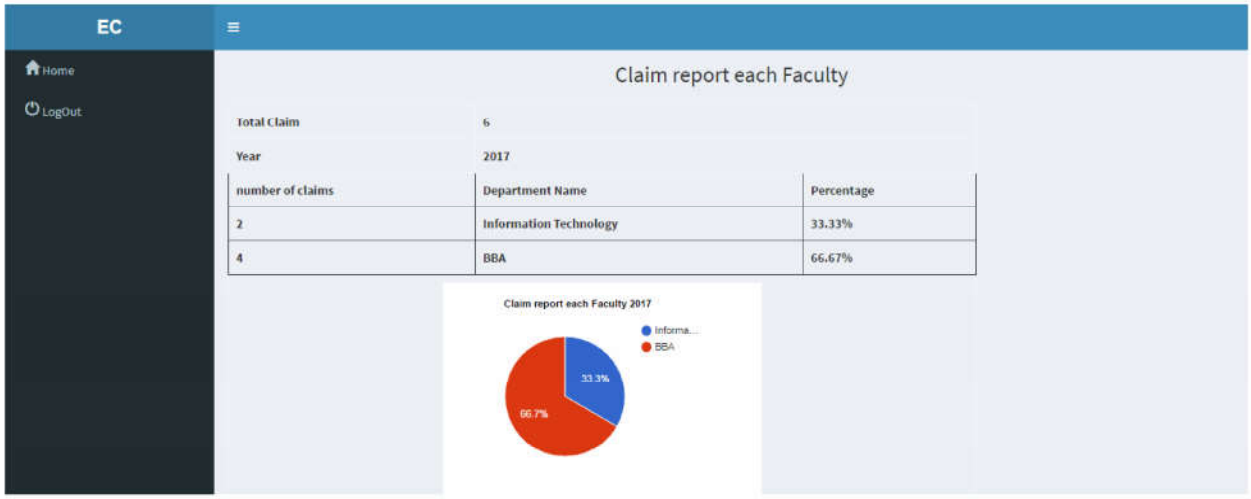


Figure 16: yearly percentage report of claims

Number of Student complained in Department: This page shows admin about how many student claims in each department.

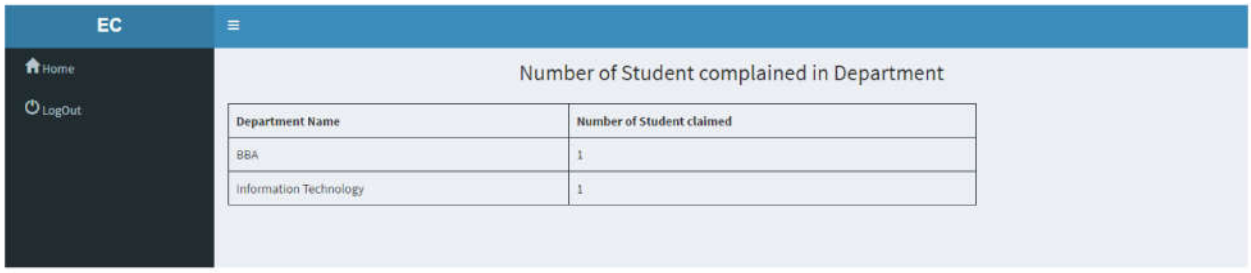


Figure 17: Number of student claim per department

EC Coordinator:

The EC Coordinator is also called coordinator. As like as administrator and manager he/she simply enter his/her username or email and password in login form and the system automatically redirect him/her to coordinator home page according his role if his/her username and password is considered as correct.

- He/she can view claims with evidences or
- view claims with evidences without evidences,
- claims that passed more than 14 days from claim submission date without any decision,
- Can also accept or deny claims within 14 days counted from the submission date-time of student.
- He/she also get an email notification when new claim uploaded.

Coordinator functionalities and their screenshots are given below:

Login: login form is necessary to enter in coordinator home. Without login with admin username or email and password no one can access coordinator pages.

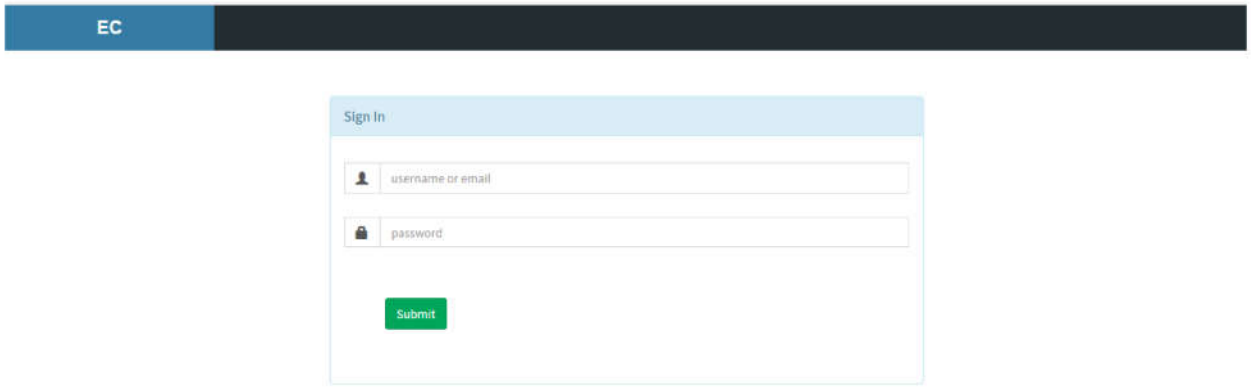


Figure 18: login page

Coordinator Home: After login as coordinator this page appears. It is a dashboard page and coordinator needs to visit this page to see his functionalities.

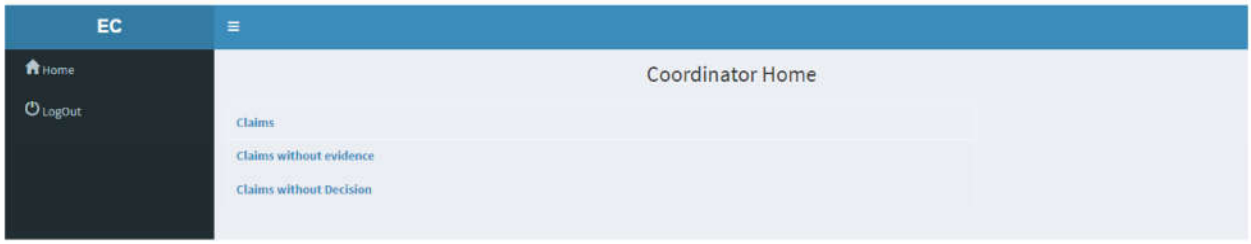


Figure 19: coordinator home page

Claims with evidence: This page shows the claim list that has evidences. Coordinator cannot accept or deny any

Figure 20: claim list with evidence

To accept claims coordinator has to select accept button.

Figure 21: button for accept claim

If deny button clicked then the claim will be denied.

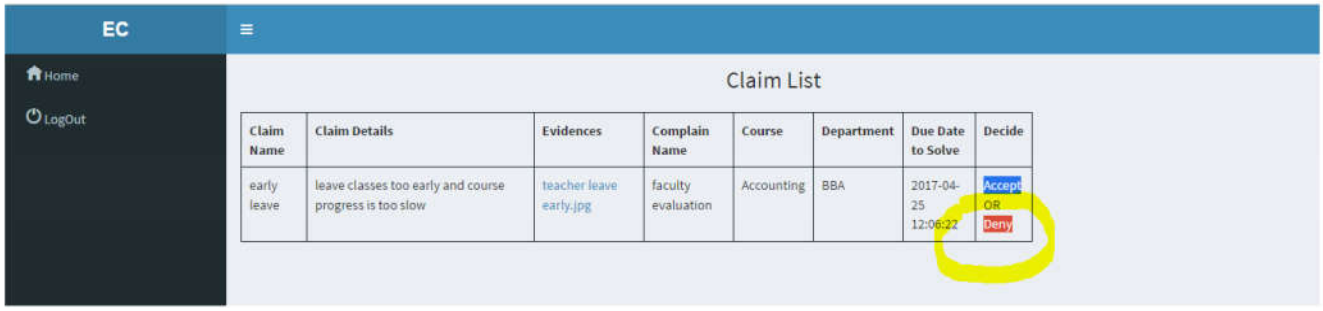


Figure 22: button for deny claim

Reply claim: After click accept button this page will arrive. If coordinator replies solution then the claim will be accepted otherwise it would be pending.

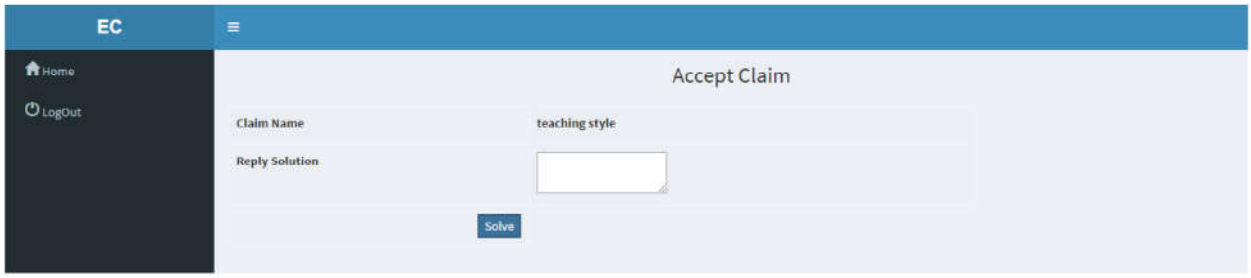


Figure 23: accept claim with a solution provided

Claims without evidence: This page will show coordinator all the claims that have no evidences

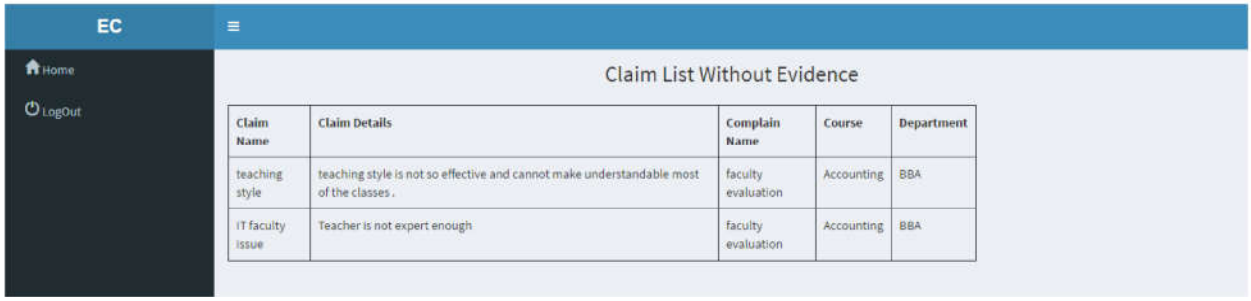


Figure 24: claims without evidences

Unsolved date over claims: Claims that passed more than 14 days from claim submission date without any decision will appear in this page to show coordinator.

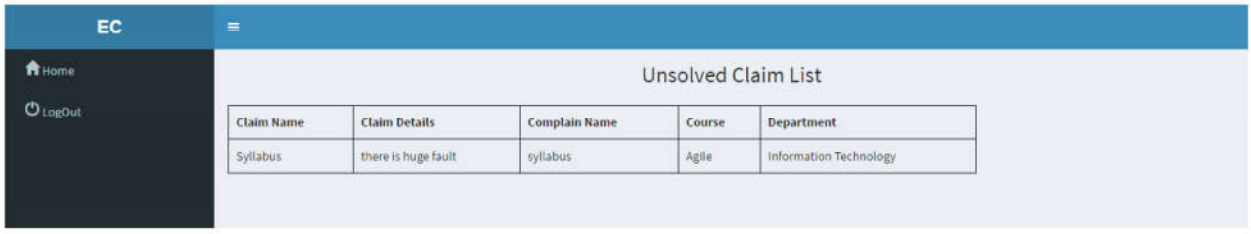


Figure 25: unsolved decision less timeout claims

Student:

As like as other users he/she simply enter his/her username or email and password in login form with proper validation and the system automatically redirect him/her to student home page.

- He/she can view available complain subjects within his/her department,
- Can view available date-time to submit a claim under any available complain,
- Can view finale date-time to upload evidences,
- Can view claims done by him/her,
- Can view claims with solutions.
- He/she also can submit one or more than one new claim under a complain subject with or without evidences.
- He/she can also edit claim description if necessary.
- Student cannot add new claim if it crosses due date of submission
- But he/she can still upload evidence until the final closer date passed away.
- He/she will receive an email notification if any solution given by EC Coordinator.

Student functionalities and their screenshots are given below:

Login: login form is necessary to enter in student home. Without login with student username or email and password no one can access student pages.

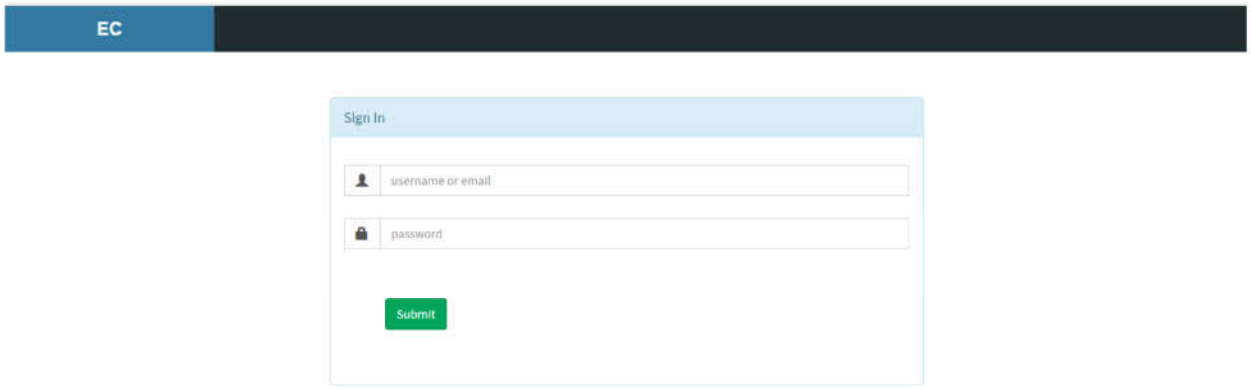


Figure 26: login page

Student home: After login as student this page appears. It is a dashboard page and student needs to visit this page to see his functionalities.

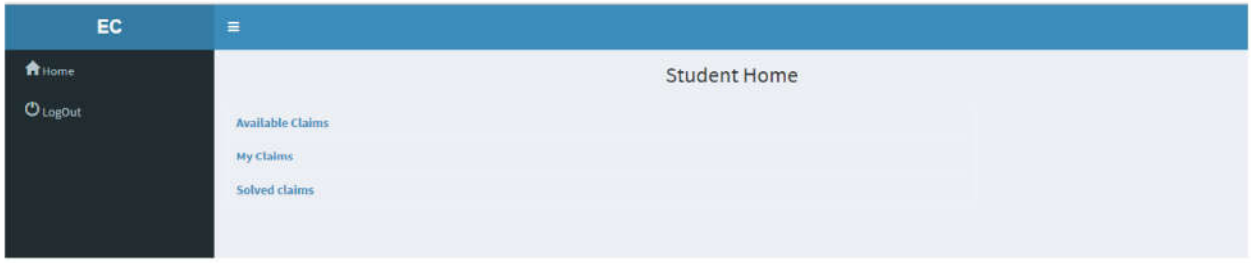


Figure 27: student home

View available claims: This page shows available complain subject their due date to claim and final closure date to upload evidence.

Figure 28: Available complain subject list for student

Show available date-time to submit a claim

Figure 29: due date to submit claim

Show finale date-time to upload evidences

Figure 30: finale closure date for upload evidence

My claims: This page will show the claim list submitted by the student

Figure 31: my claims page

Claims with solutions: In this page student can see the solved claim and the solution provided by the coordinator of his/her department.

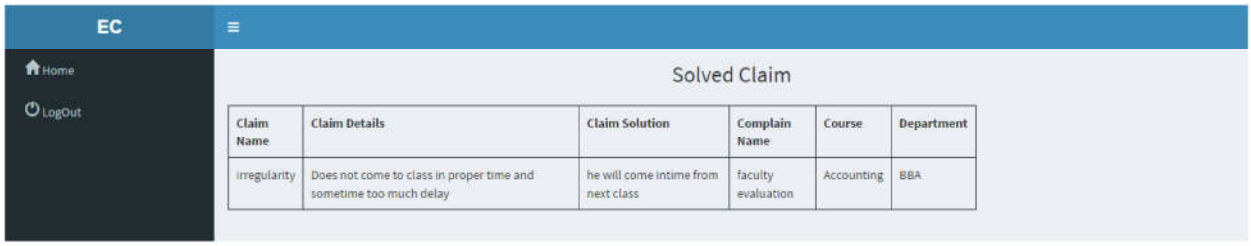


Figure 32: claims that had replied with solution

Submit complains: This page will arrive after click a complain subject. Here student can submit claim.

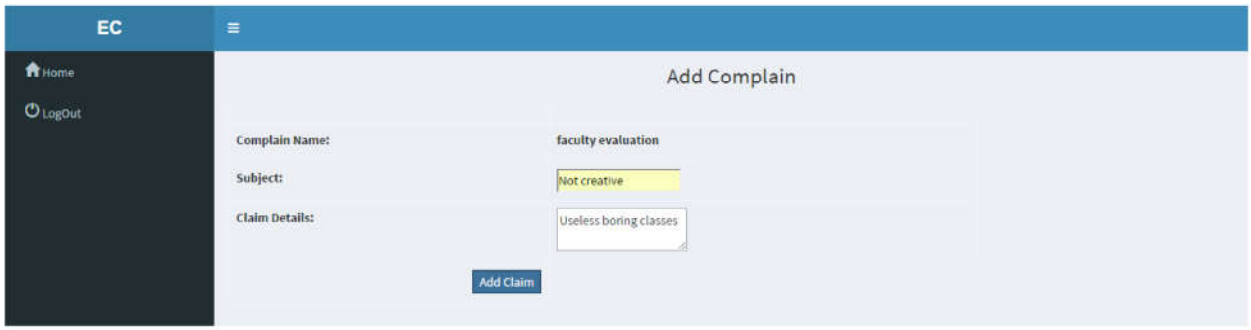


Figure 33: add claim with filled up data

Success message after submit a claim

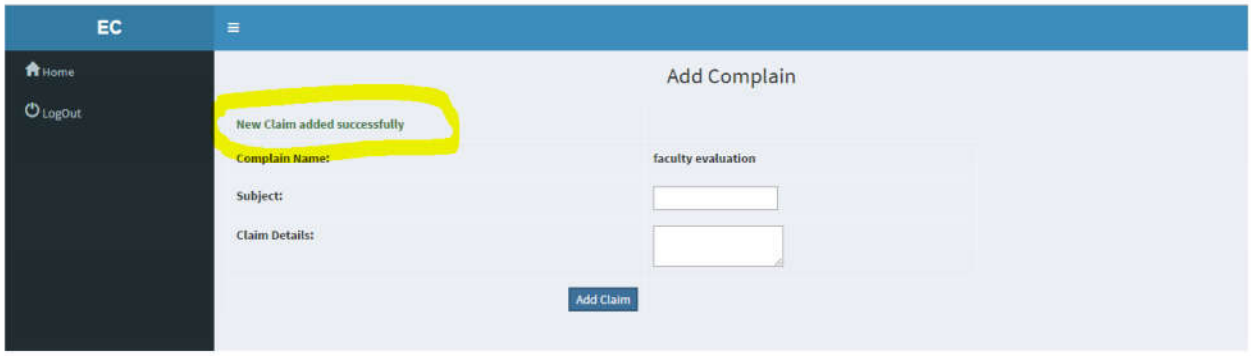


Figure 34: report of claim submit

Edit claims: In this page student can view his claim list to edit details or upload evidence.

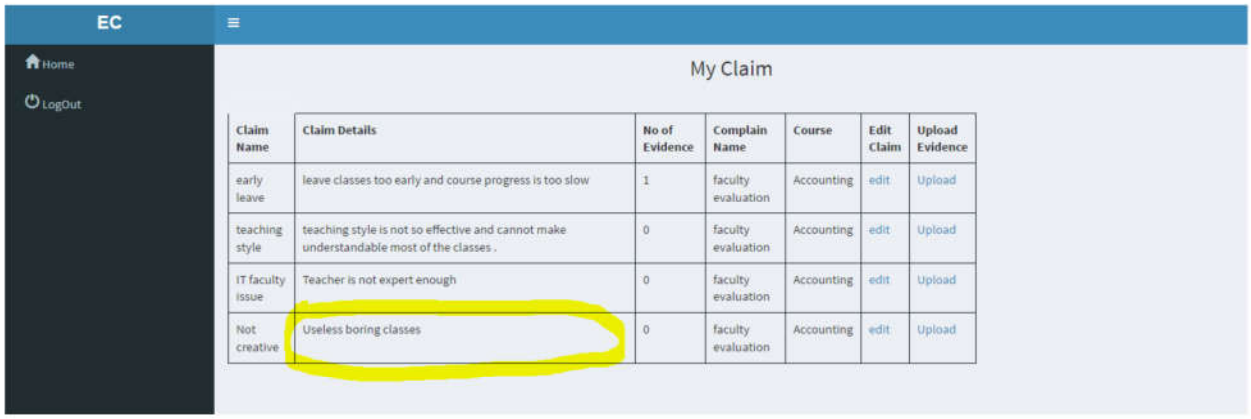


Figure 35: claim list to edit or upload evidences

After click edit link this page will arrive to edit claim details.

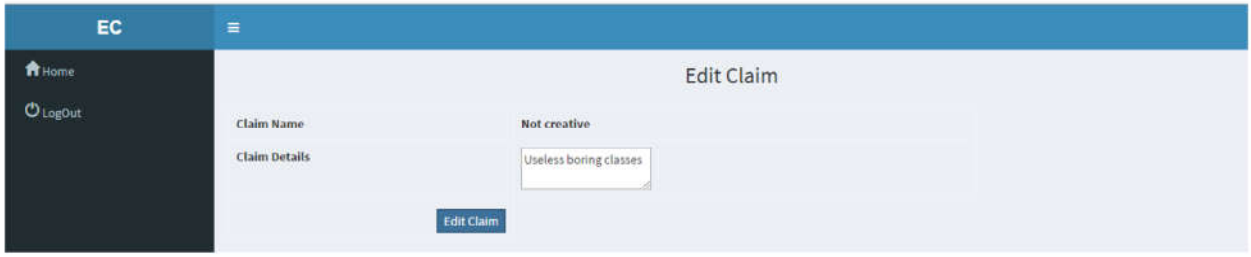


Figure 36: page to edit claim details

Claim details shown changed in the edit claim page.

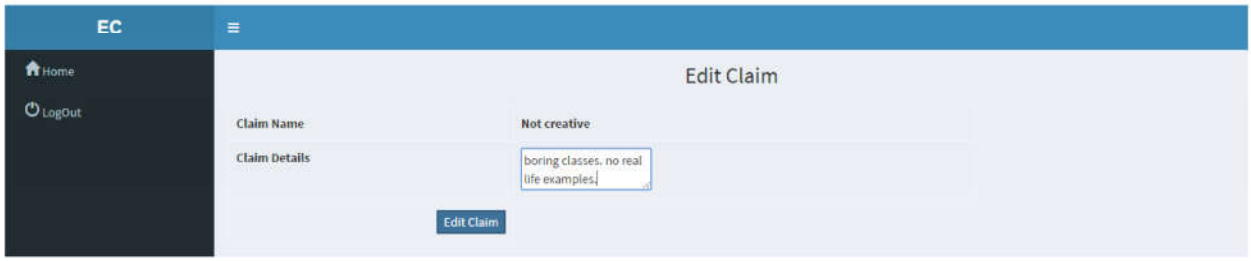


Figure 37: data changed in edit claim

In claim list data is shown as changed details.

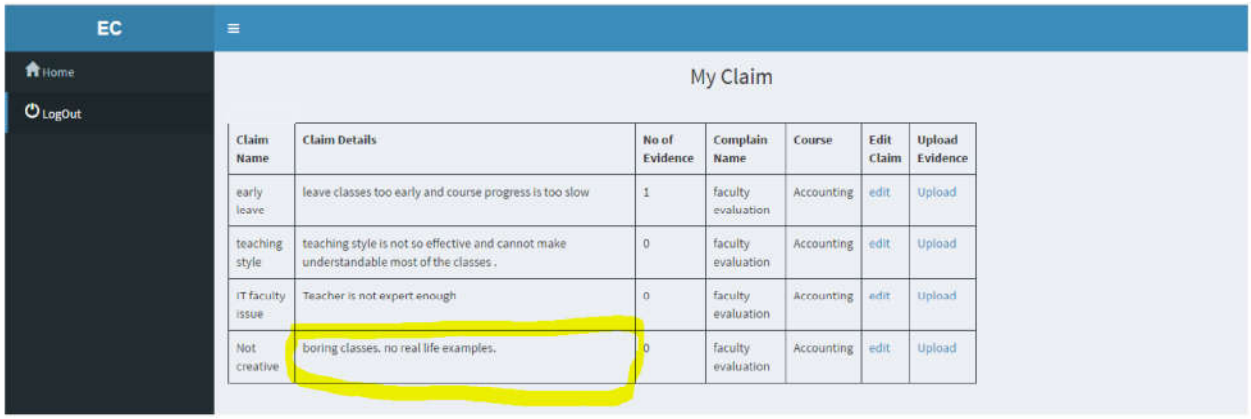


Figure 38: details changed after edit claim

Upload evidence: From claim list page user can upload file as jpeg, jpg, png, gif and pdf by clicking upload link.

Figure 39: Upload link in claim list

Add Evidence: This page will arrive if upload link from claim list page is clicked. Here student can upload the evidences.

Figure 40: add evidence page with file added

After click add evidence number of evidence is changed in claim list page

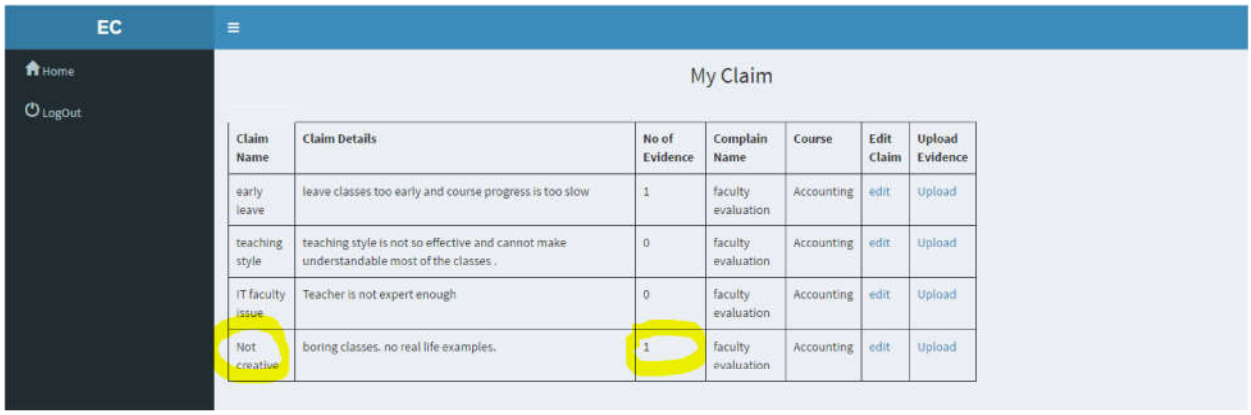


Figure 41: number of evidence changed if evidence uploaded

Evidence show: To see the uploaded evidence student have to visit add evidence page by clicking upload link again.

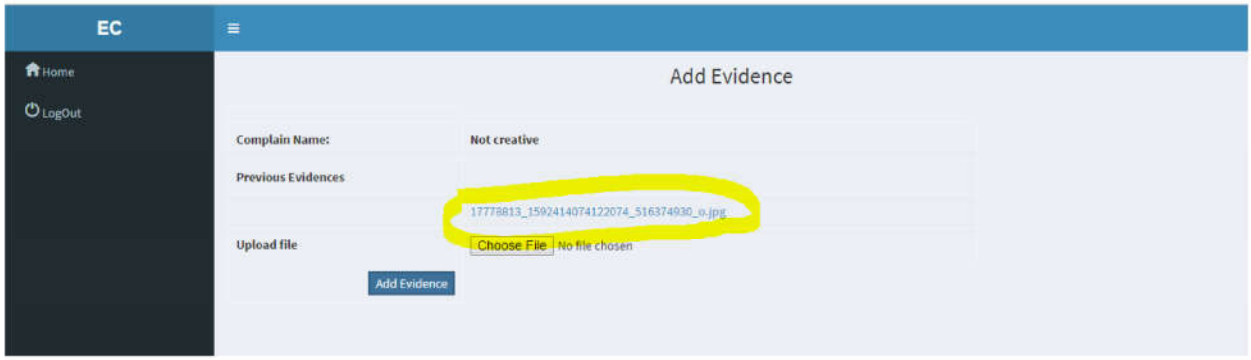


Figure 42: show link of uploaded folder

Technology used

Languages

- **Html5:** In present it is the latest standard for browsers to display web pages which was Approved in 2014. (html5, n.d.)
- **Css3:** It is the latest update of Cascading Style Sheets coding language which is basically the sets of building to design and build any website prototype. (css3, n.d.)
- **php7:** It is a major release of PHP programming language that used to build web application which can be developed and delivered for mobile to enterprises and the cloud. (php7, n.d.)
- **Javascript:** It is a cross-platform, object-oriented, small and lightweight scripting language. Which can be connected to the objects of a host environment (for example, a web browser) to provide programmatic control over them. (js, n.d.)

Database

- **mysql:** MySQL is one of the most popular open source database which has proven performance, reliability and ease-of-use for web-based applications and used by high profile web properties. (mysql, n.d.)

Framework

- **Bootstrap3:** It is a front-end framework that used for building responsive, mobile-first websites. It expenditures a mix of HTML5 markup, compiled and minified CSS styling, fonts, and JavaScript in which its CSS is customizable, and even supports Less and Sass pre-compiled CSS, in addition to plain CSS files. (bootstrap, n.d.)

Pattern

- **MVC:** MVC means model-view-controller which actually is a methodology or design pattern for successfully and efficiently relating the user interface to underlying data models in a object-oriented programing. (MVC, n.d.) It is divided in three part such as
    - **Model:** A logical structure of data modeling which does not contain any user interface.
    - **View:** A connection of classes which used for representing user interface.
    - **Controller:** It is a structure of classes which make connection and handle data passing over view and model.

Libraries

- **PDO:** PDO represents PHP Data Objects which is a lean, consistent way to access databases. Basically it is well-designed API of a general database abstraction layer with support for MySQL among many other databases. It helps to more easily move to another database. (pdo, n.d.)
- **jQuery:** jQuery is a fast, small, and feature-rich JavaScript library which can used for as like as HTML document traversal and manipulation, event handling, animation, and Ajax. It is much simpler with an easy-to-use API works across a multitude of browsers. (jQuery, n.d.)

Further Development:

Our developed system is robust though the system is developed in a very short time. So it can extend in a larger version. In future more features can be added as like as given below:

- User registration
- Searching claims
- Larger edition of database like postgresql can be implemented.
- More variety of reports.
- Guest user role.
- Now admin can see any system data but in future he can add departments
- Editing personal info can be added.

Evaluation of team

We have named our team as “The Pillar of Success” which consists of four members. There is a great balance lied beyond our team. At first we arranged a team meeting and asked everyone’s comfortable area. All the members were good enough to handle any role but within limited time the skill becomes the main factor. So we arranged an election for team leader. My team voted me as team leader position. Then I divided the tasks according to the possible roles and arranged a quick interview with my team members. According to interview I note down the area of interest of my team members and skills on that area. I also get information about their availability if any sudden meeting needed to be called. My collected info is shortly given below:

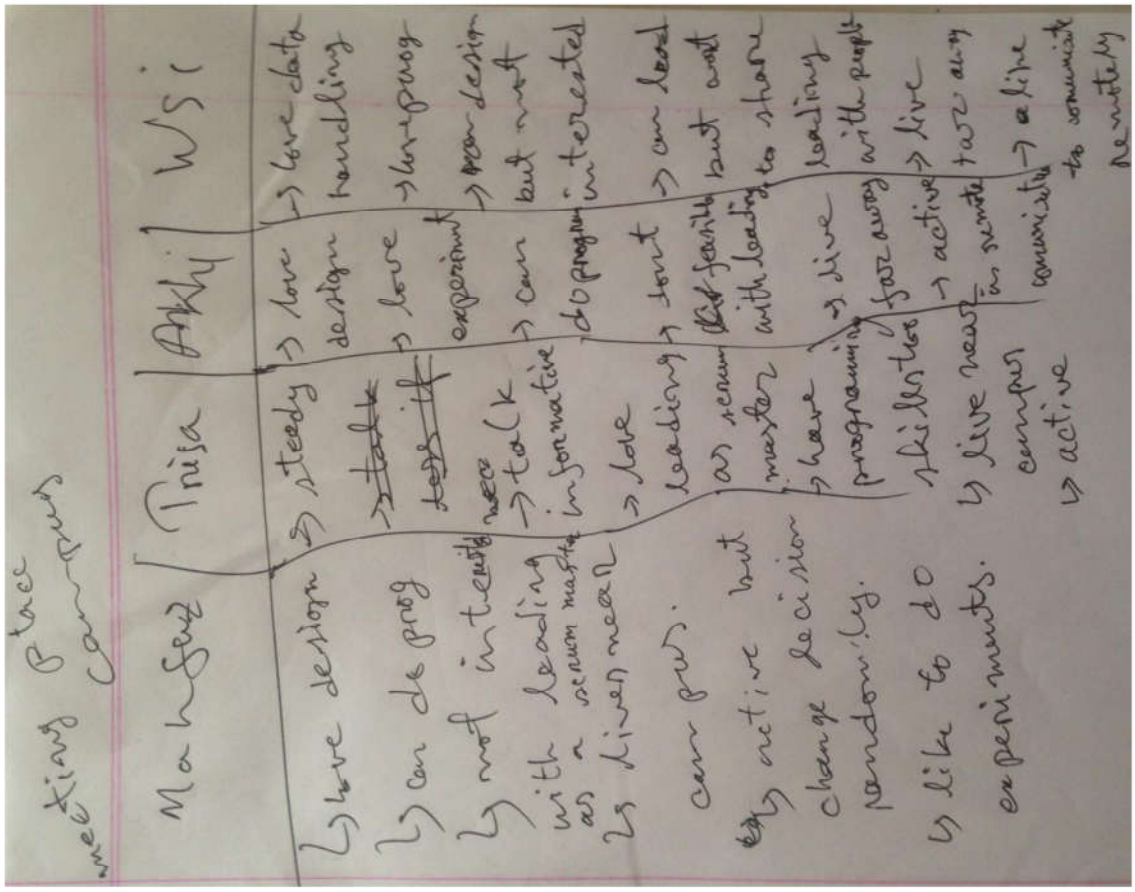


Figure 43: A rough copy of the team member interview to distribute the role

According all of those data I set a team model, get their feedback and I finalized the steam structure. The team model is described below:

Team model:

According to the feedback of all team members finalized version of role and comfort areas are given below:

S. M. Abdul Wassae:

As a team leader role played in group, arranged meeting, communicate with others, designed database, complete final implementation of product, manage repository, upload finished product on live server. Roles and comfortable areas are given below:

- Roles:
    - Team leader
    - Database designer
    - programmer
- Comfortable areas:
    - Database
    - Programming
    - Planning
    - arranging meeting Remotely
    - Learning and experimenting web technologies

Tabana Islam Trisha:

As an alternative team leader in group, made decision with others if leader get too busy with his task, arranged meeting, run the total scrum process and monitoring, split tasks in chunks according MoSCoW analysis and planned to order tasks and then assume accomplishment time, take notes of progress, compare them with estimated time, designed system architecture. Roles and comfortable areas are given below:

- Roles:
    - Alternative team leader
    - Scrum master
    - Product owner
    - Information architect
- Comfortable areas:
    - Designing architecture
    - Analyzing products
    - Planning

- Arranging any meetings
- Collecting meeting related notes

Mahfuzur Rahman bhuiyan:

As web designer in team, sketched the UI, converted them into low fidelity prototype, built high fidelity prototype and tested them on different devices and browsers. Roles and comfortable areas are given below:

- Roles:
  - Web designer
- Comfortable areas:
  - Sketching, Design and drawing layout
  - Designing and developing prototype
  - Experiment with new designs

Ankhi Akter:

As tester in group, made test plan, tested responsibility on devices, functionalities, generated error report, also made screencast and presentation. Roles and comfortable areas are given below:

- Roles:
  - Tester
  - Screen casting
  - Presentation document
- Comfortable areas:
  - Designing layout and components
  - Test planning and validation testing
  - Experiments

Methodologies:

We have used **Agile** methodology to complete the tasks of this project as a group work. Being various types of frameworks in agile methodologies we have chosen **scrum** for our teamwork. Basically scrum helps to break the whole work into some chunks. We have used iterative development to develop the system. At first we found out the important requirement. Then got meeting with the scrum master and product owner. After the agreement we created a product backlog or user story and start work following sprint back log. As well as following scrum we have different type of roles such as UI designer, system analyst, database designer, scrum master, programmer

and tester. We kept and used sticky notes and daily notes to compare our works. We all shares our files in Google repository and worked depend on the resources. The working process is given below:

Commentary:

In our team work, scrum master Trisha split the works according MoSCoW analysis. Then according to the functionalities I as database designer and Mahfuz as UI designer started according the functionalities. On the other hand, as Information architect Trisha drew the UML diagrams that would help to complete the final products and uploaded to the repository according functionality defined. UI designer designed interfaces depends on the ongoing functionalities and uploaded his designs to the repository and on a parallel contribution after full database designed I started completing functionalities as scrum master defined as task. After completing the functionalities I send it to Ankhi as tester to test any error needed to be fixed. If any error found she send me the screenshot and necessary solution on the repository. Then I tried to fix the problems within time and lopping the process until the problem solved and meet the goal. I took notes by date and of each process going on. After finishing a functional task I contact with scrum master to arrange a meeting and she call meeting to come physically in the meeting place but if not possible then we used remote meeting via Skype and TeamViewer. When the total product has developed in offline we arrange a final team meeting as closing. In closing session we discussed about our learning, further development, uploaded program file in live server and tested if any differences occurred after then Ankhi made a screencast and presentation. During while the finalized implemented program was going to test as working fine we documented the whole process as a note.

Marking

I have marked our team members depend on their availability, performance, contribution on requirements, cooperation with team, note taking about their task during meeting, skill in their area, stability on their decision, fulfilling sprint on time, repository activity and work reflection.

Criteria	S.M. Abdul Wassae	Mahfuzur Rahman bhuiyan	Tabana Islam Trisha	Ankhi Akter
Availability	10	10	10	10
Performance	10	9	10	9
Contribution on requirements	8	8	8	8
Team cooperation	10	10	10	8
Note taking	9	9	9	8
Skill in his areas	9	9	9	9
Stability on decision	10	6	10	9
Punctuality on sprints	9	8	10	9
Maintaining repository	10	10	8	10
Work reflection	10	10	10	10
Total =100	95	89	94	90

Figure 44: marking performance of group members in scale of 100

Result:

The result compare to 10 is given below:

Member Name	Mark	Commitment
S.M. Abdul Wassae	9.5	Fully committed
Mahfuzur Rahman bhuiyan	8.9	Fully committed
Tabana Islam Trisha	9.4	Fully committed
Ankhi Akter	9	Fully committed

Figure 45: result out of 10

Self-Evaluation

This is my first team project where I have followed the team leader role. To arrange such a group work I have learnt a lot of things. I realized the difference between communication skill and personal skill. As a have learnt how to deal with other team members with different roles. I also learnt about how to inherit leadership to other group member. I was good with programming and data handling. Perhaps I have very new experiences when I worked with team. Details are given below:

Own contribution

As a team leader, database designer and role played in group I have done many tasks in our team. They are given below:

- Arrange meeting,
- Communicate with others,
- Design database,
- Apply Normalization
- Draw clear ERD
- Create view to show reports
- Generate triggers to atomized
- Implement functionalities
- Ensure functionalities work properly on live server
- Complete final implementation of product,
- Manage repository,
- Share my contributions to the repository so that other members can get help
- Upload finished product on live server.

Reflection of own performance (impact)

I have done many small projects myself but as an enterprise web solution development it was my first team project. I tried to manage my team members with my best efforts. Sometimes I was gotten too busy with my program that I could not attend meeting physically. So I successfully handover the responsibility to alternative team leader by requesting him to arrange meeting remotely connect with me. The time was too much limited to play multiple roles on such kind of system where dependencies on other member were badly needed. But with proper maintenance of tasks and arranged as a team contribution the program successfully completed its all functionalities. I have theoretically known about agile but by following scrum framework I practically experienced through agile methodology.

Normalization of the database wasn't so tough but complex for this solution. Using trigger and view was my first time experiments in the solution and I believe that next time I could apply them more comfortably in less time for any solutions. I had knowledge about PDO but first time I have used them with MVC framework as a web solution. File uploading and email notification was much easy but took much time rather than my expectation. Percentage report generating on pie from data base using jQuery was much enjoyable. I think next time in I will use postgresql instead of Mysql for larger data handling.

Lesson learnt

As being told before that it is my first team work for enterprise web solution development. To arrange such a group work I have learnt a lot of things as given below:

- Distributing roles for group members based on their skills, behavior and availability.
- Continuous communication
- Never feel shy to ask help
- Implementation of PDO with MVC pattern
- Improvement of personal skill.
- How to use repository for team work
- How a sudden change from any role can impact the development process.
- How to deal with other team members with different roles.
- How to inherit leadership to other group member.
- How to use trigger for atomization
- How to use view as database security purposes
- How to use view for generating reports
- How to generate email notifications.
- Documenting about a product completed as a group
- Note taking of every works
- Usage of agile methodologies
- Implementation of the product to the live server.
- Finally I have learnt that team work is much better than being solo but neve leave team without a leader if get busy. At least setting co-leader is necessary.

Conclusion

Extenuating Circumstances is an enterprise web solution which has to be completed within limited time. So we followed scrum framework of agile methodology as a group project. It is the first version of our product and it has fully met all the functional requirements. It also has some limitations in different places which couldn't be made ok for the time limitations. However within this limited time duration, our team shows their best effort and always active till the finalization of product. Moreover we all learn many things such as team working that not only help in further development of the product's next version but also be helpful for any kind of team project in future.

References and Bibliography

bootstrap, [Online], Available: https://www.upwork.com/hiring/development/bootstrap-3-front-end-framework-responsive-mobile-first-sites/.

css3, [Online], Available: https://courses.telegraph.co.uk/article-details/181/what-is-css3/.

html5, [Online], Available: https://code.tutsplus.com/tutorials/what-is-html5--cms-25803.

jQuery, [Online], Available: https://jquery.com/.

js, [Online], Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Introduction.

MVC, [Online], Available: http://whatis.techtarget.com/definition/model-view-controller-MVC.

mysql, [Online], Available: https://www.mysql.com/about/.

pdo, [Online], Available: https://www.drupal.org/docs/7/system-requirements/what-is-pdo.

php7, [Online], Available: https://www.tutorialspoint.com/php7/php7_introduction.htm.

Appendix A: Security

The term security in database contains a process which ensures “CIA”. The CIA represents confidentiality, Integrity and Accessibility. For confidentiality enums and unique keys are used. To maintain integrity primary key and foreign keys are declared. Accessibility is maintained in config.php file where host, username, password and database name have to be defined. Besides view are implemented for secure accessibility too.

Using Primary key, Foreign key, and Unique Key:

Primary key: It is a key as a unique identifier of each record in a table in relational database. This key cannot be duplicated or contain null.

Foreign Key: It is a field of a table in a relational database where a unique identifier of another table or same table comes to make relation. This key can be duplicated or contains null.

Unique key: It is a key that confirms all the values of a column in a table is different from each other. As like as Primary key unique key cannot be duplicated or null.

The attributes which are used as primary key, foreign key, and unique key are used in database are given below with their containing tables and reference table.

Table Name	Attribute Name	Key Name	Reference table
User	Uid	Primary	
	Uname	Unique	
	Email	Unique	
Student	Sid	Primary	
	Uid	Foreign	user
	Did	Foreign	department
facultycoord	Fid	Primary	
	Uid	Foreign	user

department	Did	Primary	
	Dname	Unique	
	Fid	Foreign	facultycoord
Course	Coursed	Primary	
	Did	Foreign	department
complain	cid	Primary	
	Coursed	Foreign	course
subjectcomplain	Scid	Primary	
	cid	Foreign	complain
	Sid	Foreign	student
evidence	Eid	Primary	
	Scid	Foreign	subjectcomplain
Solution	Solid	Primary	
	scid	Foreign	subjectcomplain
	fid	Foreign	facultycoord

Figure 46: table that shows attribute that are used as primary key, foreign key, unique key

Using Enums:

Enum is an object type data type which contains a defined list of strings.

Table Name	Attribute Name	Values
User	ucatagori	student, administrator, coordinator, manager
subjectcomplain	status	solved, unsolved, denied
evidence	etype	jpeg, jpg, png, gif, pdf

Figure 47: use of enaums

Using View:

View is a virtual table based on the result-set of the sql query. It can be used as a replica of original table.

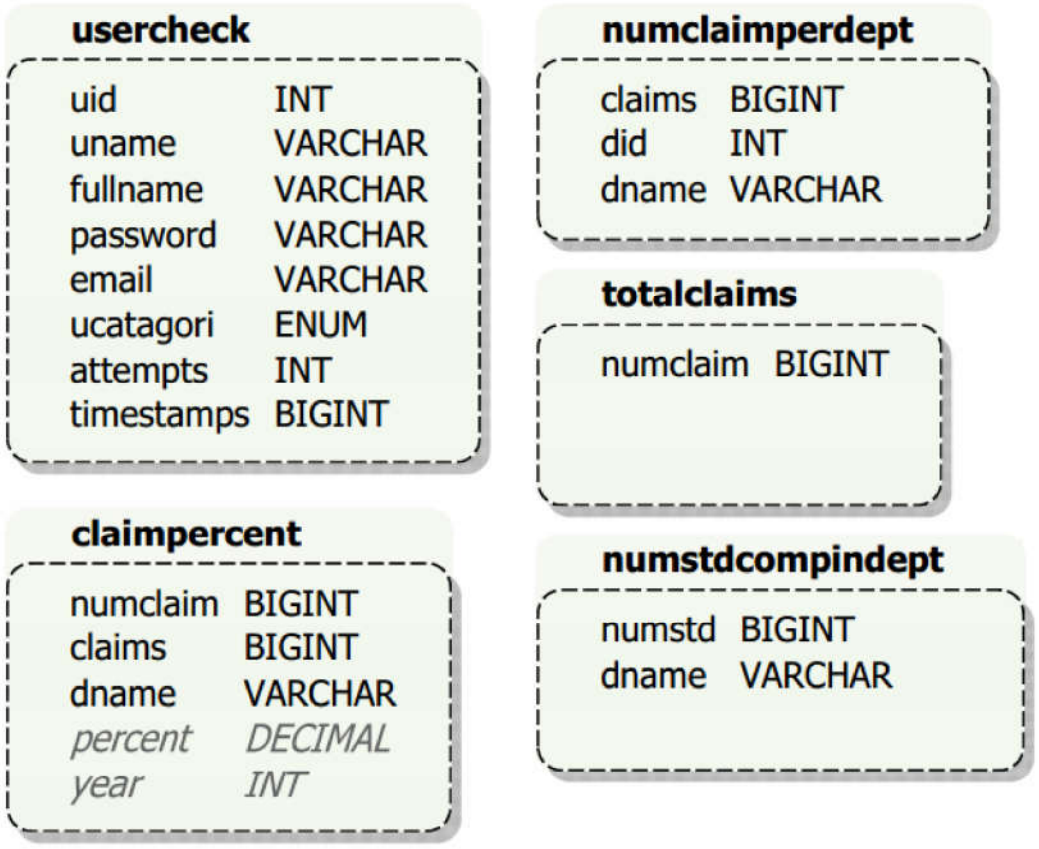


Figure 48: Use of view

Appendix A: appropriate data types and validation

Appropriate data types and validations used for get entry and representation of proper data. It ensures secure data entry via data type and data length. It also checks the data with relational integrities and data which are predefined as domain names or enums.

Data dictionary and Metadata

Data dictionary is basically a structure that contains metadata and metadata is such kind of data which represents the properties of the data.

Tables

Metadata of each table from data dictionary is given below:

User table: it contains user information.

User								
Attribute name	**Data type**	**Length**	**Default**	**NULL**	**Unique key**	**Primary key**	**Foreign key**	**Reference**
Uid	int	11		No	Yes	Yes		
Uname	varchar	250		No	Yes			
fullname	varchar	250		No				
password	varchar	50		No				
email	varchar	250		No	Yes			
ucatagori	enum	student, administrator, coordinator, manager		No				
attempts	int	11	0	No				
timestamps	bigint	20	0	No				

Figure 49: user table

Facultycoord table: it contains only the coordinator list from user table

facultycoord								
Attribute name	**Data type**	**Length**	**Default**	**NULL**	**Unique key**	**Primary key**	**Foreign key**	**Reference**
fid	int	11		No	Yes	Yes		
uid	int	11		No	Yes		Yes	user=> uid

Figure 50: facultycoord table

Department table: it contains department information. One coordinator must have to stay in department but one coordinator can manage multiple departments.

department								
Attribute name	**Data type**	**Length**	**Default**	**NULL**	**Unique key**	**Primary key**	**Foreign key**	**Reference**
did	int	11		No	Yes	Yes		
dname	varchar	250		No	Yes			
fid	int	11		Yes			Yes	facultycoord=> fid

Figure 51: department table

Student table: it separates students from user table. Student must be the member of department.

student								
Attribute name	**Data type**	**Length**	**Default**	**NULL**	**Unique key**	**Primary key**	**Foreign key**	**Reference**
sid	int	11		No	Yes	Yes		
did	int	11		No	Yes		Yes	department=> did
uid	int	11		No	Yes		Yes	user=> uid

Figure 52: student table

Course table: it contains course information which must be under department.

course								
Attribute name	**Data type**	**Length**	**Default**	**NULL**	**Unique key**	**Primary key**	**Foreign key**	**Reference**
courseid	int	11		No	Yes	Yes		
coursename	varchar	250		No				
did	int	11		No			Yes	department=> did

Figure 53: course table

Complain table: this table contains complain subject information under a course.

complain								
Attribute name	**Data type**	**Length**	**Default**	**NULL**	**Unique key**	**Primary key**	**Foreign key**	**Reference**
cid	int	11		No	Yes	Yes		
cname	varchar	250		No				
cdetails	text			No				
duedate	datetime			No				
finaldate	datetime			No				
courseid	int	11		No			Yes	course=> courseid

Figure 54: complain table

Subjectcomplain table: this table holds claims under a complain subject from a student.

subjectcomplain								
Attribute name	**Data type**	**Length**	**Default**	**NULL**	**Unique key**	**Primary key**	**Foreign key**	**Reference**
scid	int	11		No	Yes	Yes		
sctitle	varchar	250		No				
scdetails	text			No				
submissiontime	datetime			No				
replyabledate	datetime			No				
numevidence	int	11		No				
status	enum	solved, unsolved, denied	unsolved	No				
sid	int	11		No			Yes	student=> sid
courseid	int	11		No			Yes	complain=> cid

Figure 55: subjectcomplain table

Evidence table: this table contains evidence information of a claim submitted by student.

evidence								
Attribute name	**Data type**	**Length**	**Default**	**NULL**	**Unique key**	**Primary key**	**Foreign key**	**Reference**
eid	int	11		No	Yes	Yes		
ename	varchar	250		No				
Etype	enum	jpeg, jpg, png, gif, pdf		Yes				
eupdate	datetime			No				
scid	int	11		No			Yes	subjectcomplain=> scid

Figure 56: evidence table

Solution table: it is used for hold information related solution of a claim provided by a coordinator.

solution								
Attribute name	**Data type**	**Length**	**Default**	**NULL**	**Unique key**	**Primary key**	**Foreign key**	**Reference**
solid	int	11		No	Yes	Yes		
solvedate	datetime	250		No				
solution	longtext			Yes				
Scid	int	11		No			Yes	subjectcomplain=> scid
fid	int	11		No			Yes	facultycoord=> fid

Figure 57: solution table

Views

Metadata of each view from data dictionary is given below:

Usercheck: this view is used for a security purposes which creates replica of user table.

usercheck						
Attribute name	**Data type**	**Length**	**Default**	**NULL**	**Unique key**	**Reference**
uid	int	11		No	Yes	user=> uid
uname	varchar	250		No	Yes	user=> uname
fullname	varchar	250		No		user=> fullname
password	varchar	50		No		user=> password
email	varchar	250		No	Yes	user=> email
ucatagori	enum	student, administrator, coordinator, manager		No		user=> ucatagori
attempts	int	11	0	No		user=> attempts
timestamps	bigint	20	0	No		user=> timestamps

Figure 58: usercheck view

Totalclaim: it counts and represents total number of claims submitted in the system.

totalclaim						
Attribute name	**Data type**	**Length**	**Default**	**NULL**	**Unique key**	**Reference**
numclaims	bigint	21		No		

Figure 59: totalclaim view

Numstdcomplaindept: it count, store and represent the number of student submitted claim for each department.

numstdcomplaindept						
Attribute name	**Data type**	**Length**	**Default**	**NULL**	**Unique key**	**Reference**
numstd	bigint	21		No		
dname	varchar	250		No		department=> dname

Figure 60 numstdcomplainperdept view

Numclaimperdept: this view counts, store and process the number of claims in each department.

numclaimperdept						
Attribute name	**Data type**	**Length**	**Default**	**NULL**	**Unique key**	**Reference**
claims	bigint	21	0	No		
did	int	11		No		department=> did
dname	varchar	250		No		department=> dname

Figure 61: numclaimperdept view

Claimpercent: this view used for collect total number of claims from **totalclaim** and make department wise percentage to show a yearly report.

claimpercent						
Attribute name	**Data type**	**Length**	**Default**	**NULL**	**Unique key**	**Reference**
numclaims	bigint	21	0	No		totalclaim=> numclaims
claims	int	11		No		department=> did
dname	varchar	250		No		department=> dname
percent	decimal	(27,4)		Yes		
year	int	11		Yes		

Figure 62: claimpercent view

Trigger

Trigger is a store procedure that usually used for run IUD (Insert, Update, and Delete) operation in a table on an event called for a table component.

countevidence:

This trigger count the number of evidence if any evidence is inserted into evidence table and then updates the count result in numevidence column of subjectcomplain table.

```
-- ----------------------------
-- Trigger structure for countevidence
-- ----------------------------
DELIMITER ;;
CREATE TRIGGER `countevidence`
AFTER INSERT ON `evidence`
FOR EACH ROW update subjectcomplain,
 (SELECT COUNT(eid) as numevidence, evidence.scid as escid
 FROM subjectcomplain,evidence
 WHERE subjectcomplain.scid=evidence.scid GROUP BY escid) as countevidednce
        set subjectcomplain.numevidence= countevidednce.numevidence
        where subjectcomplain.scid=escid;;
DELIMITER ;
```

Figure 63: trigger to count and store count result

acceptsolve:

This trigger update the claim status as solved if any solution of that claim provided.

```
-- ----------------------------
-- Trigger structure for accptsolve
-- ----------------------------
DELIMITER ;;
CREATE TRIGGER `accptsolve`
AFTER INSERT ON `solution`
FOR EACH ROW UPDATE subjectcomplain
set STATUS='solved'
WHERE scid=new.scid;;
DELIMITER ;
```

Figure 64: trigger to change status to 'solved' if solution provided

Appendix A: ERD

ERD means entity relationship model. From the breakdown of the functionalities of scenario using natural language analysis (NLA) a general simple rough ERD is drawn. Then attributes are set on a Clear detail ERD including the views. The process is given below:

Using NLA

From the functionalities of the scenario, nouns and adjectives are sorted out. Then repetitive groups are removed. After removing repetitive groups synonyms are removed. Remaining nouns are possible entity and adjectives are the attributes. Then a rough copy of ERD is drawn with relationship and reviewed for optimization. Then another rough is drawn. After doing the process iteratively several times final rough of ERD is drawn that is given below:

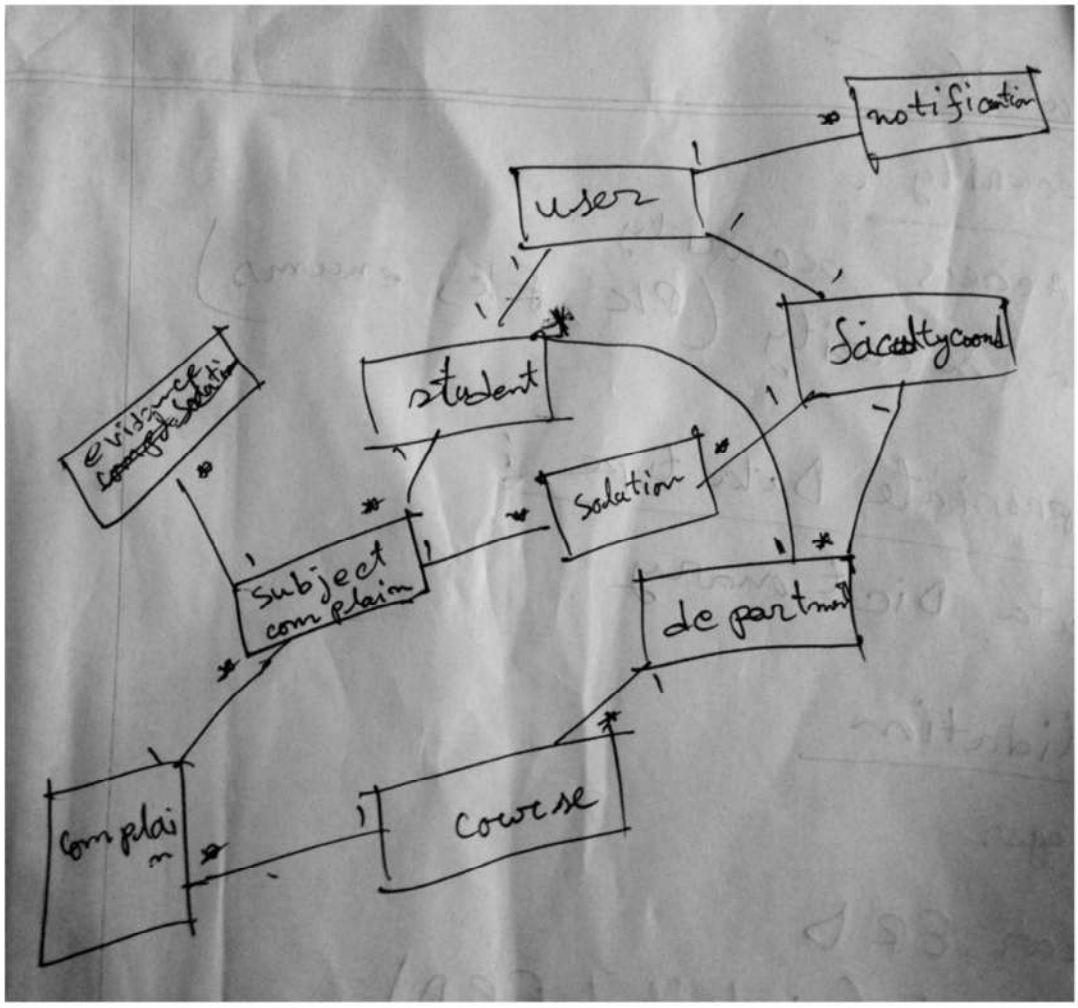


Figure 65: rough of assumed ERD from NLA

Clear ERD with attributes:

From the final rough ERD after last optimization preview run a clear ERD with attribute is drawn as given below:

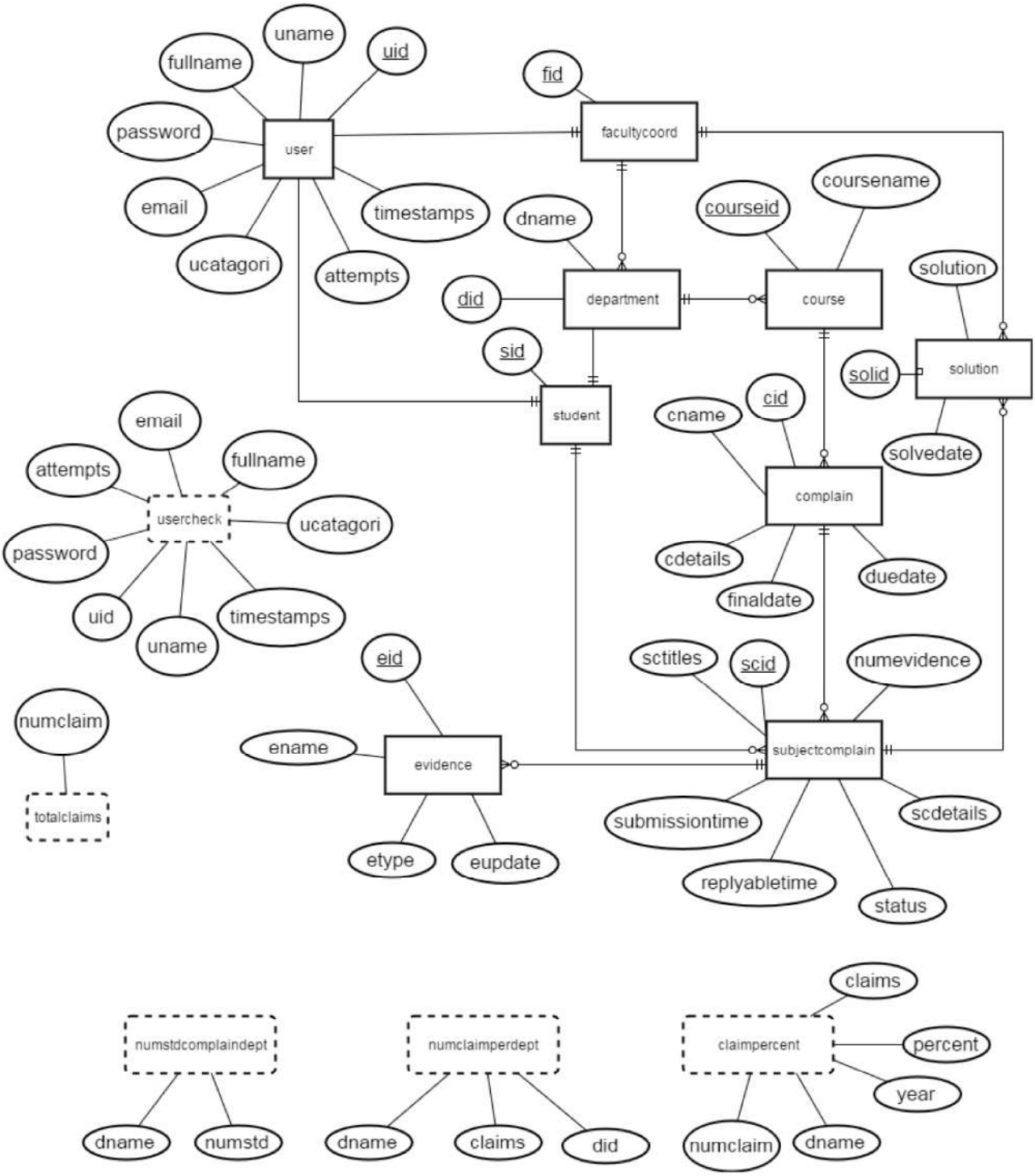


Figure 66: A clear ERD including attributes

Appendix A: referential integrity implemented

Relational integrity: referential integrity represents the relation among tables using reference of key in a word **foreign** key. To find out the referential integrity a 3rd NF (Normalization Form) of normalization of the collected attributes are processed. Then the detail ERD representing referential integrity is drawn. The normalization process and final ERD containing referential integrity is given below:

Normalization:

To normalize the data at first candidate keys are selected. Among the attribute list '**did**' and '**cid**' is the possible candidate key. According to the clear ERD given above, '**cid**' have get most priority to be the candidate key. Then comparing multiplicity of other attributes with the candidate key level are set as 1 and 2 represent one and many. In addition to next step the 1st NF; attributes are separated with same level in a side holding the candidate key to ensure no repeating group. After that, the 2nd NF identified the partial key dependencies. Finally in 3rd NF any key dependencies are checked Normalization process up to 3rd NF is given below:

UNF	level	1st NF	2nd NF	3rd NF
uid	1	**scid**	**scid**	user
uname	1	uid	uid	uid uname fullname password email ucatagori attempts timestamps
fullname	1	uname	uname	
password	1	fullname	fullname	
email	1	password	password	
ucatagori	1	email	email	
attempts	1	ucatagori	ucatagori	
timestamps	1	attempts	attempts	facultycoord
sid	1	timestamps	timestamps	fid uid*
fid	1	sid	sid	
did	1	fid	fid	department
dname	1	dname	dname	Did dname fid*
courseid	1	courseid	courseid	
coursename	1	coursename	coursename	Student
cid	1		cid	Sid did* uid*
cname	1	**scid**	cname	
cdetails	1	sctitles	cdetails	
duedate	1	scdetails	duedate	Course
finaldate	1	submissiontime	finaldate	Coursed coursename did*
scid	2	replyabledate		
sctitles	2	numevidence	**scid**	Complain
scdetails	2	status	sctitles	cid cname cdetails duedate finaldate coursed*
submissiontime	2	eid	scdetails	
replyabledate	2	ename	submissiontime	
numevidence	2	etype	replyabledate	Subjectcomplain
status	2	eupdate	numevidence	scid sctitles scdetails submissiontime replayabledate numevidence status courseid* sid*
eid	2	solid	status	
ename	2	solution		
etype	2	solvedate	**scid**	
eupdate	2		eid	
solid	2		ename	
solution	2		etype	evidence
solvedate	2		eupdate	eid ename etype eupdate scid*
			scid	
			solid	Solution
			solution	solid solution solvedate scid* fid*
			solvedate	

Figure 67: normalization process up to 3rd NF

Referential integrity with their located table name is given below:

Attribute Name	Table Name	Reference (table=> attribute)
cid	subjectcomplain	complain=> cid
courseid	complain	course=> coursed
did	course	department=> did
	student	department=> did
fid	department	facultycoord=> fid
	solution	facultycoord=> fid
scid	solution	subjectcomplain=> scid
	evidence	subjectcomplain=> scid
sid	subjectcomplain	student=> sid
uid	facultycoord	user=>uid
	student	user=>uid

Figure 68: referential integrity at a glance

Final ERD:

From the normalization process final ERD with referential integrity is drawn below:

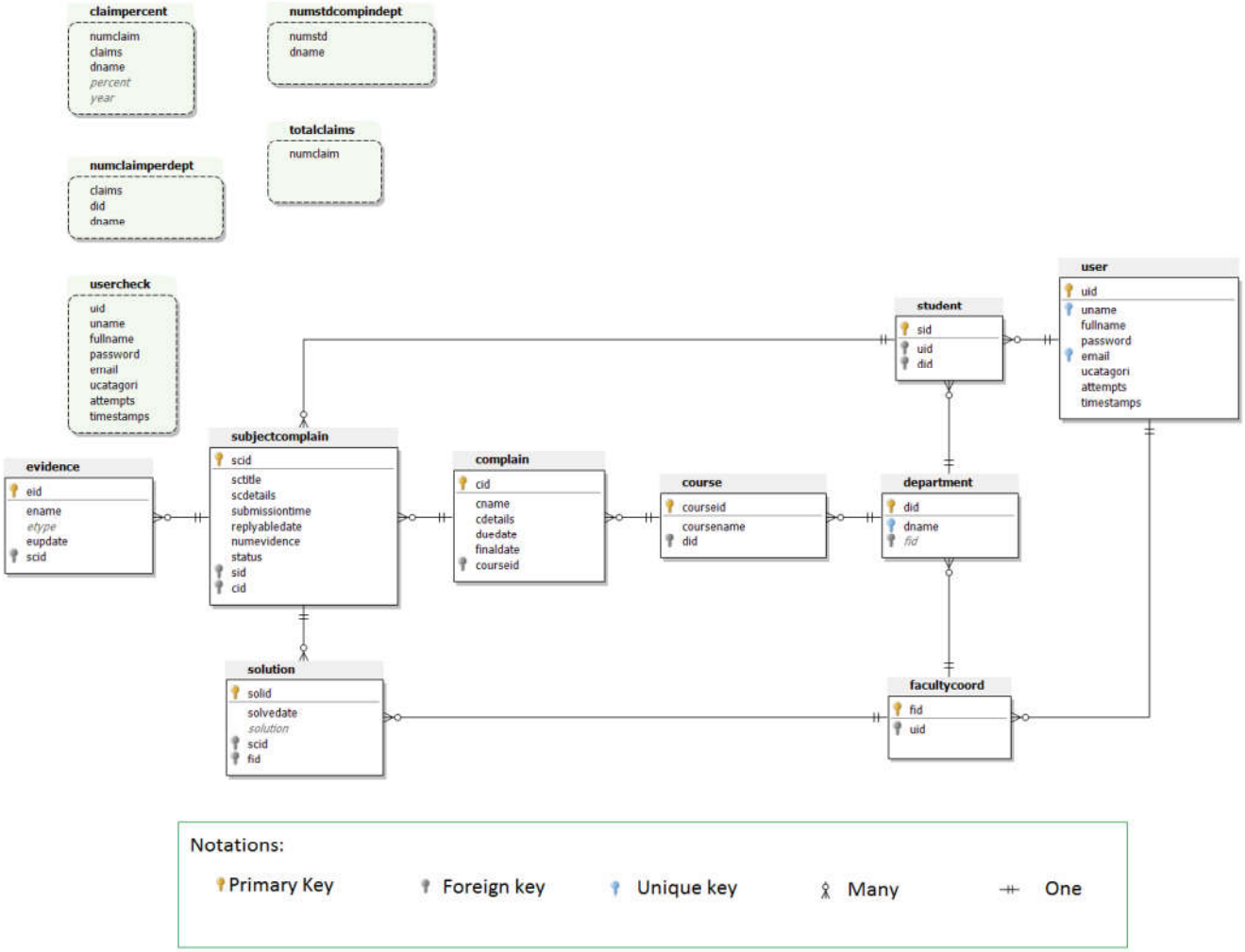


Figure 69: final ERD with relational integrity got from normalization

Appendix A: enables roles to be implemented

In the database users are defined as category to implement the roles. The catagori column of user table contains an enum where values are defined as administrator, manager, coordinator and student.

uid	uname	fullname	password	ucatagori
1	admin	Tag Jin	1234	administrator
2	stdsyn	Syn Bade	1234	student
3	coormaan	Keher Maan	1234	coordinator
4	mnglaila	Alif Laila	1234	manager
5	stdsofan	Sofan Isbah	1234	student
6	corkala	Ja kala	1234	coordinator

Figure 70: user catagory as roles

Then student and coordinator get separated to maintain their confidentiality. facultycoord table represent coordinator from user table.

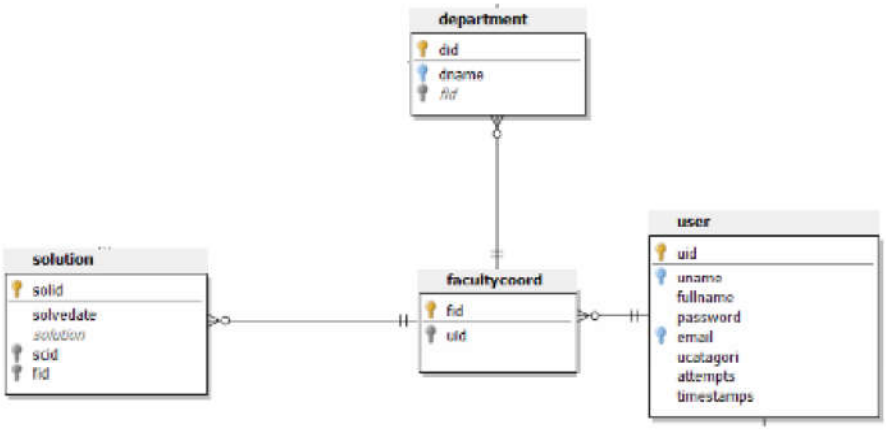


Figure 71: coordinator role in ERD

Similarly student table represent student role

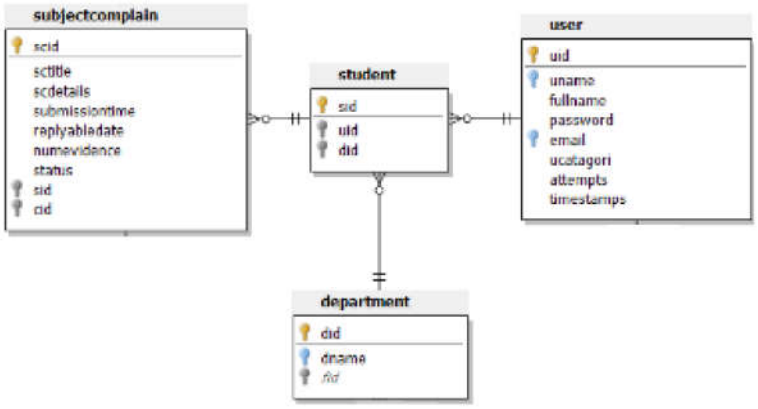


Figure 72: student role implemented in ERD

For manager different kind of reports needed to be shown. So views are created to show report.

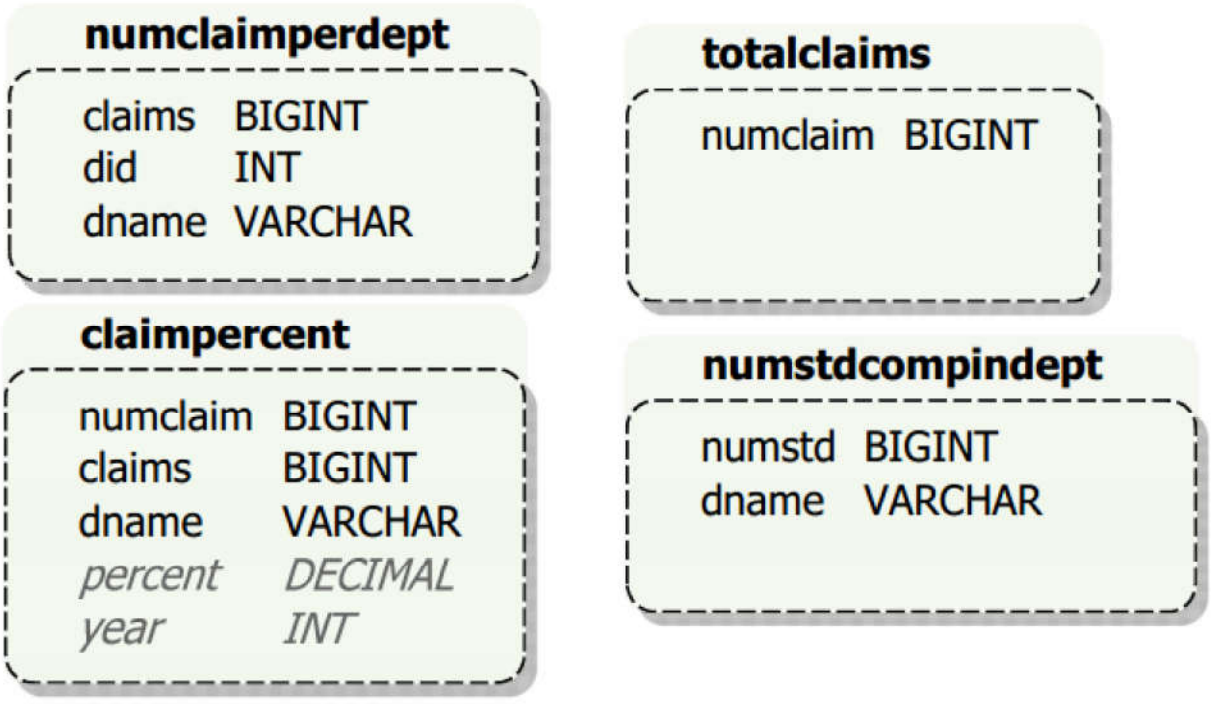


Figure 73: viwes to show report to manager

Administrator can access any system data. So his role is not created separately from user table.

Appendix B: submission of reports

The reports are mainly shown to manager. But some reports are shown to coordinator too. Reports of the developed solution are given below:

Number of claims within each faculty

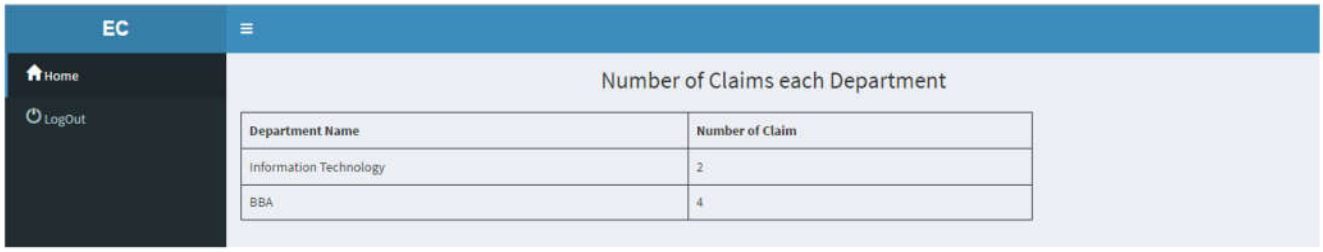


Figure 74: Number of claims within each Faculty for each academic year

Number of student claimed each department

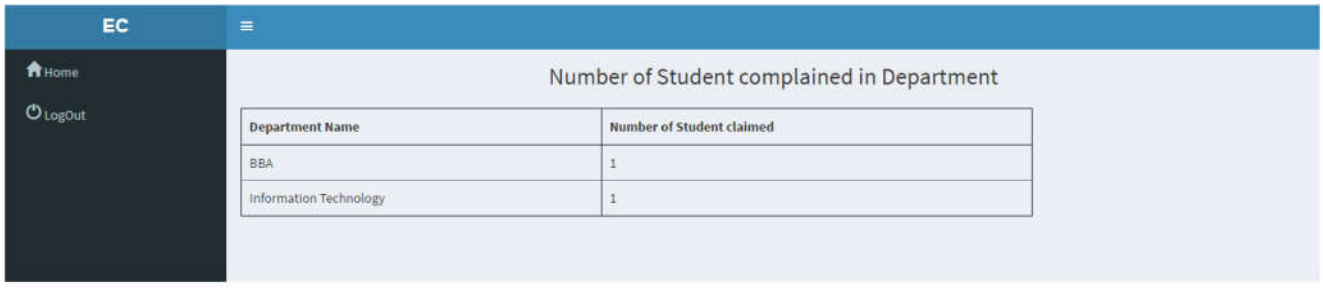


Figure 75: Number of students making a claim within each Faculty for each academic year

Appendix B: email notification

After a successful claim an email notification would send to EC coordinator email address from the student email address. Coordinator email notification screenshot is given below:

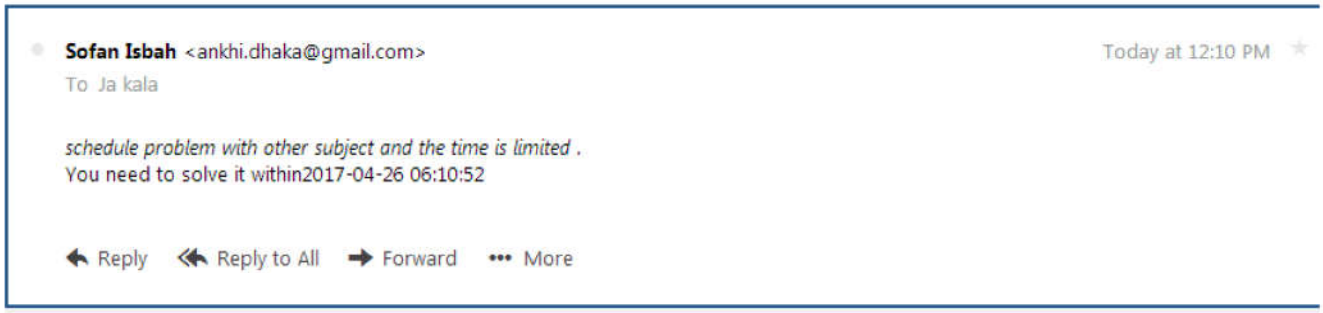


Figure 76: coordinator email notification

If the coordinator accept claim with providing solution, the solution will also be sent as notification to the student email address from coordinator email address.

Student email notification screenshot is given below:

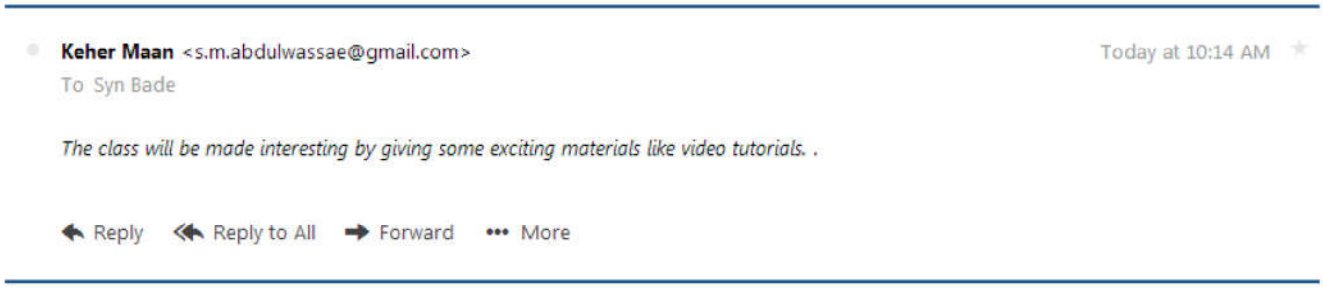


Figure 77: student email notification

Appendix B: summary and exception reports

Summary report and exception reports according to the scenario are given below:

Summary report:

Overall summary report to the manager with chart is given below

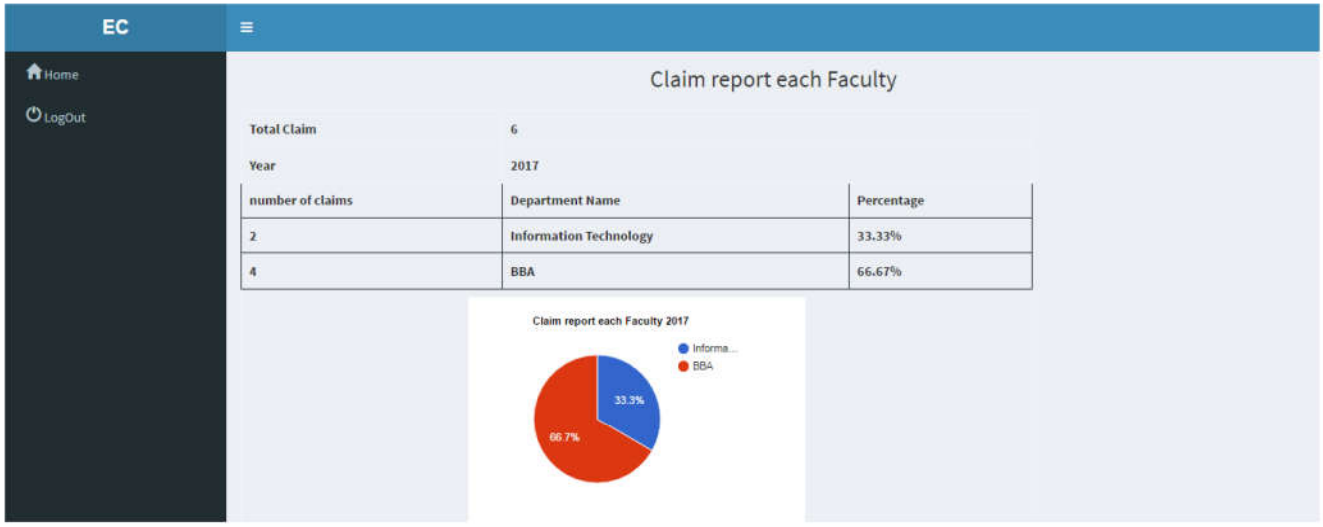


Figure 78: Percentage of claims by each Faculty for any academic year

Exception reports

The exception reports are given below:

Claims without evidence

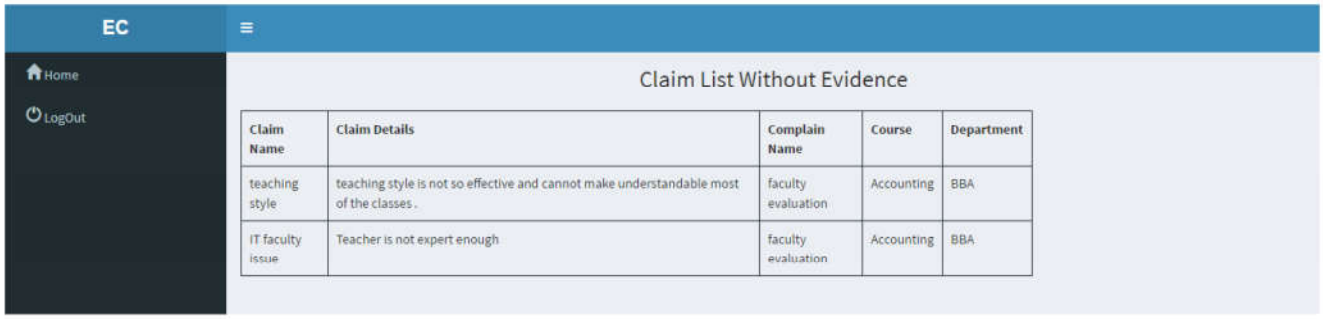


Figure 79: Claims without uploaded evidence

Claim that doesn't get reply within 14 days

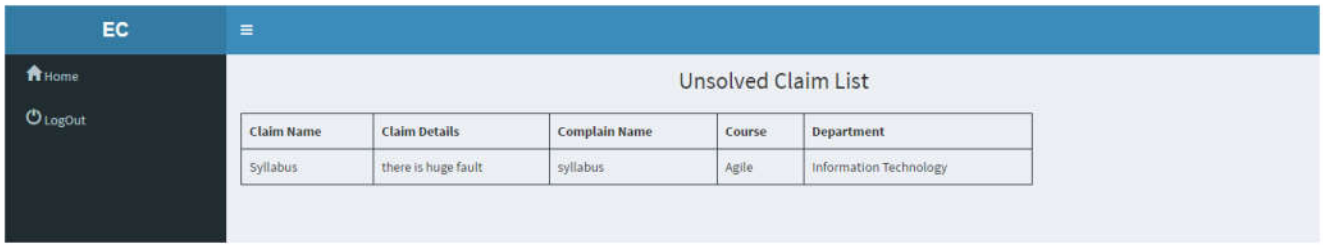


Figure 80: Claims without a decision after 14 days

Appendix B: code snippets

The code snippets of the main parts of the key functionalities are given below:

Check user role during login:

To login the system as a role, user just enters his/her username or email and password. If the data matched the system check the role and automatically redirect to that role's home page.

```
if ($tempUser->getPass() == $password) {
    if ($tempUser->getTimestamps() > time()) {
        $err = "timeout";
        header("location:../view/error.php?err=" . $err);
          echo User::$message;
        header("location:../view/error.php?err=" . $err);
    } else {
        $sql = "update usercheck set attempts='0' where uname='".$username."'";
        update($sql);
        setcookie("loggin", "true", time() + 3600, "/");
          echo $_COOKIE["loggin"];
        $_SESSION["username"] = $tempUser->getUname();
        $_SESSION["email"] = $tempUser->getEmail();
        $_SESSION["catagory"] = $tempUser->getCatagory();
          print_r($_SESSION);
          print_r($_COOKIE);
        if ($_SESSION["catagory"] == "administrator") {
            header("location:../view/pages/admin/admin.php");
        } elseif ($_SESSION["catagory"] == "manager") {
            header("location:../view/pages/manager/manager.php");
        } elseif ($_SESSION["catagory"] == "coordinator") {
            header("location:../view/pages/coordinator/coordinator.php");
        } elseif ($_SESSION["catagory"] == "student") {
            header("location:../view/pages/student/student.php");
        }
    }
} else {
```

Figure 81: role checking code snippet

Add course:

To add a complain subject, administrator need to add course. Following code is working behind add course.

```php
<?php
include '../model/dataHandler.php';
if (isset($_GET)) {
    $deptId = $_GET["dept"];
    $courseName = $_GET["course"];
    $sql = "insert into course(coursename,did) values ('".$courseName."','" . $deptId . "')";
//    echo $sql;
    insertInDatabase($sql);
    header("location:../view/pages/admin/addCourse.php?msg=1 &err=0");
} else {
        header("location:../view/pages/admin/addCourse.php");
}
```

Figure 82: code for add course

View departments:

To add course admin need to view his department list. The code given below will allow him/her to see the list of department.

```php
    <!-- Main content -->
    <div class=" col-md-9 col-lg-9">
        <table class="table table-user-information" >
            <thead>
                <tr><th>Departments</th></tr>
            </thead>
            <tbody>
                <?php
                $sql = "select dname from department order by dname asc";
                $department = showData($sql);
//                print_r($result);
                foreach ($department as $row) {
                    ?>
                    <tr>
                        <td style="border: 1px solid #222d32"><?php echo $row["dname"]; ?></td>
                    </tr><?php } ?>
            </tbody></table>
    </div>
    <!-- /.content -->
```

Figure 83: code to show department list

Change password:

To change password for a user admin need to fill a form. This is a code snippet of view part that shows admin the user list to change password.

```
<tr>
    <td><b><?php
            if ($_GET) {
                $msg = $_GET["msg"];
                if ($msg == 1) {
                    echo "<span class='text-success'>Password changed successful</span>";
                }
            }
            ?></b>
    </td>
</tr>
<tr>
    <td><span class="style19"><b>User Full Name:</b></span></td>
    <td><select name="user">
            <?php
            $sql = "select fullname,uname from usercheck order by fullname asc";
            $result = showData($sql);
            // print_r($result);
            foreach ($result as $row) {
                echo "<option value='" . $row['uname'] . "'>" . $row['fullname'] . " (" . $row['uname'] . ")</option>";
            }
            ?"
        </select>
    </td>
</tr>
```

Figure 84: user list to change password

After selecting a user admin will submit the new password and following code will run to store new password.

```
<?php

include '../model/dataHandler.php';
include '../model/User.php';

if (isset($_POST)) {
    $user = $_POST["user"];
    $chngedpass = $_POST["chngedpass"];
//    echo $user ." , ".$chngedpass;
      $sql = "update usercheck set password='" . $chngedpass . "' where uname='" . $user . "'";
//      echo $sql;
    update($sql);
    header("location:../view/pages/admin/chngpass.php?msg=1 &err=0");

}
```

Figure 85: code to store new password

Add complain subject:

The code shown below will work behind the new entry of complain subject under a course.

```php
<?php

include '../model/dataHandler.php';
if (isset($_GET)) {
    $did = $_GET["did"];
    $cname = $_GET["comp"];
    $cdetails = $_GET["cdetails"];
    $duedate = $_GET["duedate"];
    $finaldate = $_GET["finaldate"];
    $sql = "insert ignore into complain(cname,cdetails,duedate,finaldate,courseid) values "
            . "('" . $cname . "','".$cdetails."','".$duedate."','".$finaldate."','".$did."')";
//    echo $sql;
    insertInDatabase($sql);
    header("location:../view/pages/admin/addComplain.php?msg=1 &err=0");
} else {
        header("location:../view/pages/admin/addComplain.php");
}
```

Figure 86: set complain subject under course

Add claim:

Student claim will store in database using the code given below.

```php
<?php
require '../config/PHPMailer-master/PHPMailerAutoload.php';
include '../model/dataHandler.php';
if (isset($_GET)) {
    $sid = $_GET["sid"];
    $cid = $_GET["cid"];
    $sctitle = $_GET["sctitle"];
    $scdetails = $_GET["scdetails"];
    $submissiontime = $_GET["submissiontime"];
    $replyabledate = $_GET["replyabledate"];
    $sql = "insert ignore into subjectcomplain(sctitle,scdetails,submissiontime,replyabledate,sid,cid) values "
            . "('" . $sctitle . "','" . $scdetails . "','" . $submissiontime . "','" . $replyabledate . "','" . $sid . "','" . $cid . "')";
//  echo $cid;
    insertInDatabase($sql);
```

Figure 87: code for add a claim

Edit claim:

The following code is needed to be executed to change the claim description.

```php
<?php

session_start();
include '../model/dataHandler.php';
if (isset($_POST)) {
    $scid = $_POST["scid"];
    $scdetails = $_POST["scdetails"];

    $sql = "update subjectcomplain set scdetails='" . $scdetails . "' where scid='" . $scid."'";
    update($sql);
//   echo $sql;
    header("location:../view/pages/student/myclaim.php");
}
```

Figure 88: code for edit claim details

Evidence uploads:

To upload a valid evidence following code needs to be used.

```php
session_start();
include '../model/dataHandler.php';
include '../model/Evidence.php';
if (isset($_POST)) {
    $scid = $_POST["scid"];
    $target_dir = "../evidence/";
    $ename = $_FILES['ename']['name'];
    $esize = $_FILES['ename']['name'];
    $up_file = basename($_FILES["ename"]["name"]);
    $etype = pathinfo($up_file, PATHINFO_EXTENSION);
    $sql = "select max(eid) as eid from evidence";
    $result = showData($sql);
    $eid = $result[0]["eid"] + 1;
//    print_r($_FILES);
//    echo $eid;
    $euptime = date('Y/m/d h:i:s', time());
    // echo $euptime;
    $target_file = $target_dir . $eid . "." . $etype;
    $evi = new Evidence();
    $evi->setEid($eid);
    $evi->setEname($ename);
    $evi->setEsize($esize);
    $evi->setEtype($etype);
    $evi->setEupdate($euptime);

    if ($evi->getEtype() == "jpg" || $evi->getEtype() == "jpeg" || $evi->getEtype() == "png"
    || $evi->getEtype() == "gif" || $evi->getEtype() == "pdf") {
        if ($evi->getEsize() < 50000000) {
            $sql = "insert into evidence(eid,ename,etype,eupdate,scid)"
                    . "values(" . $eid . ",'" . $ename . "','" . $etype . "','" . $euptime . "','" . $scid . "')";
            // echo $sql;
            insertInDatabase($sql);
            move_uploaded_file($_FILES["ename"]["tmp_name"], $target_file);
            header("location:../view/pages/student/myclaim.php");
        }
    } else {
```

Figure 89: code for upload evidence

Email notification:

Email will send as notification to coordinator and student. The following code will send an email to Coordinator if any claim submitted by student.

```
16      $sql = "select email,fullname from student,usercheck where student.uid=usercheck.uid and sid='" . $sid . "'";
17      $result = showData($sql);
18  //  print_r($result);
19
20      $mail = new PHPMailer;
21
22      $mail->From = $result[0]["email"];
23      $mail->FromName = $result[0]["fullname"];
24
25      $sql = "select email,fullname from student,usercheck,department,facultycoord "
26              . "where student.did=department.did and "
27              . "department.fid=facultycoord.fid and "
28              . "facultycoord.uid=usercheck.uid and "
29              . "sid='" . $sid . "'";
30      $result = showData($sql);
31     // print_r($result);
32
33      $mail->addAddress($result[0]["email"], $result[0]["fullname"]);
34
35      $mail->isHTML(true);
36
37      $mail->Subject = $sctitle;
38      $mail->Body = "<i>" . $scdetails . "</i>.<br> You need to solve it within" . $replyabledate;
39      $mail->AltBody = "This is the plain text version of the email content";
40
41      if (!$mail->send()) {
42          echo "Mailer Error: " . $mail->ErrorInfo;
43      } else {
44  //          echo "Message has been sent successfully";
45      }
46      header("location:../view/pages/student/claiming.php?msg=1 &err=0& clid=$cid ");
47  } else {
48      header("location:../view/pages/student/claiming.php");
49  }
```

Figure 90: code to send email notification to coordinator

Student will receive a solution as email notification via the execution of the code given below.

```php
    $sql = "select email,fullname from facultycoord,usercheck where facultycoord.uid=usercheck.uid and fid='" . $fid . "'";
    $result = showData($sql);
//    print_r($result);

    $mail = new PHPMailer;

    $mail->From = $result[0]["email"];
    $mail->FromName = $result[0]["fullname"];

    $sql = "select sctitle,email,fullname from student,usercheck,subjectcomplain "
            . "where student.sid=subjectcomplain.sid and "
            . "student.uid=usercheck.uid and "
            . "scid='" . $scid . "'";
    $result = showData($sql);
//    print_r($result);

    $sctitle=$result[0]["sctitle"];
    $mail->addAddress($result[0]["email"], $result[0]["fullname"]);

    $mail->isHTML(true);

    $mail->Subject = "Reply of".$sctitle;
    $mail->Body = "<i>" . $solution . "</i>.<br>";
    $mail->AltBody = "This is the plain text version of the email content";

    if (!$mail->send()) {
        echo "Mailer Error: " . $mail->ErrorInfo;
    } else {
//        echo "Message has been sent successfully";
    }
    header("location: ../view/pages/coordinator/claims.php");
}
```

Figure 91: code to send solution to student as email notification

Provide solution:

The following code will make solution stored if coordinator provide.

```php
<?php

include '../model/dataHandler.php';
include '../config/PHPMailer-master/PHPMailerAutoload.php';

if ($_POST) {
    $scid = $_POST["scid"];
    $solution = $_POST["solution"];
    $fid = $_POST["fid"];
    $solvedate = date('Y/m/d h:i:s', time());
    $sql = "insert into solution(solvedate,solution,scid,fid) "
            . "values('" . $solvedate . "','" . $solution . "','" . $scid . "','" . $fid . "')";
//    echo $sql;
    insertInDatabase($sql);
```

Figure 92: code to store solution

14 days counting:

Coordinator must reply solution within 14 days of the claim submission. Following code wile count and store 14 days from claim submission.

```
8  if (isset($_POST)) {
9      $cname = $_POST["cname"];
10     $scname = $_POST["scname"];
11     $scdetails = $_POST["scdetails"];
12     $submissiontime = date('Y/m/d h:i:s', time());
13     $replyabledate = date('Y-m-d H:i:s', strtotime('+14 day', strtotime($submissiontime)));
14
```

Figure 93: code for 14 days countdown from claim submission

Logout:

This code will make user logout from the system. After logout login must be needed to access the features of the system.

```
1  <?php
2
3  session_start();
4  setcookie("loggin", "", time() + 3600, "/");
5  session_unset();
6  session_destroy();
7  //print_r($_SESSION);
8  //print_r($_COOKIE);
9  header("location:login.php");
10
```

Figure 94: code for make user logout