Indira Gandhi National Open University

MCSP-060
Connection - The Social Network

by

Vishvajeet
Ramanuj
E. no: 17543066

Under
Guidance of
Jayesh Solanki

Submitted to the School of Computer and Information Sciences, IGNOU in partial fulfillment of the requirements
for the award of the degree

Master of Computer Applications (MCA)
Year of Submission – 2020

IndiraGandhiNationalOpenUniversity
MaidanGarhi
New Delhi – 110068

Table of Contents

Introduction and Objectives of the Project

Introduction:

Social media are interactive computer-mediated technologies that facilitate the creation or sharing of information, ideas, career interests and other forms of expression via virtual communities and networks

Social Media is very powerful. It allow people to share their ideas, day to day life things, own philosophy and lot's of other things easily with people who matter to them easily.
Many people also use it to promote their business by sharing post related to their products. Some uses Advertising feature of social network to show advertisement even to users who are new and not know the business organization before.

Networks formed through social media change the way groups of people interact and communicate or stand with the votes. They "introduce substantial and pervasive changes to communication between organizations, communities, and individuals.

Some people user Social media as news Source. Social media differ from paper-based media (e.g., magazines and newspapers) and traditional electronic media such as TV broadcasting, Radio broadcasting in many ways, including quality,[5]reach, frequency, interactivity, usability, immediacy, and performance. Social media outlets operate in a dialogic transmission system (many sources to many receivers).[6] This is in contrast to traditional media which operates under a mono-logic transmission model (one source to many receivers), such as a newspaper which is delivered to many subscribers, or a radio station which broadcasts the same programs to an entire city.

Some of the most popular social media websites, with over 100 million registered users, include Facebook (and its associated Facebook Messenger), TikTok, WeChat, Instagram, QZone, Weibo, Twitter, Tumblr, Baidu Tieba, LinkedIn and VK. Other popular platforms that are sometimes referred to as social media services (differing on interpretation) include YouTube, QQ, Quora, Telegram, WhatsApp, LINE, Snapchat, Pinterest, Viber, Reddit, Discord and more.

We have seen a wide range of positive and negative impacts of social media use. Social media can help to improve an individual's sense of connectedness with real

or online communities and can be an effective communication (or marketing) tool for corporations, entrepreneurs, non-profit organizations, advocacy groups, political parties, and governments.

Objectives:

Objective of Social Network is to connect people with other peoples who matter to them and other people whom they even don't know but have similar interest.

Connection will provide tool to user to share information, idea, personal thing. Connection will also do the same as other prominent social networks do but with more freedom to users, respecting their privacy and security.

Connection is for user so user are the first priority. It also allow advertiser because not all user can afford the subscription price. Connection is giving it's service free to them and showing them advertisement.

System Analysis

Identification of need:

Social network available to user currently are doing many unfair practices like compromising with user privacy, maintaining monopoly through unfair practices, no option to control over data they generate (social networking company will use their data anyway to make profit).

I believe user should have right on their data. User should have option to opt out from advertisements by paying subscription.

Social currently operates under opacity. Many people know that social network do share our private chat with government by using backdoor of their own system. It also uses that private. On the basis of this we can estimate that what they called ENCRYPTED CHAT (Spooky) is how much secure and respect privacy. This same thing applies to what we share via post including our personal things.

Many users are irritated of unfair practice of current social network and looking for alternative. Connection is used to satisfy that need.

Preliminary Investigation:

Feasibility Study:

1.) Technical Feasibility:

Using Django we can develop secure, scalable and production grade application like what we require for connection

For allowing user to migrate there is need for common protocol or standard according to that data of user is backup and restore and making migration possible.

2.) Operational Feasibility:

It is operationally feasible to fulfill user's requirement of sharing information, chat, and update with community.

3.) Schedule Feasibility:

This Project can take 3 to 6 months for completion.

4.) Legal Feasibility:

Connection is not in conflict with existing laws. However it can come into conflict because it is not intended to share user's private information with anyone including government

5.) Economic Feasibility:

Users are provided two option free and premium subscription. Premium users are free from advertisements. While those who can't afford premium will still use connection but they has to watch. In general we can say that connection is affordable to all.

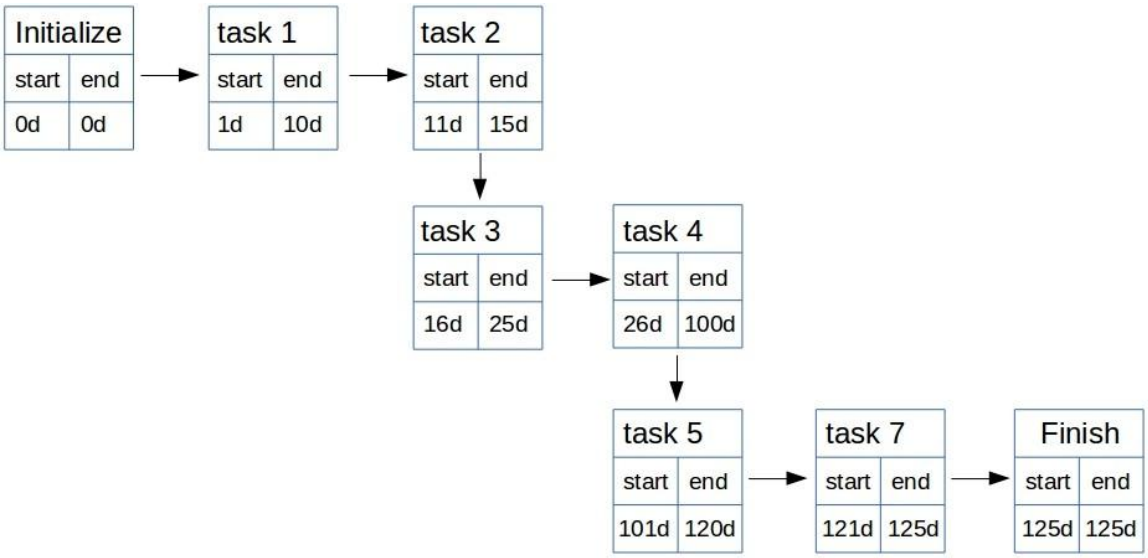
Project planning:

The software has been planned to complete within a time span of 16-17 weeks including requirement analysis, feasibility study, designing, coding and quality analysis. This period has been strategically divided among the tasks so as to provide ample amount of time for each work. Planning and feasibility study tend to be the most significant phases of software development as it studies the technical, time-bound and legal limitations of the software development based on the requirement analysis.

Scheduling:

Task No.	**Activity**	**Days**
1	Understanding user requirements & gathering information	10
2	Feasibility Study	5
3	System Analysis	10
4	System Design	15
5	UI design / Project Coding	60
6	Quality Analysis	20
7	Implementation	5
TOTAL DAYS		**125**

Project scheduling:



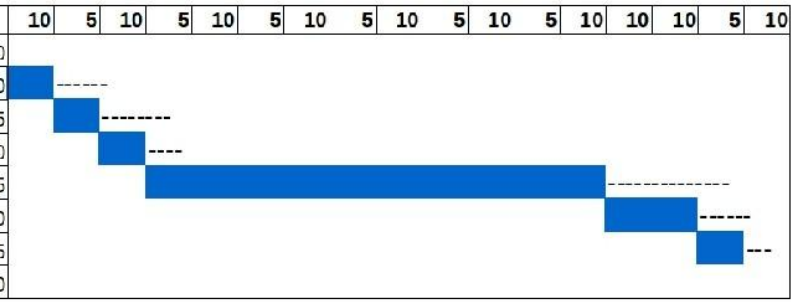
Legend:

d = days

task 1: questionnaire / information gathering
task 2: feasibility study
task 3: system analysis / system design
task 4: designing & coding
task 5: QA / QC
task 6: implementation

PERT Chart

id	Task Name	Predecessors	Days
1	Start	-	0
2	task 1	Start	10
3	task 2	task 1	5
4	task 3	task 2	10
5	task 4	task 3	75
6	task 5	task 4	20
7	task 6	task 5	5
8	Finish	task 6	0



Legend:

---------- slack time

normal time

task 1: questionnaire / information gathering
task 2: feasibility study
task 3: system analysis / system design
task 4: designing and coding
task 5: quality analysis/ quality control
task 6: implementation

GANTT Chart

SRS **Software Requirement Specification:**

What is SRS?

A software requirements specification (SRS) is a description of a software system to be developed. It lays out functional and non-functional requirements, and may include a set of use cases that describe user interactions that the software must provide.

Why SRS?

In order to fully understand one's project, it is very important that they come up with an SRS listing out their requirements, how are they going to meet it and how will they complete the project. the team to save upon their time as they are able to comprehend how are going to go about the project.

Doing this also enables the team to find out about the limitations and risks early on.

Project Plan: The Book-keeper Accounting Applications

1. Introduction

Connection provide another way to connect with people of similar interest and people who matter to them. It is more secure and respect the privacy of user.

2. Overview

Connection has some extra features which differentiate from other social network. Connection will provide facility of migration so that if user feels that they want to port to other network then they can easily do that. Same way user can also transefer from other network to our network. But currently we has to depends upon other social network from which user want to migrate. In future version It can also figure out some method to do it on it’s own.

Users

This is for the people who want privacy. It is available in free and premium service so everyone can afford it. Those who are irritated with advertisement on existing social network can switch to hear for premium membership. Also advertisement are also general in nature and only uses basic profile detail not other details at all.

Functionality

- User can express himself and share it with other via post.
- User can share things via stories
- User can know and let other people inform about events
- User can create community for uniting people with similar interest and share progress of work if they made any.
- User can follow people, organization, community they like and stay updated with latest progress.

Platform

Thesoftwareisawebbasedapplicationandcanberemotelyaccessedthrough networkconnection.

Development Responsibility

User share their personal information. It is very important that you maintain security so that users and their information stay safe and they feel free to share information on our platform.

It is also out responsibilities that prevent bad content from spreading on our platform. Currently there is no such mechanism to do so and platform depends upon other users to report bad contents.

3. Goals andScopes

- Provide Secure platform to user which also respect privacy of user.
- Help people to stay connected with one who matter for them.
- Get inform and share information about event
- Stay connected with people who has same interest via community
- share information with fun via stories.

4. Deliverable

- Featurespecification
- Productdesign
- TestPlan
- Developmentdocument
- Sourcecode

5. Scheduling andEstimates

Milestone	Description	Release Iteration
M1	Application view andDesign (Front-end development)	R1
M2	Database modeling (Back-end)	R1
M3	Integrating views and designs (Integrating front-end and back-end)	R1
M4	Testing	R2
M5	Issue tracker, bug tracing, and web design integration	R2
M6	Final release	R3

6. TechnicalProcess

Front End / GUI tools:

- HTML5
- CSS3
- JavaScript(including jQuery, Bootstrap and other libraries)
- Bootstrap

RDBMS and DBMS Control Panel:

- MySQL
- MySQLWorkbench

Backend Language:

- Python for web development (DjangoFramework)

Data Models:

Data Dictionary

Abbreviation
PK – Primary Key
FK – Foreign Key
ta = targeted_audience
fr = friend_request
ra = register_activity

common_user

Field Name	Data Type	Required	Field Length	Constraint	Description
c_user_id	Integer	Yes	10	PK	Uniquely identify
username	String	Yes	32		
dob	Date	Yes	10		Birth date of user
location	String		10		Location of user
country	String		10		Country of user
mobile	Integer	Yes	10	UNIQUE	Mobile number of user
email	String	Yes	50	UNIQUE	Email of user

community

Field Name	Data Type	Required	Field Length	Constraint	Description
community_id	Integer	Yes	10	PK	
name	String	Yes	32		Name of community
admin_id	Integer	Yes	10	FK	Foreign key of user
description	String		100		Describe about community
rules	String		1000		Rule which need to be followed by people in order to be healthy community

post

Field Name	Data Type	Required	Field Length	Constraint	Description
post_id	Integer	Yes	10	PK	Uniquely identify
text_description	String		5000		User add description to what he had posted
content	BLOB		5000		User can post image, video, feeling or activity
user_who_posted	Integer	Yes	10	FK	It is foreign_key from user table
type	String	Yes	10		It can be life event, routine post, informational post etc.

advertisement

Field Name	Data Type	Required	Field Length	Constraint	Description
advertisement_id	Integer	Yes	10	PK	
content	BLOB		5000		Image or video
text_description	String		500		Description about ads
ta_location	String	Yes	32		Location of audience to whom advertiser want to target
ta_age	Integer		3		Age of Target audience
ta_gender	String		5		Gender of target audience

story

Field Name	Data Type	Required	Field Length	Constraint	Description
story_id	Integer	Yes	10	PK	
description	String		500		Description of story of user
content	BLOB		5000		Image , text or video who user posted
user_who_posted	Integer	Yes	10	FK	Id of user who posted story. It is foreign key of user table
view	JSON				JSON object containing name of people who viewed story.

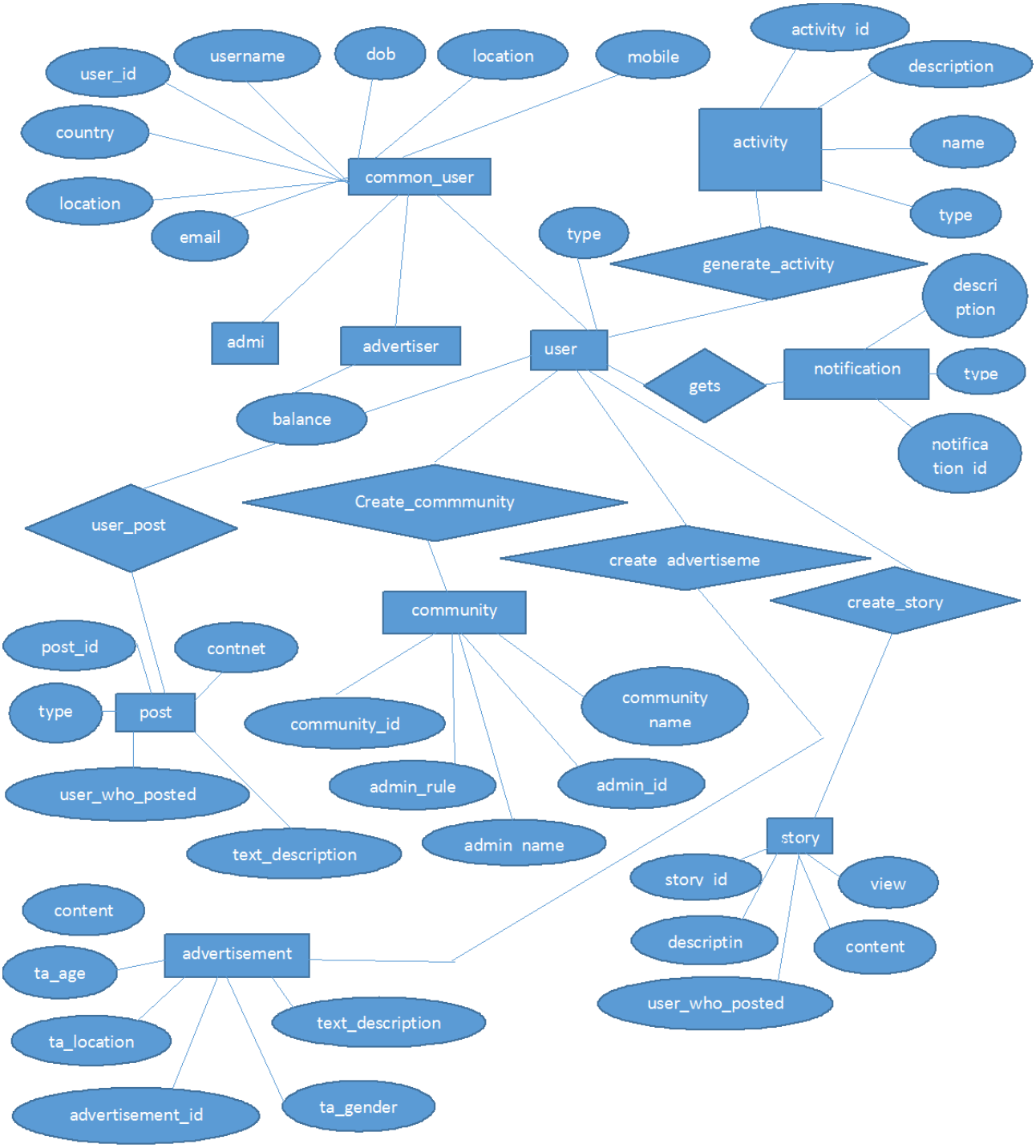
search

Field Name	Data Type	Required	Field Length	Constraint	Description
search_id	Integer	Yes	10	PK	
start_date	Date				Search start from this date
end_date	Date				Search till this date
location	String				Location in which we want to search
item_to_search	String	Yes			Item which user want to search
item_type	JSON				It contain details of item type to be search

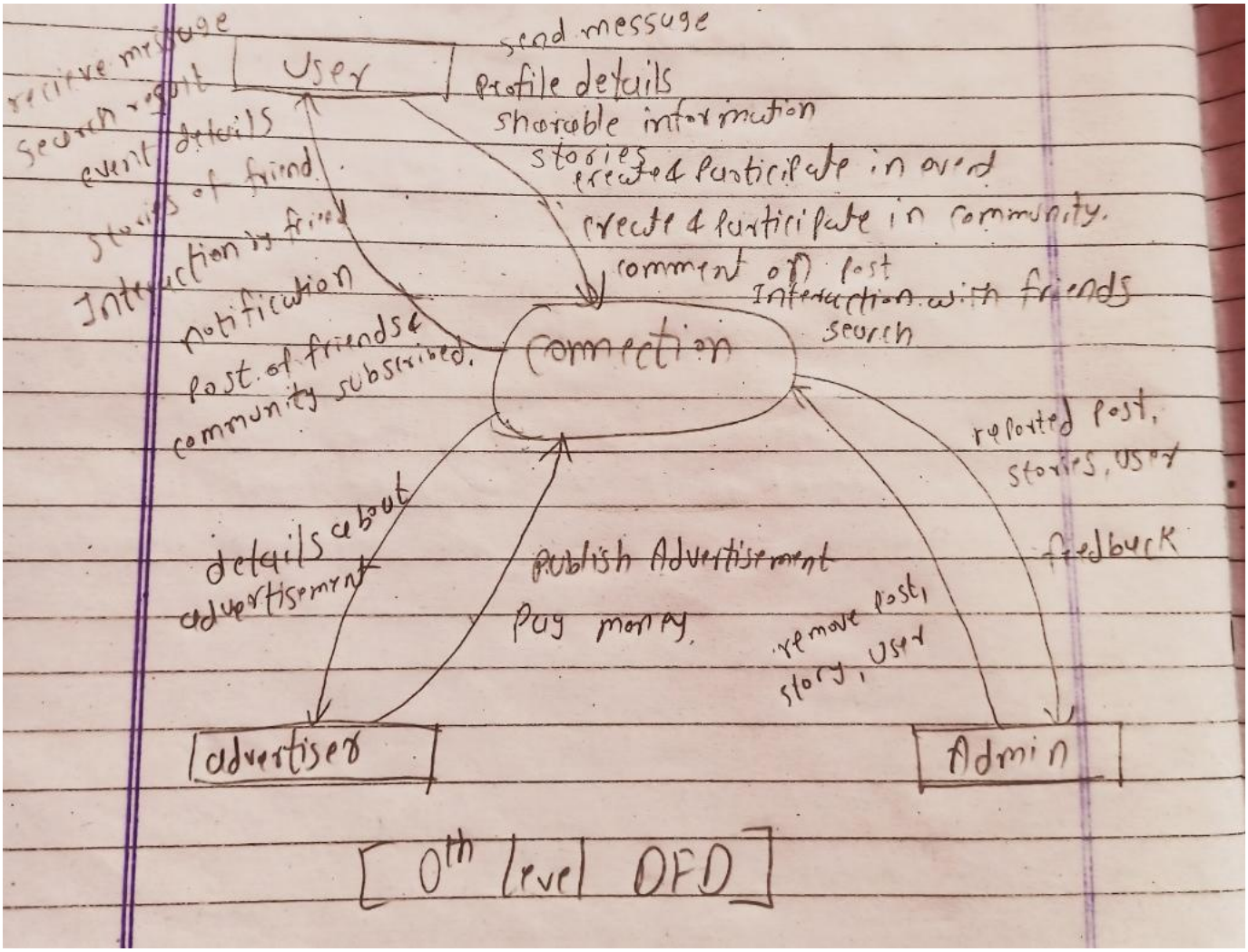
activity

Field Name	Data Type	Required	Field Length	Constraint	Description
advertisement_id	Integer	Yes	10	PK	
name	String	Yes	10		Activity name
type	String	Yes			Type of activity
description	String		100		Description of activity

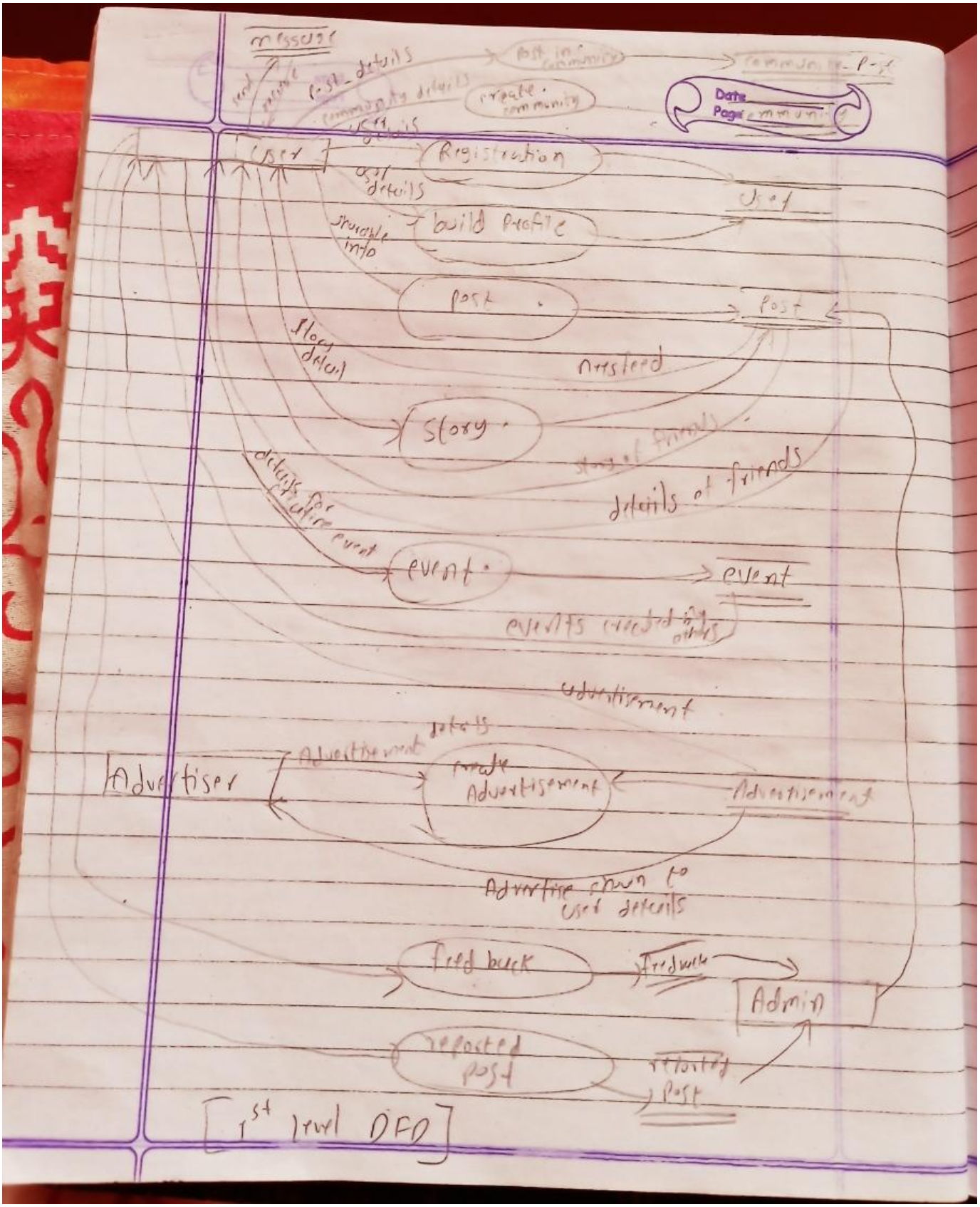
DFD Diagrams:


activity_id
description
username
dob
location
mobile
user_id
country
activity
name
common_user
location
type
email
type
generate_activity
description
admi
advertiser
user
notification
type
gets
balance
notification_id
Create_commmunity
user_post
create_advertiseme
create_story
community
post_id
contnet
community_name
type
post
community_id
user_who_posted
admin_rule
admin_id
text_description
admin_name
story
story_id
view
content
descriptin
content
user_who_posted
advertisement
ta_age
text_description
ta_location
advertisement_id
ta_gender

ER Diagram



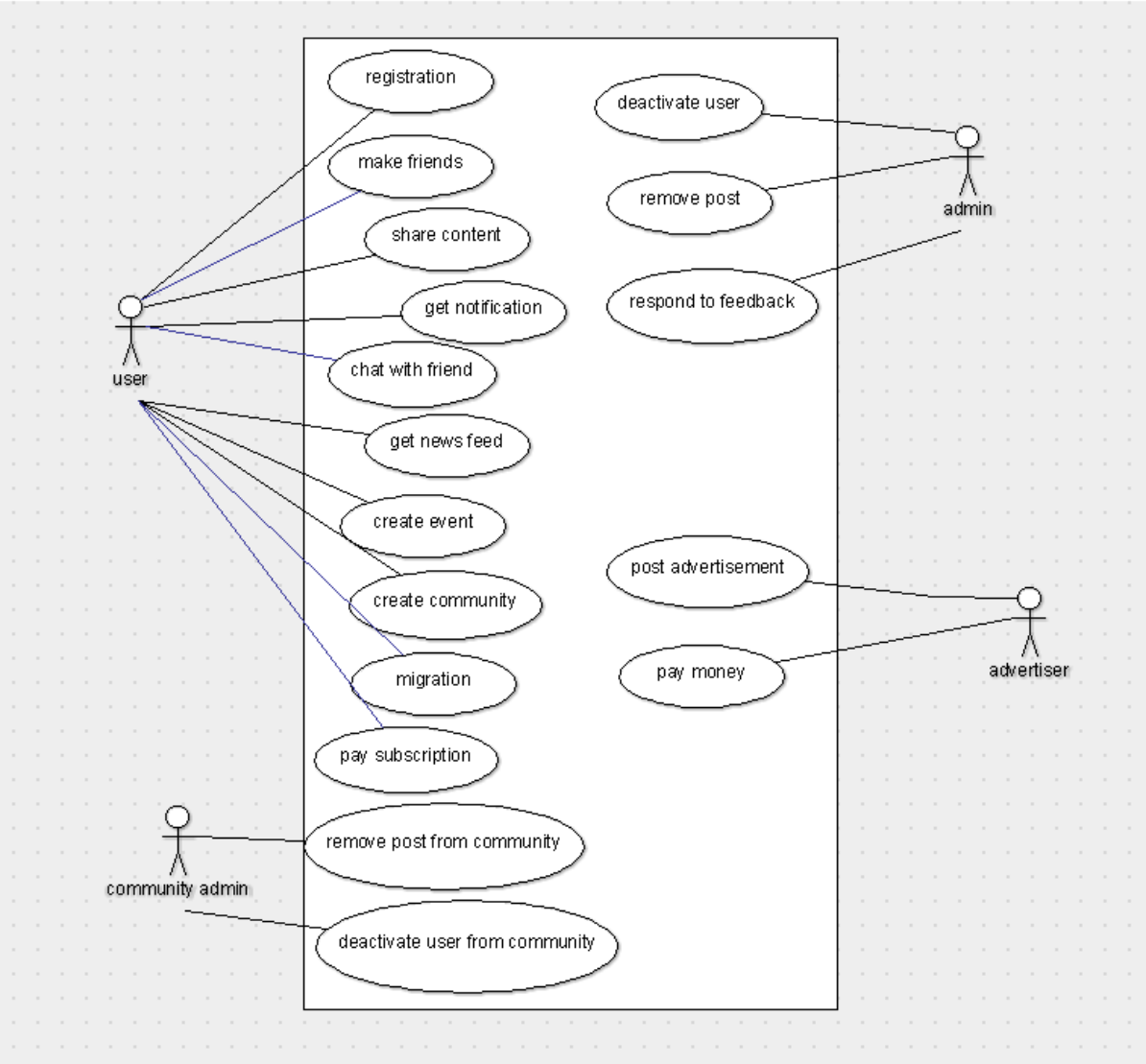
0th level DFD



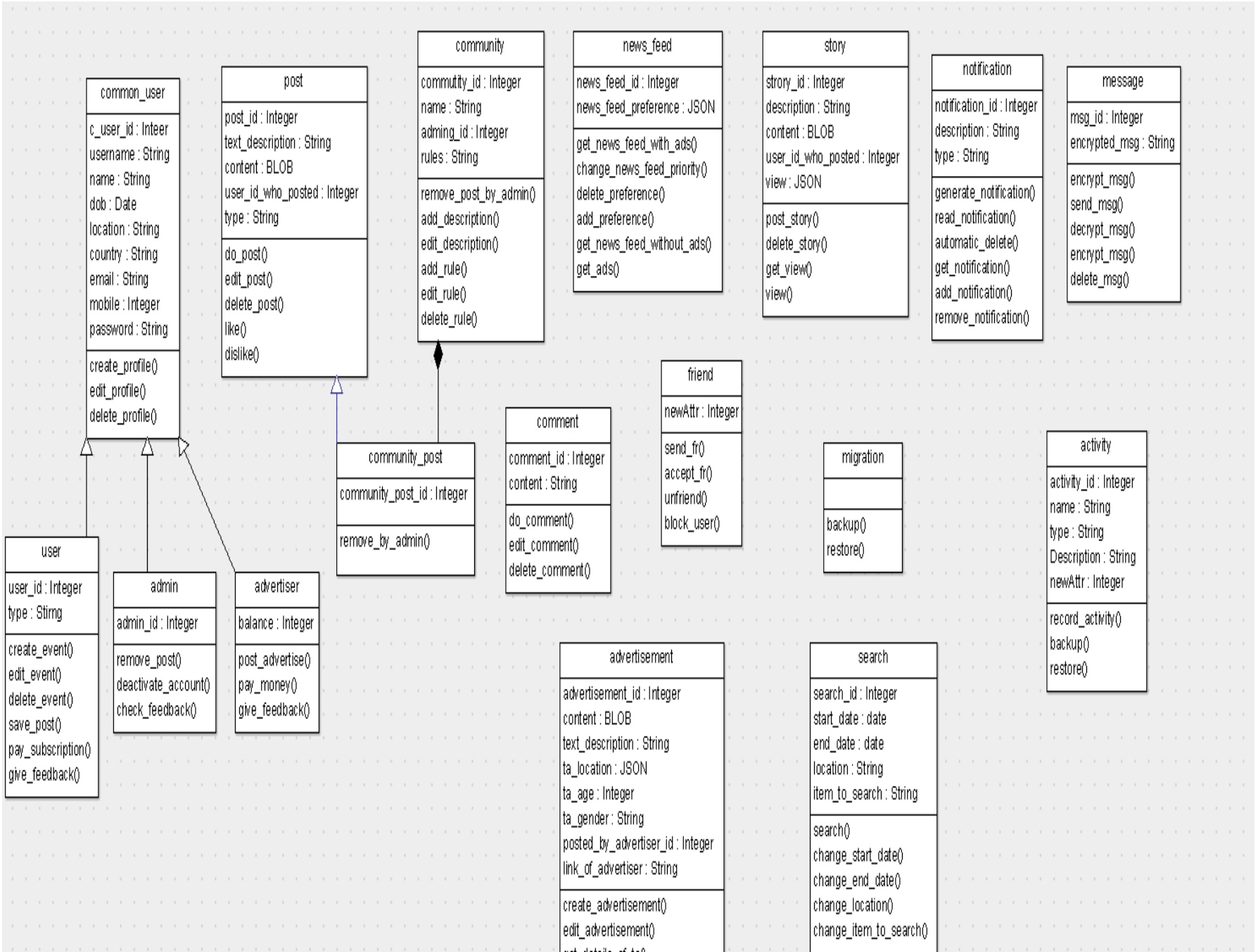
1st Level DFD

UML DIAGRAMS

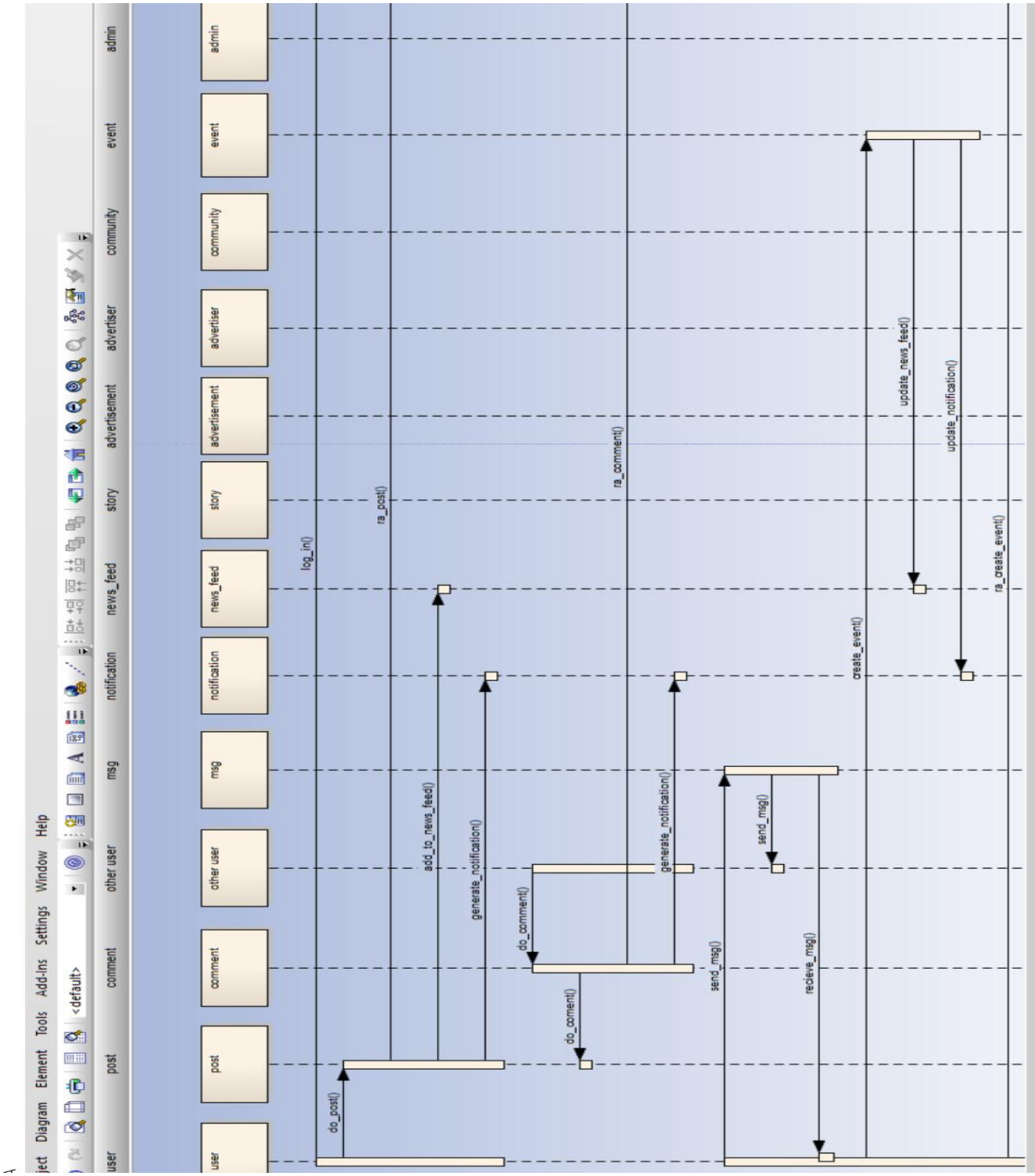
Use Case Diagram

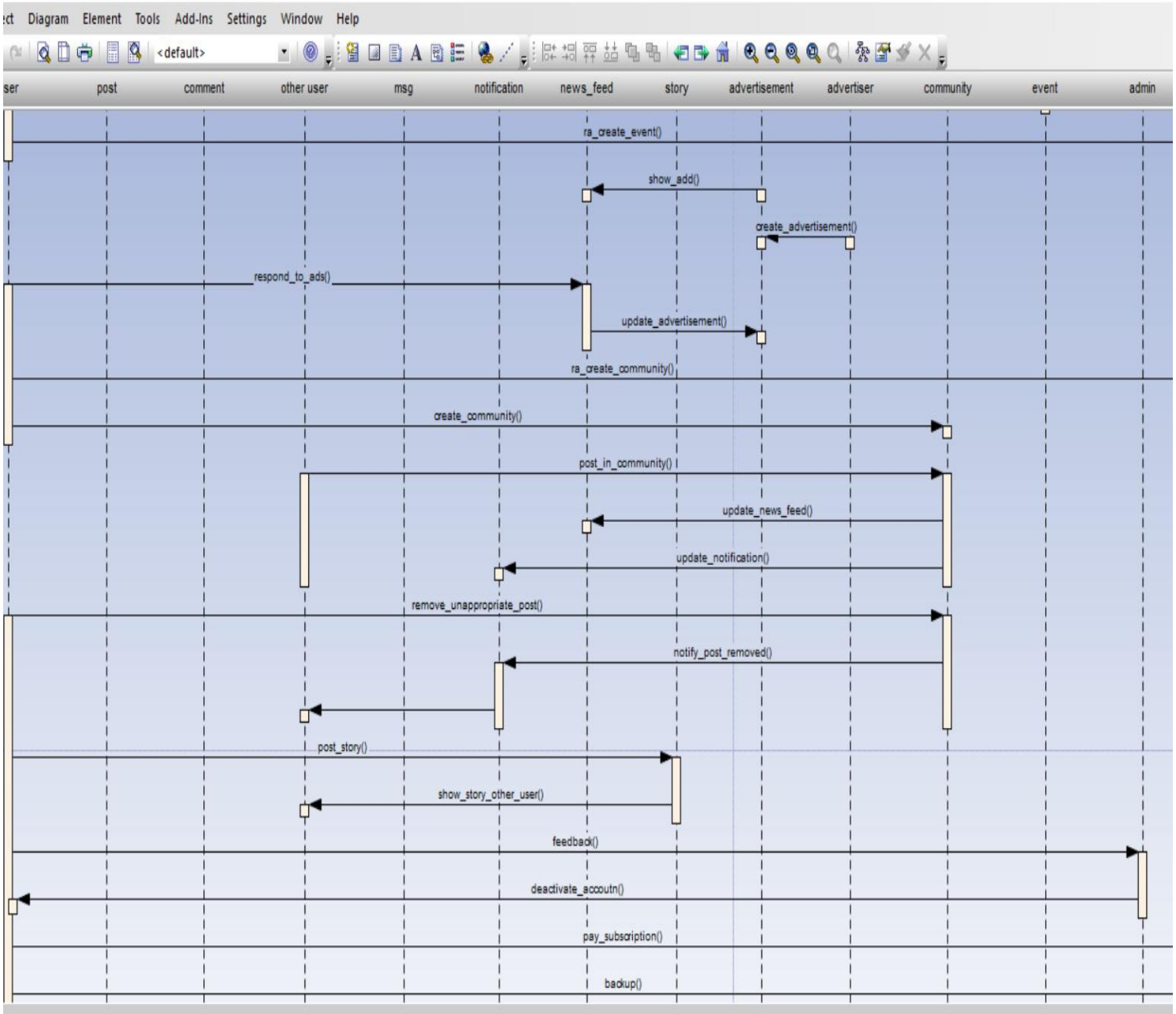


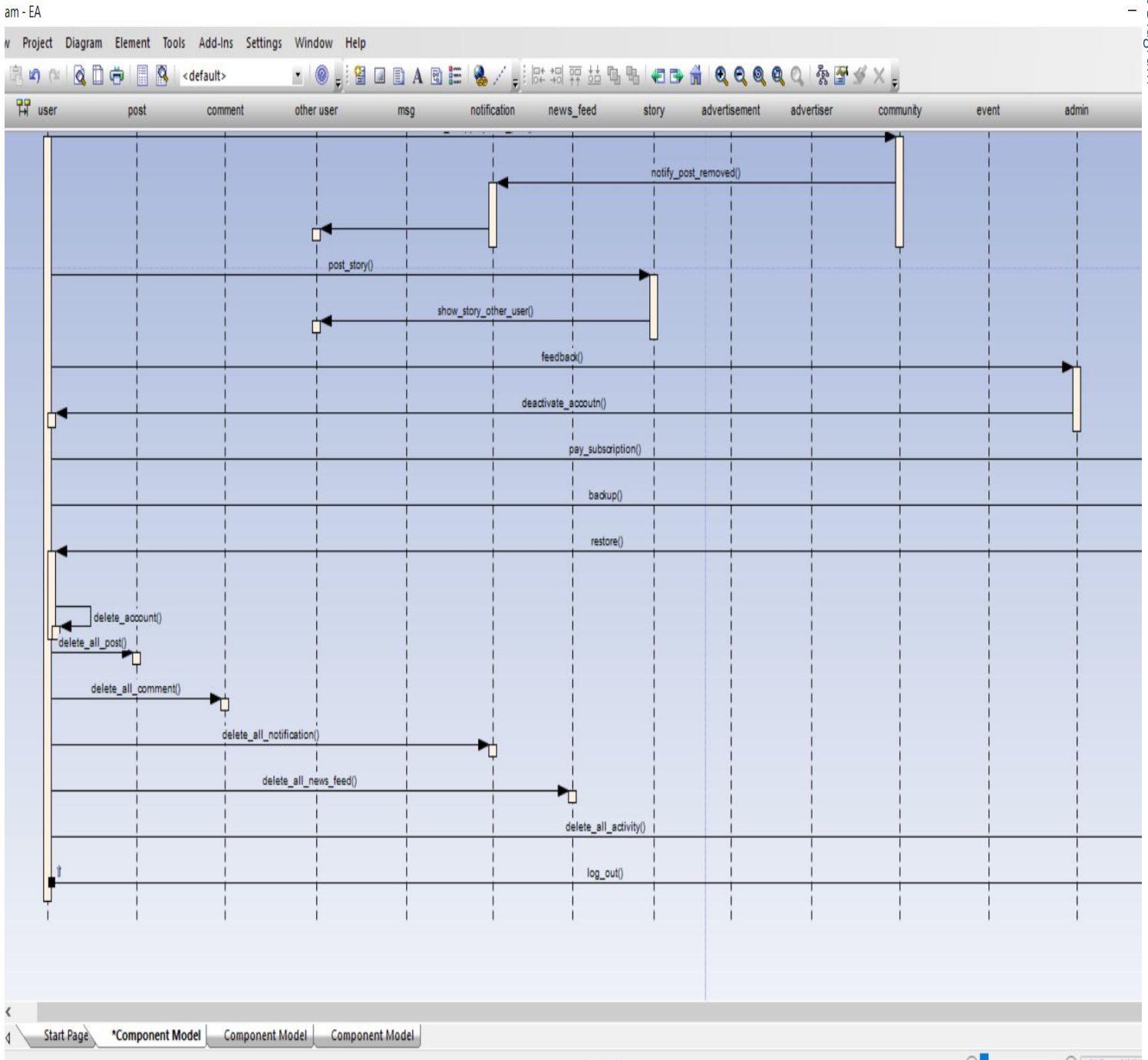
Class Diagram

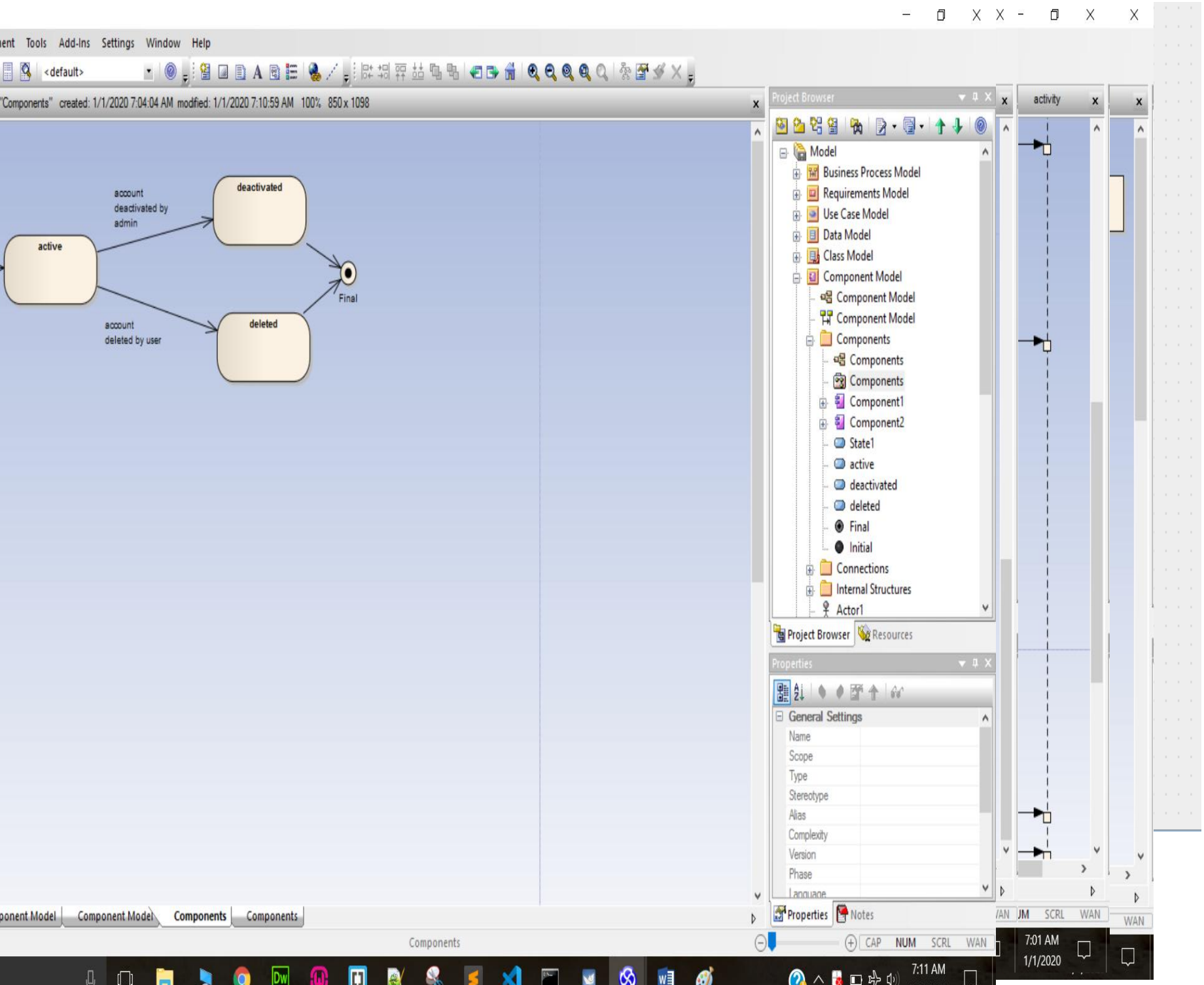

common_user
c_user_id : Inteer
username : String
name : String
dob : Date
location : String
country : String
email : String
mobile : Integer
password : String
create_profile()
edit_profile()
delete_profile()
post
post_id : Integer
text_description : String
content : BLOB
user_id_who_posted : Integer
type : String
do_post()
edit_post()
delete_post()
like()
dislike()
community
commutity_id : Integer
name : String
adming_id : Integer
rules : String
remove_post_by_admin()
add_description()
edit_description()
add_rule()
edit_rule()
delete_rule()
news_feed
news_feed_id : Integer
news_feed_preference : JSON
get_news_feed_with_ads()
change_news_feed_priority()
delete_preference()
add_preference()
get_news_feed_without_ads()
get_ads()
story
strory_id : Integer
description : String
content : BLOB
user_id_who_posted : Integer
view : JSON
post_story()
delete_story()
get_view()
view()
notification
notification_id : Integer
description : String
type : String
generate_notification()
read_notification()
automatic_delete()
get_notification()
add_notification()
remove_notification()
message
msg_id : Integer
encrypted_msg : String
encrypt_msg()
send_msg()
decrypt_msg()
encrypt_msg()
delete_msg()
user
user_id : Integer
type : Stirng
create_event()
edit_event()
delete_event()
save_post()
pay_subscription()
give_feedback()
admin
admin_id : Integer
remove_post()
deactivate_account()
check_feedback()
advertiser
balance : Integer
post_advertise()
pay_money()
give_feedback()
community_post
community_post_id : Integer
remove_by_admin()
comment
comment_id : Integer
content : String
do_comment()
edit_comment()
delete_comment()
friend
newAttr : Integer
send_fr()
accept_fr()
unfriend()
block_user()
migration
backup()
restore()
activity
activity_id : Integer
name : String
type : String
Description : String
newAttr : Integer
record_activity()
backup()
restore()
advertisement
advertisement_id : Integer
content : BLOB
text_description : String
ta_location : JSON
ta_age : Integer
ta_gender : String
posted_by_advertiser_id : Integer
link_of_advertiser : String
create_advertisement()
edit_advertisement()
search
search_id : Integer
start_date : date
end_date : date
location : String
item_to_search : String
search()
change_start_date()
change_end_date()
change_location()
change_item_to_search()

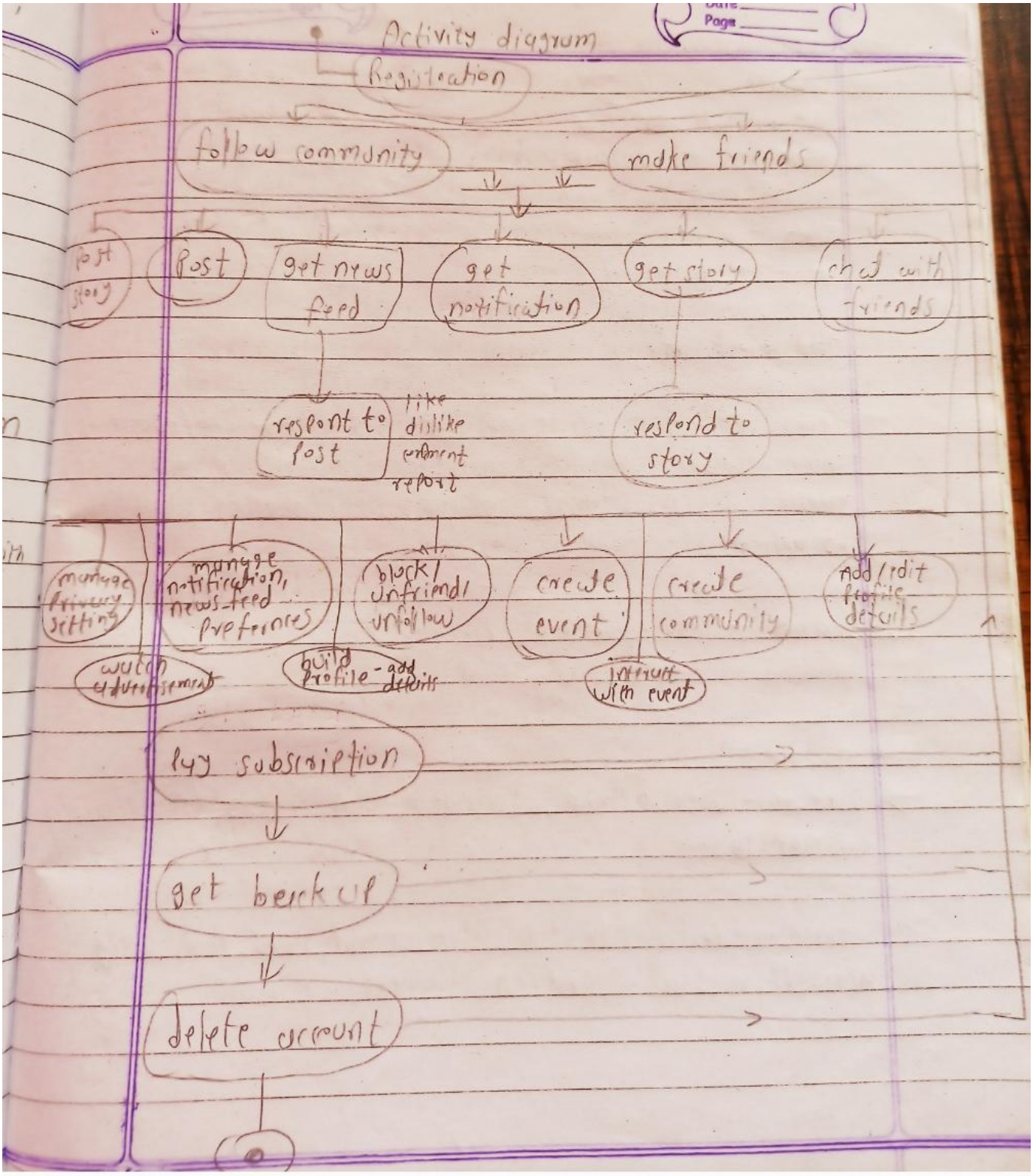
Sequence Diagram




ct Diagram Element Tools Add-Ins Settings Window Help
<default>
ser
post
comment
other user
msg
notification
news_feed
story
advertisement
advertiser
community
event
admin
ra_create_event()
show_add()
create_advertisement()
respond_to_ads()
update_advertisement()
ra_create_community()
create_community()
post_in_community()
update_news_feed()
update_notification()
remove_unappropriate_post()
notify_post_removed()
post_story()
show_story_other_user()
feedback()
deactivate_accoutn()
pay_subscription()
backup()


am - EA
Project Diagram Element Tools Add-Ins Settings Window Help
<default>
user
post
comment
other user
msg
notification
news_feed
story
advertisement
advertiser
community
event
admin
notify_post_removed()
post_story()
show_story_other_user()
feedback()
deactivate_accoutn()
pay_subscription()
backup()
restore()
delete_account()
delete_all_post()
delete_all_comment()
delete_all_notification()
delete_all_news_feed()
delete_all_activity()
log_out()
Start Page
*Component Model
Component Model
Component Model


Tools
Add-Ins
Settings
Window
Help
<default>
"Components" created: 1/1/2020 7:04:04 AM modified: 1/1/2020 7:10:59 AM 100% 850 x 1098
account deactivated by admin
deactivated
active
Final
account deleted by user
deleted
Project Browser
Model
Business Process Model
Requirements Model
Use Case Model
Data Model
Class Model
Component Model
Component Model
Component Model
Components
Components
Components
Component1
Component2
State1
active
deactivated
deleted
Final
Initial
Connections
Internal Structures
Actor1
Project Browser
Resources
Properties
General Settings
Name
Scope
Type
Stereotype
Alias
Complexity
Version
Phase
Language
Properties
Notes
activity
Component Model
Component Model
Components
Components
Components
CAP
NUM
SCRL
WAN
7:01 AM
1/1/2020
7:11 AM


Activity diagram
Registration
follow community
make friends
post story
Post
get news feed
get notification
get story
chat with friends
respond to post
like
dislike
comment
report
respond to story
manage privacy setting
manage notification, news feed preferences
block/ unfriend/ unfollow
create event
create community
Add/edit profile details
watch advertisement
build profile - add details
interact with event
pay subscription
get backup
delete account

System Design

Modularization Details:

Modularization is a technique to divide a software system in to multiple discrete and independent modules, which are expected to be capable of carrying out task(s) independently. These modules may work as basic constructs for the entire software. Designers tend to design modules such that they can be executed and / or compiled separately and independently.

Modular design unintentionally follows the rules of ‘divide and conquer’ problem-solving strategy this is because there are many other benefits attached with the modular design of software.

Advantages of modularization:

- Smaller components are easier to maintain
- Program can be divided based on functional aspects
- Desired level of abstraction can be brought in the program
- Component switch high cohesion can be re-used again
- Concurrent execution can be made possible
- Desired from security aspect

Modular components of 'The Book-keeper Accounting Application':

- activity – record activities of user. It help in migration
- advertisement – dealing with managing advertisement
- community – manages operations on community
- event – manages operations on event
- friend – manages friend request how it process and other operation related to friends
- newsfeed – it provide filtered way to give update to the user from all updates
- notification – providing a way to communicate about what is happening related to them in platforms
- post – this manages how user share things via post
- story – this module manages how user can share things via story
- user – User is the core the system. This module manages authorization, authentication. It also contain profile details of user.
- user_migration – This module manages functionalities related to migration.

Data integrity and constraints:

The data integrity in the software has been maintained with the help of various constraints on primary keys. On deletion of a primary key, the related foreign key fields will also be removed. On modification of a related key which is referenced in various other tables, the modification takes place with cascading effect. The software data structure and constraints have been designed in the above mentioned manner and makes full use of the same.

SQL Query Command for constraints:

for deletion

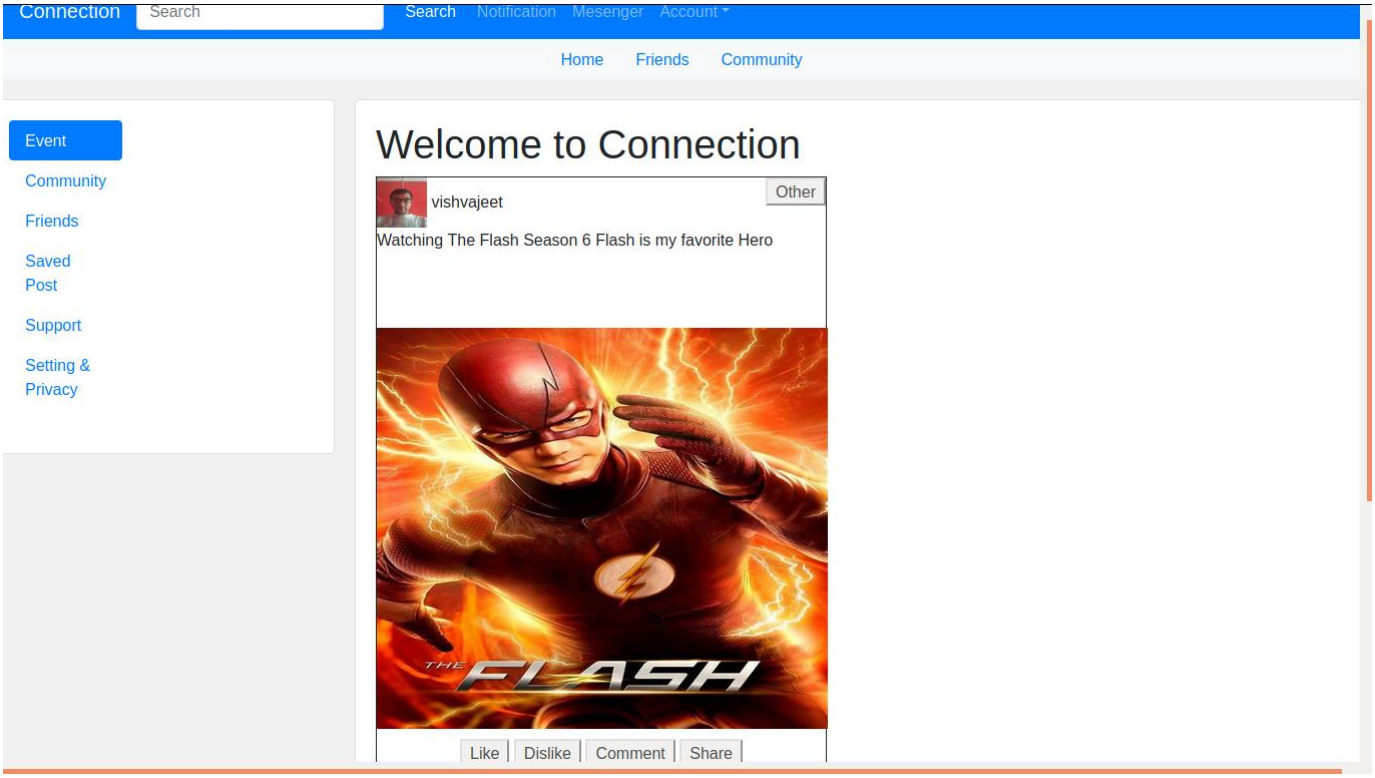
ALTER TABLE <*referencing_table_name*>
ADD CONSTRAINT <*constraint_name*>
FOREIGNKEY(<*primary_key*>)REFERENCES<*master_table*>(<*primary_key*>) ON DELETECASCADE;

for updation

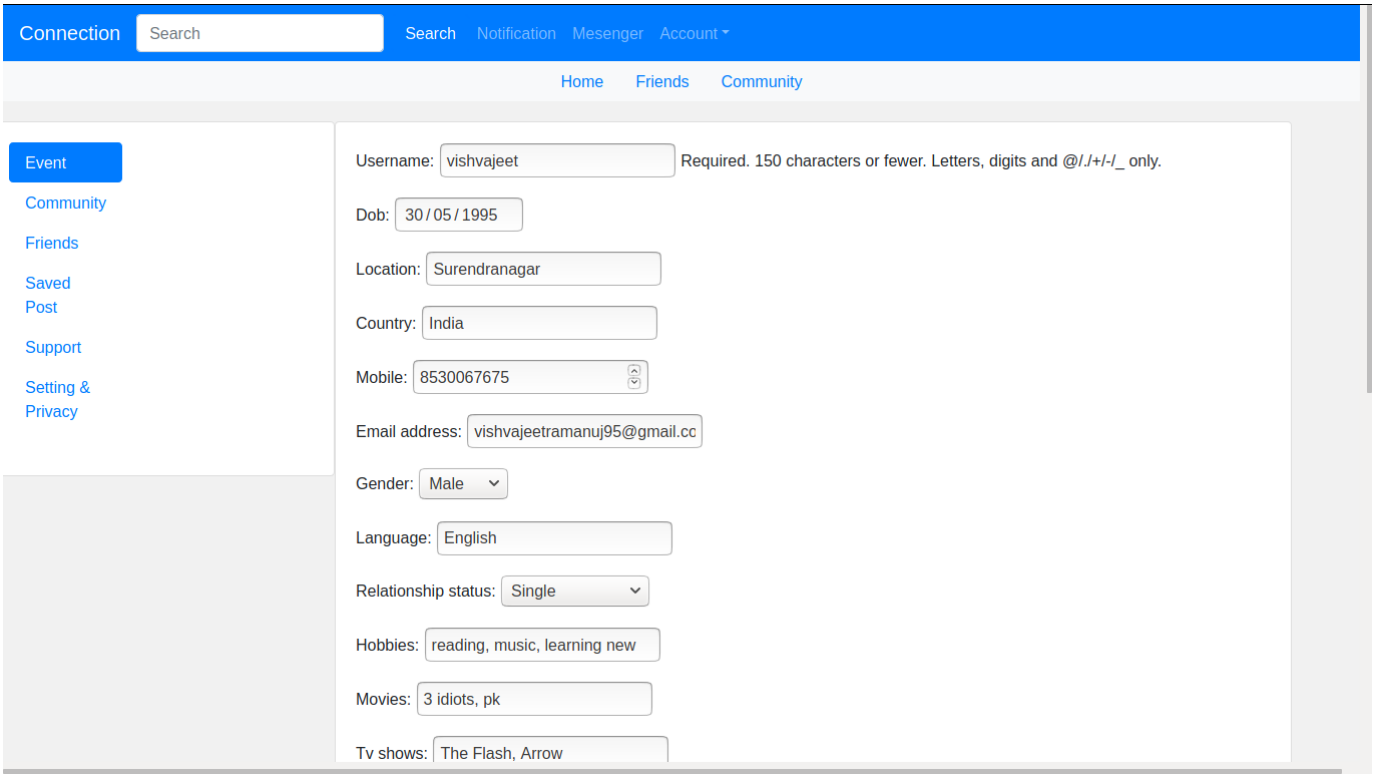
ALTER TABLE <*referencing_table_name*>
ADD CONSTRAINT <*constraint_name*>
FOREIGNKEY(<*primary_key*>)REFERENCES<*master_table*>(<*primary_key*>) ON UPDATECASCADE;

Database designs and Data model classes:

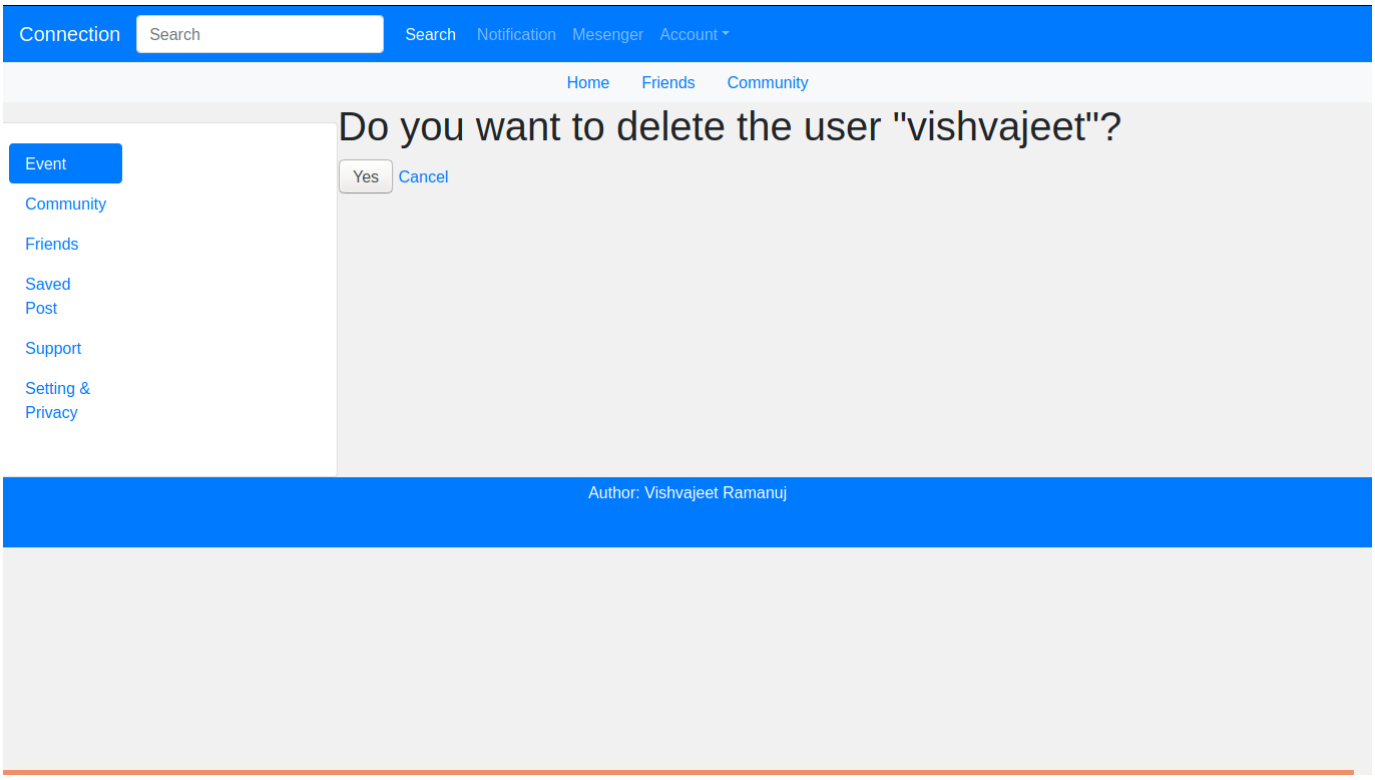
UI designs:



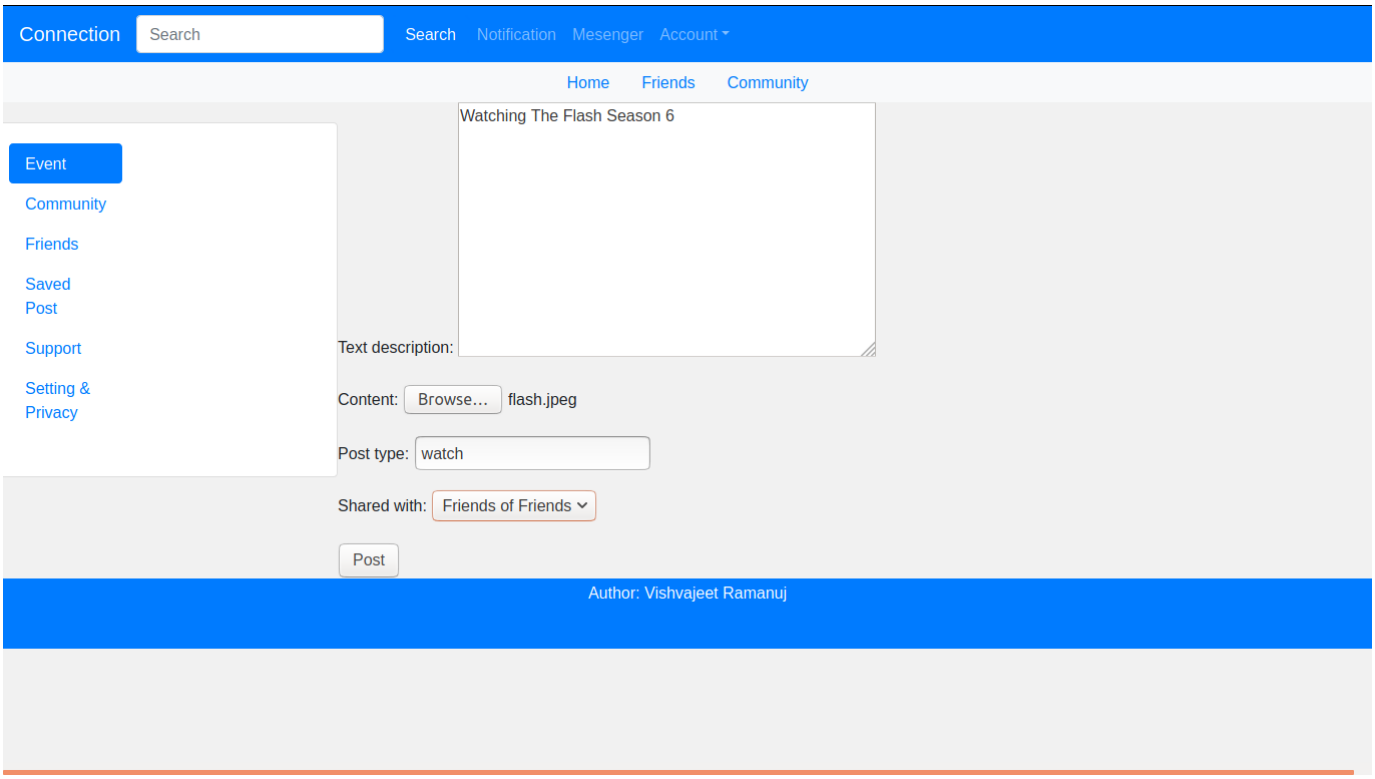
Home page where user can see post of his friends as well as people he follow (user can see that post only if the user who posted has allowed to watch)



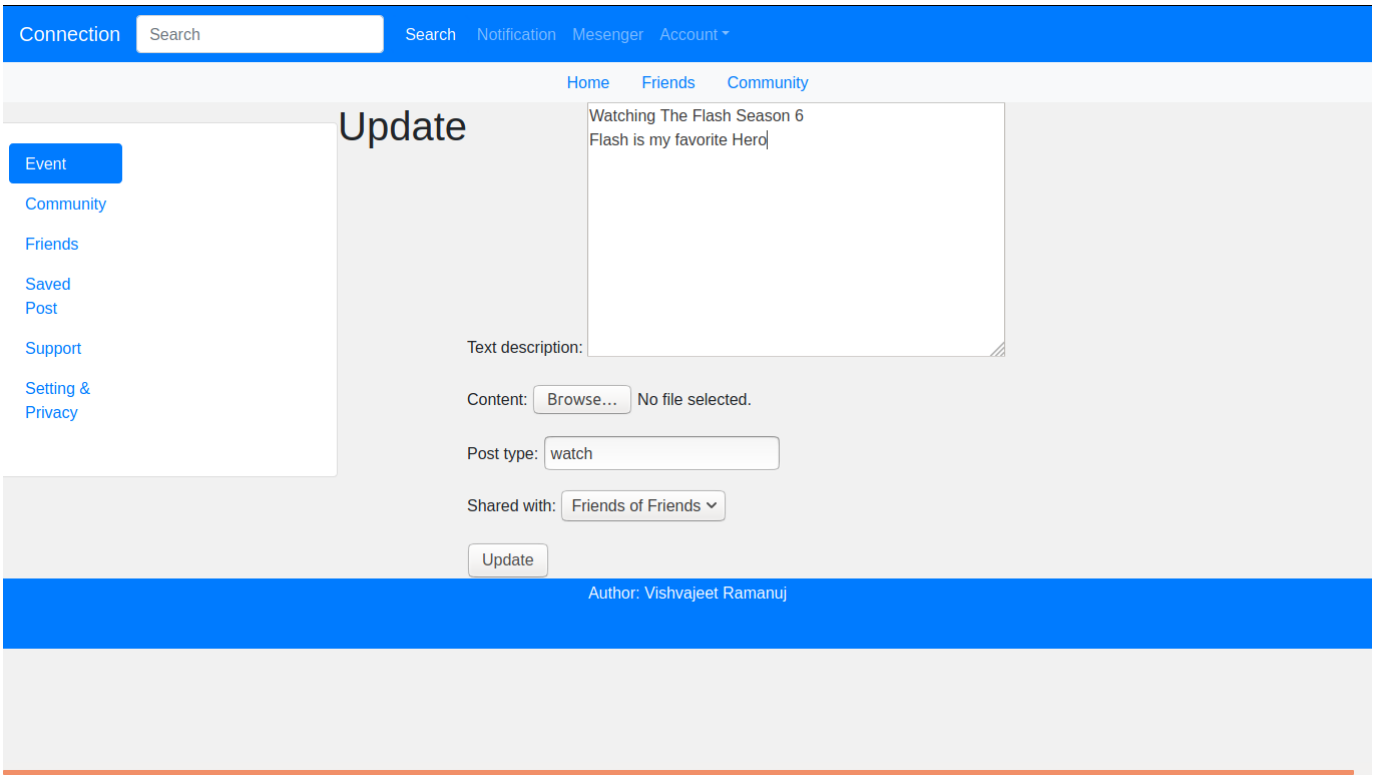
User Registration is first step. User can see things on connection only if he or she is registered.



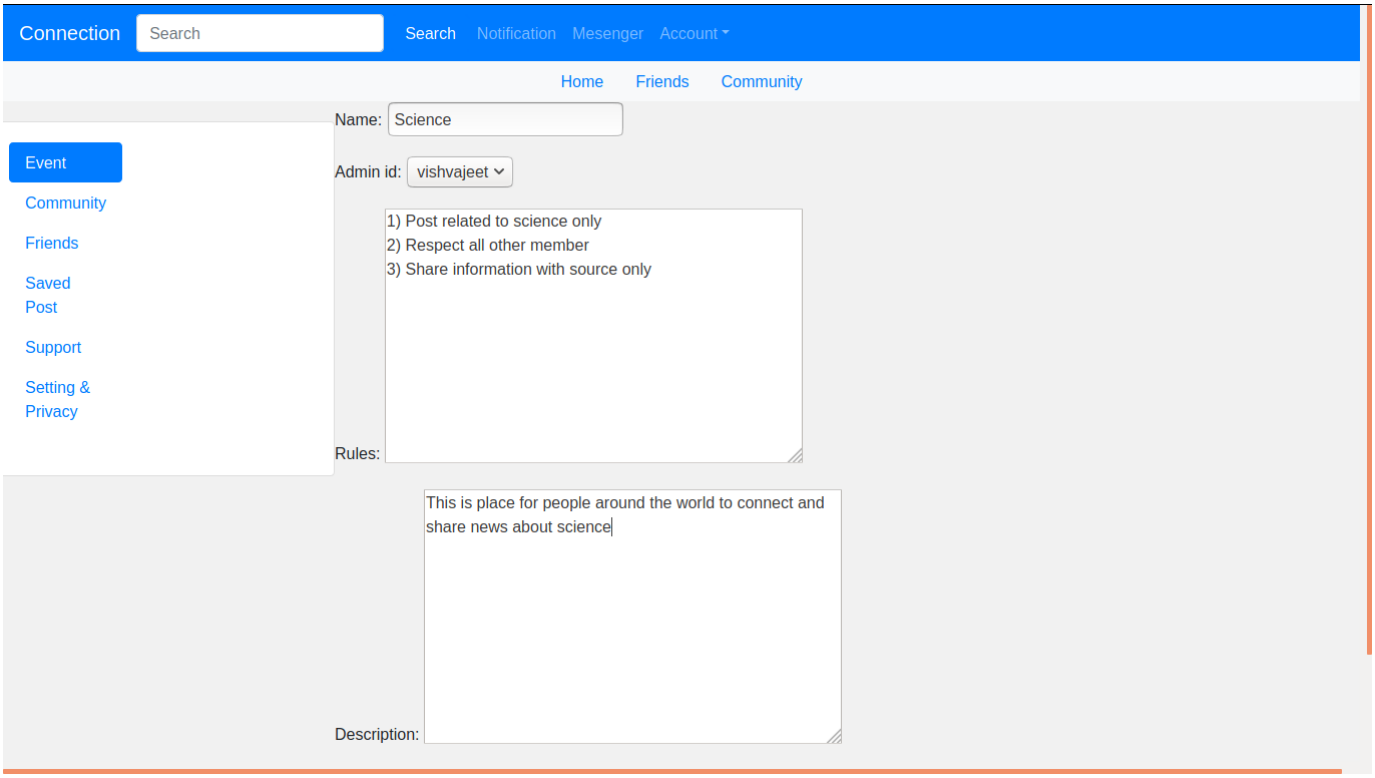
If user want to delete account. He can press yes. It will delete his account and all his personal detail from connection. Post and commented done by him will still stay visible.



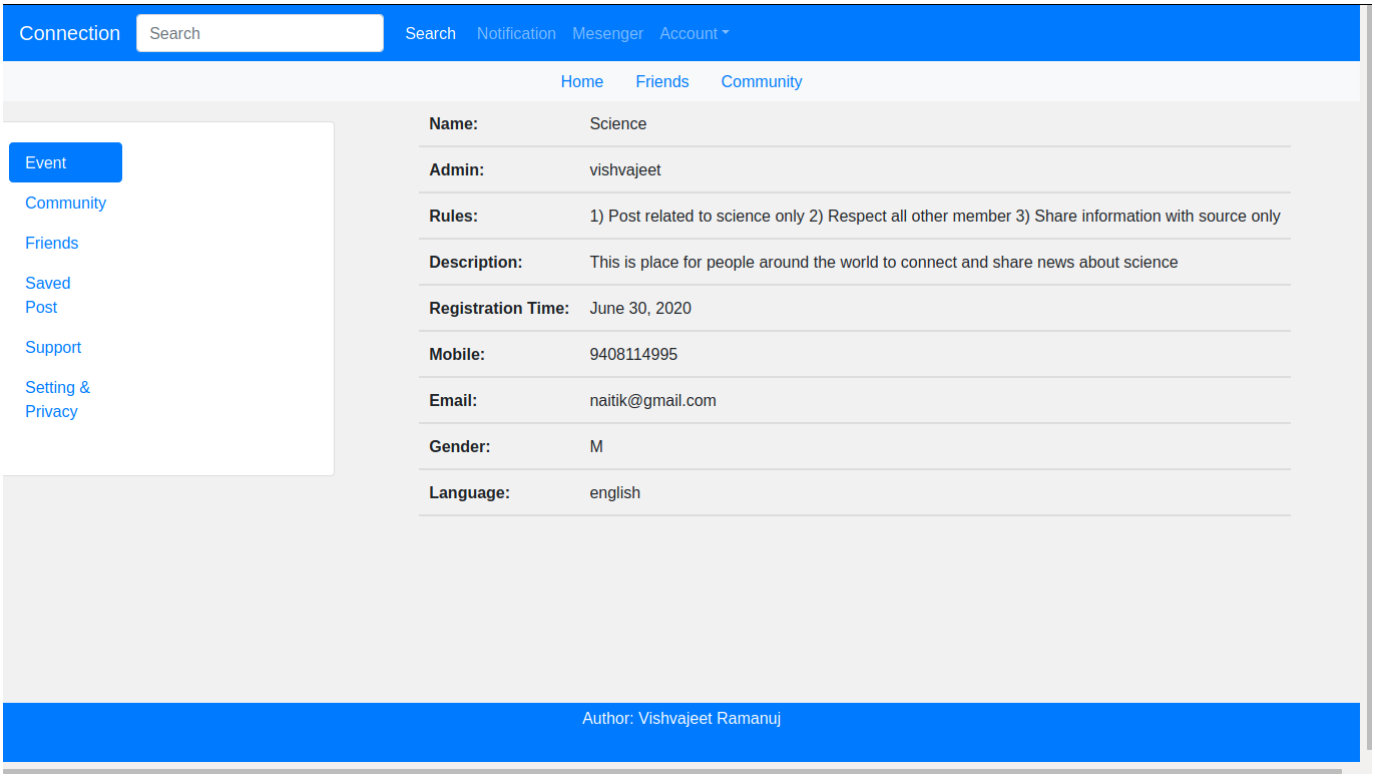
If user want to post then he can make post by writing description, selecting image which he want to share, write type of post, select option to whom user want to share post and then click post



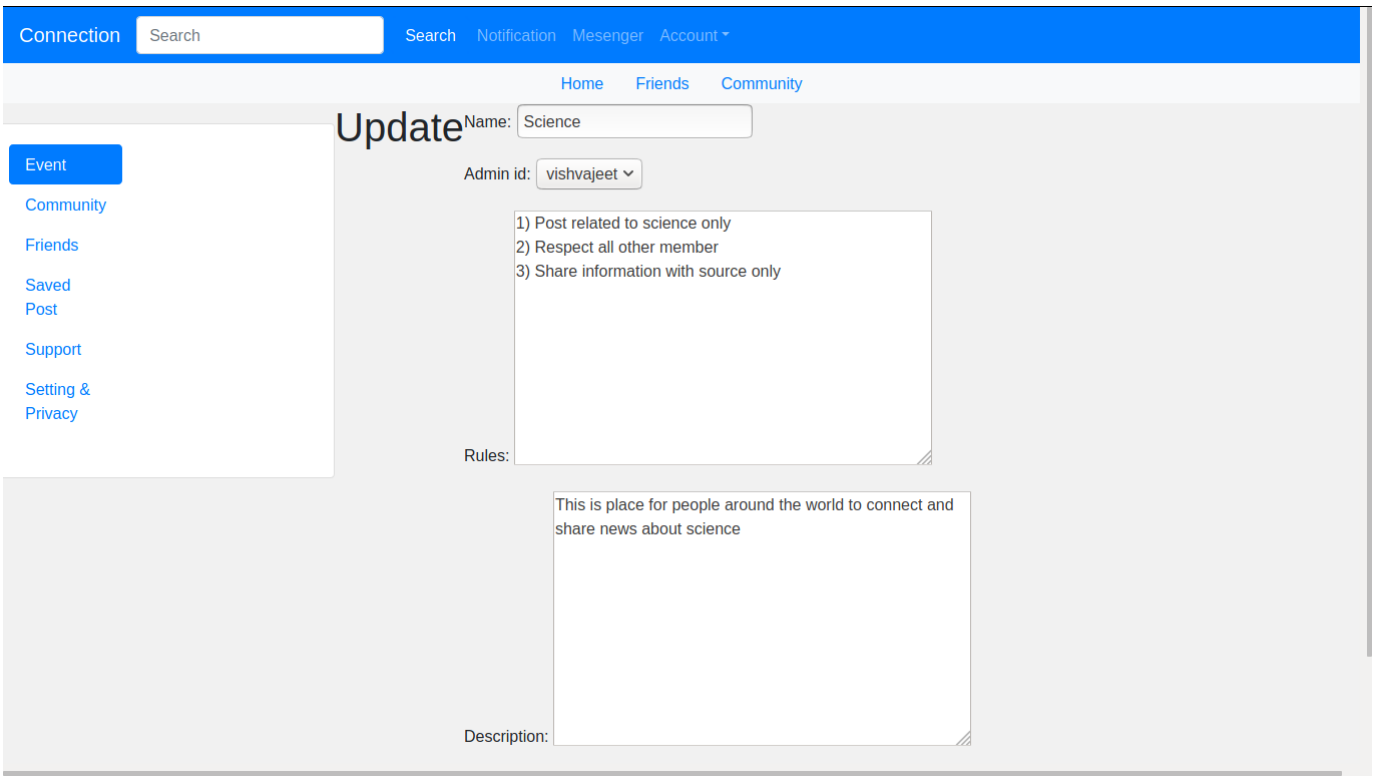
User can update post make by him



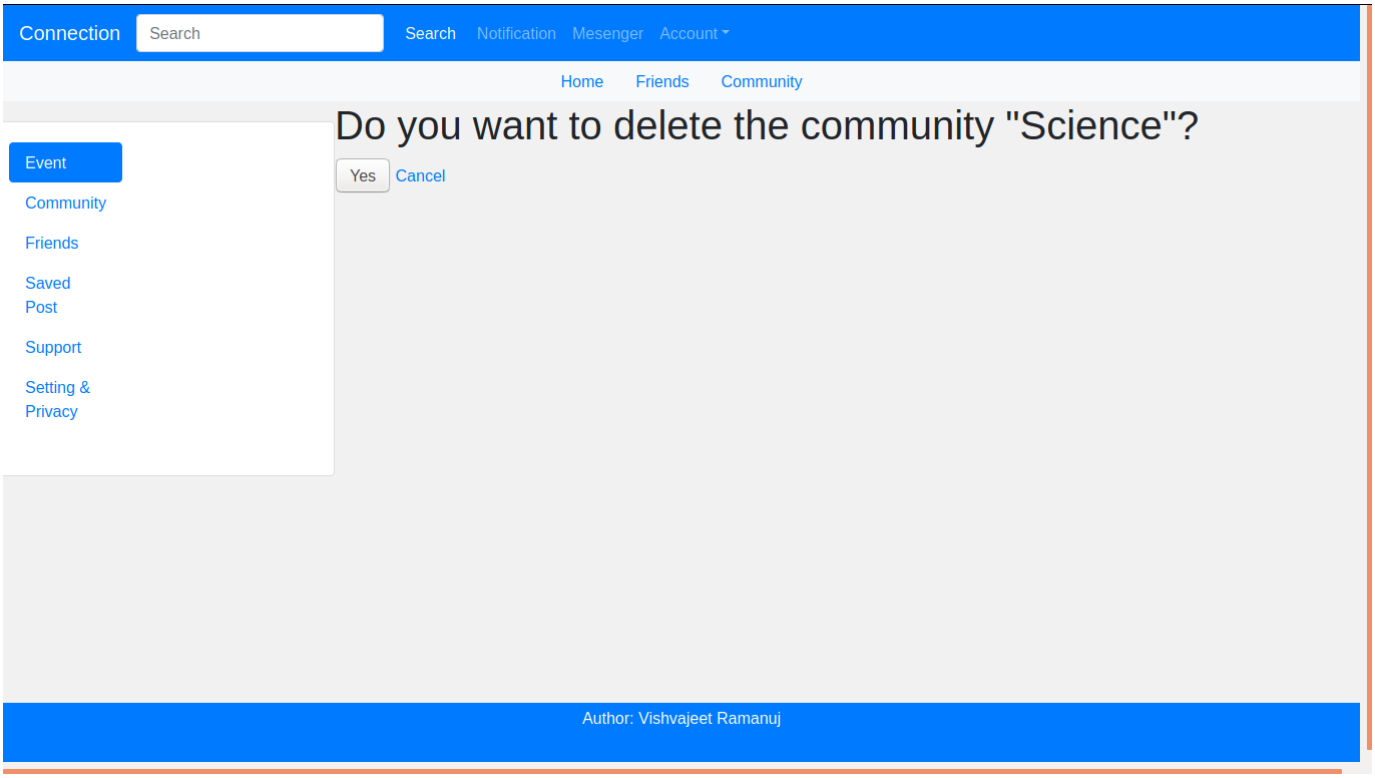
User can create community like this to stay connected with people of same field



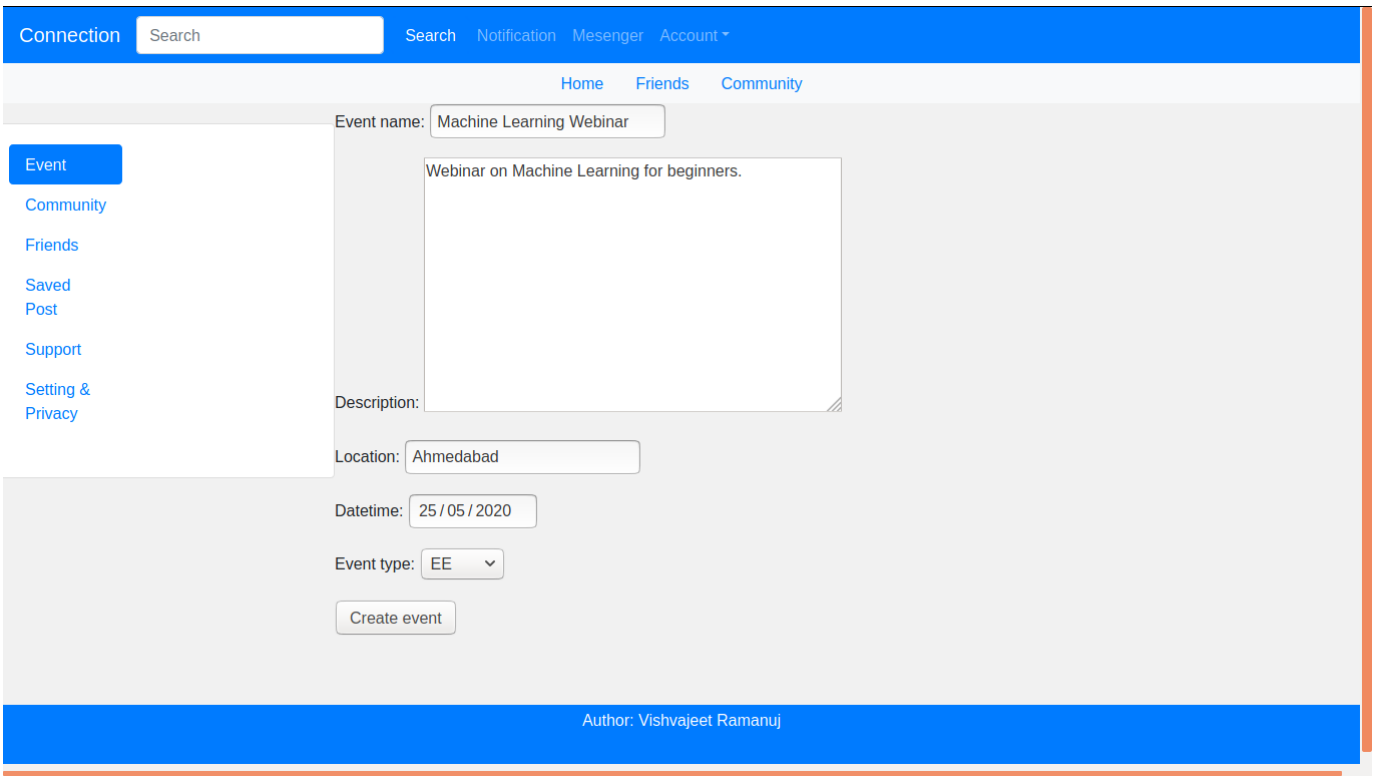
User can watch details of community already exist in platform



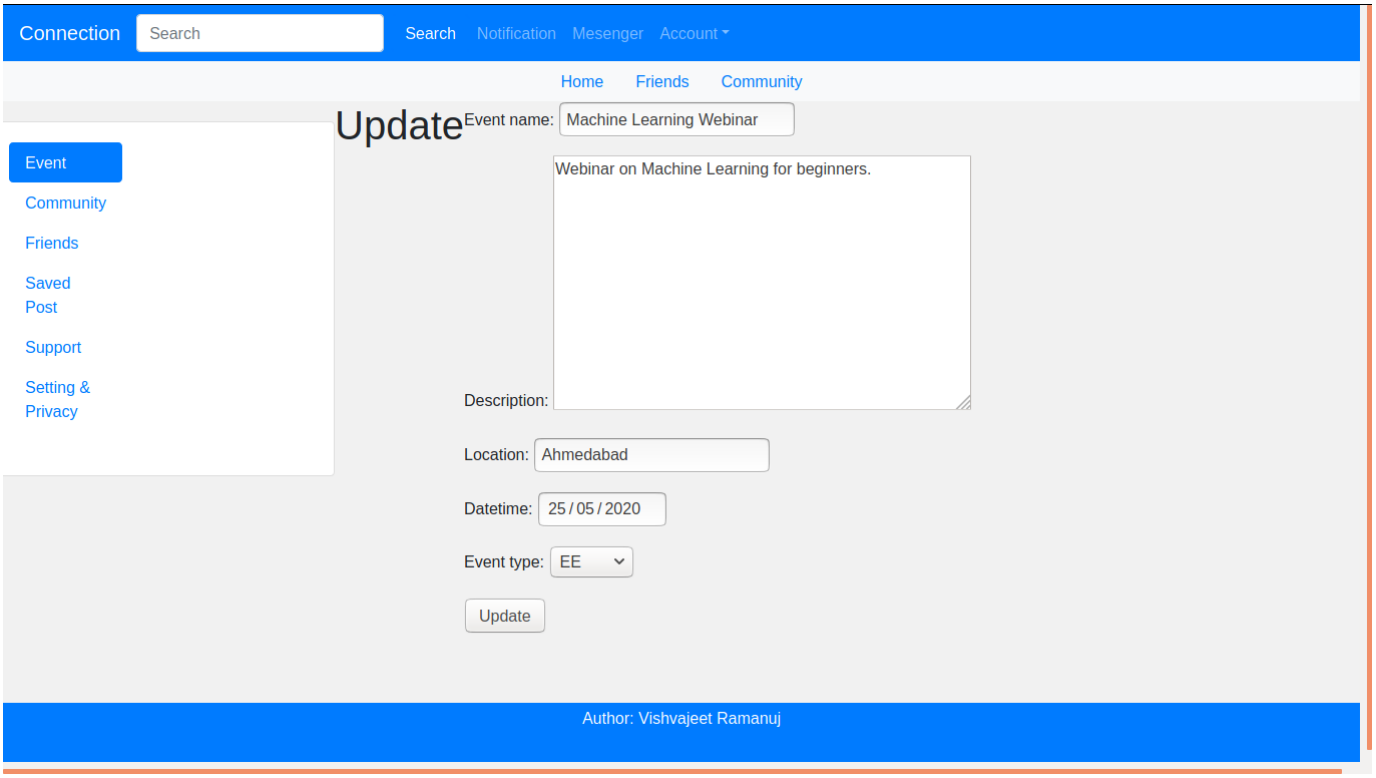
User can update community details created by him



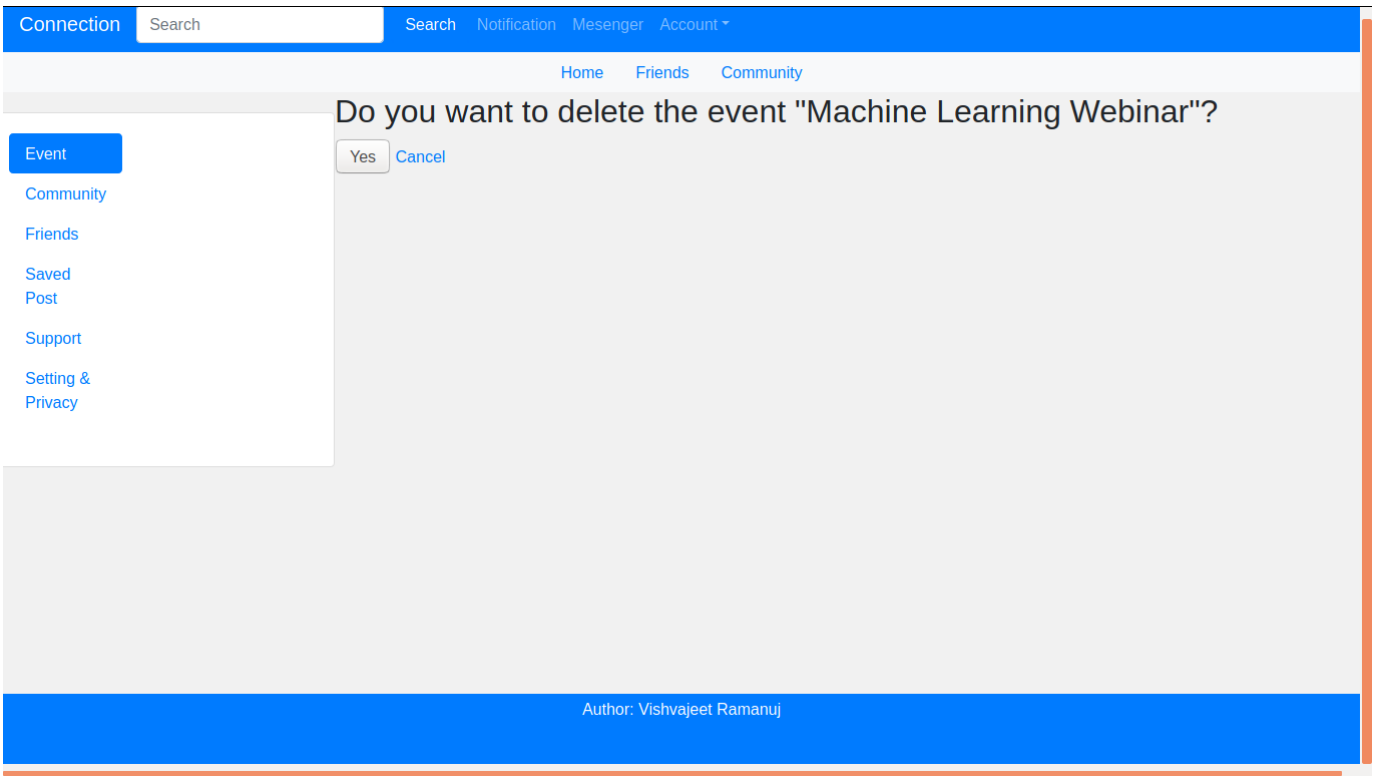
User can delete community created by him.



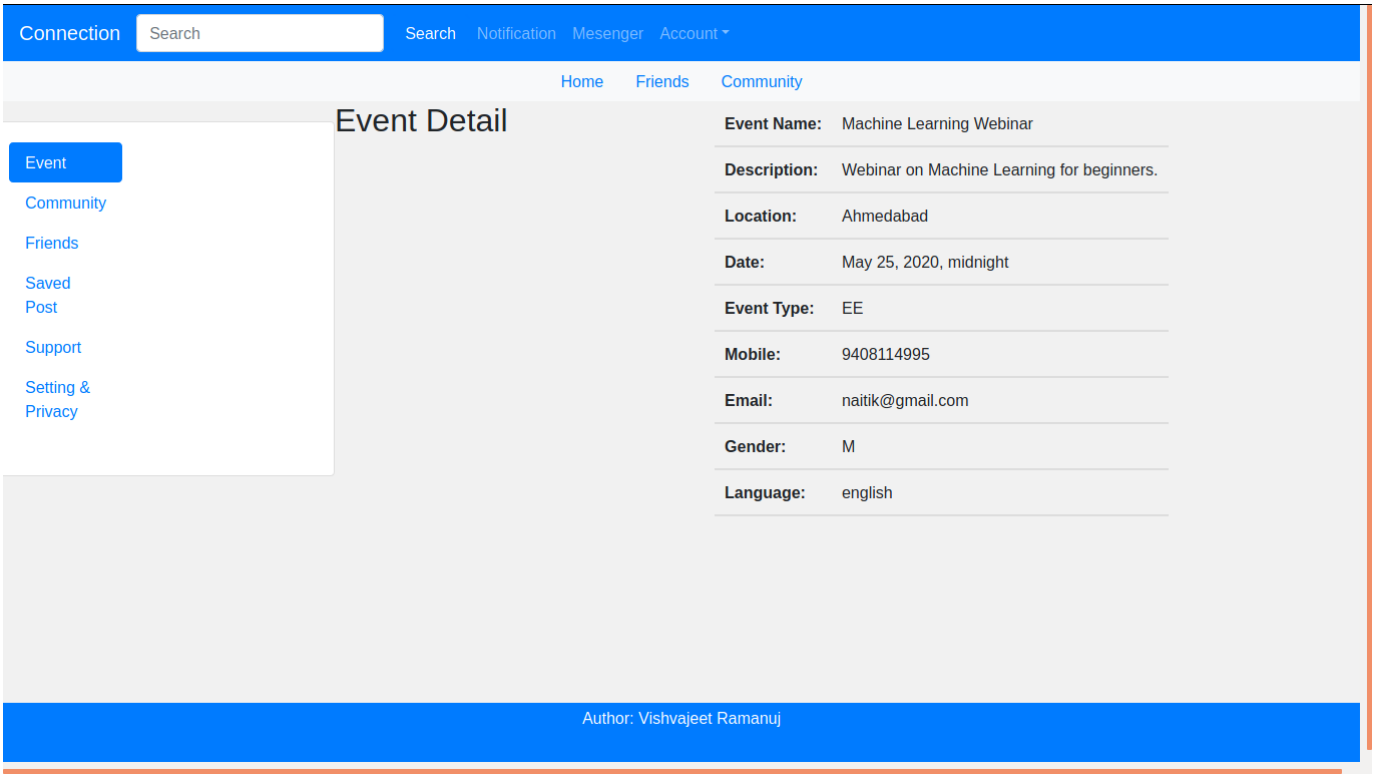
User can create event



User can update event details created by him.



User can delete event created by him.



User can watch details of event available on platform.

Test cases:

Testing is the process or activity that checks the functionality and correctnessofsoftwareaccordingtospecifieduserrequirementsinorderto improve the quality and reliability of system. It is an expensive, time consuming, and critical approach in system development which requires proper planning of overall testing process.

A successful test is one that finds the errors. It executes the program with explicit intention of finding error, i.e., making the program fail. It is a process of evaluating system with an intention of creating a strong system and mainly focuses on the weak are as of the system or software.

Test Strategy:

It is a statement that provides information about the various levels, methods, tools, and techniques used for testing the system. It should satisfy all the needs of an organization.

Test Plan:

It provides a plan for testing the system and verifies that the system under testing fulfills all the design and functional specifications. The test plan provides the following information −

- Objectives of each test phase
- Approaches and tools used for testing
- Responsibilities and time required for each testing activity

- Availability of tools, facilities, and test libraries
- Procedures and standards required for planning and conducting the tests
- Factors responsible for successful completion of testing process

Types of testing conducted on the software:

- Unit testing:

  Also known as Program Testing, it is a type of testing where the analyst tests or focuses on each program or module independently. It is carried out with the intention of executing each statement of the module at least once.

- Functionaltesting:

  Functiontestingdetermineswhetherthesystemisfunctioningcorrectly accordingtoitsspecificationsandrelevantstandardsdocumentation. Functionaltestingtypicallystartswiththeimplementationofth e system,whichisverycriticalforthesuccessofthesystem.

- Integratedtesting:

  In Integration Testing, the analyst tests multiple module working together.Itisusedtofinddiscrepanciesbetweenthesystemandits originalobjective,currentspecifications,andsystemsdocumentatio n.

Coding

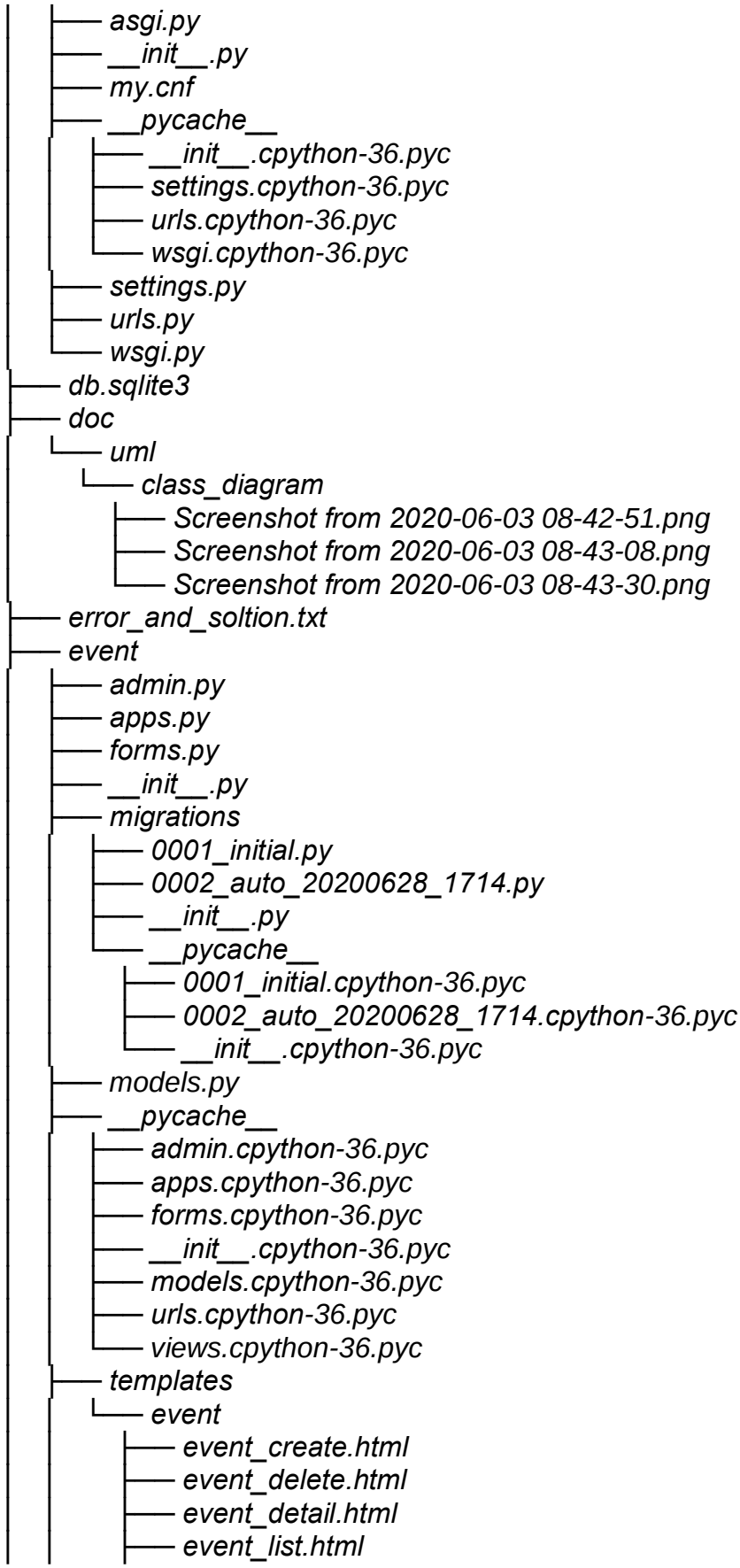
SQL commands for model creation:

Python (Django Framework) model classes:

Directory Structure

```
.
├── activity
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── migrations
│   │   ├── 0001_initial.py
│   │   ├── 0002_activity_user_id.py
│   │   ├── __init__.py
│   │   └── __pycache__
│   │       ├── 0001_initial.cpython-36.pyc
│   │       ├── 0002_activity_user_id.cpython-36.pyc
│   │       └── __init__.cpython-36.pyc
│   ├── models.py
│   ├── __pycache__
│   │   ├── admin.cpython-36.pyc
│   │   ├── apps.cpython-36.pyc
│   │   ├── __init__.cpython-36.pyc
│   │   └── models.cpython-36.pyc
│   ├── tests.py
│   └── views.py
├── advertisement
│   ├── admin.py
│   ├── apps.py
│   ├── forms.py
│   ├── __init__.py
│   ├── migrations
│   │   ├── 0001_initial.py
│   │   ├── 0002_auto_20200628_1714.py
│   │   ├── __init__.py
│   │   └── __pycache__
│   │       ├── 0001_initial.cpython-36.pyc
│   │       ├── 0002_auto_20200628_1714.cpython-36.pyc
│   │       └── __init__.cpython-36.pyc
│   ├── models.py
│   ├── __pycache__
│   │   ├── admin.cpython-36.pyc
│   │   ├── apps.cpython-36.pyc
```

```
        ├── forms.cpython-36.pyc
        ├── __init__.cpython-36.pyc
        ├── models.cpython-36.pyc
        ├── urls.cpython-36.pyc
        └── views.cpython-36.pyc
    ├── templates
    │   └── advertisement
    │       ├── advertisement_create.html
    │       ├── advertisement_delete.html
    │       ├── advertisement_detail.html
    │       └── advertisement_update.html
    ├── tests.py
    ├── urls.py
    └── views.py
├── community
    ├── admin.py
    ├── apps.py
    ├── forms.py
    ├── __init__.py
    ├── migrations
    │   ├── 0001_initial.py
    │   ├── 0002_auto_20200628_1714.py
    │   ├── __init__.py
    │   └── __pycache__
    │       ├── 0001_initial.cpython-36.pyc
    │       ├── 0002_auto_20200628_1714.cpython-36.pyc
    │       └── __init__.cpython-36.pyc
    ├── models.py
    ├── __pycache__
    │   ├── admin.cpython-36.pyc
    │   ├── apps.cpython-36.pyc
    │   ├── forms.cpython-36.pyc
    │   ├── __init__.cpython-36.pyc
    │   ├── models.cpython-36.pyc
    │   ├── urls.cpython-36.pyc
    │   └── views.cpython-36.pyc
    ├── templates
    │   └── community
    │       ├── community_create.html
    │       ├── community_delete.html
    │       ├── community_detail.html
    │       ├── community_list.html
    │       └── community_update.html
    ├── tests.py
    ├── urls.py
    └── views.py
├── connection
```


asgi.py
__init__.py
my.cnf
__pycache__
__init__.cpython-36.pyc
settings.cpython-36.pyc
urls.cpython-36.pyc
wsgi.cpython-36.pyc
settings.py
urls.py
wsgi.py
db.sqlite3
doc
uml
class_diagram
Screenshot from 2020-06-03 08-42-51.png
Screenshot from 2020-06-03 08-43-08.png
Screenshot from 2020-06-03 08-43-30.png
error_and_soltion.txt
event
admin.py
apps.py
forms.py
__init__.py
migrations
0001_initial.py
0002_auto_20200628_1714.py
__init__.py
__pycache__
0001_initial.cpython-36.pyc
0002_auto_20200628_1714.cpython-36.pyc
__init__.cpython-36.pyc
models.py
__pycache__
admin.cpython-36.pyc
apps.cpython-36.pyc
forms.cpython-36.pyc
__init__.cpython-36.pyc
models.cpython-36.pyc
urls.cpython-36.pyc
views.cpython-36.pyc
templates
event
event_create.html
event_delete.html
event_detail.html
event_list.html

```
        └── event_update.html
    ├── tests.py
    ├── urls.py
    └── views.py
├── friend
    ├── admin.py
    ├── apps.py
    ├── __init__.py
    ├── migrations
        ├── 0001_initial.py
        ├── 0002_auto_20200628_1714.py
        ├── __init__.py
        └── __pycache__
            ├── 0001_initial.cpython-36.pyc
            ├── 0002_auto_20200628_1714.cpython-36.pyc
            └── __init__.cpython-36.pyc
    ├── models.py
    ├── __pycache__
        ├── admin.cpython-36.pyc
        ├── apps.cpython-36.pyc
        ├── __init__.cpython-36.pyc
        └── models.cpython-36.pyc
    ├── templates
        └── friend
            └── friend_list.html
    ├── tests.py
    ├── urls.py
    └── views.py
├── manage.py
├── media
    ├── None
        └── flash.jpeg
    └── user
        └── profile_pic
            ├── disha_patani002.jpeg
            ├── disha_patani003.jpeg
            ├── disha_patani132.jpeg
            ├── IMG_20190822_074428.jpg
            └── test01.jpg
├── newsfeed
    ├── admin.py
    ├── apps.py
    ├── forms.py
    ├── __init__.py
    ├── migrations
        ├── 0001_initial.py
        ├── 0002_newsfeed_post_id.py
```

```
        ├── 0003_newsfeed_user_id.py
        ├── __init__.py
        └── __pycache__
            ├── 0001_initial.cpython-36.pyc
            ├── 0002_newsfeed_post_id.cpython-36.pyc
            ├── 0003_newsfeed_user_id.cpython-36.pyc
            └── __init__.cpython-36.pyc
    ├── models.py
    ├── __pycache__
    │   ├── admin.cpython-36.pyc
    │   ├── apps.cpython-36.pyc
    │   ├── __init__.cpython-36.pyc
    │   ├── models.cpython-36.pyc
    │   ├── urls.cpython-36.pyc
    │   └── views.cpython-36.pyc
    ├── templates
    │   └── newsfeed
    │       └── newsfeed_list.html
    ├── tests.py
    ├── urls.py
    └── views.py
├── notification
    ├── admin.py
    ├── apps.py
    ├── __init__.py
    ├── migrations
    │   ├── 0001_initial.py
    │   ├── 0002_notification_user_id.py
    │   ├── __init__.py
    │   └── __pycache__
    │       ├── 0001_initial.cpython-36.pyc
    │       ├── 0002_notification_user_id.cpython-36.pyc
    │       └── __init__.cpython-36.pyc
    ├── models.py
    ├── __pycache__
    │   ├── admin.cpython-36.pyc
    │   ├── apps.cpython-36.pyc
    │   ├── __init__.cpython-36.pyc
    │   ├── models.cpython-36.pyc
    │   ├── urls.cpython-36.pyc
    │   └── views.cpython-36.pyc
    ├── templates
    │   └── notification
    │       └── notification_list.html
    ├── tests.py
    ├── urls.py
    └── views.py
```

```
├── other
│   ├── db_query.py
│   ├── notes.txt
│   └── personal_setting.txt
├── pages
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── migrations
│   │   ├── __init__.py
│   │   └── __pycache__
│   │       └── __init__.cpython-36.pyc
│   ├── models.py
│   ├── __pycache__
│   │   ├── admin.cpython-36.pyc
│   │   ├── apps.cpython-36.pyc
│   │   ├── __init__.cpython-36.pyc
│   │   ├── models.cpython-36.pyc
│   │   ├── urls.cpython-36.pyc
│   │   └── views.cpython-36.pyc
│   ├── static
│   │   └── pages
│   │       └── style.css
│   ├── templates
│   │   └── pages
│   ├── tests.py
│   ├── urls.py
│   └── views.py
├── post
│   ├── admin.py
│   ├── apps.py
│   ├── forms.py
│   ├── __init__.py
│   ├── migrations
│   │   ├── 0001_initial.py
│   │   ├── 0002_auto_20200628_1714.py
│   │   ├── 0003_auto_20200629_0956.py
│   │   ├── __init__.py
│   │   └── __pycache__
│   │       ├── 0001_initial.cpython-36.pyc
│   │       ├── 0002_auto_20200628_1714.cpython-36.pyc
│   │       ├── 0003_auto_20200629_0956.cpython-36.pyc
│   │       └── __init__.cpython-36.pyc
│   ├── models.py
│   ├── __pycache__
│   │   ├── admin.cpython-36.pyc
│   │   ├── apps.cpython-36.pyc
```

```
        ├── forms.cpython-36.pyc
        ├── __init__.cpython-36.pyc
        ├── models.cpython-36.pyc
        ├── urls.cpython-36.pyc
        └── views.cpython-36.pyc
    ├── templates
    │   └── post
    │       ├── comment_create.html
    │       ├── comment_delete.html
    │       ├── comment_detail.html
    │       ├── comment_list.html
    │       ├── comment_update.html
    │       ├── post_create.html
    │       ├── post_delete.html
    │       ├── post_detail.html
    │       ├── post_list.html
    │       └── post_update.html
    ├── tests.py
    ├── urls.py
    └── views.py
├── search
    ├── admin.py
    ├── apps.py
    ├── __init__.py
    ├── migrations
    │   ├── 0001_initial.py
    │   ├── __init__.py
    │   └── __pycache__
    │       ├── 0001_initial.cpython-36.pyc
    │       └── __init__.cpython-36.pyc
    ├── models.py
    ├── __pycache__
    │   ├── admin.cpython-36.pyc
    │   ├── apps.cpython-36.pyc
    │   ├── __init__.cpython-36.pyc
    │   └── models.cpython-36.pyc
    ├── templates
    │   └── advertisement
    │       ├── advertisement_create.html
    │       ├── advertisement_delete.html
    │       ├── advertisement_detail.html
    │       ├── advertisement_update.html
    │       └── user_list.html
    ├── tests.py
    ├── urls.py
    └── views.py
├── sent_emails
```

```
│   └── 20200625-192529-140633007567704.log
├── story
│   ├── admin.py
│   ├── apps.py
│   ├── __init__.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── templates
│   │   └── advertisement
│   │       ├── advertisement_create.html
│   │       ├── advertisement_delete.html
│   │       ├── advertisement_detail.html
│   │       ├── advertisement_update.html
│   │       └── user_list.html
│   ├── tests.py
│   ├── urls.py
│   └── views.py
├── templates
│   ├── base.html
│   ├── community_standerds.html
│   ├── data_policy.html
│   ├── footer.html
│   ├── guest_home.html
│   ├── header.html
│   ├── home.html
│   ├── left_side.html
│   ├── registration
│   │   ├── login.html
│   │   ├── password_reset_complete.html
│   │   ├── password_reset_confirm.html
│   │   ├── password_reset_done.html
│   │   └── password_reset_form.html
│   ├── static
│   │   └── images
│   │       ├── chat.jpeg
│   │       ├── community2.jpeg
│   │       ├── community.jpeg
│   │       ├── friends.jpeg
│   │       └── search_icon.jpeg
│   └── terms_of_service.html
├── temp.py
├── to do.txt
├── todo.txt
└── u_migration
    ├── admin.py
    ├── apps.py
```

```
    ├── __init__.py
    ├── migrations
    │   ├── __init__.py
    │   └── __pycache__
    │       └── __init__.cpython-36.pyc
    ├── models.py
    ├── __pycache__
    │   ├── admin.cpython-36.pyc
    │   ├── __init__.cpython-36.pyc
    │   └── models.cpython-36.pyc
    ├── tests.py
    └── views.py
├── update_for_version2.txt
├── user
    ├── admin.py
    ├── apps.py
    ├── forms.py
    ├── i_learned.txt
    ├── __init__.py
    ├── migrations
    │   ├── 0001_initial.py
    │   ├── __init__.py
    │   └── __pycache__
    │       ├── 0001_initial.cpython-36.pyc
    │       └── __init__.cpython-36.pyc
    ├── models.py
    ├── __pycache__
    │   ├── admin.cpython-36.pyc
    │   ├── apps.cpython-36.pyc
    │   ├── forms.cpython-36.pyc
    │   ├── __init__.cpython-36.pyc
    │   ├── models.cpython-36.pyc
    │   ├── urls.cpython-36.pyc
    │   └── views.cpython-36.pyc
    ├── templates
    │   └── user
    │       ├── advertiser_create.html
    │       ├── advertiser_delete.html
    │       ├── advertiser_detail.html
    │       ├── advertiser_update.html
    │       ├── user_create.html
    │       ├── user_delete.html
    │       ├── user_detail.html
    │       ├── user_list.html
    │       └── user_update.html
    ├── tests.py
    └── urls.py
```

```
│   └── views.py
└── user_setting
    ├── admin.py
    ├── apps.py
    ├── __init__.py
    ├── migrations
    │   ├── 0001_initial.py
    │   ├── __init__.py
    │   └── __pycache__
    │       ├── 0001_initial.cpython-36.pyc
    │       └── __init__.cpython-36.pyc
    ├── models.py
    ├── __pycache__
    │   ├── admin.cpython-36.pyc
    │   ├── apps.cpython-36.pyc
    │   ├── __init__.cpython-36.pyc
    │   └── models.cpython-36.pyc
    ├── templates
    │   └── advertisement
    │       ├── advertisement_create.html
    │       ├── advertisement_delete.html
    │       ├── advertisement_detail.html
    │       ├── advertisement_update.html
    │       └── user_list.html
    ├── tests.py
    ├── urls.py
    └── views.py
```

user

user/templates/user/advertiser_create.html

```
{% extends 'base.html' %}

{% block content %}

<formaction='.'method='post'> {% csrf_token %}
{{ form.as_p }}
<inputtype="submit",value="Sign Up">
</form>

{% endblock %}
```

user/templates/user/advertiser_delete.html

```
{% extends 'base.html' %}

{% block content %}

<formaction='.'method='POST'>{% csrf_token %}
```

```
<h1>Do you want to delete the advertiser "{{ advertiser.username }}"?</h1>
<p><inputtype='submit'value='Yes' /><ahref='../'>Cancel</a></p>

</form>

{% endblock %}
```

user/templates/user/advertiser_details.html

```
{% extends 'base.html' %}

{% block content %}
{% comment %} {{ form.as_p }} need to know how to display form hear {% endcomment %}
{{ advertiser.username }}
{{ advertiser.dob }}
{{ advertiser.country }}
{{ advertiser.location }}
{{ advertiser.profile_pic }}


{% endblock %}
```

user/templates/user/advertiser_update.html

```
{% extends 'base.html' %}

{% block content %}
<h1>Update</h1>
<formaction='.'enctype='multipart/form-data'method='post'> {% csrf_token %}
{{ form.as_p }}
<inputtype="submit",value="Update">
</form>

{% endblock %}
```

user/templates/user/user_create.html

```
{% extends 'base.html' %}

{% block content %}

<formaction='.'method='post'> {% csrf_token %}
{{ form.as_p }}
<inputtype="submit",value="Sign Up">
</form>

{% endblock %}
```

user/templates/user/user_delete.html

```
{% extends 'base.html' %}
```

```
{% block content %}

<formaction='.'method='POST'>{% csrf_token %}
<h1>Do you want to delete the user "{{ user.username }}"?</h1>
<p><inputtype='submit'value='Yes' /><ahref='../'>Cancel</a></p>
</form>

{% endblock %}
```

user/templates/user/user_detail.html

```
{% extends 'base.html' %}

{% block content %}

<div>
<center><imgsrc="{{user.profile_pic.url}}"id="profile_pic"></center>
<h1align="center">{{user.username}}</h1>
</div>

<h2align="center">Intro</h2>
<tablealign="center"cellpadding="10px">
<tr>
<th>Name: </th>
<td>Disha Patani</td>
</tr>
<tr>
<th>username: </th>
<td>{{user.username}}</td>
</tr>
<tr>
<th>Date of Birth: </th>
<td> {{user.dob}} </td>
</tr>
<tr>
<th>Location: </th>
<td> {{user.location}}</td>
</tr>
<tr>
<th>Country: </th>
<td> {{user.country}}</td>
</tr>
<tr>
<th>Mobile: </th>
<td>{{user.mobile}}</td>
</tr>
<tr>
```

```
<th>Email: </th>
<td>{{user.email}}</td>
</tr>
<tr>
<th>Gender: </th>
<td>{{user.gender}}</td>
</tr>
<tr>
<th>Language: </th>
<td>{{user.language}}</td>
</tr>
</table>
<h2align="center">About</h2>
<tablealign="center"cellpadding="10px">
<tr>
<th>Relationship Status: </th>
<td>{{user.relationship_stauts}}</td>
</tr>
<tr>
<th>Hobbies: </th>
<td>{{user.hobbies}}</td>
</tr>
<tr>
<th>Movies: </th>
<td>{{user.movies}}</td>
</tr>
<tr>
<th>TV Shows: </th>
<td>{{user.tv_shows}}</td>
</tr>
<tr>
<th>Books: </th>
<td>{{user.books}}</td>
</tr>
</table>



{% endblock %}
```

user/templates/user/user_list.html

```
{% extends "base.html" %}

{% block content %}
<h2>Users</h2>
<ul>
```

```
{% for user in object_list %}
<li>{{ user.username }}</li>
{% endfor %}
</ul>
{% endblock %}
```

user/templates/user/user_update.html

```
{% extends 'base.html' %}

{% block content %}
<!-- <h1>Update</h1> -->
<divclass="card">
<formaction='.'enctype='multipart/form-data'method='post'> {% csrf_token %}
{{ form.as_p }}
<inputtype="submit",value="Update">
</form>
</div>
{% endblock %}
```

user/admin.py

```
from django.contrib import admin
from django.contrib.auth import get_user_model
from django.contrib.auth.admin import UserAdmin

from .models import User
from .forms import UserModelForm, UserProfileModelForm

classCustomUserAdmin(UserAdmin):
add_form = UserModelForm
# form = UserProfileModelForm
model = User
# list_display = ['username', 'email']

# class WorkPlaceInline(admin.StackedInline):
# model = WorkPlace
# extra = 2

# class UserAdmin(admin.ModelAdmin):
# fieldsets = [
# ('Basic', {'fields':['username']}),
# ]
# inlines = [WorkPlaceInline]

# Register your models here.
admin.site.register(User)

# admin.site.register(CustomUserAdmin)
```

user/app.py

```
from django.apps import AppConfig


class UserConfig(AppConfig):
name ='user'
```

user/forms.py

```
from django import forms
from django.contrib.auth.forms import UserCreationForm, UserChangeForm
from .models import User, Advertiser

# defineing custom widget for rendering html form with that widget
class DateInput(forms.DateInput):
input_type ='date'

class UserModelForm(UserCreationForm):
class Meta:
model = User
fields = [
'username',
'dob',
'location',
'country',
'mobile',
'email',
'gender',
# 'password',
]
widgets = {
'dob': DateInput()
}

class UserProfileModelForm(UserChangeForm):
# class UserProfileModelForm(forms.Form):
class Meta:
model = User
fields = [
'username',
'dob',
'location',
'country',
'mobile',
'email',
'gender',
```

```
'language',
'relationship_status',
'hobbies',
'movies',
'tv_shows',
'books',
'profile_pic',
'cover_pic',
'friends',
'user_community',
]
widgets = {
'dob': DateInput()
}

class AdvertiserModelForm(forms.ModelForm):
class Meta:
model = Advertiser
fields = [
# 'username',
# 'dob',
# 'location',
# 'country',
# 'mobile',
# 'email',
# 'gender'
# 'password',
'balance',
]
widgets = {
'dob': DateInput()
}
```

user/models.py

```
from django.db import models
from django.contrib.auth.models import AbstractUser
from django.urls import reverse
import datetime


# classes for which there is no main role in application but used in Field of Model
class WorkPlace(models.Model):
name = models.CharField(max_length=32)

class Education(models.Model):
degree_name = models.CharField(max_length=32)
University_name = models.CharField(max_length=50)
```

```
start_date = models.DateField()
end_date = models.DateField()

classCityInLived(models.Model):
city = models.CharField(max_length=32)

MALE='M'
FEMALE='F'
OTHER='O'
GENDER= [
(MALE, 'Male'),
(FEMALE, 'Female'),
(OTHER, 'Other'),
]
USER='U'
ADMIN='ADM'
ADVERTISER='ADV'
USER_TYPE= [
(USER, 'User'),
(ADMIN, 'Admin'),
(ADVERTISER, 'Advertiser'),
]

# Create your models here.
# class BaseUser(AbstractUser):
# user_id = models.AutoField(primary_key=True)
# # duser = models.OneToOneField(DUser, on_delete=models.CASCADE)
# # username = models.CharField(max_length=32)
# dob = models.DateField(null=True)
# location = models.CharField(max_length=15) # only cityname
# country = models.CharField(max_length=10)
# mobile = models.DecimalField(max_digits=10, decimal_places=0, null=True)
# # email = models.EmailField()
# # password = models.CharField(max_length=4096)
# gender = models.CharField(max_length=6, choices=GENDER, null=True)
# user_type = models.CharField(max_length=50, choices=USER_TYPE, null=True, default=USER)

# def forgot_password(self):
# pass


# # bellow method will be overriden in subclass
# def create_profile(self):
# bdate = datetime.datetime(1995, 5, 30)
# print('createing user')
# obj = BaseUser(username='vishvajeet', dob=bdate, location='surendranager', country='india',
mobile=1234567890, email='vishvajeet@gmail.com', password='1234', gender='M')
```

```
# obj.save()

# def edit_profile(self):
# pass

# def delete_account(self):
# pass


class User(AbstractUser):
SINGLE='S'
IN_RELETIONSHIP='IR'
MARRIED='M'
RELATIONSHIP_STATUS= [
(SINGLE, 'Single'),
(IN_RELETIONSHIP, 'In a relationship'),
(MARRIED, 'Married'),
]
FREE='F'
PREMIUM='P'
SUBSCRIPTION_STATUS= [
(FREE, 'Free'),
(PREMIUM, 'Premium'),
]
ACTIVE='A'
BLOCKED='B'
USER_STAUS= [
(ACTIVE, 'Active'),
(BLOCKED, 'Blocked'),
]
user_id = models.AutoField(primary_key=True)
# duser = models.OneToOneField(DUser, on_delete=models.CASCADE)
# username = models.CharField(max_length=32)
dob = models.DateField(null=True)
location = models.CharField(max_length=15) # only cityname
country = models.CharField(max_length=10)
mobile = models.DecimalField(max_digits=10, decimal_places=0, null=True)
# email = models.EmailField()
# password = models.CharField(max_length=4096)
gender = models.CharField(max_length=6, choices=GENDER, null=True)
user_type = models.CharField(max_length=50, choices=USER_TYPE, null=True, default=USER)
# work_experiance = models.ManyToManyField(WorkPlace)
# education = models.ManyToManyField(Education)
# place_where_you_lived = models.CharField(max_length=50)
language = models.CharField(max_length=50, blank=True)
relationship_status = models.CharField(max_length=15, choices=RELATIONSHIP_STATUS, blank=True)
family_members = models.ManyToManyField("self", blank=True) # need to add relation also
```

```
hobbies = models.CharField(max_length=150, blank=True)
movies = models.CharField(max_length=150, blank=True)
tv_shows = models.CharField(max_length=150, blank=True)
books = models.CharField(max_length=150, blank=True)
# registration_time = models.DateTimeField(auto_now_add=True)
profile_pic = models.ImageField(upload_to='user/profile_pic', blank=True)
cover_pic = models.ImageField(upload_to='user/cover_pic', blank=True)
# status = # active, deactive
# user_subscribed_community =
# user_pinned_community =
friends = models.ManyToManyField("self", blank=True)
follow = models.ManyToManyField("self", blank=True)
aquaintances = models.ManyToManyField("self", blank=True)
subscription_status = models.CharField(max_length=15, choices=SUBSCRIPTION_STATUS)
subscription_exp_date = models.DateField(auto_now_add=True)
saved_post = models.ManyToManyField("post.Post", blank=True)
# community user joined
user_community = models.ManyToManyField("community.Community", blank=True)

defpay_subscription(self):
pass

defgive_feedback(self):
pass

defupdate_profile(self):
pass

# this method delete account. This method is also overriden
defdelete_account(self):
pass

defadd_details(self):
pass

defget_privacy_setting(self):
pass

defset_privacy_setting(self):
pass

defremove_details(self):
pass

defupdate_profile_pic(self):
pass
```

```
defupdate_cover_pic(self):
pass

# django stuff
def__str__(self):
returnself.username

defget_absolute_url(self):
return reverse("user:user_detail", kwargs={"pk": self.pk})


# class Admin(BaseUser):
# # class method
# def remove_post(self):
# pass

# def deactivate_account(self):
# pass

# def check_feedback(self):
# pass

# # this method is overriden
# def delete_account(self):
# pass

# def remove_story(self):
# pass

# def watch_feedback(self):
# pass

# # django stuff
# # class Meta:
# # verbose_name = _("Admin")
# # verbose_name_plural = _("Admins")

# def __str__(self):
# return self.username

# def get_absolute_url(self):
# return reverse("user:admin_detail", kwargs={"pk": self.pk})

classAdvertiser(models.Model):
user_id = models.AutoField(primary_key=True)
# duser = models.OneToOneField(DUser, on_delete=models.CASCADE)
username = models.CharField(max_length=32)
```

```
email = models.EmailField()
password = models.CharField(max_length=4096)
dob = models.DateField(null=True)
location = models.CharField(max_length=15) # only cityname
country = models.CharField(max_length=10)
mobile = models.DecimalField(max_digits=10, decimal_places=0, null=True)
gender = models.CharField(max_length=6, choices=GENDER, null=True)
user_type = models.CharField(max_length=50, choices=USER_TYPE, null=True, default=USER)
balance = models.DecimalField(max_digits=5, decimal_places=2)

# methods of class
# def post_advertisement(self):
# pass

# def pay_money(self):
# pass

# def give_feedback(self):
# pass

# def delete_account(self):
# pass

# django stuff
# class Meta:
# verbose_name = _("Advertiser")
# verbose_name_plural = _("Advertisers")

def __str__(self):
returnself.username

defget_absolute_url(self):
return reverse("user:advertiser_detail", kwargs={"pk": self.pk})

# class Advertiser(models.Model):
# balance = models.DecimalField(max_digits=5, decimal_places=2)
```

user/urls.py

```
from django.urls import path
from .views import (
UserCreateView, UserUpdateView, UserDeleteView, UserDetailView, UserListView,
AdvertiserCreateView, AdvertiserUpdateView, AdvertiserDeleteView, AdvertiserDetailView)


app_name ='user'
```

```
urlpatterns = [
path('create/', UserCreateView.as_view(), name='user_create'),
path('<int:pk>/update/', UserUpdateView.as_view(), name='user_update'),
path('<int:pk>/delete/', UserDeleteView.as_view(), name='user_delete'),
path('<int:pk>/', UserDetailView.as_view(), name='user_detail'),
path('user_list/', UserListView.as_view(), name='user_list'),

# for advertiser it will be user/advertiser
path('advertiser/create/', AdvertiserCreateView.as_view(), name='advertiser_create'),
path('advertiser/<int:pk>/update/', AdvertiserUpdateView.as_view(), name='advertiser_update'),
path('advertiser/<int:pk>/delete/', AdvertiserDeleteView.as_view(), name='advertiser_delete'),
path('advertiser/<int:pk>/', AdvertiserDetailView.as_view(), name='advertiser_detail'),

]
```

user/views.py

```
from django.shortcuts import render, get_object_or_404
from django.urls import reverse_lazy
from django.views.generic import (
CreateView,
UpdateView,
DeleteView,
DetailView,
ListView
)

from .forms import UserModelForm, UserProfileModelForm, AdvertiserModelForm
from .models import User, Advertiser

# Create your views here.
# User Views
classUserObjectMixin(object):
model = User
defget_object(self):
pk =self.kwargs.get("pk")
obj =None
if pk isnotNone:
obj = get_object_or_404(self.model, pk=pk)
return obj

classUserCreateView(CreateView):
template_name ='user/user_create.html'
model = User
form_class = UserModelForm # this works good but overriden because we need both get and post
```

```
classUserUpdateView(UserObjectMixin, UpdateView):
model = User
template_name ='user/user_update.html'
form_class = UserProfileModelForm
# success_url = reverse_lazy('user:user-detail', kwargs={"pk": self.pk})

classUserDeleteView(UserObjectMixin, DeleteView):
model = User
template_name ='user/user_delete.html'
success_url = reverse_lazy('home-view') # try to replace this static url

classUserDetailView(UserObjectMixin, DetailView):
model = User
template_name ='user/user_detail.html'

classUserListView(ListView):
model = User
template_name ='user/user_list.html'

# Advertiser Views
classAdvertiserObjectMixin(object):
model = Advertiser
defget_object(self):
pk =self.kwargs.get("pk")
obj =None
if pk isnotNone:
obj = get_object_or_404(self.model, pk=pk)
return obj

classAdvertiserCreateView(CreateView):
template_name ='user/advertiser_create.html'
form_class = AdvertiserModelForm

classAdvertiserUpdateView(AdvertiserObjectMixin, UpdateView):
model = Advertiser
template_name ='user/advertiser_update.html'
form_class = AdvertiserModelForm

classAdvertiserDeleteView(AdvertiserObjectMixin, DeleteView):
model = Advertiser
template_name ='user/advertiser_delete.html'
success_url = reverse_lazy('home-view')

classAdvertiserDetailView(AdvertiserObjectMixin, DetailView):
model = Advertiser
template_name ='user/advertiser_detail.html'
```

```
classAdvertiserListView(ListView):
model = Advertiser
template_name ='user/advertiser_list.html'
```

Project tree, coding structure and description:

DjangoframeworkstrictlyfollowstheMVC(ModelViewController) model of applicationdevelopment.

MVC (Model View Controller):

MVC (Model View Controller):

The model-view-controller pattern proposes three main components or objects to be used in software development:

- A ***Model***, which represents the underlying, logical structure of data in a software application and the high-level class associated with it. This object model does not contain any information about the user interface.
- A ***View***, which is a collection of classes representing the elements in the user interface (all of the things the user can see and respond to on the screen, such as buttons, display boxes, and so forth)
- A ***Controller***, which represents the classes connecting the model and the view, and is used to communicate between classes in the model and view.

Main files inside the folder of apps of Django application:

Main app folder

|--- admin.py

|--- apps.py

|--- forms.py

|--- models.py

|--- tests.py

|--- urls.py

|--- views.py

|---__init__.py

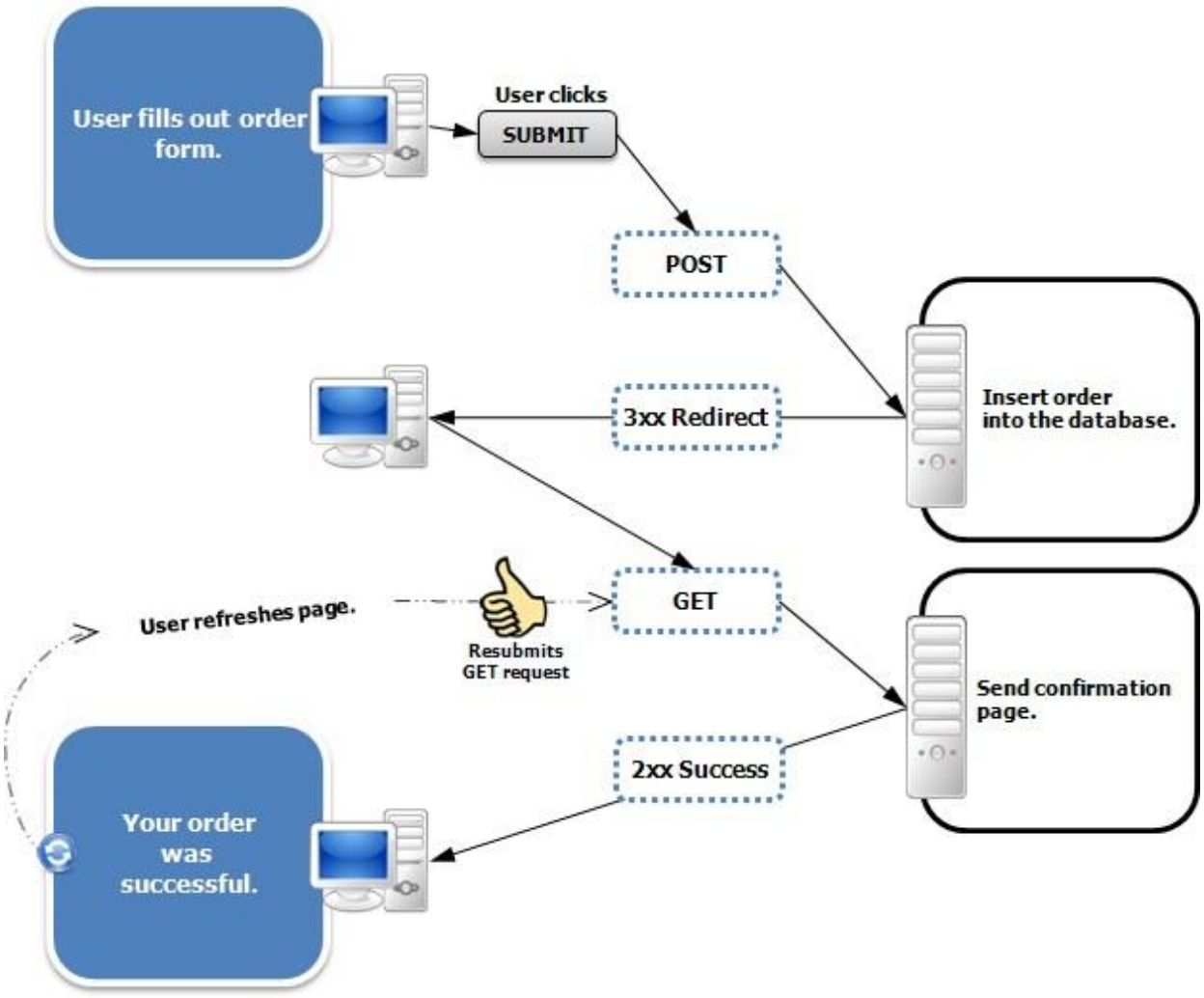
Each folder of a django project can either be considered as an individual app or a cluster of methods comprising of a bigger app. A single app folder can serve single form / template or multiple forms / templates depending on the application design.

Post/Redirect/Get (PRG):

Post/Redirect/Get (**PRG**) is a web development design pattern that prevents some duplicate form submissions, creating a more intuitive interface for user agents (users). PRG supports bookmarks and the refresh button in a predictable way that does not create duplicate form submissions.

When a web form is submitted to a server through an HTTP POST request, a web user that attempts to refresh the server response in certain user agents can cause the contents of the original POST request to be resubmitted, possibly causing undesired results, such as a duplicate web purchase.

Multiple post causing error if PRG is not applied



Standardization

Code efficiency:

Code efficiency is a broad term used to depict the reliability, speed and programming methodology used in developing codes for an application. Code efficiency is directly linked with algorithmic efficiency and the speed of runtime execution for software. It is the key element in ensuring high performance. The goal of code efficiency is to reduce resource consumption and completion time as much as possible with minimum risk to the business or operating environment. The software product quality can be accessed and evaluated with the help of the efficiency of the code used.

The software is intended to be run mostly on local computer network setup only. The software runs heavily on javascript and backend processing in case of bigger entries. The requests and responses have been packaged into JSON objects, so it may not be reliably fast over a WAN internet connection. Moreover if multiple users are accessing the software, it is better that the load of application is distributed in the local connection only

Being an accounting software the algorithm utilized to generate the software logic is complex. And for the same the modules of the software have been divided into multiple vouchers and the entry within vouchers can be made by dynamic forms which make extensive use of AJAX and JSON to post data to backend database.

Most of the validation has been included on the client side javascript so that there is no additional load on the server to process the validation of

forms in case of incorrect entries or invalid transaction limitations.

e.g. - The current cash balance is stored in a local variable instance so whenever a cash transaction is being made, if the transaction value is greater than the available cash balance the software will print an error saying there is insufficient cash balance.

Recommendations for code efficiency include:

- To remove unnecessary code or code that goes to redundant processing
- To make use of optimal memory and nonvolatile storage
- To ensure the best speed or run time for completing the algorithm
- To make use of reusable components wherever possible
- To make use of error and exception handling at all layers of software, such as the user interface, logic and data flow

- To create programming code that ensures data integrity and consistency
- To develop programming code that's compliant with the design logic and flow

Error handling:

Error handling refers to the response and recovery procedures from error conditions present in a software application. In other words, it is the process comprised of anticipation, detection and resolution of application errors, programming errors or communication errors. Error handling helps in maintaining the normal flow of program execution. In fact, many applications face numerous design challenges when considering error-handling techniques.

The following errors were encountered and rectified while creating the project.

Logical errors:

Logical errors are difficult to find, because in most cases they do not show any messages in compiler, console or screen. It is only at a later stage when generating reports we can figure out the logical errors.

The benefit in the case of accounting software has been that the accounting software follows a set of rules to arrive at various amounts based on formula. Accounting softwares follow a dual-entry system, meaning every debit entry should have a corresponding credit entry. This essentially makes the the ledgers self balancing and it becomes easy to pin point any discrepancies.

Compile-time errors:

The python programming language does not need one to manually compile the program source code so as to deploy. While running the project

the compiled files are automatically created and kept in the same folder where the source files exist.

The console is used to run the django application server. If any compilation error is generated during compilation, the console prints out a compilation error.
If there is an error in a python source file, the errors are printed on the console.
But if there is an error in the template file (HTML template), the error is printed out in the browser while loading the respective page.

Runtime errors:

Runtime errors are caused while the program is in execution mode. The program is successfully running after compiling when the runtime errors occur. In most cases the runtime errors are an outcome of invalid of incorrect inputs provided. Validation of forms help in avoiding a lot of runtime errors, but some inputs can creep in which cause a runtime error.

There are other types of runtime errors as well, like memory leaks. In the case of an accounting project, the memory requirement increases in proportion to the amount of transaction that the company records.

Another runtime error encountered was missing object error of the database. In this case an error catching function was set where if the program was unable to fetch a database object, instead of returning a NULL object, the program returns zero.

Parameter calling:

Calling a function in Python:

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code.

Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Following is the example to call printme() function −

Decorators in python:

In Python, everything is object. Functions in Python are first-class objects which means that they can be referenced by a variable, added in the lists, passed as arguments to another function etc.

Decorators are callable objects which are used to modify functions or classes.

Callable objects are objects which accepts some arguments and returns some objects. *functions* and *classes* are examples of callable objects in Python.

Function decorators are functions which accepts function references as arguments and adds a wrapper around them and returns the function with the wrapper as a new function.

e.g. - The following is an example of a decorator used in the project. It checks if a company has been registered in order to view the requested page or not.

In the get method we use *@is_comp_registered* to call the function is_comp_registered on the current view.

Similarly in the case of *@is_rights_given* the employee will be only provided access to the pages he is authorized to. If the user tries to open a url by the address bar the application will throw the user to a not authorized page. The difference with this decorator is that it takes in an argument of a primary key which lets decide, based on the user, whether to give the user rights to the page or not.

Validation checks:

A validation check ascertains that the value (or data) input into a computer is valid. Validation checks are performed automatically by a computer to ensure that entered data is correct and reasonable.

Required field validation:

A required field validation check is commonly used in filling out online forms. It determines whether or not a user has filled out the important fields in a form.

In the software, javascript checks for unfilled ‘input’ fields and highlights the field in red. Unless all the required fields are filled, the javascript library will not allow the form to be submitted to the backend.

Range validation:

A range validation check determines whether or not an entered value falls within a predetermined range. For instance, if a text box allows values that fall within a set range -- say, from 5 to 10 -- all user-entered values that fall outside this range are not accepted.

In the software, for purchase and sales entry, in the fields of ‘units’ and ‘per unit’ fields, HTML field validations are set to type=’number’ and step=’1’ and

step=’0.01’ respectively because units can only be in whole integers and the per unit price can be in decimals with upto 2 points indicating rupees and paise.

Numeric & Text validations:

Numeric validation checks determine whether or not data entered in a field or a text box has a numeric value (1, 2, 3, etc.), an alphabetic value (a, b, c, etc.) or an alphanumeric value (which includes symbols: &, %, #, etc.). Numeric validation checks allow users to only enter numeric values, and a pop-up alerts the user if he enters values other than numbers.

In the software, batch numbers are set to accept 8 characters of alphanumeric batch numbers. Alphabets in capital only and it accepts no special characters.

Below is the javascript code to sanitize the batch input code

Testing

Testing techniques:

Testing technique refers to the method or way to test a software or a part of the software. Each testing technique has its own benefits. Different techniques target different types of defects. So, it would be wrong to call one technique best. Based on the software and its requirements, one test technique may suit better than the other to serve the purpose. And sometimes, a combination of different testing technique might be a good way to test a software.

Black box testing:

The technique of testing without having any knowledge of the interior workings of the application is called black-box testing. The tester is oblivious to the system architecture and does not have access to the source code.

Typically, while performing a black-box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

White box testing:

White-box testing is the detailed investigation of internal logic and structure of the code. White-box testing is also called **glass testing** or **open- box testing**. In order to perform **white-box** testing on an application, a tester needs to know the internal workings of the code.

The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

Grey box testing:

Grey-box testing is a technique to test the application with having a limited knowledge of the internal workings of an application. In software testing, the phrase the more you know, the better carries a lot of weight while testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black-box testing, where the tester only tests the application's user interface; in grey-box testing, the tester has access to design documents and the database. Having this knowledge, a tester can prepare better test data and test scenarios while making a test plan.

Ad hoc testing:

Adhoc testing is an informal testing type with an aim to break the system. This testing is usually an unplanned activity. It does not follow any test design techniques to create test cases. In fact is does not create test cases altogether! This testing is primarily performed if the knowledge of testers in the system under test is very high. Testers randomly test the application without any test cases or any business requirement document.

Ad hoc Testing does not follow any structured way of testing and it is randomly done on any part of application. Main aim of this testing is to find defects by random checking. Adhoc testing can be achieved with the testing technique called **Error Guessing.** Error guessing can be done by the people having enough experience on the system to "guess" the most likely source of errors.

This testing requires no documentation/ planning /process to be followed.

Since this testing aims at finding defects through random approach, without any documentation, defects will not be mapped to test cases. Hence, sometimes, it is very difficult to reproduce the defects as there are no test steps or requirements mapped to it.

Combined testing carried out on The Book-keeper accounting application:

For the most part of the software testing ad hoc technique was used, as an exhaustive system knowledge was required in order to test the software errors and to rectify and debug them as and when they occur.

But at the same time the software was provided to users of different category, i.e. the ones with system and software knowledge and the ones with exclusive accounting knowledge.

The whitebox testing was conducted by software specialists, who in turn helped to find the system and validation errors. Whereas the blackbox testing was conducted by finance and accounting specialists who checked for any logical, conceptual or principle error in the financial reports generated from the software.

Testing Plan:

Test planning, the most important activity to ensure that there is initially a list of tasks and milestones in a baseline plan to track the progress of the project. It also defines the size of the test effort.

Although there is no official documentation of test plans, the rough notes of test cases and test plans were built upon the following identifiers.

Test plan identifiers:

S.No.	**Parameter**	**Description**
1	Test plan identifier	Unique identifying reference.
2	Introduction	A brief introduction about the project and to the document.
3	Test items	A test item is a software item that is the application under test.
4	Features to be tested	A feature that needs to tested on the testware.
5	Features not to be tested	Identify the features and the reasons for not including as part of testing.
6	Approach	Details about the overall approach to testing.
7	Item pass/fail criteria	Documented whether a software item has passed or failed its test.

8	Test deliverables	The deliverables that are delivered as part of the testing process,such as test plans, test specifications and test summary reports.
9	Testing tasks	All tasks for planning and executing the testing.
10	Environmental needs	Defining the environmental requirements such as hardware, software, OS, network configurations, tools required.
11	Responsibilities	Lists the roles and responsibilities of the team members.
12	Staffing and training needs	Captures the actual staffing requirements and any specific skills and training requirements.
13	Schedule	States the important project delivery dates and key milestones.
14	Risks and Mitigation	High-level project risks and assumptions and a mitigating plan for each identified risk.
15	Approvals	Captures all approvers of the document, their titles and the sign off date.

Test planning activities:

- To determine the scope and the risks that need to be tested and that are NOT to be tested.
- Making sure that the testing activities have been included.
- Deciding Entry and Exit criteria.
- Evaluating the test estimate.
- Planning when and how to test and deciding how the test results will be evaluated, and defining test exit criterion.

Debugging and code improvement:

Debugging, in computer programming and engineering, is a multistep process that involves identifying a problem, isolating the source of the problem, and then either correcting the problem or determining a way to work around it. The final step of debugging is to test the correction or workaround and make sure it works.

The debugging process:

In software development, debugging involves locating and correcting code errors in a computer program. Debugging is part of the software testing process and is an integral part of the entire software development lifecycle.
The debugging process starts as soon as code is written and continues in successive stages as code is combined with other units of programming to form a software product.

Developing software programs undergo heavy testing, updating, troubleshooting and maintenance. Normally, software contains errors and bugs, which are routinely removed. In the debugging process, complete software programs are regularly compiled and executed to identify and rectify issues.

Code improvement for python:

Python has developed a reputation as a solid, high-performance language. Lots has been done in recent years to get to this point. Python has helped simplify the list by not predefining the data type that should go into the array.

Instead python list are different from arrays in other programming languages. Python lists can hold different data types at once within the list, be it integer, string, floating point number, dictionary element or even another list object. Lists can be nested within another list.

Another benefit of using python programming language is that the programmer does not have to explicitly define variable data types before declaring them. Python takes care of it based on the value provided to the variable. This helps in further compacting the code and encouraging reusability.

Decorators in python has helped in resusing code that is redundant throughout the software. In python, you can pass in a function inside another function as an argument and the wrapper function within the function will return the appropriate result object. All that is required in python is to call that function by the '@' handle before the function name and above the function on whom the the function has to be called.

e.g. If the function name is somefunc(), to call it inside anotherfunc(), the following is the syntax.

System Security Measures

Database / data security:

Database security refers to the collective measures used to protect and secure a database or database management software from illegitimate use and malicious threats and attacks.

It is a broad term that includes a multitude of processes, tools and methodologies that ensure security within a database environment.

Database security:

Database security covers and enforces security on all aspects and components of databases. This includes:

- Data stored in database
- Database server
- Database management system (DBMS)
- Other database workflow applications

Authentication:

In mysql database one has to provide a username and password to connect to the database using a connector plugin. By default the mysql server runs on port number 3306. If the database is hosted on another server or system, then the connection string should point to the named instance of that server or the IP of that server.

The connection string in python to connect to mysql database is as mentioned below.

Database constraints:

Various constraints were used in the software to keep the database in sync with the design of the software.

Like all the transactions in the application are linked to a 'TrnCtr' table in the database. The field 'TrnID' is a primary key which is linked to every other table and contains a corresponding foreign key in the table.

In all the database tables where the id is referenced, a constraint has been put, when the 'TrnID' is removed from the table 'TrnCtr', all the corresponding entries will also be deleted.

This constraint helps in removing all the entries of transaction that have been made pertaining to an accounting transaction.

There are also constraints of cascading update in the database. The names of ledgers and stock have been kept to be unique and referenced in foreign keys. So when the names are altered, the application changes the name respectively throughout the database in all the tables wherever the name is being referenced.

Django application creates default constraints in the application if the developer has not explicitly instructed the application otherwise. To avoid this when declaring model classes the developer has to use the attribute 'db_constraint=False' for every corresponding foreign key entry.

In the book keeper application constraints were manually entered with the help of raw SQL commands. These constraints help in maintaining the integrity of the data entered.

All the DB constraints used in the software:

<u>User profiles and access rights:</u>

If the database is hosted on a server where there are more than one database and there are different clients that are connected to the server, security of individual databases becomes important. To ensure that the clients are able to access only their respective databases, a user for every client or database must be created with a strong password.

The respective user must be given access only to his database schema. For the application to run properly, it is essential to provide all insert, update and delete rights

to the user in the database as these authentication and validation is taken care of at the application level.

User creating and rights allocation in MySQL:

To create a new user in mysql, first one has to login as the root user.

If phpmyadmin is being used, that application provides an intuitive user interface to create a user which is easy to understand and create.

But if the developer wants to create a user from the command line SQL, the following commands must be executed:

The above command will create a new user in the database. Now in our application we want to assign all rights to the respective user so to assign the rights to the user the following command is executed.

Different rights provided in mysql database systems:

- ALL PRIVILEGES- as we saw previously, this would allow a MySQL user full access to a designated database (or if no database is selected, global access across the system)
- CREATE- allows them to create new tables or databases
- DROP- allows them to them to delete tables or databases
- DELETE- allows them to delete rows from tables
- INSERT- allows them to insert rows into tables
- SELECT- allows them to use the SELECT command to read through databases
- UPDATE- allow them to update table rows
- GRANT OPTION- allows them to grant or remove other users' privileges

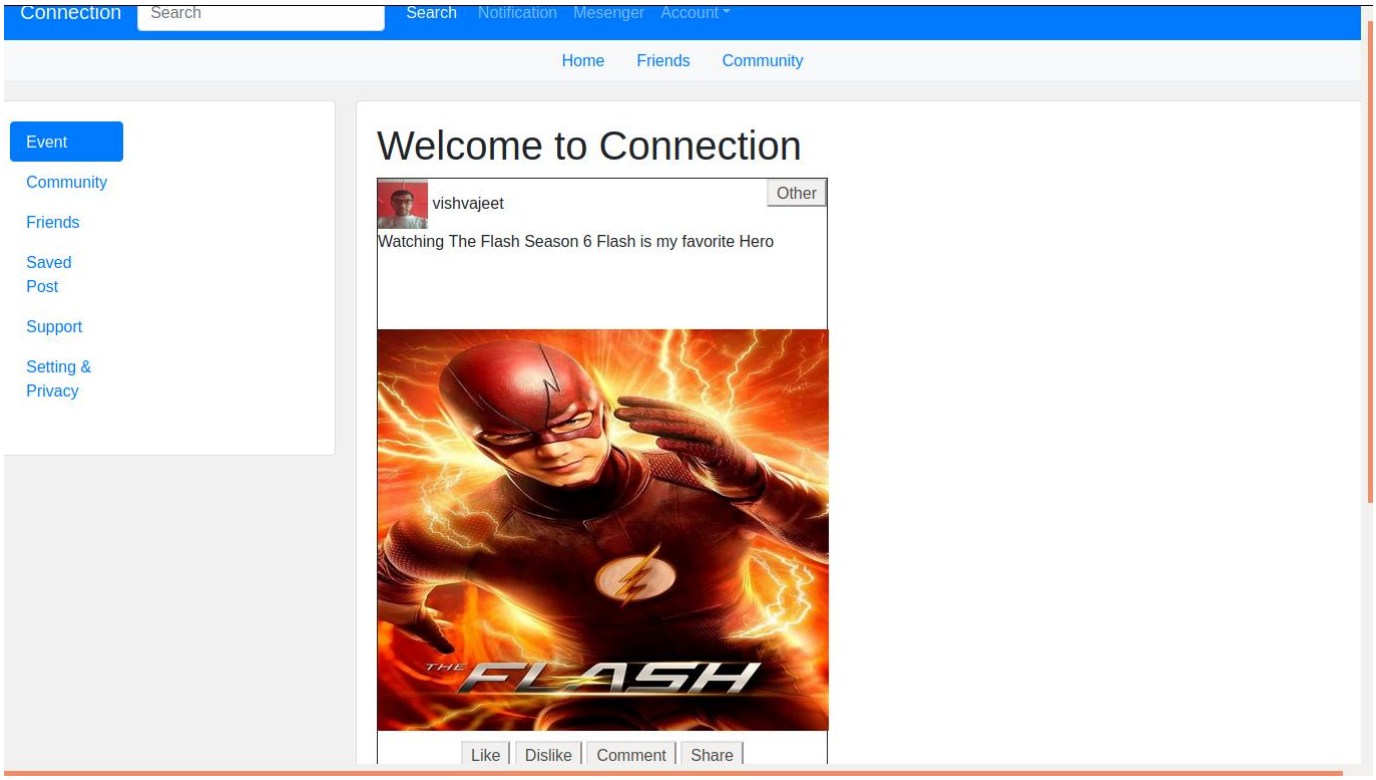
Cost estimation of the project

FOSS (Free and open source) projects gained relevance in the late 1990s. Since then, the amount of companies involved in FOSS projects has been growing, and new collaborations have emerged. While during its beginnings FOSS was mostly developed by volunteers, nowadays the collaboration between developers is more varied, and projects range from those still developed only by volunteers, to those which are based on the collaboration of companies with volunteers (e.g., GNOME, Linux), to the clear industrial ones, in which the main driving force are companies.

Being an FOSS, the application is free to be installed, modified and re-distributed. The main focus of any FOSS software is the freedom to modify and reuse the the software. Being a scalable application, if any user feels the need to add any additional modules, he can hire a development team or request to the community of developers for a change in the software. This in turn makes the software more modular and relevant to the needs of the client.

Reports

Because the nature of this application. It does not generate reports. But what user see is result of process so it is one kind of report so screenshot of this is attached hear



Future scope of the project

Current version of connection is more like prototype. Functionality of this can be extended to compete with other social network giants. Many features added for mining

data and which can be used to help user in their day to day life. Finding people like them and much more

Bibliography

- Leif Azzopardi and David Maxwell, *How to Tango with Django - Release 1,* pub. yr. 2016
- Julia Elman and Mark Lavin - O'Reilly Media, Inc., *Lightweight Django,* pub.yr. 2015
- https://www.youtube.com channels
  - Codebites
  - CodingEntrepreneurs
  - Derek Banas
  - Get Set Python
  - LearnWebCode
  - LevelUp Tuts
  - MIT OpenCourseWare

- edX | Online courses from the world's best universities

  https://www.edx.org

- Understanding CSS Writing Methodologies – Hongkiat,

  http://www.hongkiat.com/blog/css-writing-methodologies/

- Code School - Try jQuery,

  http://try.jquery.com/

- Coding For Entrepreneurs,

  https://www.codingforentrepreneurs.com/

- Stack Overflow

  https://www.stackoverflow.com

- Django documentation | Django

  https://docs.djangoproject.com/

- Django Save Form Data to Database and Retrieve It

  https://www.simplifiedpython.net/django-save-form-data-to-database/

- Python Django Tutorials

  http://djangobook.com/

- Dynamic web pages with Django, Ajax and jQuery | Racing Tadpole

  http://racingtadpole.com/blog/django-ajax-and-jquery/

- How to Render Django Form Manually

  https://simpleisbetterthancomplex.com/article/2017/08/19/how-to- render-django-form-manually.html

- Django with Ajax, a modern client-server communication practise – IMPYTHONIST

  https://impythonist.wordpress.com/2015/06/16/django-with-ajax-a- modern-client-server-communication-practise/

- Chosen: A jQuery Plugin by Harvest to Tame Unwieldy Select Boxes

  https://harvesthq.github.io/chosen/

- Bootstrap + Chosen

  http://alxlit.name/bootstrap-chosen/

- Writing view decorators for Django - Passing Curiosity

  https://passingcuriosity.com/2009/writing-view-decorators-for-django/

- Django and AJAX Form Submissions - say 'goodbye' to the page refresh - Real Python

  https://realpython.com/blog/python/django-and-ajax-form-submissions/

- Computer Glossary, Computer Terms - Technology Definitions and Cheat sheets

  https://whatis.techtarget.com/

Glossary

application program:

a program designed to perform a specific function directly for the user or, in some cases, for another application program.

bug:

a coding error in a computer program.

build:

a version of a program, usually pre-release, and identified by a build number, rather than by a release number. As a verb, to build can mean either to write code or to put individual coded components of a program together.

data modeling:

the analysis of data objects that are used in a business or other context and the identification of the relationships among these data objects.

debugging:

the process of locating and fixing or bypassing bugs (errors) in computer program code or the engineering of a hardware device.

development environment:

the set of processes and programming tools used to create the program or software product.

development process:

a set of tasks performed for a given purpose in a software development project.

entity-relationship diagram:

a data modeling technique that creates a graphical representation of the entities, and the relationships between entities, within an information system.

Gantt chart:

a horizontal bar chart frequently used in project management that provides a graphical illustration of a schedule that helps to plan, coordinate, and track specific tasks in a project.

human factors:

the study of how humans behave physically and psychologically in relation to particular environments, products, or services.

iterative:

describes a heuristic planning and development process where an application is developed in small sections called iterations.

object-oriented programming:

a programming model organized around objects rather than actions and data rather than logic, based on the idea that what we really care about are the objects we want to manipulate, rather than the logic required to manipulate them..

open source:

describes a program whose source code is made available for use or modification as users or other developers see fit.

PERT chart:

(Program Evaluation Review Technique)

a project management tool used to schedule, organize, and coordinate tasks within a project developed by the U.S. Navy in the 1950s.

project planning:

a discipline for stating how to complete a project within a certain timeframe, usually with defined stages, and with designated resources.

prototyping:

a systems development method (SDM) in which a prototype (an early approximation of a final system or product) is built, tested, and then reworked as necessary until an acceptable prototype is finally achieved from which the complete system or product can now be developed.

runtime:

when a program is running.

user acceptance testing:

a phase of software development in which the software is tested in the "real world" by the intended audience.

user interface:

everything designed into an information device with which a human being may interact -- including display screen, keyboard, mouse, light pen, the appearance of a desktop, illuminated characters, help messages, and how an application program or a Web site invites interaction and responds to it.

web services:

services made available from a business's Web server for Web users or other Web-connected programs.

Synopsis Aproval with biodata of guide

(17)

Indira Gandhi National Open University

इंदिरा गांधी राष्ट्रीय मुक्त विश्वविद्यालय

Maidan Garhi, New Delhi-110068, INDIA

MASTER OF COMPUTER APPLICATIONS
[MCA 6th Semester]

MCSP-060

Enrolment No - 175430667

MCA
SYNOPSIS
2019 – 2020

SCHOOL OF COMPUTER AND INFORMATION SCIENCES
IGNOU, MAIDAN GARHI, NEW DELHI – 110 068

II. PROFORMA FOR THE APPROVAL OF MCA PROJECT PROPOSAL (MCSP-060)

(Note: All entries of the proforma of approval should be filled up with appropriate and complete information. Incomplete proforma of approval in any respect will be summarily rejected.)

Project Proposal No :.....
(for office use only)

Enrolment No.: 175430667
Study Centre: Ahmedabad
Regional Centre:..... **RC Code:**.....
E-mail: Vishvajeet_Ramanuj
Mobile/Tel No.: 85300 67675

1. Name and Address of the Student: Vishvajeet Ramanuj
2. Title of the Project***: Connection - The Social Network
3. Name and Address of the Guide: Jignesh Solanki, L.D. Arts College, Ahmd
4. Educational Qualification of the Guide (Attach bio-data also)

Ph.D*	M.Tech.*	B.E*/B.Tech.*	MCA	M.Sc.*
☐	☐	☐	☑	☐

(*in Computer Science / IT only)

5. Working / Teaching experience of the Guide**: Over 23+ Years in IT & Academic field

(**Note: At any given point of time, a guide should not provide guidance for more than 5 MCA students of IGNOU)

6. Software used in the Project***: Python, Django
(*** Please refer to section VIII of these guidelines)
7. If already pursued BCA/BIT from IGNOU, mention the title of the project (CS-76) and the s/w used.....N.A.
8. Project title of the Mini Project (MCS-044) and the s/w used... EyeOfAI - python, dlib
9. Is this your first submission? ☑ Yes ☐ No

V.M. Ramanuj
Signature of the Student
Date: 30-12-19

J. Solanki
Signature of the Guide
Date: 30-12-19

For Office Use Only

☑ Approved ☐ Not Approved

Name: Dr. Hardik Jardosh
Asst. Prof. Gujarat Univ. Ahmedabad
Signature. Designation. Stamp of the Project Proposal Evaluator
Date: 10/01/2020

Suggestions for reformulating the Project:
- Formulate proper SRS
- Improve DFD

ORACLE Certified Professional

JAYESH SOLANKI

Email: solankijayesh@gmail.com

PERSONAL PROFILE:

Name: JAYESH SOLANKI **Date of Birth:** January 3, 1968

Voice: 079-26306155(O) 9825028565(M)

Email ID: solankijayesh@gmail.com Website: www.jayeshsolanki.in

EDUCATION:

M.C.A. from Rollwala Computer Centre Gujarat University, 1991
B.Sc. from ST Xavier's College, Gujarat University, 1988
M.C.A. = *Master of computer Applications*, B.Sc. = *Bachelor of Sciences*
Pursing **Ph. D**. in Computer Science using Blockchain Technology

CERTIFICATIONS:

- **Oracle 10g/9i/8i/7 DBA Certified from Oracle Corporation**
  (OCP ID: SP2569635 - Oracle Corporation)

SUMMARY:

A senior level computer academician and software professional with **25 years of IT and Academic** experience in a variety of responsible roles such as Associate professor, Academic Consultant, Program Co-ordinator, Officer on special duty, Technical advisor, Database/System Administrator, Programmer/System analyst. Technically sound in **Database Design, Administration of Oracle Database, Web technology** and administration using different Operating Systems.Proficiency in deployment of technology in various project related works, General administration, Co-ordination, Teaching, Counselling, problem solving etc.

PRESENT STATUS:

1) Working for L. D. Arts College as a **Head of the Computer Department cum Associate Professor.**

2) Associated with Indira Gandhi National Open University (IGNOU), Study Centre (0901), Ahmedabad as a **Programme (Assistant) Co-ordinator** IT related programs for last twenty-three years.

3) Working for Gujarat University as Officer on Special Duty and Technical advisor for online admission project for last six years.

HONORS & KEY ACHIEVEMENTS:

1) Received **CICC** (Center of the International Co-operation for Computerization) **International Scholarship** in **Information Technology** supported by Association for Overseas Technical Scholarship **(AOTS)** under the co-operation of **Japanese Government (Ministry of International Trade and Industry)** for attending 'Network/**Internet System Development Course'** from Sep. 2003 to Nov. 2003 in **Tokyo, Japan.**

2) Received an invitation to participate in World convention from AOTS, Japan two time in the year of 2014 & 2019 **in Tokyo**.

3) Elected as a senate member of Gujarat University for the term 2011-2015.

4) Elected as Secretary of Computer Society of India, Ahmedabad chapter and selected later as Vice Chairman and Chairman.
5) Invited as panel members in the event / conference organized by prestigious organization Federation of Indian Chambers of Commerce and Industry (FICCI) and InfoComm India in Mumbai.
6) Invited to participate the event held in Delhi on January 26, 2015 on behalf of representative of Gujarat and got an opportunity to have photograph with Hon'ble Prime minister of Japan Mr. Shinzo Abe.

CERTIFICATE OF APPRECIATION

- Received the appreciation certificate by Hon'ble Vice Chancellor of Gujarat University in August 2016 for outstanding services provided & valuable contribution towards successful implementation of online admission projects for UG/PG courses.
- Recognised generous contributions in a CSI (Computer Society of India) as Secretary & Chairman of Ahemdabad chapter.

JOB PROFILE:

Sr. No.	Name of the Organization	Period From – To	Designation	Nature of work
1.	L. D. Arts College	Dec. 1994 – till date (25 yrs.)	Head-Computer & Associate Professor	Teaching, Database, System and Network administration, System Design and Development
2.	Indira Gandhi National Open University	Jan. 1995 - till date (23 yrs.)	Program Co-ordinator	Co-ordination, Counselling, teaching & Consultation, System Design & Development, Administration (Databases/Systems/Networks), User training etc.
3	Vivekanad College	Nov. 1992 To Nov. 1994 (2 yrs.)	Programmer	Programming, Teaching
4	Mastek Pvt. Ltd., Mumbai	Dec 1991 To Nov. 1992 (1 yrs.)	Engineer	Support & Programming

TECHNICAL SKILLS:

Hardware: IBM Compatibles, VAX 11/730

Operating Systems: Red Hat Linux V10.x/9.x, UNIX - SUN OS 5.7 Solaris (2.x), SCO Open Server Rel. 5.x, Cent O.S. , Windows 2012/2008.2000 Advanced Server, Windows 98, VAX/VMS, Novell NetWare V(3.12 & 4.11), MS DOSV6.x

Languages: PHP, C, Java, Cobol, FoxPro

ORACLE Certified Professional

JAYESH SOLANKI

Email: solankijayesh@gmail.com

Technology:	Web & Blockchain Technology
Scripting Languages:	HTML, VB Script, Java Script, Shell scripts (csh, bsh, ksh)
RDBMS:	Oracle V11/10/9.x/8.x/7.x, SQL Server V2008, MS Access 2010
Front end Development	Visual Studio 2010 (.NET 4.5 & ASP.NET), Visual Basic 6.0, Oracle Forms & Reports,
Tools:	Crystal Reports v10, MS-Front Page 2000
Networking & Protocols:	TCP/IP, IPX/SPX, NETBUI
Web Server:	Oracle 9iAS Portal, Oracle Application Server 4.x, IIS 5.0 and Apache
Concepts:	System Analysis and Design, Software Engineering

TECHNICAL TRAINING:

- Oracle 10g DBA – New features & Oracle 9i DBA from Oracle Education
- Participated in several training programs on various topics

ACADEMIC/PROFESSIONAL ACHIEVEMENTS:

- Head of the Computer Department Cum **Associate Professor** of L. D. Arts College for last twenty-four years.
- **Programme Co-ordinator** of **IGNOU**, study Centre (0901), Ahmedabad for twenty-three years.
- Elected as a **senate member** of Gujarat University for the term 2011-2105
- Appointed as Hon. Secretary for CSI, Ahmedabad Chapter for the term 2012-14.
- Selected as a Vice President & Secretary of AOTS (Association for Overseas Technical Scholarship), Gujarat Chapter
- Appointed as a member of Executive Committee for **Computer Society of India** (CSI), A'bad chapter for six years (three terms) and active participation in various activities since my student's tenure.
- Appointed as a **Consultant** (Computer) of Regional Centre, **Indira Gandhi National Open University (IGNOU)**, Ahmedabad.
- Appointed as an **Academic Counselor** in **IGNOU** for lasts twenty-three years
- Certified as an **Oracle DBA (Database Administrator)** in Oracle 10g/9i/8i/7 from **Oracle Corporation**.
- Member of various committee of our organization. i.e. Admission, Examination, Advisory, UGC projects, Development of computer Centre etc.
- Appointed as a Chairman of Ad-hoc board of Computer Science in Gujarat University.
- Introduced **Computer Subject** as **Main Subject** in **Arts** Stream of **Gujarat University and**
- **Working for InfoComm India as Volunteer to support the Summit cum Exhibition on Audio – Video Technology and Education based conference in Mumbai for last six years.**
- Appointed as a **Chairman / Examination Co-ordinator** for the examination of (Computer Science) in Gujarat University for various courses.
- Appointed as a member of constitution committee for UGC's project 'Computer Networking & Up-gradation of Computer Centre' at Gujarat University.

- Appointed as Consultant at Regional office of IGNOU, Ahmedabad and implemented the structured cabling (Network project) of storied building including 50 computers in network.
- Appointed as a member of Local Inquiry committee for computer courses in Gujarat University.
- **Appointed as Project Guide and evaluator** in **IGNOU** and **Gujarat University** for IT related programs and provided guidance to several students.
- Appointed as a **Radio Counsellor** to give a talk on air on Computer subject through Gyanvani channel.
- Appointed as Assistant Co-ordinator and counselor in Dr. Baba Shaheb Ambedkar Open University
- Appointment as a paper setter and examiner in several Universities like Gujarat University, IGNOU, BAOU and Guj. Vidyapith for under and post graduate courses for IT related programs.
- Selected as member of the Editorial board for publishing Newsletter of CSI for Ahmedabad chapter for lasts two years.
- Represented and attended Annual Convention and several conferences at different levels.
- Invited as panel member in prestigious conference conducted by FICCI & InfoComm India.
- Serving as volunteer for InfoComm India to support the event on Exhibition cum Summit on Digital Audio-Video Technology organize in each year in Mumbai
- Appointed as representative of Hon'ble Vice Chancellor and Panel expert for conducting interview of various programs of Gujarat University.
- Appointed as core member of various Local Inquiry Committee for granting affiliation to various colleges of Gujarat University.
- Appointed as Advisor of School of Multimedia centre of Gujarat University.

Life Membership:

- Computer Society of India.
- AOTS-JAPAN(Association of Overseas Technical Scholarship).
- Google Business Group.
- Ahmedabad and Internet Society (ISOC).

Publication of Papers: Recently, I have presented my accepted paper titled on "Comparative Study Indian Electoral Reforms in Indian Context" at IEEE conference during the International Conference on "Issues and Challenges in Intelligent Computing Techniques" (ICICT-2019) on September 27-28, 2019 in New Delhi organized by Krishna Institute of Engineering & Technology (KIET). Several Papers on different topics like Rural Health Development in IT way, E-Governance, Value based Higher Education etc.

PUBLICATIONS:

- Published several Newsletter as member of Editorial board of CSI, Ahmedabad..
- Prepared material for the computer students in on several topics.

PARTICIPATION OF WORKSHOP/SEMINAR:

Participated actively in several Orientation program, Refresher course, Conference, Symposiums Seminars/Workshops, Faculty Development program at State/National/International level on different topics.

PROJECT GUIDANCE

- Guided number of students for preparing project synopsis and report as Project Guide in Gujarat University & IGNOU.

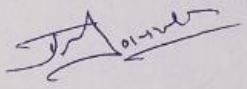
- Evaluated several projects of various technologies of students & professionals.

CONTRIBUTIONS AND SERVICES OFFERED TO IGNOU

- Providing service as Asst. Co-ordinator and academic Counsellor for last twenty-three years
- Effective administration & Co-ordination
- IT related programs run consistently by 23+ years.
- Build reputation and image of centre through effective support and services
- Implemented Digitized service for our learners for effective communication and record Keeping of assignments.
- Providing e-support services through implementation IT based project OCMS Web based and Mobile App which has become fruitful and useful to learners of IGNOU.

References:

1) Prof. A. U. Patel, Advisor KCG-Government of Gujarat (Former Vice-Chancellor, Guj. Uni.)
2) Dr. M. N. Patel, Advisor of Parul University (Former Vice Chancellor) of Gujarat University

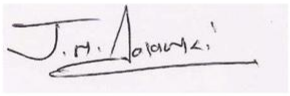


Certificate of Originality

XI. CERTIFICATE OF ORIGINALITY

This is to certify that the project report entitled Connection – The Social Network submitted to **Indira Gandhi National Open University** in partial fulfilment of the requirement for the award of the degree of **MASTER OF COMPUTER APPLICATIONS (MCA) ,** is an authentic and original work carried out by Mr./ Ms. Vishvjitdas Mangaldas Ramanuj with enrolment no. 175430667 under my guidance.

The matter embodied in this project is genuine work done by the student and has not been submitted whether to this University or to any other University / Institute for the fulfilment of the requirements of any course of study.



.....

Signature of the Guide

Date: 30-06-2020..

Name and Address of the student

Jayesh Solanki

L. D. Arts College

.....

.....

Signature of the Studnet

Date: .....

Name, Designation and Address of the Guide:

.....

.....

.....

.....

Enrolment No...175430667.....

Synopsis

Connection – The Social Network

Contents

Introduction and Objective of Project

In this era of Technology importance of social network is increasing day by day. The Populer Social networking site are providing good service but threre are certain limitations. Like user have to watch adds compulsory, privacy and other ethical issues are also threre. There is also lack of competition in this field which cause big compay running social network be dictatorship.

Purspose of making this is to provide alternative social network which give option to pay subscription instead of watching adds compulsory. This also has some other feature which does not seen in popular social networking website. Like dislike button to the things which they don't like

Project Category – Object Oriented Programming / Web Application

Tools/Platform, Hardware and Software Requirement Specification

Software Requirement

It is platform independent .It can run on windows xp and above or linux or mac os.
Web browser

Hardware Requirement

- Intel Pentium 4 processor
- RAM (1 GB)
- Device Storage (200 MB)

Project Definition, Requirement Specification, Project Planning and Scheduling

Project Definition

Purpose of making this is to provide alternative social network which give option to pay subscription instead of watching adds compulsory. This also has some other feature which does not seen in popular social networking website. Like dislike button to the things which they don't like

In this era of Technology importance of social network is increasing day by day. The Popular Social networking site are providing good service but there are certain limitations. Like user have to watch adds compulsory, privacy and other ethical issues are also there. There is also lack of competition in this field which cause big company running social network be dictatorship.

Requirement

- There is no alternative to some popular network. Such as Facebook. Theoretically there are other networks but they does not provide the kind of service Facebook provide. It is well understood that when there is no alternative competition then monopolystarts doing bad things.
- User need to connect with each other. Sometime distance are on the other side of the globe. The connection social network will help people to connect with each other
- People usually have lots of different characteristics. It is very hard for people to find other people which have same interest. Social Network will help people to find people with same interest.
- Users can create community. Power of Community is more than power of individual all combine. People can stay in touch with each
- People need to communicate with each other. The Connection provide this facility. It Provide secure communication channel for user so that user can communicate without being in fear to compromise their privacy,

Gantt chart

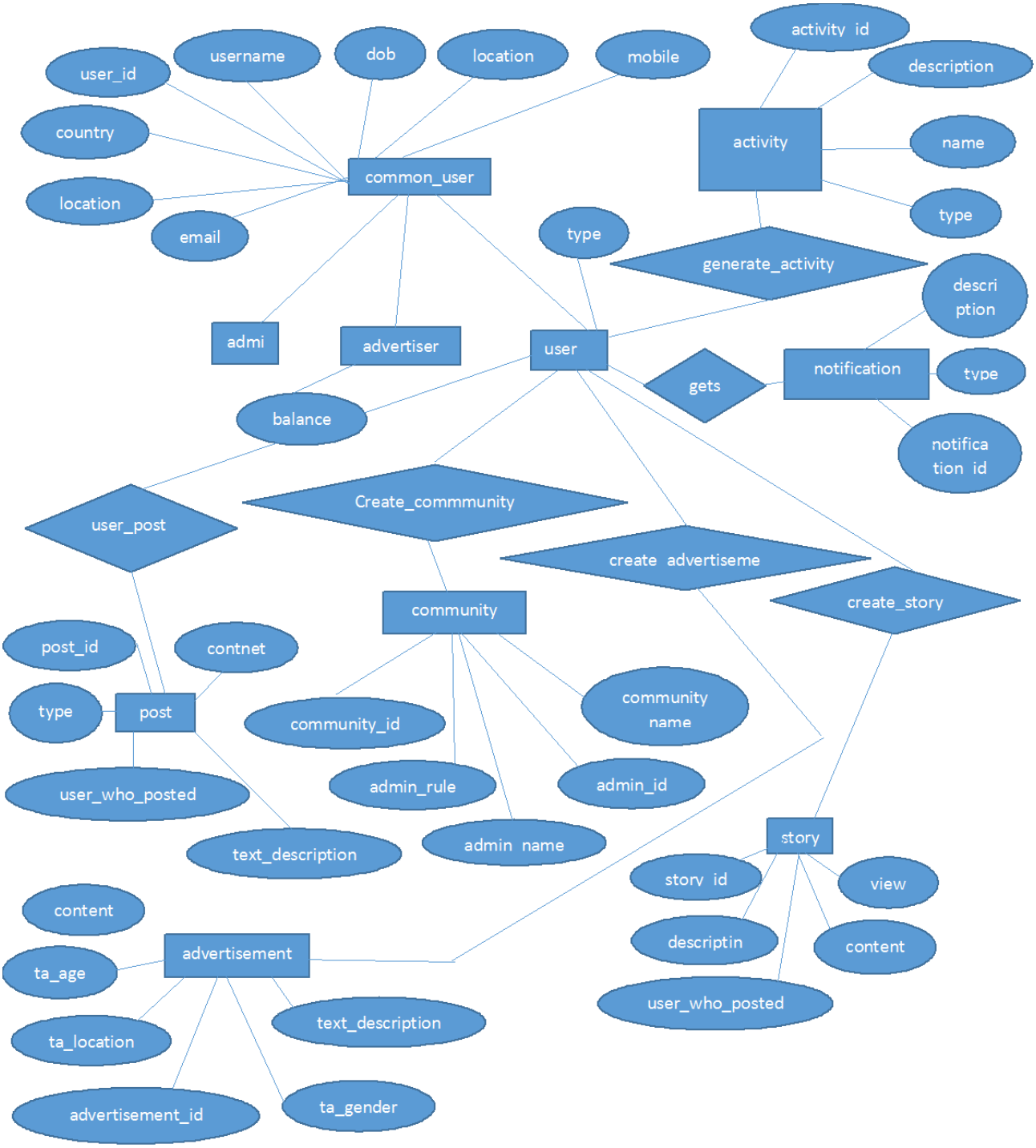
Task	January	February	March	April	May
Requirement analysis					
Design					
Coding Module -user_module					
Coding Module – service_module					
Coding Module – db_connection_module					
Coding Module – search_module					
Integration					
Debugging					
Testing (white box)					
Testing (Black box)					

scope of the solution

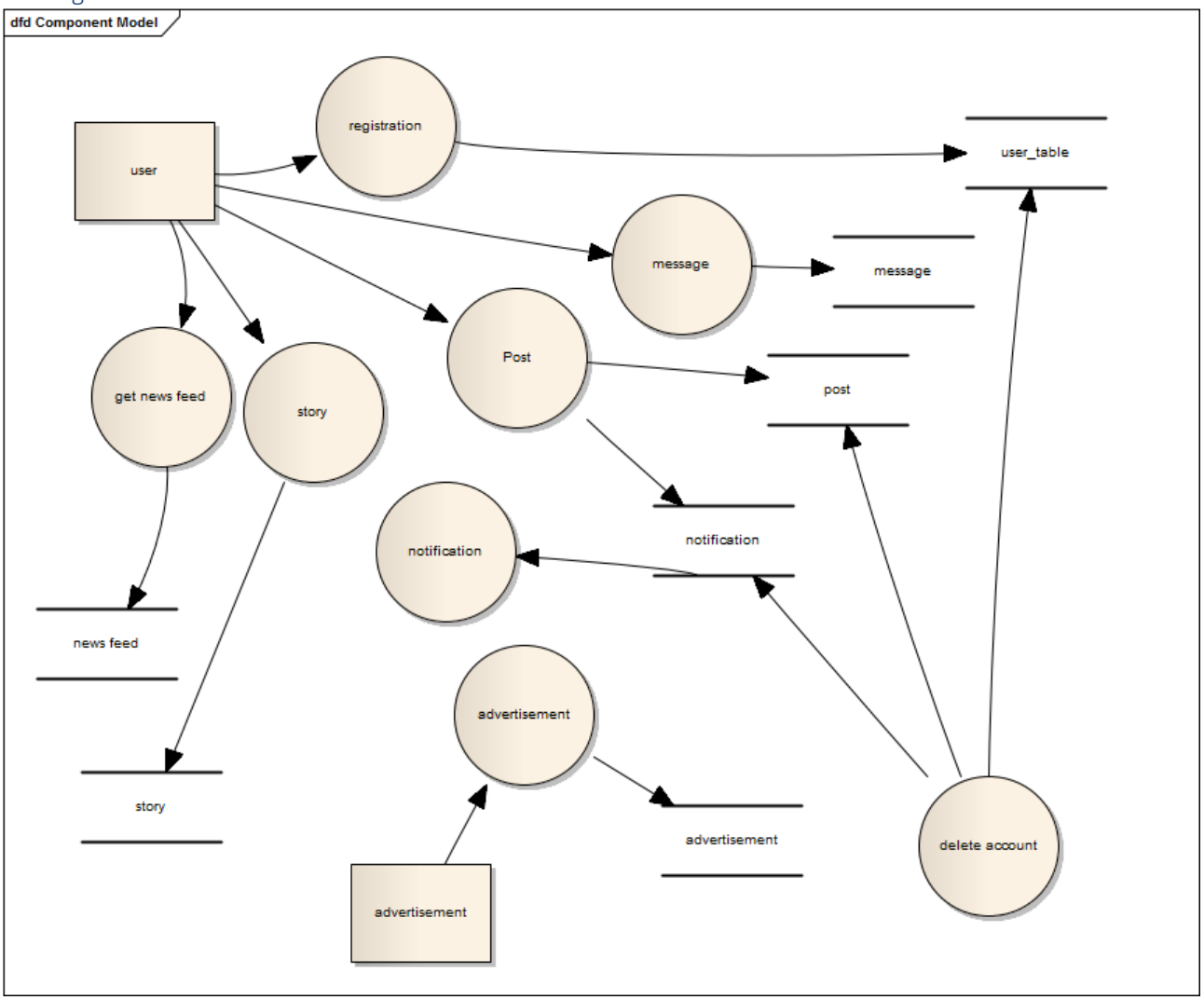
Project Defination

This project is at aim to provide alternative solution of the social network where people does not need to compulsorily watch adds.

Analysis

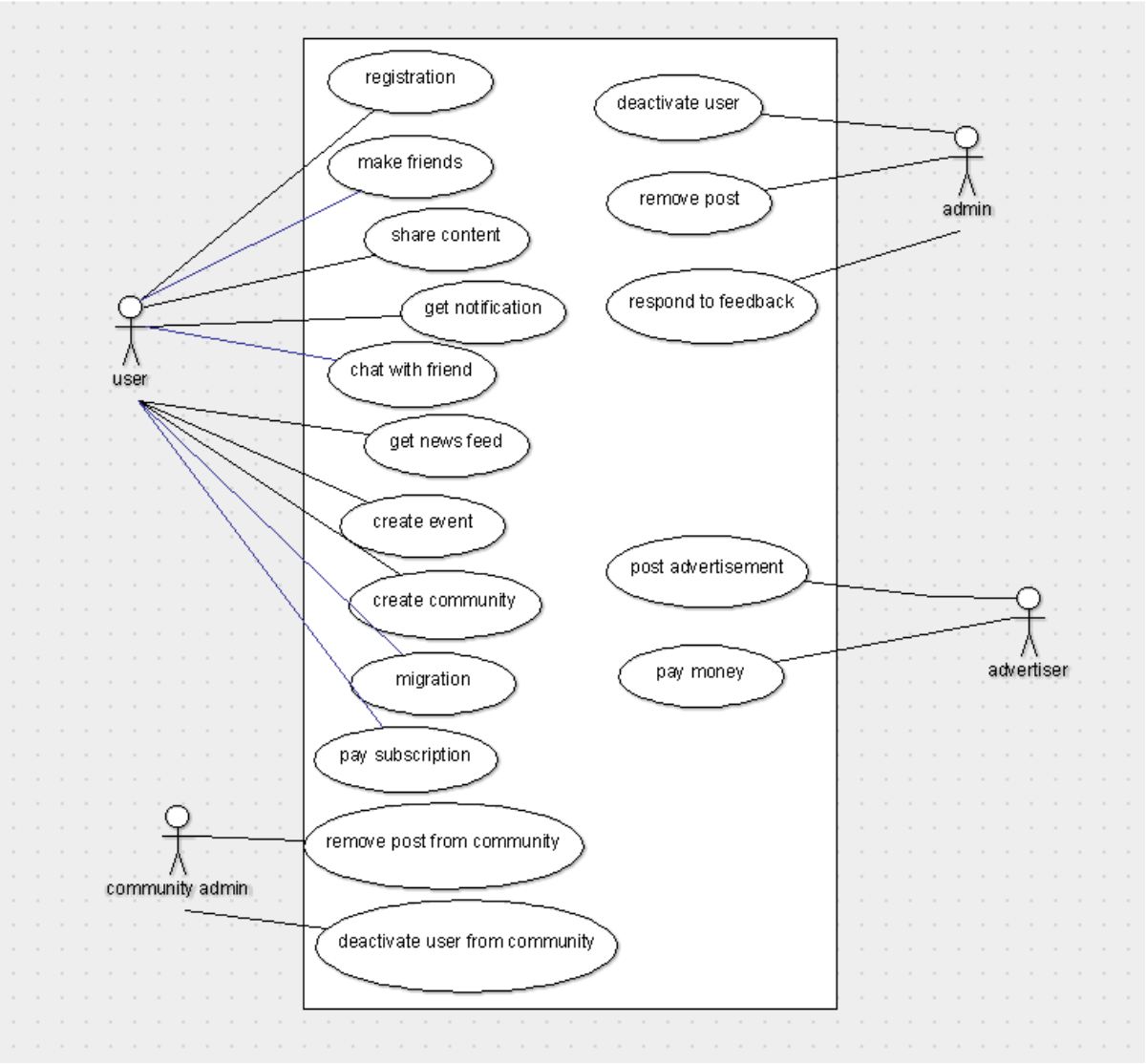

activity_id
description
username
dob
location
mobile
user_id
country
activity
name
common_user
location
type
email
type
generate_activity
descri
ption
admi
advertiser
user
notification
type
gets
balance
notifica
tion_id
Create_commmunity
user_post
create_advertiseme
create_story
community
post_id
contnet
community
name
type
post
community_id
admin_rule
admin_id
user_who_posted
admin_name
text_description
story
story_id
view
content
descriptin
content
user_who_posted
advertisement
ta_age
ta_location
text_description
advertisement_id
ta_gender

ER Diagram

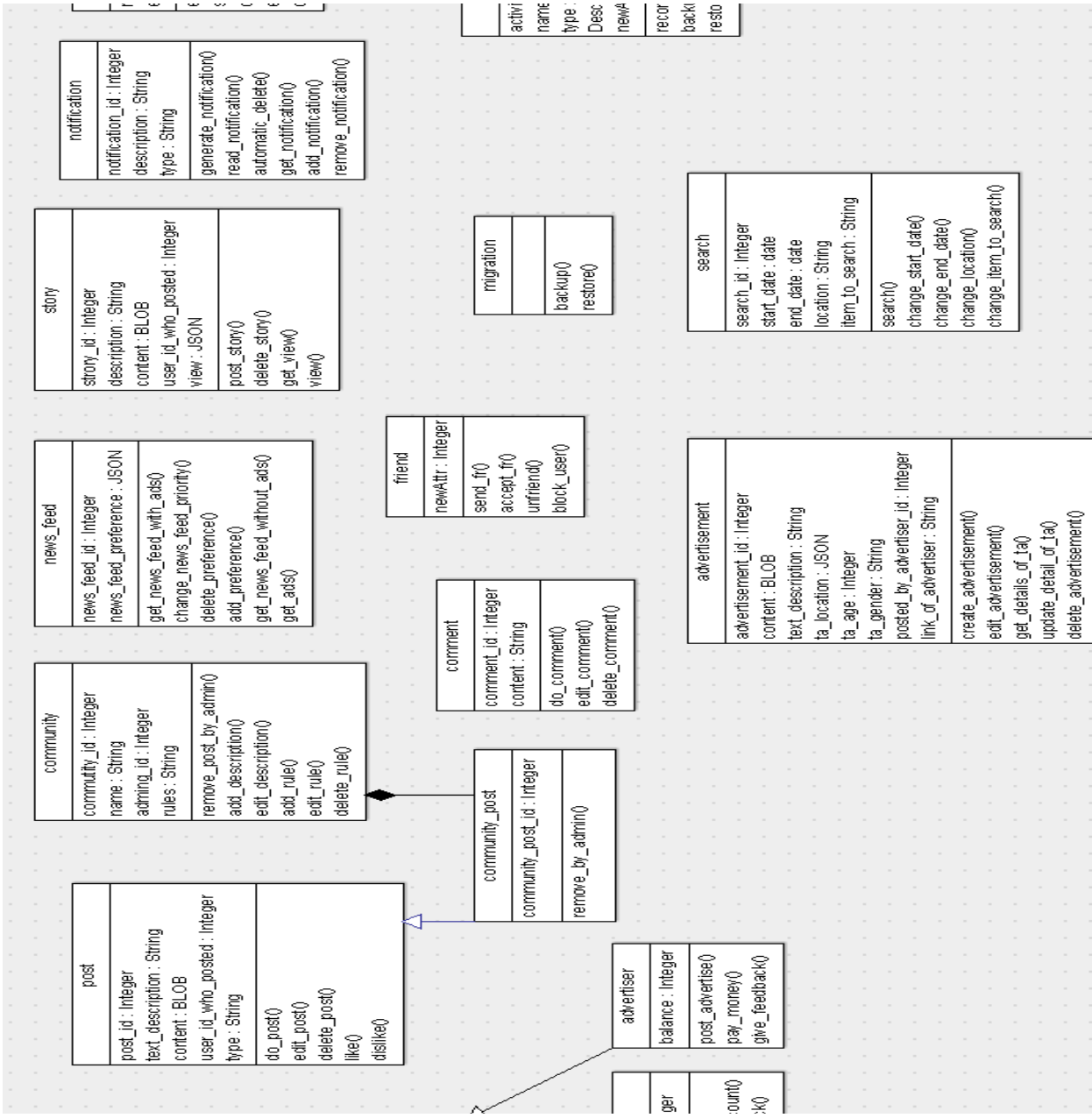

dfd Component Model
user
registration
user_table
message
message
Post
post
get news feed
story
notification
notification
news feed
advertisement
story
advertisement
advertisement
delete account

UML DIAGRAMS

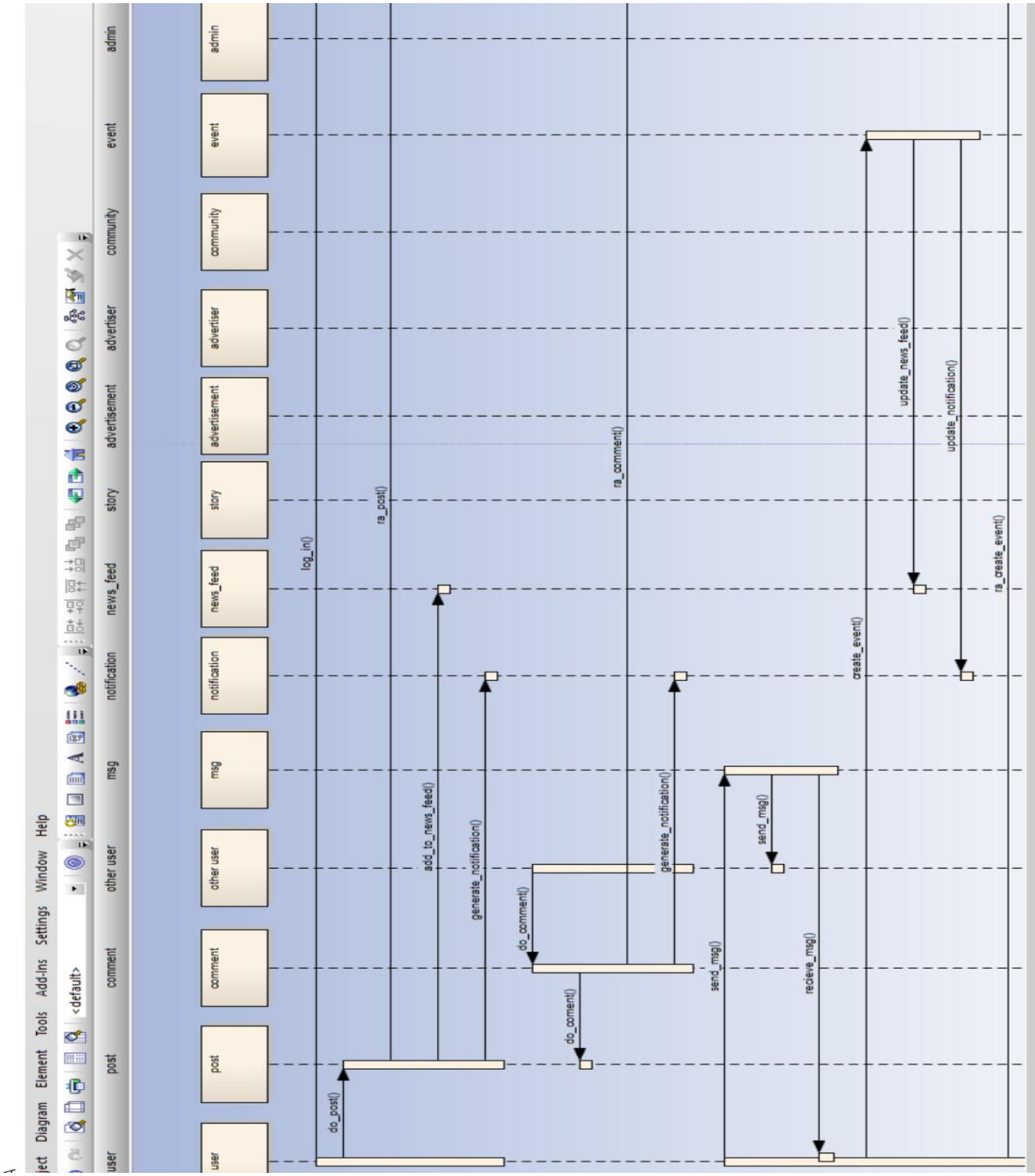
Use Case Diagram

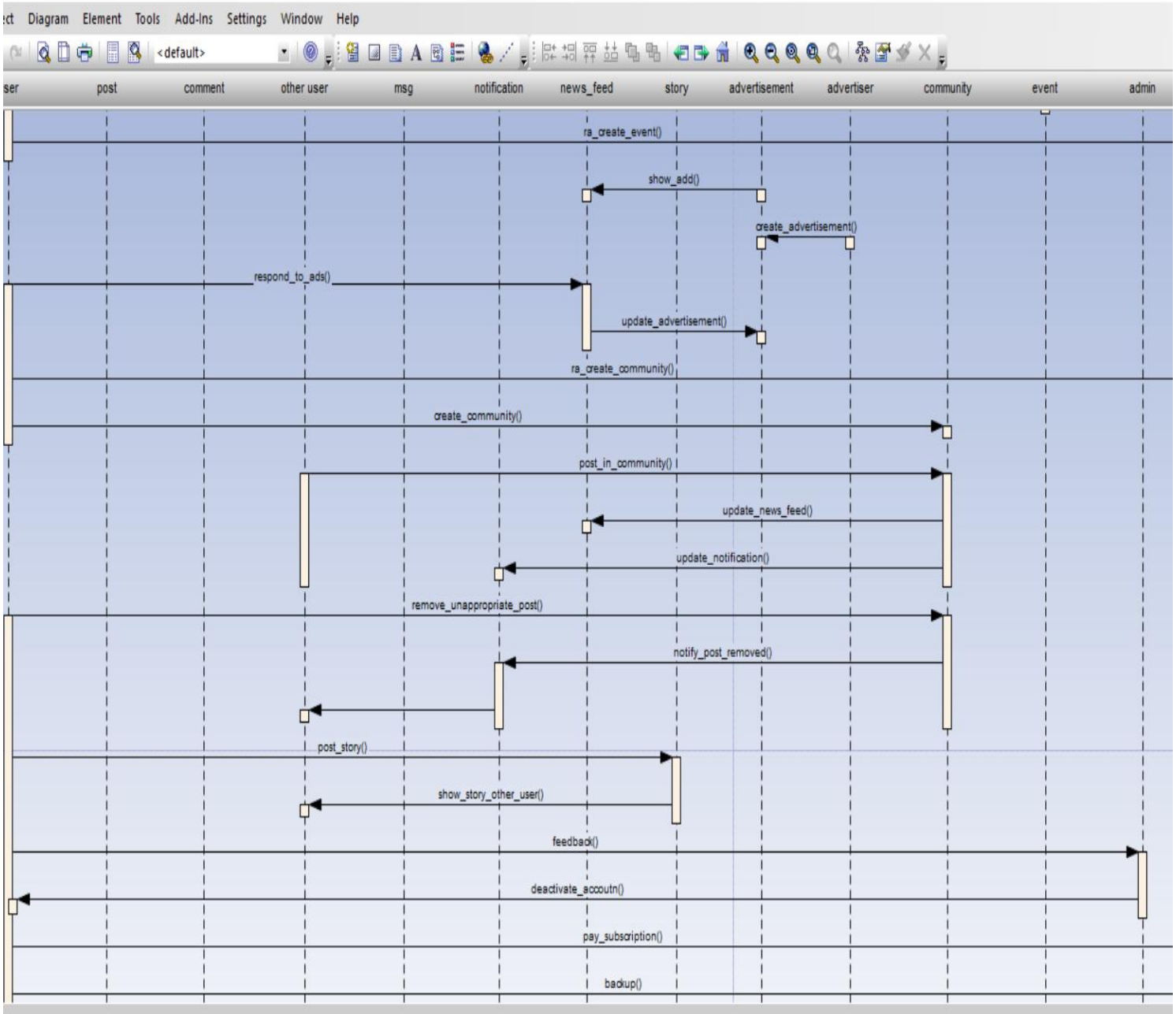


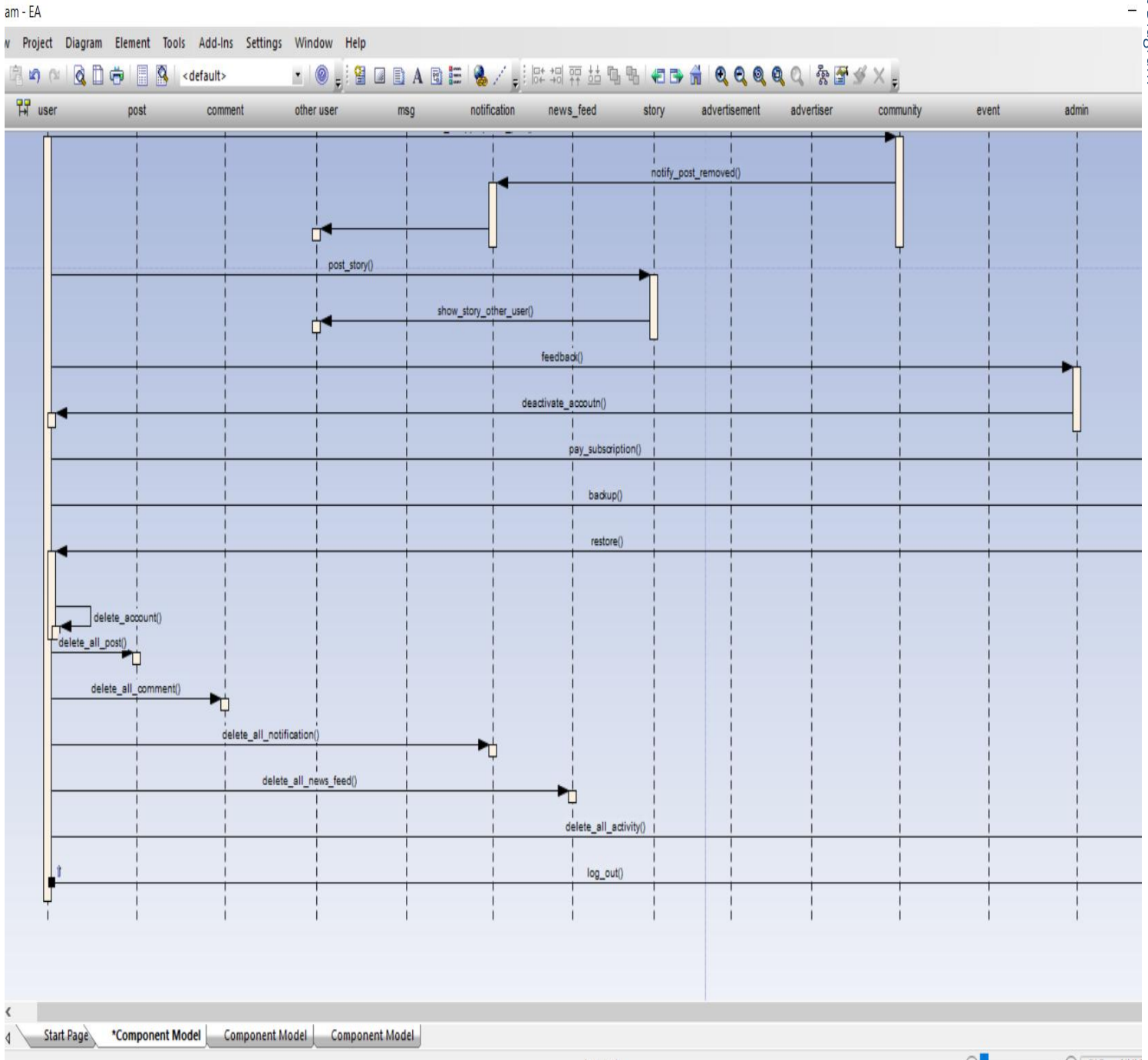
Class Diagram



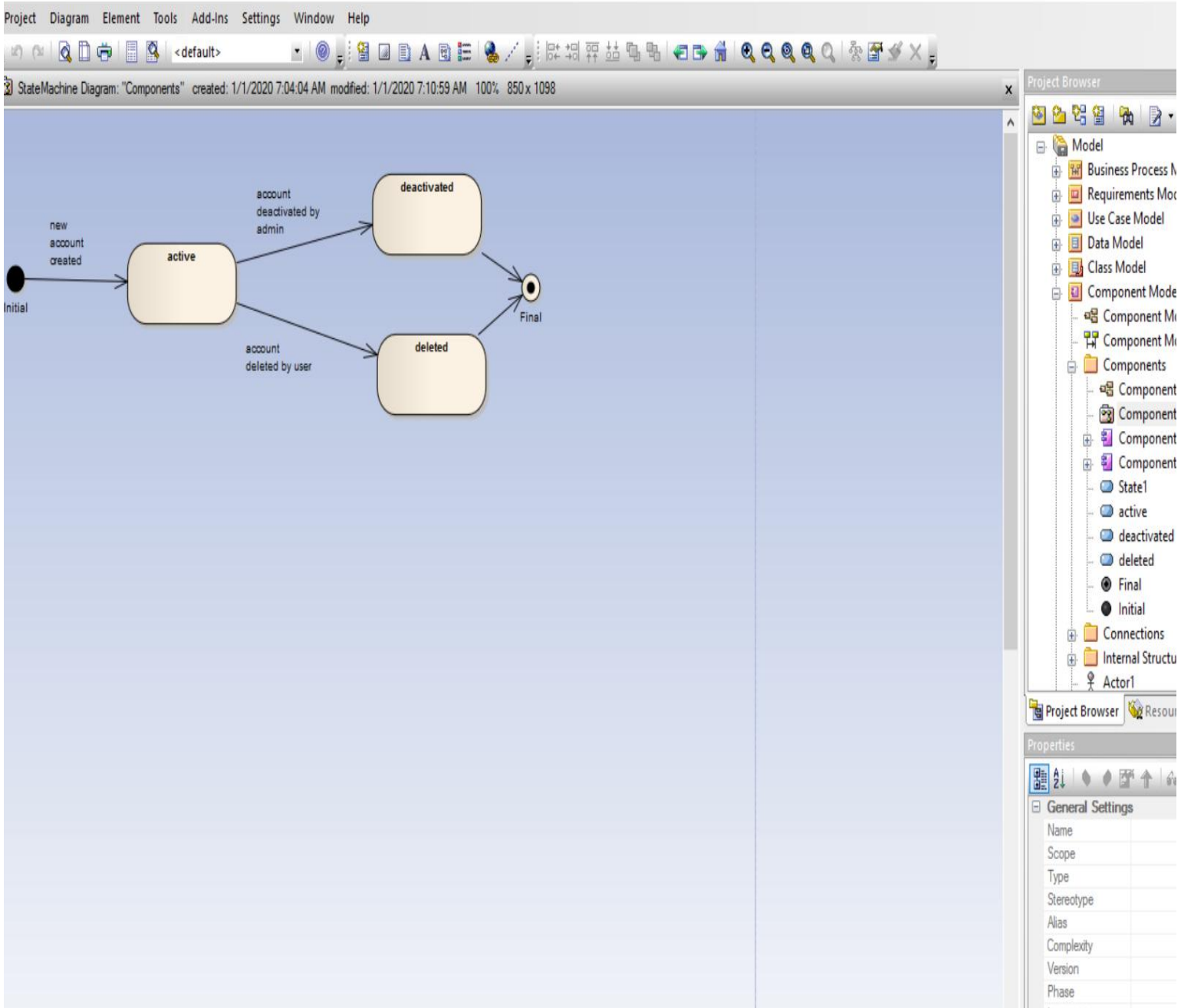
Sequence Diagram




ct Diagram Element Tools Add-Ins Settings Window Help
<default>
ser
post
comment
other user
msg
notification
news_feed
story
advertisement
advertiser
community
event
admin
ra_create_event()
show_add()
create_advertisement()
respond_to_ads()
update_advertisement()
ra_create_community()
create_community()
post_in_community()
update_news_feed()
update_notification()
remove_unappropriate_post()
notify_post_removed()
post_story()
show_story_other_user()
feedback()
deactivate_accoutn()
pay_subscription()
backup()


am - EA
Project Diagram Element Tools Add-Ins Settings Window Help
<default>
user
post
comment
other user
msg
notification
news_feed
story
advertisement
advertiser
community
event
admin
notify_post_removed()
post_story()
show_story_other_user()
feedback()
deactivate_accoutn()
pay_subscription()
backup()
restore()
delete_account()
delete_all_post()
delete_all_comment()
delete_all_notification()
delete_all_news_feed()
delete_all_activity()
log_out()
Start Page
*Component Model
Component Model
Component Model

- EA


Project Diagram Element Tools Add-Ins Settings Window Help
<default>
StateMachine Diagram: "Components" created: 1/1/2020 7:04:04 AM modified: 1/1/2020 7:10:59 AM 100% 850 x 1098
new account created
active
account deactivated by admin
deactivated
account deleted by user
deleted
Initial
Final
Project Browser
Model
Business Process M
Requirements Mod
Use Case Model
Data Model
Class Model
Component Mode
Component Mo
Component Mo
Components
Component
Component
Component
Component
State1
active
deactivated
deleted
Final
Initial
Connections
Internal Structu
Actor1
Project Browser
Resou
Properties
General Settings
Name
Scope
Type
Stereotype
Alias
Complexity
Version
Phase

Data Dictionary

Abbreviation
PK – Primary Key
FK – Foreign Key
ta = targeted_audiance
fr = friend_request
ra = register_activity

common_user

Field Name	Data Type	Required	Field Length	Constraint	Description
c_user_id	Integer	Yes	10	PK	Uniquely identify
username	Stirng	Yes	32		
dob	Date	Yes	10		Birth date of user
location	String		10		Location of user
country	String		10		Country of user
mobile	Integer	Yes	10	UNIQUE	Mobile number of user
email	String	Yes	50	UNIQUE	Email of user

community

Field Name	Data Type	Required	Field Length	Constraint	Description
community_id	Integer	Yes	10	PK	
name	String	Yes	32		Name of community
admin_id	Integer	Yes	10	FK	Foreign key of user
description	String		100		Describe about community
rules	String		1000		Rule which need to be followed by people in order to be healthy community

post

Field Name	Data Type	Required	Field Length	Constraint	Description
post_id	Integer	Yes	10	PK	Uniquely identify
text_description	String		5000		User add description to what he had posted
content	BLOB		5000		User can post image, video, feeling or activity
user_who_posted	Integer	Yes	10	FK	It is foreign_key from user table
type	String	Yes	10		It can be life event, routine post, informational post etc.

advertisement

Field Name	Data Type	Required	Field Length	Constraint	Description
advertisement_id	Integer	Yes	10	PK	
content	BLOB		5000		Image or video
text_description	String		500		Discription about ads
ta_location	String	Yes	32		Location of audience to whom advertiser want to target
ta_age	Integer		3		Age of Target audience
ta_gender	String		5		Gender of target audience

story

Field Name	Data Type	Required	Field Length	Constraint	Description
story_id	Integer	Yes	10	PK	
description	String		500		Description of story of user
content	BLOB		5000		Image , text or video who user posted
user_who_posted	Integer	Yes	10	FK	Id of user who posted story. It is foreign key of user tabel
view	JSON				JSON object containing name of people who viewed story.

search

Field Name	Data Type	Required	Field Length	Constraint	Description
search_id	Integer	Yes	10	PK	
start_date	Date				Search start from this date
end_date	Date				Search till this date
location	String				Location in which we want to search
item_to_serach	String	Yes			Item which user want to search
item_type	JSON				It contain details of item type to be search

activity

Field Name	Data Type	Required	Field Length	Constraint	Description
advertisement_id	Integer	Yes	10	PK	
name	String	Yes	10		Activity name
type	String	Yes			Type of activity
description	String		100		Description of activity

Security

The project use django which provides it's in built sequrity on web. Besides it's other security features included bellow

- Cross Site Scripting Protection
- Cross Site Request Forgery Protection
- SQL Injection Protection

- Clickjacking Protection
- SSL/HTTPS
- Host header validation
- Referrer Policy
- Session Security

It use Object Oriented Modeling(OOM) So user would be able to access only data for which we grated access and prevent accidental changing.
Users were Provided rights only they required
Password of user and admin must be atleast 8 character long, contain alpha-numeric and special character. It is recommended to use strong password which are not easy to guess.(eg. Date 0f birth, Mobile Number etc.)

Message Send by user to each other are which are end to end encrypted with 4096 bit key. Which is considerd secure.

Future

Future version of this can include bellow features

- Live Stream
- Finding appropriate partner or finding date
- Cross Platform post like on Facebook, Twitter or other popular social networking website
- Verification of user with government id to get rid of problem of fake users

Bibliography

1. https://docs.djangoproject.com/en/3.0/
2. https://www.tutorialspoint.com/uml/index.htm
3. Ignou study material MCS-032x`x`x``