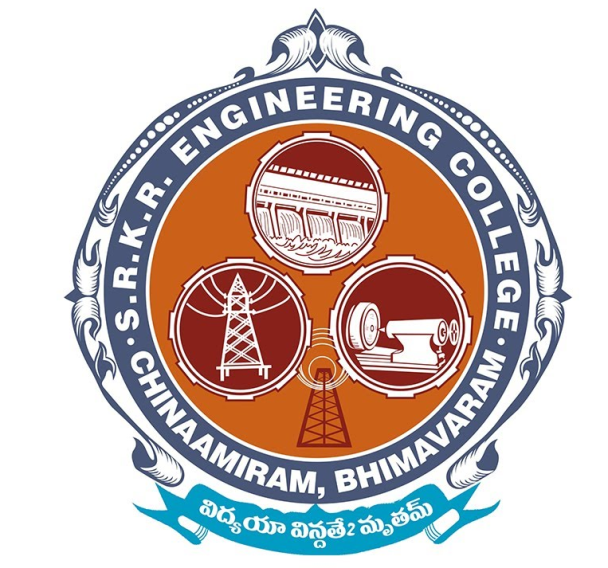
**Project Report**

**On**

**Teach For Friend**

****

**Submitted By**

**Mr.P.N.V.Ranga Raju Regd No: 315175710173**

**Mr.N.Kishore Raju Regd No: 314175710148**

**DEPARTMENT OF COMPUTER SCIENCE ENGINEERING**

**S.R.K.R ENGINEERING COLLEGE**

**(Affiliated to ANDHRA UNIVERSITY, VISAKHAPATNAM)**

**BHIMAVARAM-534204**

**(2017-2018)**

**ACKNOWLEDGEMENT**

Initially I must be thankful to persons who supported me in all aspects of my project. I convey my sincere thanks to **Prof.Chandra Sekhar**, computer science and Engineering, S.R.K.R Engineering College, Bhimavaram for their kind cooperation in the completion of work.

I express my sincere thanks to **Prof. G.V PADMA RAJU,** Head of the Department of computer science and engineering. S.R.K.R Engineering college, Bhimavaram for providing an initiative to this project and giving valuable timely suggestions for the work.

I express my sincere thanks to **Dr. G.P.S.VARMA**, principal, S.R.K.R Engineering College, Bhimavaram for giving me the opportunity for the successful completion of my degree.

I also thanks to other teaching and non-teaching staff for their assistance and help extended. I thank one and all that have contributed directly or indirectly to my project.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **S.No** | **CONTENTS** | **Page No** |
| **1.** | **ABSTRACT** | 1 |
| **2.** | **SOFTWARE REQUIREMENT SPECIFICATION** | 2 |
| 2.1. Introduction | 2 |
| 2.2. Current System | 3 |
| 2.3. Proposed System | 3 |
| **3.** | **UML DESIGN** | 8 |
|  | 3.1. Use Case Diagram | 9 |
|  | 3.2. Class Diagram | 13 |
|  | 3.3. Sequence Diagram | 14 |
|  | 3.4. Collaboration Diagram | 17 |
|  | 3.5. State Chart Diagram | 20 |
|  | 3.6. Activity Diagram | 23 |
| **4.** | **SYSTEM DESIGN DOCUMENT** | 27 |
|  | 4.1. Introduction | 27 |
|  | 4.2. Current Software Architecture | 28 |
|  | 4.3. Proposed Software Architecture | 28 |
|  | 4.4. Subsystem Services | 29 |
|  | 4.5. Database Design | 31 |
| **5.** | **IMPLEMENTATION SCREENS** | 33 |
| **6.** | **SYSTEM TESTING** | 35 |
| **7.** | **CONCLUSION** | 39 |
| **8.** | **REFERENCES** | 40 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **Figure no** | **Name of Figure** | **Page no** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

**LIST OF ABBREVIATIONS**

* **HTTP:** Hypertext Transfer Protocol.
* **JDBC:** Java Database Connectivity.
* **HTML:** Hypertext Markup Language.
* **JSP:**  Java Server Pages.

1. **ABSTRACT**

Teachforfriend is a type of website which acts like an interface among student community to enhance the overall development of all sections of students at any graduation level.

The case with which the teachforfriend deals is all about the problem that might occur among the student community when the examination is commences.

1)**Problem 1**: This problem is facing by many top graded students or the student who knows about a particular topic which most of the other poor student may not know. The problem is most of the poor student want an assistance from whom they are comfortable with, like other friends. So they ask their friends who knows about the corresponding topic well . the top graded student just because of his friend asked him to teach, if he is willing to do he may do it , but what happens when more friends asked him to teach the same thing at different timings. He may not be able to satisfy all of his friends. Instead of teaching the same topic which he already learned again and again if the particular student uses the same time properly in learning additional topics then he may get good marks in their respective examinations.

2)**Problem 2:** This problem is facing by many students who are poor in studies. Since either they didn’t concentrate well on studies right from the beginning or they are able to understand the topics even. When the examinations are commencing that particular student is thinking that if anyone who teaches the particular topic from whom he is comfortable with, But he may not able to find that person right before the exam commences. Many students didn’t think of asking the corresponding professor/teacher. If they get a correct assistance in learning a subject the student can have a chance to enhance their marks in the exam/ even pass the eligibility criteria about a particular exam.

**Solution:**  The solution to the above problem we have framed is to build a system that acts as an interface between these two categories of students. We have to actors in this system one was

the **Tutor**  and the other was the **learner.**

The **tutor** refers to the person who already learned a particular topic and willing to share his knowledge.

The **learner** refers to the person who is in search of the friend or anyone who is able to teach a particular topic that he may understand quickly, so that he may be able to enhance his marks in the examination.

The tutor just post the tutorial about the corresponding topic that he already learned instead of teaching it each and every one again and again/ instead of wasting his time. So that the learner can watch those videos so that it makes the friend available 24X7 virtually to assist the other section of student. This results in overall development of all sections of students.

1. **SOFTWARE REQUIREMENT SPECIFICATION**

**2.1 Introduction:**

**Teachforfriend**  is basically a website which is useful to all category of students in all schools

and colleges by sharing the knowledge of top grade students voluntarily via this website eiher by

making an video and publish it on our webiste or by sharing the materials via our website.

**2.1.1 Purpose of the System:**

The purpose of the system is mainly to reduce the time that a top grade student to teach his friends one aftet the other.It takes so much time for an top grade student to teach the subject he/she knows at the time of examinations. And another problem is though the average and below average students since they have some fear to ask their doubts and other subject related issues. So they approch their friend whoo is cofortable to him in learning a particular topic,but teaching one guy may be easy for a top grade student.If the number increases per person at different timings the person who teaches to his/her friends has to teach the same content again and again.He feels frustrating at some point of time,but he/she may be unable to avoid his/her friends .By utilising the same time without teaching the same topic again and again the top grade students may be able to gain extra knowledge to excel in their parrticular studies.

our website **teachforfriend** is basically allowing its users to publish their tutorial or material in this website once it was completed the other users who want to learn from their friends searches their friend or search for a particular topic to get the same lecture any number of times he/she want and we provide discussion form so that the users can discuss among them get the dought clarified.

By this we can ensure that it is beneficial to all sections of students.

**2.1.2 Scope of the System:**

* Types of users-tutor

-learner

-administrator

* Types of Modules

**Login module:**

In this the user is able to login using google account.

**Profile module:**

In this module the user is able to view their own tutorials and personal info.

**Add tutorial module:**

In this module the user is able to add tutorial by stating the particular subject,chapter, topic, topic name,topic url(youtube embed url of the users youtube channel).

**Search module:**

In this module the user can able to search for other tutorials by specifying the details of either the the other user or specifying the details of the tutorial(ie.. subject or chapter or topic name etc).

**Help module:**

In this module the user is able to see the details about the website and some videos about how to use the website effectively(ie.. how to upload a tutorial into the website?,how to search for an tutorial etc.).

**2.1.3 Objective :**

The main objective of this application is to enhance the progress of all sections of students via sharing their knowledge among themselves.

**2.1.4 Definitions, acronyms and abbreviations:**

**Tutor**: A person who may be a student who want teach to others.

**Learner:** Astudent who want to learn from his friend.

**Admin:** A person who have all privilages in accessing the website and can access and manipulate the info about the other users.

**2.2 Current System:**

At present in this system the user can only be able to add tutorials and view the tutorials made by others or by himself/herself and he/she can search for a particular tutorial using the search box. The user can be able to edit the personal info and info about the tutorials etc

**2.3 Proposed System:**

In future enhancement of the system chat system.

**2.3.1 Functional Requirements:**

Functional requirements are the requirements which deals with the operational requirements of the system and the requirements that are requested by the user.

**Tutor:**

* A tutor can login using google+ api.
* A tutor can view his/her profile.
* A tutor can edit his/her profile.
* A tutor can be able to add tutorials.
* A tutor can be able to search for other friend/tutor.
* A tutor can be able to view other tutorials.
* A tutor can be able to edit the tutorial.

**Learner:**

* A learner can login using google+ api.
* A learner can view his/her profile.
* A learner can edit his/her profile.
* A learner can be able to search for other friend/tutor.
* A learner can be able to view other tutorials.

**Administrator**:

Administrator is the one who can manages the entire site and prevents unauthorised access of database.

**2.3.2 Non Functional Requirements:**

Non functional requirements describe user visible aspects of the system that are not directly related with the functional behaviour of the system. Non functional requirements include quantitative constraints such as response time or accuracy.

* Responsive size of the tutorials based on the device width.
* The Search Result must be obtained within few seconds.

**2.3.2.1 Hardware considerations:**

* Minimum Requirements

|  |  |  |  |
| --- | --- | --- | --- |
| Software | Processor | RAM | Disk Space |
| Internet Explorer – 6 | Intel Pentium III  (or)  AMD – 800MHz | 128 MB | 100 MB |
|  |  |  |  |

* Recommended Requirements:

|  |  |  |  |
| --- | --- | --- | --- |
| Software | Processor | RAM | Disk Space |
| Internet Explorer – 8 | Intel I3 (higher)  (or)  AMD – 1 GHz | 1GB | 40GB |
|  |  |  |  |

**Software considerations**:

* Programming language: php
* UML : Rational rose
* Operating system :Windows , ios , linux.
* DBMS : mysql.
* Web technology : php , bootstrap.
* Web server : Apache Tomcat 6.0.
  + 1. **SYSTEM MODELS:**

Object oriented design is concerned with developing an object oriented model of a software system to implement the identified requirements. It is the process of defining components, interfaces,objects, classes, attributes, and operations that will satisfy therequirements. We typically start with candidate objects defined during analysis, but add much more rigor to their definitions. Then we add or change objects as needy to refine a solution.

Object oriented design can yield the following benefits.

**Maintainability:**Through simplified mapping to the problem domain, which provides for less analysis effort, less complexity in system design, and easier verification by the user.

**Reusability:**

Reusability of the design saves time and cost.

**Productivity:**

Productivity gains through direct mapping to features of object oriented programming.

**2.4. GLOSSARY:**

HTTP:

Hypertext Transfer Protocol is a transaction oriented client orserver protocol between web browser and web server.

HTML:

Hypertext Mark up Language. It is a mark up language used to design static web pages. An HTML file can be created using text editor.

**BOOTSTRAP**: Bootstrap is an open source toolkit for developing with HTML, CSS, and JS. Quickly prototype your ideas or build your entire app with our Sass variables and mixins, responsive grid system, extensive prebuilt components, and powerful plugins built on jQuery.

**PHP:** PHP is a server scripting language, and a powerful tool for making dynamic and interactive Web pages.PHP is a widely-used, free, and efficient alternative to competitors such as Microsoft's ASP.

**MYSQL:** It  enables users to meet the **database** challenges of next generation web, cloud, and communications services with uncompromising scalability, uptime and agility.

1. UML DESIGN

Object-oriented design is concerned with developing an object-oriented model of a software system to implement the identified requirements. It is the process of defining the components, interfaces, objects, classes, attributes, and operations that will satisfy the requirements. I am typically start with the candidate objects defined during analysis, but add much more rigor to their definitions. Then I am adding or change objects as needed to refine a solution.

**3.1 UML Design:**

The Unified Modeling Language is a general purpose visual modeling language that is used to specify, visualize, construct, and document the artifacts of a software system. It captures decisions and understanding about systems that must be constructed. It is used to understand, design, browse, configure, maintain, and control information about such systems. The Unified Modeling Language is very important parts of developing object oriented software and the software development process.  The Unified Modeling Language uses mostly graphical notations to express the design of software projects.  Using the Unified Modeling Language helps project teams communicate, explore potential designs, and validate the architectural design of the software. The primary goals in the design of the Unified Modeling Language are:

* Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models.
* Provide extensibility and specialization mechanisms to extend the core concepts.
* Be independent of particular programming languages and development processes.
* Provide a formal basis for understanding the modeling language.
* Encourage the growth of the object oriented tools market.
* Support higher-level development concepts such as collaborations, frameworks, patterns and components.

Each Unified Modeling Language diagram is designed to let developers and customers view a software system from a different perspective and in varying degrees of abstraction. UML diagrams commonly created in visual modeling tools include.

1. Use Case Diagram
2. Class Diagram.
3. Sequence Diagram.
4. Collaboration Diagram.
5. State Chart Diagram.
6. Activity Diagram.

**Use Case Diagram**

A **use case diagram** shows a set of use cases and actors (a role of a class) and their relationships. Use case diagrams address the static use case view of a system. These diagrams are especially important in organizing and modeling the behaviors of a system.

A use case diagram commonly contain

* Use cases
* Actors
* Dependency, generalization and association relationships.

A Use case diagram may also contain packages, which are used to group elements of the model into larger chunks.

Use case diagram is used in one of two ways.

* To model the context of a system.
* To model the requirements of a system.

**Modeling Procedures:**

1. Identify the actors that surround the system by considering which groups require help from the system to perform their tasks; which groups are needed to execute the systems functions, which groups interact with external hardware or other software systems; and which groups perform secondary functions for administration and maintenance.
2. Organize actors that are similar to one another in a generalization\specialization hierarchy.
3. Where it aids understandability, provide a stereotype for each such actor.
4. Populate a use case diagram with these actors and specify the paths of communication from each other to the systems use cases.
5. In it’s simplest form, a use case can be described as a specific way of using the system from a user’s(actor’s)perspective.

A more detailed description might characterize a use case as:

* A pattern of behavior the system exhibits
* A sequence of related transactions performed by an actor and the
* Delivering something of value to the actor.

Use case provides a means to:

* Capture system requirement
* Communicate with the end users and domain experts
* Test the system.

**3.1.1 Construction Of Use-Case Diagram**

Use case diagrams graphically depict system behavior (use cases).These diagrams present a high level view of the system is used as viewed from an outsider’s(actor’s) perspective. A use case diagram may depict all or sum of the use cases of a system. A use case diagram can contain: Actors (“things” outside the system)Use cases (system boundaries identifying what the system should do)Interactions or relationships between actors and use cases in the system include the associations,dependencies,and generalizations.

**Uses Associations:**

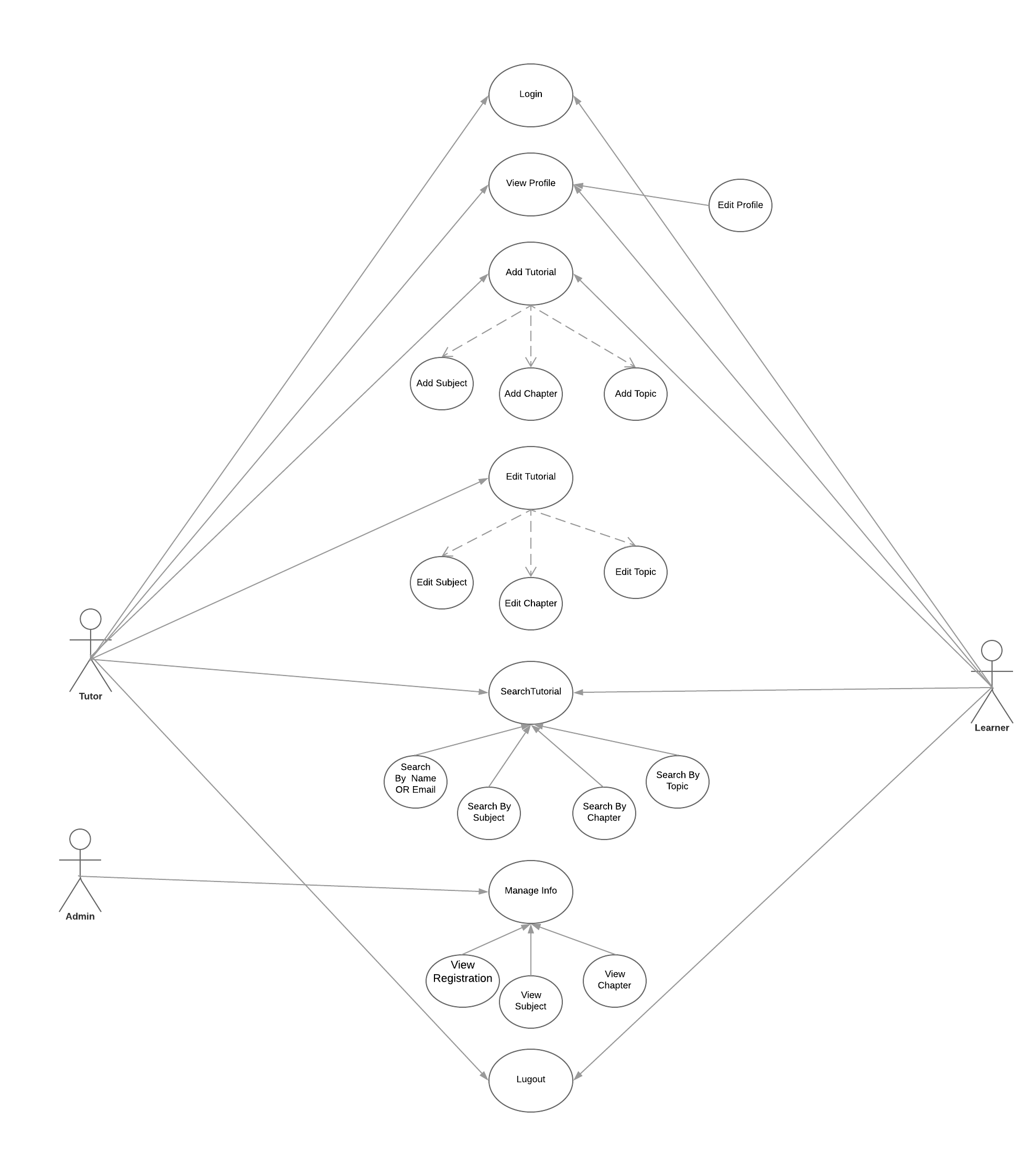
The uses association occurs when we are describing our use-cases and notice that some of them have sub flows in common. To avoid describing a sub flow mare then once in several use-cases, you can extract the common sub flow and make it a use-cases of its own. This new use-case then can be used by other use-cases.The relationships among the other use-cases and this new extracted use-case are called uses association**.**

**Extends Association:**

An extends association is a stereotyped association that specifies how the functionality of one use case can be inserted into the functionality of another use case. Extend relationships between use cases are modeled as dependencies by using the Extend stereotype.

**Include Association:**

An includes association is a stereo typed association that connects a base use case to an inclusion use-case.

**Usecasediagram**:  Fig3.1.1. Use case diagram for Teachforfriend

**List of Use cases:**

* Login.
* View Profile.
* Edit Profile.
* Add Tutorial.
* Edit Tutorial.
* Search Friend.
* View Tutorial.
* Manage Info.
* Add Subject.
* Add Chapter.
* Add Topic.
* View Registration.
* View subjects.
* View Chapter.

**2.3.4.4 Use Case Description** :

**Use case name : Login**

Participating actors : Tutor, Learner, Administrator.

Entry conditions : User enter into the website by using login button.

Flow of events : click on login button.

Click on signin with google.

Enter the credentials of google.

Click on sigi button to signin and redirect to profile.

Exit condition : The system stores the information of the user provided by the google api if the user is signin for the first time. Otherwise the system checks the details of the user an redirect to the user profile with related information about the user.

Special requirements :The response should be obtained within 5-10sec.

|  |
| --- |
|  |

**Use case name : View profile**

Participating actors : Tutor, Learner, Administrator.

Entry conditions : User enter into the website by using login button.

Flow of events : click on login button.

Click on signin with google .

Enter the credentials of google.

Click on sigin button to signin and redirect to profile.

Exit condition : The system stores the information of the user provided by the google api if the user is signin for the first time. Otherwise the system checks the details of the user an redirect to the user profile with related information about the user.

Special requirements :The response should be obtained within 5-10sec.

|  |
| --- |
|  |

**Use case name : Edit profile.**

Participating actors : Tutor, Learner, Administrator.

Entry conditions : User enter into the website by using login button.

Flow of events : click on login button.

Click on signin with google .

Enter the credentials of google.

Click on sigin button to signin and redirect to profile.

Exit condition : The system stores the information of the user provided by the google api if the user is signin for the first time. Otherwise the system checks the details of the user an redirect to the user profile with related information about the user.

Special requirements :The response should be obtained within 5-10sec.

|  |
| --- |
|  |

**Use case name : Add Tutorial.**

Participating actors : Tutor, Learner, Administrator.

Entry conditions : User enter into the website by using login button.

Flow of events : click on login button.

Click on signin with google .

Enter the credentials of google.

Click on sigin button to signin and redirect to profile.

Exit condition : The system stores the information of the user provided by the google api if the user is signin for the first time. Otherwise the system checks the details of the user an redirect to the user profile with related information about the user.

Special requirements :The response should be obtained within 5-10sec.

|  |
| --- |
|  |

**Use case name : Edit tutorial.**

Participating actors : Tutor, Learner, Administrator.

Entry conditions : User enter into the website by using login button.

Flow of events : click on login button.

Click on signin with google .

Enter the credentials of google.

Click on sigin button to signin and redirect to profile.

Exit condition : The system stores the information of the user provided by the google api if the user is signin for the first time. Otherwise the system checks the details of the user an redirect to the user profile with related information about the user.

Special requirements :The response should be obtained within 5-10sec.

|  |
| --- |
|  |

**Use case name : Sign in.**

Participating actors : Tutor, Learner, Administrator.

Entry conditions : User enter into the website by using login button.

Flow of events : click on login button.

Click on signin with google .

Enter the credentials of google.

Click on sigin button to signin and redirect to profile.

Exit condition : The system stores the information of the user provided by the google api if the user is signin for the first time. Otherwise the system checks the details of the user an redirect to the user profile with related information about the user.

Special requirements :The response should be obtained within 5-10sec.

|  |
| --- |
|  |

**Use case name : View tutorial.**

Participating actors : Tutor, Learner, Administrator.

Entry conditions : User enter into the website by using login button.

Flow of events : click on login button.

Click on signin with google .

Enter the credentials of google.

Click on sigin button to signin and redirect to profile.

Exit condition : The system stores the information of the user provided by the google api if the user is signin for the first time. Otherwise the system checks the details of the user an redirect to the user profile with related information about the user.

Special requirements :The response should be obtained within 5-10sec.

|  |
| --- |
|  |

**Use case name : Manage info.**

Participating actors : Tutor, Learner, Administrator.

Entry conditions : User enter into the website by using login button.

Flow of events : click on login button.

Click on signin with google .

Enter the credentials of google.

Click on sigin button to signin and redirect to profile.

Exit condition : The system stores the information of the user provided by the google api if the user is signin for the first time. Otherwise the system checks the details of the user an redirect to the user profile with related information about the user.

Special requirements :The response should be obtained within 5-10sec.

|  |
| --- |
|  |

**Use case name : Add subject.**

Participating actors : Tutor, Learner, Administrator.

Entry conditions : User enter into the website by using login button.

Flow of events : click on login button.

Click on signin with google .

Enter the credentials of google.

Click on sigin button to signin and redirect to profile.

Exit condition : The system stores the information of the user provided by the google api if the user is signin for the first time. Otherwise the system checks the details of the user an redirect to the user profile with related information about the user.

Special requirements :The response should be obtained within 5-10sec.

|  |
| --- |
|  |

**Use case name : Add chapter.**

Participating actors : Tutor, Learner, Administrator.

Entry conditions : User enter into the website by using login button.

Flow of events : click on login button.

Click on signin with google .

Enter the credentials of google.

Click on sigin button to signin and redirect to profile.

Exit condition : The system stores the information of the user provided by the google api if the user is signin for the first time. Otherwise the system checks the details of the user an redirect to the user profile with related information about the user.

Special requirements :The response should be obtained within 5-10sec.

|  |
| --- |
|  |

**Use case name : Add topic.**

Participating actors : Tutor, Learner, Administrator.

Entry conditions : User enter into the website by using login button.

Flow of events : click on login button.

Click on signin with google .

Enter the credentials of google.

Click on sigin button to signin and redirect to profile.

Exit condition : The system stores the information of the user provided by the google api if the user is signin for the first time. Otherwise the system checks the details of the user an redirect to the user profile with related information about the user.

Special requirements :The response should be obtained within 5-10sec.

|  |
| --- |
|  |

**Use case name : View Registrations.**

Participating actors : Tutor, Learner, Administrator.

Entry conditions : User enter into the website by using login button.

Flow of events : click on login button.

Click on signin with google .

Enter the credentials of google.

Click on sigin button to signin and redirect to profile.

Exit condition : The system stores the information of the user provided by the google api if the user is signin for the first time. Otherwise the system checks the details of the user an redirect to the user profile with related information about the user.

Special requirements :The response should be obtained within 5-10sec.

|  |
| --- |
|  |

**Use case name : View subject.**

Participating actors : Tutor, Learner, Administrator.

Entry conditions : User enter into the website by using login button.

Flow of events : click on login button.

Click on signin with google .

Enter the credentials of google.

Click on sigin button to signin and redirect to profile.

Exit condition : The system stores the information of the user provided by the google api if the user is signin for the first time. Otherwise the system checks the details of the user an redirect to the user profile with related information about the user.

Special requirements :The response should be obtained within 5-10sec.

|  |
| --- |
|  |

**Use case name : View chapter.**

Participating actors : Tutor, Learner, Administrator.

Entry conditions : User enter into the website by using login button.

Flow of events : click on login button.

Click on signin with google .

Enter the credentials of google.

Click on sigin button to signin and redirect to profile.

Exit condition : The system stores the information of the user provided by the google api if the user is signin for the first time. Otherwise the system checks the details of the user an redirect to the user profile with related information about the user.

Special requirements :The response should be obtained within 5-10sec.

|  |
| --- |
|  |

**Use case name : View topics.**

Participating actors : Tutor, Learner, Administrator.

Entry conditions : User enter into the website by using login button.

Flow of events : click on login button.

Click on signin with google .

Enter the credentials of google.

Click on sigin button to signin and redirect to profile.

Exit condition : The system stores the information of the user provided by the google api if the user is signin for the first time. Otherwise the system checks the details of the user an redirect to the user profile with related information about the user.

Special requirements :The response should be obtained within 5-10sec.

|  |
| --- |
|  |

**3.1.2 Identification Of Relationships Of Classes**

**Class Diagram:**

A **class diagram** shows a set of classes, interfaces, and collaborations and their relationships. These diagrams are the most common diagrams found in modeling object-oriented systems. Class diagrams address the static design view of a system. Class diagrams that include active classes address the static process view of a system. Component diagrams are variants of class diagrams.

Class diagram commonly contains the following things:

* Classes
* Interfaces
* Collaborations
* Dependency, generalization and association relationships

Class diagrams may also contain packages or sub systems both of which also contain packages or subsystems, both of which are used to group elements of the model into layer chunks.

Class diagrams are used in one of 3 ways:

To model the vocabulary of system.

To model simple collaborations

To model a logical database schema.

**Modeling Procedures**:

When creating a class diagram, just model a part of the things and relationships that make up system design view. For this reason, each class diagram should focus on, one collaboration at time.

Identify the mechanism that would like to model. A mechanism represents some function or behavior of the part of the system which are modeling results from the interaction of a society of classes, interfaces and other things.

For each mechanism, identify the classes, interfaces, and other collaborations that participate in this collaboration. Identify the relationships among these things as well.Be sure to populate these elements with their contents. For classes, start with getting a good balance of responsibilities. Then, overtime, turn these into concrete attribute and operations.

**Class Diagram:**

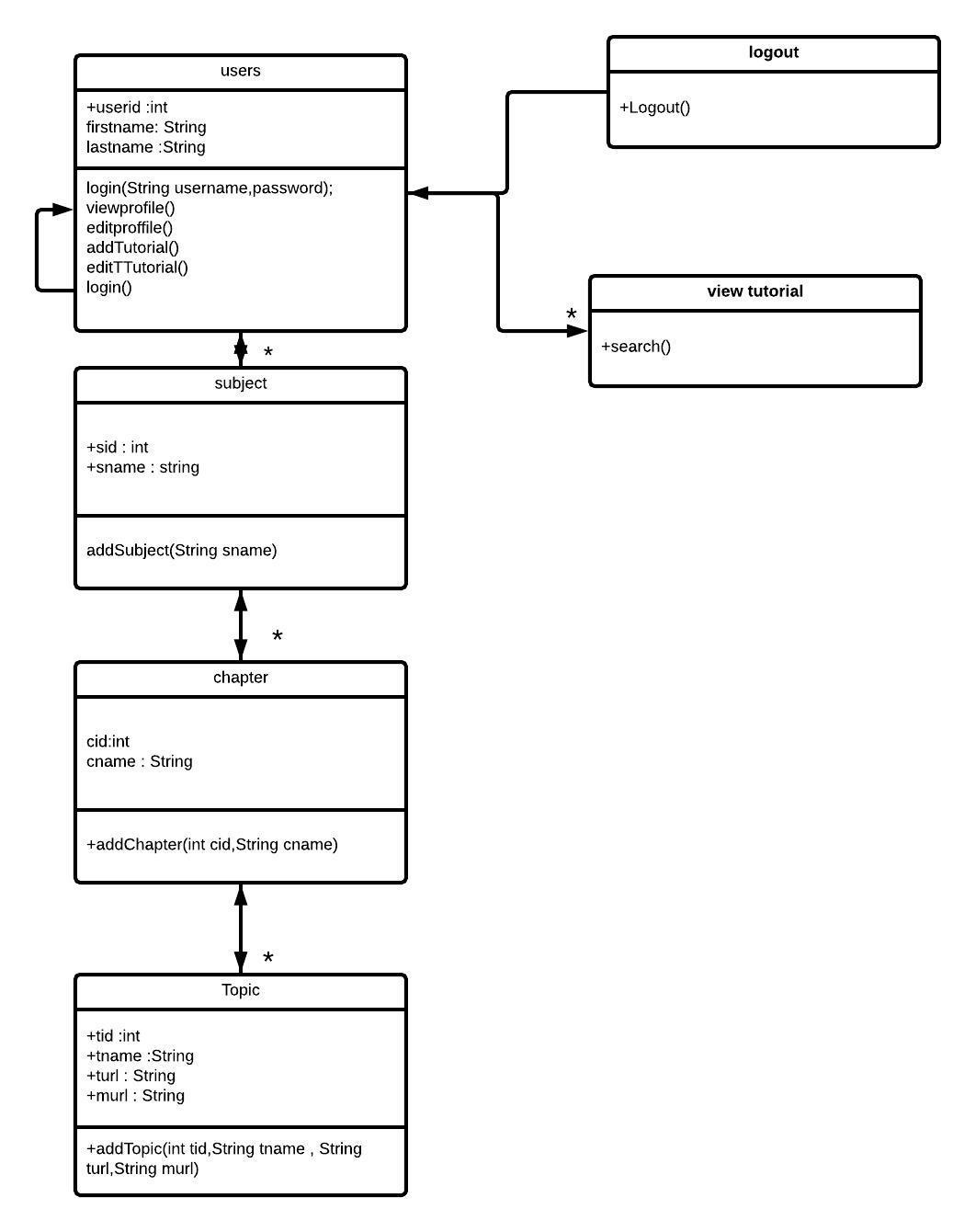
****

Fig3.2.1class diagram for teachforfriend.

**3.1.3 Construction Of Sequence Diagram:**

A sequence diagram is graphical view of scenario that shows object interaction in a time-based sequence what Happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces.

The following tools located on the sequence diagram toolbox which enable to model sequence diagrams.

* **Object**: an object has state, behavior, and identity. The structure and behavior of similar objects are defined in their common classes. Each object in a diagram indicates some instance of the same class. An object that is not named is referred to as a class instance.
* **Message Icons:** A message icon represents the communication between objects indicating that an action will follow. The message icon is a horizontal, solid arrow connecting two lifelines together.
* **Focus of Control**: Focus of control (FOC) is an advanced notational technique that enhances sequence diagrams. It shows the period of time during which an object is performing an action, either directly or through an underlying procedure.
* **Message to Self**: A message to self is a tool that sends a message from one object back to the same object. It does not involve other objects because the message returns to the same object. The sender of a message is the same as the receiver.
* **Note**: A note captures the assumptions and decisions applied during analysis and design. Notes may contain any information, including plain text, fragments of code, or references to other document.
* **Note Anchor**: A note anchor connects a note to the element that it affects.

**Sequence Diagrams:**

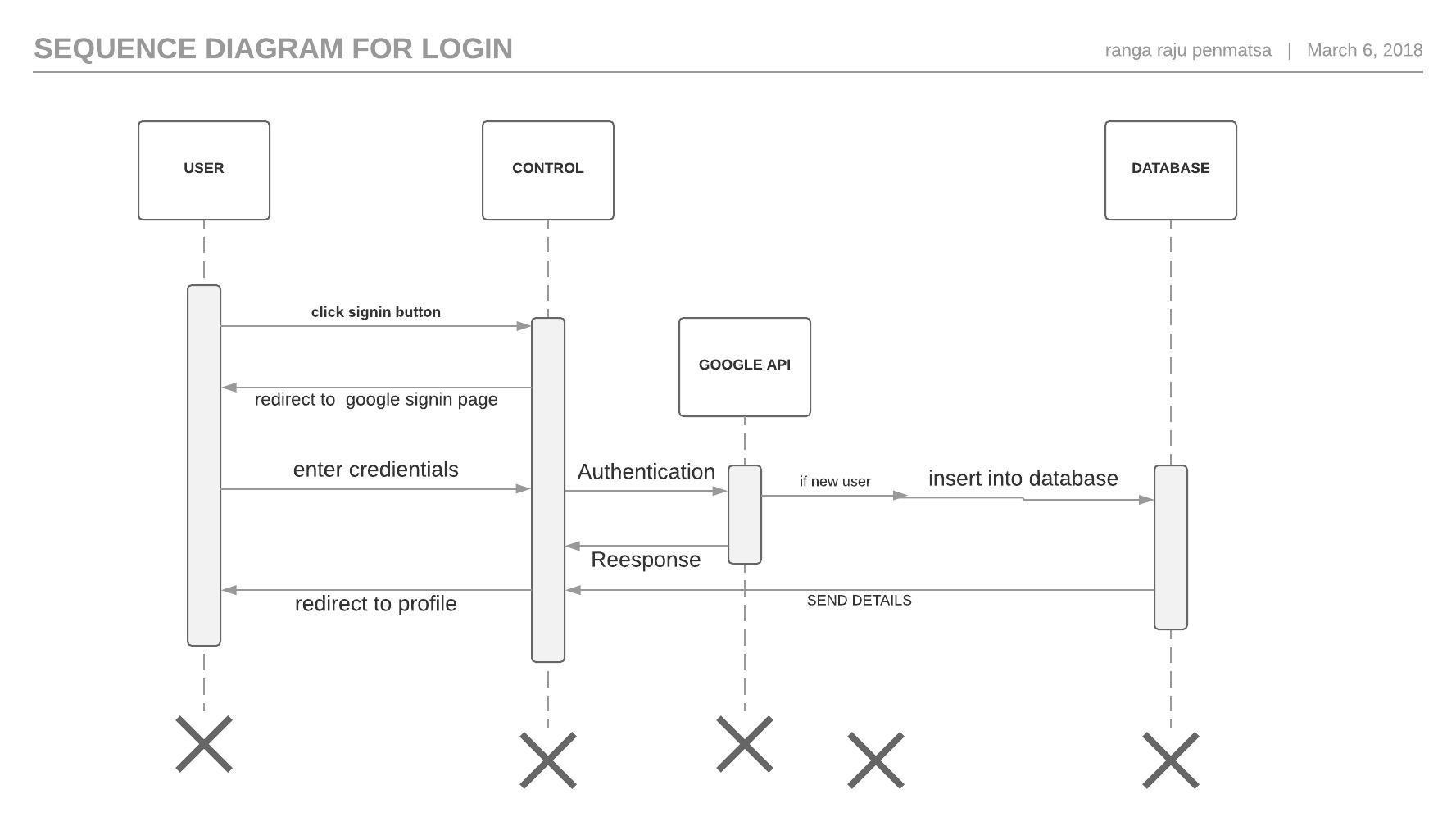


Fig3.3.1. Sequence diagram for Login

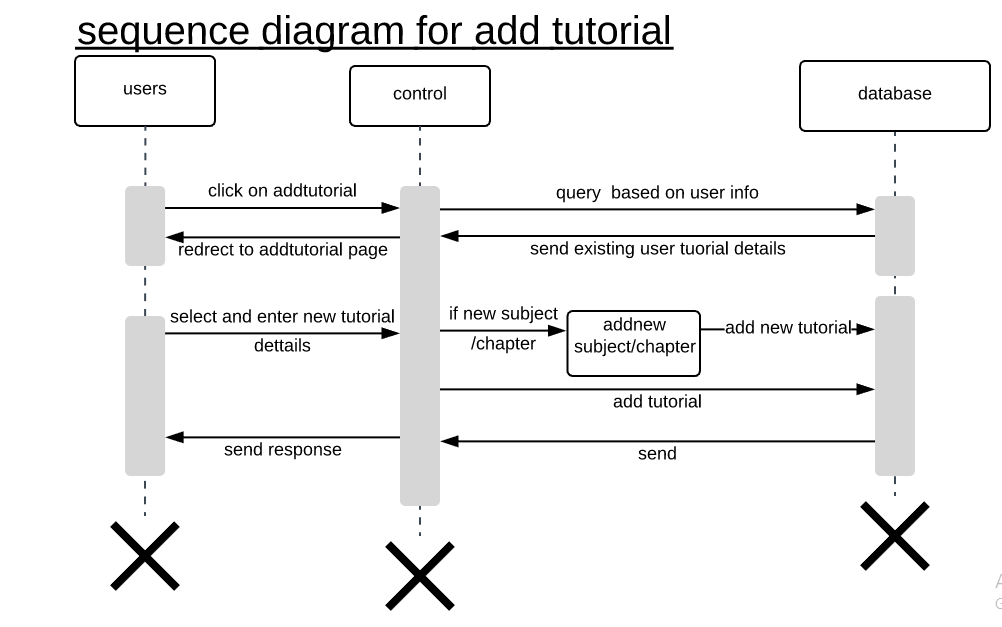


Fig3.3.2.Sequence Diagram for addtutorial

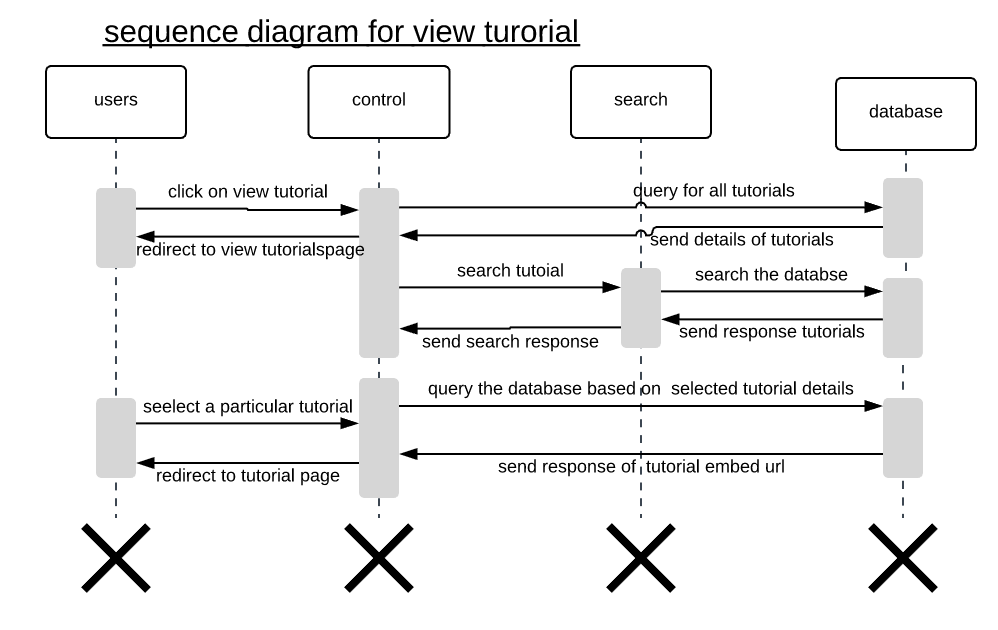
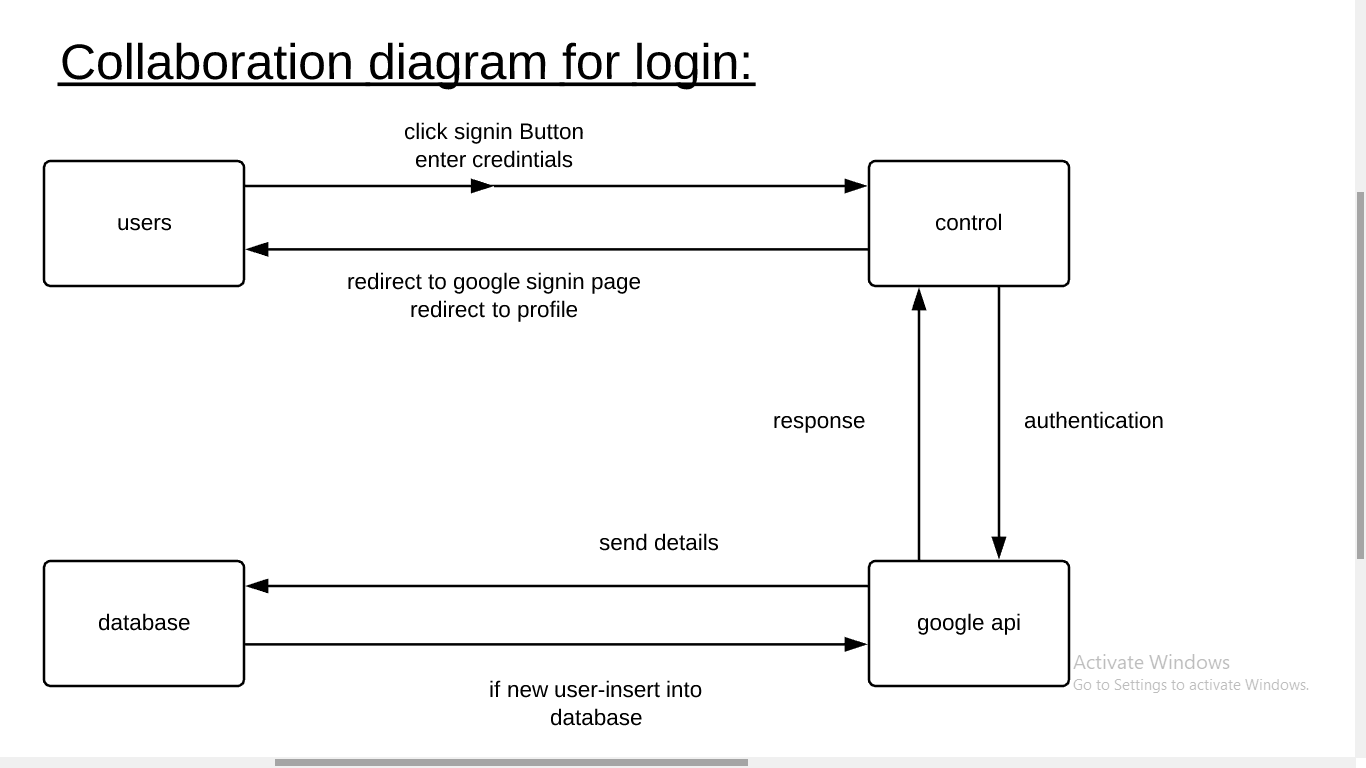
****

Fig3.3.2.Sequence Diagram for viewtutorial.

**COLLABORATION DIAGRAM**

Collaboration diagrams and sequence diagrams are alternate representations of an interaction. A collaboration diagram is an interaction diagram that shows the order of messages that implement an operation or a transaction. A sequence diagram shows object interaction in a time based sequence. Collaboration diagrams show objects, their links and their messages. They can also contain simple class instances and class utility instances. Each collaboration diagram provides a view of the interactions or structural relationships that occur between objects and object-like entities in the current model.

The create collaboration diagram command creates a collaboration diagram from information contained in the sequence diagram. The create sequence diagram command creates sequence diagram from information contained in the interaction’s collaboration diagram. The go to sequence diagram and go to collaboration diagram commands traverse between an interaction’s two representations. Collaboration diagrams contain icons representing objects. You can create one or more collaboration diagrams to depict interactions for each logical package in your model. Such collaboration diagrams are themselves contained by the logical package enclosing the objects they depict. An object specification enables you to display and modify the properties and relationships of an object. The information can also be displayed inside the icons representing objects in collaboration diagrams.

****

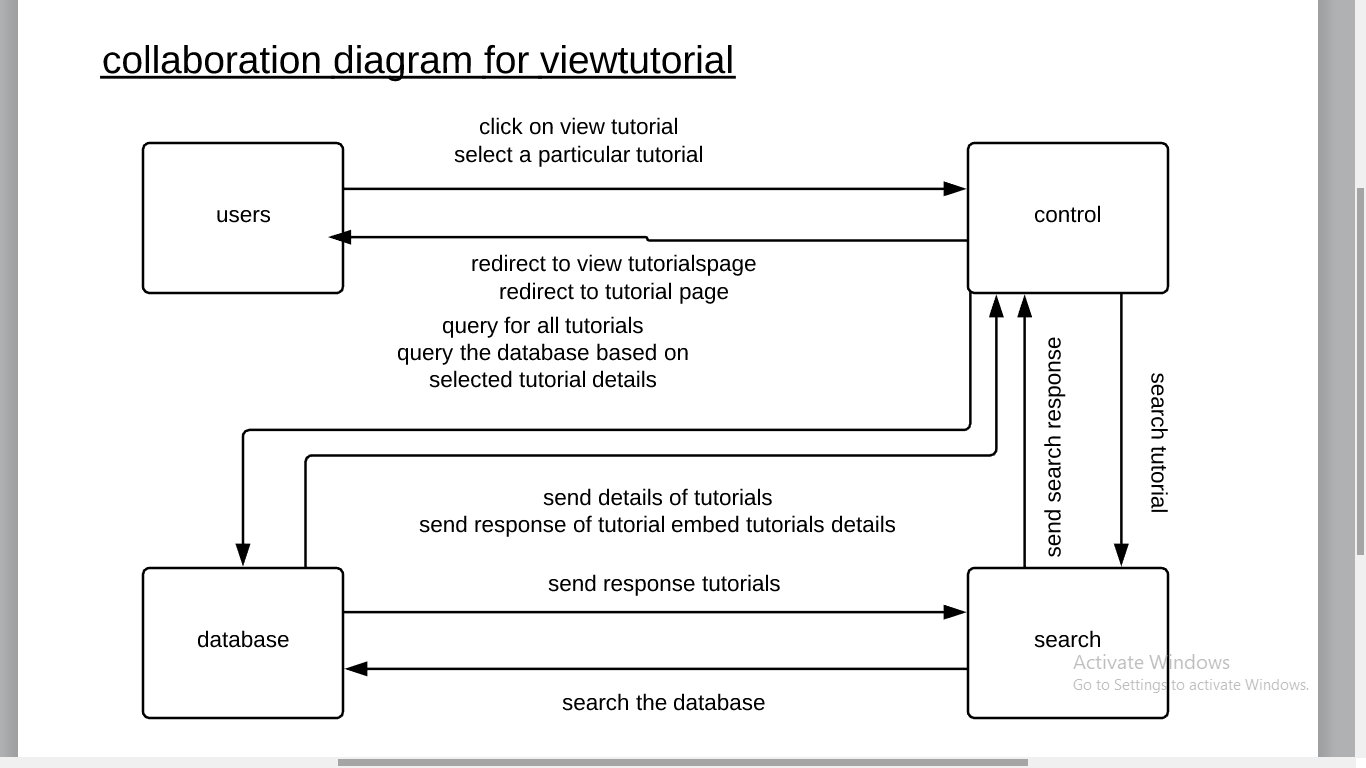
****

Fig 2.5.1: Collaboration diagram for Admin

**3.1.5construction Of Uml State Chart Diagram**

State chart diagrams model the dynamic behavior of individual classes or any other kind of object. They show the Sequences of states that an object goes through, the events that cause a transaction from one state to another and the actions that result from a state change.

State chart diagrams are closely related to activity diagrams. The main difference between the two diagrams are state centric, while activity diagrams are activity centric. A state chart diagram is typically used to model the discrete stages of an objects lifetime, where as an activity diagram is better suited to model the sequence of activities in a process.

Each state represents a method condition during the life of an object during which it satisfies some condition or waits for an event. State chart diagram typically contains one start state and end states. Transactions connect the various states on the diagram.

The following tools are used on the state chart diagram toolbox to model state chart diagrams:

* **Decisions:** A decision represents a specific location on state chart diagram where the workflow may branch based upon guard conditions.
* **Synchronizations:** Synchronizations visually define forks and joins representing parallel workflow.
* **Forks and Joins:** A fork construct is used to model a single flow of control that divides into two of more flows of control that unite into a single flow of control.
* **States:** A state represents a condition or situation during the life of an object during which it satisfies some condition or waits for some event.
* **Transactions:** A state transaction indicates that an object in the source state will perform certain specified actions and enter the destination sate when a specified event occurs or when certain conditions are satisfied.
* **Start States:** A start state (also called an “initial state”) explicitly shows the beginning of a work flow.
* **End States:** An end state represents a final or terminal state.

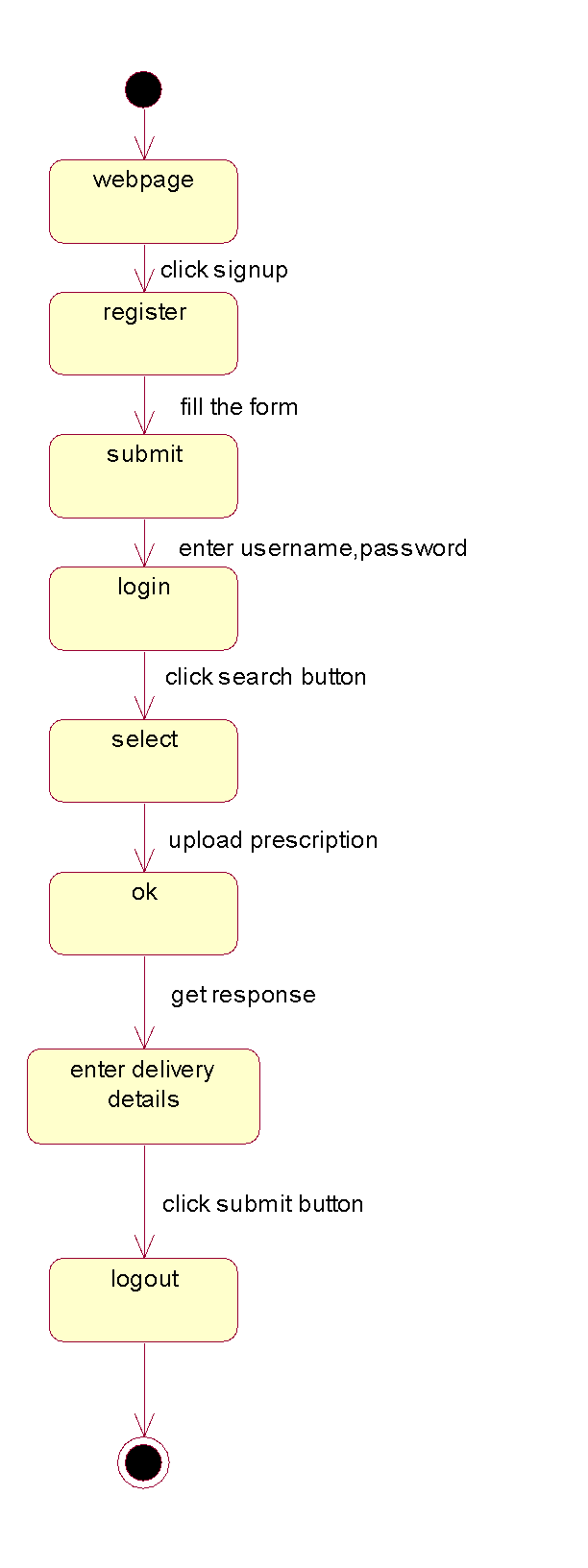


Fig 3.5.1 state chart diagram for online pharmacy

**3.1.4 Uml Activity Diagram**

An **Activity diagram** shows the flow from activity to activity. An activity is a going non-atomic execution within a state machine. An activity results in some action, results in a change of state or return of a value. Action encompasses calling operation, sending a signal, creating or destroying objects, or a pure computation such as evaluating some expression. Activity diagram commonly contains

1. activity states and action states
2. transitions
3. Objects, it may contains nodes and constraints.

**Activity states and action states:**

An executable atomic computation is called action state, which cannot be decomposed. They rendered as lozenge shape. Activity state is non-atomic, decomposable and take some duration to execute.

**Transition:**

It is the path from one state to the next state, represented as simple directed line.

**Branching:**

When alternate paths exist, branching arises which is represented by open diamond. It has one incoming transition, two or more outgoing transitions.

**Forking and joining:**

The Synchronization bar, when splits one flow into two or more flows is called Fork. When two or more flows are combined at synchronization bar, the bar is called Join.

**Swim lanes:**

Grouped work flow is called swing lane. All groups are portioned by vertical solid lines. Each swim lane specifies locus of activities and has a unique name. Each swim lane is implemented by one or more classes. Transition may occur between objects across swim lanes.

**Activity Diagram:**



Fig3.4.1. Activity Diagram For online pharmacy

|  |  |
| --- | --- |
| **4.** | **SYSTEM DESIGN DOCUMENT** |
|  | **3.1 DATA BASE DESIGN** |
|  | **Schema**: |
|  | **USERS TABLE:** |
|  | |  |  |  |  | | --- | --- | --- | --- | | **c** | **FIELD NAME** | **DATA TYPE** | **DESCRIPTION** | | 1 | userid | Varchar(50) | This is the email of the user. | | 2 | givenname | Varchar(50) | This is the firstname of the user. | | 3 | familyname | Varchar(50) | This is the lastname of the user | | 4 | number | Bigint(12) | This is the phone number of user. | | 5 | eduqual | Varchar(20) | This is the eduqulification of user | |

**SUBJECT TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SNO** | **FIELD NAME** | **DATA TYPE** | **DESCRIPTION** |
| 1 | userid | Varchar(50) | This is the email of the user. |
| 2 | subjectid | Int(50) | This is the subject id. |
| 3 | subjectname | Varchar(50) | This is the subject name. |

**CHAPTERS TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SNO** | **FIELD NAME** | **DATA TYPE** | **DESCRIPTION** |
| 1 | userid | Varchar(50) | This is the email of the user. |
| 2 | chapterid | Int(50) | This is the chatper id. |
| 3 | chaptername | Varchar(50) | This is the chapter name. |

**TOPICS TABLE:**

|  |  |  |  |
| --- | --- | --- | --- |
| **SNO** | **FIELD NAME** | **DATA TYPE** | **DESCRIPTION** |
| 1 | userid | Varchar(50) | This is the email of the user. |
| 2 | subjectid | Int(50) | This is the subject id. |
| 3 | chapterid | Int(50) | This is the chapter id. |
| 4 | topicid | Int(50) | This is the topic id. |
| 5 | topicname | Varchar(50) | This is the topic name. |
| 6 | topicurl | Varchar(255) | This is the topic embed url string. |
| 7 | description | Varchar(255) | This is the description |
| 8 | materialurl | Varchar(255) | This is the material url associated with the topic. |
| 9 | date | datetime | This is the current time stamp of date and time |

**NORMALISATION:**

## First normal form (1NF)

As per the rule of first normal form, an attribute (column) of a table cannot hold multiple values. It should hold only atomic values.

For above schema: the above schema is in First normal form, since we all attributes have atomic values.

Second normal form (2NF)

A table is said to be in 2NF if both the following conditions hold:

* Table is in 1NF (First normal form)
* No non-prime attribute is dependent on the proper subset of any candidate key of table.

An attribute that is not part of any candidate key is known as non-prime attribute.

For above schema: The above schema is in 2NF, since there is no partial dependencies.

## Third Normal form (3NF)

A table design is said to be in 3NF if both the following conditions hold:

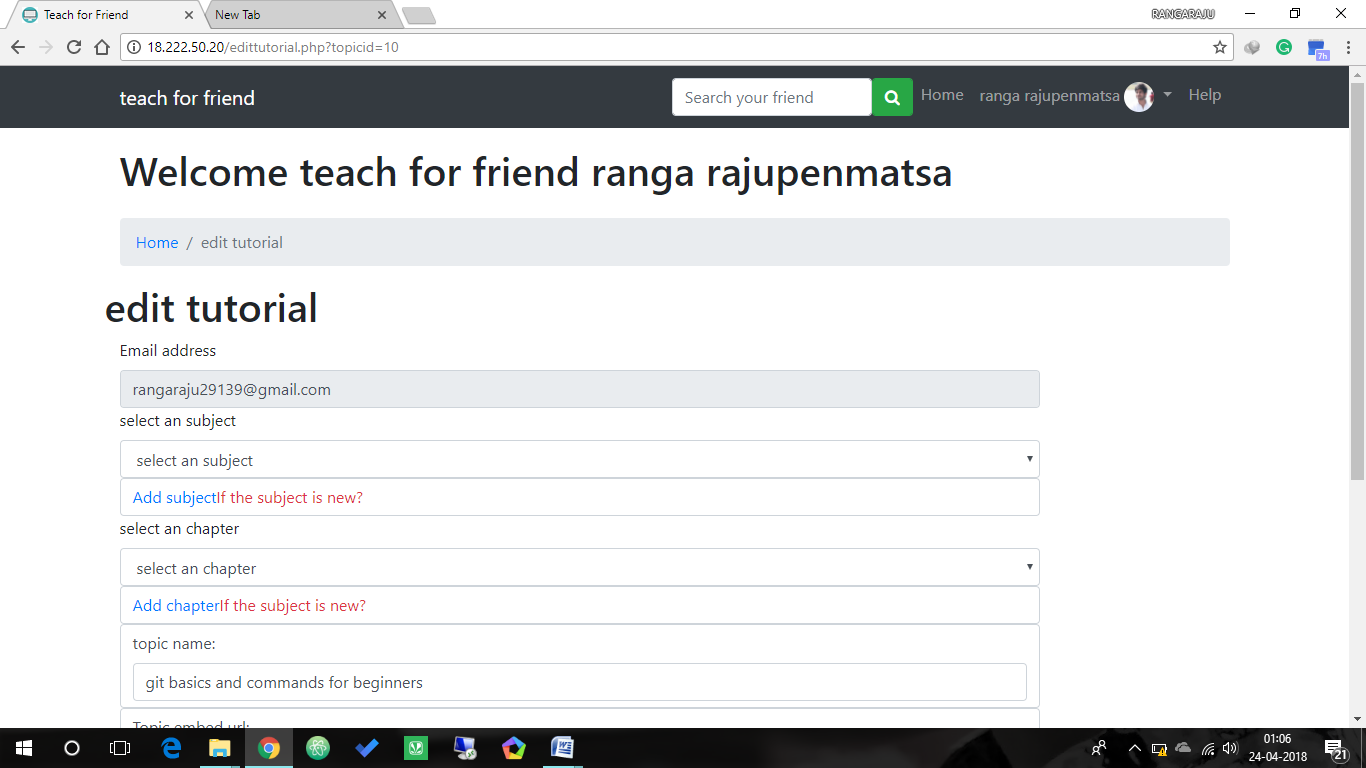
* Table must be in 2NF
* Transitive functional dependency of non-prime attribute on any super key should be removed.

An attribute that is not part of any candidate key is known as non-prime attribute.

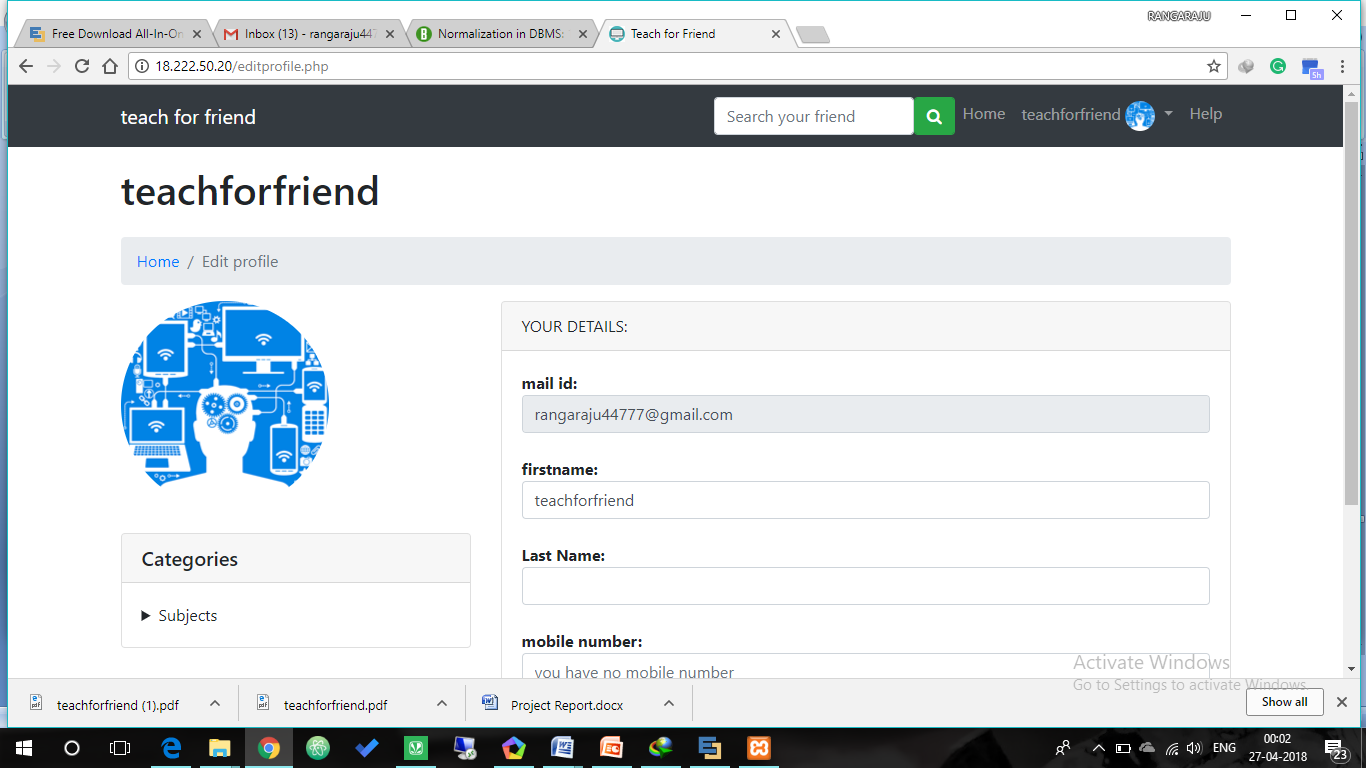
For above Schema: This schema is in 3NF,since there is no transitive dependencies.

|  |  |
| --- | --- |
| **5.** | **IMPLEMENTATION SCREENS**  Screen 5.1: Index page. |
|  | **Screen 5.2: About page.**    **SCREEN 5.3: CONTACT PAGE.**    **SCREEN 5.4: SIGNIN PAGE** |
|  |  |
|  |  |
|  | **SCREEN 5.5: SELECTING GOOGLE ACCOUNT**    **SCREEN 5.6: ENTERING PASSWORD PAGE**    **SCREEN 5.7: PROFILE PAGE.**    **SCREEN 5.8: HOME PAGE AFTER SIGNING IN.**    **SCREEN 5.9: MYTYTORIALS PAGE.**    **SCREEN 5.10: HELP PAGE.**    **SCREEN 5.11: ADD TUTORIAL PAGE.**    **SCREEN 5.12: ADDSUBJECT PAGE.**    **SCREEN 5.13: ADDCHAPTER PAGE.**    **SCREEN 5.14:VIEW TUTORIAL PAGE.** |

**SCREEN 5.15: EDIT TUTORIAL PAGE.**

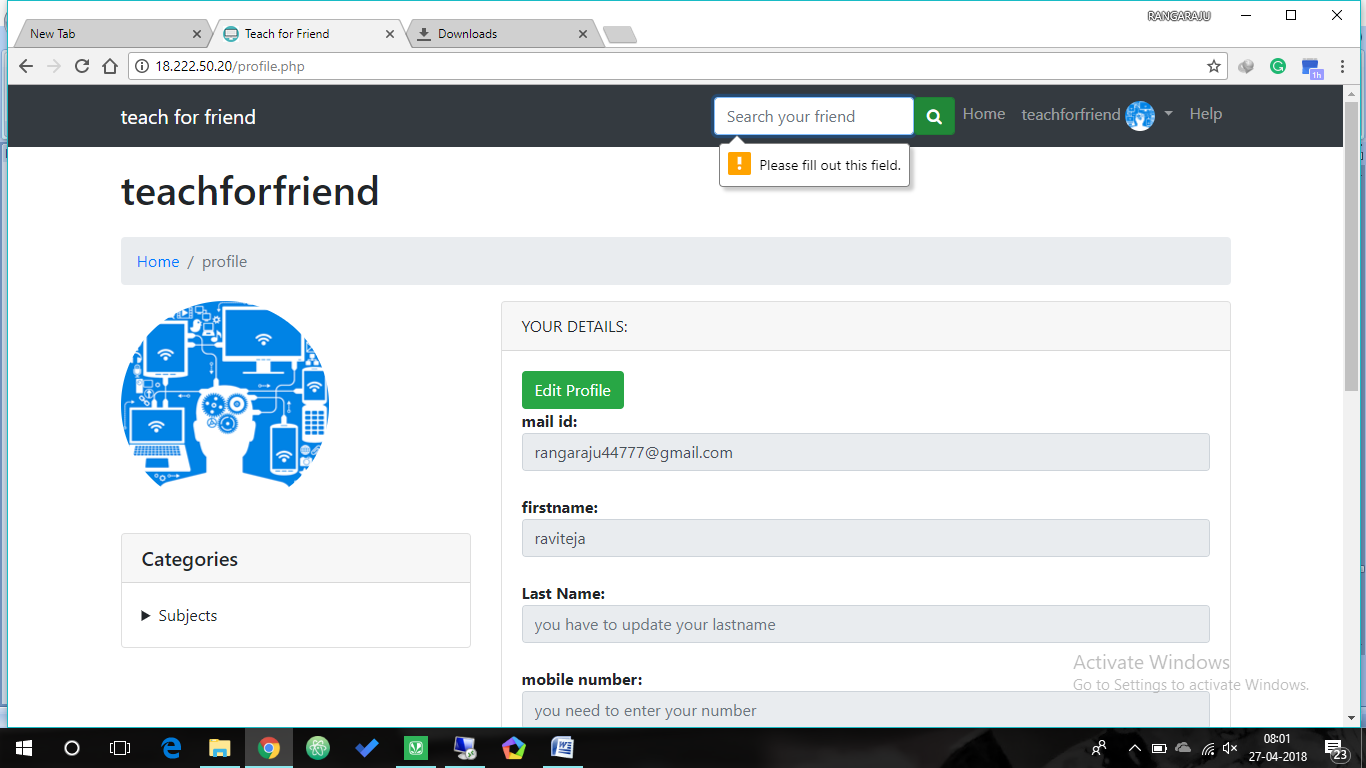


**SCREEN 5.16: EDIT PROFILE PAGE.**

****

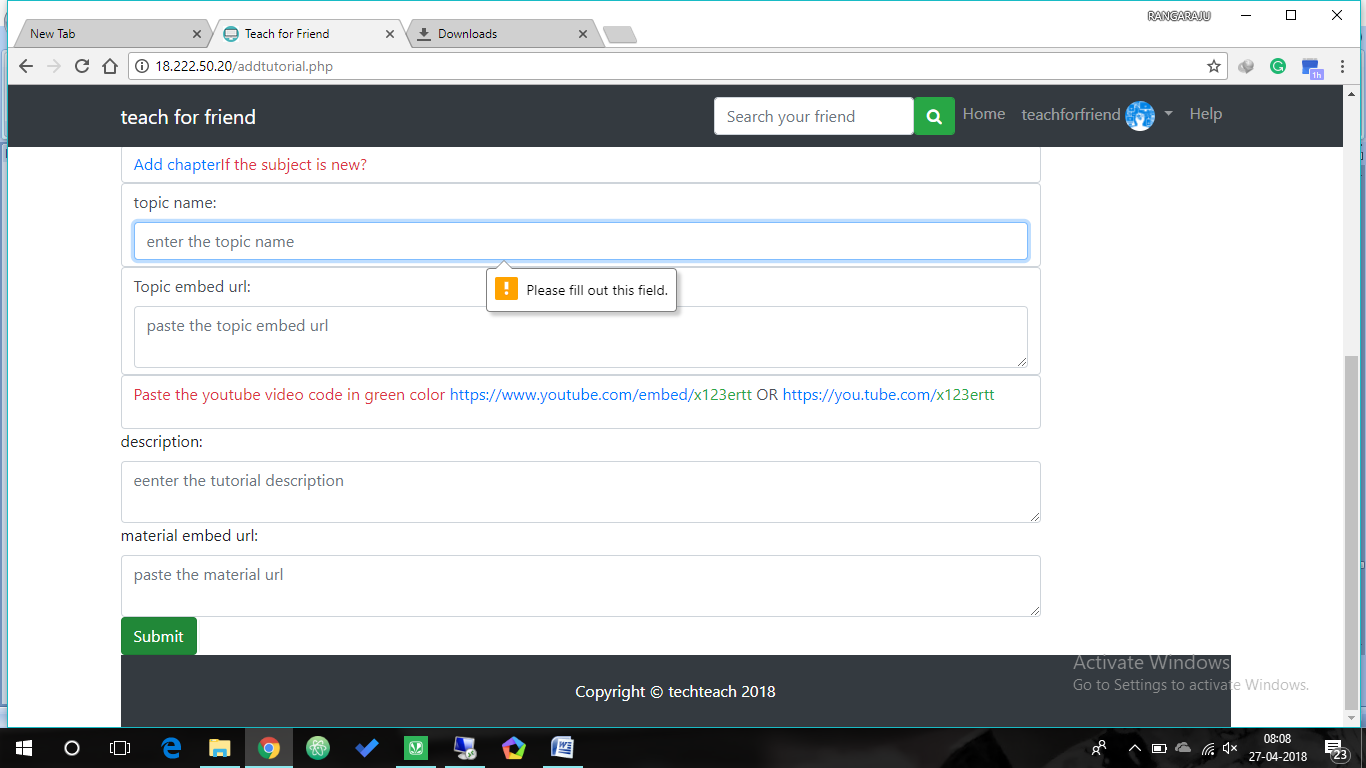
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **6** | **SYSTEM TESTING**  The purpose of testing is to discover errors. Testing is the process of trying to discover  every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.  ***Types Of Testing***  ***5.1 Unit testing***  Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produces valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.  ***5.2 Functional test***  Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.  Functional testing is centered on the following items:  Valid Input : identified classes of valid input must be accepted.  Invalid Input : identified classes of invalid input must be rejected.  Functions : identified functions must be exercised.  Output : identified classes of application outputs must be exercised.  Systems/Procedures : interfacing systems or procedures must be invoked.  ***5.3 System Test***  *System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.*  **5.4 Performance Test**  The Performance test ensures that the output is produced within the time limits, and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results  **5.5 Integration testing**  Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.  The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.  **5.6 Acceptance Testing**  User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.  **5.7 Test Cases**  In general a test case is a set of test data and test programs and their expected results. A test case in software engineering normally consists of a unique identifier, requirement references from a design specification, preconditions, events, a series of steps (also known as actions) to follow, input, output and it validates one or more system requirements and generates a pass or f    **Test Case 1:**  **Test Objective:** Preventing the users to register with invalid password:   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **ITEM NO** | **TEST CONDITION** | **INPUT**  **SPECIFICATION** | **OUTPUT SPECIFICATION** | **PASS/FAIL** | | 1 | Login of user with invalid password. | 1.If all the details of user are correct  2.If password of user is invalid | Redirect to profile page  Show an error of incorrect password | PASS  PASS |   **SCREEN 1:the sigin page after clicking signin.**    **SCREEN 2: the google login page after clicking the signin with google.**    **SCREEN 3: pops up an error if an wrong password is entered.**    **SCREEN 4: if correct password is enterd then go to the profile.**    **Test Case 2:**  **Test objective:**preventing the user to enter null in add subject.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **ITEM NO** | **TEST CONDITION** | **INPUT**  **SPECIFICATION** | **OUTPUT SPECIFICATION** | **PASS/FAIL** | | 2 | Adding a subject into subject. | 1.If any filed is empty then it is invalid. | Display a message  “please fill out the field ” | PASS |   **Screen 2.1 Display a message please fill out the field:**  **Test Case 3:**  **Test Objective:** preventing the user to enter null in add tutorails.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **ITEM NO** | **TEST CONDITION** | **INPUT**  **SPECIFICATION** | **OUTPUT SPECIFICATION** | **PASS/FAIL** | | 3 | Adding a Chapter into subject. | 1.If any field is empty then it is invalid. | Display a message  “please fill out the field ” | PASS |   **Screen : when user enters the null value.**    **Test Case 4:**  **Test Objective:**preventing the user to enter null in Search bar.   |  |  |  |  |  | | --- | --- | --- | --- | --- | | **ITEM NO** | **TEST CONDITION** | **INPUT**  **SPECIFICATION** | **OUTPUT SPECIFICATION** | **PASS/FAIL** | | 4 | Inputting the Search String. | 1.If Search field is empty then it is invalid. | Display a message  “please fill out the field ” | PASS | |
|  |  |
|  |  |

**Screen: Screen when user inputs an empty search string.**

****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ITEM NO** | **TEST CONDITION** | **INPUT**  **SPECIFICATION** | **OUTPUT SPECIFICATION** | **PASS/FAIL** |
| 3 | Adding a Chapter into subject. | 1.If any field is empty then it is invalid. | Display a message  “please fill out the field ” | PASS |

**Test Case 5:** preventing the user to enter null in add tutorial.

**Screen : if any field in add tutorial is not entered.**

|  |  |
| --- | --- |
| **7.** | **CONCLUSION**  In this project, We explained how a user can search and select other user/tutor/learner to view his profile/videos to the user. It serves as a helpful approach for the users. It reduces the time taken by the student to learn the topics quickly. Thus the project is the user friendly approach.  Our project met all the requirements specified in Teachforfriend system and also it can handle various routine errors caused by irrelevant inputs given by user or administrator software. A possible future extension is to make it chat system. |
|  |  |

|  |  |
| --- | --- |
| **8.** | **REFERENCES**  **References:**   1. Object-oriented software engineering-Berned Bruegge, Allen H. Dutoit. 2. Object Oriented Software Engineering Practical Software Development using UML and Java 3. Web technology and design-C. Xavier. 4. [www.w3.schools.com](http://www.w3.schools.com) 5. www.projects-forum.com |