Design Document for Resource Sharing Portal (StudyKit)

1. Overview

After reviewing the Use Case analysis, following are the basic classes and actions that emerge out:-

Classes: (Basic building blocks of Resource sharing portal - StudyKit)

Sl no.	Class	Principle Responsibility
1	Account creation	Creation of account for new user.
2	Login	Logging into account by an already registered user.
3	Upload material	Uploading study material by an valid user.
4	Accessing material	Showing the contents to everyone.
5	Search and filter	Search for required material among all the available ones.
6	Vote	Valid user can upvote or downvote a material.
7	Report material	Objectionable content can be reported as issue.
8	Edit profile	Profile of user can be edited by him/her later.
9	Forgot password	Account password forgot can be recovered.
10	Save material	Material that user wants for future reference can be saved.

Note: *Other subsidiary classes may get added to the list in course of implementation for the purpose of load balancing and modularity.*

Actions:

Sl. no.	Action
1	Upload/Delete/Rename Material.
2	Action on reports by admin.
3	Load Current Content.
4	Check/Set/Delete Alerts.
5	Validate User.

Note: *There are other minor actions that does not play major role in modeling.*

2. System Structure

Here we describe the final structure. It should, however, be kept in mind that the obtaining the final structure is an iterative exercise – an initial structure is refined as the design progresses. In particular, the dynamic modeling has an impact on the structure.

2.1. Inheritance Structure

There does not seem to be any inheritance structure because of the lack of commonality between the classes. In some places inheritance seems intuitive, for example in specializing Security into BankSecurity and ShareSecurity and specializing Transaction into Buy and Sell. The figure below shows the inheritance structures.

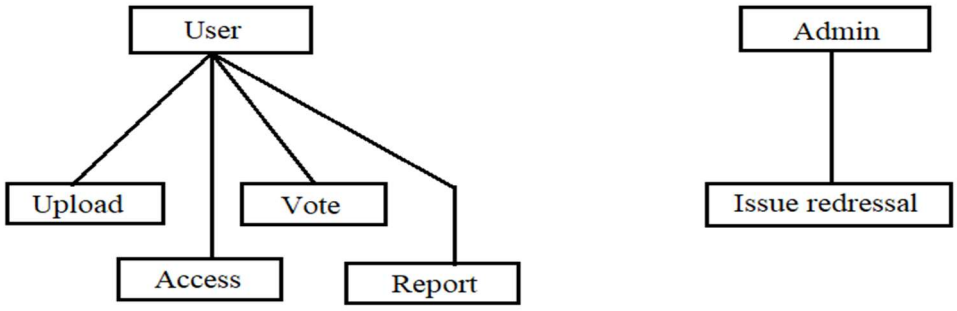


Fig 2.1: Possible Inheritance

2.2. Aggregations

The logical structure of materials suggests the following aggregation between the classes.

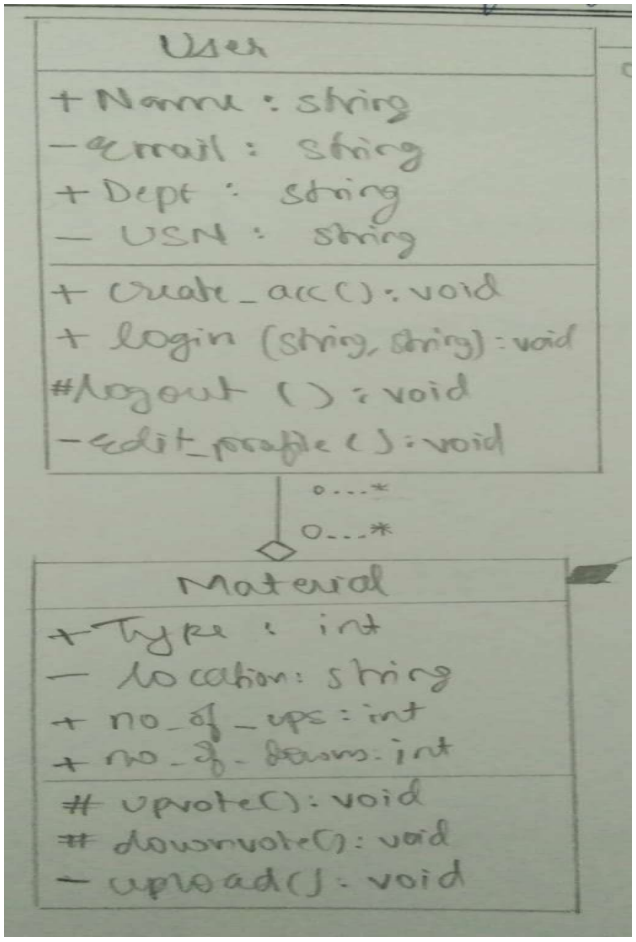


Fig 2.2.1: Aggregation Structure

2.3. Associations

We figure out the association between classes in the process of modeling the principle actions.

Example: Classes (with aggregations and associations) involved in the principle action User, Admin and Issue.

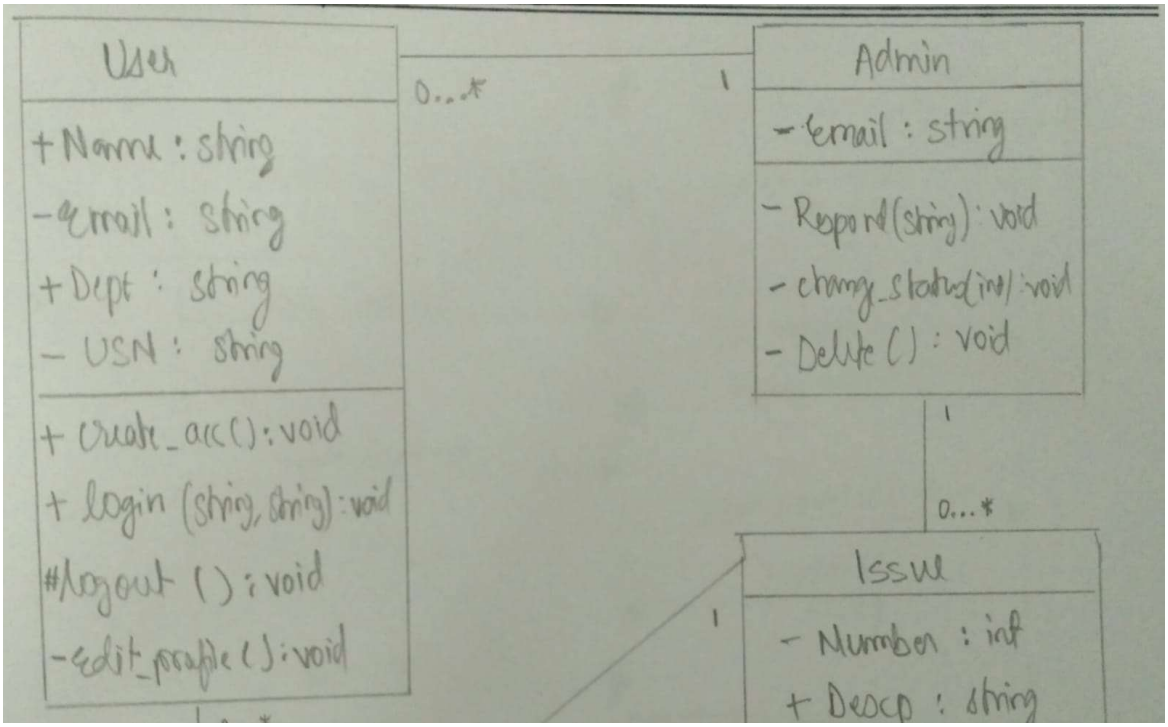


Fig 2.3.1: Class diagram showing associations for action by admin

2.4. Complete class diagram

Finally, after considering all the major actions the complete association + aggregation + composition + multiplicity structure is arrived at.

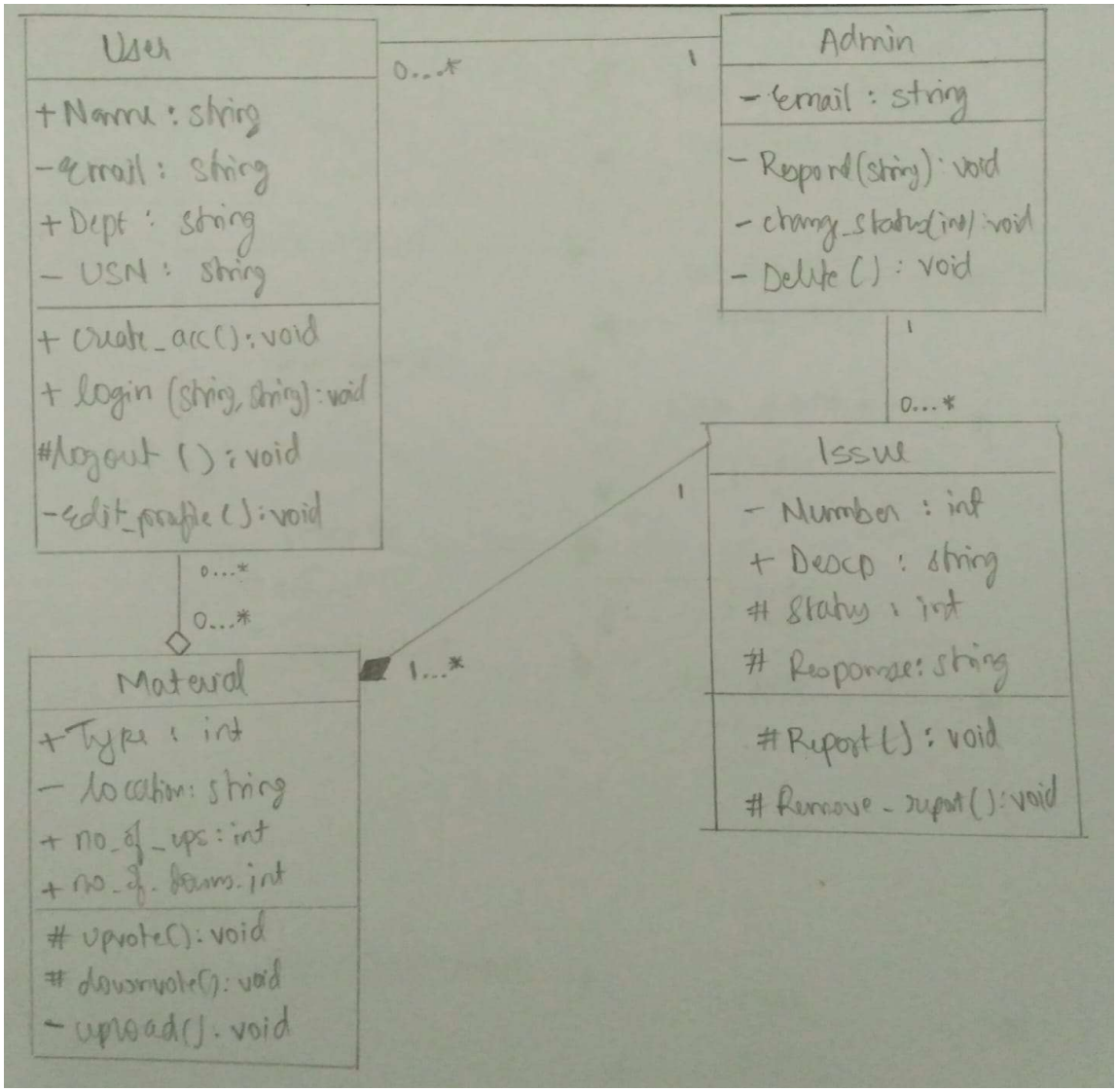


Fig 2.4.1: Class diagram showing all classes and associations in the system

3. System Behavior Sequence Diagram

The dynamic behavior of the system is modeled by figuring out the interactions between the classes involved in each principal action. We are showing the final diagram here. It should be remembered that this model have an impact in refining and enhancing the class diagram

2.5. Principle Action: Combined all the major actions.

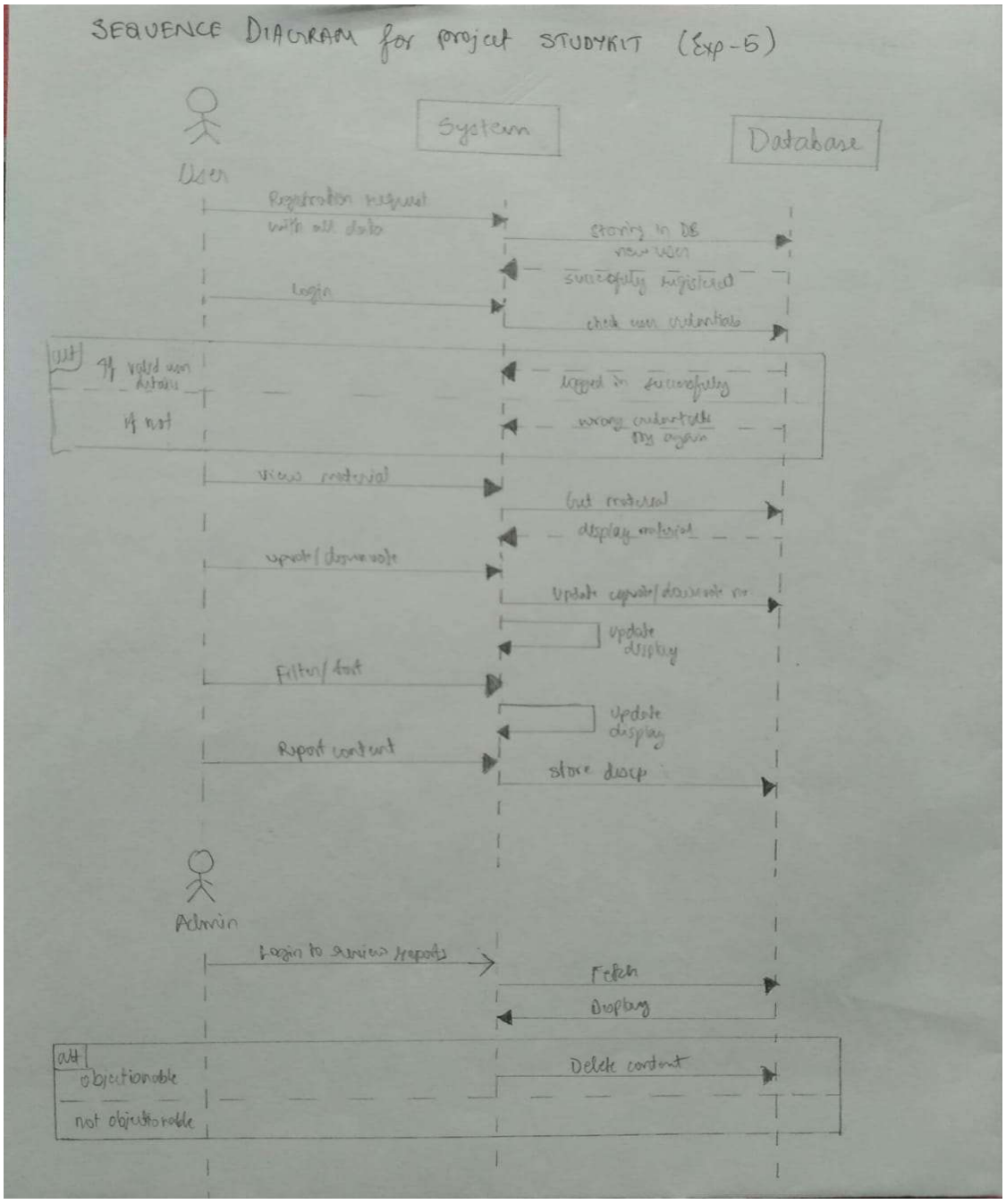


Fig 3.1.1: Sequence diagram for the project

Now we are in a position to start with the design specification as we have all the attributes and methods of all the classes.

4. Detail Design Specification:

It consists of a list of main classes and their attributes and methods with proper comments.

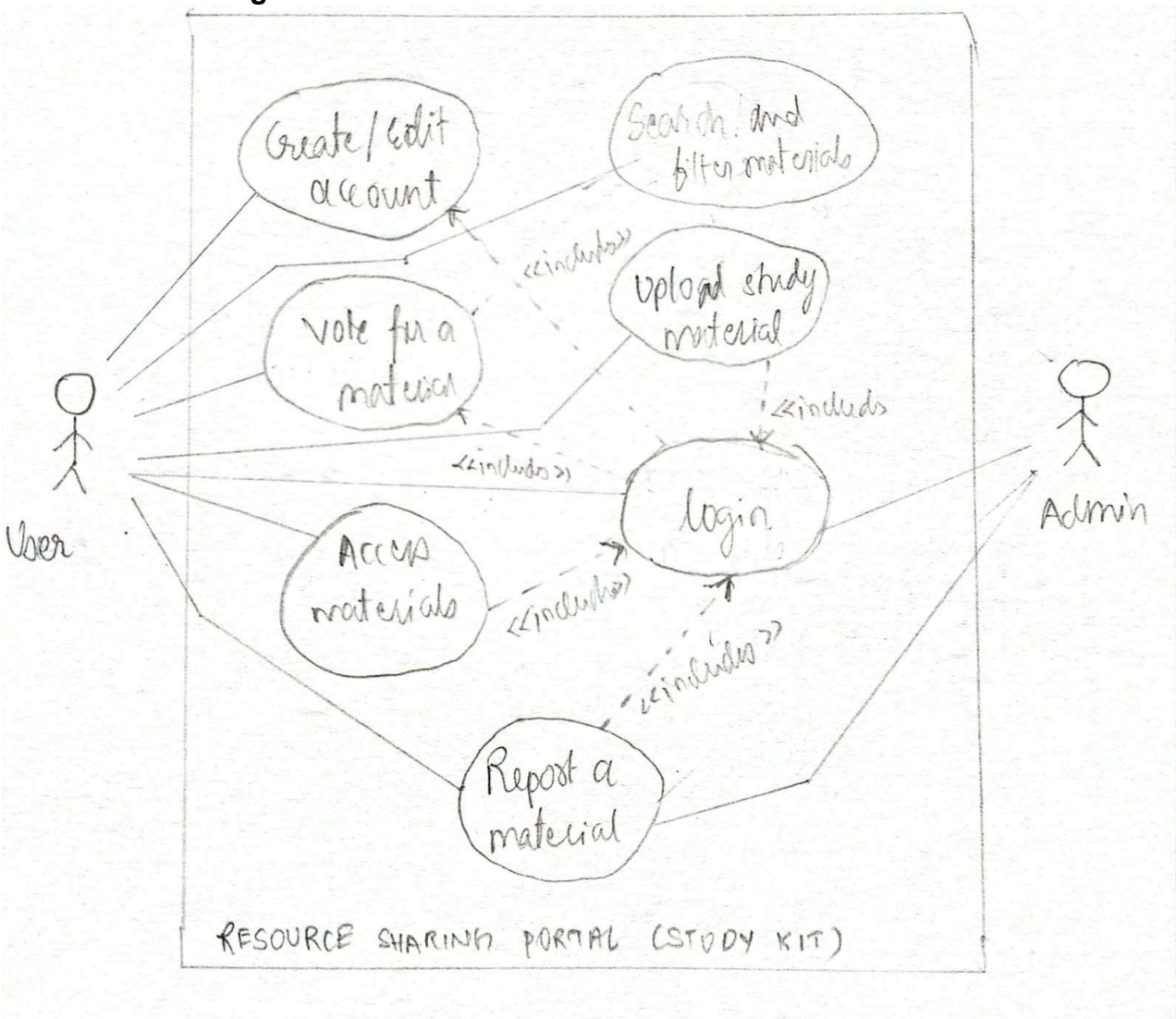
1. class **User**{
   //**attributes**//
   string Name //Details of user
   string Email
   string Dept
   string USN
   //**methods**//
   void create_acc(); *//creates the User account*
   void login(string,string); *//Logs in user*
   void logout(); *//logs out user*
   void edit_profile(); *//edit user profile*
   boolean changePassword(String oldPassword, String newPassword); *// Changes the password of the authorized user*

   }
2. class **Material**{
   //**attributes**//
   string location //location of material
   int type //type of content
   int no_of_ups
   int no_of_downs
   //**method**//
   void upvote()
   void downvote()
   void upload() //publish study material
   }
3. class **Admin**{
   //**attributes//**
   string email //special mailed of admin
   //**method**//
   void respond(string) //resolve issue
   vois change_status() //update status of ongoing issue
   void delete() //delete issue by admin after resolving
   }
4. class **Issue**{
   //**attributes**//
   int number //reported autogenerated
   string descp //description of the objection on content
   int status //status on the reported issue
   string response //reply to an issue by admin

//**method**//
void report() //report an objectionable content
void remove_report() //Remove an issue if user no longer considers content as objectionable.

}

5. Use Case Diagram



Add details:

Use Case #	Use Case	Description
1	Account creation	Creation of account for new user.
2	Login	Logging into account by an already registered user.
3	Upload material	Uploading study material by an valid user.
4	Accessing material	Showing the contents to everyone.
5	Search and filter	Search or filter for required material among all the available ones.
6	Vote	Valid user can upvote or downvote a material.
7	Save a material	User can save material in his profile for future reference.
8	Report material	Objectionable content can be reported as issue.

Details of each use case in the below given format

Use Case	1 - Register
Description	Allows a member to create account
Assumptions	The user knows the details required.
Actors	User
Steps	1. User clicks on register button 2. User types in all the details 3. Click on submit button.
Variations	User can reach register page directly or when he tries to vote, save or report without having a account.
Non-functional	Same user can't have multiple accounts
Issues	When total users increase, registrations can take time.

Use Case	2 - Login
Description	Allows a member to login to the system using his user ID and password
Assumptions	The user remembers his/her ID and password
Actors	• Member

Steps	4. User types in user ID 5. User types in password 6. User clicks on the 'Login' button 7. IF successful THEN show home page ELSE display error
Variations	When a user forgets password he may change it and then login again
Non-functional	Searching for user in database and then login should be quick
Issues	When total users increase, logins can take time.

Use Case	**3 – Upload material**
Description	A vaild user can upload material by specifying asked details correctly
Assumptions	The material is not too large
Actors	User
Steps	1. Login into account 2. Click upload button 3. Fill in details 4. Click on submit button
Variations	Material can be document, video or a link.
Non-functional	Large files shouldn't upload, error should be displayed
Issues	Too many files can burden database.

Use Case	**6 – Vote**
Description	Allows a member to upvote or downvote a content
Assumptions	The user honestly likes or dislikes the content
Actors	User
Steps	1. Login into account 2. Clicks on the upvote/downvote button of a particular material

Variations	User may remove upvote downvote by clicking the button again
Non-functional	More than one vote per user per content can't be given
Issues	When total users increase votes can be too large can take time.

Use Case	**7 - Save**
Description	Allows a member to save a content
Assumptions	The user actually wants the material in future
Actors	User
Steps	1. Login into account 2. Clicks on the save button of a particular material
Variations	User may unsave by clicking the button again
Non-functional	More than one save per user per content can't be given
Issues	When total users increase saves can take time.

Use Case	**8 - Report**
Description	Allows a member to report objectionable content
Assumptions	The user actually finds the content objectionable
Actors	User
Steps	1. Login into account 2. Clicks on the report button of a particular material. 3. Fill the details of the issue. 4. Click submit.
Variations	User may remove report.
Non-functional	More than one issue per content can't be given
Issues	When contents increase, admin may be burdened with too many reports to redress.

5. Data Flow Diagram