

Files2Upload (s3Uploader)

A Project Work-I Report

Submitted in partial fulfillment of requirement of the

Degree of

BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

BY

HRISHABH BAGHRECHA(EN16CS301114)

Under the Guidance of
Miss Annapurna Chitta



Department of Computer Science and Engineering
Faculty of Engineering
MEDI-CAPS UNIVERSITY, INDORE- 453331

May 2020

Report Approval

The project work “**Files2Cloud(Amazon s3Uploader)**” is hereby approved as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warrant its acceptance as prerequisite for the Degree for which it has been submitted.

It is to be understood that by this approval the undersigned do not endorse or approved any statement made, opinion expressed, or conclusion drawn there in; but approve the “Project Report” only for the purpose for which it has been submitted.

Internal Examiner

Name:

Designation:

Affiliation:

External Examiner

Name:

Designation:

Affiliation:

Declaration

I hereby declare that the project entitled **“Files2Cloud(Amazon s3Uploader)”** submitted in partial fulfillment for the award of the degree of Bachelor of Technology in ‘Computer Science Department’ completed under the supervision of **Miss. Annapurna Chitta, HR and Interns Manager, West Agile Labs Hyderabad** is an authentic work.

Further, I declare that the content of this Project work, in full or in parts, have neither been taken from any other source nor have been submitted to any other Institute or University for the award of any degree or diploma.

Signatures and names of students:

HRISHABH BAGHRECHA

Certificate

I, **Miss Annapurna Chitta** certify that the project entitled “**Files2Cloud(Amazon s3Uploader)**” submitted in partial fulfillment for the award of the degree of Bachelor of Technology by **Hrishabh Baghrecha**, is the record carried out by them under my guidance and that the work has not formed the basis of award of any other degree elsewhere.

Miss Annapurna Chitta

HR and Interns Manager

West Agile Labs, Hyderabad

Dr. Suresh Jain

Head of the Department

Computer Science and Engineering

Medi-Caps University, Indore

Acknowledgements

I would like to express my deepest gratitude to Honorable Chancellor, **Shri R C Mittal**, who has provided me with every facility to successfully carry out this project, and my profound indebtedness to **Prof. (Dr.) Sunil K Somani**, Vice Chancellor, Medi-Caps University, whose unfailing support and enthusiasm has always boosted up my morale. We also thank **Prof. (Dr.) DK Panda**, Dean, Faculty of Engineering, Medi-Caps University, for giving me a chance to work on this project. We would also like to thank my Head of the Department **Prof. (Dr.) Suresh Jain** for his continuous encouragement for betterment of the project.

I express our heartfelt gratitude to my **Guide, Miss. Annapurna Chitta, HR and Interns Manager, West Agile labs Hyderabad** without whose continuous help and support, this project would ever have reached to the completion.

We would also like to thank to my team who extended their kind support and help towards the completion of this project.

It is their help and support, due to which we became able to complete the design and technical report.

Without their support this report would not have been possible.

HRISHABH BAGHRECHA(EN16CS301114)

B.Tech. IV Year

Department of Computer Science

Faculty of Engineering

Medi-Caps University, Indore

Abstract

With the advent of computer technology and the internet ,consumers prefer to conduct their business digitally .These digital transactions can be monitored and carried out efficiently .This paper focuses on the need and construction of digital software system to establish modern medium to carry out regular food business.

The main aim of any imdb website is to provide correct information about the movies. In many mess, there are no facility to see movies details and then no watchlist, favoritelist and actorlist for user. Here we are giving features of see details of movies and also allow user to add to the movies in favoritelist and watchlist. It would be possible to reduce the time wastage in searching the movies every time and without using much efforts and manpower if there existed a software for the same. Thus, there arises a need to create an app for the same.

Table of Contents

		Page No.
	Report Approval	ii
	Declaration	iii
	Certificate	iv
	Acknowledgement	v
	Abstract	vi
	Table of Contents	vii
	List of figures	ix
	List of tables	x
	Abbreviations	xi
Chapter 1	Introduction	1
	1.1 Introduction of Company	1
	1.2 Introduction of Project	1
	1.3 Objectives	2
	1.4 Scope	3
	1.5 Source of Data/ Problem in existing system	3
Chapter 2	System Requirement Analysis	4
	2.1 Information Gathering	4
	2.2 System Feasibility	5
	2.2.1 Economical Feasibility	5
	2.2.2 Technical Feasibility	5
	2.2.3 Behavioral Feasibility	6
	2.3 Platform Specification (Development & Deployment)	6
	2.3.1 Hardware	6
	2.3.2 Software Implementation Language/ Technology	7
	2.3.2.1 JavaScript	7

	2.3.2.2 XML	8
	2.3.2.3 JSON	9
	2.3.2.4 MySQL	9
	2.3.2.5 HTML	10
Chapter 3	System Analysis	11
	3.1 Information Flow Representation	11
	3.1.1 ER Diagram	11
	3.1.2 Activity Diagram	13
	3.1.3 Data Flow Diagram	14
	3.1.4 Use Case Diagram	16
	3.1.5 Class Diagram	18
Chapter 4	Design	20
	4.1 Architectural Design	20
	4.2 Procedural/Modular Approach	23
	4.2.1 Modules Used	23
	4.2.2 Internal Data Structures	24
	4.2.3 Algorithm Design for Operations	25
	4.3 Interface Design	26
Chapter 5	Testing	33
	5.1 Testing Objective	33
	5.2 Testing Scope	36
	5.3 Testing Principles	36
	5.4 Testing Methods Used	38
	5.5 Test Cases	44
	5.6 Sample Test Data & Results	46
Chapter 6	Limitations	49
Chapter 7	Future Scope	50
Chapter 8	Conclusion	51

Chapter 9	Bibliography and References	52
-----------	-----------------------------	----

List of Figures

Figure No.	Figure Name	Page No.
Fig 2.1	Database Process	9
Fig 3.1	ER Diagram	12
Fig 3.2	Activity Diagram	13
Fig 3.3	DFD for cloud storage	14
Fig 3.4	DFD for level1	15
Fig 3.5	DFD for level 2	15
Fig 3.6	Use Case Diagram	17
Fig 3.7	Class Diagram	19
Fig 4.1	MVC Architecture	22
Fig 4.2	Internal Data Structure	24
Fig 4.3	Algorithm for Database	25
Fig 4.4	Sign Up Algorithm	26
Fig 4.5	Algorithm for Adding Filter	27
Fig 4.6	Code for Login	28
Fig 4.7	Actor Profile	29
Fig 4.8	Movie Profile	30
Fig 5.1	Testing Principles	39
Fig 5.2	Testing Structures	42
Fig 5.3	Rating	46
Fig 5.4	Movie Entry Page	47
Fig 5.5	Validation Testing WorkFlow	48

List of Tables

Table No.	Table Name	Page No.
Table 2.1	Technical Specification	6
Table 2.2	Hardware Specification	7
Table 5.1	Login Information	46
Table 5.2	SignUp Information	46
Table 5.3	File Information	46

Abbreviations

Abbreviation	Description
API	Application Program Interface
DFD	Data Flow Diagram
JSP	Java Server Pages
JSON	JavaScript Object Notation
Http	Hypertext Transfer Protocol
Https	Hypertext Transfer Protocol Secure
SRS	System Requirement Specifications
MVC	Model-View-Controller

Chapter – 1

Introduction

1.1 Introduction of Company:

West Agile Labs was founded with the concept of Product in mind. This allows us to focus on what truly adds value and on our commitment to our partners. Our strong desire to see the Product succeed drives our efforts and inspires our creativity. This is our foundation, which drives our business, our culture, and everything we do. We capture our core beliefs in these five pillars.

- Strong partnerships, built to last.
- Creativity that explores all options
- Transparently, we work together for the future
- Quality is a core tenet, never an afterthought
- Success happens when we come and stay together.

1.2 Introduction of Project:

The project is Amazon s3 Service based project. The s3 which is called as **Simple Storage Service** is used to upload the local device file to the cloud. The development of the application reduces the storage consumption of a local device. So before developing the tool it is necessary to determine the time factor, integrity and security of the system. Once these things are satisfied, then next step is to determine the operating system and language can be used for developing the tool. Many systems and applications have been developed in this regard to solve to get the proper movies details, but almost none of them fulfill the whole requirements. Many problems can be seen on those existing applications, some lack GUI, some lack automating the process of informing the care taker or guardians.

There are software's available for automating such problem of details of movies but being the fact that desktop consumes more energy or power than the mobile. Nowadays, People have alot of data and files to store but the problem is lack of memory. Apart from the lack of memory, There is always a risk is there regarding the security and safety of the data. Because once if sytem got crashed then it is very difficult or can said as impossible to retrieve the files again. Considering the situation, Storing the data to a place where it always kept like a money in a Bank's Strong Room. The strong room here in the project is nothing but a Amazon cloud based service named as s3.

Use of local device memory is now replaced by the most trending Service i.e. Cloud Based Service for storing data. Today's fast paced world is overloaded with new and ever changing technologies, to cope up with these technologies and constant rise in the user demands one need to maintain a very high quality of each and every product being launched. The software industry is no different and is constantly striving hard to develop and maintain high quality software.

s3Uploader helps users not only in reducing memory consumption but also in accessing those files remotely. Much such software is available in market but they are tedious and complicated with complicated user interface. But **s3Uploader** Service is web application so we can provide very user friendly UI which is easy to understand and manage. The approach/model used for developing software depends on the kind of system we want to develop.

We analyzed the requirements thoroughly first and then spent lots of time in System Designing. But when we went for implementing the desired system, we realized some shortcomings in the design and had to revise the design again. We aimed at following waterfall model but finished up using Modified waterfall Model. While making design of the system emphasis was put on having an efficient modular design. It has been implemented during implementation phase by taking care of basic things which ensure effective modular design. Software testing is very critical element of software quality assurance and represents the ultimate reviews of specification, design and coding.

Testing represents an interesting anomaly for the software. Testing is vital to the success of the system. Errors can be injected at any stage during development. System testing makes a logical assumption that all the parts of system are correct; the goal will be successfully achieved. During testing, the program to be tested is executed with set of test data and the output of program for the test data is evaluated to determine if the program is performing as expected.

1.3 Objectives:

The technology has evolved drastically with time. It has been a boon for humankind. We are planning to create a system that can help technology to reach to every part of the small business and increase the management efficiency. Our motive is to decrease the wastage of memory and providing access to the files from anywhere.

We are creating an platform for users to store the files by loggingIn and also users can view their already uploaded data once they loggedIn. Anyone who is a registered user of the **s3Uploader** can avail the services provided by our system.

The system will not only meet the requirements of the customers but it will also provide a platform to the owners to help them render their services easily.

Our motto is to create such a platform that can modernize these era and use of the evolved technology.

1.4 Scope:

The system is aimed to modernize traditional data storing.

This app has immense potential at saving tones of data being wastage every day in storing the files. The system makes meaningful utilization of technology in doing such.

The product is an web based application used to maintain the data of the local device and providing access to the cloud service remotely from any of the device. Objective of the system is to provide a user friendly cloud storing sytem that is easy to manage, maintain and query.

1.5 Source of Data/ Problem in existing system:

As a use case, backup is probably the most obvious way to consume cloud storage. The nature of backup -- sequential access, less of a focus on performance, infrequent read access, not latency sensitive -- makes it the ideal type of data to be placed in what appears to be nothing more than a large data repository. This single use case, however, limits opportunities to use cloud storage for other workloads and to take advantage of the operational benefits that result from not having to manage infrastructure.

In this project, we take s3 service as response variable and focus on operating predictions by analyzing the rest of variables in the Amazon s3 services. The results can help users to backup their system data to a secure place.

Chapter – 2

System Requirement Analysis

2.1 Information Gathering:

Amazon Simple Storage Service is storage for the Internet. It is designed to make web-scale computing easier for developers.

Amazon S3 has a simple web services interface that you can use to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any developer access to the same highly scalable, reliable, fast, inexpensive data storage infrastructure that Amazon uses to run its own global network of web sites. The service aims to maximize benefits of scale and to pass those benefits on to developers.

The information are gathered from various users who are facing such type of problem. The main aim of this app is to reduce the wastage of storage in storing the users data.

So Before developing the tool it is necessary to determine the time factor, integrity and security of the system. Once these things are satisfied, then next step is to determine the operating system and language can be used for developing the tool.

Many systems and applications have been developed in this regard to solve the automating the process of finding and watching the movies, but almost none of them fulfill the whole requirements. Many problems can be seen on those existing applications, some lack GUI, some lack automating the process of informing the care taker or guardians. There are software's available for automating such problem of fake information but being the fact that desktop consumes more energy or power than the mobile.

s3Uploader System will help users to manage their systems memory. This project is developed for keeping the BackUp of files. Much such software is available in market but they are tedious

and complicated with complicated user interface. But s3Uploader is web application so we can provide very user friendly UI which is easy to understand and manage. The approach/model used for developing software depends on the kind of system we want to develop.

We analyzed the requirements thoroughly first and then spent lots of time in System Designing. But when we went for implementing the desired system, we realized some shortcomings in the design and had to revise the design again. We aimed at following waterfall model but finished up using Modified waterfall Model. While making design of the system emphasis was put on having an efficient modular design. It has been implemented during implementation phase by taking care of basic things which ensure effective modular design. Software testing is very critical element of software quality assurance and represents the ultimate reviews of specification, design and coding.

2.2 System Feasibility:

Once the scope of the project had been defined, it was reasonable to ask: “Can we build system to meet this scope? Is the project feasible?” Our feasibility study focused more on the cost, efforts and resources required for the system.

2.2.1 Economical Feasibility:

As the development work for the system going on smoothly as was planned during the project-planning phase and the company had licensed copies of the software’s required for the development of the project and hence need not pay any additional cost for the same and hence the system is economically feasible.

Economic analysis is the most frequently used method for evaluating the effectiveness of a new system. More commonly known as cost/benefit analysis. JS, html, css and postgres, sequelize database are easily available on internet.

2.2.2 Technical Feasibility:

The technical feasibility in the proposed system deals with the technology used in the system. It deals with the hardware and software used in the system whether they are of latest technology or not and if it happens that after a system is prepared, a new technology arises and the user wants the system based on that technology. This system use web application as platform, apache server, sql and postgresql for database, Javascript, as the language and html or css as user interface.

Technological feasibility is carried out to determine whether the company has the capability, in terms of software, hardware, personnel and expertise, to handle the completion of the project when writing a feasibility report.

Table 2.1 Technical specifications

Front End	ReactJS,HTML,CSS
Back End	NodeJS
Database	PostgreSQL
Platform	Browser

The proposed system can be implemented with some existence technology. The company is already having the hardware and software required for proposed system. The company already has a Local Area Network (LAN). The proposed application will be installed on the server and the interfaces, resources and related data of the proposed system will be shared to all workstation. The workstations will be connected to the server to all workstation. The workstation will be connection to the server over the network so that all users are able to share the application's resources and work individually. Thus s3Uploader is technically feasible

2.2.3 Behavioral Feasibility:

The project has been developed in such a way that it becomes very easy even for a person with little computer knowledge to operate it. This software is very user friendly and does not require any technical person to operate.

The project has behavior relation between users and software. It is user friendly. The project has the features of attendance and see menu and also manager have full access control to the app.

Thus the project is even Behavioral feasible.

2.3 Platform Specification (Development & Deployment):

2.3.1 Hardware:

Hardware versions-can be internal devices or external devices; allow specialized software programs to integrate speech output. Depending on the software program used to read the screen and play the notes material.

Table 2.2 Hardware Specifications

Processor	Intel Core i7,i5 and i3
RAM	4 GB(64 bit operating system)
Hard Disk Capacity	256 GB(at least excluding data size)
Input Devices	Mouse, Keyboard and Microphone, Mobiles

2.3.2 Software Implementation Language/ Technology:

2.3.2.1 JavaScript:

JavaScript is a dynamic computer programming language. It is lightweight and most commonly used as a part of web pages, whose implementations allow client-side script to interact with the

user and make dynamic pages. It is an interpreted programming language with object-oriented capabilities.

JavaScript was first known as LiveScript, but Netscape changed its name to JavaScript, possibly because of the excitement being generated by Java. JavaScript made its first appearance in Netscape 2.0 in 1995 with the name LiveScript. The general-purpose core of the language has been embedded in Netscape, Internet Explorer, and other web browsers.

Client-side JavaScript is the most common form of the language. The script should be included in or referenced by an HTML document for the code to be interpreted by the browser.

It means that a web page need not be a static HTML, but can include programs that interact with the user, control the browser, and dynamically create HTML content.

The JavaScript client-side mechanism provides many advantages over traditional CGI server-side scripts. For example, you might use JavaScript to check if the user has entered a valid e-mail address in a form field.

The JavaScript code is executed when the user submits the form, and only if all the entries are valid, they would be submitted to the Web Server.

JavaScript can be used to trap user-initiated events such as button clicks, link navigation, and other actions that the user initiates explicitly or implicitly.

2.3.2.2 XML:

Extensible Markup Language (XML) is a markup language that defines a set of rules for encoding documents in a format that is both human-readable and machine-readable. The design goals of XML emphasize simplicity, generality, and usability across the Internet. It is a textual data format with strong support via Unicode for different human languages. Although the design of XML focuses on documents, the language is widely used for the representation of arbitrary data structures such as those used in web services.

2.3.2.3 JSON:

JSON (JavaScript Object Notation) is an open-standard format that uses human-readable text to transmit data objects consisting of attribute value pairs. It is the most common data format used for asynchronous browser/server communication, largely replacing XML, and is used by AJAX. JSON is a language-independent data format. It was derived from JavaScript, but as of 2017 many programming languages include code to generate and parse JSON-format data. The official Internet media type for JSON is application/JSON. JSON filenames use the extension .json. When exchanging data between a browser and a server, the data can only be text. JSON is text, and we can convert any JavaScript object into JSON, and send JSON to the server. We can also convert any JSON received from the server into JavaScript objects. This way we can work with the data as JavaScript objects, with no complicated parsing and translations.

2.3.2.4 PostgreSQL:

PostgreSQL is a powerful, open source object-relational database system that uses and extends the SQL language combined with many features that safely store and scale the most complicated data workloads. The origins of PostgreSQL date back to 1986 as part of the **POSTGRES** project at the University of California at Berkeley and has more than 30 years of active development on the core platform.

PostgreSQL has earned a strong reputation for its proven architecture, reliability, data integrity, robust feature set, extensibility, and the dedication of the open source community behind the software to consistently deliver performant and innovative solutions. PostgreSQL runs on **all major operating systems**, has been **ACID**-compliant since 2001, and has powerful add-ons such as the popular **PostGIS** geospatial database extender. It is no surprise that PostgreSQL has become the open source relational database of choice for many people and organisations.

Below is an inexhaustive list of various features found in PostgreSQL, with more being added in every **major release**:

- Data Types
 - Primitives: Integer, Numeric, String, Boolean
 - Structured: Date/Time, Array, Range, UUID
 - Document: JSON/JSONB, XML, Key-value (Hstore)
 - Geometry: Point, Line, Circle, Polygon
 - Customizations: Composite, Custom Types
- Data Integrity
 - UNIQUE, NOT NULL
 - Primary Keys
 - Foreign Keys
 - Exclusion Constraints
- Concurrency, Performance
 - Indexing: B-tree, Multicolumn, Expressions, Partial
 - Parallelization of read queries and building B-tree indexes
 - Table partitioning
 - All transaction isolation levels defined in the SQL standard, including Serializable
- Reliability, Disaster Recovery
 - Write-ahead Logging (WAL)
 - Replication: Asynchronous, Synchronous, Logical
 - Point-in-time-recovery (PITR), active standbys
 - Tablespaces
- Security
 - Authentication: GSSAPI, SSPI, LDAP, SCRAM-SHA-256, Certificate, and more
 - Robust access-control system
 - Column and row-level security
 - Multi-factor authentication with certificates and an additional method
- Extensibility

- Stored functions and procedures
- Procedural Languages: PL/PGSQL, Perl, Python (and many more)
- SQL/JSON path expressions
- Customizable storage interface for tables
- Internationalisation, Text Search
 - Support for international character sets, e.g. through ICU collations
 - Case-insensitive and accent-insensitive collations
 - Full-text search

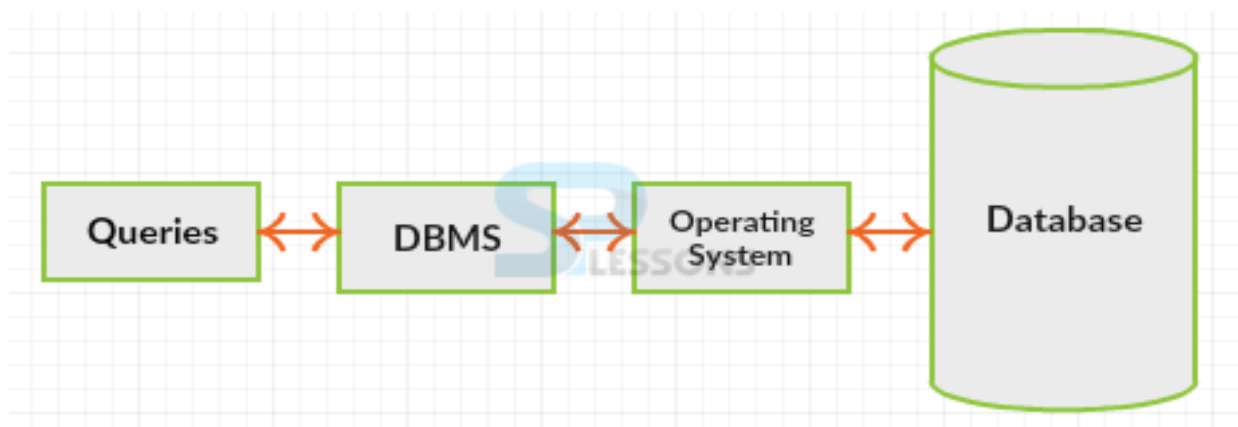


Fig 2.2 Database process

2.3.2.5 HTML:

Hypertext Markup Language(HTML) is the standard markup language for documents designed to be displayed in a web browser. It can be assisted by technologies such as Cascading Style Sheets(CSS) and scripting languages such as JavaScript.

Web browsers receive HTML documents from a **web server** or from local storage and **render** the documents into multimedia web pages. HTML describes the structure of a **web page semantically** and originally included cues for the appearance of the document.

HTML elements are the building blocks of HTML pages. With HTML constructs, **images** and other objects such as **interactive forms** may be embedded into the rendered page. HTML provides a means to create **structured documents** by denoting structural **semantics** for text such as headings, paragraphs, lists, **links**, quotes and other items. HTML elements are delineated by *tags*, written using **angle brackets**. Tags such as **** and **<input />** directly introduce content into the page. Other tags such as **<p>** surround and provide information about document text and may include other tags as sub-elements. Browsers do not display the HTML tags, but use them to interpret the content of the page.

Chapter – 3

System Analysis

3.1 Information Flow Representation:

Information flow representations UML behavior diagram which shows the exchange of information between system entities at some high levels of abstraction. Information flows may be useful to describe circulation of information through a system by representing aspects of models not yet fully specified or with fewer details.

Information flows do not specify the nature of the information, mechanisms by which it is conveyed, sequences of exchange, or any control conditions. Information items can be used to represent the information that flows through a system along the information flows before details of their realization have been designed.

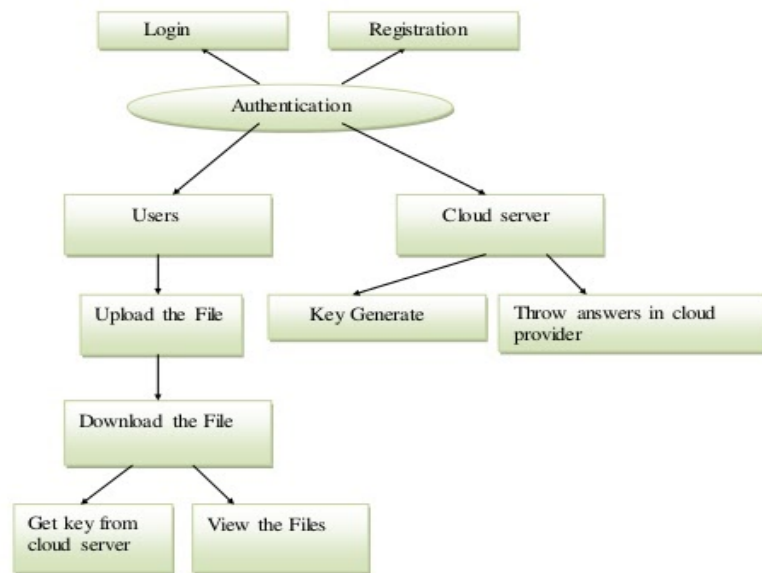
3.1.1 ER Diagram:

The ER or (Entity Relational Model) is a high-level conceptual data model diagram. Entity-Relation model is based on the notion of real-world entities and the relationship between them. ER modeling helps you to analyze data requirements systematically to produce a well-designed database. So, it is considered a best practice to complete ER modeling before implementing your database.

Entity relationship diagram displays the relationships of entity set stored in a database. In other words, we can say that ER diagrams help you to explain the logical structure of databases. At first look, an ER diagram looks very similar to the flowchart. However, ER Diagram includes many specialized symbols, and its meanings make this model unique.

An ER diagram shows the relationship among entity sets. An entity set is a group of similar entities and these entities can have attributes. In terms of DBMS, an entity is a table or attribute

of a table in database, so by showing relationship among tables and their attributes, ER diagram shows the complete logical structure of a database. Lets have a look at a simple ER diagram to understand this concept.



3.1.3 System Flow Diagram:

Fig 3.1 ER Diagram

3.1.2 Activity Diagram:

Activity diagram is defined as a UML diagram that focuses on the execution and flow of the behavior of a system instead of implementation. It is also called object-oriented flowchart. Activity diagrams consist of activities that are made up of actions which apply to behavioral modeling technology.

Activity diagrams are used to model processes and workflows. The essence of a useful activity diagram is focused on communicating a specific aspect of a system's dynamic behavior. Activity diagrams capture the dynamic elements of a system.

Activity diagram is similar to a flowchart that visualizes flow from one activity to another activity. In simple words, an activity diagram is used to activity diagrams that describe the flow of execution between multiple activities.

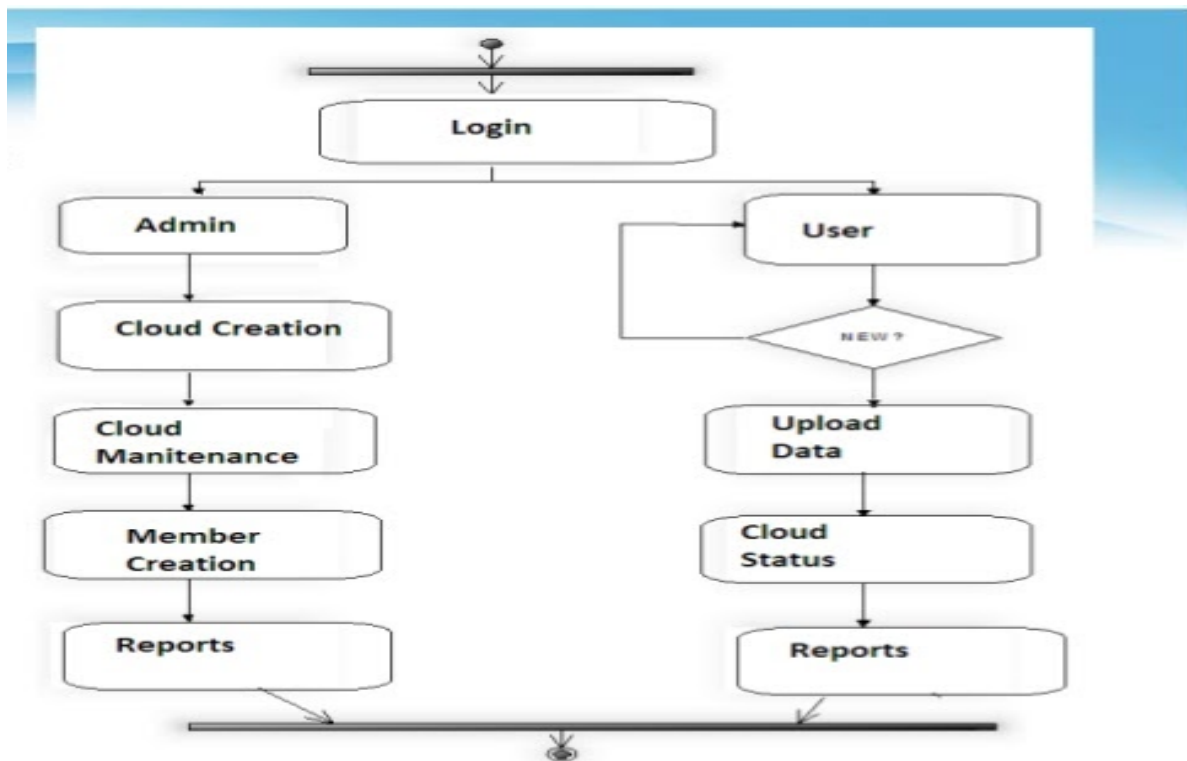


Fig 3.2 Activity Diagram

3.1.3 Data Flow Diagram:

A data-flow diagram (DFD) is a way of representing a flow of a data of a process or a system (usually an information system). The DFD also provides information about the outputs and inputs of each entity and the process itself. A data-flow diagram has no control flow, there are no decision rules and no loops. Specific operations based on the data can be represented by a flowchart.

For each data flow, at least one of the endpoints (source and / or destination) must exist in a process. The refined representation of a process can be done in another data-flow diagram, which subdivides this process into sub-processes.

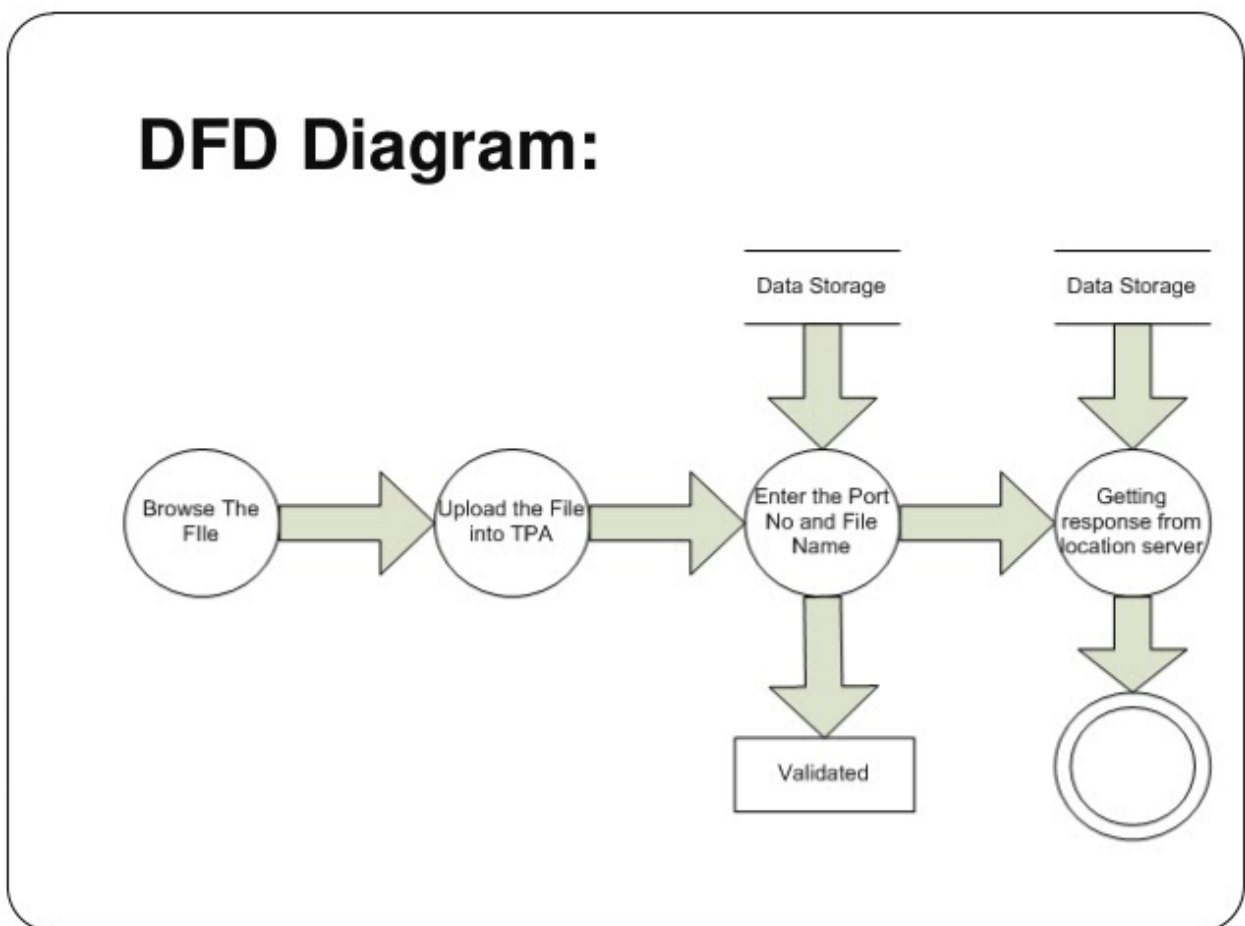


Fig 3.3 Data Flow Diagram For movie review

The data-flow diagram is part of the structured-analysis modeling tools. When using UML, the activity diagram typically takes over the role of the data-flow diagram. A special form of data-flow plan is a site-oriented data-flow plan.

Data-flow diagrams can be regarded as inverted Petri nets, because places in such networks correspond to the semantics of data memories. Analogously, the semantics of transitions from Petri nets and data flows and functions from data-flow diagrams should be considered equivalent.

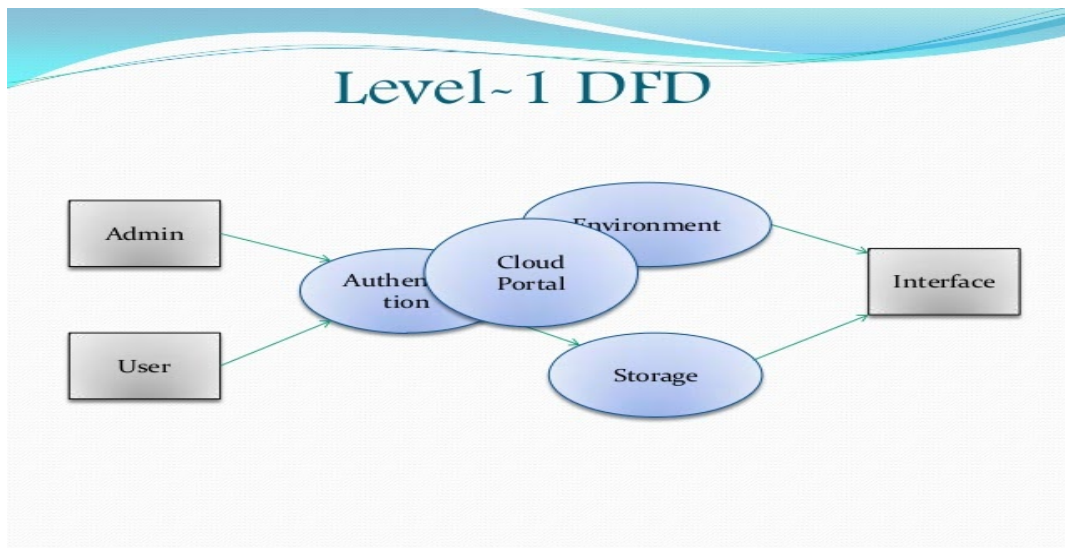


Fig 3.4 Data Flow Diagram level 1

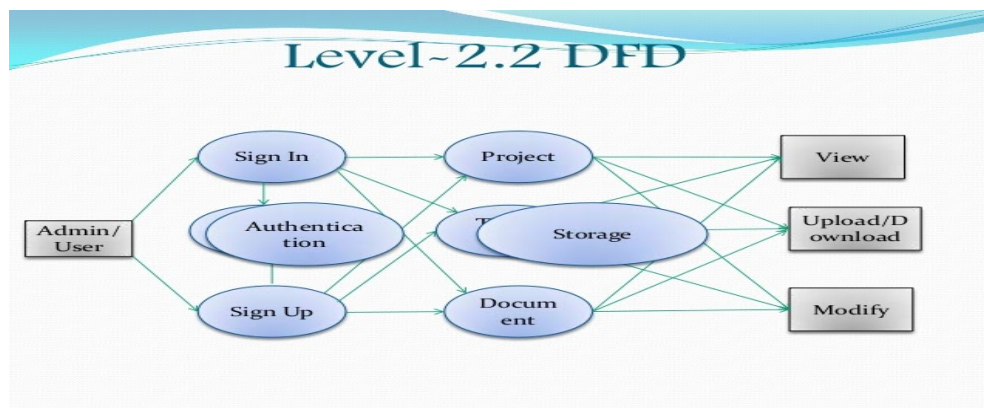


Fig 3.5 Data Flow Diagram level 2

3.1.4 Use Case Diagram:

A use case diagram is a dynamic or behavior diagram in UML. Use case diagrams model the functionality of a system using actors and use cases. Use cases are a set of actions, services, and functions that the system needs to perform. In this context, a "system" is something being developed or operated, such as a web site. The "actors" are people or entities operating under defined roles within the system.

They provide a good high level analysis from outside the system. Use case diagrams specify how the system interacts with actors without worrying about the details of how that functionality is implemented.

Use case diagrams are valuable for visualizing the functional requirements of a system that will translate into design choices and development priorities.

They also help identify any internal or external factors that may influence the system and should be taken into consideration.

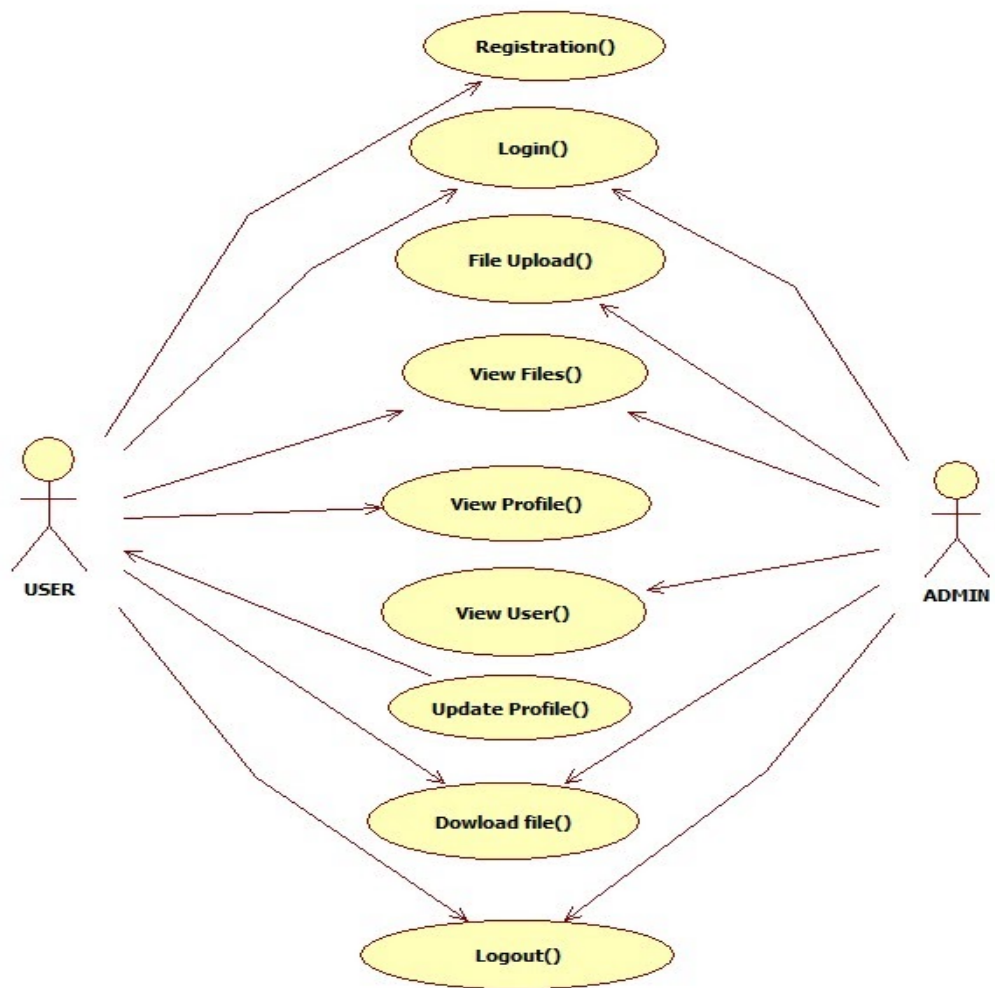


Fig 3.7 Use Case Diagram

3.1.5 Class Diagram:

A class diagram in the [Unified Modeling Language](#) (UML) is a type of static structure diagram that describes the structure of a system by showing the system's [classes](#), their attributes, operations (or methods), and the relationships among objects.

The class diagram is the main building block of [object-oriented](#) modeling. It is used for general [conceptual modeling](#) of the structure of the application, and for detailed modeling translating the models into [programming code](#). Class diagrams can also be used for [data modeling](#). The classes in a class diagram represent both the main elements, interactions in the application, and the classes to be programmed.

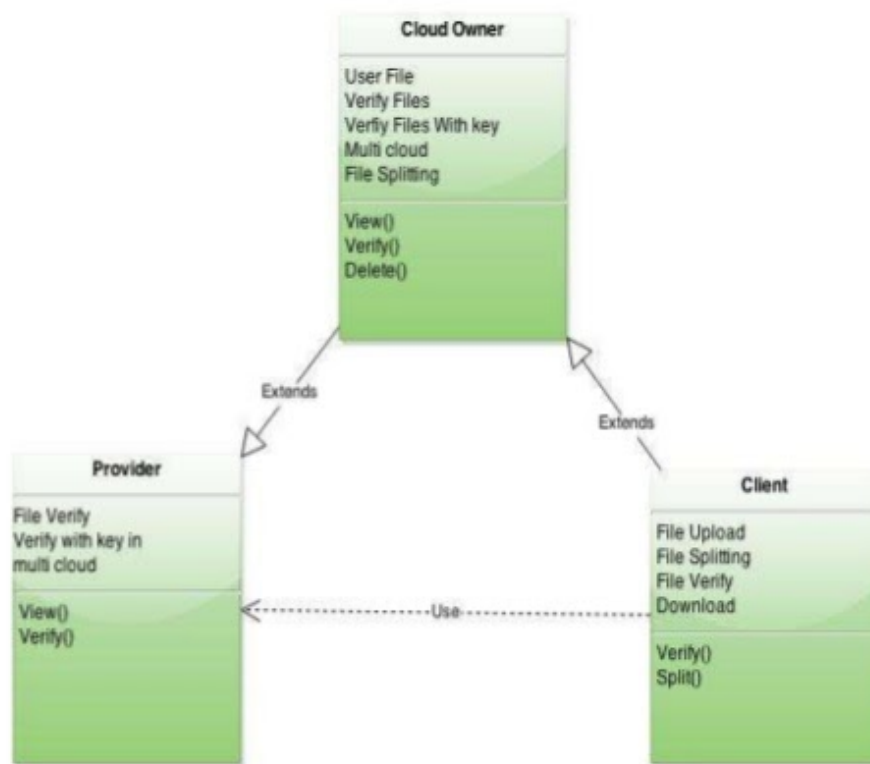


Fig 3.8 Class Diagram

Chapter – 4

Design

4.1 Architectural Design:

Conceptually, Software Architecture is the depiction of the building-blocks of a software system. It describes how subsystems and their components are organized and how they are associated with each other aiming to meet functional and non-functional properties of the system.

Architectural patterns are documented representations of predefined high-level subdivisions of software systems, which also specify their responsibilities and relationships. Each of these patterns intends to support the implementation of global properties required by the application to be built, and may be found categorized into groups according to the problems it proposes to address.

Thus, the definition of an architectural pattern is the first step to conceive the architecture of the product itself.

The architectural patterns, which intend to support the development of systems that requires wide interaction with human-beings, can be gathered into a specific category dedicated to interactive systems. Although they are structurally different, their common purpose is to decouple the user interface from the business logic, giving the possibility to adapt and evolve them separately.

Some of the most prominent interactive architectural patterns are the Model View-Controller (MVC) and its variants, such as Model-View-Presenter (MVP), and the Martin Fowler's patterns – Supervising Controller and Passive View, which in turn are derived from MVP.

The pattern used in the project is MVC and is described as followed:-

MVC Pattern

The MVC pattern splits code into one of three MVC components. When we create a new class or file, we must know which component it belongs to:

- **Model:** contains application data and business logic (the rules of the system). For example, user accounts, products we sell, a set of dishes etc. The model component has no knowledge of the interface. In the project, it will meals and their information.
- **View:** contains everything that is visible on the screen and offers interaction to the user. In the project, this component is defined in the activity_main.html and login.html files.
- **Controller:** This is the "glue" between the view and the model, which also manages the application logic. The controller reacts to the user's input and presents the data requested by the user. Where does it retrieve the data from? In the project, we have controllers called as: the Index page.

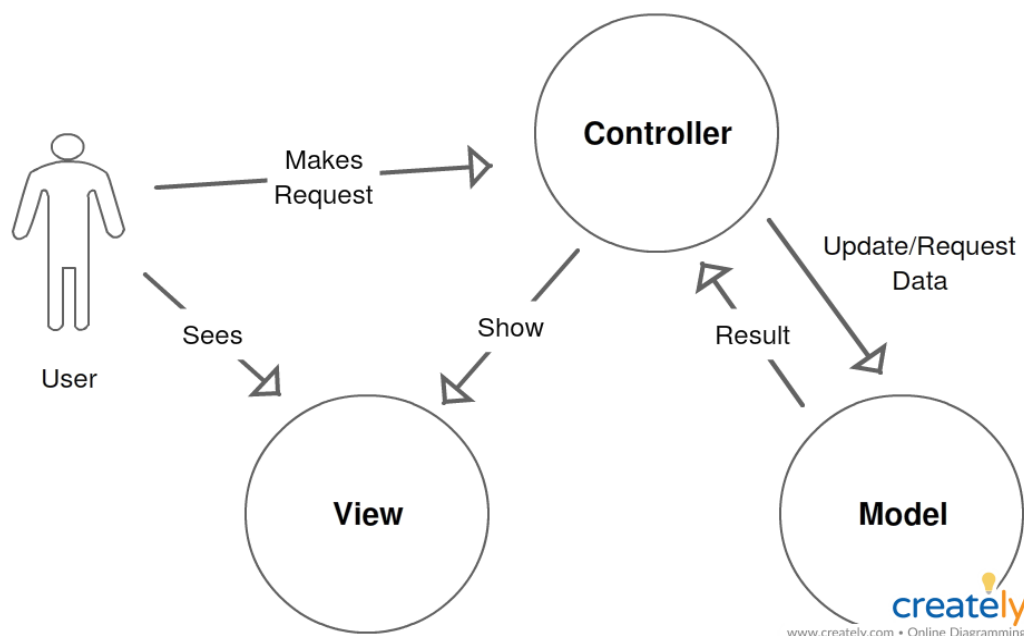


Fig 4.1 MVC Architecture

4.2.1 Modules Used:

Different modules use in making this project are:-

1. **Login/Sign-up:**

This module is used to register and login to the app.

2. **Add Files:**

This module is used for enabling admin to add local device files.

3. **View Files:**

This module is used for providing access to see already uploaded file

4. **Upload Files:**

This module is used for letting users for uploading files to s3.

5. **Delete Files:**

This module is used for letting users for deleting files from s3.

6. **Download Files:**

This module is used for letting users for downloading uploaded files from s3.

4.2.2 Infrastructure:

The project is based on JavaScript and hence the main data structures are in form of class and objects.

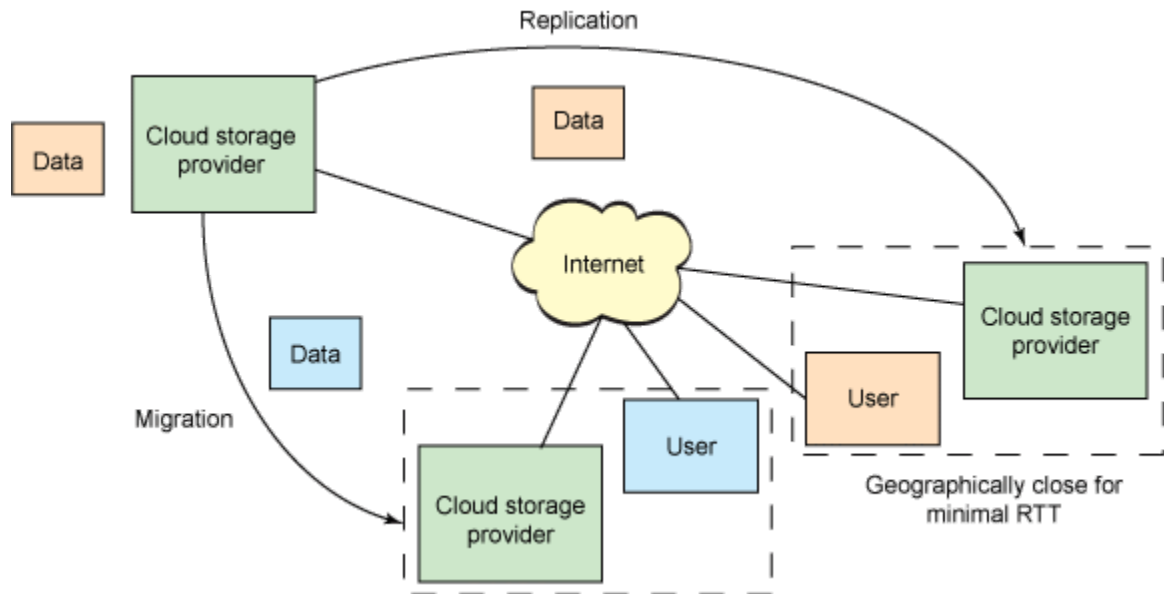


Fig 4.2 Infrastructure

4.2.3 Algorithm Design for Operations

Algorithm for registration and creating new bucket in Amazon s3:-

```
let hashedPassword = passwordHash.generate(req.body.password);
req.body.password = hashedPassword;

const user = await models.UserT.create(req.body);
if (user) {
  const BUCKET_FOLDER =
    user.firstName + "_" + user.lastName + "_" + user.id;

  s3.headBucket({ Bucket: BUCKET_NAME }, function (err, data) {
    if (err) {
      s3.createBucket(params, function (err, data) {
        if (err) console.log(err, err.stack);
        else {
          console.log("Bucket Created Successfully", data.Location);

          var params = {
            Bucket: BUCKET_NAME,
            Key: BUCKET_FOLDER + "/",
            ACL: "public-read",
            Body: "body does not matter",
          };
          s3.upload(params, function (err, data) {
            if (err) {
              console.log("Error creating the folder: ", err);
            } else {
              console.log("Successfully created a folder on S3");
            }
          });
        }
      });
    }
  });
} else {
  console.log("Bucket Already Exists ");
  var params = {
    Bucket: BUCKET_NAME,
    Key: BUCKET_FOLDER + "/",
    ACL: "public-read",
    Body: "body does not matter",
  };
  s3.upload(params, function (err, data) {
    if (err) {
      console.log("Error creating the folder: ", err);
    } else {
      console.log("Successfully created a folder on S3");
    }
  });
}
});
logger.info('SIGNUP SUCCESS!')
res.status(200).json({
  success: true,
  data: user,
  message: "SUCCESSFULLY SIGN-UP",
});
```

Fig 4.3 Algorithm for Registration

```

const user = await models.UserT.findOne({
  where:
  {
    email: req.body.email
  }
})
const BUCKET_FOLDER = user.firstName + "_" + user.lastName + "_" + user.id;

let imageFile = req.files.file;
const dataArray = []
let i = 0
let imageFileArr = []
console.log(Object.keys(imageFile).length, "SSSSSSSSSS")
if (!Array.isArray(imageFile)) {
  imageFileArr.push(imageFile)
}
else {
  imageFileArr = imageFile
}
console.log(imageFileArr, 'AAAAAAAAAA')

imageFileArr.map((item, key) => {
  var params = {
    Bucket: BUCKET_NAME,
    Key: BUCKET_FOLDER + "/" + item.name,
    Body: item.data,
    ACL: "public-read",
  };
  s3.upload(params, async function (err, data) {
    if (err) {
      console.log("Error creating the folder: ", err);
    } else {
      dataArray.push(data)
      console.log("Successfully UPLOADED file on S3");
      const uData = await models.Data.create({
        fName: data.Key.replace(BUCKET_FOLDER + '/', ''),
        fType: item.mimetype,
        fSize: (item.size / 1000) + ' KB',
        file: data.Location,
        fDate: moment(today).format('L'),
        email: user.email
      })
      i = i + 1
      if (i === imageFileArr.length) {
        res.status(200).json({
          success: true,
          data: dataArray,
          message: "SUCCESSFULLY UPLOADED",
        });
      }
    }
  })
})
}

```

Fig 4.4 Algorithm for Uploading files

```

/** @description Method for Deleting User Data
 * @async
 * @method
 * @param {object} req - Request object contains User details --attribute email and File to delete
 * @param {object} res - Reponse object contains User Data details.
 * @param {function} next(error) {
}} next - calls the error handling middleware.
*/
async function deleteData(req,res,next){
  console.log('ENTER DELETE')
  const user = await models.UserT.findOne({
    where: {
      email: req.body.email
    }
  })
  console.log(req.body)
  let delFile = req.body.delete
  BUCKET_FOLDER = user.firstName + "_" + user.lastName + "_" + user.id;

  var params = {
    Bucket: BUCKET_NAME,
    Delimiter: '/',
    Prefix: BUCKET_FOLDER + '/'
  };

  var params = { Bucket: BUCKET_NAME, Key: BUCKET_FOLDER+'/'+delFile };
  console.log(params)

  s3.deleteObject(params, function (err, data) {
    console.log("DELETEOBJECTS")
    if (err) console.log(err, err.stack);
    else console.log('DELETED');
  });
  console.log('DELETE METHOD')
  const del = await models.Data.destroy({
    where:
    {
      email: req.body.email,
      fName: delFile
    }
  })

  logger.info('FILE DELETED from S3')

  res.status(200).json({
    success: true,
    message: "SUCCESSFULLY DELETED",
  });
}

module.exports={deleteData}

```

Fig 4.5 Algorithm for Adding filter

4.3 Interface Design:

The react code for login screen:-

```
render() {  
  return [  
    <div>  
      <form>  
        <div>  
          <h3>Sign Up</h3>  
  
          <div className="form-group">  
            <label>First name</label>  
            <input type="text" id='fname' className="form-control" placeholder="First name" onChange={this.onFirstName} value={this.state.firstname} pattern='[A-Za-z]' title="Must be Alphabet" minLength='6' required />  
          </div>  
  
          <div className="form-group">  
            <label>Last name</label>  
            <input type="text" id='lname' className="form-control" placeholder="Last name" onChange={this.onLastName} value={this.state.lastname} pattern='[A-Za-z]' title="Must be Alphabet" minLength='6' required />  
          </div>  
  
          <div className="form-group">  
            <label>Email address</label>  
            <input type="email" id='mail' className="form-control" placeholder="Enter email" onChange={this.onEmail} value={this.state.mail} />  
          </div>  
  
          <div className="form-group">  
            <label>Password</label>  
            <input type="password" className="form-control" placeholder="Enter password" onChange={this.onPassword} value={this.state.pass} />  
          </div>  
  
          <button type="button" className="btn btn-primary btn-block" onClick={this.handleSubmit}>Submit</button>  
  
          <ToastsContainer store={ToastsStore} position={ToastsContainerPosition.TOP_RIGHT} />  
  
          <p className="forgot-password text-right">  
            Already registered <Link className="nav-link" to={"/signin"}>Login</Link>  
          </p>  
        </div>  
      </form>  
    </div>  
  ];  
}
```

Fig 4.6 Code for login

The code results in following UI:

uploading@s3

Login Sign up

Sign Up

First name

Last name

Email address

Password

Submit

Already registered

[Login](#)

Fig 4.7 SignUp Page UI

UPLOAD FILE:
 Choose Files
 No file chosen

UPLOAD TO s3







FILE NAME	FILE TYPE	FILE SIZE	UPLOADED ON	ACTION
 Screenshot from 2020-03-20 16-32- 10 (2).png	image/png	272.211 KB	04/27/2020	 
 Screenshot from 2020-03-23 20-14- 57 (1).png	image/png	225.198 KB	04/27/2020	 

Fig 4.8 User Profile with Uploaded data.

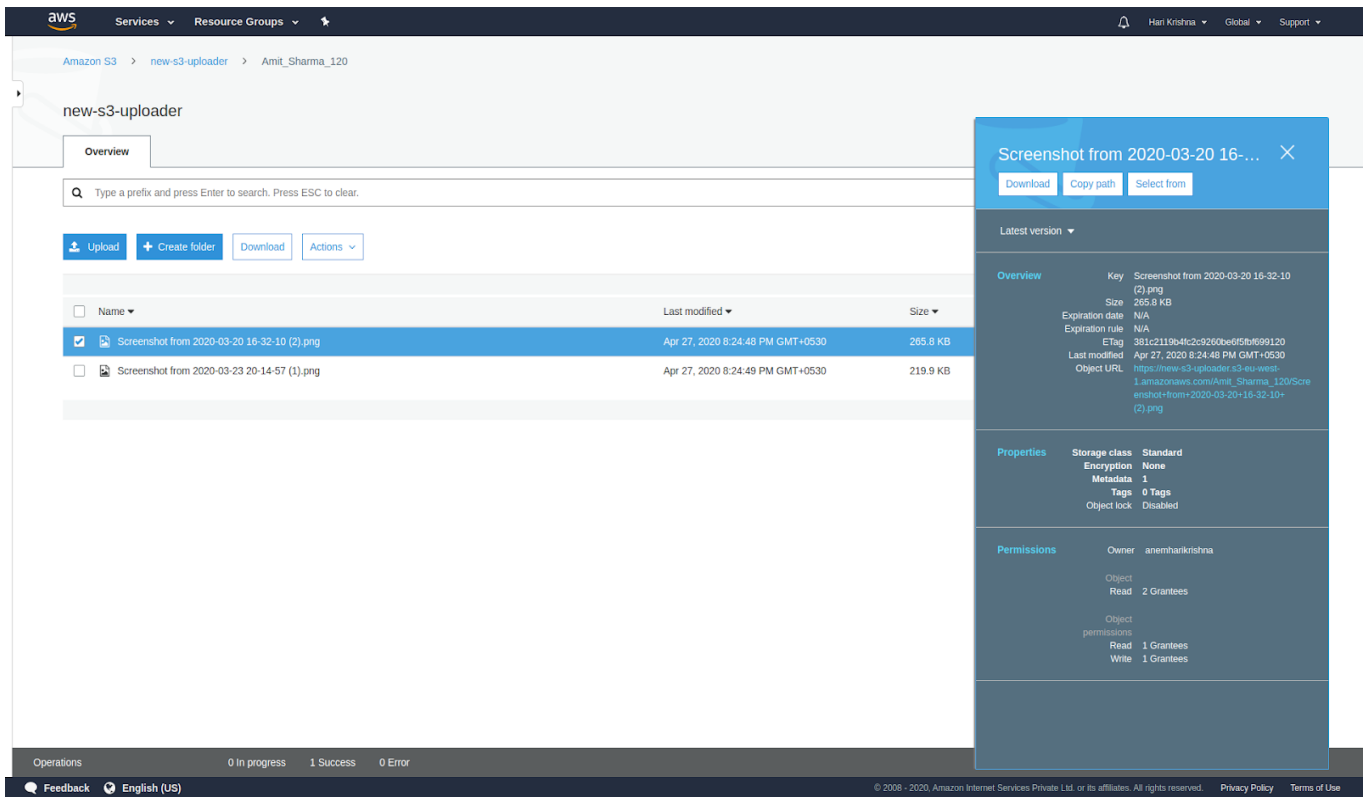


Fig 4.9 s3 Bucket with user folder's files

Edit Data - localhost (127.0.0.1:5432) - s3data - publicData									
File Edit View Tools Help									
100 rows									
	id	fName	fType	fSize	fDate	email	createdAt	updatedAt	file
	[PK] Integer	character varying(255)	character varying(255)	character varying(255)	character varying(255)	character varying(255)	timestamp with time zone	timestamp with time zone	character varying(255)
1	504	Screenshot from 2020-03-20 16-32-10 (2).png	image/png	272.211 KB	04/27/2020	amit@gmail.com	2020-04-27 20:24:48.007+05:30	2020-04-27 20:24:48.007+05:30	https://new-s3-uploa
2	505	Screenshot from 2020-03-23 20-14-57 (1).png	image/png	225.198 KB	04/27/2020	amit@gmail.com	2020-04-27 20:24:48.242+05:30	2020-04-27 20:24:48.242+05:30	https://new-s3-uploa
3	506	Screenshot from 2020-04-12 11-08-07.png	image/png	150.731 KB	04/27/2020	hrishabhjain67@gmail.com	2020-04-27 20:30:08.286+05:30	2020-04-27 20:30:08.286+05:30	https://new-s3-uploa
4	507	Screenshot from 2020-03-23 20-14-57 (1) (1).png	image/png	225.198 KB	04/27/2020	hrishabhjain67@gmail.com	2020-04-27 20:30:08.293+05:30	2020-04-27 20:30:08.293+05:30	https://new-s3-uploa
5	508	Screenshot from 2020-03-27 11-49-27.png	image/png	256.268 KB	04/27/2020	hrishabhjain67@gmail.com	2020-04-27 20:30:08.619+05:30	2020-04-27 20:30:08.619+05:30	https://new-s3-uploa
*									

Fig 4.10 Database entry for user's data

Chapter – 5

Testing

Implementation is the stage in the project where the theoretical design is turned into a working system and is giving confidence on the new system for the users that it will work efficiently and effectively. It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the changeover, an evaluation of change over methods. Apart from planning major task of preparing the implementation are education and training of users. The implementation process begins with preparing a plan for the implementation of the system. According to this plan, the activities are to be carried out, discussions made regarding the equipment and resources and the additional equipment has to be acquired to implement the new system.

In network backup system no additional resources are needed. Implementation is the final and the most important phase. The most critical stage in achieving a successful new system is giving the users confidence that the new system will work and be effective. The system can be implemented only after thorough testing is done and if it is found to be working according to the specification. This method also offers the greatest security since the old system can take over if the errors are found or inability to handle certain type of transactions while using the new system.

5.1 Testing Objectives:

Software Testing has different goals and objectives. The major objectives of Software testing areas follows:

Objectives of software testing:-

- Ensures the quality of product.

- Defect prevention and detection.
- Verify and validate the user requirement.
- Ready to integration and revise the component.
- Focus on accurate and reliable result.
- Work should be performed under predefined process and SRS.
- Discuss on generating test cases test cases.
- Provide information to take decision for next phase.
- Gain confidence of work.
- Evaluates the capabilities of a system and system performance.

Software testing makes sure that the testing is being done properly and hence the system is ready for use. Good coverage means that the testing has been done to cover the various areas like functionality of the application, compatibility of the application with the OS, hardware and different types of browsers, performance testing to test the performance of the application and load testing to make sure that the system is reliable and should not crash or there should not be any blocking issues. It also determines that the application can be deployed easily to the machine and without any resistance. Hence the application is easy to install, learn and use.

The goals and objectives of software testing are numerous, which when achieved help developers build a defect less and error free software and application that has exceptional performance, quality, effectiveness, security, among other things. Though the objective of testing can vary from company to company and project to project, there are some goals that are similar for all. These objectives are:

1. **Verification:** A prominent objective of testing is verification, which allows testers to confirm that the software meets the various business and technical requirements stated by the client before the inception of the whole project. These requirements and specifications guide the design and development of the software, hence are required to be followed rigorously. Moreover, compliance with these requirements and specifications is important for the success of the project as well as to satisfy the client.

2. **Validation:** Confirms that the software performs as expected and as per the requirements of the clients. Validation involves checking the comparing the final output with the expected output and then making necessary changes if their is a difference between the two.
3. **Defects:** The most important purpose of testing is to find different defects in the software to prevent its failure or crash during implementation or go live of the project. Defects if left undetected or unattended can harm the functioning of the software and can lead to loss of resources, money, and reputation of the client. Therefore, software testing is executed regularly during each **stage of software development** to find defects of various kinds. The ultimate source of these defects can be traced back to a fault introduced during the specification, design, development, or programming phases of the software.
4. **Providing Information:** With the assistance of reports generated during the process of software testing, testers can accumulate a variety of information related to the software and the steps taken to prevent its failure. These, then can be shared with all the stakeholders of the project for better understanding of the project as well as to establish transparency between members.
5. **Preventing Defects:** During the process of testing the aim of testes to identify defects and prevent them from occurring aging in the future. To accomplish this goal, software is tested rigorously by a independent testers, who are not responsible for software development.
6. **Quality Analysis:** Testing helps improve the **quality of the software** by constantly measuring and verifying its design and coding. Additionally, various types of testing techniques are used by testers, which help them achieve the desired software quality.
7. **Compatibility:** It helps validate application's compatibility with the implementation environment, various devices, Operating Systems, user requirements, among other things.

8. **For Optimum User Experience:** Easy software and application accessibility and optimum user experience are two important requirements that need to be accomplished for the success of any project as well as to increase the revenue of the client. Therefore, to ensure this software is tested again and again by the testers with the assistance of stress testing, load testing, spike testing, etc.
9. **Verifying Performance & Functionality:** It ensures that the software has superior **performance** and functionality. This is mainly verified by placing the software under extreme stress to identify and measure its all plausible failure modes. To ensure this, performance testing, usability testing, functionality testing, etc. is executed by the testers.

5.2 Testing Scope:

The document mainly targets the GUI testing and validating data in report output as user Requirements Specifications provided by Client.

The s3Uploader app is being tested using ChaiMocha JS Test Framework

Functions which are tested:

- Api modules on routes.
- GUI
- Developers also tested the functions of registration and login.
- All databases and tables are tested.
- Validate correct username and password.

Main functions:

- GUI
- Uploaded Files

5.3 Testing Principles:

Software testing is a process of executing a program with the aim of finding the error. To make our software perform well it should be error free. If testing is done successfully it will remove all the errors from the software.

There are seven principles in software testing:

1. Testing shows presence of defects
2. Exhaustive testing is not possible
3. Early testing
4. Defect clustering
5. Pesticide paradox
6. Testing is context dependent
7. Absence of errors fallacy.

1) Testing shows presence of defects:

The goal of software testing is to make the software fail. Software testing reduces the presence of defects. Software testing talks about the presence of defects and doesn't talk about the absence of defects. Software testing can ensure that defects are present but it can not prove that software is defects free. Even multiple testing can never ensure that software is 100% bug-free. Testing can reduce the number of defects but not removes all defects.

2) Exhaustive testing is not possible:

It is the process of testing the functionality of a software in all possible inputs (valid or invalid) and pre-conditions is known as exhaustive testing. Exhaustive testing is impossible means the software can never test at every test cases. It can test only some test cases and assume that software is correct and it will produce the correct output in every test cases. If

the software will test every test cases then it will take more cost, effort, etc. and which is impractical.

3) **Early Testing:**

To find the defect in the software, early test activity shall be started. The defect detected in early phases of SDLC will be very less expensive. For better performance of software, software testing will start at initial phase i.e. testing will perform at the requirement analysis phase.

4) **Defect clustering:**

In a project, a small number of the module can contain most of the defects. Pareto Principle to software testing states that 80% of software defect comes from 20% of modules.

Hence, testing principle states that - Testing talks about the presence of defects and don't talk about the absence of defects. i.e. Software Testing reduces the probability of undiscovered defects remaining in the software but even if no defects are found, it is not a proof of correctness.

But what if, you work extra hard, taking all precautions & make your software product 99% bug-free. And the software does not meet the needs & requirements of the clients.

5) **Pesticide paradox:**

Repeating the same test cases again and again will not find new bugs. So it is necessary to review the test cases and add or update test cases to find new bugs.

6) **Testing is context dependent:**

Testing approach depends on context of software developed. Different types of software need to perform different types of testing. For example, The testing of the e-commerce site is different from the testing of the Android application.

7) **Absence of errors fallacy:**

If a built software is 99% bug-free but it does not follow the user requirement then it is unusable. It is not only necessary that software is 99% bug-free but it also mandatory to fulfill all the customers requirements.

5.4 Testing Methods Used:

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. Infact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software.

Testing is the set of activities that can be planned in advance and conducted systematically.

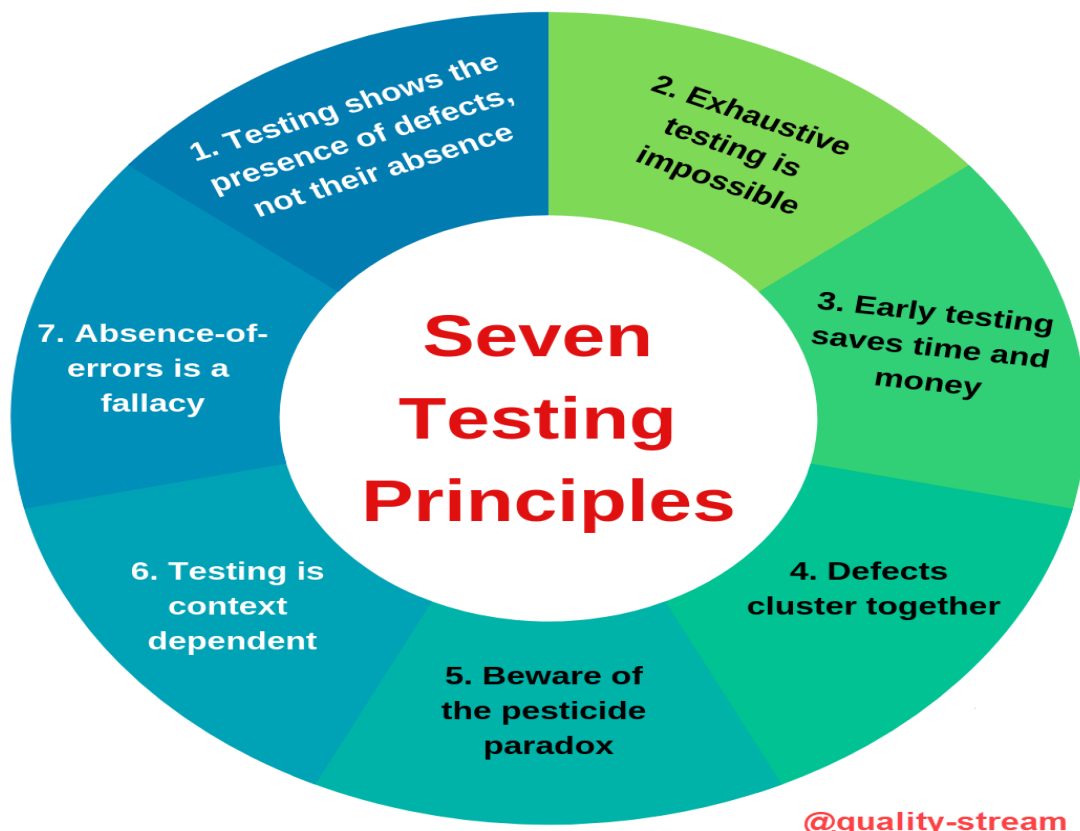


Fig 5.1 Testing Principles

The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

According to ANSI/IEEE 1059 standard, Testing can be defined as- A process of analyzing a software item to detect the differences between existing and required conditions(that is defects/errors/bugs) and to evaluate the features of the software item.

It involves identifying bug/error/defect in a software without correcting it. Normally professionals with a quality assurance background are involved in bug identification. Testing is performed in the testing phase.

STRATEGIC APPROACH TO SOFTWARE TESTING

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behavior, performance, constraints and validation criteria for software are established. Moving in ward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along stream lines that decrease the level of abstraction on each turn.

A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software As implemented in source code. Testing progress by moving outward along the spiral.

To integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the

software that has been constructed. Finally we arrive at system testing, where the software and other system elements are tested as a whole.

Unit Testing

The software units in the system is are modules and routines that are assembled and integrated to perform a specific function. As a part of unit testing we executed the program for individual modules independently. This enables, to detect errors in coding and logic that are contained within each of the three module. This testing includes entering data that is filling forms and ascertaining if the value matches to the type and entered into the database. The various controls are tested to ensure that each performs its action as required.

Objective of Unit Testing:

The objective of Unit Testing is:

1. To isolate a section of code.
2. To verify the correctness of code.
3. To test every function and procedure.
4. To fix bug early in development cycle and to save costs.
5. To help the developers to understand the code base and enable them to make changes quickly.
6. To help for code reuse.

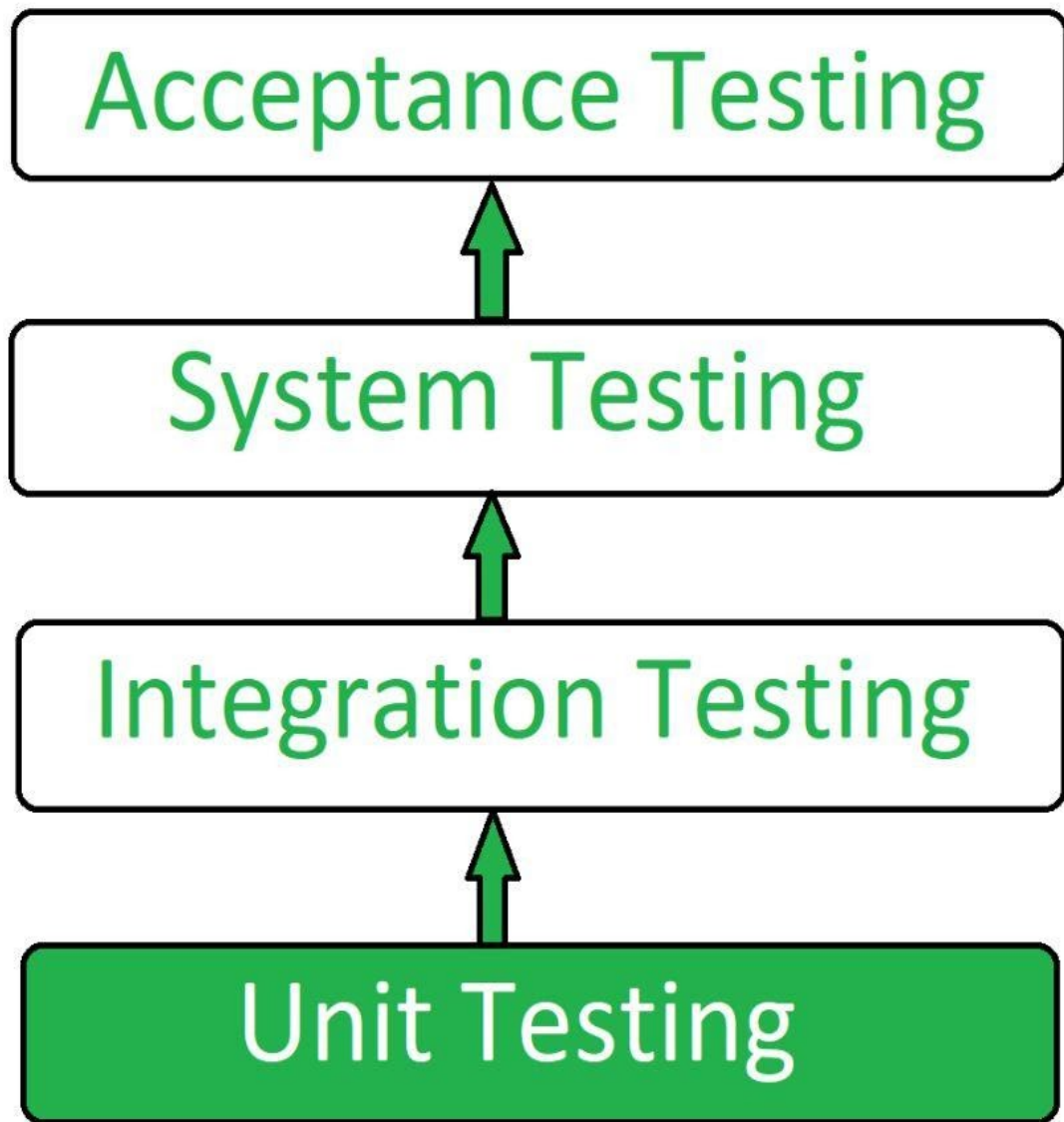


Fig 5.2 Testing Structures

Integration Testing

Data can be lost across any interface, one module can have an adverse effect on another, sub functions when combined, may not produce the desired major functions. Integration testing is a systematic testing to discover errors associated within the interface. The objective is to take unit tested modules and build a program structure. All the modules are combined and tested as a whole. Here the admin module, employee module and student module options are integrated and tested.

This testing provides the assurance that the application is well integrated functional unit with smooth transition of data.

Black Box Testing

Black box testing is defined as a testing technique in which functionality of the Application Under Test (AUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on software requirements and specifications.

Types of Black Box Testing

There are many types of Black Box Testing but the following are the prominent ones -

- **Functional testing** - This black box testing type is related to the functional requirements of a system; it is done by software testers.
- **Non-functional testing** - This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
- **Regression testing** - Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

Tools used for Black Box Testing:

Tools used for Black box testing largely depends on the type of black box testing you are doing.

- For Functional/ Regression Tests you can use – QTP, Selenium
- For Non-Functional Tests, you can use - LoadRunner, Jmeter

Black Box Testing Techniques

Following are the prominent Test Strategy amongst the many used in Black box Testing

- **Equivalence Class Testing:** It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.
- **Boundary Value Testing:** Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable by the system or not. It is very useful in reducing the number of test cases. It is most suitable for the systems where an input is within certain ranges.
- **Decision Table Testing:** A decision table puts causes and their effects in a matrix. There is a unique combination in each column.

5.5 Test Cases:

- **Login Information Maintenance**

Table 5.1 Login Information

The email id should be in “@abc.abc” form.
Password Should be strong and length should greater than 6 characters.
Login id can’t be null.
Password can’t be null.

- **SignUp Information Maintenance**

Table 5.2 SignUp Information

The email id should be in “@abc.abc” form.
Password Should be strong and length should greater than 6 characters.
Firstname and Lastname cant be null.
All fields are Compulsory

- **Files Information Maintenance**

Table 5.3 Manager Information

File must be in .jpeg, .jpg, .png, .doc or .pdf extensions
--

Chapter – 6

LIMITATIONS

- Time and financial constraints were the major factors that hindered the progress of this project.
- To produce a comprehensive research report within the time limit was tiresome. Nevertheless, the quality of this study was not compromised.
- The study involved a lot of financial obligations such as the cost of stationary, printing, photocopying and transportation.
- Although we have put best effort to create the app flexible easy to operate but the limitation cannot be ruled out by us.
- The software have board range of options for its customers, some intricate options could not be covered into it.
- Lack of time also compelled us to ignore some parts such as maintaining the record of staff working in the mess and about their behaviour and rating.
- Although we have try our level best to make web-based app such a simple so that it is easy to operate for the customer who is not related to computer field but it is acknowledge that a few customer find it a bit problematic at the first instance.

Chapter – 7

Future Scope

- We have left all the option open so that if there is any future requirement in the system by the user for the enhancement of the system then it is possible to implement them.
- We can also add the different extensions file rather than only sticking to image and doc format.
- It is easy to upload the data in cloud and backing up for future access.
- Adding features for the notification for each and every action perform by admin like about the new updates

Chapter – 8

Conclusion

To conclude the description about the project:

- The project is based on the requirement specification of the user and the analysis of the existing system, with flexibility for future enhancement.
- The expanded functionality of today's software requires an appropriate approach towards software development.
- s3Uploader is designed for the customer who wants to keep track on their past data.
- The numbers of online cloud platform are increasing day by day and they all try to provide best facility to customer and hence all thing are becoming digital therefore we provide facility for the customer such as :-
 - 1) Easy way to upload the files from the device.
 - 2) It will also help to download the uploaded files directly from the User profile.
 - 3) Users can see the file just on hover over the file icon without even downloading it.
- The project is developed using NodeJS, JavaScript, ReactJS and SQL and Postgresql.

Chapter – 9

Bibliography and References

Bibliography

- Pressman, Roger Setal, “Analysis modelling” “Software Engineering”, PP299-334, McGraw-Hill, New York, Fifth edition.
- Pressman, Roger S. Etal, “Software Testing Strategies” “Software Engineering”, PP-477-498, McGraw-Hill, New York, Fifth edition.
- Wilson, Rodney, C. , etal, “Data Flow Diagram” “Software Documentation”, PP-293-315, MIT Press, Cambridge, London, Third edition.
- Lyons, J., etal, “Software Requirement Specification” “Key to Software Engineering”, PP-105-150, Cambridge University Press, New York.

References

- R. Groff and P. N Weinberg. Complete Reference SQL Second Edition.
- Udemy Tutorial React - The Complete Guide (incl Hooks, React Router, Redux) Created by Academind by Maximilian Schwarzmüller, Maximilian Schwarzmüller
- <https://sequelize.org/v5/>
- <https://aws.amazon.com/s3/>
- <https://mochajs.org/>