

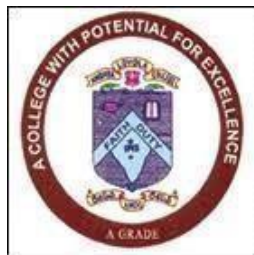
A PROJECT REPORT
ON
EFFICIENT VERTICAL MINING OF HIGH AVERAGE UTILITY
ITEMSETS OF NOVEL UPPER BOUND

**Submitted to Krishna University, Machilipatnam in partial
fulfilled of the requirements for the award of the degree**

Of
MASTER OF COMPUTER APPLICATIONS

By
P. MONIKA SAI (Y18MCA035)

UNDER ESTEEMED GUIDANCE OF
Mr. ANANDA BABU.R, M.Sc. (IS), M. Tech (CSE)
DEPARTMENT OF COMPUTER APPLICATIONS
ANDHRA LOYOLA COLLEGE



“A College with potential for Excellence”-UGC

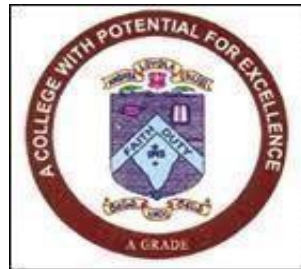
Re-accredited at 'A++' Grade-NAAC

Affiliated to Krishna University,

Machilipatnam, Krishna district-520008

2019-2021

ANDHRA LOYOLA COLLEGE
DEPARTMENT OF
MASTER OF COMPUTER APPLICATIONS



CERTIFICATE

This is to certify that the project work entitled, "**Efficient vertical mining of high average utility item sets of novel upper bound**" submitted by **P. MONIKA SAI** bearing **Regd. No: Y18MCA035** partial fulfilment of the requirements for the award of the degree of **MASTER OF COMPUTER APPLICATIONS** from Krishna University, is a benefited work to the best of my knowledge and may be placed before the examination board for their consideration.

LECTURER INCHARGE

Mr. R. Ananda Babu

HEAD OF THE DEPARTMENT

Mrs. Dr. R. P. L. D.B. Poonam

EXTERNAL EXAMINER

CERTIFICATE

This is to certify that Miss PALLANTI MONIKA SAI (Regd. No: Y18MCA035), student of ANDHRA LOYOLA COLLEGE, VIJAYAWADA, affiliated to KRISHNA UNIVERSITY has Successfully completed the project work titled "EFFICIENT VERTICAL MINING OF HIGH AVERAGE UTILITY ITEMSETS OF NOVEL UPPER BOUND" as part of her course curriculum in our organization.

She has done her project using PYTHON Programming during the period February 2021 to June 2021. She has completed the assigned project well within the time. She is sincere, hardworking and her conduct during the project is commendable.

We wish her All the Best in her future endeavor.

A handwritten signature in blue ink is written over a circular purple stamp. The stamp contains the word "zessta" at the top and "HR Manager" at the bottom. Below the signature, the text "(HR Manager)" is printed in black.

(HR Manager)

ACKNOWLEDGEMENT

It has been a great pleasure doing project work at Andhra Loyola College. I express profound gratitude to Rev. Fr. S. Raju. S. J M. A (English), M. Phil, Vice Principal (PG), Andhra Loyola College, Vijayawada, who gave me an opportunity to work out my project. I wish to express my special thanks to **Mrs. R. Ananda Babu M.Sc. (IS), M. Tech (CSE)** lecturer of MCA, ALC Vijayawada. For her constant encouragement throughout this course.

P. MONIKA SAI

Regd. No: Y18MCA035

DECLARATION

This is to declare that the dissertation / project report entitled "**Efficient vertical mining of high average utility item sets of novel upper bound**" done by me in is an authentic work carried out for the partial fulfilment of the requirements for the award of the degree of MCA in computer applications under the guidance of **Mr. R. Ananda Babu** M.Sc. (IS), M. Tech (CSE) lecture in the Department of MCA. The matter embodied in this project work has not been submitted earlier for award of ant degree or diploma to the best of my knowledge and belief.

Signature of the student

P. MONIKA SAI

Regd. No: Y18MCA035

Andhra Loyola College

ABSTRACT

Utility mining has recently been discussed in the field of data mining. A utility item set considers both profits and quantities of items in transactions, and thus its utility value increases with increasing item set length. To reveal a better utility effect, an average-utility measure, which is the total utility of an item set divided by its item set length, is proposed. However, existing approaches use the traditional average-utility upper-bound model to find high average-utility item sets, and thus generate a large number of unpromising candidates in the mining process. The present study proposes an improved upper-bound approach that uses the prefix concept to create tighter upper bounds of average-utility values for item sets, thus reducing the number of unpromising item sets for mining. Results from experiments on two real databases show that the proposed algorithm outperforms other mining algorithms under various parameter settings.

Keywords:

- Data Mining
- Average Utility Mining
- High Average Utility Item sets
- Upper bound strategy

Organization Profile

It has high energy IT Specialist Company offering solutions in AI, Machine Learning, Computer vision, and all the related technologies. Established in 2011, our expertise lies in devising holistic and end-to-end solutions in contemporary technologies to help businesses bring their ideas into reality.

From our inception we have been fortunate to have partnered with pioneering start-ups, leading enterprises and some of the world's largest technology brands. Over the past six years we have been able to establish a strong foothold in the Smart solutions space and have devised best in-class solutions for different sectors including healthcare, entertainment, life-sciences, banking and financial services. Our visionary mobile solutions have helped many organizations to accelerate their enterprise mobility journey.

INDEX

	CONTENTS	PAGE NO
1	INTRODUCTION	01
1.1	Introduction	02
1.2	Description of the Project	02
1.3	Need for software	03
1.3.1	Existing System	03
1.3.2	Proposed System	04
1.4	Goals and Objectives	05
1.5	Environment Involved in the Project	05
1.6	Methodology Involved in the Project	17
2	STUDY PHASE	22
2.1	Introduction to Analysis	23
2.2	Data Collection	26
2.3	Data Analysis	27
2.3.1	Identifying Users and Actors	28
2.3.2	Use Case Diagrams	30
2.3.3	Activity Diagrams	31
2.3.4	Sequence Diagrams	32
2.3.5	Identifying the Classes	33
2.3.6	Identifying the Relationships	39
2.3.7	Identifying the Methods	40
2.3.8	Identifying the Attributes	41
2.3.9	Business Class Diagram	41

3	DESIGN PHASE	42
3.1	Design Process	43
3.2	Design Axioms	46
3.3	Refining Attributes	46
3.4	Refining Methods	47
3.5	Refining class diagrams	48
3.6	Coupling and Cohesion	48
3.7	Design of Access Layer	49
3.7.1	Designing Access Layer Classes	49
3.7.2	Listing of Methods of Access Layer	49
3.8	Design of View Layer	50
3.8.1	Designing of View Layer Classes	50
3.8.2	View Layer	50
3.8.3	Designing Interface Behavior	51
4	DEVELOPMENT PHASE	52
4.1	Frontend Features	53
4.1.1	Physical Specification of Attributes	53
4.1.2	Special Features of Language	59
4.2	Backend Features	60
4.3	Test Cases	62
4.4	Sample Screens	65
4.5	How our program is best	79
4.6	Future Extension	82
5	CONCLUSION	83
5.1	Conclusion	84
5.2	Bibliography	85

1 - INTRODUCTION

INTRODUCTION

1.1. Introduction

The average utility measure is adopted in this paper to reveal a better utility effect of combining several items than the original utility measure. A mining algorithm is then proposed to efficiently find the high average-utility item sets. It uses the summation of the maximal utility among the items in each transaction including the target item set as the upper bounds to overestimate the actual average utilities of the item set and processes it in two phases. As expected, the mined high average-utility item sets in the proposed way will be fewer than the high utility item set under the same threshold. Experiments results also show the performance of the proposed algorithm.

1.2. Description of the project

Mining High Average-Utility Item sets (HAUIs) in a quantitative database is an extension of the traditional problem of frequent item set mining, having several practical applications. Discovering HAUIs is more challenging than mining frequent item sets using the traditional support model since the average-utilities of item sets do not satisfy the downward-closure property. To design algorithms for mining HAUIs that reduce the search space of itemsets, prior studies have proposed various upper-bounds on the average-utilities of item sets. However, these algorithms can generate a huge amount of unpromising HAUI candidates, which result in high memory consumption and long runtimes. To address this problem, this paper proposes four tight average-utility upper-bounds, based on a vertical database representation, and three efficient pruning strategies. Furthermore, a novel generic framework for comparing average-utility upper-bounds is presented. Based on these theoretical results, an efficient algorithm named dHAUIM is introduced for mining the complete set of HAUIs. dHAUIM represents the search space and quickly compute upper-bounds using a novel IDUL structure. Extensive experiments show that dHAUIM outperforms three state-of-the-art algorithms for mining HAUIs in terms of runtime on both real-life and synthetic databases. Moreover, results show that the proposed pruning strategies dramatically reduce the number of candidate HAUIs.

1.3. Need for software

1.3.1. Existing System

An item set A is a subset of P and is called k -item set if it contains k items, i.e. $k=|A|$. Without loss of generality, we can assume that all items in each item set are sorted in ascending order with respect to the lexicographical order relation $<$. Moreover, each item a is associated with a positive number (a) called its unit profit (or external utility), which represents its relative importance to the user in terms of criteria such as unit profit, price, weight or importance. The profit vector $P \stackrel{\text{def}}{=} ((aj), \square J)$ consists of the unit profits of all items in P . A q -item is a pair (a, q) , where $a \in P$ and q is a non-negative number indicating the purchase quantity (or internal utility) of the item a . A transaction or q -itemset is a set of q -items, $ti \stackrel{\text{def}}{=} \{(aj, q_{ij}), j \in J\}$, where $J \subseteq I$ is an index subset of I . A quantitative database (QDB) \mathcal{D} is a finite set of transactions, $\mathcal{D} \stackrel{\text{def}}{=} \{ti \mid ti \stackrel{\text{def}}{=} \{(aj, q_{ij}), j \in J\}, J \subseteq I, i \in I\}$, where each transaction ti is associated with a unique identifier TID (e.g. $TID = i$). The utility of a q -item (a, q) is denoted and defined as $u((a, q)) \stackrel{\text{def}}{=} q * p(a)$. Intuitively, it represents the amount of profit yield by the sale of item a in the transaction where the q -item (a, q) appears.

- a. When mining a quantitative database to find high utility patterns, to avoid repeatedly computing the utility $u((aj, q_{ij})) = q_{ij} * p(aj)$ of a q -item (aj, q_{ij}) in a transaction ti , the utility of each q -item can be calculated once and stored in the corresponding transaction. The transformed database resulting from applying this process to a quantitative database QDB $\mathcal{D} \stackrel{\text{def}}{=} \{ti \stackrel{\text{def}}{=} \{(aj, q'_{ij}), j \in J\}, J \subseteq I, i \in I\}$ is called an integrated quantitative matrix (or briefly integrated matrix), where $N \times M \stackrel{\text{def}}{=} \{q'_{ij}, i \in I, j \in J\}$, and q'_{ij} is defined as follows, $q'_{ij} \stackrel{\text{def}}{=} \{q_{ij} * p(aj), \text{if } (aj, q_{ij}) \in ti\}$, otherwise $0, \forall i \in I, j \in J$.
- b. Furthermore, consider an operator $\square : 2^P \rightarrow 2^{\mathcal{D}}$ defined as follows: $\square P: A \subseteq P, \square (A) \stackrel{\text{def}}{=} \{ti \in \mathcal{D} \mid q'_{ij} > 0, \square aj \in A\}$ and as convention \mathcal{D} . Intuitively, this operator maps each nonempty item set A to transactions where all items in A have purchase quantities greater than zero. Hereafter, we only consider item sets in the set $\mathcal{I} \stackrel{\text{def}}{=} \{A \subseteq P \mid \square(A) \neq \emptyset\}$, that is item sets appearing in at least one transaction.

- c. Let mu be a minimum average-utility threshold defined by the user (a positive number). An item set A in \mathcal{IS} is called high average-utility (HAU) iff $au(A) \geq mu$. Otherwise, it is called unpromising or low average utility (LAU). The problem of high average-utility item set mining is to find the set of all HAUIs, (mu) or briefly $HAUS \stackrel{\text{def}}{=} \{A \in \mathcal{IS} \mid (A) \geq mu\}$. Note that the minimum threshold mu can be also defined as $mu \stackrel{\text{def}}{=} \delta * TU$, where δ is a minimum percentage threshold set by the user.

1.3.2 Proposed System

In this section, we present new theoretical results for the design of an efficient algorithm for mining high average-utility item sets. In particular, this section explains how the utility measure u and novel upper-bounds on the average-utility measure au can be defined using a vertical form (columns of the matrix), contrarily to previous studies that have used the horizontal form. These results are the basis of the proposed algorithm. Let $(A) \Sigma q'ijti \square \square (A)$ be the column utility of an item set A in the j th column of the matrix. Furthermore, let $\mathcal{V}(A) \stackrel{\text{def}}{=} (vj(A), j \geq \minInd(A))$ be the set of all column utility values associated to A starting from the $\minInd(A)$ column, where $\minInd(A) = \min\{j \in J \mid aj \in A\}$ is the minimum index (or first index) of items aj in A . As a convention, based on these definitions, the utility of A is defined in vertical form as follows. During the expansion of the search space of HAU candidate item sets, an item set P is extended with an item set B such that $P < B$ (i.e. $\forall a \in P, \forall b \in B, a < b$). The result is an extension C (also called a forward extension) of P (with item set B), denoted as $C = P \oplus B$, i.e. $C = PUB$ under the condition that $P < B$. In this situation, the item set P is said to be a prefix of C . For example, if $B = \{b\}$ and $P < \{b\}$, an item-extension of P (with item b) is $P \oplus \{b\}$, concisely denoted as Pb . Note that the au measure is neither monotonic nor anti-monotonic. Indeed, for example, $a \square ab \square abc$, but $(a) = 90 < (abc) = 290/3 < au(ab) = 304$. To design pruning strategies for reducing the search space in HAUIM, an UB named $auub$ has been proposed [14]. The $auub$ measure is an upper bound on au , which satisfies the DCP (or anti-monotonicity property), i.e. $\forall A \subseteq B \subseteq P$, it follows that $(A) \leq auub(A)$ and $auub(B) \leq auub(A)$. Therefore, if A is low- $auub$, i.e. $(A) < mu$, then all its super item sets B are also low- $auub$ and LAU. To prune more unpromising candidates from the search space, several upper-bounds on the au measure have been proposed. However, the set of candidate item sets considered when using these upper-bounds is often very large. To overcome this problem, this article

proposes four new UBs that are tighter than *au ub*, named *au b1*, *aub*, *iaub* and *laub*, which can be quickly calculated using the proposed vertical form (*A*).

1.4. Goals and Objectives

- a. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.
- b. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.
- c. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

1.5. Environment involved in the project

PYTHON

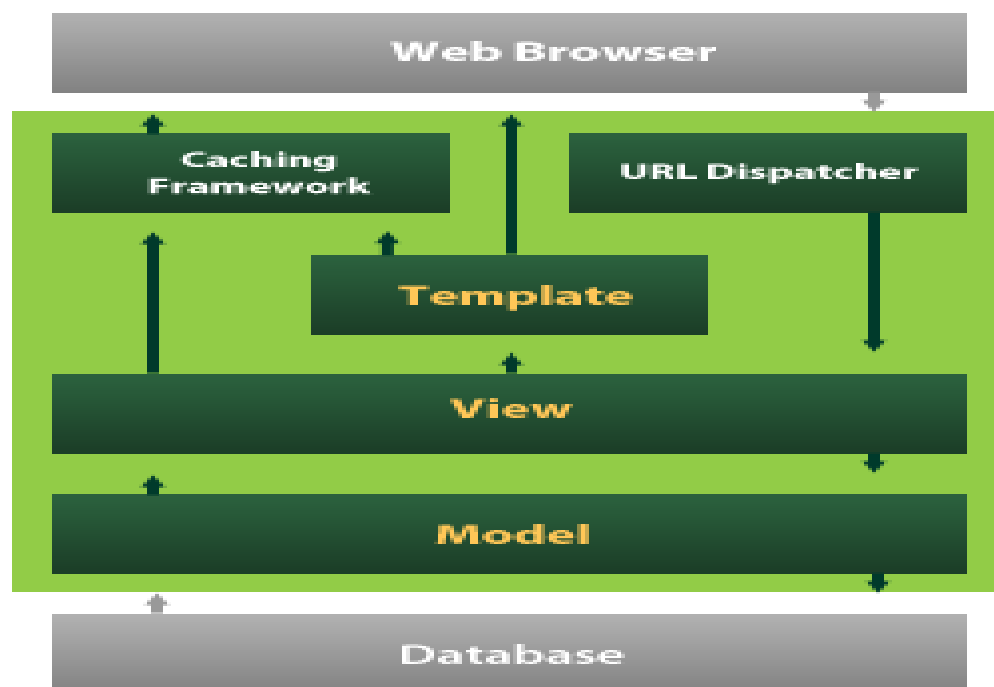
Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](#), Python has a design philosophy that emphasizes [code readability](#) (notably using [whitespace](#) indentation to delimit [code blocks](#) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](#) than might be used in languages such as [C++](#) or [Java](#). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](#).

[C Python](#), the [reference implementation](#) of Python, is [open source](#) software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit [Python Software Foundation](#). Python features a [dynamic type](#) system and automatic [memory management](#). It supports multiple [programming paradigms](#), including [object-oriented](#), [imperative](#), [functional](#) and [procedural](#), and has a large and comprehensive [standard library](#)

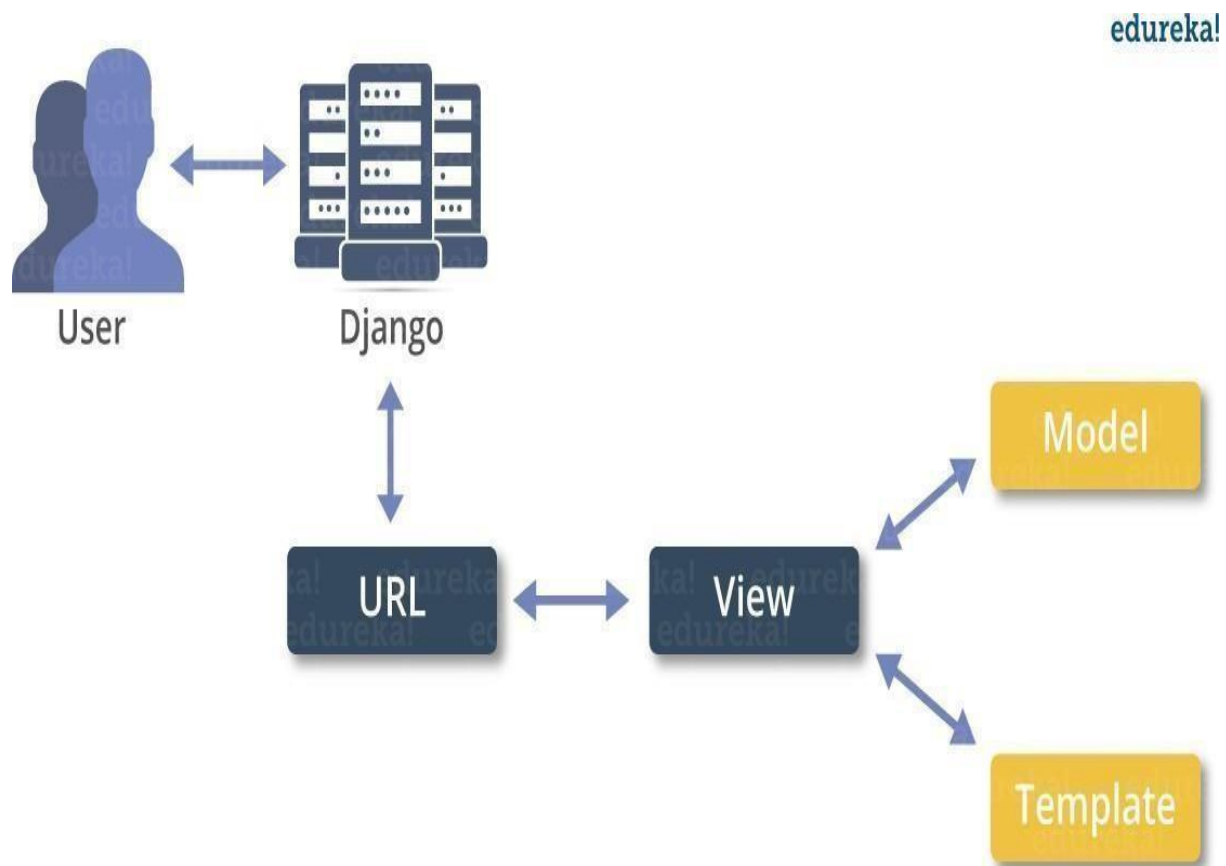
DJANGO

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes [reusability](#) and "plug ability" of components, rapid development, and the principle of [don't repeat yourself](#). Python is used throughout, even for settings files and data models.



Django also provides an optional administrative [create](#), [read](#), [update](#) and [delete](#) interface that is generated dynamically through [introspection](#) and configured via admin models



Python is a programming language, which means it's a language both people and computers can understand. Python was developed by a Dutch software engineer named Guido van Rossum, who created the language to solve some problems he saw in computer languages of the time.

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, and a syntax that allows programmers to express concepts in fewer lines of code, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems. C Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit Python Software Foundation.

You Can Use Python for Pretty Much Anything

One significant advantage of learning Python is that it's a general-purpose language that can be applied in a large variety of projects. Below are just some of the most common fields where Python has found its use:

- Data science
- Scientific and mathematical computing
- Web development
- Computer graphics
- Basic game development
- Mapping and geography (GIS software)

Python Is Widely Used in Data Science

- Python's ecosystem is growing over the years and it's more and more capable of the statistical analysis.
- It's the best compromise between scale and sophistication (in terms of data processing).
- Python emphasizes productivity and readability.
- Python is used by programmers that want to delve into data analysis or apply statistical techniques (and by those that turn to data science)
- There are plenty of Python scientific packages for data visualization, machine learning, natural language processing, complex data analysis and more. All of these factors make

Python a great tool for scientific computing and a solid alternative for commercial packages such as MatLab. The most popular libraries and tools for data science are:

Pandas

A library for data manipulation and analysis. The library provides data structures and operations for manipulating numerical tables and time series.

NumPy

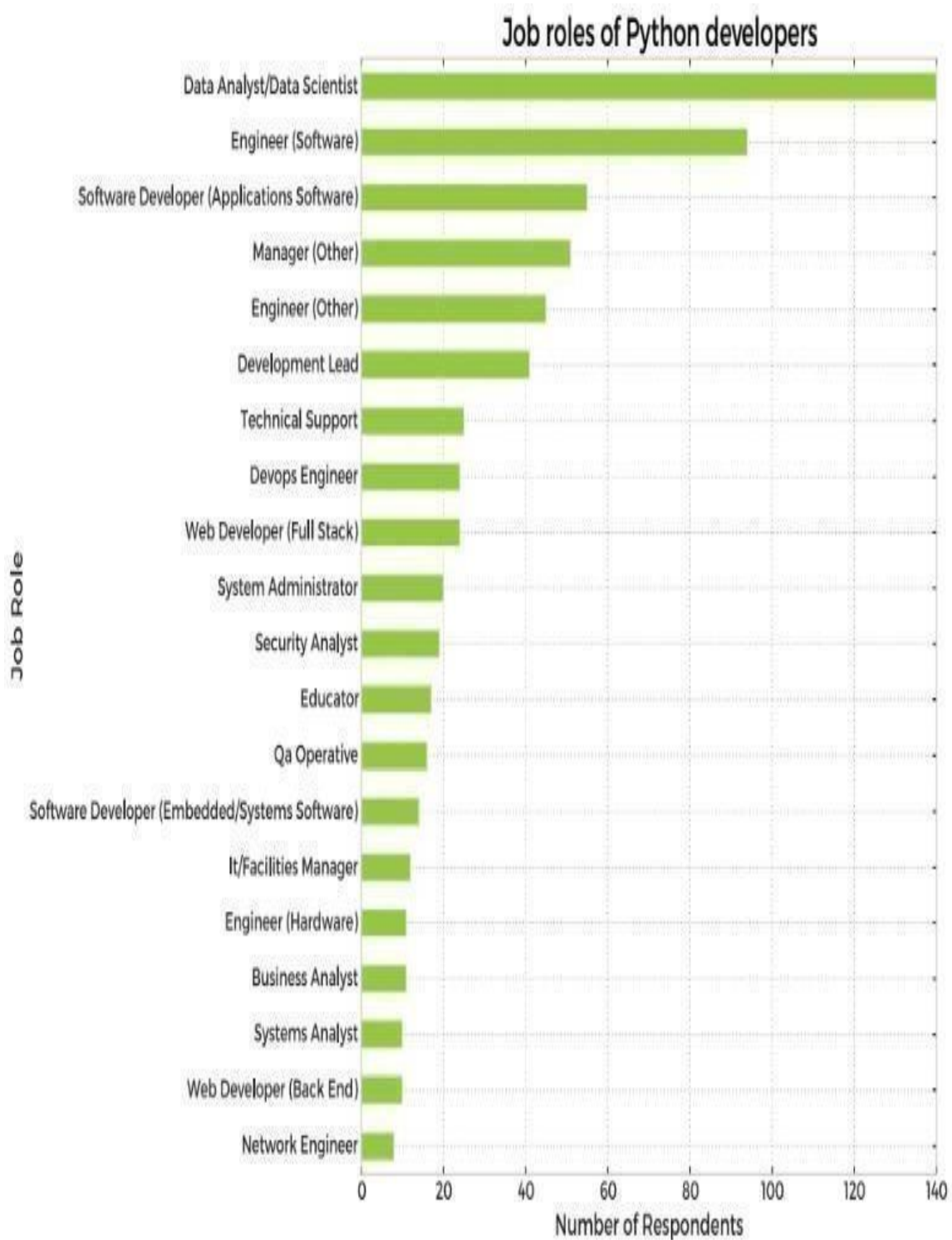
The fundamental package for scientific computing with Python, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

SciPy

A library used by scientists, analysts, and engineers doing scientific computing and technical computing.

Being a free, cross-platform, general-purpose and high-level programming language, Python has been widely adopted by the scientific community. Scientists value Python for its precise and efficient syntax, relatively flat learning curve and the fact that it integrates well with other languages (e.g. C/C++).

As a result of this popularity there are plenty of Python scientific packages for data visualization, machine learning, natural language processing, complex data analysis and more. All of these factors make Python a great tool for scientific computing and a solid alternative for commercial packages such as MatLab.



Here's our list of the most popular Python scientific libraries and tools

Astropy

The Astropy Project is a collection of packages designed for use in astronomy. The core astropy package contains functionality aimed at professional astronomers and astrophysicists, but may be useful to anyone developing astronomy software.

Biopython

Biopython is a collection of non-commercial Python tools for computational biology and bioinformatics. It contains classes to represent biological sequences and sequence annotations, and it is able to read and write to a variety of file formats.

Cubes

Cubes is a light-weight Python framework and set of tools for the development of reporting and analytical applications, Online Analytical Processing (OLAP), multidimensional analysis and browsing of aggregated data.

DEAP

DEAP is an evolutionary computation framework for rapid prototyping and testing of ideas. It incorporates the data structures and tools required to implement most common evolutionary computation techniques such as genetic algorithm, genetic programming, evolution strategies, particle swarm optimization, differential evolution and estimation of distribution algorithm.

SCOOP

SCOOP is a Python module for distributing concurrent parallel tasks on various environments, from heterogeneous grids of workstations to supercomputers.

PsychoPy

PsychoPy is a package for the generation of experiments for neuroscience and experimental psychology. PsychoPy is designed to allow the presentation of stimuli and collection of data for a wide range of neuroscience, psychology and psychophysics experiments.

Pandas

Pandas is a library for data manipulation and analysis. The library provides data structures and operations for manipulating numerical tables and time series.

Mlpy

Mlpy is a machine learning library built on top of NumPy/SciPy, the GNU Scientific Libraries. Mlpy provides a wide range of machine learning methods for supervised and unsupervised problems and it is aimed at finding a reasonable compromise between modularity, maintainability, reproducibility, usability and efficiency.

Matplotlib

Matplotlib is a python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib allows you to generate plots, histograms, power spectra, bar charts, error charts, scatterplots, and more.

NumPy

NumPy is the fundamental package for scientific computing with Python, adding support for large, multi-dimensional arrays and matrices, along with a large library of high-level mathematical functions to operate on these arrays.

NetworkX

NetworkX is a library for studying graphs which helps you create, manipulate, and study the structure, dynamics, and functions of complex networks.

TomoPy

TomoPy is an open-sourced Python toolbox to perform tomographic data processing and image reconstruction tasks. TomoPy provides a collaborative framework for the analysis of synchrotron tomographic data with the goal to unify the effort of different facilities and beamlines performing similar tasks.

Theano

Theano is a numerical computation Python library. Theano allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently.

SymPy

SymPy is a library for symbolic computation and includes features ranging from basic symbolic arithmetic to calculus, algebra, discrete mathematics and quantum physics. It provides computer algebra capabilities either as a standalone application, as a library to other applications, or live on the web.

SciPy

SciPy is a library used by scientists, analysts, and engineers doing scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering.

Scikit-learn

Scikit-learn is a machine learning library. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

Scikit-image

Scikit-image is a image processing library. It includes algorithms for segmentation, geometric transformations, color space manipulation, analysis, filtering, morphology, feature detection, and more.

ScientificPython

ScientificPython is a collection of modules for scientific computing. It contains support for geometry, mathematical functions, statistics, physical units, IO, visualization, and parallelization.

SageMath

SageMath is mathematical software with features covering many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus. SageMath uses the Python, supporting procedural, functional and object-oriented constructs.

Veusz

Veusz is a scientific plotting and graphing package designed to produce publication-quality plots in popular vector formats, including PDF, PostScript and SVG.

Graph-tool

Graph-tool is a module for the manipulation and statistical analysis of graphs.

SunPy

SunPy is a data-analysis environment specializing in providing the software necessary to analyze solar and heliospheric data in Python.

Bokeh

Bokeh is a Python interactive visualization library that targets modern web browsers for presentation. Bokeh can help anyone who would like to quickly and easily create interactive plots, dashboards, and data applications. Its goal is to provide elegant, concise construction of novel graphics in the style of D3.js, but also deliver this capability with high-performance interactivity over very large or streaming datasets.

TensorFlow

TensorFlow is an open source software library for machine learning across a range of tasks, developed by Google to meet their needs for systems capable of building and training neural networks to detect and decipher patterns and correlations, analogous to the learning and reasoning which humans use. It is currently used for both research and production at Google products, often replacing the role of its closed-source predecessor, DistBelief.

Nilearn

Nilearn is a Python module for fast and easy statistical learning on Neuro Imaging data. Nilearn makes it easy to use many advanced machine learning, pattern recognition and multivariate statistical techniques on neuroimaging data for applications such as MVPA (Mutli-Voxel Pattern Analysis), decoding, predictive modelling, functional connectivity, brain parcellations, and connectomes.

Dmelt

DataMelt, or DMelt, is a software for numeric computation, statistics, analysis of large data volumes ("big data") and scientific visualization. The program can be used in many areas, such as natural sciences, engineering, modeling and analysis of financial markets. DMelt can be used with several scripting languages including Python/Jython, BeanShell, Groovy, Ruby, as well as with Java.

Python-weka-wrapper

Weka is a suite of machine learning software written in Java, developed at the University of Waikato, New Zealand. It contains a collection of visualization tools and algorithms for data analysis and predictive modeling, together with graphical user interfaces for easy access to these functions. The python-weka-wrapper package makes it easy to run Weka algorithms and filters from within Python.

Dask

Dask is a flexible parallel computing library for analytic computing composed of two components: 1) dynamic task scheduling optimized for computation, optimized for interactive computational workloads, and 2) Big Data collections like parallel arrays, dataframes, and lists that extend common interfaces like NumPy, Pandas, or Python iterators to larger-than-memory or distributed environments.

Python Saves Time

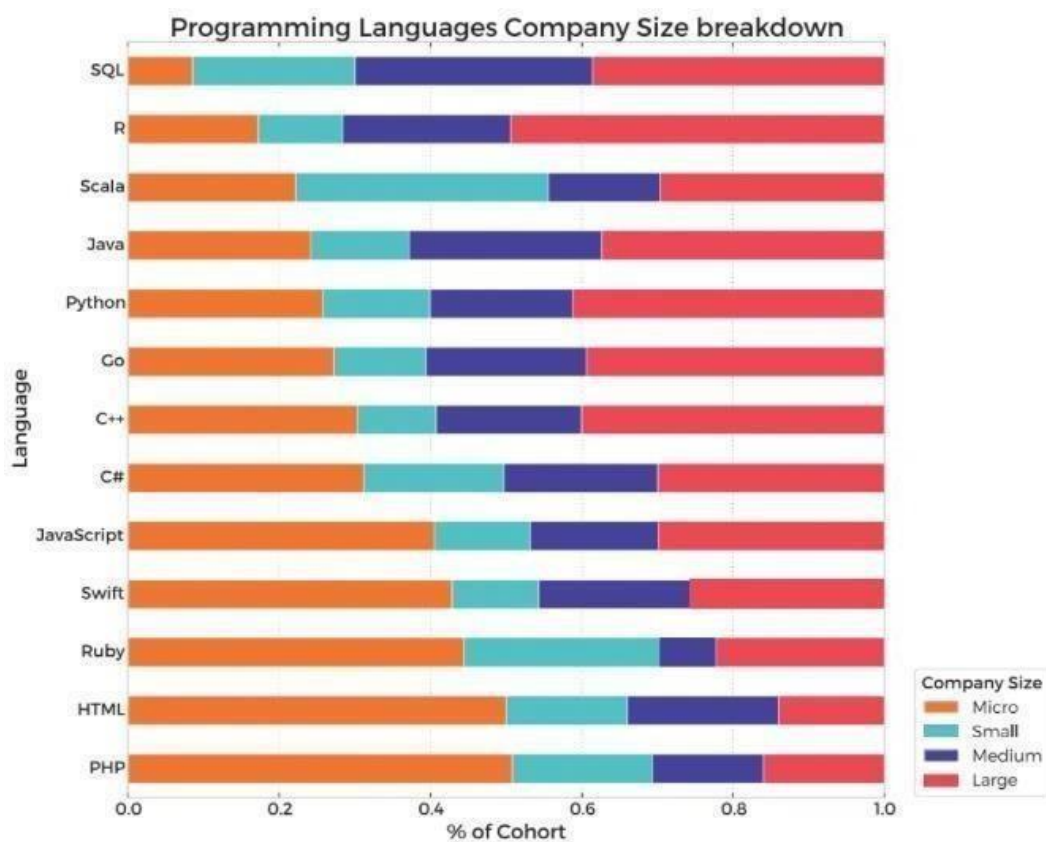
Even the classic “Hello, world” program illustrates this point:

```
print("Hello, world")
```

For comparison, this is what the same program looks like in Java:

```
public class HelloWorld {  
  
    public static void main(String[] args) {  
  
        System.out.println("Hello, world");  
  
    }  
  
}
```

All the Big Names Use Python



Python

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems. C Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit Python Software Foundation.



Python Logo

1.6. Methodology involved in the project

○ **Four novel UBs**

In this subsection, to answer the question (Q_1), we propose four novel UBs named aub_1 , aub , $iaub$ and $laub$ that are gradually tighter than the traditional UB $auub$ on au , have gradually weaker anti-monotone-like properties and pruning effects as shown below. Firstly, we define two new UBs gradually tighter than $auub$ named aub_1 and aub , which satisfy two stronger anti-monotone-like properties, and have respectively strong width and width pruning effects as it will be explained in next sections.

○ Generic framework for evaluating UBs

(Anti-monotone-like (PMf) criteria of UBs). Let C, D, E, P, R be item sets in fS and ub be an UB on the average utility, i.e. $au(P) \leq ub(P), \forall P \in fS$. Then ub is said to satisfy

- The anti-monotonicity property (or criterion), denoted as (ub) , iff $ub(C) \leq ub(P), \forall C \sqsubseteq P$.
- The anti-monotonicity property w.r.t. bi-directional extension, denoted as $Bi\mathcal{E}PM(ub)$, iff $ub(C) \leq ub(P), \forall P = R \oplus D, C = R \oplus E$ such that $R \neq \square$ and $D \subseteq E$.
- The anti-monotonicity property w.r.t. forward extension, denoted as $\mathcal{F}EPM(ub)$, iff $u(C) \leq ub(P), \forall P, C = P \oplus E$ such that $P \neq \square$ (i.e. C is a forward extension of the non-empty prefix P).
- The depth pruning condition by forward extension, denoted as $P\mathcal{J}(ub)$, if $au(C) \leq ub(P), \forall P, C = P \oplus E$ such that $P \neq \square$.

It is clear that if (ub) , then ub satisfies the down-ward closure property. This means that if P is low-ub.

○ Three novel pruning strategies based on the proposed UBs

Although aub_1, aub and $iaub$ are respectively gradually tighter UBs on au , and $HAUS \sqsubseteq HIAUB \sqsubseteq HAUB \subseteq HAUB_1$, each UB has its own pruning ability or effect. For any UB ub , we say that the pruning condition $PC_{ub}(P)$ for a non-empty item set P holds iff $(P) < mu$ (or P is low-ub). For the sake of brevity, we denote the set of P and all item set extensions of P as $branch(P)$. Furthermore, the notation $[P] \stackrel{\text{def}}{=} \{Px, \dots, Py, \dots, Pz, \dots\}$ will denote the equivalence class consisting of all item-extensions of P (item sets having the same prefix P). Based on these concepts, the search space of HAUIM can be viewed as a prefix-tree, where each node represents an item set, such that the root is the empty set and each child of a node is single item-extension of that item set. These definitions will be used in the proposed dHAUIM algorithm, presented in Section 4.

Pruning Strategy 1

(Depth Pruning (DP) strategy w.r.t. forward extensions based on $iaub$ and lau). For any non-empty item set P , if $PCiau(P)$ or $PClaub(P)$ hold, the whole $branch(P)$ of the search space can be pruned.

Since $iaub$ and $laub$ are incomparable, both of these UBs should be used to eliminate unpromising candidate branches of the prefix-tree.

Pruning Strategy 2

(Width Pruning (WP) strategy w.r.t. bi-directional extensions on projected databases based on aub). For the second UB au , which is larger than $iaub$, if $PCaub(Ry)$ holds for a non-empty item set R and item-extension Ry in $[R]$, the item set Ry can be removed from the set $[R]$, i.e. not only the whole $branch(Ry)$ is immediately pruned, but also all branches $branch(Rxy)$ and $branch(Ryz)$ are eliminated from the search tree

(Where Rxy and Ryz are respectively the backward and forward extensions of Ry).

In other words, such item y will not appear in $branch(R)$ and we can remove y from the projected database [16] of R .

Pruning Strategy 3

(Strong Width Pruning (SWP) strategy on the initial QDB based on aub_1). For the aub_1 UB, which is the largest among the four new UBs, if $PCaub_1(P)$ holds, then $PCaub_1(C)$ also holds for all extensions C of P .

In particular, for each item a of P , if $PCaub_1(aj)$ holds, i.e. $aub_1(aj) < mu$, and we can remove aj from the database or delete the j^{th} column (according to aj) of the integrated matrix.

○ THE dHAUIM ALGORITHM

Based on the novel aub_1 , aub , $iaub$ and $laub$ UBs and the recursive formulas of Proposition 3, this section presents an efficient algorithm, named dHAUIM (**d**iffset-based **H**igh **A**verage **U**tility **I**tem set **M**iner), for mining the set of all high average-utility item sets $HAUS$

$\{(C, au(C)) \mid au(C) \geq mu\}$. The proposal of this algorithm answers the third research question (Q_3). The pseudo code of the algorithm is shown in Fig. 2. During the mining process, HAU candidate item sets are stored in a prefix-tree using a novel structure named IDUL (Item set-Diffset-Utility List). This structure contains the information of a node

C of the form $(C, d(C), \mathcal{V}(C))$. Recall that the notation $[P]$ denotes the set of item-extensions of a parent node P .

HAUS dHAUIM(, mu)

- *Input*: a QDB, the minimum AU threshold mu .
- *Output*: the set of high-average utility itemsets $HAUS$.
- 1. Create the integrated matrix $_{n \times m}$.
- 2. Scan $_{n \times m}$ once to calculate the vectors $\mathcal{V}(\square)$ and $\mathcal{V}(a_j)$ for each $a_j \in P$;
- 3. $[\square] = \{(a_j, (a_j), \mathcal{V}(a_j)) \mid a_j \in P \text{ and } \frac{au(a_j)}{1} \geq mu\}$;
//strong width pruning
- 4. $HAUS = \square$;
- 5. **HAU-Search**($[\square]$, $HAUS$);
- 6. **return** $HAUS$;

The dHAUIM algorithm

For instance, consider the integrated matrix of Table 3. For each item $a_j \in P$, we calculate $\square(a_j)$, (a_j) to obtain

$(\square = (90, 1120, 320, 28, 31))$ as shown in Table 5. For example, $\square(b) = 1356$, $v2(\square) \stackrel{\text{def}}{=} u(a_2 = b) = q'_{12} + q'_{32} +$

$$q'52 + q'62 = 160 + 400 + 240 + 320 = 1120.$$

Based on

(\square) and formulas (6)-(9), (2)-(5), we can quickly calculate

the vectors $\mathcal{V}(a_j)$ and the UB values $aub1(a_j)$, $aub(a_j) = iaub(a_j)$, $laub(a_j)$, $\forall j \in J$

HAU-Search([P], HAUS)

- *Input*: the set [P] of all item-extensions of P, the set HAUS.

- *Output*: the updated HAUS set.

```

1. if ([P]  $\neq \square$ ) then {
2.   for each ( $C_i, d(C_i), \mathcal{V}(C_i)$ ) in [P] do {
3.     if ( $\overline{au}(C_i) \geq mu$  or  $laub(C_i) \geq mu$ ) then {
                                     //depth pruning
4.       if ( $au(C_i) \geq mu$ ) then
5.         HAUS.Add( $C_i, au(C_i)$ );
6.       if ( $|[P]| > 1$ ) then {
7.         [ $C_i$ ] =  $\square$ ;
8.         for each ( $C_j, d(C_j), \mathcal{V}(C_j)$ ) in [P], with  $j > i$  do {
9.            $E = C_i \sqcup C_j$ ; ( $E$ ) =  $d(C_j) \setminus d(C_i)$ ;
10.          Calculate ( $E$ );
11.          if ( $aub(E) \geq mu$ ) then //width pruning
12.            [ $C_i$ ] = [ $C_i$ ]  $\sqcup \{(E, (E), \mathcal{V}(E))\}$ ;
13.          }
14.          HAU-Search([ $C_i$ ], HAUS);
15.        }
16.      }
17.    }
18.  }
19. return;

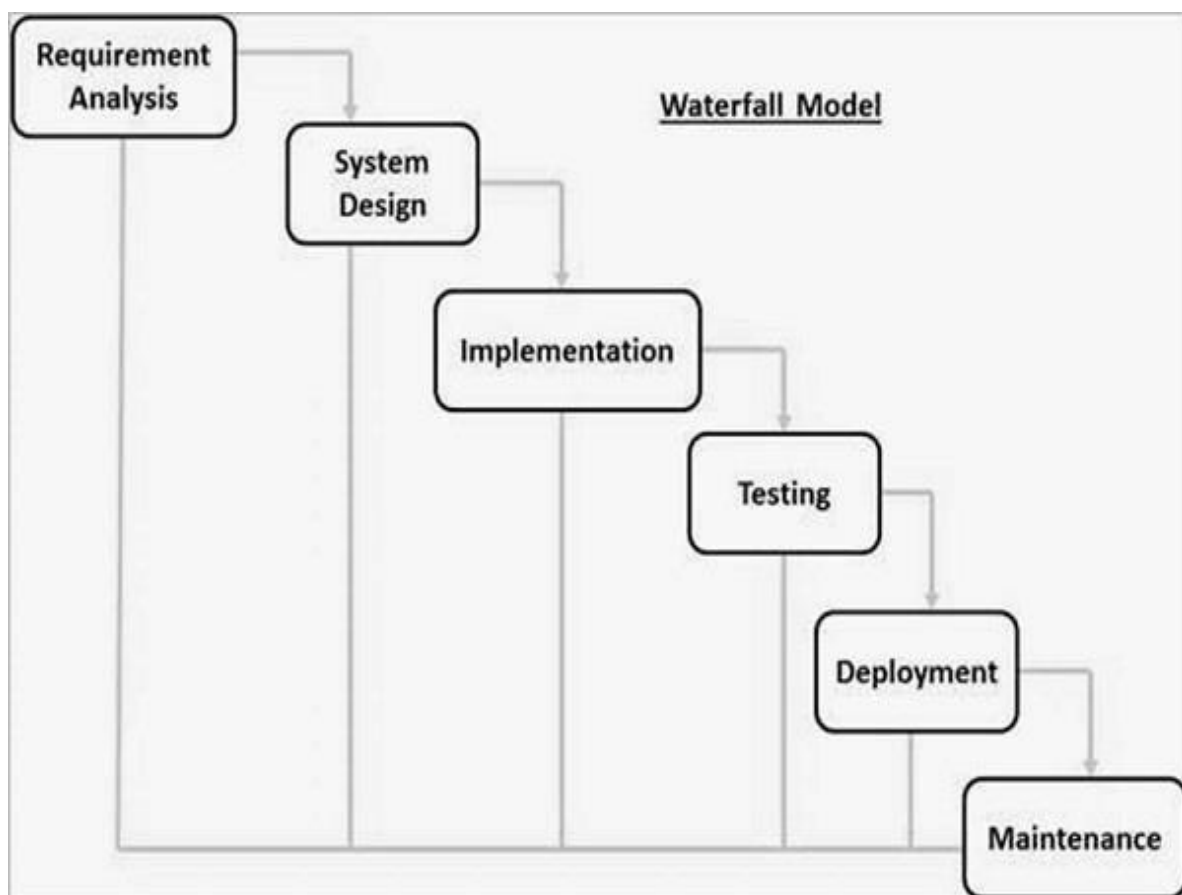
```

2 - STUDY PHASE

STUDY PHASE

2.1. Introduction to Analysis

Waterfall model is the basic **software development life cycle** model. It is very simple but idealistic. It is very important because all the other software development life cycle models are based on the classical waterfall model. Waterfall model divides the life cycle into a set of phases. This model considers that one phase can be started after completion of the previous phase. That is the output of one phase will be the input to the next phase. Thus the development process can be considered as a sequential flow in the waterfall. The different sequential phases of the classical waterfall model are shown in the below figure:



- **Requirement Gathering and analysis** – All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.
- **System Design** – The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.
- **Implementation** – With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.
- **Integration and Testing** – All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures.
- **Deployment of system** – Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.
- **Maintenance** – There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY
- **ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

- **TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

- **SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it.

2.2. Data Collection

Data may be grouped into four main types based on methods for collection: observational, experimental, simulation, and derived. The type of research data you collect may affect the way you manage that data. For example, data that is hard or impossible to replace (e.g. the recording of an event at a specific time and place) requires extra backup procedures to reduce the risk of data loss. Or, if you will need to combine data points from different sources, you will need to follow best practices to prevent data corruption.



Observational Data

Observational data are captured through observation of a behaviour or activity. It is collected using methods such as human observation, open-ended surveys, or the use of an instrument or sensor to monitor and record information -- such as the use of sensors to observe noise levels at the Mpls/St Paul airport. Because observational data are captured in real time, it would be very difficult or impossible to re-create if lost.



Experimental Data

Experimental data are collected through active intervention by the researcher to produce and measure change or to create difference when a variable is altered. Experimental data typically allows the researcher to determine a causal relationship and is typically projectable to a larger population. This type of data are often reproducible, but it often can be expensive to do so.



Simulation Data

Simulation data are generated by imitating the operation of a real-world process or system over time using computer test models. For example, to predict weather conditions, economic models, chemical reactions, or seismic activity. This method is used to try to determine what would, or could, happen under certain conditions. The test model used is often as, or even more, important than the data generated from the simulation.



Derived / Compiled Data

Derived data involves using existing data points, often from different data sources, to create new data through some sort of transformation, such as an arithmetic formula or aggregation. For example, combining area and population data from the Twin Cities metro area to create population density data. While this type of data can usually be replaced if lost, it may be very time-consuming (and possibly expensive) to do so.

2.3. Data Analysis

○ ANALYSIS OF ITEMSET

The rate of buying product by the users is noticed. The Item set can be analyzed according to the purchasing chart. The lower numbers of purchased products are so called average-utility item set. To up the utility of item set we need to do analysis of what need to do, in order to improve the sale.

- **PRODUCT RE-ARRANGE**

The average-utility item set can be improved to High Average-utility Item set. To improve this we going to handle some functional techniques. The Item set can be promoted and add or remove some features to effectively sale the product. The re-arrangement of product can be done with the high utilization of item set property. Based on this module Item set again publish to users and analysis the Item set again.

- **GRAPH ANALYSIS OF SYSTEM**

The graphs are utilized to analysis the proposed system based on the selling details. The more important part is to analysis the data which are user bought products and number that lies on the average utility. The graphs are plot to improve the way of analysis and ease of making understand the pictured data.

2.3.1. Identifying users and actors

Number of Project Server users

When you determine the number of Project Server users that your organization needs to support, also consider the maximum number of concurrent users. This is especially critical if your organization plans to support the time tracking scenario.

It is helpful to categorize users to determine the different types that you need to support, as well as how many of each type. For example, project managers who use Project Professional create the greatest load on the system; viewers create the smallest amount of load.

Types of Project Server users

The types of users that you need to support, and the percentage of each compared to the total number, affects the configuration decisions that you make during your planning process. Each user type places a load on the system. The most common user types are as follows:

- Administrator
- General users

Administrators

Administrators deploy and manage Project Server and related applications. These users manage access to the server. Administrators use Project Web App to do the following:

- Implement new activities like adding another product.
- Feedback analysis
- Create standardized analysis reports for product analysis.
- Manage multiple users.

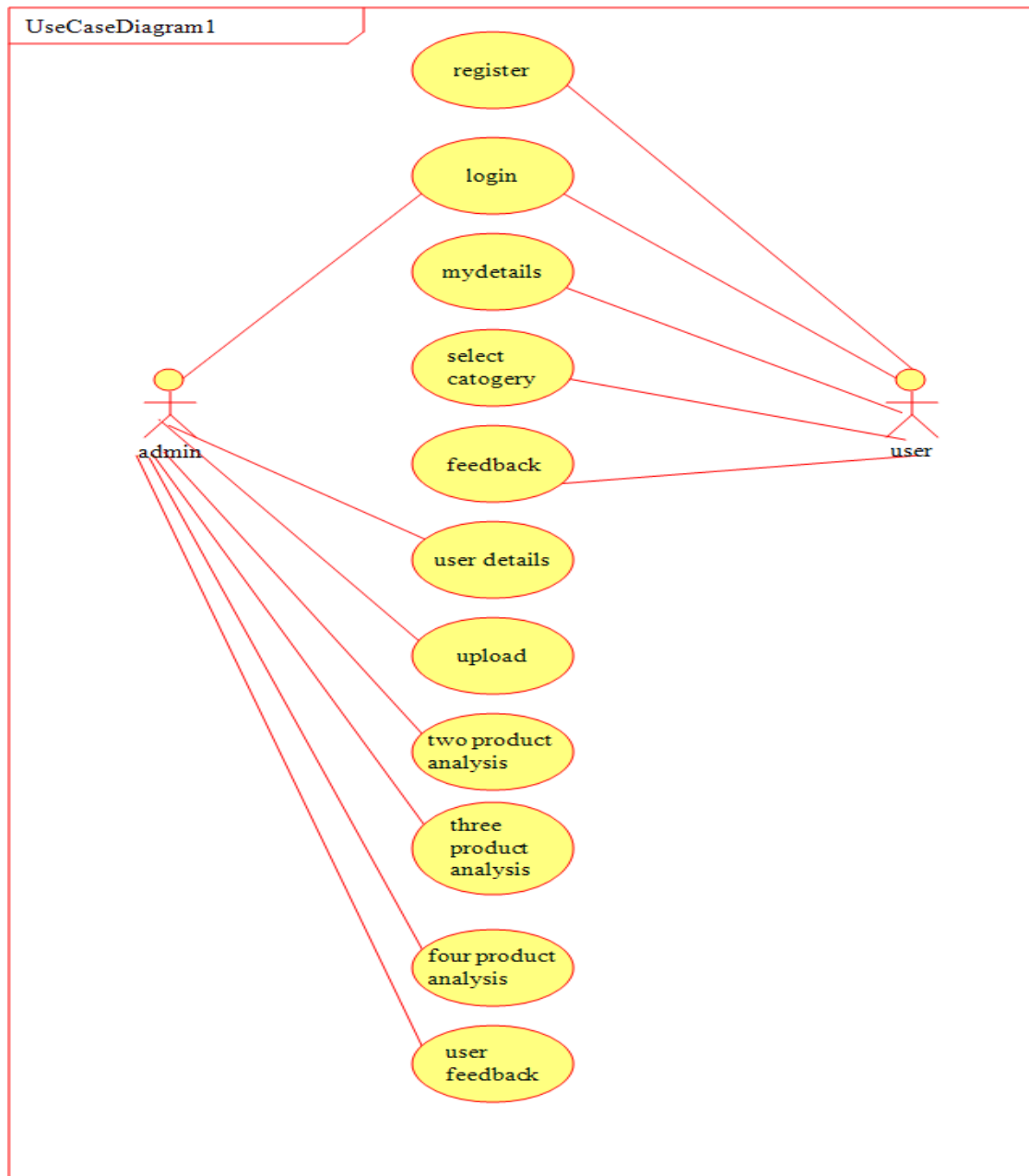
General users

Access into the web page / web app to analyze different products. The user can be able analyze two or three products at a time. The user can also give reviews / feedback to the product for latest improvement.

A user who uses Project Web App to view status or reporting on a project or multiple projects. For example, a portfolio viewer can oversee several different projects that are managed by different project managers to gain an overall perspective on schedule and budget. Portfolio viewers use Project Web App to do the following:

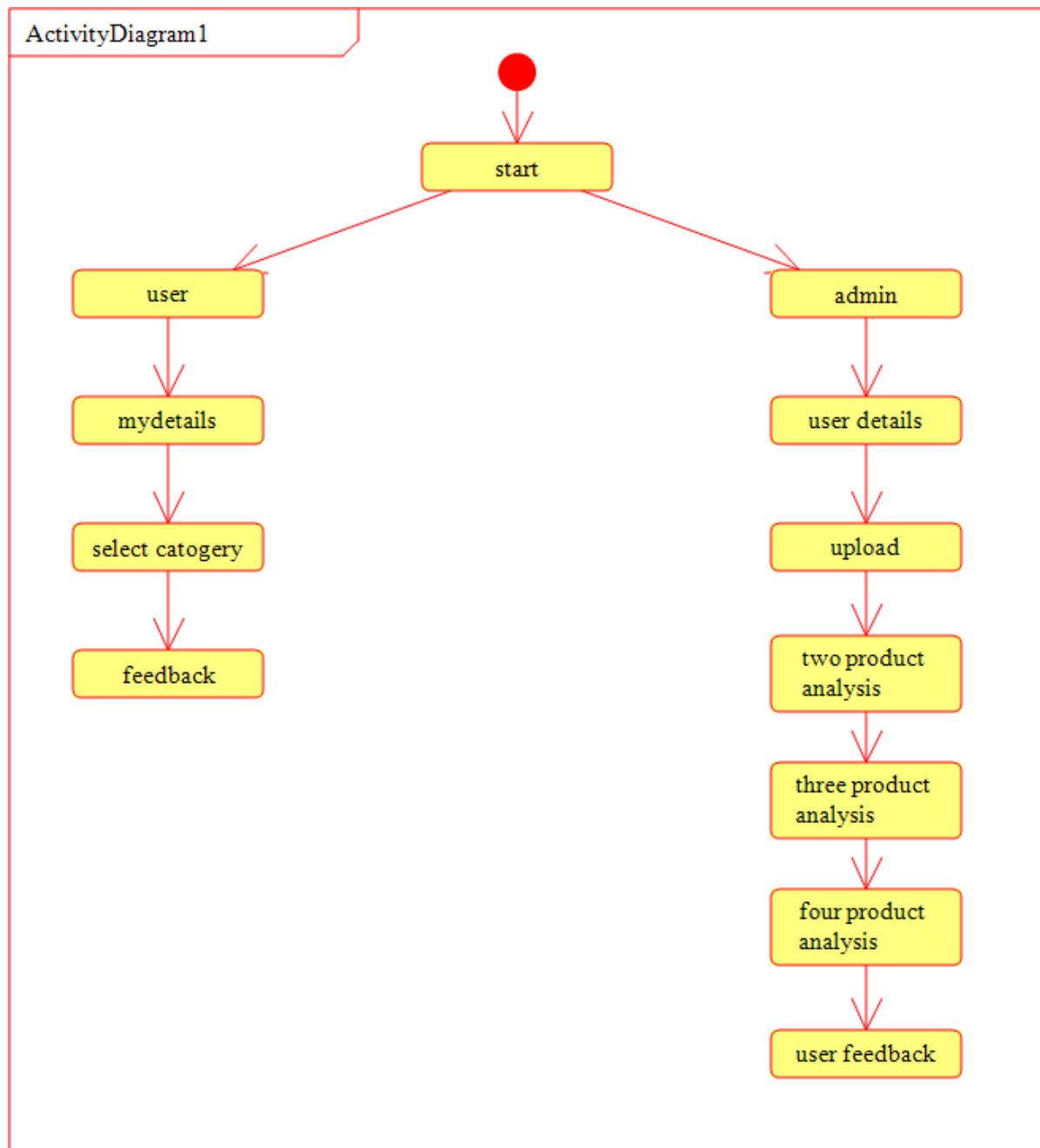
- View project and resource reports.
- Submit issues to project and resource managers.

2.3.2. Use Case Diagram



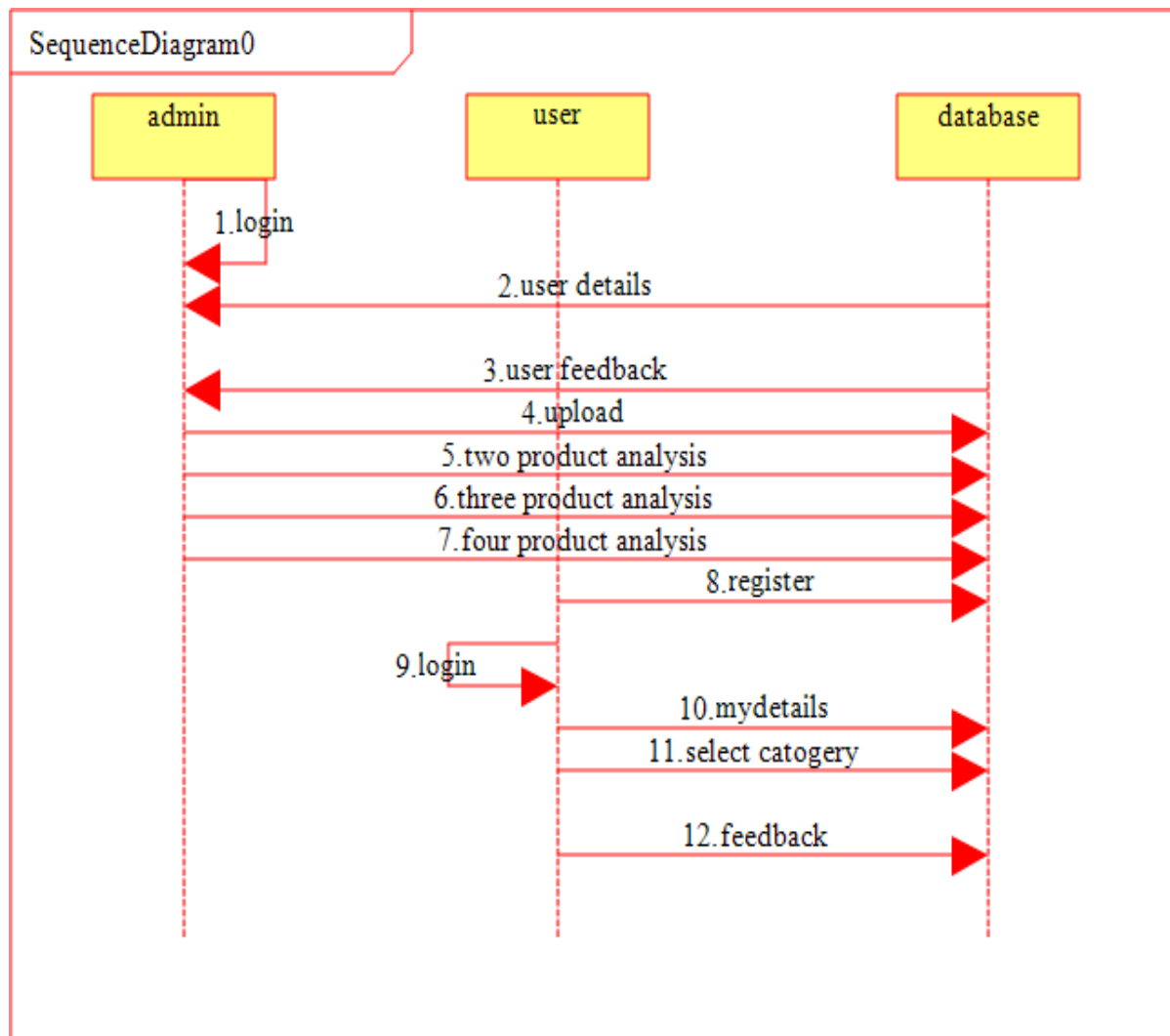
It describes how users will perform tasks on the website. It outlines, from a user's point of view, a system's behavior as it responds to a request. Each **use case** is represented as a sequence of simple steps, beginning with a user's goal and ending when that goal is fulfilled.

2.3.3. Activity Diagram



Activity diagrams are an essential part of the modeling process. They are used to clarify complicated use cases, illustrate control among objects, or to show the logic of an algorithm.

2.3.4. Sequence Diagram



A sequence diagram is a type of interaction diagram because it describes how and in what order a group of objects works together. Sequence diagrams are sometimes known as event diagrams or event scenarios.

2.3.5. Identifying classes

In this project we use four classes. They represent as models. They are as follows:

- Two product model – This is used to analyze two products.
- Three product model – This is used to analyze three products.
- Four product model – This is used to analyze four products.
- Feedback model – This is used to analyze feedback of users to the product.

Identifying object-oriented classes is both a skill and an art. It's a process that one gets better at over time. For example, it's not unusual for inexperienced designers to identify too many classes. Modelling too many classes' results in poor performance, unnecessary complexity and increased maintenance. On the other hand, too few classes tend to increase couplings, and make classes larger and unwieldy. In general, strive for class cohesiveness where behaviour is shared between multiple, related classes rather than one very large class.

Cohesive classes reduce coupling, enable extensibility and increase maintainability.

Moreover, classes that seem obvious wind up being poor choices and classes that are initially hidden or that rely on problem domain knowledge wind up as the best choices.

Therefore, begin class modeling by identifying candidate classes - an initial list of classes from which the actual design classes will emerge. Design classes are the classes that are modeled. Candidate classes exist for the sole purpose of deriving the design classes. Initially, there will be a lot of candidate classes – that's good. However, through analysis, their number will be reduced as they are dropped, combined and merged.

Candidate classes provide the initial impetus to produce cohesive classes.

Candidate classes can be discovered in a variety of ways. Here are three:

- Noun and noun phrases: Identify the noun and noun phrases, verbs (actions) and adjectives (attributes) from the Use Cases, Actor-Goal List, Application Narrative and Problem Description.

- CRC cards: an informal, group approach to object modeling.
- GRASP: A formal set of principles that assign responsibilities.

Each of these methods will yield a list candidate classes. The list won't be complete nor will every class be appropriate and there will likely be a mix of business and system oriented classes; i.e.; **Student** and **StudentRecord**, for example. That's fine. The goal is to identify the major classes - the obvious ones. Other classes will become apparent as the design process continues.

Once the list has been created, analyze the candidate classes for associations with other classes. Look for collaborating classes. How does each relate to each other and to the business process? Sometimes, it's helpful to ask, "Why keep this class?" In other words, assume the class is redundant or unnecessary. Keep it only if it plays a collaborating role. Often you'll find the class' functionality is accomplished by another class or within the context of another class.

When a class is kept, move it to the list of design classes. Eventually, a list of design classes will result that provides the foundational structure for the application.

Candidate Classes	Design Classes
Student	Student
Teacher	Professor
Class	Course
Subject	Assessment
Grades	
Tests	

Design classes will emerge from the analysis of the candidate classes.

Associations are the key to identifying cohesive classes. The following subsections identify the various associations that can exist between classes and suggestions to identify them.

Associations

The classes in an application system don't exist in a vacuum. Classes are associated with, or related to, other classes. These relationships occur when a class has, uses, knows about, or is acquainted with, one or more classes.

A relationship is an association between classes.

When identifying relationships, start with the class that interacts with as many other classes as possible; perhaps, the core classes of the application. It's helpful to ask, "Who cares about this class?", "Who is interested in this class?", "Why is this class necessary?" Starting with the core classes will quickly identify the other relationships.

Start with core associations.

An association is usually modeled using a solid line that connects two classes.



Figure 1: The Professor class is associated with the Student class

Most times, a single line doesn't provide enough information. Form a practice of giving each association a name to clarify the relationship. Use verbs or simple verb phrases as association names.

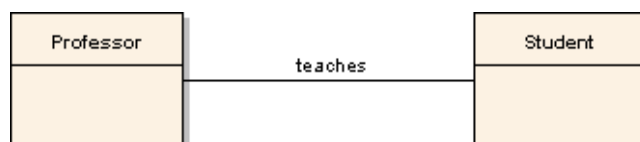


Figure 2: Professor "teaches" a Student

Association roles

Associations also have roles. Each class in an association has a role that describes its meaning in the relationship. Roles are optional and if, used, should describe the role as a noun. For example, a **Professor** is an instructor to a **Student** who is a learner or pupil.

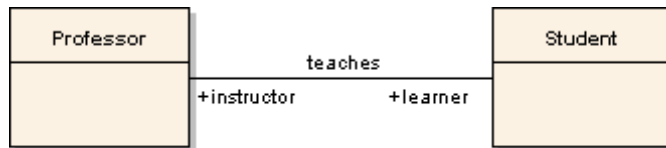


Figure 3: Association Roles

Multiplicity Indicators

A role can have multiplicity - an indication of how many objects participate in the relationship. Multiplicity indicators can be conditional or unconditional.

Examples of multiplicity indicators are:

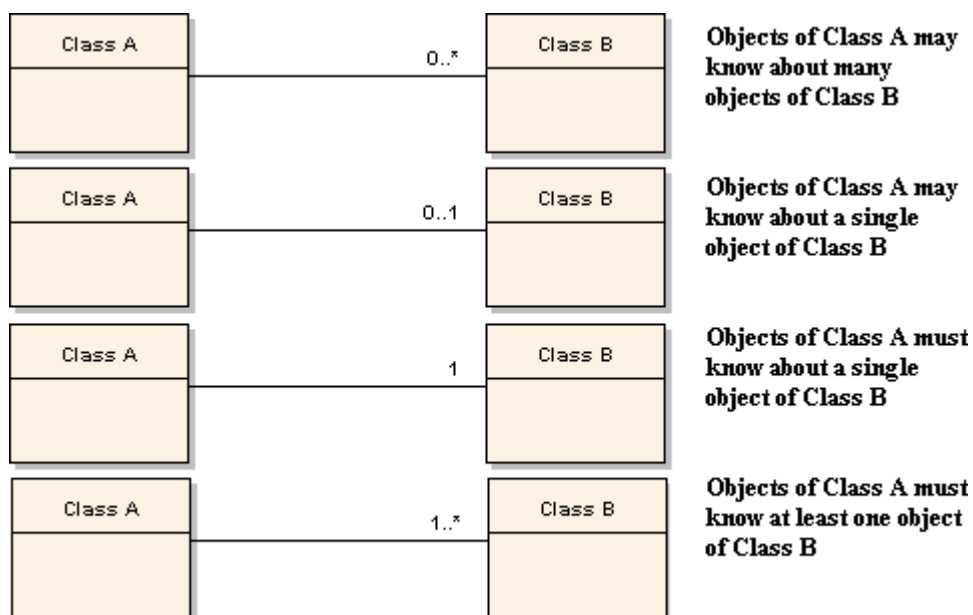


Figure 4: Multiplicity indicators

In Figure 4, the first two indicators (that start with "0...") are conditional meaning no objects need be present in the relationship. The last two indicators (that start with "1..") are unconditional meaning at least one object must be present in the relationship. For example:

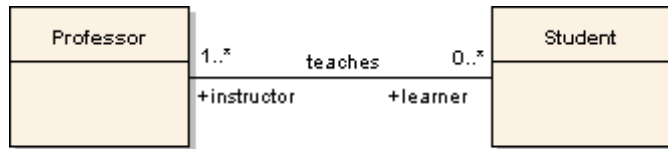


Figure 5: Conditional and Unconditional Indicators

In Figure 5, a **Professor** can exist without the existence of a **Student**. Therefore, the multiplicity for the **Student** is $0..*$. However, the same is not true for the **Student**. A **Student** must have at least one **Professor**. Hence, the $1..*$ indicator for the **Professor**.

Multiplicity indicators are also referred to as cardinalities.

Unconditional indicators may impose a referential integrity constraint. Consider the following example:



Figure 6: Special Case Condition

Class **A** and class **B** depend on the existence of the other. Since the multiplicity is unconditional the following is implied:

- When class **A** (or class **B**) is removed, the corresponding class **B** (or class **A**) must also be removed.
- When class **A** (or class **B**) is added, the corresponding class **B** (or class **A**) must also be added.

Unconditional indicators must be checked for referential integrity constraints..

Association class

An association can also possess its own attributes and behavior – just like a class. Sometimes, data exist that does not strictly belong to any of the participating classes. In these cases, an Association class is created to map the data to the participating classes. The class,

then, becomes the association. The association class will contain attributes that include pointers, or references, to instances of the two classes.

For example, a **Student** class has an association to a **Course** class. A student can take many courses, and a course can be taken by many students. However, who is responsible for the grade? Placing the grade in the **Student** class gives a student the same grade for all courses. Placing the grade in the **Course** class gives all students taking the same course the same grade.

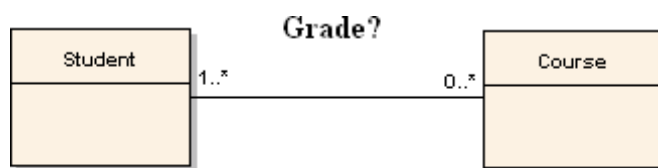


Figure 7: Where to place the grade?

The Association class resolves this issue by linking (or mapping) the grade (and other attributes) to the **Student** and the **Course** classes. Now, a **Student** can have many grades for many courses, but each grade in the **StudentCourseAssociation** is associated to a single student and course.

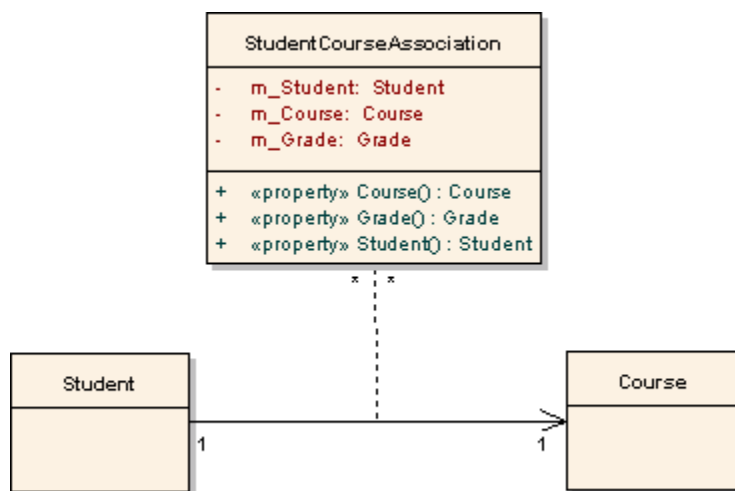


Figure 8: Association Class

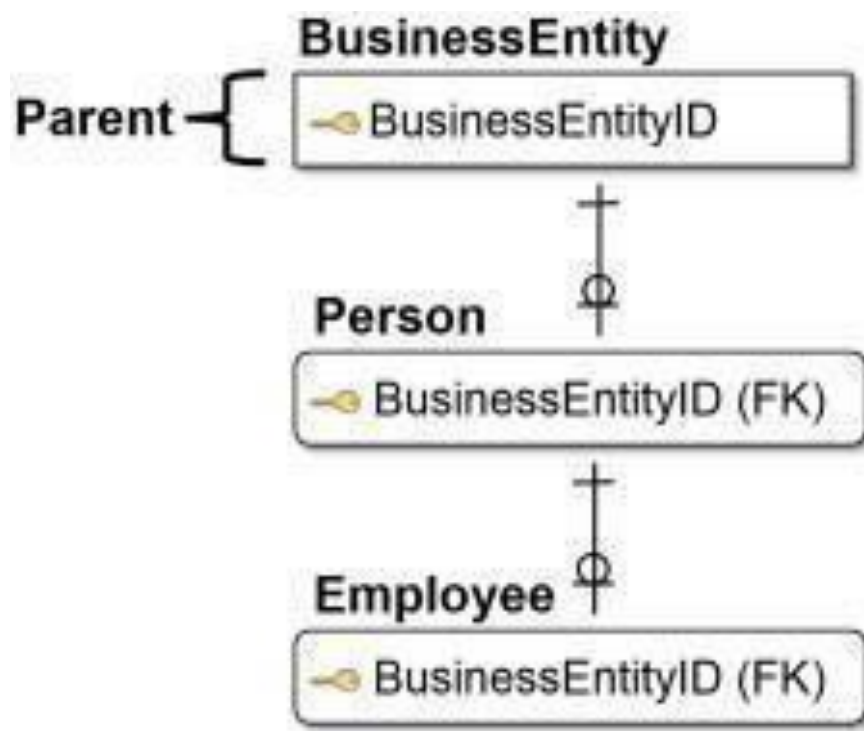
Association classes often occur in many-to-many associations.

2.3.6. Identifying the relationships

As the name implies, the identifying relationship establishes the parent as a way to identify and classify the child. In this type of relationship, the primary key from the parent migrates through the relationship to become part of the primary key, or identity, of the child. Therefore, the child entity is dependent upon the parent for its identification or classification, and it cannot exist without the parent.

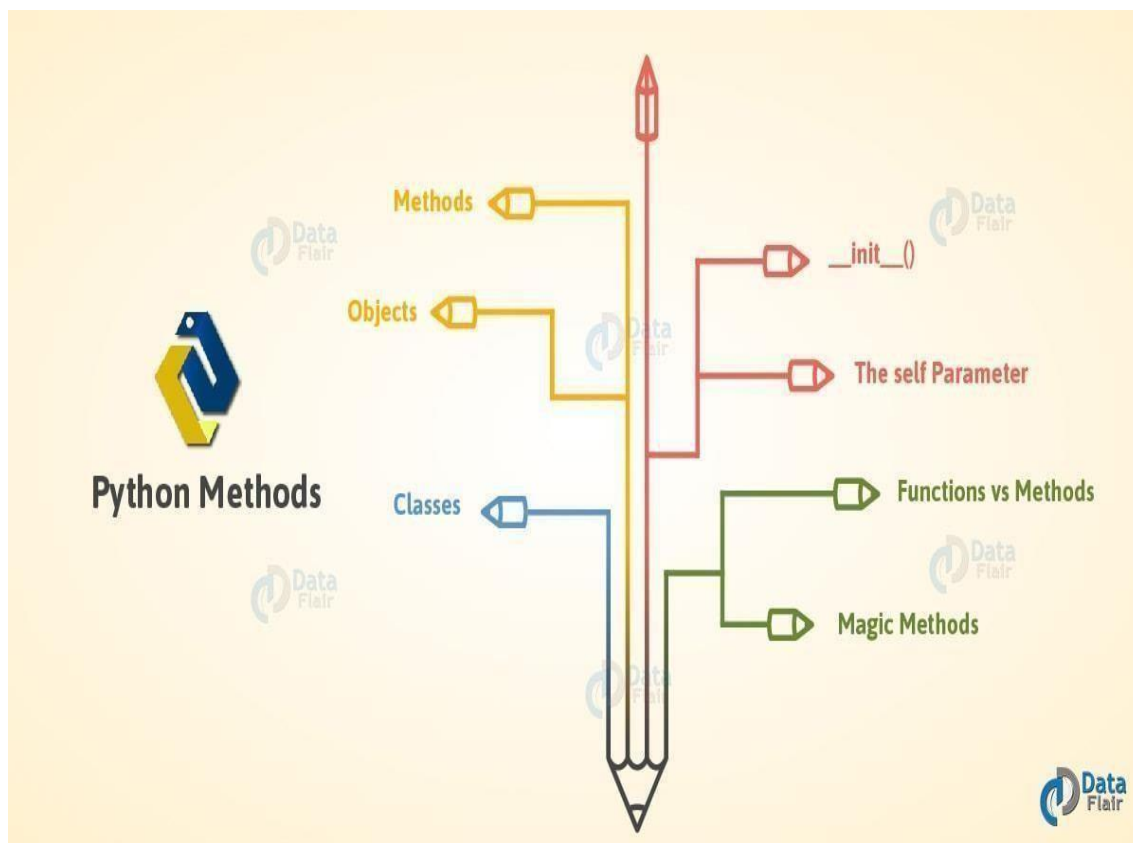
Shows the highest level entity, *Business Entity*. This entity can define multiple types of entities, such as an organization, asset, or person. *Business Entity* is the top-level parent and identifies the child entity *Person*, which uses the *Business Entity ID* as one of its primary keys. Next, the *Employee* entity is the child of the *Person* entity that then defines the *Employee* entity.

Each of these entities has different attributes that are inherited by the children under them, as well as their identity. It is a classic object-oriented development approach called inheritance, and it is a way to reuse attributes and entity characteristics.



2.3.7. Identifying the methods

- You are aware of the fact that Python is an object-oriented language. This means that it can deal with classes and objects to model the real world.
- A method is a label that you can call on an object; it is a piece of code to execute on that object.
- But before we begin getting any deeper, let's take a quick look at classes and objects.
- A Class is an Abstract Data Type (ADT). Think of it like a blueprint. A rocket made from referring to its blueprint is according to plan.
- It has all the properties mentioned in the plan, and behaves accordingly. Likewise, a class is a blueprint for an object.

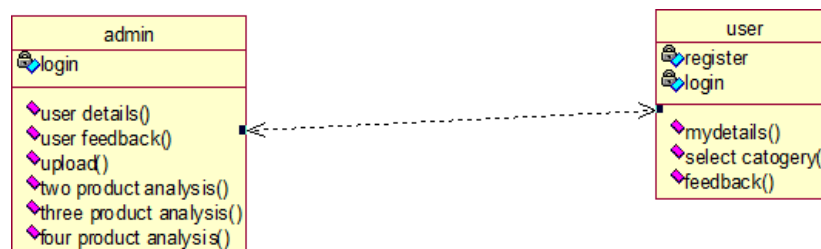


2.3.8. Identifying the attributes

Entities contain *attributes*, which are characteristics or modifiers, qualities, amounts, or features. An attribute is a fact or no decomposable piece of information about an entity. Later, when you represent an entity as a table, its attributes are added to the model as new columns.

You must identify the entities before you can identify the database attributes. After you determine the entities, ask yourself, “What characteristics am I required to know about each entity?” For example, in an *address* entity, you probably require information about *street*, *city*, and *zip code*. Each of these characteristics of the *address* entity becomes an attribute.

2.3.9. Business class diagram



Class diagrams are the main building block in object-oriented modelling. They are used to show the different objects in a system, their attributes, their operations and the relationships among them.

3 - DESIGN PHASE

DESIGN PHASE

3.1. Design Process

INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system.

The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OUTPUT DESIGN

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
2. Select methods for presenting information.
3. Create document, report, or other formats that contain information produced by the system.

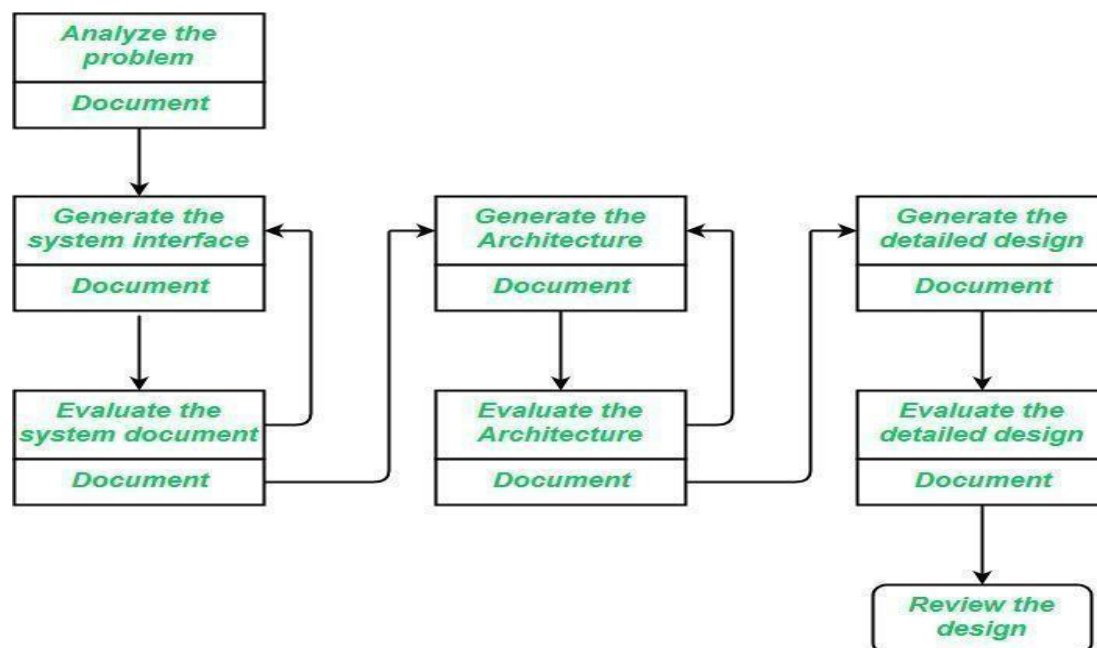
The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

The design phase of software development deals with transforming the customer requirements as described in the SRS documents into a form implementable using a programming language.

The software design process can be divided into the following three levels of phases of design:

1. Interface Design
2. Architectural Design
3. Detailed Design



Interface Design:

It is the specification of the interaction between a system and its environment. this phase proceeds at a high level of abstraction with respect to the inner workings of the system i.e, during interface design, the internal of the systems are completely ignored and the system is treated as a black box. Attention is focussed on the dialogue between the target system and the users, devices, and other systems with which it interacts. The design problem statement produced during the problem analysis step should identify the people, other systems, and devices which are collectively called *agents*.

Interface design should include the following details:

- Precise description of events in the environment, or messages from agents to which the system must respond.
- Precise description of the events or messages that the system must produce.
- Specification on the data, and the formats of the data coming into and going out of the system.
- Specification of the ordering and timing relationships between incoming events or messages, and outgoing events or outputs.

Architectural Design:

It is the specification of the major components of a system, their responsibilities, properties, interfaces, and the relationships and interactions between them. In architectural design, the overall structure of the system is chosen, but the internal details of major components are ignored.

Issues in architectural design includes:

- Gross decomposition of the systems into major components.
- Allocation of functional responsibilities to components.
- Component Interfaces
- Component scaling and performance properties, resource consumption properties, reliability properties, and so forth.
- Communication and interaction between components.

The architectural design adds important details ignored during the interface design. Design of the internals of the major components is ignored until the last phase of the design.

Detailed Design:

It is the specification of the internal elements of all major system components, their properties, relationships, processing, and often their algorithms and the data structures. The detailed design may include:

- Decomposition of major system components into program units.
- Allocation of functional responsibilities to units.
- User interfaces
- Unit states and state changes
- Data and control interaction between units
- Data packaging and implementation, including issues of scope and visibility of program elements
- Algorithms and data structures

Attention reader! Don't stop learning now. Get hold of all the important CS Theory concepts for SDE interviews with the [CS Theory Course](#) at a student-friendly price and become industry ready.

3.2. Design axioms

The Design Axioms describe the minimal rule set for designing interfaces. The foundational concepts that are required knowledge for engineers and designers to create usable and elegant interfaces. The design of the content (and book) will follow the same tenets outlined within it.

3.3. Refining attributes

Refine attributes. Design methods and the protocols by utilizing a UML, activity diagram to represent the method's algorithm. Refine the associations between classes, (if required). Refine the class hierarchy and design with inheritance (if required).

3.4. Refining methods

Modern software systems are often large and complicated. To better understand, develop, and manage large software systems, researchers have studied software architectures that provide the top level overall structural design of software systems for the last decade. One major research focus on software architectures is formal architecture description languages, but most existing research focuses primarily on the descriptive capability and puts less emphasis on software architecture design methods and formal analysis techniques, which are necessary to develop correct software architecture design. Refinement is a general approach of adding details to a software design.

A formal refinement method can further ensure certain design properties. This dissertation proposes refinement methods, including a set of formal refinement patterns and complementary verification techniques, for software architecture design using Software Architecture Model (SAM), which was developed at Florida International University.

First, a general guideline for software architecture design in SAM is proposed. Second, specification construction through property-preserving refinement patterns is discussed. The refinement patterns are categorized into connector refinement, component refinement and high-level Petri nets refinement. These three levels of refinement patterns are applicable to overall system interaction, architectural components, and underlying formal language, respectively.

Third, verification after modeling as a complementary technique to specification refinement is discussed. Two formal verification tools, the Stanford Temporal Prover (STeP) and the Simple Promela Interpreter (SPIN), are adopted into SAM to develop the initial models. Fourth, formalization and refinement of security issues are studied. A method for security enforcement in SAM is proposed.

The Role-Based Access Control model is formalized using predicate transition nets and Z notation. The patterns of enforcing access control and auditing are proposed. Finally, modeling and refining a life insurance system is used to demonstrate how to apply the refinement patterns for software architecture design using SAM and how to integrate the access control model.

The results of this dissertation demonstrate that a refinement method is an effective way to develop a high assurance system. The method developed in this dissertation extends existing work on modeling software architectures using SAM and makes SAM a more usable and valuable formal tool for software architecture design.

3.5. Refining class diagrams

- Refinement is a relation between two diagrams, not between.
- Individual entities.
- For a diagram refinement to be valid, these rules must hold.
- Every class in the abstract diagram must map too exactly.
- One class in the concrete diagram.

3.6. Coupling and Cohesion

Cohesion

- Cohesion is the indication of the relationship within module.
- It is concept of intra-module.
- Cohesion has many types but usually high cohesion is good for software.
- Cohesion is the concept of intra module.
- Cohesion represents the relationship within module.
- Increasing in cohesion is good for software.
- Cohesion represents the functional strength of modules.
- Highly cohesive gives the best software.
- In cohesion, module focuses on the single thing

Coupling

- Coupling is also the indication of the relationships between modules.
- It is concept of Inter-module.
- Coupling has also many types but usually low coupling is good for software.
- Coupling is the concept of inter module.
- Coupling represents the relationships between modules.
- Increasing in coupling is avoided for software.
- Coupling represents the independence among modules.

3.7. Design of Access Layer

The **access layer** ensures that packets are delivered to end user devices. This **layer** is sometimes referred to as the desktop **layer**, because it focuses on connecting client nodes to the network. **Access layer** devices include hubs, multi-station **access** units and switches.

3.7.1. Designing of Access Layer classes

- Creating access layer is to create a set of classes that know how to communicate with the places where the actual data resides.
- The access layer must be able to translate the data request from the business layer into protocol for data access.
- Also retrieved data into the business layer format.

3.7.2. Listing of methods of Access Layer

Core Peripheral Access Layer

- Name definitions, address definitions, and helper functions to access core registers and core peripherals like the NVIC, SCB, and SysTick

Middleware Access Layer (work in progress)

- Common method to access peripherals for typical embedded systems
- Targeted at communication interfaces including UART, Ethernet, and SPI
- Allows embedded software to be used on any Cortex microcontrollers that support the required communication interface

Device Peripheral Access Layer (MCU specific)

- Register name definitions, address definitions, and device driver code to access peripherals

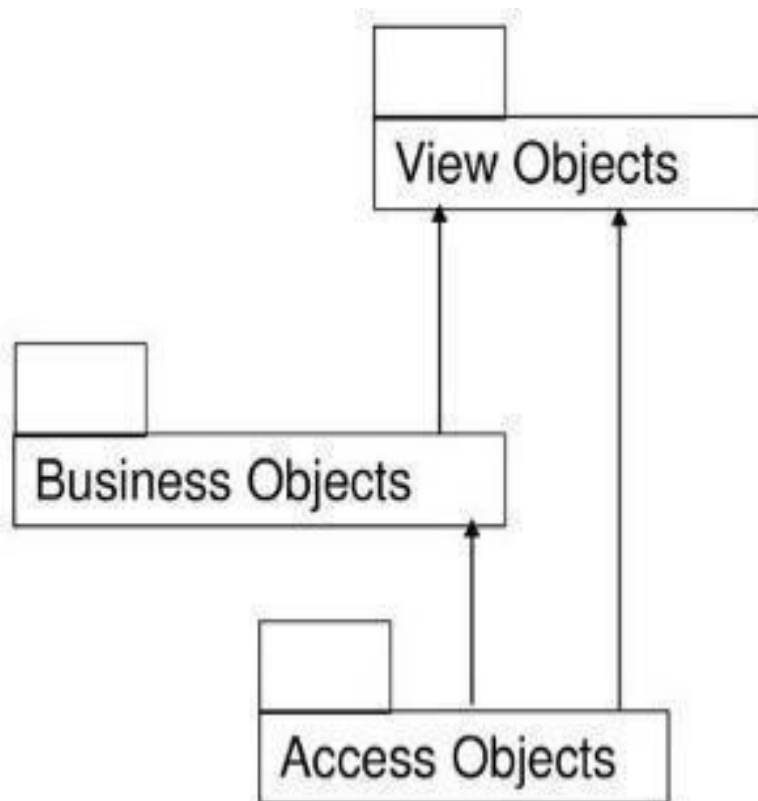
Access Functions for Peripherals (MCU specific)

- Optional helper functions for peripherals

3.8. Design of View Layer

3.8.1. Designing view layer classes

- UI **layer** consists of – objects with which user interacts – Objects needed to manage or control UI
- Responsibility of **view layer** objects – Input: responding to user interaction
- Translating user's action into an appropriate response – Output: displaying or printing business objects.
- An implicit benefit of three layer Architecture.
- Separation of the layer from the business and access layer.



3.8.2. View Layer

The **view layer class** or interface's objects are objects, represents some operations in the business that user must perform, to complete their tasks. Two major aspects of the applications are responsible by **view layer** objects. Renders can be separated into **layers**, to composite them back together afterwards. Some example usages are applying compositing

effects to characters separately, blurring the background and foreground **layers** separately for depth of field, or rendering different lighting variations of the same scene.

3.8.3. Designing interface behaviour

User interface (UI) design patterns are reusable/recurring components which designers use to solve common problems in user interface design. For example, the breadcrumbs design pattern lets users retrace their steps. Designers can apply them to a broad range of cases, but must adapt each to the specific context of use.

- **Keep the interface simple.** The best interfaces are almost invisible to the user. They avoid unnecessary elements and are clear in the language they use on labels and in messaging.
- **Create consistency and use common UI elements.** By using common elements in your UI, users feel more comfortable and are able to get things done more quickly. It is also important to create patterns in language, layout and design throughout the site to help facilitate efficiency. Once a user learns how to do something, they should be able to transfer that skill to other parts of the site.
- **Be purposeful in page layout.** Consider the spatial relationships between items on the page and structure the page based on importance. Careful placement of items can help draw attention to the most important pieces of information and can aid scanning and readability.
- **Strategically use colour and texture.** You can direct attention toward or redirect attention away from items using colour, light, contrast, and texture to your advantage.
- **Use typography to create hierarchy and clarity.** Carefully consider how you use typeface. Different sizes, fonts, and arrangement of the text to help increase scanability, legibility and readability.
- **Make sure that the system communicates what's happening.** Always inform your users of location, actions, changes in state, or errors. The use of various UI elements to communicate status and, if necessary, next steps can reduce frustration for your user.
- **Think about the defaults.** By carefully thinking about and anticipating the goals people bring to your site, you can create defaults that reduce the burden on the user. This becomes particularly important when it comes to form design where you might have an opportunity to have some fields pre-chosen or filled out.

4 – DEVELOPMENT PHASE

DEVELOPMENT PHASE

4.1. Frontend Features

- Coding Language : Python.
- Front-End : Python.
- Designing : Html, CSS, JavaScript.
- Framework : Django

4.1.1. Physical specification of attributes

In the physical world, no one builds anything without detailed blueprints, because people's lives are on the line. In the digital world, the stakes just aren't as high. It's called "software" for a reason: because when it hits you in the face, it doesn't hurt as much. No one is going to die if your website goes live with the header's left margin 4 pixels out of alignment with the image below it.

But, while the users' lives might not be on the line, design blueprints (also called design specifications, or specs) could mean the difference between a correctly implemented design that improves the user experience and satisfies customers and a confusing and inconsistent design that corrupts the user experience and displeases customers.

CSS Specification

1. Writing specifications

Code indentation: four empty spaces

The selector separate line, attributes and attribute values for each row and the attributes end with a semicolon;

{ } Will not increase the vertical blank lines, a separate line }, a blank line between each attribute

Prepared by component pattern block, and add the appropriate annotations

Notes unity with / ** /

For color parameter or attribute value, less than 1 0 omitted front

Hexadecimal values should be all lowercase, for example, #fff. When scanning documents, lowercase characters are easy to distinguish, because they form more easily distinguishable.

Avoid the specified value of 0 units,

e.g., `withmargin: 0;`

`Substitutemargin: 0px;`

2. Declaration order

Related attribute declaration should be grouped together, and arranged in the following order:

- Positioning
- Box model
- Typographic
- Visual

3. The position of media queries

The media query on the rules as far as possible in the vicinity. Do not pack them in a single style file or on the bottom of the document.

4. Attributes prefixed

When a vendor-specific attribute with the prefix, by way of indentation, so that the value of each attribute are aligned in the vertical direction, which facilitates multi-line edit

5. Shorthand property

In case of need to set all values displayed, use the short form:

- padding
- margin
- font
- background
- border
- border-radius

6. The selector

- For common elements use class, so conducive to optimize rendering performance.
- For components often avoid the use attribute selectors (e.g.,[class^="..."]). The browser's performance will be affected by these factors.
- Selectors as short as possible, and try to limit the number of elements selectors, it is recommended not to exceed 3.
- onlyWhen necessary, it will be limited to the last class of the parent element.
- Supplement css multi-level class name and use the selector.
- Simply put, it is to do the outermost namespace prefix subset, css plus a unique parent name (when setting specific style can only option on the line, have flexible use), the selector preferably not more than three nested

7. class name

- Lowercase characters and dashes (dashe) (not underlined, nor hump nomenclature) class name can only appear. Dash associated class name should be used (similar to the named space) (e.g.,.btn with .btn-danger) 。
- Avoid excessive arbitrary shorthand..btnOn behalf ofbutton, But.sNot express any meaning. (A set of common abbreviations)
- class name should be as short and clear meaning.
- Use a meaningful name. Organized or purposeful use of the name, do not use expressions (presentational) name.
- Based on recent parent class or base (base) class as a prefix of the new class.
- Use.js-*class to identify behavior (as opposed to the style) and do not include these class to the CSS file.

Additional information, common abbreviations and naming, downloads view:

[HTML DIV + CSS naming Daquan .txt](#)

8. font-family default settings

font-family: "PingFangSC", "Segoe UI", "Lucida Grande", Helvetica, Arial, "Microsoft YaHei", FreeSans, Arimo, "Droid Sans", "wenquanyi micro hei", "Hiragino Sans GB", "Hiragino Sans GB W3", Arial, sans-serif;

HTML specification

1. Doctype standard wording H5, and add the language code attribute with the default en
<!DOCTYPE html>
Language Reference Code:<https://www.sitepoint.com/web-foundations/iso-2-letter-language-codes/>
2. the character encoding settings
<meta charset="utf-8">
3. nesting indents, four spaces, some code structure clearer
4. using semantic tags, on the one hand to improve the readability of the code, on the other hand, more friendly to seo

The following are some common semantic tags H5

Header element

- It stands for "page" or "section" of the header.
- Usually contains h1-h6 or elements group as the entire page or a piece of content title.
- It can also be part of a package of directory, a search box, a navigation or any related logo.

Footer elements

footerElement represents a "page" or "section" footer usually contains some basic information about the festival, such as: author, links to related documents, copyright data. in case footerElement contains an entire section, then they are representative of the appendix, index, promoted a number of other similar information license, tags, category, etc.

hgroup element

hgroupThe title element represents a "page" or "section", and when there are multiple levels of the element, the element can be h1 to h6 elements on the inside, such as a combination of the main title and subtitle of the article

nav element

navNavigation link element represents the area of the page. The main navigation section for the definition of the page.

aside element

asideAttached information element is included as part of the main content of the article element, the contents of which may be relevant information related to the current article, labels, ranking interpretation. (Special section)

section element

sectionElement represents the document "Festival" or "block", "block" may refer to an article in accordance with sub-themes; "section" may refer to a page in the packet.

section usually with the title, although in html5 section will automatically give the title h1-h6 downgrade, but it is best to give them a manual downgrade

article element

articleElements most likely to followsectionwithdivConfusing, in fact,articleOn behalf of a self-contained body of content in a document, page or site, its purpose is to allow developers to independently develop or reuse.

5. tag attributes double quotes, all lowercase attribute names, attribute written according to the order

Referring sequence properties:

- class
- id, name
- data-*
- src, for, type, href, value
- title, alt
- role, aria-*

- required, readonly, disabled
6. Boolean attributes without assigningSuch as disable, checked
`<input type="text" disabled>`
`<input type="checkbox" value="1" checked>`
`<select>`
`<option value="1" selected>1</option>`
`</select>`
 7. to minimize redundant labels and nesting parent element, HTML nesting preferably not more than four layers.
 8. automatic closing at the end of the label to be used slas
``
 9. The introduction of CSSAnd JSThere is no need to specify the typeAs text / cssAnd text / javascriptThey are the default values
 10. JS file content to generate labels to make it more difficult to find, harder to edit, worse performance. We should tryto avoid this situation

JS specification

- indents
- the length of the single line
- semicolon
- space
- blank lines
- line
- single-line comments
- multi-line comments
- Documentation Comments
- quotes
- variable declaration
- function
- brackets
- null
- undefined

4.1.2. Special Features of Language

Python

- Python supports both procedure-oriented and object-oriented programming which is one of the key python features. It also supports multiple inheritance, unlike Java. A class is a blueprint for such an object. It is an abstract data type and holds no values.
- Python is a general-purpose language, which means it can be used to build just about anything, which will be made easy with the right tools/libraries.

Professionally, Python is great for backend web development, data analysis, artificial intelligence, and scientific computing.

HTML

HTML definition is Hyper Text Markup Language and it is a markup language. A markup language is a set of markup tags and the tags describe document content. HTML documents contain HTML tags and plain text. HTML documents are also called web pages.

JavaScript

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with **the** user and make dynamic pages. It is **an** interpreted programming language with object-oriented capabilities.

CSS

- Advanced Animations.
- Multiple Backgrounds & Gradient.
- Multiple Column layouts.
- Opacity.
- Rounded Corner:
- Selectors.

4.2. Backend Features

- Coding Language : Python.
- Data Base : MySQL.

MySQL is a free-to-use, open-source database that facilitates effective management of databases by connecting them to the software. It is a stable, reliable and powerful solution with advanced features like the following:

1. Data Security

MySQL is globally renowned for being the most secure and reliable database management system used in popular web applications like WordPress, Facebook and Twitter. The data security and support for transactional processing that accompany the recent version of MySQL, can greatly benefit any business especially if it is an eCommerce business that involves frequent money transfers.

2. On-Demand Scalability

MySQL offers unmatched scalability to facilitate the management of deeply embedded apps using a smaller footprint even in massive warehouses that stack terabytes of data. On-demand flexibility is the star feature of MySQL. This open source solution allows complete customization to eCommerce businesses with unique database server requirements.

3. High Performance

MySQL features a distinct storage-engine framework that facilitates system administrators to configure the MySQL database server for a flawless performance. Whether it is an eCommerce website that receives a million queries every single day or a high-speed transactional processing system, MySQL is designed to meet even the most demanding applications while ensuring optimum speed, full-text indexes and unique memory caches for enhanced performance.

4. Comprehensive Transactional Support

MySQL tops the list of robust transactional database engines available on the market. With features like complete atomic, consistent, isolated, durable transaction support, multi-version transaction support, and unrestricted row-level locking, it is the go-to solution for full data integrity. It guarantees instant deadlock identification through server-enforced referential integrity.

5. Complete Workflow Control

With the average download and installation time being less than 30 minutes, MySQL means usability from day one. Whether your platform is Linux, Microsoft, Macintosh or UNIX, MySQL is a comprehensive solution with self-management features that automate everything from space expansion and configuration to data design and database administration.

6. Reduced Total Cost Of Ownership

By migrating current database apps to MySQL, enterprises are enjoying significant cost savings on new projects. The dependability and ease of management that accompany MySQL save your troubleshooting time which is otherwise wasted in fixing downtime issues and performance problems.

7. The Flexibility of Open Source

All the fears and worries that arise in an open source solution can be brought to an end with MySQL's round-the-clock support and enterprise indemnification. The secure processing and trusted software of MySQL combine to provide effective transactions for large volume projects. It makes maintenance, debugging and upgrades fast and easy while enhancing the end-user experience.

4.3. Test Cases

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 TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce 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.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields.

Integration tests demonstrate that although the components were individually satisfactory, as shown by successful unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

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.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

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.

White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

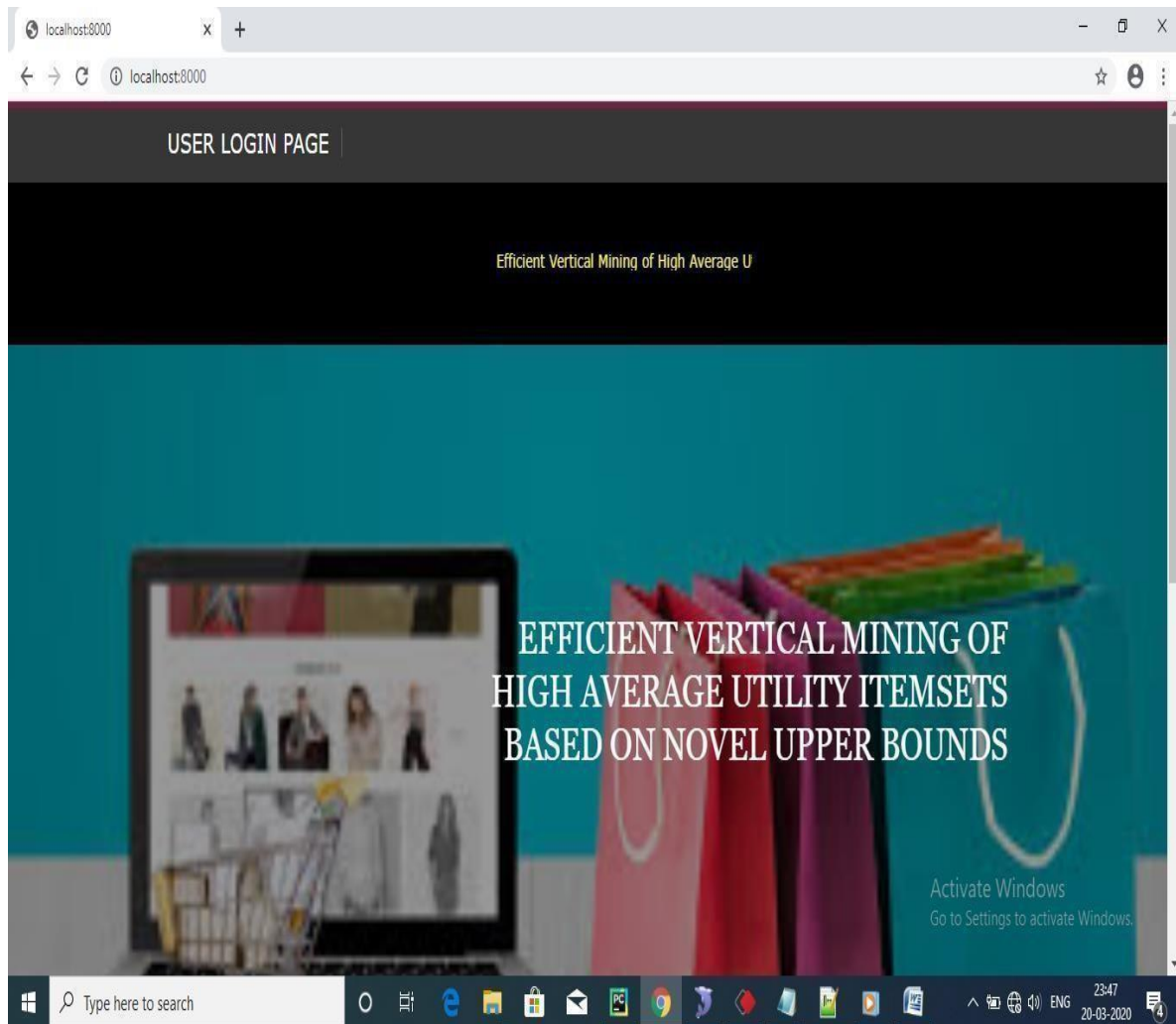
- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

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.

4.4. Sample Screens

Home page:



User register:

localhost:8000/register

localhost:8000/register

First Name	<input type="text" value="nivas"/>
Last Name	<input type="text" value="john"/>
User Id	<input type="text" value="ramavath"/>
Password	<input type="password" value="12345"/>
Mobile Number	<input type="text" value="9087654321"/>
Email Id	<input type="text" value="projectcse1234@gmail.co"/>
Gender	<input type="text" value="male"/>
<input type="submit" value="SUBMIT"/>	

REGISTRATION

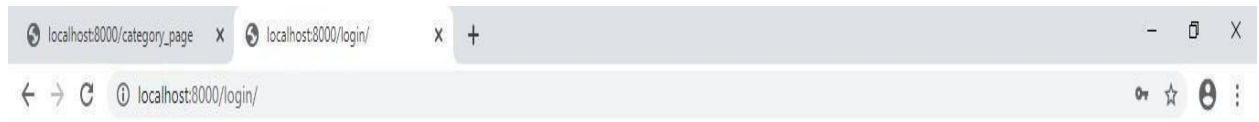
Activate Windows
Go to Settings to activate Windows

Type here to search

23:48
20-03-2020

User login:

Admin login:



- ADMIN LOGIN PAGE

•
•
•

Efficient Vertical Mining of High Average Utility Itemsets based

Efficient Vertical Mining of High Average Utility Itemsets based on Novel Upper Bounds

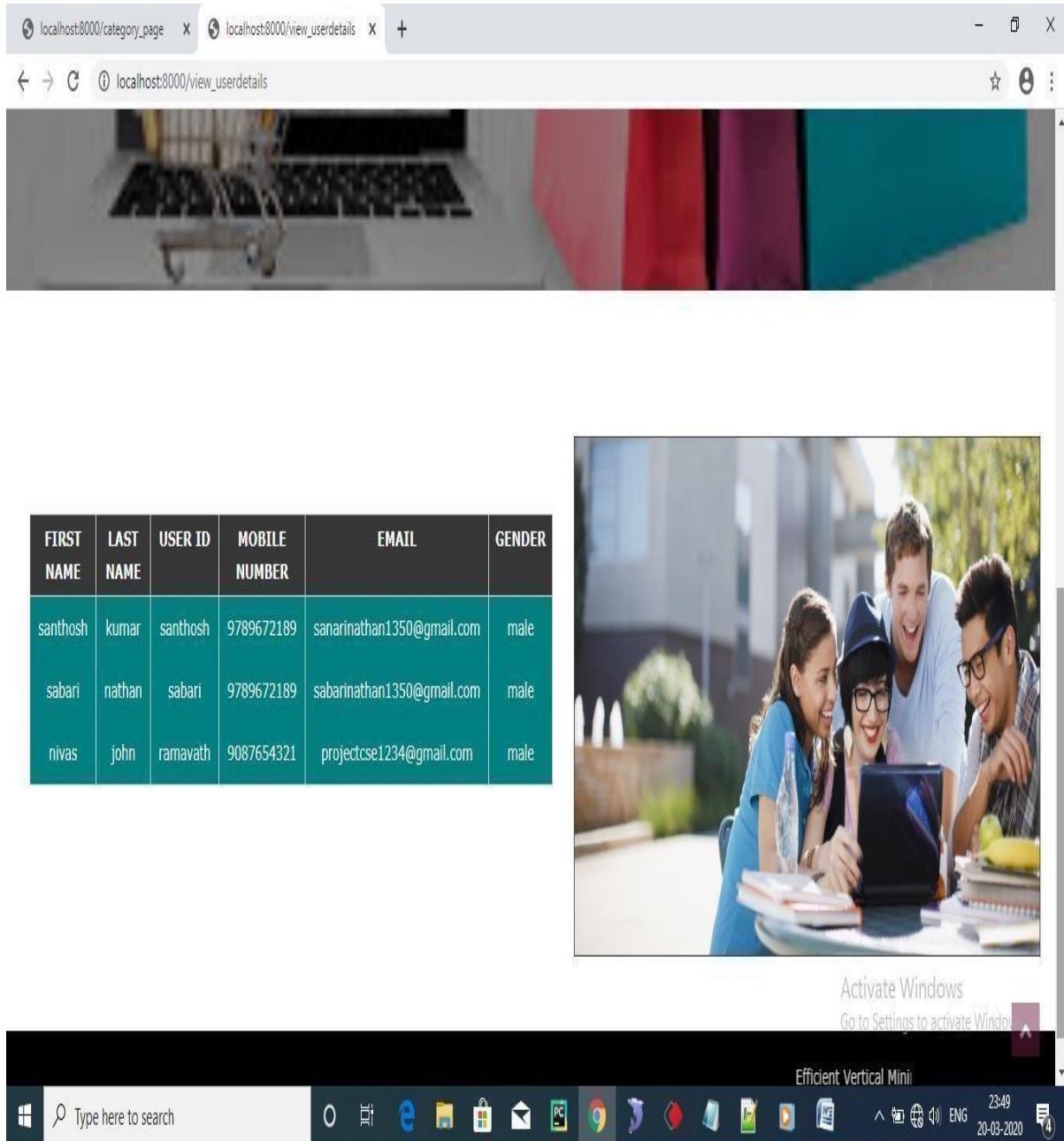
A login form with a grey background. It has a text input field with 'admin' and an eye icon, a password input field with dots and a lock icon, and a green 'LOGIN HERE' button.

Activate Windows

Go to Settings to activate Windows.

Efficient Vertical Mining of High Average Utility Itemsets based



User details:

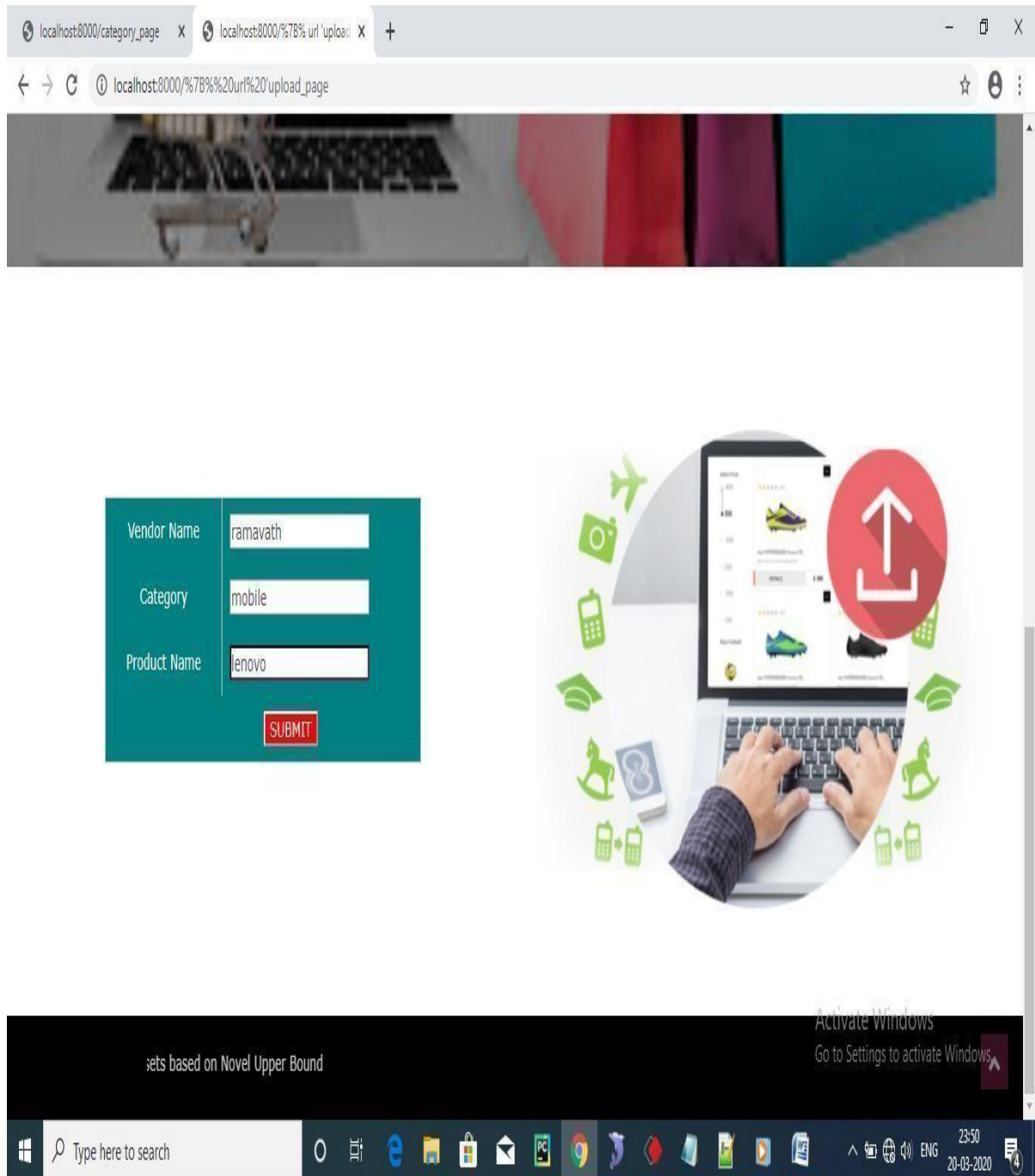
FIRST NAME	LAST NAME	USER ID	MOBILE NUMBER	EMAIL	GENDER
santhosh	kumar	santhosh	9789672189	sanarinathan1350@gmail.com	male
sabari	nathan	sabari	9789672189	sabarinathan1350@gmail.com	male
nivas	john	ramavath	9087654321	projectcse1234@gmail.com	male


Activate Windows
Go to Settings to activate Windows

Efficient Vertical Mini

Type here to search

23:49
20-03-2020

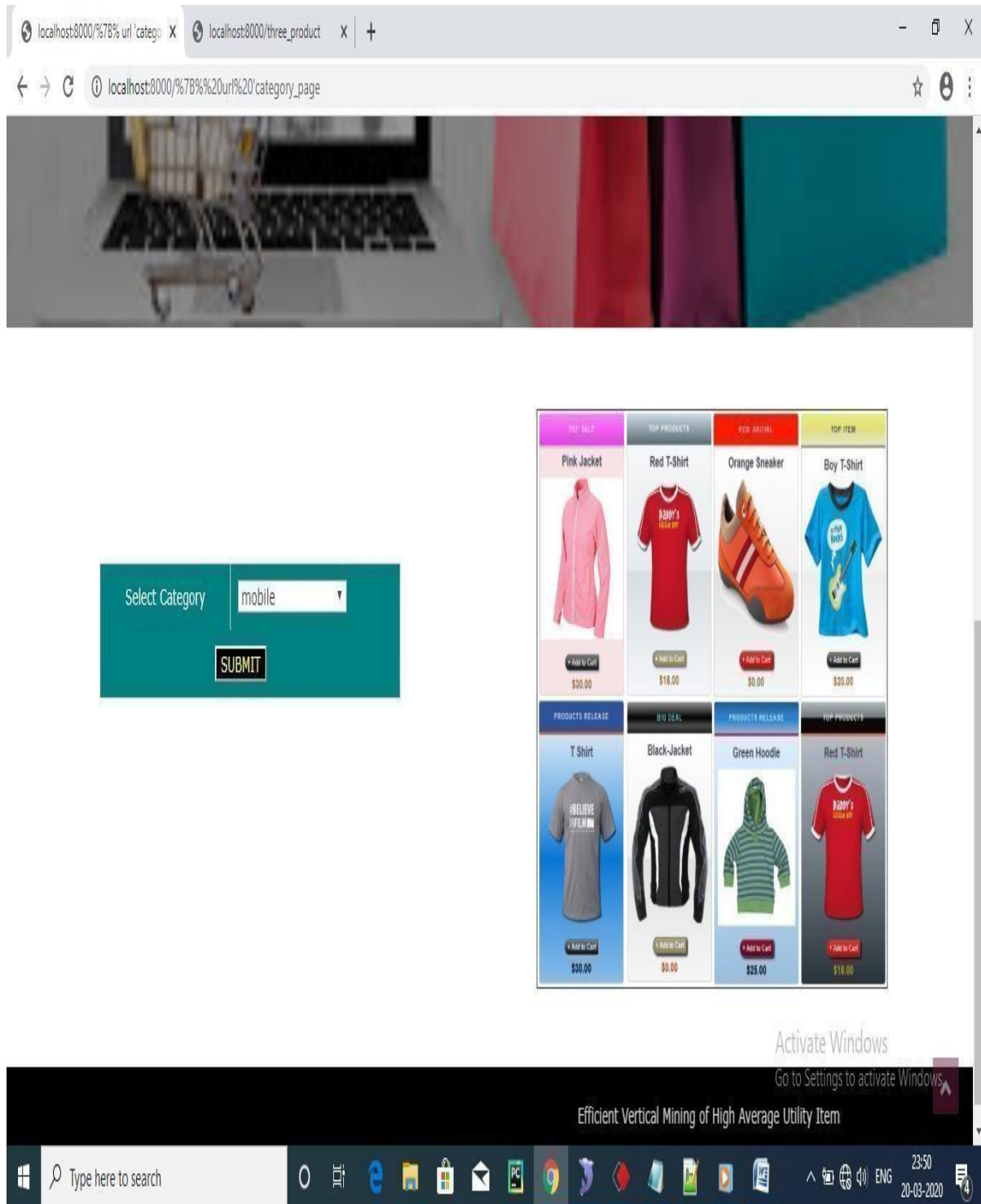
Upload:



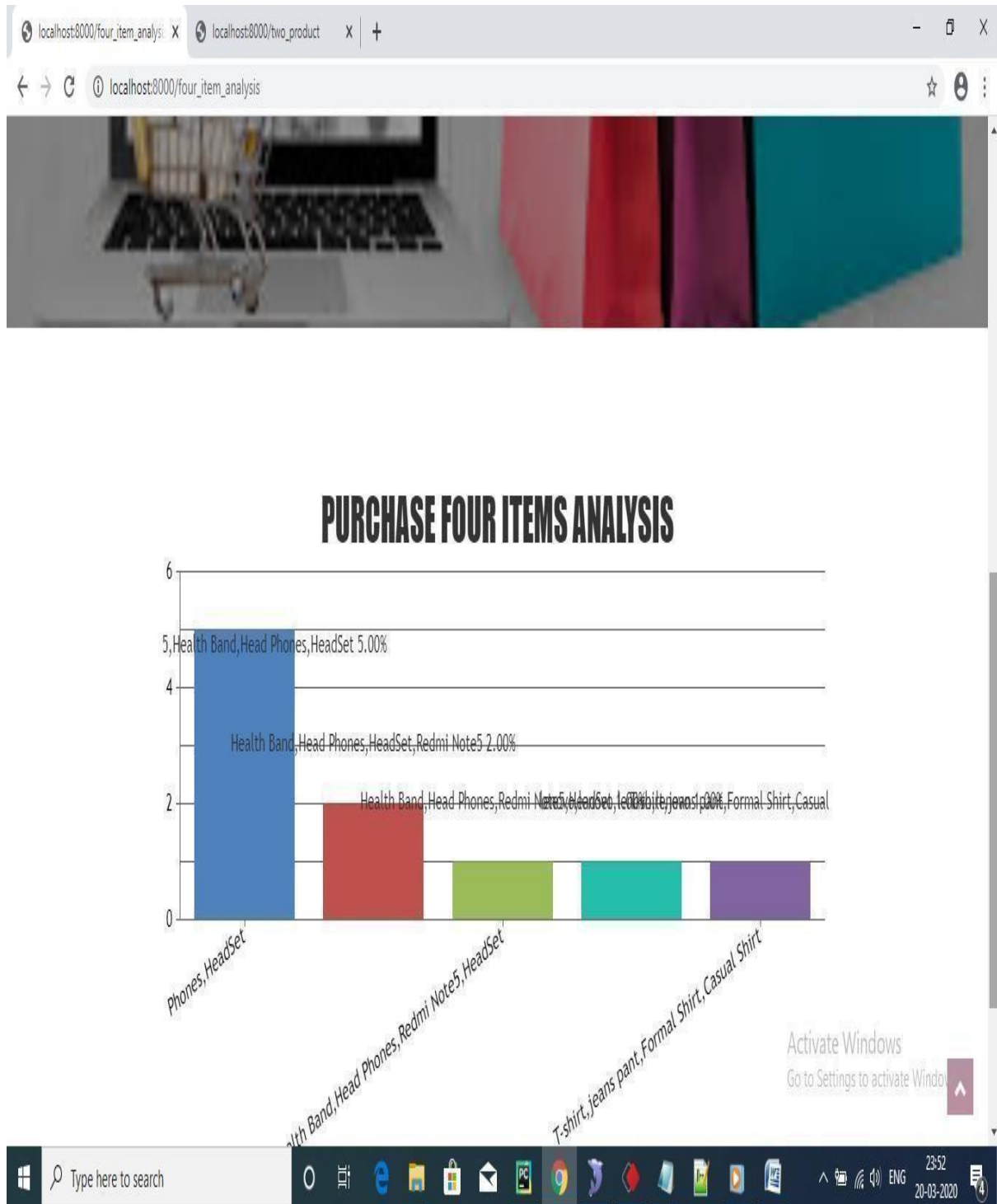
The screenshot shows a web browser with two tabs: 'localhost:8000/my_details' and 'localhost:8000/three_product'. The address bar shows 'localhost:8000/my_details'. The main content area features a blurred image of a shopping cart on a laptop screen. Below this, there is a table with user details:

First Name	nivas
Last Name	john
Userid	ramavath
Password	12345
Mobile Number	9087654321
Email	projectcse1234@gmail.com
Gender	male

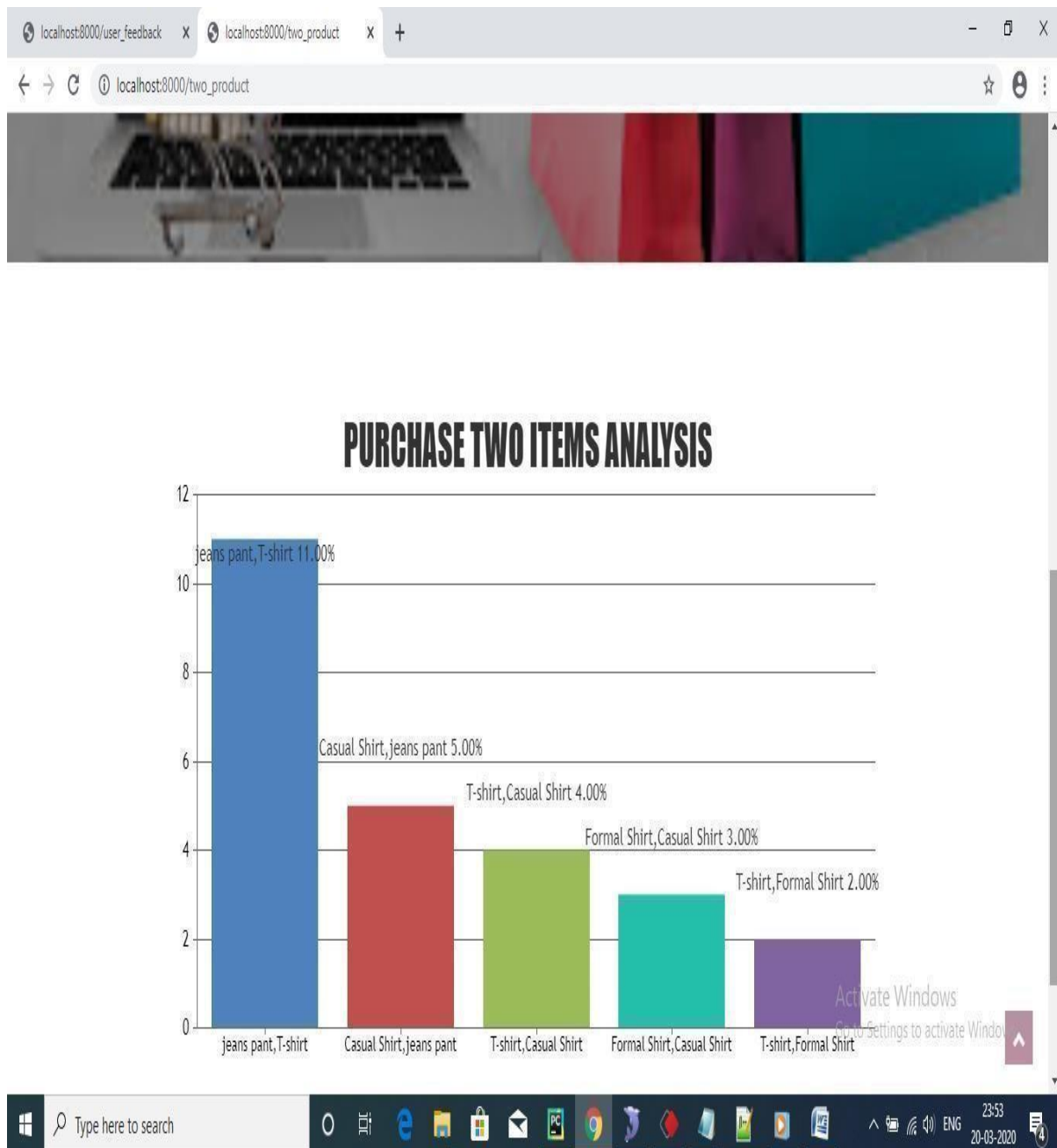
To the right of the table is a blurred image of three people looking at a laptop. The Windows taskbar at the bottom shows the search bar, task view, and various application icons. The system tray on the right indicates the time is 23:50 on 20-03-2020, with language set to ENG and 4 notifications.

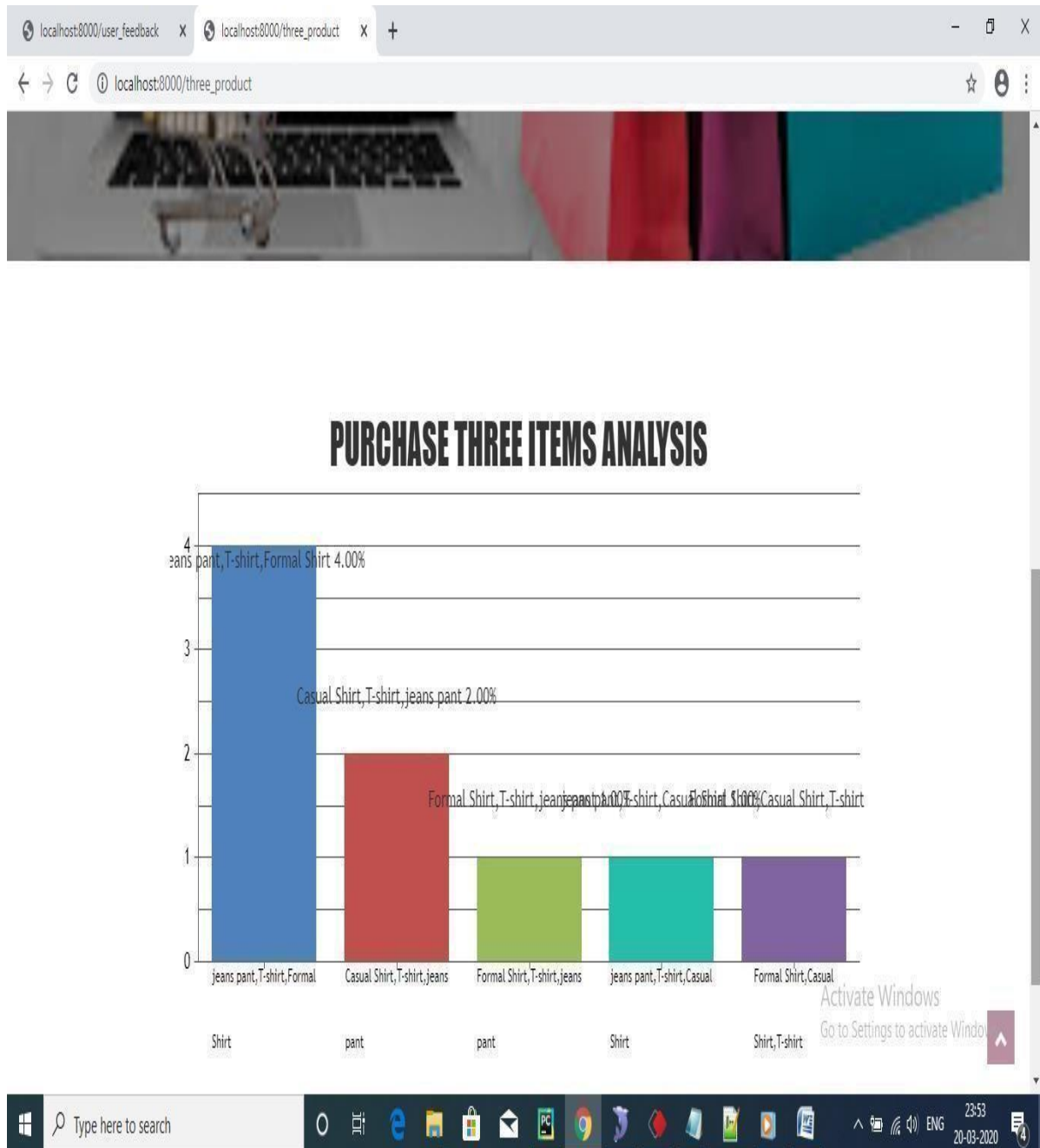








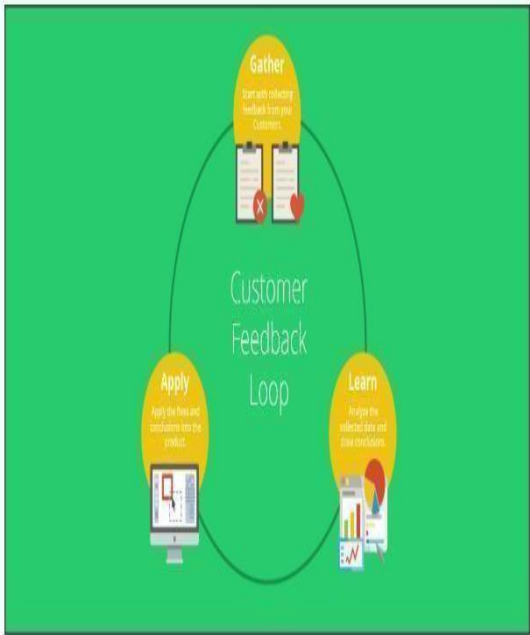




localhost:8000/user_feedback X localhost:8000/user_viewfeedback X +

localhost:8000/user_viewfeedback

USER NAME	FEEDBACK
santhosh	Thank you
ramavath	good



The diagram illustrates the Customer Feedback Loop on a green background. It features three yellow ovals connected by a circular arrow. The top oval is labeled 'Gather' and contains the text 'Start with collecting feedback from your customers' along with icons of a clipboard and a heart. The bottom-left oval is labeled 'Apply' and contains the text 'Apply the feedback and incorporate into the product' along with a laptop icon. The bottom-right oval is labeled 'Learn' and contains the text 'Analyze the collected data and draw conclusions' along with a bar chart icon. The central text 'Customer Feedback Loop' is written in a light blue font.

Activate Windows
Go to Settings to activate Windows.
Efficient Vertical Mir

Type here to search

23:53
20-03-2020

4.5. How our program is best

1. Mining High-Average Utility Item sets with Positive and Negative External Utilities

High-utility item set mining (HUIM) is an emerging data mining topic. It aims to find the high-utility item sets by considering both the internal (i.e., quantity) and external (i.e., profit) utilities of items. High-average-utility item set mining (HAUIM) is an extension of the HUIM, which provides a more fair measurement named average-utility, by taking into account the length of item sets in addition to their utilities.

In the literature, several algorithms have been introduced for mining high-average-utility item sets (HAUIs).

However, these algorithms assume that databases contain only positive utilities. For some real-world applications, on the other hand, databases may also contain negative utilities. In such databases, the proposed algorithms for HAUIM may not discover the complete set of HAUIs since they are designed for only positive utilities.

In this study, to discover the correct and complete set of HAUIs with both positive and negative utilities, an algorithm named MHAUIPNU (mining high-average-utility item sets with positive and negative utilities) is proposed.

MHAUIPNU introduces an upper bound model, three pruning strategies, and a data structure. Experimental results show that MHAUIPNU is very efficient in reducing the size of the search space and thus in mining HAUIs with negative utilities.

2. An Efficient Tree Based Algorithm for Mining High Average-Utility Item set

High-utility item set mining (HUIM), which is an extension of well-known frequent item set mining (FIM), has become a key topic in recent years. HUIM aims to find a complete set of item sets having high utilities in a given dataset. High average-utility item set mining (HAUIM) is a variation of traditional HUIM.

HAUIM provides an alternative measurement named the average-utility to discover the item sets by taking into consideration both of the utility values and lengths of item sets.

HAUIM is important for several application domains, such as, business applications, medical data analysis, mobile commerce, streaming data analysis, etc. In the literature, several algorithms have been proposed by introducing their own upper-bound models and data structures to discover high average utility item sets (HAUIs) in a given

database. However, they require long execution times and large memory consumption to handle the problem. To overcome these limitations, this paper, first, introduces four novel upper-bounds along with pruning strategies and two data structures.

Then, it proposes a pattern growth approach called the HAUL- Growth algorithm for efficiently mining of HAUIs using the proposed upper-bounds and data structures.

Experimental results show that the proposed HAUL-Growth algorithm significantly outperforms the state- of-the-art dHAUIM and TUB-HAUIM algorithms in terms of execution times, number of join operations, memory consumption, and scalability

3. A Survey of High Utility Item set Mining

High utility pattern mining is an emerging data science task, which consists of discovering patterns having a high importance in databases. The utility of a pattern can be measured in terms of various objective criteria as such as its profit, frequency, and weight.

Among the various kinds of high utility patterns that can be discovered in databases, high utility item sets are the most studied. A high utility item set is a set of values that appears in a database and has a high importance to the user, as measured by a utility function.

High utility item set mining generalizes the problem of frequent item set mining by considering item quantities and weights.

A popular application of high utility item set mining is to discover all sets of items purchased together by customers that yield a high profit. This chapter provides an introduction to high utility item set mining, reviews the state-of-the-art algorithms, their extensions, applications, and discusses research opportunities.

This chapter is aimed both at those who are new to the field of high utility item set mining, as well as researchers working in the field.

4. Utility-Driven Mining of Trend Information for Intelligent System

Useful knowledge, embedded in a database, is likely to change over time. Identifying recent changes in temporal databases can provide valuable up-to-date information to decision-makers. Nevertheless, techniques for mining high-utility patterns (HUPs) seldom consider recency as a criterion to discover patterns.

Thus, the traditional utility mining framework is inadequate for obtaining up-to-date insights about real world data. In this paper, we address this issue by introducing a

novel framework, named utility-driven mining of Recent/trend high-Utility Patterns (RUP) in temporal databases for intelligent systems, based on user-specified minimum recency and minimum utility thresholds.

The utility-driven RUP algorithm is based on novel global and conditional downward closure properties, and a recency-utility tree. Moreover, it adopts a vertical compact recency-utility list structure to store the information required by the mining process.

The developed RUP algorithm recursively discovers recent HUPs. It is also fast and consumes a small amount of memory due to its pattern discovery approach that does not generate candidates.

Two improved versions of the algorithm with additional pruning strategies are also designed to speed up the discovery of patterns by reducing the search space. Results of a substantial experimental evaluation show that the proposed algorithm can efficiently identify all recent high-utility patterns in large-scale databases, and that the improved algorithm performs best.

4.6. Future Extension

The proposed generic framework to evaluate utility upper-bounds using anti-monotone-like criteria and the corresponding pruning strategies proposed in this study can be extended for the more general problem of mining all high utility sequences in quantitative sequence databases. This will be considered for future work.

Based on the idea of computing utility values using a vertical form in quantitative databases, this paper has proposed four tight UBs, called aub_1 , aub , $iaub$ and $laub$, a generic framework to evaluate UBs in terms of their pruning effects, and three pruning strategies to eliminate unpromising candidates early. A new IDUL tree structure was also developed to quickly calculate the average utility and UBs of item sets using a recursive process.

A novel algorithm named dHAUIM has been further presented to efficiently mine high average-utility item sets. An extensive experimental evaluation was carried out. Results have shown that dHAUIM outperforms four state-of-the-art HAUI mining algorithms in terms of execution time and number of join operations on both real-life and synthetic databases.

Note that, the proposed generic framework to evaluate utility upper-bounds using anti-monotone-like criteria and the corresponding pruning strategies proposed in this study can be extended for the more general problem of mining all high utility sequences in quantitative sequence databases. This will be considered for future work.

5 - CONCLUSION

5.1. Conclusion

Based on the idea of computing utility values using a vertical form in quantitative databases, this paper has proposed four tight UBs, called \overline{u} , \overline{ub} and \overline{laub} , a generic framework to evaluate UBs in terms of their pruning effects, and three pruning strategies to eliminate unpromising candidates early.

A new IDUL tree structure was also developed to quickly calculate the average utility and UBs of item sets using a recursive process. A novel algorithm named dHAUIM has been further presented to efficiently mine high average-utility item sets. An extensive experimental evaluation was carried out. Results have shown that dHAUIM outperforms four state-of-the-art HAUI mining algorithms in terms of execution time and number of join operations on both real-life and synthetic databases.

5.2. Bibliography

- [1] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," *ACM SIGMOD Rec.*, vol. 22, no. 2, pp. 207–216, Jun. 1993, <https://doi.org/10.1145/170036.170072>.
- [2] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation", *ACM SIGMOD Record*, vol. 29, no. 2, pp. 1–12, jun 2000, <https://doi.org/10.1007/s00354-008-0072-6>.
- [3] Z.-H. Deng and S.-L. Lv, "Fast mining frequent itemsets using nodesets," *Expert Systems with Applications*, vol. 41, no. 10, pp. 4505–4512, aug 2014, <https://doi.org/10.1016/j.eswa.2014.01.025>.
- [4] Z.-H. Deng, "DiffNodesets: An efficient structure for fast mining frequent itemsets," *Applied Soft Computing*, vol. 41, pp. 214–223, apr 2016, <https://doi.org/10.1016/j.asoc.2016.01.010>.
- [5] H. Yao, H. J. Hamilton, and C. J. Butz, "A foundational approach to mining itemset utilities from databases," in *Proceedings of the 2004 SIAM International Conference on Data Mining*. Society for Industrial and Applied Mathematics, apr 2004, pp. 482–486, <https://doi.org/10.1137/1.9781611972740.51>.
- [6] Y. Liu, W. keng Liao, and A. Choudhary, "A two-phase algorithm for fast discovery of high utility itemsets," in *Advances in Knowledge Discovery and Data Mining*. Springer Berlin Heidelberg, 2005, pp. 689–695, https://doi.org/10.1007/11430919_79.
- [7] T.-P. Hong, C.-H. Lee, and S.-L. Wang, "Effective utility mining with the measure of average utility," *Expert Systems with Applications*, vol. 38, no. 7, pp. 8259–8265, 2011, <https://doi.org/10.1016/j.eswa.2011.01.006>.
- [8] V. S. Tseng, B.-E. Shie, C.-W. Wu, and P. S. Yu, "Efficient algorithms for mining high utility itemsets from transactional databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 8, pp. 1772–1786, aug 2013, <https://doi.org/10.1109/tkde.2012.59>.

- [9] J. C. W. Lin, S. Ren, P. Fournier-Viger, and T. P. Hong, "Ehaupm: Efficient high average-utility pattern mining with tighter upper bounds," *IEEE Access*, vol. 5, pp. 12927–12 940, 2017, <https://doi.org/10.1109/access.2017.2717438>.
- [10] J. M.-T. Wu, J. C.-W. Lin, M. Pirouz, and P. Fournier-Viger, "TUB-HAUPM: Tighter upper bound for mining high average- utility patterns," *IEEE Access*, vol. 6, pp. 18655–18 669, 2018, <https://doi.org/10.1109/access.2018.2820740>.
- [11] M. Zihayat, H. Davoudi, and A. An, "Mining significant high utility gene regulation sequential patterns," *BMC System Biology*, vol. 11, no. Suppl 6, p. 109, 2017, <https://doi.org/10.1186/s12918-017-0475-4>.
- [12] U. Yun, D. Kim, E. Yoon, and H. Fujita, "Damped window based high average utility pattern mining over data streams," *Knowledge-Based Systems*, vol. 144, pp. 188–205, mar 2018, <https://doi.org/10.1016/j.knosys.2017.12.029>.
- [13] H. Ryang and U. Yun, "High utility pattern mining over data streams with sliding window technique," *Expert Systems with Applications*, vol. 57, pp. 214–231, September 2016, <https://doi.org/10.1016/j.eswa.2016.03.001>.
- [14] D. Kim and U. Yun, "Efficient algorithm for mining high average-utility itemsets in incremental transaction databases," *Applied Intelligence*, vol. 47, no. 1, pp. 114–131, 2017, <https://doi.org/10.1007/s10489-016-0890-z>. [Online]. Available: <https://doi.org/10.1007/s10489-016-0890-z>
- [15] I. Yildirim and M. Celik, "FIMHAUI: Fast incremental mining of high average-utility itemsets," in 2018 International Conference on Artificial Intelligence and Data Processing (IDAP). IEEE, sep 2018, <https://doi.org/10.1109/idap.2018.8620819>