**Attribute based encryption approach for storing, sharing and retrieval of encrypted data in the cloud.**

**Abstract**

One of the most cost-effective services in cloud computing is storage, used by businesses and individuals to outsource their massive data to untrusted servers. Efforts have studied problems around this application scenario in different fronts: efficiency, flexibility, reliability, and security. In this paper we address the security concerns of cloud storage under the scenario where users encrypt-then-outsource data, share their outsourced data with other users, and the service provider can be queried for searching and retrieval of encrypted data. As main distinctive, we propose a security approach for storage, sharing and retrieval of encrypted data in the cloud fully constructed on the basis of attribute-based encryption (ABE) thus enabling access control mechanisms over both the encrypted data and also for the information retrieval task through search access control. Compared to related works, our approach considers efficient encryption at three different levels: i) bulk encryption of data outsourced to the cloud, ii) keys management for access control over encrypted data by means of digital envelopes from attribute based encryption, and iii) novel construction for attribute based searchable encryption (ABSE). Our underlying ABE algorithms are carefully selected from the body of knowledge and novel constructions for ABSE are provided over the asymmetric setting (Type-III pairings) to support security levels of 128-bits or greater. Experimental results on benchmark data sets demonstrate the viability of our approach for practical realizations using Barreto-Naehrig curves.

**Introduction**

The high availability (access anytime, anywhere) and reliability of data at low cost are the main incentives for organizations and individuals to adopt cloud storage services. These services are in increasing demand due to the high amount of data generated by different sources (Internet of Things) and cloud enabled applications. However, data owners (DOs) outsourcing their data to untrusted servers in the cloud face the security concern that the cloud service provider (CSP) honestly stores the DO’s data and follows the agreed protocol but tries to learn as much as possible from the computations and interactions with the users (DU) that access the DO’s The associate editor coordinating the review of this manuscript and approving it for publication was Sun-Yuan Hsieh . data. This issue can be solved by providing DOs with a confidentiality security service for DOs to encrypt data before uploading it to the cloud. A straightforward encryption approach to prevent DO’s data disclosure and to keep DO’s data private from CSP or from any other entity, causes the provider cannot manipulate data, that is, loss of utility appears as the encrypted data cannot be used by the CSP for retrieval/searching purposes. Due to that inconvenience, DUs should download large volume of encrypted data, decrypt, and then search over the plaintext data (locally), re-encrypt and upload again its data to the cloud. Of course, so one approach incurs in huge communications and computations overhead and is completely inefficient. Searchable encryption (SE) [1] has been the most known approach to cope with the problem of searching over encrypted data stored in untrusted servers. SE is defined as the ability to identify and retrieve a set of objects from an encrypted collection that satisfy a query. In SE, the CSP executes DU’s encrypted queries over encrypted data without decryption, so it does not learn anything about the data content, search criteria, nor search patterns

**Purpose**

The purpose of the Attribute-Based Encryption (ABE) approach for storing, sharing, and retrieving encrypted data in the cloud is to provide a secure and flexible method of controlling access to sensitive data stored in cloud environments. ABE allows data owners to encrypt their data and define access policies based on attributes or characteristics of the users, rather than specifying individual user identities. This approach offers several benefits:

1. Fine-Grained Access Control: ABE enables fine-grained access control by allowing data owners to define access policies based on attributes such as user roles, organizational affiliations, or other attributes specific to the users. This provides more flexibility in managing access to sensitive data, as access decisions can be based on user attributes rather than specific user identities.

2. Data Confidentiality: ABE ensures data confidentiality by encrypting the data before storing it in the cloud. The encrypted data can only be decrypted and accessed by users who possess the corresponding access credentials or attributes specified in the access policies. This protects the data from unauthorized access or exposure.

3. Data Sharing and Collaboration: ABE enables secure data sharing and collaboration in cloud environments. Data owners can specify access policies that allow specific groups of users with the required attributes to access and decrypt the shared data. This facilitates secure collaboration and data sharing among authorized users while maintaining confidentiality.

4. Scalability and Flexibility: ABE provides scalability and flexibility in managing access control in cloud environments. Access policies can be easily modified or updated to accommodate changes in user attributes or organizational roles without requiring significant changes to the underlying encryption scheme. This flexibility allows for efficient management of access control as users join or leave an organization or as their attributes change over time.

5. Outsourced Data Storage: ABE supports secure data storage in the cloud, where data owners can encrypt their sensitive data before outsourcing it to third-party cloud service providers. The encrypted data remains protected even if the cloud provider is compromised, as only authorized users possessing the required attributes can decrypt and access the data.

6. Privacy Preservation: ABE helps preserve user privacy by eliminating the need to reveal specific user identities during access control decisions. Instead, access policies are defined based on user attributes, allowing users to maintain their anonymity while still accessing the encrypted data they are authorized to retrieve.

**Scope**

Scope of Attribute-Based Encryption (ABE) for Storing, Sharing, and Retrieval of Encrypted Data in the Cloud:

The scope of Attribute-Based Encryption (ABE) for storing, sharing, and retrieving encrypted data in the cloud encompasses various aspects related to data security, access control, and data sharing in cloud environments. The key areas within the scope can be described as follows:

1. Data Confidentiality: ABE ensures the confidentiality of sensitive data stored in the cloud. The scope includes encrypting the data using attribute-based keys, preventing unauthorized access and ensuring that only users with the required attributes can decrypt and access the data.

2. Fine-Grained Access Control: The scope includes providing fine-grained access control mechanisms based on user attributes. Data owners can define access policies specifying the attributes or attribute combinations required to access the encrypted data. This allows for flexible and granular control over who can access the data.

3. Secure Data Sharing: The scope encompasses secure data sharing capabilities. Data owners can selectively share encrypted data with specific attribute-based access requirements. Authorized users possessing the necessary attributes can securely retrieve and decrypt the shared data, promoting secure collaboration and controlled data sharing.

4. Access Policy Management: The scope includes providing tools and features for data owners to define, manage, and update access policies. Data owners can specify the attributes or attribute combinations required for data access, allowing for dynamic management of access control rules.

**Related work**

Attribute-based encryption (ABE) has become an enabler technology for secure storage and sharing in the cloud, as described in Section I-A. ABE allows many-to-many encryption, which is not possible to achieve in traditional public key cryptosystems (PKC), i.e., RSA. Thus, potential receivers may not necessarily be known at the time of encryption, a task that is done by using an access policy that enforces all those whose attributes satisfy the policy can decrypt and gain access to plaintext data. Atomically, ABE provides both confidentiality and fine-grained access control over data. Attributes are taken from a universe U. The access policy, expressed as an access structure A, restricts the decryption capabilities of the intended destinations depending on the attributes set possessed. The two prominent approaches for attribute-based encryption that have been proposed in the state-of-the-art are KP-ABE and CP-ABE. In KP-ABE [20], policies are associated to decryption keys and attributes to the ciphertext. Contrary, in CP-ABE [21] the ciphertext is created with a policy and decryption keys are associated to user’s attributes. CP-ABE is conceptually closer to Role Based Access Control and more natural to apply than is KP-ABE to enforce access control over encrypted data. Therfore, the cryptosystems based on CP-ABE are considered an attractive option for providing confidentiality service and fine-grained access control mechanisms at the same time [11]. Attribute-based searchable encryption (ABSE) is defined using either KP-ABE or CP-ABE as basis. Table 1 summarizes the main aspects of previous works proposing searchable encryption approaches for cloud-based storage systems. None of them use ABE for both access control over outsourced encrypted data neither for the information retrieval task (ABSE) as we propose in this work. It is relevant to note from Table 1 is that most of the reported works do not consider any key management mechanism M(k1) (as it is specified in the R1 requirement), and such a mechanism is crucial in practice, otherwise data could not be decrypted neither accessed by DUs. For the information retrieval task, the most common technique used has been SSE, either for single keyword search (SKS) or multi-keyword search (MKS). Furthermore, it is worth to mention that some solutions consider evaluation based on known benchmarks (IRB), which include a ranking mechanism (R3 requirement) to evaluate the effectiveness of the retrieval task. It is also worth to mention, from Table 1, that some works have carried out experiments (Exp) that have been limited only to the retrieval task (if R3 is checked) or only considering a specific security value in R4 for the encryption (R1) and search (R2) tasks.

.

**Need for system**

**Existing system**

There are several existing systems and frameworks that employ Attribute-Based Encryption (ABE) for storing, sharing, and retrieving encrypted data in the cloud. Here are a few examples:

1. CipherCloud: CipherCloud is a cloud security platform that offers Attribute-Based Encryption as one of its features. It provides data protection and encryption capabilities that allow organizations to securely store and share sensitive data in the cloud. CipherCloud's ABE solution enables fine-grained access control based on user attributes, ensuring data confidentiality and privacy.

2. Google Cloud Data Protection: Google Cloud offers a data protection suite that includes attribute-based access control features. It provides tools and services for encrypting data and defining access policies based on attributes such as user roles, organization affiliations, or custom-defined attributes. This allows organizations to securely store and share sensitive data in the Google Cloud platform.

3. Microsoft Azure Confidential Computing: Microsoft Azure Confidential Computing is a cloud service that supports secure data storage and processing using ABE techniques. It enables organizations to encrypt their data and enforce fine-grained access control based on user attributes or policies. Azure Confidential Computing provides a trusted execution environment to protect sensitive data while stored and processed in the cloud.

4. SecureSphere Data Protection: SecureSphere Data Protection is a data security solution offered by Imperva. It includes ABE capabilities for securing sensitive data stored in the cloud. The system allows organizations to define access policies based on user attributes and encrypt data accordingly. It provides granular control over data access, ensuring confidentiality and privacy.

5. Vormetric Data Security Platform: Vormetric Data Security Platform by Thales is a comprehensive data security solution that incorporates ABE for secure data storage and sharing. It supports encryption of sensitive data and allows access policies to be defined based on user attributes or roles. Vormetric's ABE capabilities help organizations protect their data in cloud environments.

**Proposed system**

Proposed System of Attribute-Based Encryption (ABE) for Storing, Sharing, and Retrieval of Encrypted Data in the Cloud:

Our proposed system aims to provide an efficient and secure solution for storing, sharing, and retrieving encrypted data in the cloud using Attribute-Based Encryption. The system incorporates the following components:

1. Data Encryption: The proposed system will encrypt sensitive data before storing it in the cloud. The encryption process will utilize Attribute-Based Encryption, where access policies are defined based on user attributes. The data will be encrypted using a symmetric key that is associated with specific attributes specified in the access policies.

2. Access Policy Management: The system will provide a user-friendly interface or API for data owners to define and manage access policies. Data owners can specify the required attributes or attribute combinations that authorized users must possess to decrypt and access the encrypted data. The access policies can be modified or updated as needed.

3. Key Management: The system will manage the generation, storage, and distribution of cryptographic keys required for attribute-based decryption. It will generate attribute-based keys that correspond to the attributes specified in the access policies. The keys will be securely stored and distributed to authorized users.

4. Secure Data Sharing: The proposed system will facilitate secure data sharing in the cloud. Authorized users possessing the required attributes can request access to the encrypted data. The system will verify the users' attributes against the access policies and provide the necessary attribute-based keys for decryption.

5. Encrypted Data Retrieval: The system will allow authorized users to retrieve the encrypted data from the cloud storage. Users will present their attribute-based keys and the system will perform attribute-based decryption to provide the decrypted data. Only users with the appropriate attributes specified in the access policies will be able to successfully decrypt and access the data.

**Software Requirement analysis and specification**

**Product perpective**

Product Perspective of Attribute-Based Encryption (ABE) for Storing, Sharing, and Retrieval of Encrypted Data in the Cloud:

The product perspective of Attribute-Based Encryption (ABE) for storing, sharing, and retrieving encrypted data in the cloud focuses on viewing the system as a valuable solution that provides enhanced security and flexibility for data storage and sharing in cloud environments. Here are key aspects of the product perspective:

1. Target Users: The product targets organizations and individuals who store sensitive data in the cloud and require a robust and flexible access control mechanism. This includes industries such as healthcare, finance, government, and any organization dealing with confidential or private data.

2. Integration with Cloud Services: The product perspective emphasizes seamless integration with popular cloud service providers and platforms. It should be compatible with major cloud infrastructure providers, allowing users to leverage their existing cloud storage and computing resources without significant modifications or disruptions.

3. User Interface and Experience: The product should provide a user-friendly interface that allows data owners to define access policies, manage attribute-based keys, and monitor access to encrypted data. The user experience should be intuitive and provide clear visibility into data sharing and access control mechanisms.

4. Data Encryption and Decryption: The product should offer robust encryption and decryption functionalities using Attribute-Based Encryption techniques. It should provide efficient encryption of sensitive data before storing it in the cloud and facilitate attribute-based decryption for authorized users based on the access policies defined by data owners.

**Product function**

Product Function of Attribute-Based Encryption (ABE) for Storing, Sharing, and Retrieval of Encrypted Data in the Cloud:

The product function of Attribute-Based Encryption (ABE) for storing, sharing, and retrieving encrypted data in the cloud revolves around providing secure and flexible access control mechanisms for sensitive data. The key functions of the product can be described as follows:

1. Data Encryption: The product function involves encrypting sensitive data before storing it in the cloud. It employs ABE techniques to encrypt the data using attribute-based keys. The encryption process ensures that the data remains confidential and protected from unauthorized access.

2. Access Policy Management: The product allows data owners to define and manage access policies for their encrypted data. Data owners can specify the required attributes or attribute combinations that authorized users must possess to decrypt and access the data. The access policy management function provides flexibility and granularity in controlling data access.

3. Attribute-Based Decryption: The product enables authorized users to decrypt and access the encrypted data based on their attributes or attribute-based keys. Users possessing the required attributes specified in the access policies can perform attribute-based decryption to retrieve the original data. This function ensures that only authorized users can access the encrypted data.

4. Secure Data Sharing: The product facilitates secure data sharing among authorized users in the cloud. It allows data owners to selectively share encrypted data with specific attribute-based access requirements. Authorized users possessing the necessary attributes can securely retrieve and decrypt the shared data, fostering collaboration and controlled data sharing.

5. Key Management: The product manages the generation, storage, and distribution of attribute-based keys. It generates attribute-based keys corresponding to the attributes specified in the access policies. These keys are securely stored and distributed to authorized users, ensuring that they possess the necessary credentials to decrypt the encrypted data.

**User characteristics**

User Characteristics of Attribute-Based Encryption (ABE) for Storing, Sharing, and Retrieval of Encrypted Data in the Cloud:

The user characteristics of Attribute-Based Encryption (ABE) for storing, sharing, and retrieving encrypted data in the cloud can vary depending on the roles and responsibilities within an organization. Here are some key user characteristics:

1. Data Owners: Data owners are individuals or organizations who have ownership or control over the sensitive data stored in the cloud. They define access policies and attribute requirements for accessing the encrypted data. Data owners possess the authority to manage access control rules and make decisions regarding data sharing and retrieval.

2. Authorized Users: Authorized users are individuals or entities who are granted permission to access the encrypted data based on their possession of the required attributes. These users have the necessary attributes specified in the access policies set by the data owners. They can securely retrieve and decrypt the data for authorized purposes.

3. IT Administrators: IT administrators are responsible for managing the technical aspects of the ABE system. They handle the implementation, configuration, and maintenance of the ABE infrastructure, including key management, access policy management, and system updates. IT administrators ensure the smooth operation and security of the ABE system.

4. Compliance Officers: Compliance officers are individuals who are responsible for ensuring that the organization adheres to applicable regulations and compliance requirements. They may be involved in defining the access policies and ensuring that the ABE system meets the necessary security standards and compliance obligations.

5. Cloud Service Providers: Cloud service providers play a crucial role in implementing the ABE system and providing the underlying infrastructure and storage services. They may offer specific APIs or integration points to support the ABE functionality. Cloud service providers ensure the availability, reliability, and security of the cloud environment where the encrypted data is stored.

**Constraints**

Constraints of Attribute-Based Encryption (ABE) for Storing, Sharing, and Retrieval of Encrypted Data in the Cloud:

While Attribute-Based Encryption (ABE) offers several benefits for secure data storage and sharing in the cloud, there are also certain constraints and challenges that need to be considered. These constraints may impact the implementation, performance, and usability of an ABE system. Some common constraints include:

1. Key Management Complexity: ABE requires the management of attribute-based keys, which can become complex, especially when dealing with a large number of attributes and users. Key management involves generating, storing, and distributing attribute-based keys securely. Handling key revocation and updates can also be challenging, especially in dynamic environments where attributes and access policies change frequently.

2. Computational Overhead: ABE operations can be computationally intensive, requiring significant processing power and resources. The encryption and decryption processes in ABE involve complex mathematical operations, resulting in increased computational overhead. This can affect system performance, especially when dealing with large volumes of data or a high number of users.

3. Scalability: Scaling an ABE system to handle a large number of attributes, users, and data can be challenging. As the number of attributes and users increases, the complexity and size of access policies and attribute-based keys also grow. Ensuring efficient and scalable key management, access control, and data retrieval becomes crucial to maintain system performance.

4. Attribute Granularity and Data Sensitivity: The granularity of attributes used in access policies should align with the sensitivity of the data being protected. Defining fine-grained attributes can increase the flexibility of access control, but it also increases the complexity and management overhead. Striking a balance between attribute granularity and practicality is important to ensure both security and usability.

**Functional Requirements**

Functional Requirements of Attribute-Based Encryption (ABE) for Storing, Sharing, and Retrieval of Encrypted Data in the Cloud:

The functional requirements of Attribute-Based Encryption (ABE) for storing, sharing, and retrieving encrypted data in the cloud define the key capabilities and features that the system should possess to effectively implement ABE. These requirements include:

1. Encryption and Decryption: The system should provide encryption and decryption functions using ABE techniques. It should support attribute-based encryption schemes that allow data owners to encrypt their sensitive data and authorized users to decrypt the data based on the specified access policies.

2. Access Policy Management: The system should offer tools and interfaces for data owners to define, manage, and update access policies. This includes specifying the required attributes or attribute combinations that authorized users must possess to access and decrypt the encrypted data.

3. Key Generation and Distribution: The system should generate attribute-based keys associated with the access policies. It should securely distribute these keys to authorized users based on their possession of the required attributes. Key distribution mechanisms should ensure confidentiality and integrity.

4. User Authentication and Authorization: The system should authenticate users and verify their possession of the required attributes before granting access to the encrypted data. It should enforce access control rules based on the defined access policies, allowing only authorized users to decrypt and access the data.

5. Secure Data Sharing: The system should facilitate secure data sharing among authorized users. It should provide mechanisms to selectively share encrypted data based on the access policies and attribute-based keys. This includes the ability to revoke or modify access privileges as needed.

**Performance Requirements**

Performance Requirements of Attribute-Based Encryption (ABE) for Storing, Sharing, and Retrieval of Encrypted Data in the Cloud:

The performance requirements of Attribute-Based Encryption (ABE) for storing, sharing, and retrieving encrypted data in the cloud outline the desired performance characteristics and benchmarks that the system should meet to ensure efficient and reliable operation. These requirements include:

1. Encryption and Decryption Speed: The system should provide fast encryption and decryption processes to ensure timely data protection and retrieval. Encryption should be performed efficiently to minimize the processing time required to encrypt the data before storing it in the cloud. Similarly, decryption should be performed quickly to enable authorized users to access the encrypted data in a timely manner.

2. Scalability: The system should be scalable to handle a growing number of users, attributes, and data volumes. It should be capable of efficiently managing and processing access policies, attribute-based keys, and data retrieval requests as the system scales. This ensures that the system can accommodate increasing demands without significant degradation in performance.

3. Attribute-Based Key Management: The system should efficiently manage attribute-based keys, including key generation, storage, and distribution. Key management operations should be performed in a manner that minimizes computational overhead and ensures quick and reliable key access for authorized users. Efficient key management contributes to overall system performance.

4. Data Retrieval Speed: The system should facilitate fast data retrieval for authorized users. Retrieving and decrypting the encrypted data based on the specified access policies and attribute-based keys should be performed efficiently to minimize the time taken to access the requested data. This ensures that authorized users can retrieve the required data promptly.

**SDLC methologies**

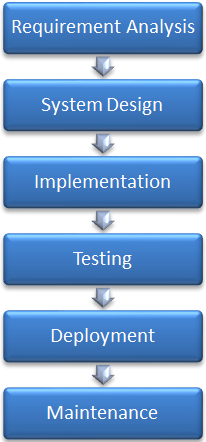
**SDLC METODOLOGIES**

### 1) Waterfall Model

[Waterfall Model](https://www.softwaretestinghelp.com/what-is-sdlc-waterfall-model/)also known as a linear sequential model is the traditional model in the Software development process. In this model, the next phase starts only when the previous one gets completed.

The output of one phase acts as the input for the next phase. This model does not support any changes to be done once it has reached the testing phase.

**The waterfall model follows the phases as shown below in a linear order.**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/12/Waterfall-Model.png)

**Advantages:**

* The waterfall model is a simple model.
* It is easily understood as all the phases are done step by step.
* No complexity as the deliverables of each phase are well defined.

**Disadvantages:**

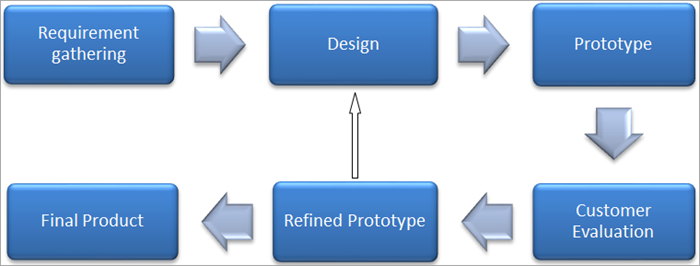
* This model cannot be used for the Project wherein the requirement is not clear or the requirement keeps on changing.
* A working model can only be available once the software reaches the last stage of the cycle.
* It is a time-consuming model.

### #2) Prototype Methodology

Prototype Methodology is the software development process in which a prototype is created before developing an actual product.

A prototype is demonstrated to a customer to evaluate the product if it is as per their expectation or if any changes are required. The refined prototype is created after the customer’s feedback and is again evaluated by the customer. This process goes on until the customer is satisfied.

Once the customer approves the prototype, the actual product is built by keeping the prototype as a reference.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/12/Prototype-Methodology-2-new.png)

**Advantages:**

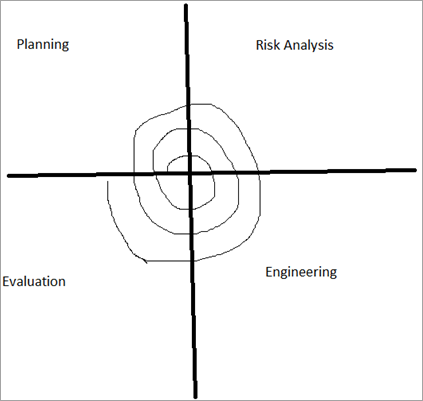
* Any missing feature or change in requirement can be easily accommodated in this model as it can be taken care of while creating a refined prototype.
* Reduces the cost and time of development as potential risks are identified in the prototype itself.
* As a customer is involved, it is easy to understand the requirement and any confusion can be easily sorted.

**Disadvantages:**

* As the customer is involved in every phase, the customer can change the requirement of the end product which increases the complexity of the scope and may increase the delivery time of the product.

### #3) Spiral Methodology

[Spiral Model](https://www.softwaretestinghelp.com/spiral-model-what-is-sdlc-spiral-model/) focuses mainly on risk identification. The developer identifies the potential risks and their solution is implemented. Later a prototype is created to verify risk coverage and check for other risks.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/12/Spiral-Methodology-3.png)

**Advantages:**

* Risk analysis done here reduces the scope of risk occurrence.
* Any requirement change can be accommodated in the next iteration.
* Model is good for large projects that are prone to risks and the requirement keeps on changing.

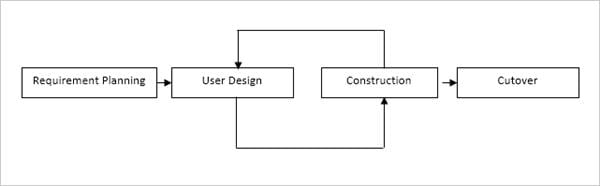
**Disadvantages:**

* The spiral model is best suited for large Projects only.
* The cost can be high as it might take a large no of iterations that can take high time to reach the final product.

### #4) Rapid Application Development

Rapid Application Development methodology helps to get high-quality results. It focuses more on the adaptive process than on planning. This methodology accelerates the entire development process and takes maximum advantage of developing software.

**Rapid Application Development divides the process into four phases:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/06/Rapid-Application-Development.jpg)

* **The requirement planning** phase combines the planning and analysis phase of the Software Development Lifecycle. Requirement gathering and analysis are done in this phase.
* **In the user design** phase, the user requirement is converted into a working model. A prototype is created as per the user requirement that represents all the system processes. In this phase, a user is constantly involved to get the model output as expected.
* **The construction** phase is the same as the development phase of SDLC. Since users are involved in this phase as well, they keep on suggesting any changes or improvements.
* **The cutover** Phase is similar to the implementation phase of SDLC including testing, and deployment. The new system built is delivered and goes live much sooner when compared to the other methodologies.

**Advantages:**

* It helps the customer to take a quick review of the project.
* A high-quality product is delivered as the users interact continuously with the evolving prototype.
* This model encourages feedback from a customer for improvement.

**Disadvantages**:

* This model cannot be used for small Projects.
* Requires experienced developers to handle complexities.

### #5) Rational Unified Process Methodology

Rational Unified Process Methodology follows the **Iterative software development** process. It is an Object-oriented and Web-enabled development methodology.

**RUP has four phases:**

1. Inception Phase
2. Elaboration Phase
3. Construction Phase
4. Transition Phase

**A brief description of each phase is given below.**

* **Inception Phase:** The scope of the project is defined.
* **Elaboration Phase:** Project requirements and their feasibility are done in-depth and the architecture of the same is defined.
* **Construction Phase:** Developers create a source code i.e. the actual product is developed in this phase. Also, the integrations with other services or existing software occur in this phase.
* **Transition Phase:** Product/application/system developed is delivered to the customer.

As RUP follows an iterative process, it provides a prototype at the end of each iteration. It emphasizes the development of components so that they can be used in the future as well. All the above four phases involve the workflows – Business Modeling, Requirement, Analysis and Design, Implementation, Testing, and Deployment.

* **Business Modeling**: In this workflow business context, the scope of the project is defined.
* **Requirement**: Here, the requirement of the product to be used in the entire development process is defined.
* **Analysis & Design**: Once the requirement is frozen, in the analysis & design phase, the requirement is analyzed i.e. the feasibility of the project is determined and then the requirement is transformed into a design.
* **Implementation**: The output of the design phase is used in the Implementation phase i.e. coding is done. Development of the Product takes place in this phase.
* **Testing**: Testing of the developed product takes place in this phase.
* **Deployment**: In this phase, the tested Product is deployed to the production environment.

**Advantages:**

* Adaptive to changing requirements.
* Focuses on accurate documentation.
* As the integration process goes through the development phase, it requires very little integration.

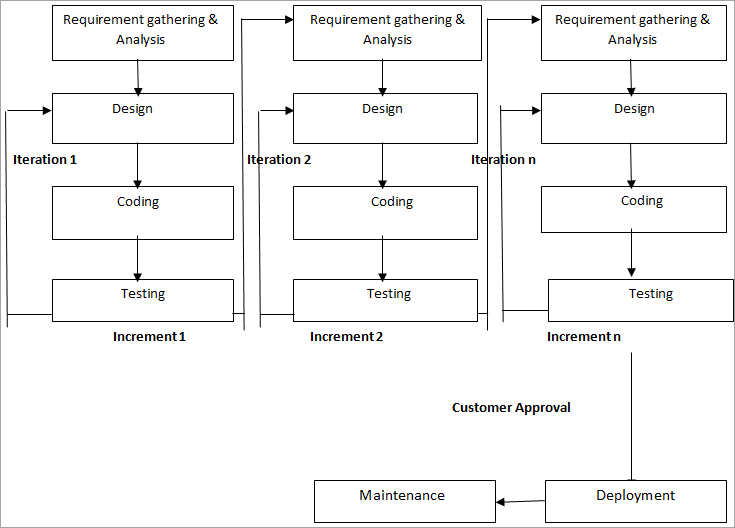
**Disadvantages:**

* RUP method requires highly experienced developers.
* As the integration is done throughout the development process, it might cause confusion as it can conflict in the testing phase.
* It is a complicated model.

### #6) Agile Software Development Methodology

[Agile Software Development](https://www.softwaretestinghelp.com/agile-scrum-methodology-for-development-and-testing/) methodology is an approach that is used to develop software in an iterative and incremental manner that allows frequent changes in the project. In agile, rather than focusing on requirements, the emphasis is on flexibility and an adaptive approach while developing a product.

**Example:** In agile, the team discusses the core features of the product and decides which feature can be taken up in the first iteration, and starts developing the same following the SDLC phases.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/12/Agile-Software-Development-Methodology-4.png)The next feature is taken up in the next iteration and is developed on the previously developed feature. Hence, a product is incremented in terms of features. After every iteration, the working product is delivered to the customer for their feedback and each iteration lasts for 2-4 weeks.

**Advantages:**

* Changes in requirements can be accommodated easily.
* Focus on flexibility and adaptive approach.
* Customer satisfaction as feedback and suggestions are taken at every stage.

**Disadvantages:**

* Lack of documentation as the focus is on the working model.
* Agile needs experienced and highly skilled resources.
* If a customer is not clear about what exactly they want the Product to be, then the project would fail.

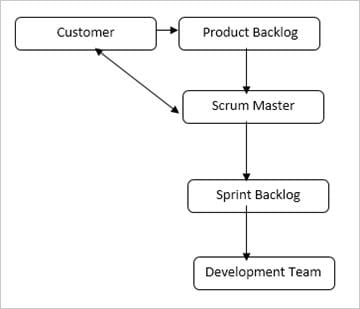
### #7) Scrum Development Methodology

Scrum is an iterative and incremental agile software development framework. It is a more time-boxed and planned method.

It is best suited for Projects in which requirements are not clear and keeps on changing rapidly. The scrum process includes planning, meeting & discussions, and reviews. Using this methodology helps in the fast development of the Project.

Scrum is organized by the Scrum Master, who helps to successfully deliver the Sprint goals. In scrum, the backlog is defined as the work to be done as a priority. The backlog items are completed in small sprints that last for2-4 weeks.

Scrum meeting is done on a daily basis to explain the progress of backlogs and to discuss possible obstacles.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/06/Scrum.jpg)

**Advantages:**

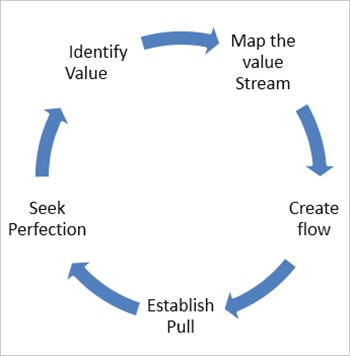
* Decision-making is completely in the hands of the team.
* The daily meeting helps the developer to know the productivity of individual team members thereby leading to improvement in productivity.

**Disadvantages:**

* Not Suitable for small-sized Projects.
* Needs highly experienced resources.

### #8) Lean Development Methodology

The lean development methodology is a method that is used in software development to decrease cost, effort, and waste. It helps in developing software one-third time when compared to the others that too within a limited budget and fewer resources.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/06/Lean-development-methodology.jpg)

* Identify value refers to the identification of products to be delivered at a specific time and cost.
* Mapping the value refers to the requirement of what is required to deliver the product to the customer.
* Creating flow refers to delivering a product to the customer on time as the customer needs it.
* Establish pull is establishing the product as per the customer’s needs only. It should be as per the customer’s requirement.
* Seek Perfection refers to delivering a product as expected by the customer within the time allocated and cost decided.

**Lean Development focuses on 7 principles as explained below:**

**Waste Elimination:**Anything that hinders the delivery of the product on time or reduces the quality of the product comes under waste. Unclear or inadequate requirements, coding delays, and insufficient testing comes under the causes of waste. The lean development method focuses on eliminating this waste.

**Amplifying Learning:**Amplify learning via learning the technologies required for the delivery of the product and understanding the requirement of the customer for what exactly they need. This can be achieved by taking feedback from the customer after every iteration.

**Late Decision Making:**It is better to make late decisions so that any change in the requirement can be accommodated with less cost. Taking early decisions while the requirement is uncertain leads to high costs as changes need to be done in all phases.

**Fast Delivery:**For fast delivery of the product or any change request or enhancement, an iterative development approach is used as it delivers the working model at the end of each iteration.

**Team Empowerment:**The team should be motivated and should be allowed to make their own commitments. Management should be supportive and should allow the team to explore and learn. The team should be helped to eliminate bad practices.

**Built-in Integrity:**The software is integrated to make sure it as a complete system it works well.

**View Application as a whole:**A product is developed in small iterations wherein the features are taken up to deliver. Different teams work on different aspects to deliver the product on time. The product as a whole should be optimized i.e. developer, tester, Customer, and Designer should work in an effective way to give the best results.

**Advantages:**

* Low Budget and efforts.
* Less time-consuming.
* Deliver the product very early when compared to the other methods.

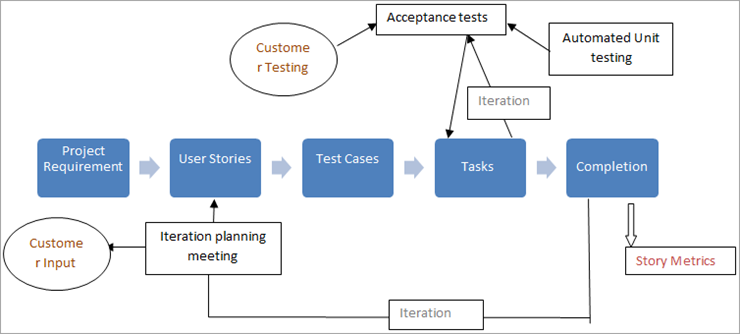
**Disadvantages:**

* The success of development depends entirely on the team’s decisions.
* As the developer is flexible to work, it can also lead to losing his focus.

### #9) Extreme Programming Methodology

Extreme Programming methodology is also known as XP methodology. This methodology is used to create software wherein the requirement is not stable. In the XP model, any change in requirement at the later stages leads to high costs for the Project.

This methodology requires more time and resources to complete the project when compared to the other methods. It focuses to reduce the cost of the software with continuous testing & planning. XP provides iterative and frequent releases throughout the SDLC phases of the Project.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/12/Extreme-Programming-Methodology-5.png)

**Core Practices of Extreme Methodology:**

**Fine-scale feedback**

* TDD (test-driven development)
* Pair Programming
* Planning game
* Whole team

**Continuous Process**

* Continuous Integration
* Design Improvement
* Small releases

**Shared understanding**

* Coding Standard
* Collective code ownership
* Simple Design
* System Metaphor

**Programmer welfare**

* Sustainable Pace

**Advantages:**

* Emphasis is on customer involvement.
* It delivers a high-quality product.

**Disadvantages:**

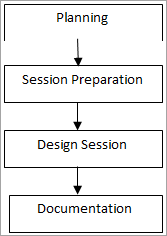
* This model requires meetings at frequent intervals which thereby increases the cost to customers.
* Development changes are too much to handle every time.

### #10) Joint Application Development Methodology

The joint application development methodology involves the developer, end-user, and clients for meetings and JAD sessions to finalize the software system to be developed. It accelerates the product development process and increases the developer’s productivity.

This methodology provides customer satisfaction as the customer is involved throughout the development phase.

**JAD Lifecycle:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/12/JAD-Lifecycle-7.png)

**Planning:**The very first thing in JAD is to select the executive sponsor. The planning stage includes selecting the executive sponsor, and team members for the definition stage, and defining the scope of the session. The deliverables from the definition stage can be completed by conducting a JAD session with high-level managers.

Once it is finalized that the project is to be taken, the executive sponsor and facilitator select the team for the Definition phase.

**Preparation:** The preparation phase includes preparation for conducting a kickoff meeting for the design sessions. Design sessions are conducted for the design team with an agenda.

This meeting is conducted by the executive sponsor wherein he explains the JAD process in detail. He takes up the concerns of the team and makes sure that members of the team are confident enough to work on Project.

**Design Sessions:**In the design session, the team should go through the Definition document to understand the requirement and Project scope. Later, the technique to be used for designing is finalized. The point of contact is finalized by the facilitator for the resolution of any issues/concerns.

**Documentation:**The documentation stage is completed when the sign-off on the design document is done. Based on the requirement in the document, the prototype is developed and another document is prepared for the deliverables to be given in the future.

**Advantages:**

* The quality of the Product is improved.
* Team productivity increases.
* Lowers the development and maintenance cost.

**Disadvantages:**

* Takes an excessive amount of time for planning and schedule.
* Requires significant investment of time and effort.

### #11) Dynamic System Development Model Methodology

Dynamic System Development methodology is based on the RAD method. It uses an iterative & incremental approach. DSDM is a simple model that follows best practices to be implemented in the project.

**Best Practices followed in DSDM:**

1. Active User Involvement.
2. The team must be empowered to make decisions.
3. The focus is on frequent delivery.
4. Fit for business purposes as the criteria for acceptance of Product.
5. The iterative and incremental development approach ensures that the right product is being created.
6. Reversible changes during development.
7. Requirements are baselined at a high level.
8. Integrated testing throughout the cycle.
9. Collaboration & co-operation between all the stakeholders.

**Techniques used in DSDM:**

**Timeboxing:**This technique is of 2-4 weeks of the interval. In exceptional cases, it goes up to 6 weeks also. A disadvantage of a longer interval is that the team can lose focus. At the end of the interval, the product has to be delivered. It can contain several tasks.

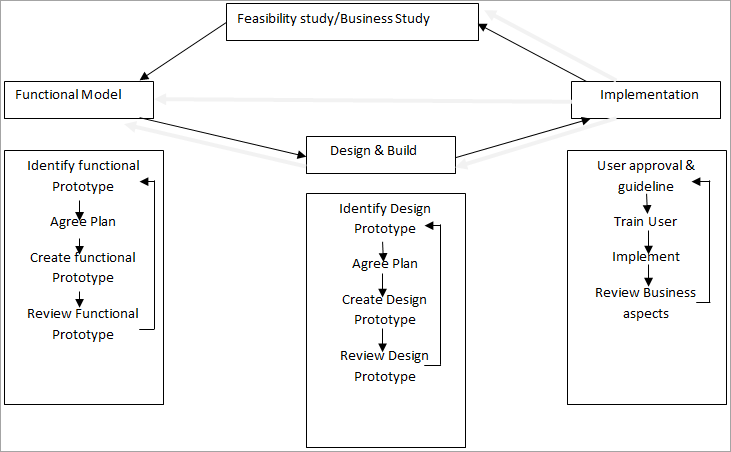
**MoSCoW :**

**It follows the below rule:**

* **Must-Have:** All the features defined should be delivered, or else the system would not work.
* **Should Have:** These features should be there in the product, but can be dropped in case of time constraints.
* **Could Have:** These features can be reassigned to a later time box.
* **Want to Have:** These features are not of much value.

**Prototyping**

The prototype is created first for the main functionality and then the other functionalities and features are implemented incrementally on the previous build.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2020/12/Prototyping-6.png)

**Advantages:**

* Iterative & Increment approach.
* Decision-making power to the team.

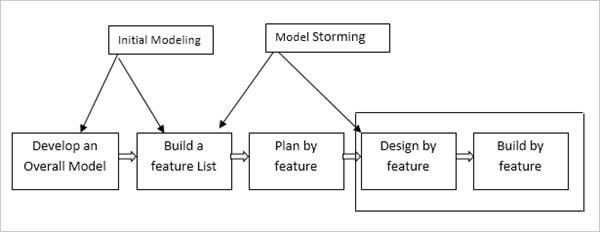
**Disadvantages:**

* Not good for small Organizations as this technique is costly to implement.

### #12) Feature-Driven Development

FDD also follows an iterative & incremental approach to delivering the working software. The feature is a small, client-valued function. **E.g.** “Validate the password of a user”. The project is divided into features.

**FDD has 5 process Steps:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/06/FDD.jpg)

**#1) Develop an Overall model:**An overall model which is basically a merge of detailed domain models is developed in this step. The model is developed by the developer wherein the customer is also involved.

**#2) Build a feature list:**In this step, the features list is prepared. The complete project is divided into features. Features to FDD have the same relation as user stories to scrum. A feature has to be delivered in two weeks of time.

**#3) The plan by feature:**Once the feature list is built, the next step is to decide the order in which the features should be implemented and who would be the owner of the feature i.e. teams are selected and features to be implemented are assigned to them.

**#4) Design by feature:**Features are designed in this step. The chief programmer selects the features to be designed in the time span of 2 weeks. Along with the feature owners, detailed sequence diagrams are drawn for each feature. Then the class and method prologues which are followed by the design inspection are written.

**#5) Build by feature:**Once the design inspection is successful, the owner of the class develops the code for their class. Code developed is unit tested & inspected. The chief programmer’s acceptance of the code is developed to let the complete feature be added to man build.

**Advantages:**

* Scalability of FDD to large projects.
* It is a simple methodology that can be easily adopted by companies.

**Disadvantages:**

* Not suitable for smaller projects.
* No written documentation is provided to the customer.

**SYSTEM REQUIREMENTS**

➢ **H/W System Configuration:-**

➢ Processor - Pentium –IV

➢ RAM - 4 GB (min)

➢ Hard Disk - 20 GB

➢ Key Board - Standard Windows Keyboard

➢ Mouse - Two or Three Button Mouse

➢ Monitor - SVGA

**Software Requirements:**

* Operating System - Windows XP
* Coding Language - Java/J2EE(JSP,Servlet)
* Front End - J2EE
* Back End - MySQL

# SYSTEM STUDY 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**

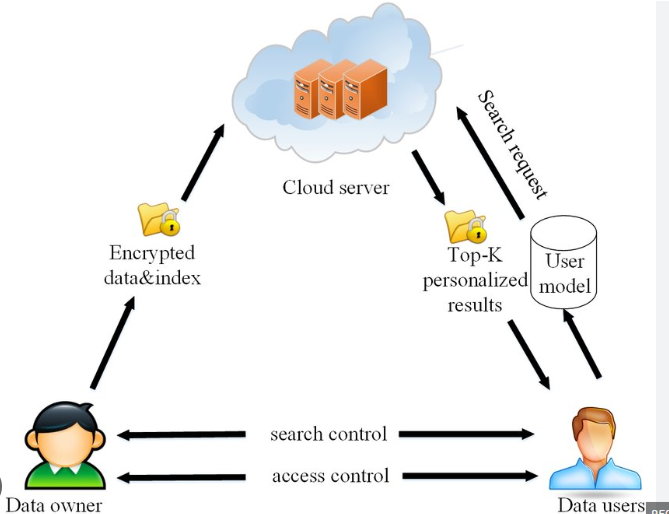
This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being 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**

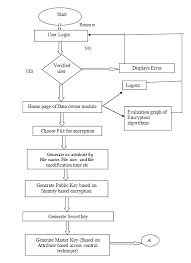
The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

# 4.SYSTEM DESIGN :

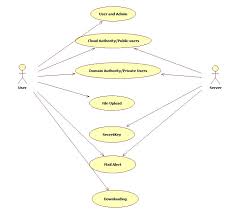
**SYSTEM ARCHITECTURE :**



**Flow chart**

****

**4.1 .UML DIAGRAMS :**

****

**The complex world of software development is now understandable by business users and those who want to recognize a system. Thanks to the standard visual language ‘UML.’**

Diagrams always attract people, and it is an effective way to brainstorm ideas, craft a plan, and communicate. Today a large number of researchers from virtually all areas sets at a unified forum known as “visual illustrations of a software or business model.”

## [A Programmer's Guide to Creating an Eclectic Bookshelf | Data Driven Investor](https://www.datadriveninvestor.com/2019/03/25/a-programmers-guide-to-creating-an-eclectic-bookshelf/" \t "_blank)

### [Every developer should have a bookshelf. The possible set of texts in his cabinet are myriad, but not every collection…](https://www.datadriveninvestor.com/2019/03/25/a-programmers-guide-to-creating-an-eclectic-bookshelf/" \t "_blank)

[www.datadriveninvestor.com](https://www.datadriveninvestor.com/2019/03/25/a-programmers-guide-to-creating-an-eclectic-bookshelf/" \t "_blank)

[Unified Modeling Language](https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/) (UML) diagram is a part of this unified forum which is used to specify errors and flaws, visualize, construct and document the objects of a business model or a software system.

There is so much to know about it; let’s dive deeper into the exciting realm of UML Diagrams.

# What is UML?

***UML is a way of imaging a software program or do an object-oriented analysis via graphical notations. It was invented to forge a standard, and semantically rich graphical modeling language for the design, architecture, and execution of complex software systems.***

Unified Modeling Language is not a programming language, but there are some tools which use UML diagrams to generate codes in different languages.

Today, UML is accepted as the standard language for modeling software systems by the Object Management Group (OMG).

It is much like the blueprints used in different fields and represents a collection of best engineering practices in the modeling of large and complex systems.

# What are UML Diagrams?

***UML diagrams are the graphical notations used to communicate, analyze, detect, and test the complexity, characteristics and other aspects of a system.***

Developers sketch UML diagrams before or after coding the application to get a better view of the system and as a form of documentation for various roles, activities, and workflows.

Through the diagram, many design issues or flaws can be revealed, which help improve the overall quality of the project.

# What are the types of UML Diagrams?

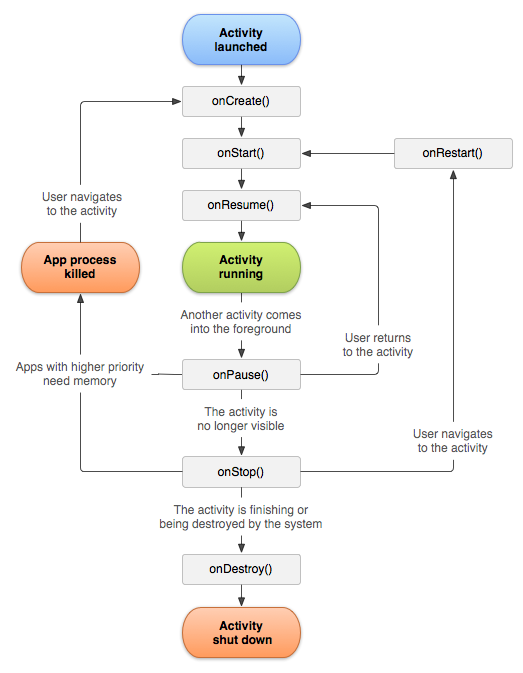
UML diagrams help developers and customers see a software system from a different perspective and in variable gradations of construction. There are [14 types of UML diagrams](https://www.smartdraw.com/uml-diagram/) divided into two broad classes and each one of them serves a different purpose.

Behavioral UML diagram and Structural UML diagrams are the two categories that encompass all other types.

# Behavioral UML Diagram

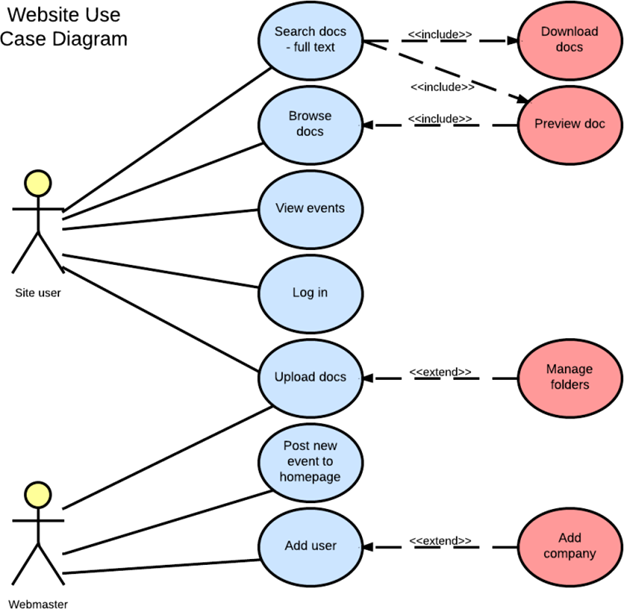
Seven types falls in this category and they are described as follows;

**Activity Diagram**



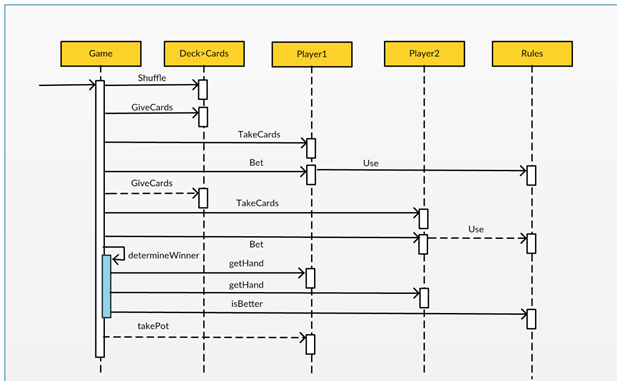
Activity Diagram displays the dynamic nature of a system by forming the flow of control from activity to activity. These diagrams are typically used to model a business process, workflow, and internal operation. This diagram concentrates on flows driven by internal processing.

**Use Case Diagram**



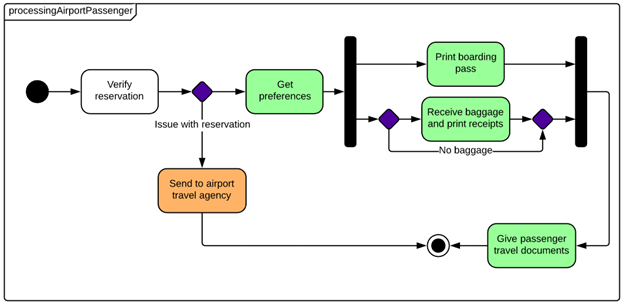
Use Case Diagram is a set of states that displays the interaction between a user and a system. It shows the relationship between use cases and actors.

**Sequence Diagram**



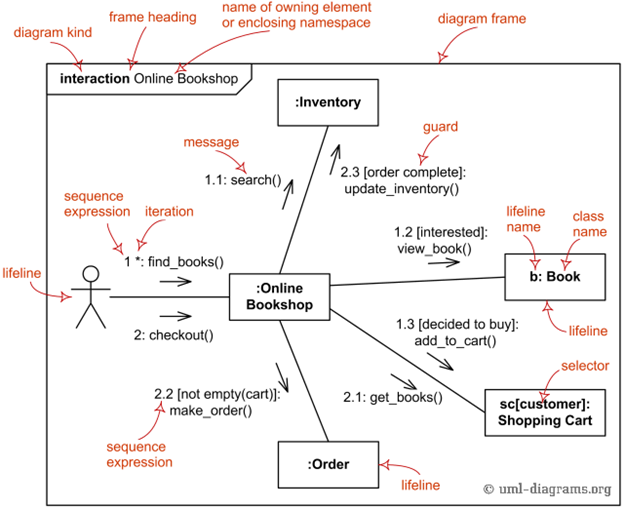
A sequence diagram is used to illustrate the time sequence of the objects participating in the interaction. The diagram consists of a vertical dimension representing time and horizontal dimension representing different objects.

**State diagram**



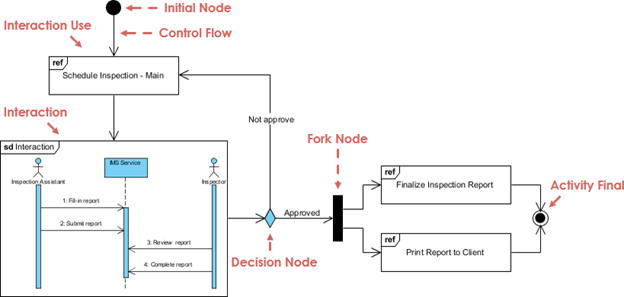
State Diagram displays the possible states that an object of interaction goes through when an event occurs. Each layout is used to understand the behavior of the object over the entire system.

**Communication Diagram**



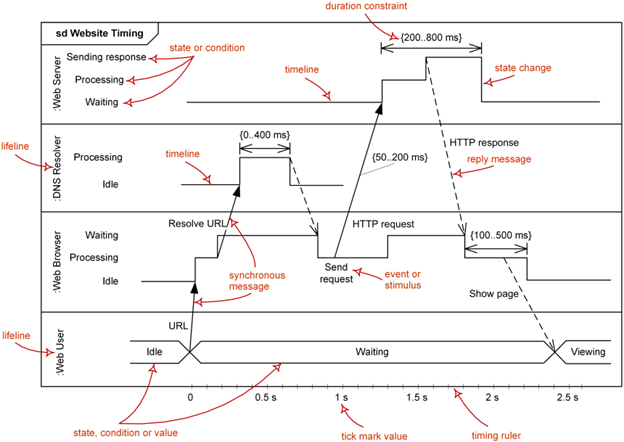
Communication UML diagrams are also called collaborative pictures because they focus on the messages that are exchanged between the objects. These diagrams can be drawn in the same way as a sequence diagram, but the only difference is that objects in communication diagrams are shown with association connections.

**Interaction overview diagram**



The interaction overview diagram is used to represent the dynamic behavior of the system. It is much like an activity diagram made of different interaction diagrams, and it is also known as a specialized activity diagram. You can use most annotations in an activity diagram, with the addition of elements like interaction, time constraint, duration, and alike.

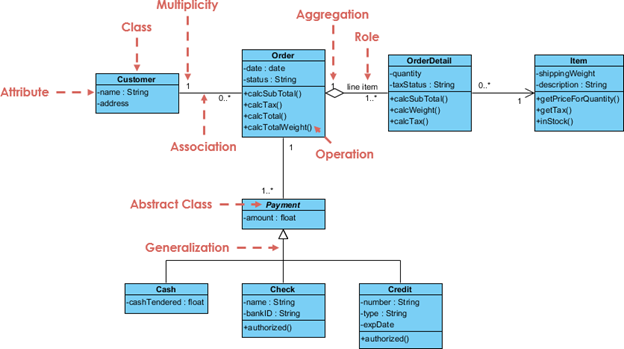
**Timing diagram**



Timing UML diagrams are used to epitomize the associations of objects when time is the epicenter. These diagrams represent how actors and objects act along a linear time axis. Lifeline, State timeline, Duration constraint, time constraint, and destruction occurrence are the main components of a timing UML diagram.

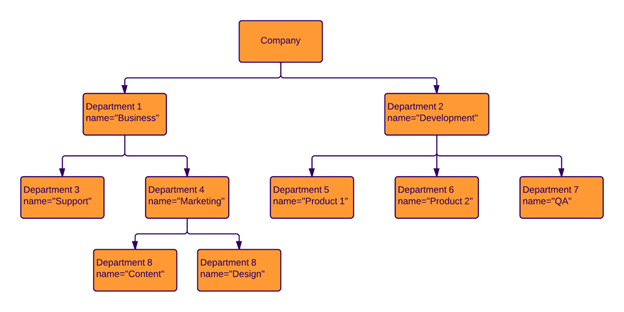
# **Structural UML Diagram**

**Class Diagram**



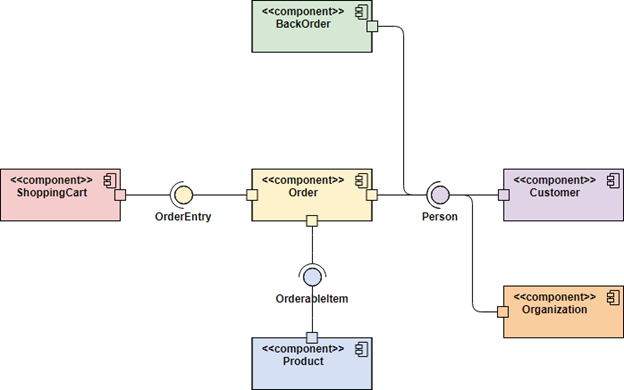
Class Diagram models are used to illustrate different types of objects and their relationships like inheritance, associations, containment using design elements such as packages, classes, and objectives.

**Object Diagram**



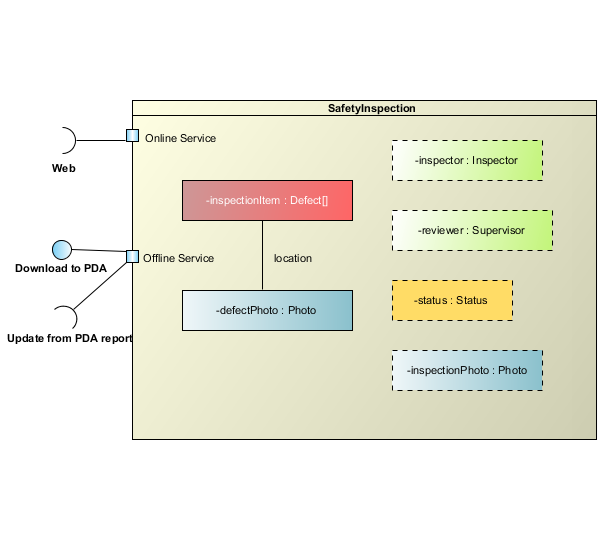
Object UML diagrams have more concerns with computer science-related concepts. In software development, objects are considered as instances of abstract class, and the classes are abstract data types. These diagrams help software developers to check whether the basic geometric structure (class diagram) they have created shows a feasible structure on implementation. Some developers take UML object diagrams as a secondary stage of accuracy checking.

**Component Diagram**



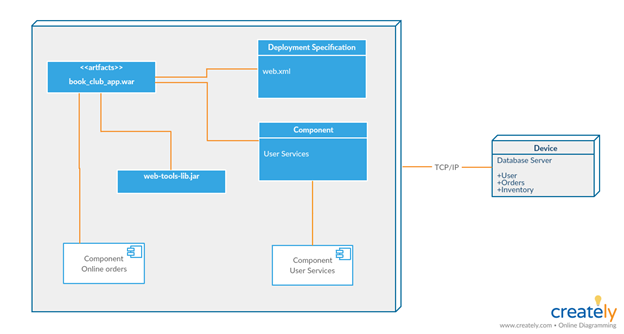
Component Diagram shows a high-level packaged structure of the code itself. These diagrams are used to illustrate dependencies among components, including binary code components, source code components, and executable components.

**Composite Structure Diagram**



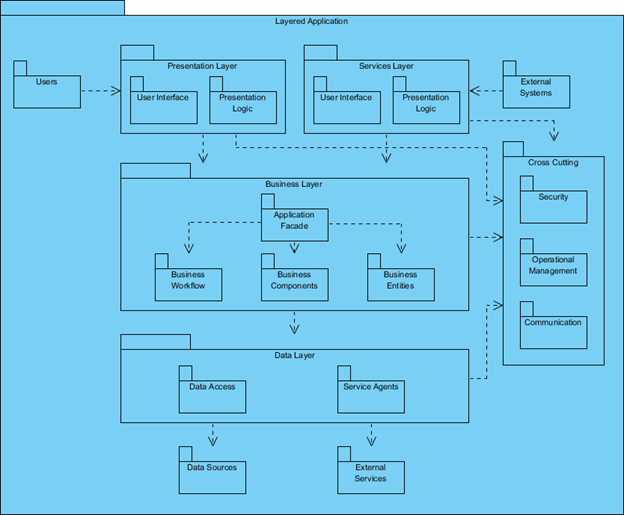
These diagrams only represent the relations between several class components and the internal structure of the class. Composite Structure UML Diagrams are not commonly used because of its limited function.

**Deployment Diagram**



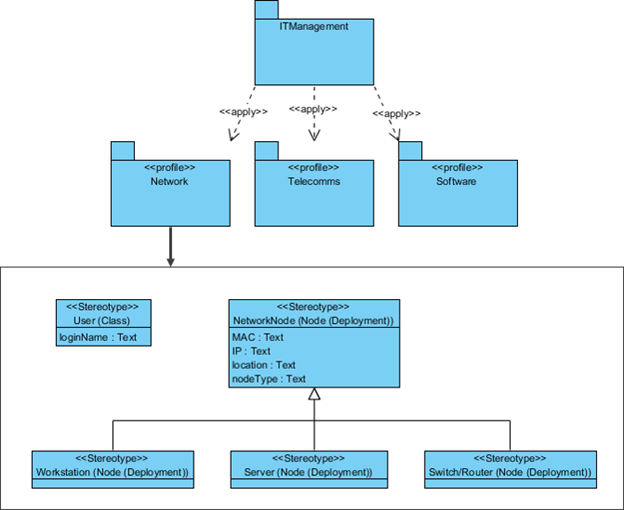
Deployment UML Diagram models the configuration of run-time processing elements, software components, processes and the objects that live on them. These diagrams display the physical relationship between software and hardware in a system.

**Package Diagram**



These diagrams are typically the subset of class diagrams, but sometimes developers use them as a separate technique. Package UML diagrams bring together the elements of a system into related groups to reduce dependencies between sets.

**Profile Diagram**



Profile diagram mostly not considered as a UML diagram because it is not a typical diagram. It is more of an extensibility mechanism rather than a diagram. These are like a language like if you speak English, you can create new sentences, similarly, if you talk about profile diagrams, you can create new semantics and properties for UML diagrams.

**Software Enivronment**

**OVERVIEW OF SOFTWARE DEVELOPMENT TOOLS**

**6.1 HTML**

Html is a language which is used to create web pages with html marking up a page to indicate its format, telling the web browser where you want a new line to begin or how you want text or images aligned and more are possible.

We used the following tags in our project.

**TABLE:**

Tables are so popular with web page authors is that they let you arrange the elements of a web page in such a way that the browser won’t rearrange them web page authors frequently use tables to structure web pages.

**<TR>:**

**<**TR**>** is used to create a row in a table encloses <TH> and <TD> elements. <TR> contain many attributes. Some of them are,

* ALIGN: specifies the horizontal alignment of the text in the table row.
* BGCOLOR: Specifies the background color for the row.
* BORDERCOLOR: Sets the external border color for the row.
* VALIGN: Sets the vertical alignment of the data in this row.

**<TH>:**

<TH> is used to create table heading.

* ALIGN: Sets the horizontal alignment of the content in the table cell. Sets LEFT, RIGHT, CENTER.
* BACKGROUND: Species the back ground image for the table cell.
* BGCOLOR: Specifies the background color of the table cell
* VALIGN: Sets the vertical alignment of the data. Sets to TOP, MIDDLE, BOTTOM or BASELINE.
* WIDTH: Specifies the width of the cell. Set to a pixel width or a percentage of the display area.

**<TD>:**

<TD> is used to create table data that appears in the cells of a table.

* ALIGN: Species the horizontal alignment of content in the table cell. Sets to LEFT, CENTER, RIGHT.
* BGCOLOR: Specifies the background image for the table cell.
* BGCOLOR: sets the background color of the table cells.
* WIDTH: Species the width of the cell

**FRAMES:**

Frames are used for either run off the page or display only small slices of what are supposed to be shown and to configure the frame we can use <FRAMESET>there are two important points to consider when working with <FRAMESET>.

* <FRAMESET> element actually takes the place of the <BODY> element in a document.
* Specifying actual pixel dimensions for frames.

<FRAME> Elements are used to create actual frames.

From the frameset point of view dividing the browser into tow vertical frames means creating two columns using the <FRAMESET> elements COLS attribute.

The syntax for vertical fragmentation is,

<FRAMESET COLS =”50%, 50%”>

</FRAMESET>

Similarly if we replace COLS with ROWS then we get horizontal fragmentation.

The syntax for horizontal fragmentation is,

<FRAMESET ROWS=”50%, 50%”>

</FRAMESET>

**FORM:**

The purpose of FORM is to create an HTML form; used to enclose HTML controls, like buttons and text fields.

**ATTRIBUTES:**

* ACTION: Gives the URL that will handle the form data.

* NAME: Gives the name to the form so you can reference it in code set to an alphanumeric string.

* METHOD: method or protocol is used to sending data to the target action URL. The GET method is the default, it is used to send all form name/value pair information in an URL. Using the POST method, the content of the form are encoded as with the GET method, but are sent in environment variables.

## 

## CONTROLS IN HTML

**<**INPUT TYPE =BUTTON>:

Creates an html button in a form.

ATTRIBUTES:

* NAME: gives the element a name. Set to alphanumeric characters.
* SIZE: sets the size.
* VALUE: sets the caption of the element.

**<**INPUT TYPE = PASSWORD>:

Creates a password text field, which makes typed input.

ATTRIBUTES:

* NAME: gives the element a name, set to alphanumeric characters.
* VALUE: sets the default content of the element.

<INPUT TYPE=RADIO>:

**C**reates a radio button in a form.

ATTRIBUTE:

* NAME: Gives the element a name. Set to alphanumeric character.
* VALUE: Sets the default content of the element.

<INPUT TYPE=SUBMIT>:

Creates a submit button that the user can click to send data in the form back to the web server.

ATTRIBUTES:

NAME: Gives the element a name. Set to alphanumeric characters.

VALUE: Gives this button another label besides the default, Submit Query. Set to alphanumeric characters.

**<**INPUT TYPE=TEXT>:

Creates a text field that the user can enter or edit text in.

ATTRIBUTES:

NAME: Gives the element a name. Set to alphanumeric characters.

VALUE: Holds the initial text in the text field. Set to alphanumeric characters.

**6.2 JAVA SCRIPT**

Java script originally supported by Netscape navigator is the most popular web scripting language today. Java script lets you embedded programs right in your web pages and run these programs using the web browser. You place these programs in a <SCRIPT> element, usually with in the <HEAD> element. If you want the script to write directly to the web page, place it in the <BODY> element.

**JAVASCRIPT METHODS:**

**Writeln:**

Document.writeln() is a method, which is used to write some text to the current web page.

**onClick:**

Occurs when an element is clicked.

**onLoad:**

Occurs when the page loads.

**onMouseDown:**

Occurs when a mouse button goes down.

**onMouseMove:**

Occurs when the mouse moves.

**onUnload:**

Occurs when a page is unloaded.

**6.3 MySQL**

The database has become an integral part of almost every human's life. Without it, many things we do would become very tedious, perhaps impossible tasks. Banks, universities, and libraries are three examples of organizations that depend heavily on some sort of database system. On the Internet, search engines, online gas booking[C:\Users\lnara\AppData\Local\Temp\ksohtml41728\wps1.jpg](http://www.devshed.com/), and even the website naming convention (http://www...) would be impossible without the use of a database. A database that is implemented and interfaced on a computer is often termed a database server.  
  
One of the fastest SQL (Structured Query Language) database servers currently on the market is the MySQL server.MySQL, available for download, offers the database programmer with an array of options and capabilities rarely seen in other database servers. What's more, MySQL is free of charge for those wishing to use it for private and commercial use. Those wishing to develop applications specifically using MySQL should consult MySQL's licensing section, as there is a charge for licensing the product.  
  
These capabilities range across a number of topics, including the following:

* Ability to handle an unlimited number of simultaneous users.
* Capacity to handle 50,000,000+ records.
* Very fast command execution, perhaps the fastest to be found on the market.
* Easy and efficient user privilege system.

A database is really nothing more than a hierarchy of increasingly complex data structures. In MySQL, the acknowledged structure for holding blocks (or **records**) of information is called the **table**.

These records, in turn, are made up of the smallest object that can be manipulated by the user, known as the **data type**. Together, one or more of these data types form a record. A table holds the collection of records that make up part of the database. We can consider the hierarchy of a database to be that of the following:

Database < Table < Record < Datatype

Datatypes come in several forms and sizes, allowing the programmer to create tables suited for the scope of the project. The decisions made in choosing proper data types greatly influence the performance of a database, so it is wise to have a detailed understanding of these concepts.

**MySQL Data types**

MySQL is capable of many of the data types that even the novice programmer has probably already been exposed to. Some of the more commonly used include:

**CHAR (M)**  
CHAR's are used to represent fixed length strings. A CHAR string can range from 1-255 characters. In later table creation, an example CHAR data type would be declared as follows:

ex.  
car model CHAR(10);

**VARCHAR (M)**  
VARCHAR is a more flexible form of the CHAR data type. It also represents data of type String, yet stores this data in variable length format. Again, VARCHAR can hold 1-255 characters. VARCHAR is usually a wiser choice than CHAR, due to it's variable length format characteristic. Although, keep in mind that CHAR is much faster than VARCHAR, sometimes up to 50%.  
(A CHAR stores the whole length of the declared variable, regardless of the size of the data contained within, whereas a VARCHAR only stores the length of the data, thus reducing size of the database file.)

ex.  
car model VARCHAR(10);

**INT (M) [Unsigned]**  
The INT data type stores integers ranging from -2147483648 to 2147483647. An optional "unsigned" can be denoted with the declaration, modifying the range to be 0 to 4294967295

ex.  
light-years INT;  
Valid integer: '-24567'. Invalid integer: '3000000000'.

ex.  
light-years INT unsigned;  
Valid integer: '3000000000'. Invalid integer: '-24567'.

**FLOAT [(M,D)]**  
A FLOAT represents small decimal numbers, used when a somewhat more precise representation of a number is required.

ex.  
rainfall FLOAT (4,2);  
This could be used to represent rainfall average in centimeters per year, which could be a decimal value. More specifically, FLOAT (4,2) states the fact that rainfall can hold up to four characters and two decimal places. Thus,

42.35 is valid, accurately represented.  
324.45 is invalid, rounded to 324.5.  
2.2 is valid, accurately represented.  
34.542 is invalid, rounded to 34.54.

*Note: Due to the fact that FLOAT is rounded, those wishing to represent money values would find it wise to use* ***DECIMAL****, a datatype found within MySQL that does not round values. Consult the documentation for a complete explanation.*

**DATE**   
Stores date related information. The default format is 'YYYY-MM-DD', and ranges from '0000-00-00' to '9999-12-31'. MySQL provides a powerful set of date formatting and manipulation commands, too numerous to be covered within this article. However, one can find these functions covered in detail within the MySQL documentation.

the\_date DATE;

**TEXT / BLOB**  
The text and blob datatypes are used when a string of 255 - 65535 characters is required to be stored. This is useful when one would need to store an article such as the one you are reading. However, there is no end space truncation as with VARCHAR AND CHAR. The only difference between BLOB and TEXT is that TEXT is compared case insensitively, while BLOB is compared case sensitively.

**SET**  
A datatype of type string that allows one to choose from a designated set of values, be it one value or several values. One can designate up to 64 values.

ex.  
transport SET ("truck", "wagon") NOT NULL;

From the above declaration, the following values can be held by transport:

""  
"truck"  
"wagon"  
"truck,wagon"

**ENUM**  
A datatype of type string that has the same characteristics as the SET datatype, but only one set of allowed values may be chosen. Usually only takes up one byte of space, thus saving time and space within a table.

ex.  
transport ENUM ("truck", "wagon") NOT NULL;

From the above declaration, the following values can be held by transport:

""  
"truck"  
"wagon"

**Records**

Together, a group of declared datatypes form what is known as a record. A record can be as small as one data variable, or as many as deemed needed. One or more records form the structure of a table.

**The Bigger Picture: Tables**

Before we can execute commands on the database, we must first create a table in which data can be stored. This is accomplished in the following manner:

mysql> CREATE TABLE test (  
> name VARCHAR (15),  
> email VARCHAR (25),  
> phone\_number INT,  
> ID INT NOT NULL AUTO\_INCREMENT,  
> PRIMARY KEY (ID));

Ensuing output:

Query OK, 0 rows affected (0.10 sec)  
mysql>

The first table in your database has now been created. *Note: no two tables can have the same name.*  
*Note(2): Each dataspace is more often referred to as a* ***column****.*

**Column Characteristics:**

* A name may not be made up of strictly numbers.
* A name may start with a number.
* A name may be up to 64 characters.

**Other table options:**

The following options can be placed after any datatype, adding other characteristics and capabilities to them.

* Primary Key. Used to differentiate one record from another. No two records can have the same primary key. This is obviously useful when it is imperative that no two records are mistaken to be the other.
* Auto\_Increment. A column with this function is automatically incremented one value (previous + 1) when an insertion is made into the record. The datatype is automatically incremented when 'NULL' is inserted into the column.
* NOT NULL. Signifies that the column can never be assigned a NULL value.

ex.  
soc\_sec\_number INT PRIMARY KEY;  
No two soc\_sec\_number records can hold the same value.

ID\_NUMBER INT AUTO\_INCREMENT;

**Insertion of records**

**Note:** The originally created table, test, created in the last section will be used to illustrate the examples in this section. Here it is again, for quick reference:

mysql> CREATE TABLE test (

> name VARCHAR (15),

> email VARCHAR (25),

> phone\_number INT,

> ID INT NOT NULL AUTO\_INCREMENT,

> PRIMARY KEY (ID));

Insertion of data into the table is accomplished, logically enough, using the INSERT command.

mysql> INSERT INTO test VALUES

mysql> ('Bugs Bunny', 'carrots@devshed.com',

mysql> 5554321, NULL);

Result, assuming the command was correctly entered:

Query OK, 1 row affected (0.02 sec)

mysql>

**Selection**

A database would not be much use if one was not able to search and extract data from it. In MySql terms, this is accomplished through the SELECT statement.

mysql> SELECT \* FROM test

mysql> WHERE (name = "Bugs Bunny");

Result:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **email** | **phone** | **ID** |
| Bugs Bunny | carrots@devshed.com | 5554321 | 1 |

Let's assume we have inserted four differing records, all bearing the same name of "Bugs Bunny", yet having different email addresses and phone numbers. The table test, would look somewhat like the following:

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **email** | **phone** | **ID** |
| Bugs Bunny | carrots@devshed.com | 5554321 | 1 |
| Bugs Bunny | peppers@devshed.com | 5554331 | 2 |
| Bugs Bunny | lettuce@devshed.com | 5554341 | 3 |
| Bugs Bunny | celery@devshed.com | 5554351 | 4 |

**Deletion**

One can also delete records inserted into the table. This is accomplished through the DELETE command.

mysql> DELETE FROM test

mysql> WHERE (name = "Bugs Bunny");

Result:  
This would result in the deletion of all records within the table test containing name "Bugs Bunny".

Another example:

mysql> DELETE FROM test

mysql> WHERE (phone\_number = 5554321);

Result: (Using the previously illustrated example)

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **email** | **phone** | **ID** |
| Bugs Bunny | peppers@devshed.com | 5554331 | 2 |
| Bugs Bunny | lettuce@devshed.com | 5554341 | 3 |
| Bugs Bunny | celery@devshed.com | 5554351 | 4 |

**Modification**

MySQL also has the capability of modifying data already entered into the table. This is accomplished through the UPDATE command.

mysql> UPDATE test SET name = 'Daffy Duck'

mysql> WHERE name = "Bugs Bunny";

|  |  |  |  |
| --- | --- | --- | --- |
| **name** | **email** | **phone** | **ID** |
| Daffy Duck | peppers@devshed.com | 5554331 | 2 |
| Daffy Duck | lettuce@devshed.com | 5554341 | 3 |
| Daffy Duck | celery@devshed.com | 5554351 | 4 |

This section, we covered the core [MySQL databaseC:\Users\lnara\AppData\Local\Temp\ksohtml41728\wps2.jpg](http://www.devshed.com/) manipulation functions, basic insertion, deletion, modification, and search. The next section will elaborate on these capabilities, providing extended functioning and flexibility when manipulating the database.

What we have covered so far is but a small part of what MySQL is capable of. Let's delve a little deeper into the language, exploring some of the more advanced commands of the language.

**Logical Operations**

MySQL includes full support of all basic logical operations.

**AND (&&)**

mysql> SELECT \* FROM test WHERE

mysql> (name = "Bugs Bunny") AND

mysql> (phone\_number = 5554321);

Result:  
All records containing the name "Bugs Bunny" AND the phone number '5554321' will be displayed to the screen.

**OR ( || )**

mysql> SELECT \* FROM test WHERE

mysql> (name = "Bugs Bunny") OR

mysql> (phone\_number = 5554321);

Result:  
All records containing the name "Bugs Bunny" OR the phone number '5554321' will be displayed to the screen.

**NOT ( ! )**

mysql> SELECT \* FROM test WHERE

mysql> (name != "Bugs Bunny");

Result:  
All records NOT containing the name "Bugs Bunny" will be displayed to the screen.

**Order By**

mysql> SELECT \* FROM test WHERE

mysql> (name = "Bugs Bunny") ORDER BY

mysql> phone\_number;

Result:  
All records containing the name "Bugs Bunny" will be displayed to the screen, ordered in respect to the phone\_number.

**Search functions**

MySQL offers the user the ability to perform both general and specific searches on data.

mysql> SELECT \* FROM test WHERE

mysql> (name LIKE "%gs Bunny");

Result:  
All records containing the partial string "gs Bunny" will be displayed to the screen. This would include such names as: "Bugs Bunny", "ags Bunny", "gs Bunny", and "234rtgs Bunny".

Notice that "LIKE" has been used instead of the equals sign (=). "LIKE" signifies that one is searching for an estimate of the data requested, and not necessarily an exact copy.

The '%' sign could be placed anywhere within the string. The method in which the server searches for a string is dependent upon where one places the '%' sign.

mysql> SELECT \* FROM test WHERE

mysql> (name LIKE "Bugs Bunny%");

Result:  
All records containing the partial string "Bugs Bunny" will be displayed to the screen. This would include such names as: "Bugs Bunnys", "Bugs Bunnyyyy453", "Bugs Bunnytrtrtrtrtr", but not "gs Bunny".

**Focused Search Results**

One can also perform searches and display only certain columns.

mysql> SELECT name FROM test WHERE

mysql> (name = "Bugs Bunny");

Result:

|  |
| --- |
| **name** |
| Bugs Bunny |

**Alter table**

Another very important function of MySQL is the ability to modify previously created tables. This is accomplished via the ALTER statement. This function allows one to add, modify, and delete columns, as well as rename the table, among other functions.

Example: Rename the table

mysql> ALTER table test RENAME mytest;

Example: Add a column

mysql> ALTER table mytest ADD birthday DATE;

Example: Modify a column

mysql> ALTER table mytest CHANGE

mysql> name newname VARCHAR (25);

Example: Delete a column

mysql> ALTER table mytest DROP newname;

Executing the above four functions would modify test, creating the following table:

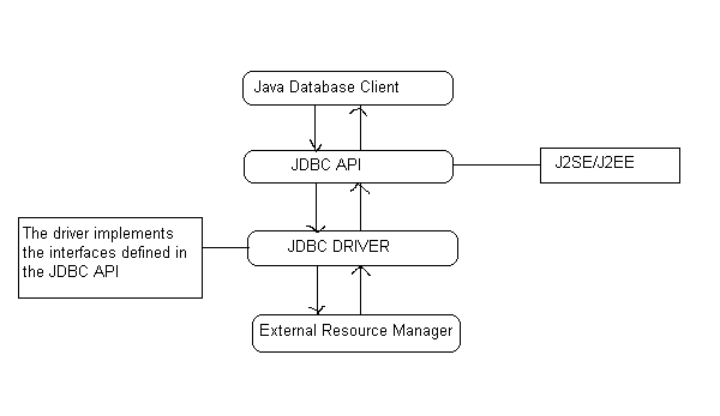
mysql> TABLE mytest (

> email VARCHAR (25),

> phone\_number INT,

> ID INT AUTO\_INCREMENT,

> birthday DATE );



**DRIVER MANAGER AND DRIVER:**

The java.sql package defines an interface called Java.sql.Driver that makes to be implemented by all the JDBC drivers and a class called java.sql.DriverManager that acts as the interface to the database clients for performing tasks like connecting to external resource managers, and setting log streams. When a JDBC client requests the DriverManager to make a connection to an external resource manager, it delegates the task to an approate driver class implemented by the JDBC driver provided either by the resource manager vendor or a third party.

**JAVA.SQL.DRIVERMANAGER:**

The primary task of the class driver manager is to manage the various JDBC drivers register. It also provides methods for:

* Getting connections to the databases.
* Managing JDBC logs.
* Setting login timeout.

**MANAGING DRIVERS:**

JDBC clients specify the JDBC URL when they request a connection. The driver manager can find a driver that matches the request URL from the list of register drivers and delegate the connection request to that driver if it finds a match JDBC URLs normally take the following format:

**<protocol>:<sub-protocol>:<resource>**

The protocol is always jdbc and the sub-protocol and resource depend on the type of resource manager. The URL for postgreSQL is in the format:

**Jdbc: postgres ://< host> :< port>/<database>**

Here host is the host address on which post master is running and database is the name of the database to which the client wishes to connect.

**MANAGING CONNECTION:**

DriverManager class is responsible for managing connections to the databases:

public static Connection getConnection (String url,Properties info) throws SQLException

This method gets a connection to the database by the specified JDBC URL using the specified username and password. This method throws an instance of SQLException if a database access error occurs.

**CONNECTIONS:**

The interface java.sql.Connection defines the methods required for a persistent

connection to the database. The JDBC driver vendor implements this interface. A database ‘vendor-neutral’ client never uses the implementation class and will always use only the interface. This interface defines methods for the following tasks:

* Statements, prepared statements, and callable statements are the different types of statements for issuing sql statements to the database by the JDBC clients.
* For getting and setting auto-commit mode.
* Getting meta information about the database.
* Committing and rolling back transactions.

**CREATING STATEMENTS:**

The interface java.sql.Connection defines a set of methods for creating database statements. Database statements are used for sending SQL statements to the database:

Public Statement createStatement () throws SQLException

This method is used for creating instances of the interface java.sql.Statement. This interface can be used for sending SQL statements to the database. The interface java.sql.Statement is normally used for sending SQL statements that don’t take any arguments. This method throws an instance of SQLException if a database access error occur:

Public Statement createStatement (int resType, int resConcurrency) throws SQLException

**JDBC RESULTSETS:**

A JDBC resultset represents a two dimentional array of data produced as a result of executing SQL SELECT statements against databases using JDBC statements. JDBC resultsets are represented by the interface java.sql.ResultSet. The JDBC vendor provider provides the implementation class for this interface.

**SCROLLING RESULTSETS:**

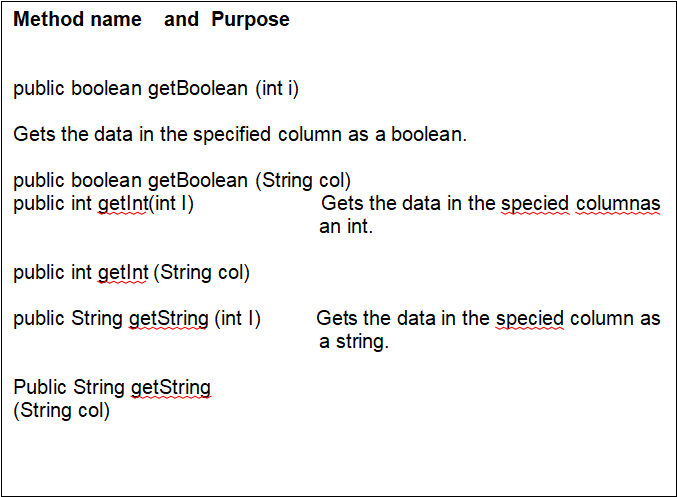
public boolean next() throws SQLException

public boolean previous() throws SQLException

public boolean first() throws SQLException

public boolean last() throws SQLException

**ACCESSING RESULTSET DATA:**



C:\Users\lnara\AppData\Local\Temp\ksohtml41728\wps5.png

**STATEMENT:**

The interface java.sql.Stament is normally used for sending SQL statements that do not have IN or OUT parameters. The JDBC driver vendor provides the implementation class for this interface. The common methods required by the different JDBC statements are defined in this interface. The methods defined by java.sql. Statement can be broadly categorized as follows:

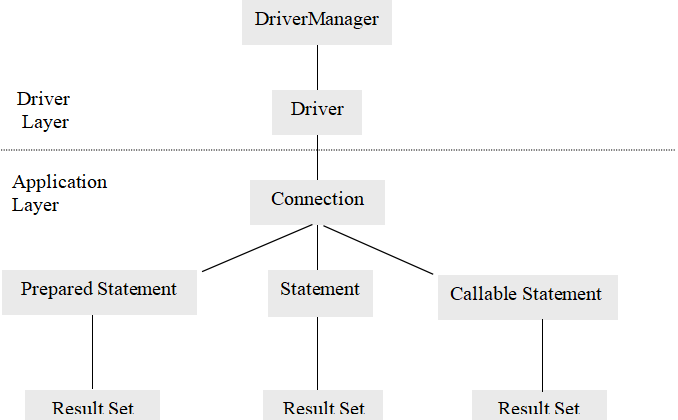
* Executing SQL statements
* Querying results and resultsets
* Handling SQL batches
* Other miscellaneous methods

The interface java.sql.statements defines

methods for executing different SQL statements like SELECT, UPDATE, INSERT, DELETE, and CREATE.

Public Resultset execute Query (string sql) throws SQLException

The following figure shows how the DriverManager, Driver, Connection, Statement, ResultSet classes are connected.



###### 6.4 JAVA SERVER PAGES (JSP)

**INTRODUCTION:**

Java Server Pages (JSP) technology enables you to mix regular, static HTML with dynamically generated content. You simply write the regular HTML in the normal manner, using familiar Web-page-building tools. You then enclose the code for the dynamic parts in special tags, most of which start with <% and end with %>.

**THE NEED FOR JSP:**

Servlets are indeed useful, and JSP by no means makes them obsolete. However,

* It is hard to write and maintain the HTML.
* You cannot use standard HTML tools.
* The HTML is inaccessible to non-Java developers.

**BENEFITS OF JSP:**

JSP provides the following benefits over servlets alone:

* It is easier to write and maintain the HTML: In this no extra backslashes, no double quotes, and no lurking Java syntax.
* You can use standard Web-site development tools:

We use Macromedia Dreamweaver for most of the JSP pages. Even HTML tools that know nothing about JSP can used because they simply ignore the JSP tags.

* You can divide up your development team:

The Java programmers can work on the dynamic code. The Web developers can concatenate on the representation layer. On large projects, this division is very important. Depending on the size of your team and the complexity of your project, you can enforce a weaker or stronger separation between the static HTML and the dynamic content.

**CREATING TEMPLATE TEXT:**

A large percentage of our JSP document consists of static text known as template text. In almost all respects, this HTML looks just likes normal HTML follows all the same syntax rules, and simply “passed through” to that client by the servlet created to handle the page. Not only does the HTML look normal, it can be created by whatever tools you already are using for building Web pages.

There are two minor exceptions to the “template text passed through” rule. First, if you want to have <% 0r %> in the out port, you need to put <\% or %\> in the template text. Second, if you want a common to appear in the JSP page but not in the resultant document,

<%-- JSP Comment -- %>

HTML comments of the form:

<!—HTML Comment -->

are passed through to the client normally.

**TYPES OF JSP SCRIPTING ELEMENTS:**

JSP scripting elements allow you to insert Java code into the servlet that will be generated from the JSP page. There are three forms:

1. **Expressions** of the form <%=Java Expression %>, which are evaluated and inserted into the servlet’s output.
2. **Sciptlets** of the form <%Java code %>, which are inserted into the servlet’s\_jspService method (called by service).
3. **Declarations** of the form<%! Field/Method Declaration %>, which are inserted into the body of the servlet class, outside any existing methods.

**USING JSP EXPRESSIONS:**

A JSP element is used to insert values directly into the output. It has the following form:

<%= Java Expression %>

The expression is evaluated, converted to a string, and inserted in the page. This evaluation is performed at runtime (when the page is requested) and thus has full access to the information about the request. For example, the following shows the date/time that the page was requested.

Current time: <%=new java.util.Date () %>

**PREDEFINED VARIABLES:**

To simplify expressions we can use a number of predefined variables (or “implicit objects”). The specialty of these variables is that, the system simple tells what names it will use for the local variables in \_jspService.The most important ones of these are:

* **request**, the HttpServletRequest.
* **response**, the HttpServletResponse.
* **session,** the HttpSession associated with the request
* **out,** the writer used to send output to clients.
* **application,** the ServletContext. This is a data structure shared by all servlets and JSP pages in the web application and is good for storing shared data.

Here is an example:

Your hostname: <%= **request.**getRemoteHost () %>

**COMPARING SERVLETS TO JSP PAGES**

JSP works best when the structure of the HTML page is fixed but the values at various places need to be computed dynamically. If the structure of the page is dynamic, JSP is less beneficial. Some times servlets are better in such a case. If the page consists of binary data or has little static content, servlets are clearly superior. Sometimes the answer is neither servlets nor JSP alone, but rather a combination of both.

**WRITING SCRIPTLETS**

If you want to do something more complex than output the value of a simple expression .JSP scriptlets let you insert arbitrary code into the servlet’s \_jspService method. Scriptlets have the following form:

<% Java code %>

Scriptlets have access to the same automatically defined variables as do expressions (request, response, session, out , etc ) .So for example you want to explicitly send output of the resultant page , you could use the out variable , as in the following example:

**6.SYSTEM TEST**

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 satisfaction, as shown by successfully 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.

**Test Results:**All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:**All the test cases mentioned above passed successfully. No defects encountered.

**TESTING**

### SOFTWARETESTING

## Testing

Testingisaprocessofexecutingaprogramwiththeaimoffindingerror.Tomakeoursoftware perform well it should be error free. If testing is done successfully, it will remove all the errors from thesoftware.

#### 6.1.1 Types ofTesting

* + - 1. White BoxTesting
      2. Black BoxTesting
      3. Unit testing
      4. IntegrationTesting
      5. AlphaTesting
      6. BetaTesting
      7. Performance Testing and so on

#### White BoxTesting

Testing technique based on knowledge of the internal logic of an application's code and includes tests like coverage of code statements, branches, paths, conditions. It is performedby softwaredevelopers

**Black BoxTesting**

A method of software testing that verifies the functionality of an application without having specificknowledgeoftheapplication'scode/internalstructure.Testsarebasedonrequirements andfunctionality.

**Unit Testing**

Software verification and validation method in which a programmer tests if individual unitsof source code are fit for use. It is usually conducted by the developmentteam.

**IntegrationTesting**

The phase in software testing in which individual software modules are combined and tested as a group. It is usually conducted by testing teams.

**Alpha Testing**

Type of testing a software product or system conducted at the developer's site. Usually it is performed by the end users.

**BetaTesting**

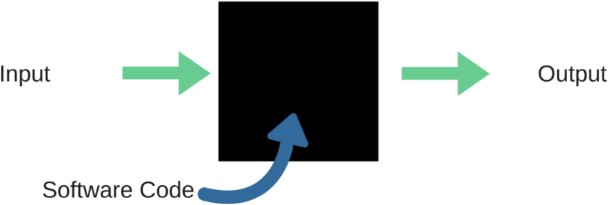
Final testing before releasing application for commercial purpose. It is typically done by end- users or others.

**PerformanceTesting**

Functional testing conducted to evaluate the compliance of a system or component with specified performance requirements. It is usually conducted by the performance engineer.

#### Black BoxTesting

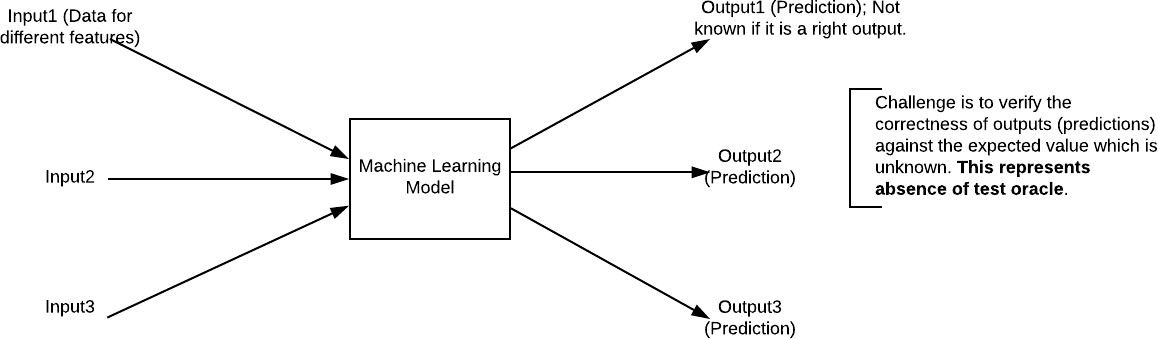
Blackbox testing is testing the functionality of an application without knowing the details of itsimplementationincludinginternalprogramstructure,datastructuresetc.Testcasesforblack box testing are created based on the requirement specifications. Therefore, it is also called as specification-based testing. Fig.4.1 represents the black box testing:



**Fig.:**Black Box Testing

When applied to machine learning models, black box testing would mean testing machine learning models without knowing the internal details such as features of the machine learning

model, the algorithm used to create the model etc. The challenge, however, is to verify the test outcome against the expected values that are known beforehand.



**Fig.:**Black Box Testing for Machine Learning algorithms

The above Fig.4.2 represents the black box testing procedure for machine learning algorithms.

**Table.4.1:**Black box Testing

|  |  |  |
| --- | --- | --- |
| **Input** | **Actual Output** | **Predicted Output** |
| [16,6,324,0,0,0,22,0,0,0,0,0,0] | 0 | 0 |
| [16,7,263,7,0,2,700,9,10,1153,832,9,2] | 1 | 1 |

The model gives out the correct output when different inputs are given which are mentioned in Table 4.1. Therefore the program is said to be executed as expected or correct program

## Testing

Testingisaprocessofexecutingaprogramwiththeaimoffindingerror.Tomakeoursoftware perform well it should be error free. If testing is done successfully it will remove all the errors from thesoftware.

#### 7.2.2 Types ofTesting

* + - 1. White BoxTesting
      2. Black BoxTesting
      3. Unit testing
      4. IntegrationTesting
      5. AlphaTesting
      6. BetaTesting
      7. Performance Testing and so on

#### White BoxTesting

Testing technique based on knowledge of the internal logic of an application's code and includes tests like coverage of code statements, branches, paths, conditions. It is performedbysoftwaredevelopers

**Black BoxTesting**

A method of software testing that verifies the functionality of an application without having specificknowledgeoftheapplication'scode/internalstructure.Testsarebasedonrequirementsandfunctionality.

**Unit Testing**

Software verification and validation method in which a programmer tests if individual unitsof source code are fit for use. It is usually conducted by the developmentteam.

**IntegrationTesting**

The phase in software testing in which individual software modules are combined and tested as a group. It is usually conducted by testing teams.

**Alpha Testing**

Type of testing a software product or system conducted at the developer's site. Usually it is performed by the end users.

**BetaTesting**

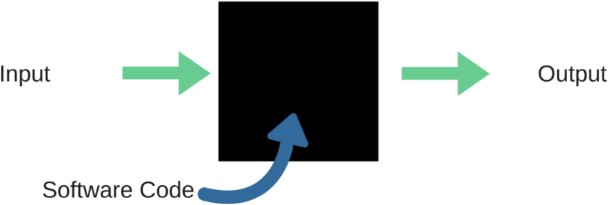
Final testing before releasing application for commercial purpose. It is typically done by end- users or others.

**PerformanceTesting**

Functional testing conducted to evaluate the compliance of a system or component with specified performance requirements. It is usually conducted by the performance engineer.

#### Black BoxTesting

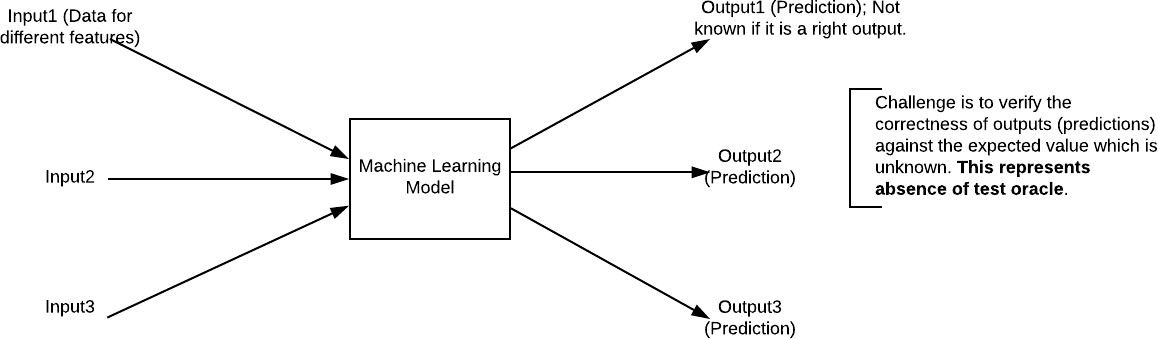
Blackbox testing is testing the functionality of an application without knowing the details of itsimplementationincludinginternalprogramstructure,datastructuresetc.Testcasesforblack box testing are created based on the requirement specifications. Therefore, it is also called as specification-based testing. Fig.4.1 represents the black box testing:



**Fig.:**Black Box Testing

When applied to machine learning models, black box testing would mean testing machine learning models without knowing the internal details such as features of the machine learning

model, the algorithm used to create the model etc. The challenge, however, is to verify the test outcome against the expected values that are known beforehand.



**Fig.:**Black Box Testing for Machine Learning algorithms

The above Fig.4.2 represents the black box testing procedure for machine learning algorithms.

**Table.4.1:**Black box Testing

|  |  |  |
| --- | --- | --- |
| **Input** | **Actual Output** | **Predicted Output** |
| [16,6,324,0,0,0,22,0,0,0,0,0,0] | 0 | 0 |
| [16,7,263,7,0,2,700,9,10,1153,832,9,2] | 1 | 1 |

The model gives out the correct output when different inputs are given which are mentioned in Table 4.1. Therefore the program is said to be executed as expected or correct program

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test | Test Case | Test Case | Test Steps | | | Test | Test |
| Cas | Name | Description | Step | Expected | Actual | Case | Priorit |
| e Id |  |  |  |  |  | Statu | Y |
|  |  |  |  |  |  | s |  |
| 01 | Start the | Host the | If it | We | The | High | High |
|  | Applicatio | application | doesn't | cannot | application |  |  |
|  | N | and test if it | Start | run the | hosts |  |  |
|  |  | starts |  | applicati | success. |  |  |
|  |  | making sure |  | on. |  |  |  |
|  |  | the required |  |  |  |  |  |
|  |  | software is |  |  |  |  |  |
|  |  | available |  |  |  |  |  |
| 02 | Home Page | Check the | If it | We | The | High | High |
|  |  | deployment | doesn’t | cannot | application |  |  |
|  |  | environmen | load. | access | is running |  |  |
|  |  | t for |  | the | successfully |  |  |
|  |  | properly |  | applicati | . |  |  |
|  |  | loading the |  | on. |  |  |  |
|  |  | application. |  |  |  |  |  |
| 03 | User | Verify the | If it | We | The | High | High |
|  | Mode | working of | doesn’t | cannot | application |  |  |
|  |  | the | Respond | use the | displays the |  |  |
|  |  | application |  | Freestyle | Freestyle |  |  |
|  |  | in freestyle |  | mode. | Page |  |  |
|  |  | mode |  |  |  |  |  |
| 04 | Data Input | Verify if the | If it fails | We | The | High | High |
|  |  | application | to take the | cannot | application |  |  |
|  |  | takes input | input or | proceed | updates the |  |  |
|  |  | and updates | store in | further | input to application |  |  |
|  |  |  | The |  |  |  |  |
|  |  |  | Database |  |  |  |  |

**You have to code paste**

**screenshort**

#### URL listing

|  |  |
| --- | --- |
| Websites | Data collected |
| <https://wikipedia.org> | Searching of any information that will be used in documentation. |
| <https://dev.sqlserver.com/doc> | SQL server it performing in mainly depending on the one of the database using. |
| <https://www.answers.com> | Answers.com, online dictionary, encyclopedia and much more. |
| <https://google.co.in> | Any information searching and downloading. |
| <https://training-classes.com> | Designing part information as gathered |

#### 

**Conclusion**

We presented for the first time a secure scheme fully based on attribute-based encryption to ensure both, the confidentiality and access control over data outsourced (in encrypted form) by data owners to the cloud and the fine-grained search control for data users when retrieving encrypted data from the cloud; we called this scheme FABECS. Through a formal analysis and experimentation, we proved the correctness and efficacy of FABECS to be used for storing, sharing and retrieval of documents in a cloud based environment. Furthermore, we provided for the first time Type-III constructions for CP-ABSE and DET-ABE as main building blocks of FABECS. This setting allows using more efficient pairing-friendly curves to achieve recommended security levels, as minimum of 128-bits. These constructions where detailed and their efficacy proved by means of experimentation, over the LISA benchmark for the retrieval task. Further work is focused in the efficiency aspect, as the results presented in this paper did not considered acceleration strategies. For example, parallelization at several levels is possible, besides the scheme is friendly enough to be deployed using parallel patterns such as the manager-worker (for processing a group of attributes at a time) or data encryption (AES on GPUs). Also, as FABECS can be realized with other efficient pairing friendly curves, experimental evaluation could consider the Barreto-Lynn-Scott Curve (BLS) that is also being promoted to be used in practical applications.

**Reference**

[1] A. Bagherzandi, B. Hore, and S. Mehrotra, Search over Encrypted Data. Boston, MA, USA: Springer, 2011, pp. 1088–1093. [2] H. Pham, J. Woodworth, and M. A. Salehi, ‘‘Survey on secure search over encrypted data on the cloud,’’ Concurrency Comput. Pract. Exper., vol. 31, p. 1– 15, Apr. 2019. [3] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, ‘‘Searchable symmetric encryption: Improved definitions and efficient constructions,’’ in Proc. 13th ACM Conf. Comput. Commun. Secur., New York, NY, USA, 2011, pp. 79–88, 2006

[4] M. Zeng, H.-F. Qian, J. Chen, and K. Zhang, ‘‘Forward secure public key encryption with keyword search for outsourced cloud storage,’’ IEEE Trans. Cloud Comput., early access, Sep. 27, 2019, doi: 10.1109/TCC.2019.2944367. [5] S. Kamara, C. Papamanthou, and T. Roeder, ‘‘Cs2: A searchable cryptographic cloud storage system,’’ Microsoft Res., Redmond, WA, USA, Tech. Rep. MSR-TR-2011-58, May 2011. [6] W. Song, B. Wang, Q. Wang, Z. Peng, W. Lou, and Y. Cui, ‘‘A privacypreserved full-text retrieval algorithm over encrypted data for cloud storage applications,’’ J. Parallel Distrib. Comput., vol. 99, pp. 14–27, Jan. 2017. [7] A. G. Kumbhare, Y. Simmhan, and V. Prasanna, ‘‘Designing a secure storage repository for sharing scientific datasets using public clouds,’’ in Proc. 2nd Int. workshop Data Intensive Comput. Clouds, 2011, pp. 31–40. [8] Z. Yang, J. Tang, and H. Liu, ‘‘Cloud information retrieval: Model description and scheme design,’’ IEEE Access, vol. 6, pp. 15420–15430, 2018. [9] H. Yin, J. Zhang, Y. Xiong, L. Ou, F. Li, S. Liao, and K. Li, ‘‘CP-ABSE: A ciphertext-policy attribute-based searchable encryption scheme,’’ IEEE Access, vol. 7, pp. 5682–5694, 2019. [10] A. Sahai and B. Waters, ‘‘Fuzzy identity-based encryption,’’ in Advances in Cryptology (Lecture Notes in Computer Science), vol. 3494, R. Cramer, Ed. Berlin, Germany: Springer-Verlag, 2005, pp. 457–473. [11] M. Morales-Sandoval, J. L. Gonzalez-Compean, A. Diaz-Perez, and V. J. Sosa-Sosa, ‘‘A pairing-based cryptographic approach for data security in the cloud,’’ Int. J. Inf. Secur., vol. 17, no. 4, pp. 441–461, Aug. 2018