TUTORIAL BUILDER SYSTEM

A REPORT

Submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF ENGINEERING

In

COMPUTER ENGINEERING

By

GEORGE CHERIAN(6408)

NEIL CHETTIAR(6409)

OMKAR NARKAR(6438)

Under the guidance of

Mrs. Swati Ringe

(Assistant Proffessor)

DEPARTMENT OF COMPUTER ENGINEERING

FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING

FR. AGNEL ASHRAM, BANDRA (W),

MUMBAI - 400 050.

UNIVERSITY OF MUMBAI

(2014 – 2015)

FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING

FR. AGNEL ASHRAM, BANDRA (W),

MUMBAI - 400 050.

CERTIFICATE

This is to certify that the following students working on the project "Title of the project" have satisfactorily completed the requirements of the project in partial fulfillment of the course B.E in Computer Engineering of the University of Mumbai during academic year 2014-2015 under the guidance of "Name of the guide".

Submitted By: George Cherian(6408)

Neil Chettiar (6409)

Omkar Narkar(6438)

_____ _____

Guide Head of the Department

_____

Principal

CERTIFICATE

This is to certify that the project synopsis entitled "Title of the project" **submitted by the following students is found to be satisfactory and is approved as a partial fulfillment of the requirement for the degree of** Bachelor of Engineering **in** Computer Engineering.

George Cherian

Neil Chettiar

Omkar Narkar

Internal Examiner	External Examiner
(Signature)	**(Signature)**
Name:	**Name:**
Date:	**Date:**

Seal of the Institute

DECLARATION OF THE STUDENT

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources.

We also declare that we have adhered to all principles of academic honesty and integrity and have not Mis-represented or fabricated or falsified any idea / data / fact / source in my submission.

We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Signature of the student **Date:**
(George Cherian)
(6408)

Signature of the student **Date:**
(Neil Chettiar)
(6409)

Signature of the student **Date:**
(Omkar Narkar)
(6438)

Abstract

The Tutorial Builder System is being built as a solution to the long standing problem of learning topics via the internet being a rather difficult and arduous process, with all the information needed to properly understand a topic scattered about multiple pages and sometimes the pages which hold information in an easy to understand format may not even be found. Thus our system aims to solve this problem by crawling the internet for data relevant to a single topic and then appropriately ranking it based on a set of criteria and passing the highest ranked tutorial out of all the raw data found to the user so that he may easily understand the topic by only looking through a single website which holds all the necessary information presented in a timely and comparatively smaller format as compared to the data which is found after researching the same topic via a search engine.

The Tutorial Builder System will mainly have a user interface, a tutorial cache, a crawler infrastructure,a general controller system and a ranking infrastructure which will all be used in conjunction with each other to give the user a tutorial for each topic.

Table of Contents

Abstract	i
List of Figures	iii
1. Chapter One: Introduction	1
1.1. Objective	1
1.2. Problem Statement	1
1.3. Scope	2
1.4. Applications	2
2. Chapter Two: Literature Review	3
2.1. Web Crawler	3
2.2. Data Ranking	8
2.3. Machine Learning	10
3. Chapter Three: Proposed System	12
3.1. Requirement Analysis	12
3.1.1.Functional Requirements	12
3.1.2.Non-Functional Requirements	12
3.2. Methodology	13
3.3. System Architecture	16
3.4. UML Diagrams	19
4. Chapter Four: Hardware and Software Requirements & Implementation plan for next Semester	31
4.1. Hardware and Software Requirements	31
4.2. Implementation Plan	31
5. Chapter Five: Summary and Conclusion	32
5.1. Summary and Conclusion	32
5.2. Future Scope	32
6. References	33

List of Figures

Figure2. 1: Web Crawler Architecture	3
Figure 2.2: Weight Table for Ontology	5
Figure 2.3: FlowChart to check Domain of Webpage	7
Figure 3.1: System Architecture	16
Figure 3.2: System Flow	18
Figure 3.3: System UseCase	19
Figure 3.4: Register Activity Diagram	20
Figure 3.5: Login Activity Diagram	21
Figure 3.6: Change Registration details Activity Diagram	22
Figure 3.7: New Search Activity Diagram	23
Figure 3.8: View Search History Activity Diagram	24
Figure 3.9: Register Sequence Diagram	25
Figure 3.10: Login Sequence Diagram	26
Figure 3.11: Change Registration Details Sequence Diagram	27
Figure 3.12: New Search Sequence Diagram	28
Figure 3.13: View previous Search Activity Diagram	29
Figure 3.14: Class Diagram	30

Chapter One

Introduction

As it stands today every time we need to gather information about any topic we need to look for data across multiple sites, read through pages and pages of data which are scattered over a host of different websites. Even sites which have data in a single place only deal with giving information which is not easy to use or understand and often prone to human errors during editing.

The Tutorial Builder System that we propose will be used to gather data from different places over the internet and compile all of this data into a set of chapters and data which is easy to understand for the user.

1.1. Objective

The Tutorial Builder System will take data from multiple sources and combine them into a bank with the data following which the system will read the data from the various sources and strip it down, clean the data up and merge it with data from other sources to get a final version of the data which is user friendly and easy to understand. With this we can eliminate the need to go through multiple web pages to understand or learn a topic with all the information being consolidated in a single space.

1.2. Problem Statement

The Tutorial Builder System is a software based system which will use data collection algorithms, data ranking algorithms, word processing and data merging algorithms to find the data regarding a certain topic and then sort through the data on the topic, combine all of the data in a easy to understand and intuitive manner.

Once the data is put into a database we will proceed to take a apart the data and then convert it back into a properly formed tutorial.

The user will give an input to the system in the form of the topic for which they want the tutorial and preferably the stream under which the tutorial comes followed by which our system will search through the databank for the appropriate tutorial and give it to the user to read the data from the databank.

The back end of the system will compile the data for the databank, by reading the data from various sources and then processing them appropriately and storing the data within the databank.

1.3.Scope

The scope of this project would be the gathering of data from the internet and checking its relevance, followed by the ranking of gathered data and storing into a database. The data stored in the database will then have to be appropriately processed and combined with other data from various sources and then put into a databank from where the GUI will read the data and present it to the user.

1.4.Applications

The Tutorial Builder System will have a wide variety of applications in any and all fields. When successfully implemented the system can make learning any subject easier. Instead of the user going to the trouble of accessing multiple sites the system will provide the user with a one stop destination for information about any topic needed by the user.

Currently we plan to implement this system from a computer science standpoint taking into account mostly topics which are within the domain of computer science.

Chapter Two

Literature Review

2.1. Web Crawler

- Architecture

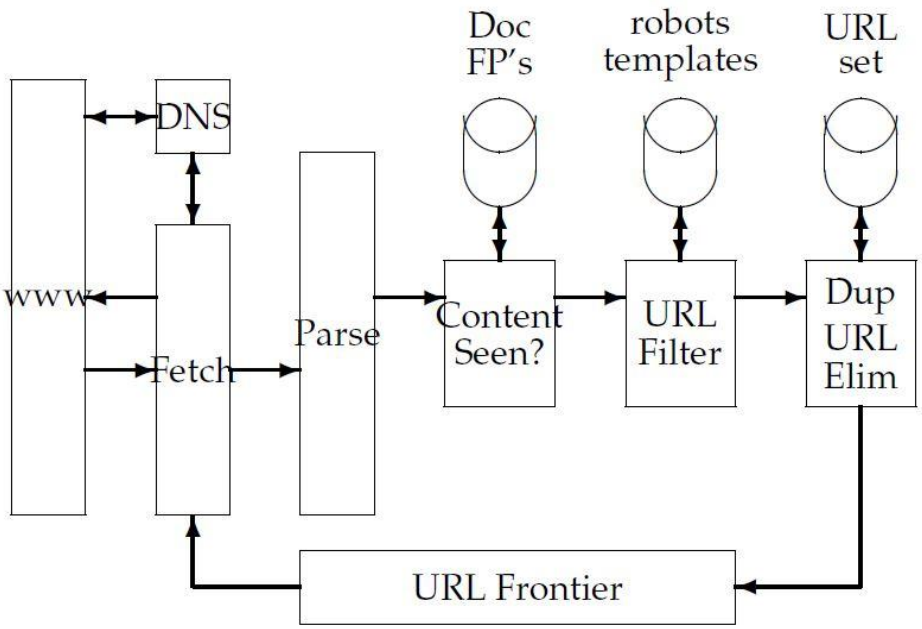

Figure 2.1: Web Crawler Architecture

The simple scheme outlined above for crawling demands several modules that fit together:

1. The URL frontier, containing URLs yet to be fetched in the current crawl (in the case of continuous crawling, a URL may have been fetched previouslybut is back in the frontier for re-fetching).

2. A *DNS resolution* module that determines the web server from which tofetch the page specified by a URL.

3. A fetch module that uses the http protocol to retrieve the web page at aURL.

4. A parsingmodule that extracts the text and set of links froma fetchedweb page.

5. A duplicate elimination module that determines whether an extracted link is already in the URL frontier or has recently been fetched.

A crawler thread begins by taking a URL from the frontier and fetching the web page at that URL, generally using the http protocol. The fetched page is then written into a temporary store, where a number of operations are performed on it. Next, the page is parsed and the text as well as the links in it are extracted. The text (with any tag information – e.g., terms in boldface) is passed on to the indexer. Link information including anchor text is also passed on to the indexer for use in ranking. In addition, each extracted link goes through a series of tests to determine whether the link should be added to the URL frontier.

First, the thread tests whether a web page with the same content has already been seen at another URL. The simplest implementation for this would use a simple fingerprint such as a checksum (placed in a store labeled "Doc FP's" in the figure). A more sophisticated test would use shingles instead of fingerprints.

Next, a *URL filter* is used to determine whether the extracted URL should be excluded from the frontier based on one of several tests. For instance, the crawl may seek to exclude certain domains (say, all .com URLs) – in this case the test would simply filter out the URL if it were from the .com domain.

A similar test could be inclusive rather than exclusive. Many hosts on the Web place certain portions of their websites off-limits to crawling, under a standard known as the *Robots Exclusion Protocol*.

This is done by placing a file with the name robots.txt at the root of the URL hierarchy at the site.

Finally, the URL is checked for duplicate elimination: if the URL is alreadyin the frontier or (in the case of a non-continuous crawl) already crawled,we do not add it to the frontier. When the URL is added to the frontier, it isassigned a priority based on which it is eventually removed fromthe frontierfor fetching. Certain housekeeping tasks are typically performed by a dedicated thread.This thread is generally quiescent except that it wakes up once every fewseconds to log crawl progress statistics (URLs crawled, frontier size, etc.), decidewhether to terminate the crawl, or (once every few hours of crawling)checkpoint the crawl. In checkpointing, a snapshot of the crawler's state (say,the URL frontier) is committed to disk. In the event of a catastrophic crawlerfailure, the crawl is restarted from the most recent checkpoint.

- Additional Features

Relevance Calculation

In this section we describe our own algorithm depending on which we calculate relevancy of a Web page on a specific domain.

Weight Table: We want to add some weights to each term in the ontology. The strategy of assigning weights is that, the more specific term will have more weight on it. And the terms which are common to more than one domain have less weight. The sample Weight table for some terms of a given ontology of the table shown below:

Ontology terms	Weight
Assistant Professor	1.0
Assistant	0.6
Student	0.4
Worker	0.1
Publication	0.1

Figure 2.2: Weight Table for ontology

Relevance calculation algorithm. In this section we design an algorithm how relevance score of a Web page is calculated.

INPUT: A Web page (P), a weight table.

OUTPUT: The relevance score of the Web page(P).

Step1 Initialize the relevance score of theWeb page (P) to 0. RELEVANCE_P=0.

Step2 Select first term (T) andcorresponding weight (W) from the weighttable.

Step3 Calculate how many times the term (T)occurs in the Web page P. Let the number ofoccurrence is calculated in COUNT.

Step4 Multiply the number of occurrencecalculated at step3 with the weight W. Letcall this TERM_WEIGHT. And TERM_WEIGHT=COUNT* W.

Step5 Add this term weight to RELEVANCE_P.So new RELEVANCE_P will be, RELEVANCE_P =RELEVANCE_P + TERM_WEIGHT.

Step6 Select the next term and weight fromweight table and go to step3, until all theterms in the weight table are visited.

Step7 End.

Checking Domain of a Web page

Using ontological knowledge we can find relevant

Web pages from the Web.

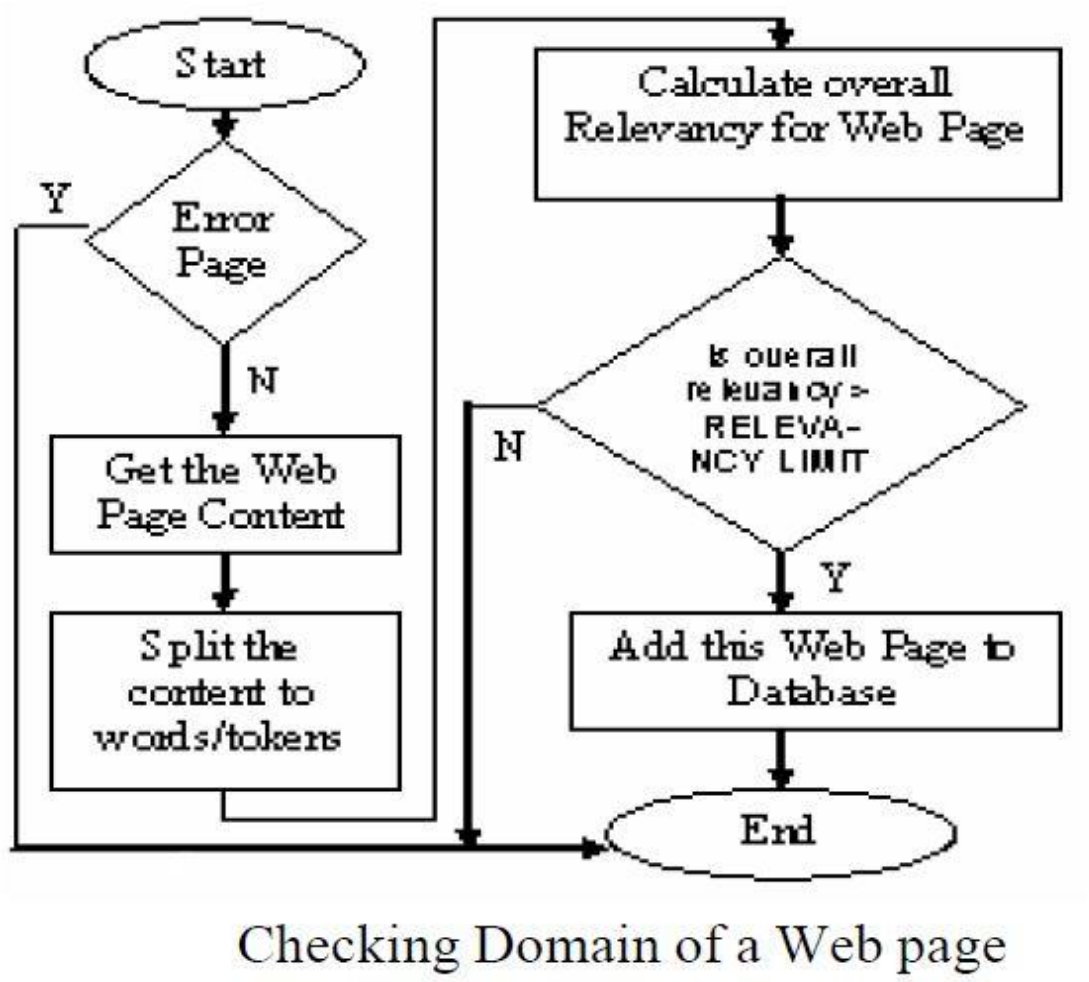

Figure 2.3: Flowchart to check domain of Web page

2.2.Data Ranking

To rank the data which has been derived from the crawler we can make use of a set of data ranking criteria which have been defined for another paper on grading essays.

1) Its visual nature
A descriptive sentence awes the reader and prompts their imagination.

2) Inclusion of pronouns

3) Its beautiful words
Beautiful word choices are thought to increase an essay's elegance, thereby its score.

4)Its emotive effectiveness
A very dry and emotionless essay is not powerful. The Subjectivity Lexicon from MPQA provides a list of words and their sentiments (positive, negative, neutral, or both) and the strength of those sentiments (MPQA, 2005). The resulting features are proportions of sentiments and strength individually and combined for the entire essay or for given sentiments and strength.

5) Its maturity
Our vocabulary expands as we grow older. Therefore, in a sense, as we mature, so does our vocabulary.

Based on the above 5 criteria for development of a grading system for essays we developed our own criteria which we could effectively use to compare and rank the data on the topics given to us so that we could provide the user with the best tutorial. The explanation of the same is provided in methodology

1) Ease of Understandability

The tutorial finally put forward to the user should be easily understandable and the words used should not be unnecessary complicated and thus this is the first criteria for data ranking

2) Diagrams

Diagrams are a very important criterion for understanding almost any topic and the more diagrammatic representations of any topic the easier that topic is to understand. Thus the presence of diagrams is a key factor in the ranking of a data set.

3) Descriptivity

Whether a tutorial can provide analogies for the things they are trying to describe is a very important part in what will end up making a particular tutorial easier for users to understand

eg: Comparing an object to a car

4) Examples and Solved Problems

The more examples and solved problems to support a particular concept that are found in a tutorial the higher the ranking of that tutorial should be because the sums or problems can help in fast forwarding the understanding of a concept.

5) Comments

Most sites also have comments from other human users who have come upon this data and read it in the past, these comments can help the system identify how good or bad the data is and whether it is suitable as a tutorial.

2.3.Machine Learning

To implement two of the previously mentioned 5 criteria, namely ease of understandability and comments, we intend to incorporate machine learning to evaluate the text and allow us to reach a viable conclusion.
We base our logic off of a paper titled *'Subjectivity Word Sense Disambiguation'* which deals with the sentiment analysis or subjectivity of words used in a sentence.

The automatic extraction of opinions, emotions, and sentiments in text (*subjectivity analysis*) to support applications such as product review mining, summarization, question answering, and information extraction is an active area of research in NLP.
Many approaches to opinion, sentiment, and subjectivity analysis rely on lexicons of words that may be used to express subjectivity. Examples of such words are the following (in bold):

(1) He is a **disease** to every team he has gone to.
Converting to SMF is a **headache**.
The concert left me **cold**.
That guy is such a **pain**.

Knowing the meaning (and thus subjectivity) of these words would help a system recognize the negative sentiments in these sentences.
Most subjectivity lexicons are compiled as lists of keywords, rather than word meanings (senses).
However, many keywords have both subjective and objective senses. False hits – subjectivity clues used with objective senses – are a significant source of error in subjectivity and sentiment analysis. For example, even though the following sentence contains all of the negative keywords above, it is nevertheless objective, as they are all false hits:

(2) Early symptoms of the **disease** include severe **headaches**, red eyes, fevers and **cold** chills, body **pain**, and vomiting.

To tackle this source of error, they define a new task, *subjectivity word sense disambiguation(SWSD)*, which is to automatically determine which word instances in a corpus are being used with subjective senses, and which are being used with objective senses. They hypothesize that SWSD is more feasible than full word sense disambiguation, because it is more coarse grained – often, the exact sense need not be pinpointed. They also hypothesize that SWSD can be exploited to improve the performance of contextual subjectivity analysis systems via sense-aware classification.

The paper consists of two parts. In the first part, they build and evaluate a targeted supervised SWSD system that aims to disambiguate members of a subjectivity lexicon. It labels clue instances as having a subjective sense or an objective sense in context. The system relies on common machine learning features for word sense disambiguation (WSD). The performance is substantially above both baseline and the performance of full WSD on the same data, suggesting that the task is feasible, and that subjectivity provides a natural coarse grained grouping of senses.
The second part demonstrates the promise of SWSD for contextual subjectivity analysis. First, they show that subjectivity sense ambiguity is highly prevalent in the MPQA opinion-annotated corpus (Wiebe et al., 2005; Wilson, 2008), thus establishing the potential benefit of performing SWSD. Then, we exploit SWSD to improve performance on several subjectivity analysis tasks, from subjective/objective sentence-level classification to positive/negative/neutral expression level classification.

The benefit of using such a system is that it reduces the accuracy of full word sense disambiguation that is provided by a standard subjectivity-annotated corpus (MPQA). For example, in the case of comments, we do not need to know how negative or how positive the comments of the users are, just that if it is negative or positive.

Chapter Three

Proposed System

The system that we plan to implement will be directly user input related where the use will enter the input for the subject they would like to search for and then choose a stream into which the search will go, following this they can choose a particular sub topic or choose from a list of topics generated by the system.

3.1. Requirement Analysis

3.1.1. Functional Requirements

1. User Registration with facility to enter multiple details.
2. Make the user choose a default stream of enquiry
3. User should be able to search for subjects in various streams with default stream always being chosen stream
4. User should be able to modify details of account
5. User should be able to specify a topic within a subject to search for
6. If user is not sure of which topic he wants or wants multiple topics in a subject the system should provide the user with a list of all topics under the system.
7. User should be able to view and sort search history
8. User should be able to view previously made searches.

3.1.2. Non-Functional Requirements

1. Searches should be fast
2. Details of every user should be private unless otherwise specifies by the user.

3. Data should be coherent
4. Tutorials should be easy to understand.
5. System should be reliable
6. System should not be vulnerable to attacks.

3.2. Methodology

We plan to set up a User Interface as either a web application or an Android app, the user will first register themselves onto the system giving all the necessary details and their main stream of interest. After Registering the user will be able to change any of the details entered except for their username. The user can then login any time using the username and password they provided when registering. To get a tutorial the user will enter the subject name on which they want the tutorial, the stream will be automatically set as the stream they had chosen while registration but if they so choose the user can change the stream for any particular search.

The User can then either provide a particular topic on which he wants a tutorial or let the system look for data and then provide a list of topics. Either way a tutorial search request is submitted to the system, The System then proceeds to use crawlers to go through the internet and pick up stream relevant and subject relevant data and put this data into a data base of all the data picked up from the internet.

Once this data has been collected from the internet the system will then proceed to go through the data according to 6 main criteria, relevance of the document with respect to the topic has already been decided by the crawler and hence only the next 5 are explained.

1) Ease of Understandability

   The tutorial finally put forward to the user should be easily understandable and the words used should not be unnecessary complicated and thus this is the first criteria for data ranking

   To check if a piece of text is easily understandable or not we compare the text with a text or language corpus and based on the word scores of the words used we can generate an average understandability score for every document

2) Diagrams

Diagrams are a very important criterion for understanding almost any topic and the more diagrammatic representations of any topic the easier that topic is to understand. Thus the presence of diagrams is a key factor in the ranking of a data set.

The diagrams in the data derived from the crawler are found and then an average score of total diagrams to length of document is found and applied to the ranking.

3) Descriptivity

Whether a tutorial can provide analogies for the things they are trying to describe is a very important part in what will end up making a particular tutorial easier for users to understand

eg: Comparing an object to a car

To find out how descriptive or comparative a particular tutorial is we can use a very fixed corpus which contains the words " like", "can be compared to", "is similar to", "works like a", etc.

4) Examples and Solved Problems

The more examples and solved problems to support a particular concept that are found in a tutorial the higher the ranking of that tutorial should be because the sums or problems can help in fast forwarding the understanding of a concept.

Examples or solved problems are very clearly defined and can be easily found because they are preceded by very clear demarcations like "eg:","Solving",etc

5) Comments

Most sites also have comments from other human users who have come upon this data and read it in the past, these comments can help the system identify how good or bad the data is and whether it is suitable as a tutorial.

The Subjectivity Lexicon from MPQA provides a list of words and their sentiments (positive, negative, neutral,or both) and the strength of those sentiments and using the

same we can find out whether the comments convey a strong emotions about the text and accordingly decrease or increase its ranking.

Followed by which the data needed for various topics is taken from various sources and then made into a tutorial. A copy of the tutorial is stored into a tutorial database and then sent to the user.

Depending on whether the user specified a topic or not, either the data needed is returned or a list of topics is returned to the user respectively. The user will have all the data in every topic ready with them from various sources.

The user will also have the option to view their search history and see all the previous searches made and also have the option to view a single previous topic or subject searched for and the data contained within that topic.

System Architecture

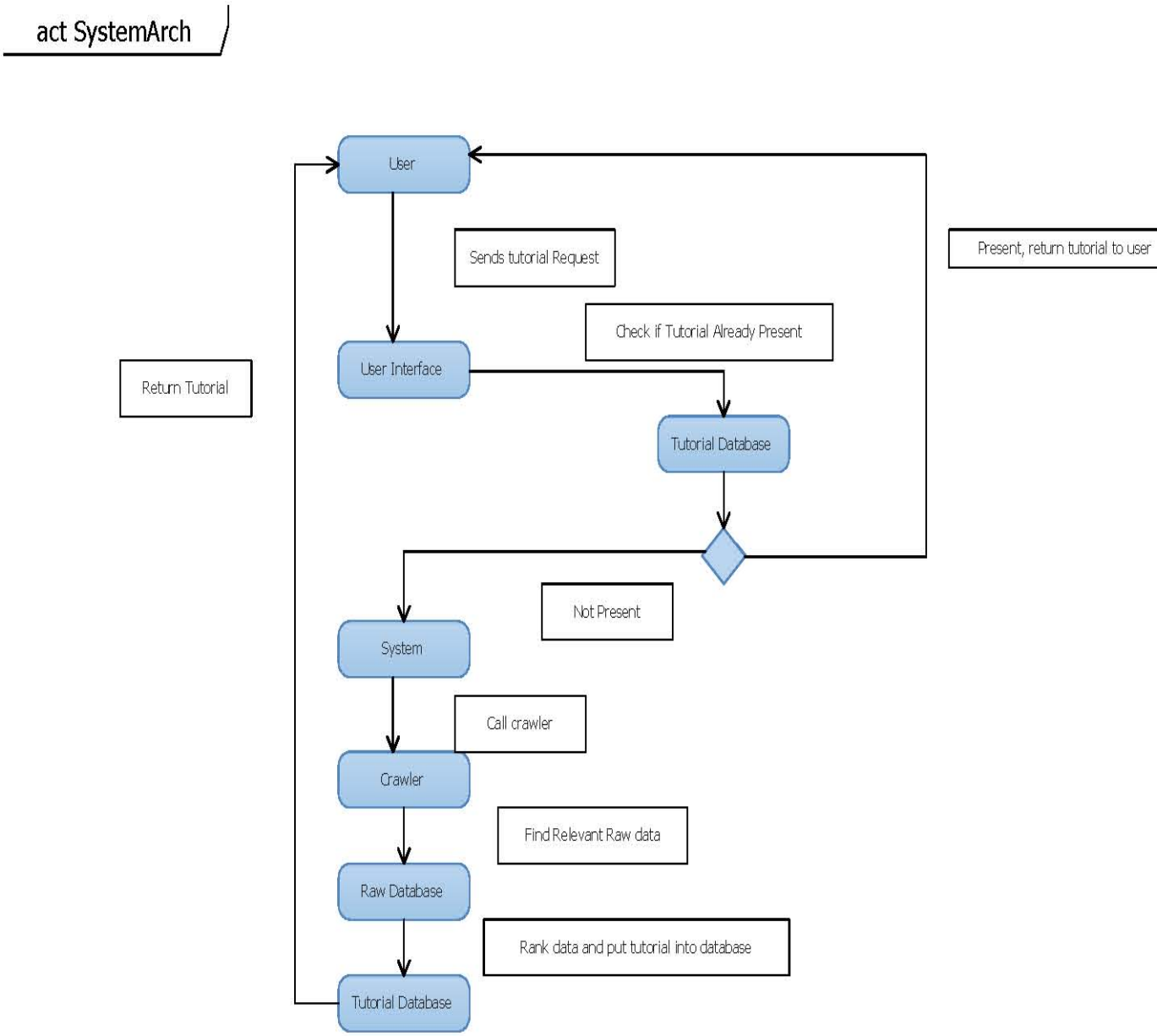

Figure 3.1: System Architecture

1. User Interface

   The interface between the user and the system it can have a direct connection to the tutorial cache or be linked to the system controller itself.

2. Tutorial Database/Cache

All previous searches and data is put into this cache so that if someone wants to access a tutorial for a topic that has already been searched for it can be done easily.

3. System Controller

   The main controller of the entire system that deals with the user interface the crawler system and the data ranking system as well as the tutorial cache.

4. Crawler

   The crawler picks up data from the internet which is relevant to the topic and then puts the data into the raw data database so that the data ranking algorithm may perform its job.

5. Raw Database

   This houses all the raw data taken by the crawler from the internet in no particular order except based on topics.

6. User

   The final user who queries for tutorials via the user interface and is the one to finally read the returned tutorial.

System Flow

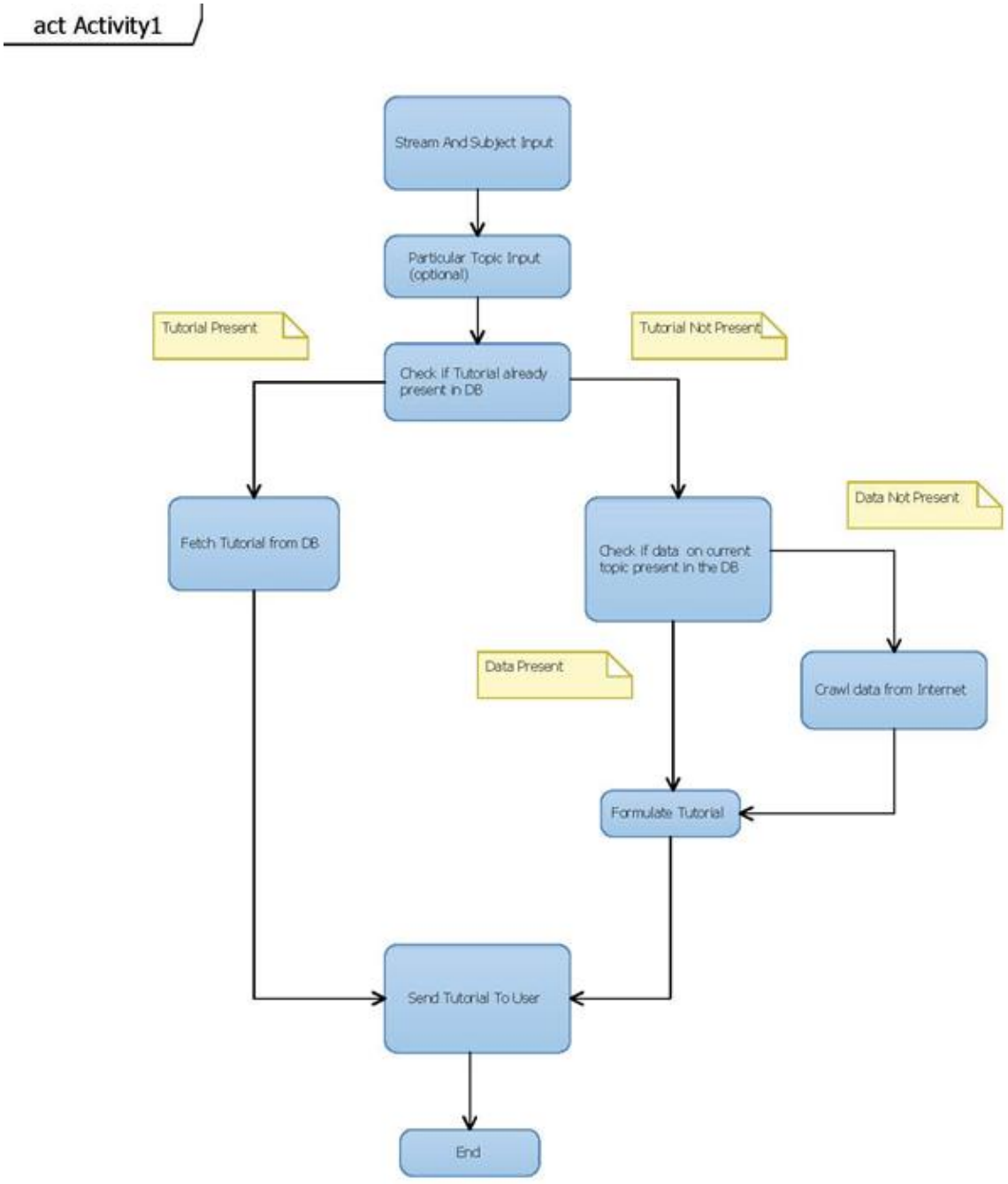

Figure 3.2: System Flow

System Use Case

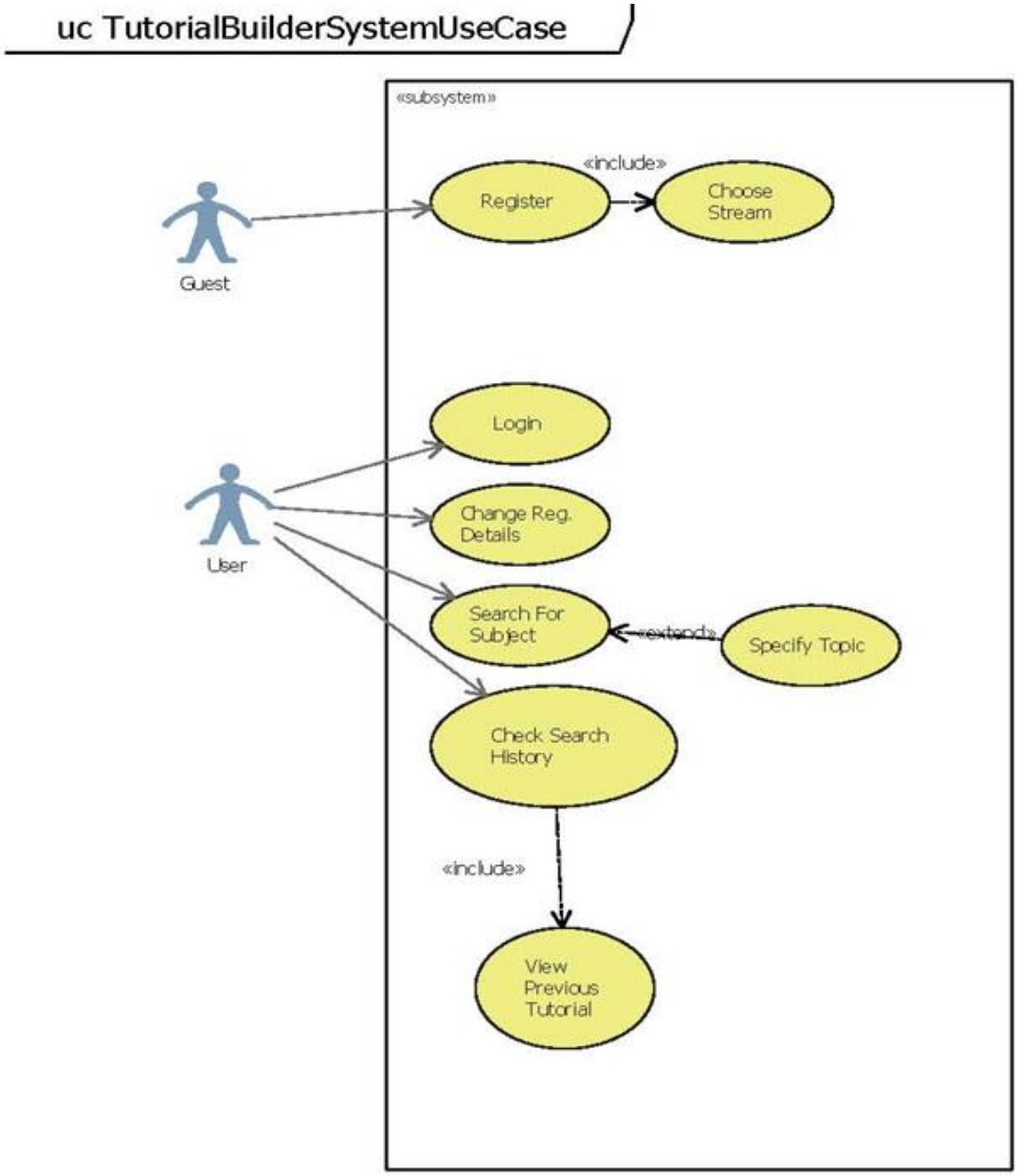

Figure 3.3: System UseCase

Register ActivityDiagram

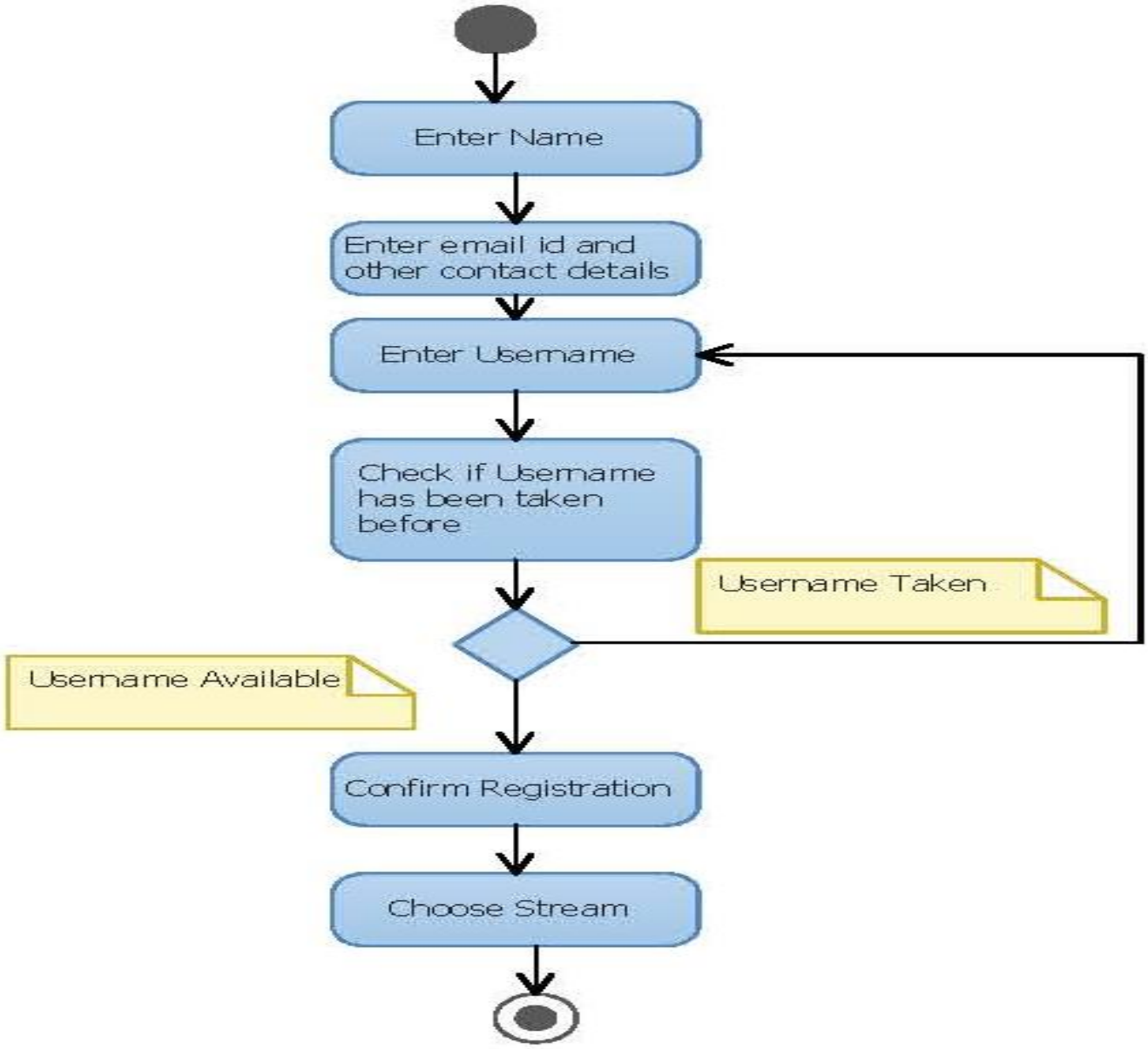

Figure 3.4: Register Activity Diagram

Login Activity Diagram

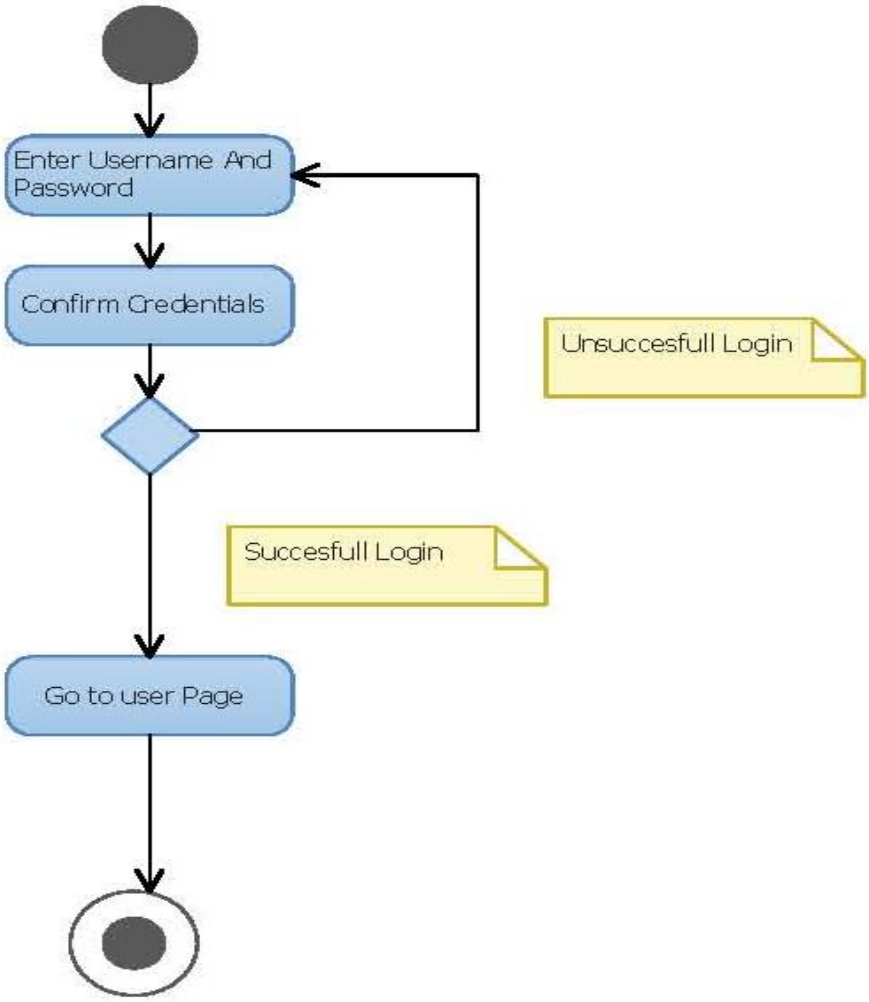

Figure 3.5: Login Activity Diagram

Change Registration Details Activity Diagram

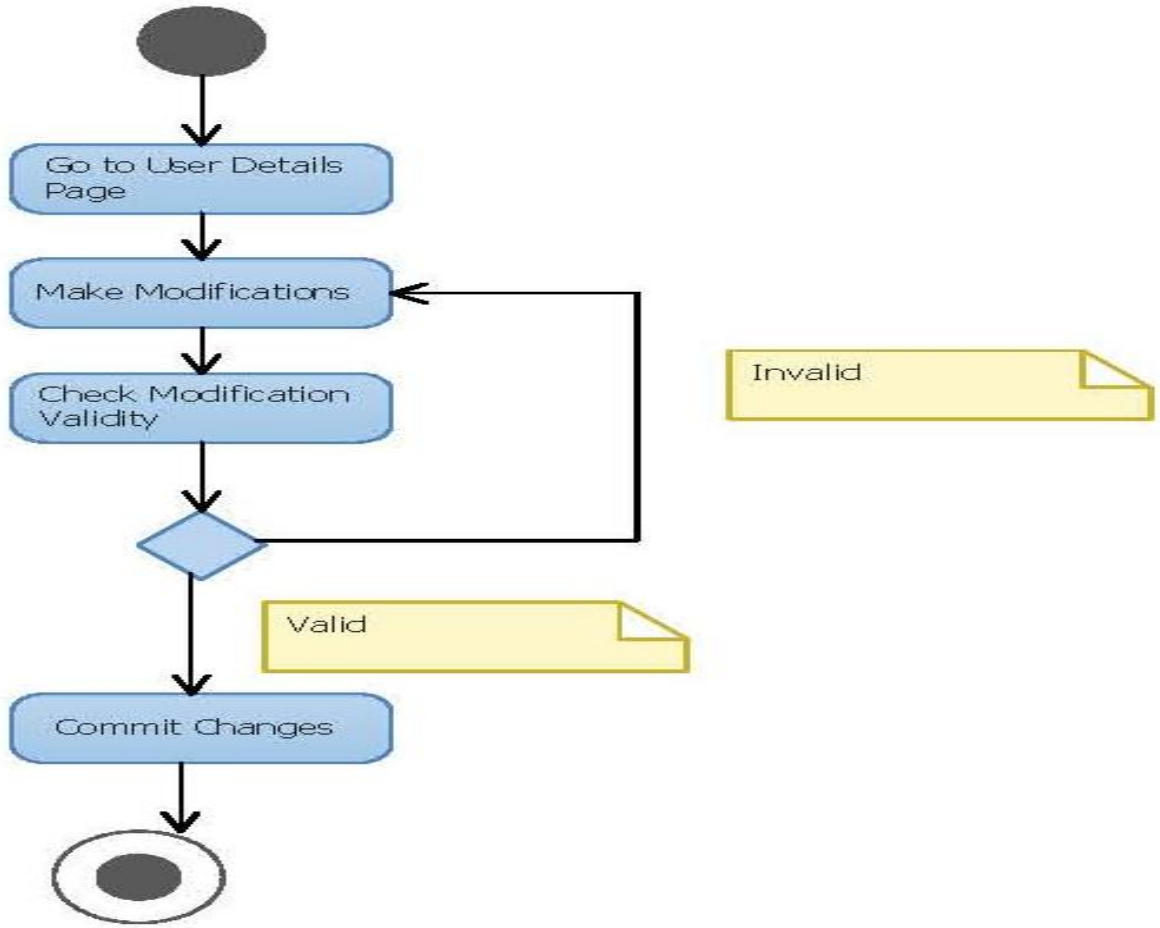

Figure 3.6*: Change Registration Details Activity Diagram*

New Search Activity Diagram

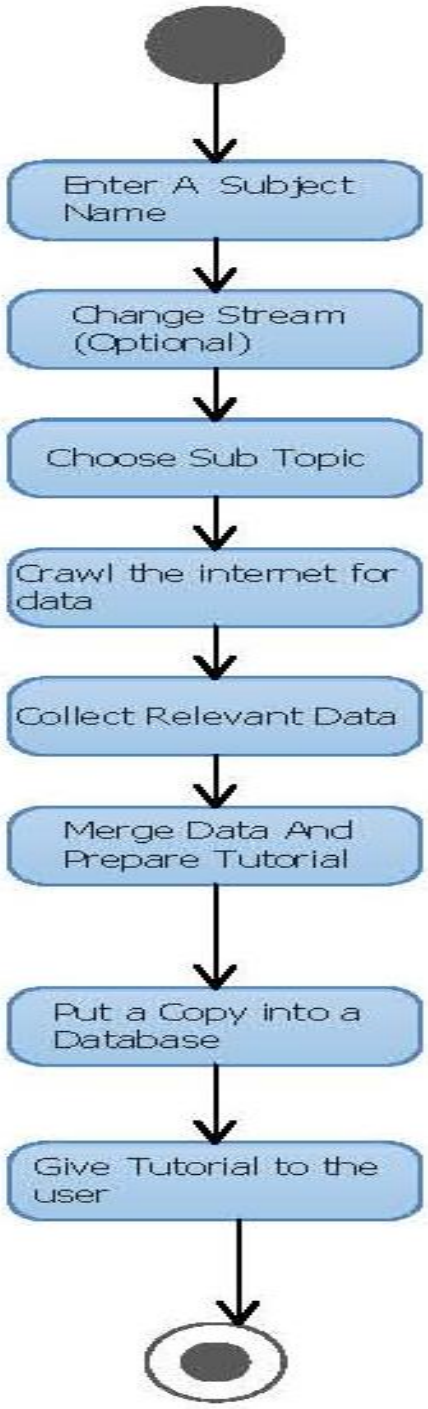

Figure 3.7*: New Search Activity Diagram*

View Search History Activity Diagram

Figure 3.8*: View Search History Activity Diagram*

Register Sequence Diagram

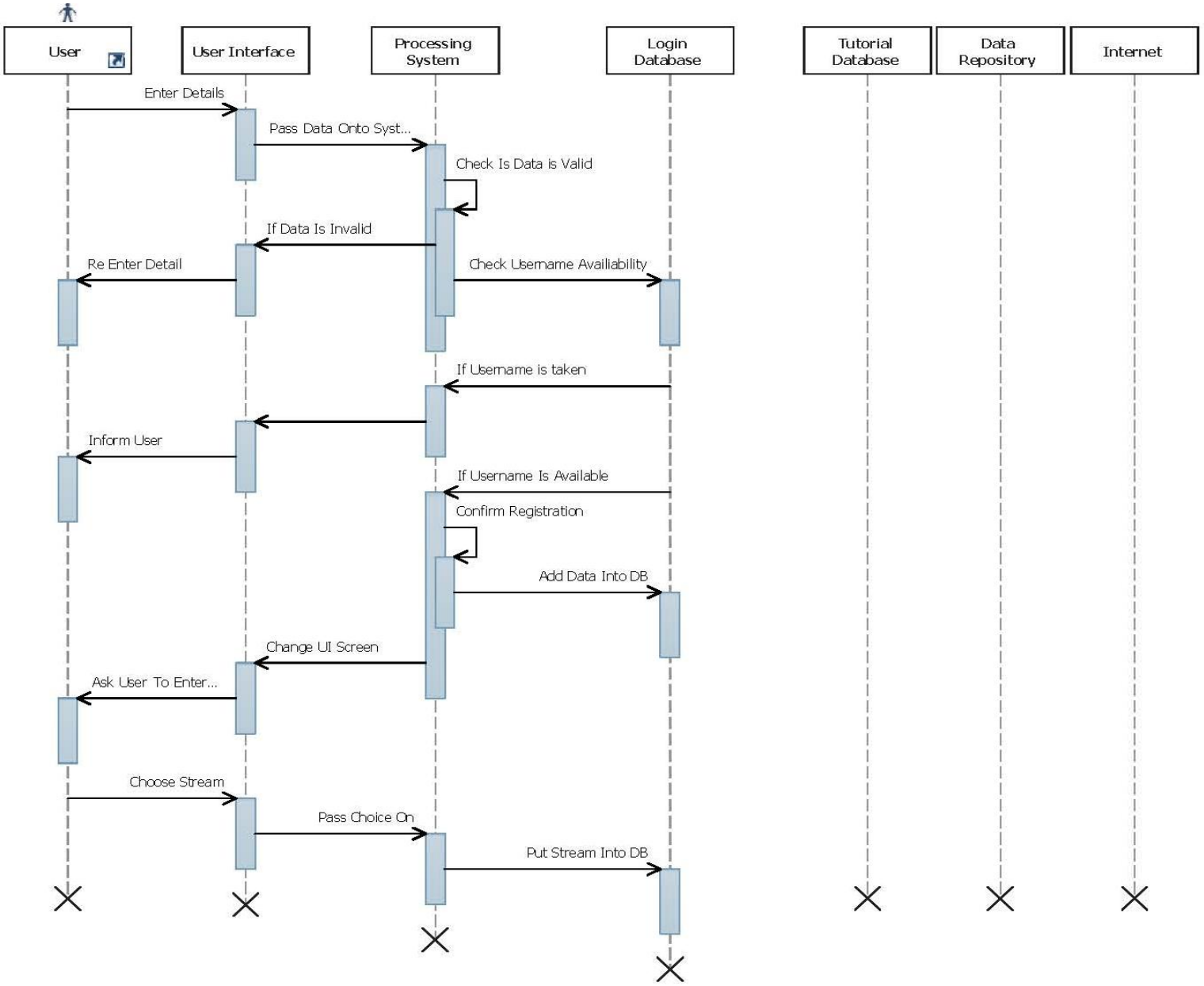

Figure 3.9*: Register Sequence Diagram*

Login Sequence Diagram

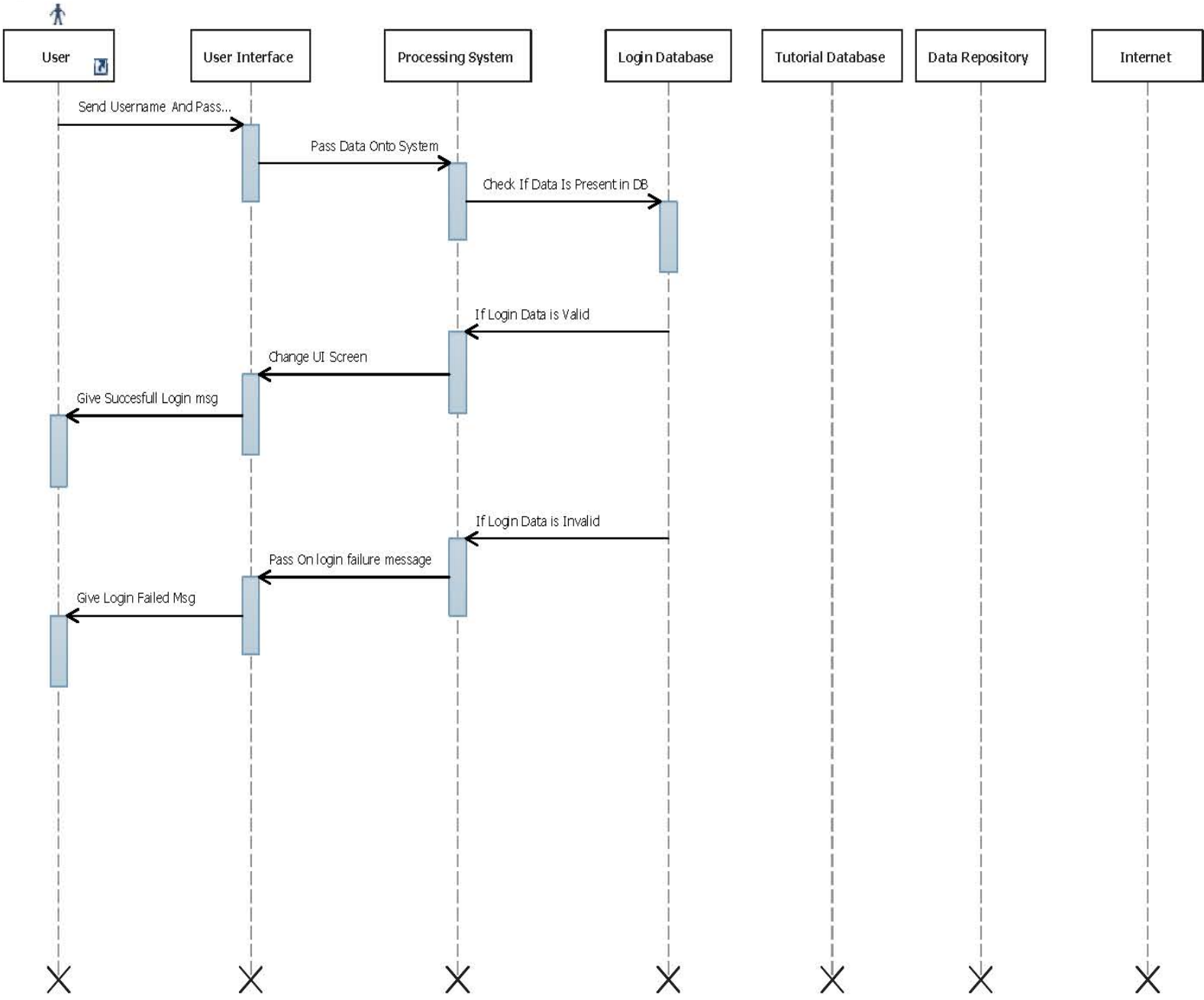

Figure 3.10*: Login Sequence Diagram*

Change Registration Details Sequence Diagram

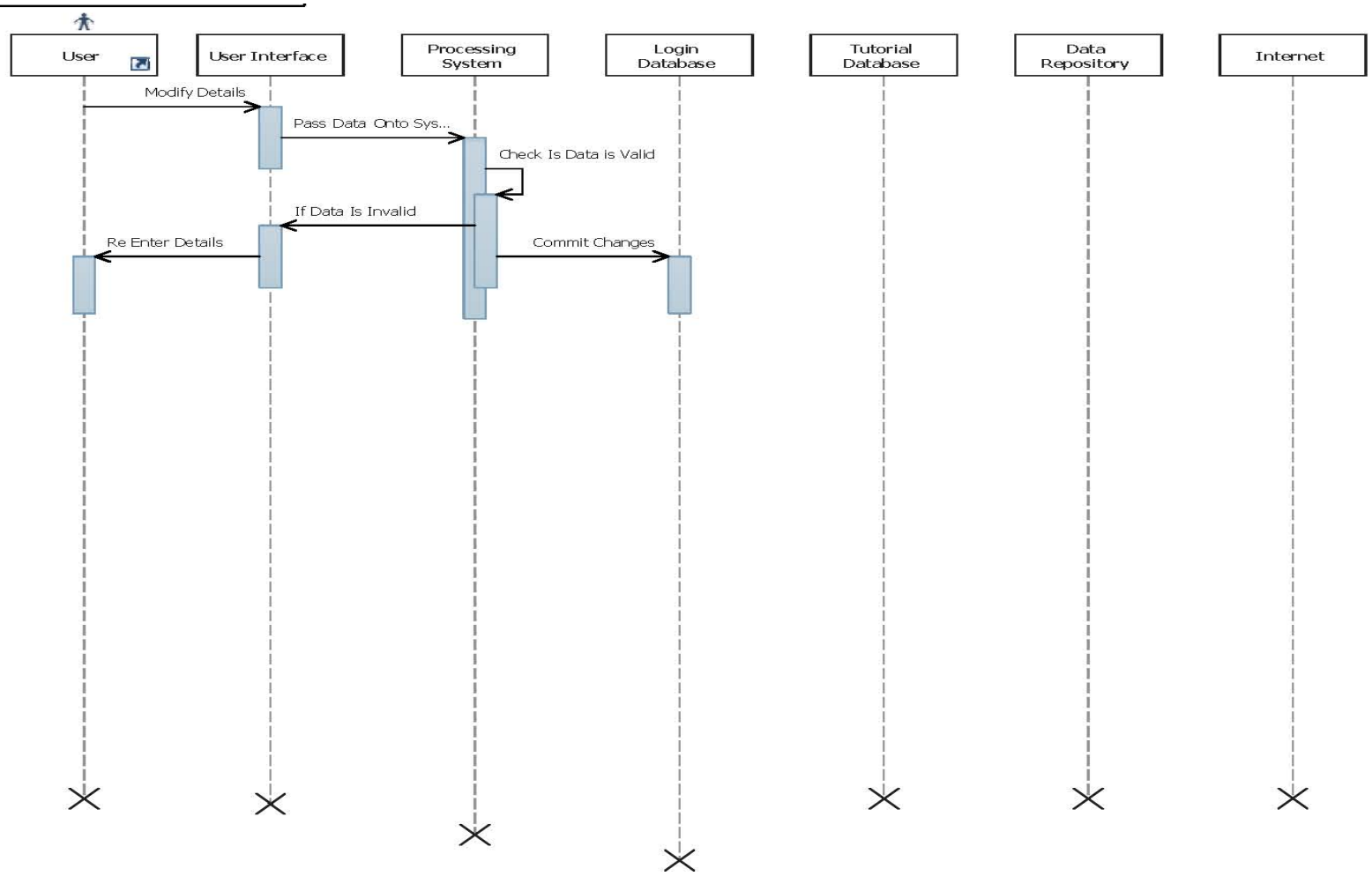

Figure 3.11*: Change Registration Details Sequence Diagram*

New Search Sequence Diagram

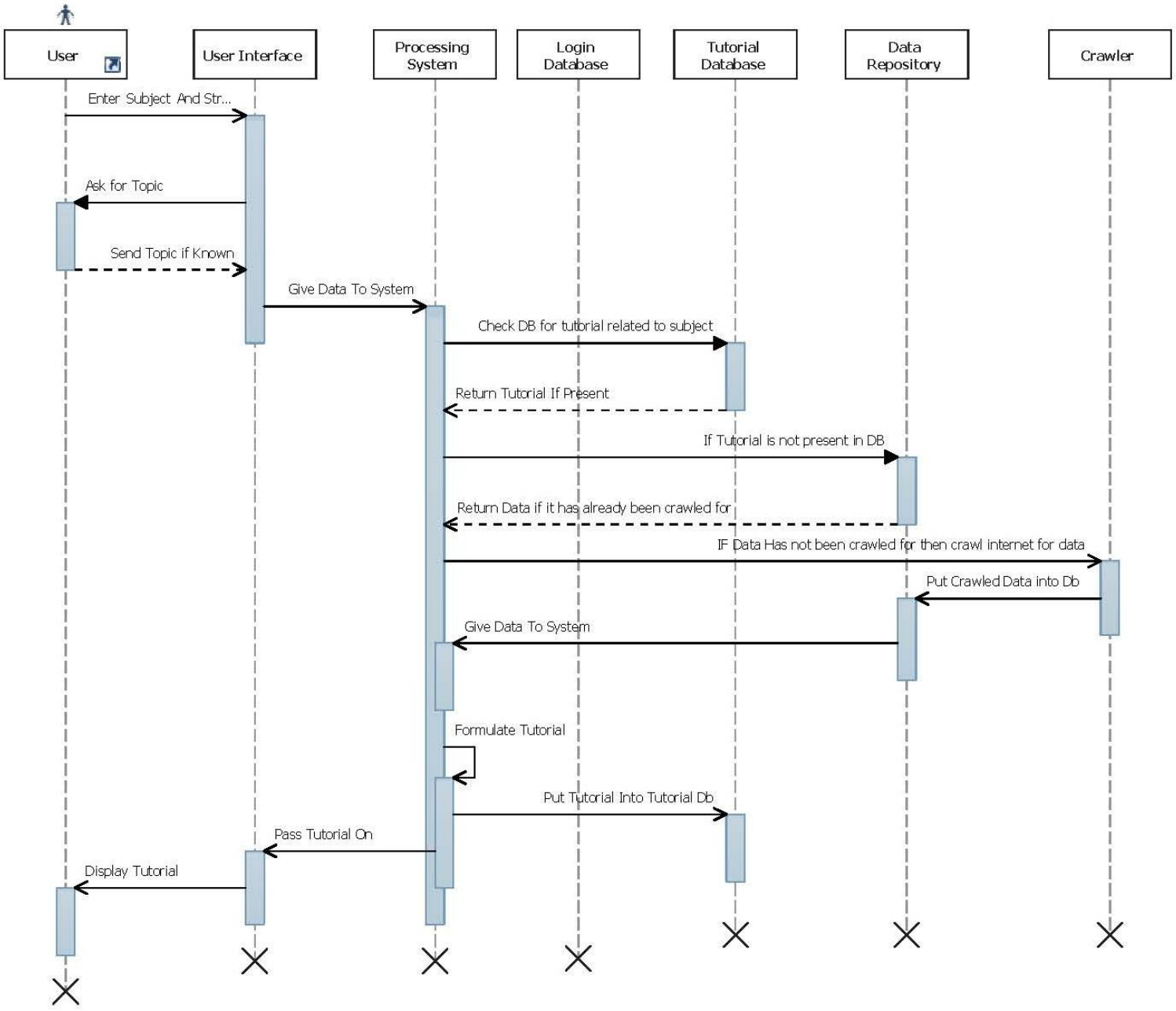

Figure 3.12 *: New Search Sequence Diagram*

View Previous Search Activity Diagram

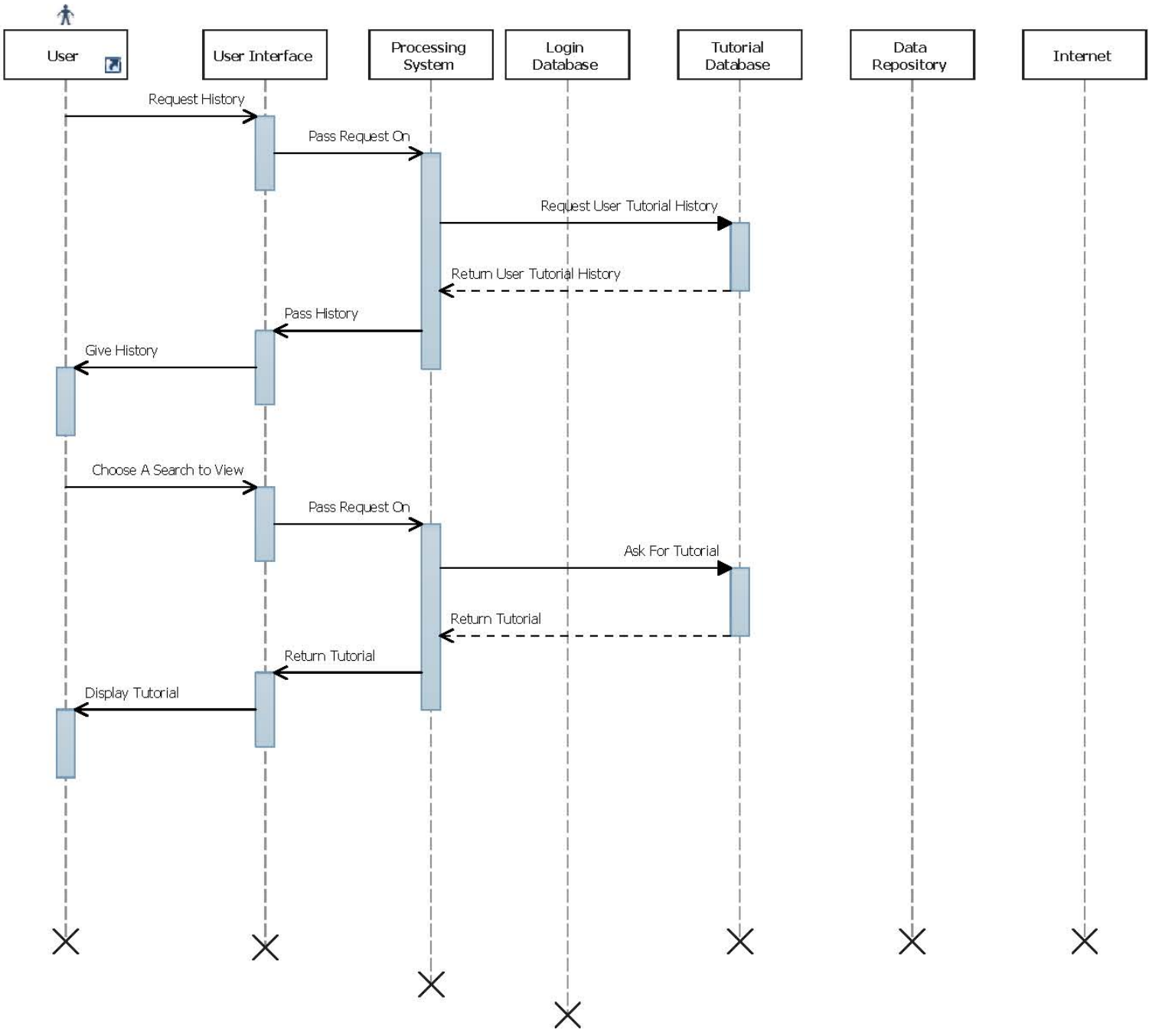

Figure 3.13*: View Previous Search Activity Diagram*

Class Diagram

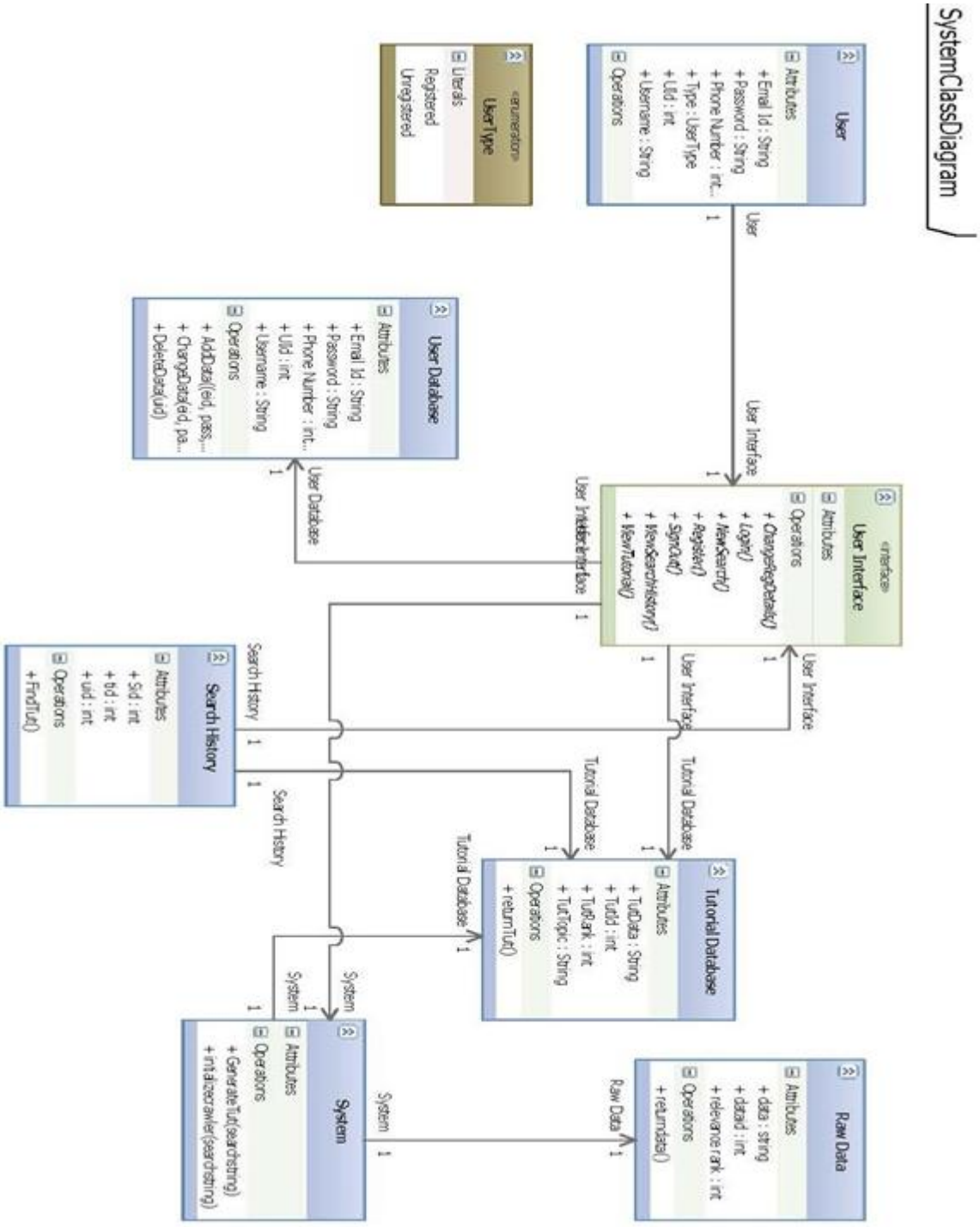

Figure 3.14*: Class Diagram*

Chapter Four

Hardware software requirements and Implementation Plan for next semester

4.1.Hardware/Software Requirements

The platform used for developing the system is Java language along with support from NetBeans IDE as well as PYTHON and the database used for the same will be MySQL. Any machine satisfying minimum requirements of 512 MB RAM and Intel Pentium Series or AMD equivalent processors with Java installed on their system can host the main system to rank the data and give the tutorial to the user, while a secondary system or the same system can be used to crawl the internet for relevant pages. The more powerful the system is, faster it will be able to process and rank the data as well as crawl the internet.

4.2.Implementation Plan

The project needs the following things which need to be implemented and finished

1) A Crawler which ranks data according to relevance
2) The data ranking system
3) The User Interface for the system
4) The system controller which will control all of the above
5) The Databases needed for the system, mainly raw database and tutorial database

Chapter Five

Summary & Conclusions

5.1.Summary & Conclusions

Thus the general implementation details have been figured out and the system architecture along with all the systems details and methodology is in place alongside the UML diagrams for the system itself.

The details for every part of the system have been found and the requirements for every part of the system have been analysed and the algorithms along which the system should work and the software to be used for the same have been found.

The crawler system to take data from the internet and put the data into a new database has been studied and architectures for that system have also been illustrated and displayed in the report.

The criteria to figure out which tutorial is the most suited for any particular topic have been discussed and the way to figure out if a topic follows those criteria have also been found and discussed.

5.2.Future Scope

The Project can be expanded to use and cover the following areas

- We can use machine learning to extract only relevant information from pages making the ranking and crawling more efficient.
- The system can learn data from the various data taken and create a tutorial of its own
- The project can be expanded to take data from all stream rather than just computer science

References

1. Automated Essay Scoring Using Machine Learning - Shihui Song, Jason Zhao
2. A Machine Learning Approach to WebpageContent Extraction - Jiawei YaoDepartment ofCSStanford University, XinhuiZuoDepartment of MS&EStanford University
3. SubjectivityWord Sense Disambiguation - CemAkkaya and JanyceWiebe University of Pittsburgh, RadaMihalceaUniversity of North Texas
4. A New Approach to Design Domain Specific Ontology Based Web Crawler - DebajyotiMukhopadhyay, Arup Biswas, Sukanta Sinha
5. Scheduling Algorithms forWeb Crawling - Carlos CastilloCenter for Web ResearchUniversidad de Chile, Mauricio MarinCenter for Web ResearchUniversidad de Magallanes, Andrea RodriguezCenter for Web ResearchUniversidad de Concepci´on
6. Downloading Hidden Web Content - AlexandrosNtoulas, PetrosZerfos, Junghoo ChoUCLA Computer Science
7. A Scalable Ranking Method for Semantic Web Data with Context - Aidan Hogan, Andreas Harth, and Stefan Decker
8. Machine Learning in Automated Text Categorization - FABRIZIO SEBASTIANI
9. Context based Re-ranking of Web Documents (CReWD) - Arijit Banerjee, JagadishVenkatraman Graduate Students, Department of Computer Science, Stanford University
10. Text Categorization with Support Vector Machines: Learning with Many Relevant Features – Thorsten Joachims