<Title>

In partial fulfillment for the award of the degree
of
BACHELOR OF TECHNOLOGY
in
COMPUTER SCIENCE AND ENGINEERING

A PROJECT REPORT

Submitted By

<NAME> (Reg. No:)

<NAME> (Reg. No:)

<NAME> (Reg. No:

<NAME> (Reg. No:)

Under the Guidance of

Mr. T. BALASATHURAGIRI, ME.,

ASSISTANT PROFESSOR, DEPARTMENT OF CSE

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PONNAIYAH RAMAJAYAM INSTITUTE OF SCIENCE AND TECHNOLOGY

THANJAVUR - 613 403

MAY - 2024

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

PONNAIYAH RAMAJAYAM INSTITUTE OF SCIENCE AND TECHNOLOGY

THANJAVUR - 613 403

BONAFIDE CERTIFICATE

This is to certify that the project titled **"<TITLE>"** is a bonafide record of work done by "**<NAME> (Reg. No:),**
"**<NAME> (Reg. No:),,** "**<NAME> (Reg. No:),** "**<NAME> (Reg. No:)** in partial fulfillment of the requirements for the Project Report of the degree of **BACHELOR OF TECHNOLOGY** in "**COMPUTER SCIENCE AND ENGINEERING"** of **PONNAIYAH RAMAJAYAM INSTITUTE OF SCIENCE AND TECHNOLOGY,** Thanjavur.

Internal Guide

Head of the Department

Submitted for the University Viva-Voce examination held on ----------

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

First of all, we offer our humble thanks at the sacred feet of the Lord Almighty, by whom this project was made successful.

We are grateful to our beloved **CHANCELLOR, Dr. M. P. NAGESHWARAN, MBA.**, for all the encouragement and support extended to us during the tenure of this project.

We are grateful to our beloved **VICE CHANCELLOR, Dr. T. V. CHRISTY, ME., Ph.D.**, for his espousal and for having instilled in us the confidence to complete our project on time.

We are grateful to our beloved **DIRECTOR, Dr. V.S. SRINIVASAN, ME., Ph.D.**, for giving us the opportunity to carry out this project.

We sincerely thank our guide **Mr. T. BALASATHURAGIRI, ME., ASSISTANT PROFESSOR**, Department of ECE for her guidance, assistance and co-operation that facilitated the successful completion of this project.

Finally, we thank and feel a deep sense of gratitude to our parents and friends of **"PONNAIYAH RAMAJAYAM INSTITUTE OF SCIENCE AND TECHNOLOGY"** who had been all along encouraging us to endeavor this project.

CHAPTERNO.	CHAPTERNAME	PAGENO

	LIST OF FIGURES	I
	LIST OF SCREENSHOTS	II
	ABBREVIATIONS AND SYMBOLS	III-IV
	ABSTRACT	V
1	INTRODUCTION	1
	1.1 INTRODUCTION	1
	1.1.1 PURPOSE	2
	1.1.2 OBJECTIVES	2
	1.2 LITERATURE REVIEW	3-4
2	SYSTEM ANALYSIS	5
	2.1 EXISTING SYSTEM	5
	2.2 PROPOSED SYSTEM	5
	2.3 FEASIBILITY STUDY	5
	2.3.1 ECONOMICAL FEASIBILITY	6
	2.3.2 TECHNICAL FEASIBILITY	6
	2.3.3 SOCIAL FEASIBILITY	6
3	SYSTEM REQUIREMENTS & SPECIFICATIONS	7
	3.1 REQUIREMENTS ANALYSIS	7
	3.2 FUNCTIONAL REQUIREMENTS	7
	3.3 NON-FUNCTIONAL REQUIREMENTS	7
	3.4 INPUT & OUTPUT DESIGN	7
	3.5 SYSTEM REQUIREMENTS & SPECS	8
	3.5.1 HARDWARE REQUIREMENTS	8
	3.5.2 SOFTWARE REQUIREMENTS	9-10
4	SYSTEM DESIGN	11
	4.1 INTRODUCTION	11-12
	4.2 ARCHITETURE	12
	4.3 DAT FLOW DIAGRAMS	13
	4.4 E-R DIAGRAMS	14
	4.5 CLASS DIAGRAMS	14-18
	4.6 USE-CASE DIAGRAMS	18-20
	4.7 SEQUENCE DIAGRAMS	20-23
	4.8 ACTIVITY DIAGRAMS	24-29
	4.9 UML DIAGRAMS	30
5	IMPLEMENTATION	31
	5.1 MODULE & DESCRIPTION	31
	5.2 TECHNOLOGY DESCRIPTION	31-34
	5.3 CODING	35-36
	5.4 OUTPUT SCREENS	36-37
6	SYSTEM TESTING	38
	6.1 SYSTEM TEST	38
	6.2 TYPES OF TESTING	38
	6.2.1 UNIT TESTING	38

	6.2.2 INTEGRATION TESTING	38
	6.2.3 FUNCTIONAL TEST	39
	6.2.4 SYSTEM TEST	39
	6.2.5 WHITE BOX TESTING	40
	6.2.6 BLACK BOX TESTING	40
	6.2.7 UNIT TESTING	40
	6.3 TEST STRATEGY & APPROACH	40
	6.3.1 TEST OBJECTIVES	40
	6.3.2 FEATURES TO BE TESTED	40-41
	6.3.3 ACCEPTANCE TESTING	41`
7	CONCLUSION	42
8	FUTURE SCOPE	43
9	REFERENCES	44

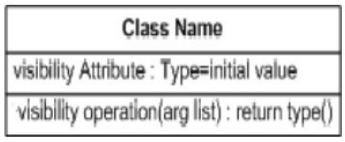
LIST OF FIGURES

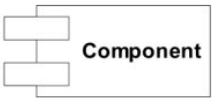
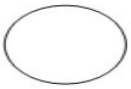
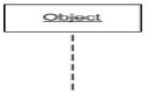
FIGURE NAME	PAGE NO
Architecture Diagram	25
Data flow Diagram	26
E-R Diagram	26
Class Diagrams	27-30
Use-Case Diagrams	30-31
Sequence Diagrams	32-34
Activity Diagrams	35-38
UML-Diagrams	39

LIST OF SCREENSHOTS

SCREENSHOTS	PAGE NO
Output Screenshot 1.1	43
Output screenshot 1.2	44

LIST OF SYMBOLS

S.NO	SYMBOL NAME	SYMBOL	DESCRIPTION
1.	Class		Classes represent acollection of similarentities groupedtogether.
2.	Association		Association representsa static relationshipbetweencl asses.
3.	Aggregation		Aggregation is a formof association. Itaggregatesseveral classes into singleclass.
4.	Actor		Actors are the users ofthe system and otherexternal entity thatreact withthesystem.
5.	Usecase		A use case is ainteraction betweenthe system and theexternalenvironme nt.
6.	Relation(Uses)		It is used foradditional processcommunic ation.
7.	Communication		It is thecommunicationb etween various usecases.
8.	State		It represents the stateofa process. Each state goes throughvariousflo ws.

9.	InitialState		It represents the initialstateof theobject.
10.	**FinalState**		It represents the finalstateof theobject.
11.	**Component**		Components representthe physicalcomponents used inthesystem.
12.	**Node**		Deployment diagramsuse the nodes forrepresenting physicalmodules, which is acollection ofcomponents.
13.	**DataProcess/State**		A circle in DFDrepresents a state orprocess whichhas been triggered due tosomeeventoraction.
14.	**ExternalEntity**		It represent anyexternal entity such askeyboard, sensors etcwhich are used in thesystem.
15.	**Transition**		It represent anycommunication thatoccurs between theprocesses.
16.	**ObjectLifeline**		Object lifelinesrepresents the verticaldimension that objectscommunicates.
17.	**Message**		It represents themessagesexchange d.

ABSTRACT

An online resume builder is software developed to simplify the task of creating a resume for individuals. The application provides an effective means of designing desired resume in fact a professional looking resume. The system is flexible to be used and reduces the need of thinking and designing an appropriate resume according to qualifications. Usually, individuals get confused while creating a resume especially for a novice person such as graduate students. They don't get a clear idea of what things and information must be included in a resume.

Hence the system is developed to provide them an easy way for creating a professional looking resume. This project is user-friendly and requires minimum human intervention. Individuals just have to fill up a form that specifies questions from all required fields such as personal questions, educational, qualities, interest, skills and so on. The answers provided by the users are stored and the system automatically generates a well-structured resume.

The resume builder application made in Django helps to make impressive resumes easily within a few minutes. Building a resume with no help would be a difficult task like deciding layout, text, display necessary field, etc.

So, to overcome these difficulties, we came up with a web resume builder that takes basic information such as name, email address, mobile number, address, some skills, past experience etc. of the aspirant required when applying for a job. It provides professional resumes, instantly generated by the resume builder.

The important aim of this project, help employees to land their dream job with the perfect resumes. The forms for the resume builder are done using crispy forms.

CHAPTER-1

INTRODUCTION

1.1 INTRPDUCTION

A resume is a document used by individuals to present their background and skill sets. A resume also spelled resume or resume also called curriculum vitae or CV. A document that has a brief summary or listing about relevant education and experience. The resume or CV is typically the first item that a potential user encounters regarding the job seeker and is mostly used for screening an applicant's which is often followed by an interview, while seeking employment in the job search process and well-designed resume. The Resume Builder will help user build his/her personal advertisement through Resume Builder system develop a resume builder with job placement system. Many large employers use electronic resume processing systems to handle large number of resumes. Job portal advertisement may direct applicants to email his resume to their company or visit their website and submit a resume in electronic format.

In recent years, there has been continuing trend among youths to pursue higher education in their zeal to become highly qualified and skilled. The new technologies, specially, an internet has made a huge impact on knowledge management and information dissemination in education. In many organizations including universities, the web portal is knowledge - management system is among most popular topics. Universities have been at the forefront of website development, which further led to the development of the web portals to provide more useful links to information resources. Portals have different applications or services to solve various problems. One of the aims of web portals is to allow information access and sharing over the Internet. For e.g., in a university, the new students in the faculty need access to information resources to select different courses and to decide on the different areas and majors available, in the faculty.

This need can be addressed through the knowledge portal which should contain appropriate data about the Requirements of the students and user. The

increased number of jobless youths and graduates has become one of the serious issue existing both in the developing and developed countries, today. The Internet has changed the way of looking for an employment, through the development of online job portals. A job portal is a type of web portal that provides an efficient way for searching the Internet or the web for vacant job positions available. This research will go through various types of web/job portals but will, in exact look, at job portals as a knowledge management system based on a standard framework.

An online resume builder is a software developed to simplify the task of creating a resume for individuals. The application provides an effective means of designing desired resume in fact a professional looking resume. The system is flexible to be used and reduces the need of thinking and designing an appropriate resume according to qualifications. Usually individuals get confused while creating a resume especially for a novice person such as graduate students. They don't get a clear idea of what things and information must be included in a resume. Hence the system is developed to provide them an easy way for creating a professional looking resume.

This project is user-friendly and requires minimum human intervention. Individuals just have to fill up a form that specifies questions from all required fields such as personal questions, educational, qualities, interest, skills and so on. The answers provided by the users are stored and the system automatically generates a well-structured resume. Users have option to create resume in any format and file.

1.1.1 PURPOSE:

Purpose of Online Resume Builder is to provide a way to the customers to design their resumes.

1. Creating resumes online.
2. Customizing the look and details.

1.1.2 OBJECTIVES:

The first step is gathering primary details of user for resume generation. Create an intuitive and easy-to-use interface that allows users to navigate

through the resume-building process seamlessly.

After gathering all details from user, system will generate user's CV.

1.2 LITERATURE REVIEW:

Compared with an application form, job applicants can choose what information to include in their resumes. Some of the options for categories to include are: (a) personal information; (b) personal opening, job objective, career objective, and summary of qualifications; (c) education; (d) work experience; (e) references; (f) scholarships, awards, and honours; (g) hobbies, interests, and extracurricular activities; and (h) willingness to relocate and travel. Next, the empirical research literature regarding the information that should be included in the resume for each of the categories is reviewed.

A large and well-established body of research has suggested that the applicant's name, address, and phone number should be included in the resume (Hornsby & Smith, 1995; Hutchinson, 1984; Hutchinson & Brefka, 1997; Mansfield, 1976; Wells, Spinks, & Hargrave, 1981). Although not focused on stylistic resume issues, a recent study by Burns, Christiansen, Morris, Periard, and Coaster (2014) has provided support for the inclusion of a school email address over a personal email address as their sample of human resource professionals provided more favourable judgments for resumes that included email addresses containing ".Edu".

The advice to include a school email address over a personal email address is especially applicable for student applicants and recent graduates but may also have implications for additional applicants. Specifically, future research should address whether more professional email addresses (e.g., givenname.surname@emailprovider.com) are rated more favourably than less professional email addresses.

Although recent research has provided support for the notion that there is no need for a personal opening in a resume (Burns et al., 2014), including a job objective and/or a career objective has traditionally been found to be important information to include in a resume (Harcourt & Krizan, 1989; Harcourt, Krizan, & Merrier, 1991; Hornsby & Smith, 1995; Hutchinson,

1984; Hutchinson & Brefka, 1997; Schramm & Dortch, 1991). Harcourt and colleagues' (1991) sample of 212 campus recruiters demonstrated a preference for a career objective over a job objective or a combined career and job objective. However, future research is also needed to determine whether including a summary of qualifications is effective if a career objective, a job objective, or a combined career and job objective has already been included.

CHAPTER-2

SYSTEM ANALYSIS

2.1 EXISTING SYSTEM

If someone wants to make his/her resume, then he/she must look for every single detail such as formatting, alignment, designs, patterns. So, it is a very complex process and thus, the productivity of the resume also decreases. The existing system is very complex and making resume via word files and excels is a tough task. Sometimes, while making a resume, a person ends up forgetting some very crucial information while going through all these designs and formats.

Disadvantages:

1. Have much work and not convenient for users.
2. Resume not looks professional

2.2 PROPOSED SYSTEM

In the proposed Online resume builder, a person must only fill the questionnaire asked by the online platform. All the other alignments and formats of the resume are being directly taken care of by the system. Even, the final resume which came out is very professional as per the industry requirement and the candidate can download the resume as per the file format, he/she requires. This is avoiding need of putting manual effort for creating resume.

Advantages:

1. Provides instant resume to individuals.
2. Reduces work and is convenient for users.
3. Provides quick access and is affordable.
4. The system saves time and reduces human efforts.
5. This is avoiding need of putting manual efforts.

2.3 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

2.3.1 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.

2.3.2 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.

2.3.3 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.

CHAPTER-3

SYSTEM REQUIREMENTS & SPECIFICATIONS

3.1 REQUIREMENT ANALYSIS

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well-ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

3.2 FUNCTIONAL REQUIREMENTS

Functional requirement should include function performed by a specific screen outline work-flows performed by the system and other business or compliance requirement the system must meet. Functional requirements specify which output file should be produced from the given file they describe the relationship between the input and output of the system, for each functional requirement a detailed description of all data inputs and their source and the range of valid inputs must be specified. The functional specification describes what the system must do, how the system does it is described in the design specification. If a user requirement specification was written, all requirements outlined in the user requirements specifications should be addressed in the functional requirements.

3.3 NON-FUNCTIONAL REQUIREMENTS

Describe user-visible aspects of the system that are not directly related with the functional behavior of the system. Non-Functional requirements include quantitative constraints, such as response time (i.e. how fast the system reacts to user commands.) or accuracy (.e. how precise are the systems numerical answers.).

3.4 INPUT & OUTPUT DESIGN

3.4.1 INPUT DESIGN

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

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

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow.

3.4.2 OUTPUT DESIGN

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

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

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

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

3.5 SYSTEMS REQUIREMENT AND SPECIFICATION

3.5.1 Hardware Requirements

The most common set of requirements defined by any application or software application for virtual computer applications, also known as hardware, Hardware Requirements list is usually accompanied by a hardware compliance list (HCL), especially if there are applications. The HCL list checks hardware devices that are tested, compatible, and sometimes not compatible with a specific application or application. The following sections discuss various aspects of hardware requirements.

1. RAM: 4GB and Higher
2. Processor: Intel i3 and above

3. Hard Disk: 500GB: Minimum

3.5.2 Software Requirements

Software requirements address the definition of software application requirements and pre-requisites that require computer installation to provide System performance. These prerequisites or requirements are not usually included in the software installation package and need to be installed separately before the software can be installed.

1. Operating system : Windows 7 Ultimate.
2. Coding Language : Python.
3. Front-End : tki nter.
4. Back-End : Python
5. Data Base : csv

CHAPTER-4

SYSTEM DESIGN

4.1 INTRODUCTION

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement has been specified and analysed, system design is the first of the three technical activities -design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage. The purpose of the design phase is to plan a solution of the problem specified by the requirement document.

This phase is the first step in moving from the problem domain to the solution domain. In other words, starting with what is needed, design takes us toward how to satisfy the needs. The design of a system is perhaps the most critical factor affection the quality of the software; it has a major impact on the later phase, particularly testing, maintenance. The output of this phase is the design document. This document is similar to a blueprint for the solution and is used later during implementation, testing and maintenance. The design activity is often divided into two separate phases System Design and Detailed Design.

System Design also called top-level design aims to identify the modules that should be in the system, the specifications of these modules, and how they interact with each other to produce the desired results. At the end of the system design all the major data structures, file formats, output formats, and the major modules in the system and their specifications are decided.

During, Detailed Design, the internal logic of each of the modules specified in system design is decided. During this phase, the details of the data of a module is usually specified in a high-level design description language, which is independent of the target language in which the software will eventually be implemented.

In system design the focus is on identifying the modules, whereas during detailed design the focus is on designing the logic for each of the modules. In other works, in system design the attention is on what components are needed, while in detailed design how the components can be implemented in software is the issue.

Design is concerned with identifying software components specifying relationships among components. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is minimal clearly specified.

During the system design activities, Developers bridge the gap between the requirements specification, produced during requirements elicitation and analysis, and the system that is delivered to the user.

Design is the place where the quality is fostered in development. Software design is a process through which requirements are translated into a representation of software. Specifying software structure and providing blue print for the document phase. Modularity is one of the desirable properties of large systems. It implies that the system is divided into several parts. In such a manner, the interaction between parts is minimal clearly specified. The attention is on what components are needed.

4.2 Architecture

The architecture of a resume builder typically involves several components and layers to handle different functionalities efficiently.

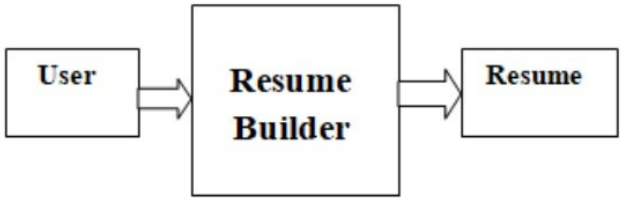


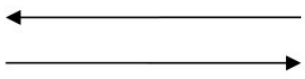
Fig1: Architecture

4.3 Data Flow Diagrams

A graphical tool used to describe and analyse the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also know as a data flow graph or a bubble chart.

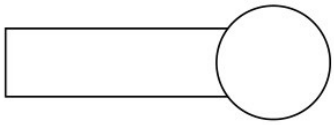
DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system's structure charts. The Basic Notation used to create a DFD's are as follows:

1. Dataflow: Data move in a specific direction from an origin to a destination.



2. Process: People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.

3. Source: External sources or destination of data, which may be People, programs, organizations or other entities.



4. Data Store: Here data are stored or referenced by a process in the System.

4.4 E-R Diagram

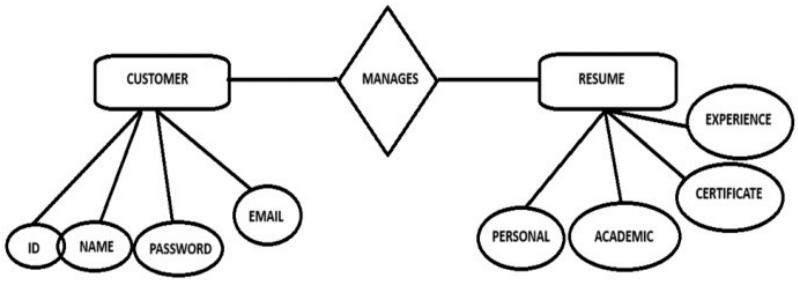


Figure: Entity-Relationship Diagram

4.5 class diagrams

Class diagrams are the backbone of almost every object-oriented method including UML. They describe the static structure of a system.

Basic Class Diagram Symbols and Notations

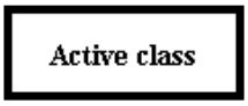
Classes represent an abstraction of entities with common characteristics. Associations represent the relationships between classes.

Illustrate classes with rectangles divided into compartments. Place the name of the class in the first partition (cantered, bolded, and capitalized.

Class Name
attribute:Type = initialValue
operation(arg list):return type

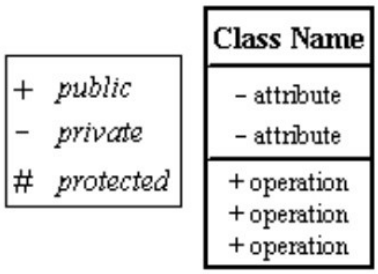
Active Class

Active classes initiate and control the flow of activity, while passive classes store data and serve other classes. Illustrate active classes with a thicker border.



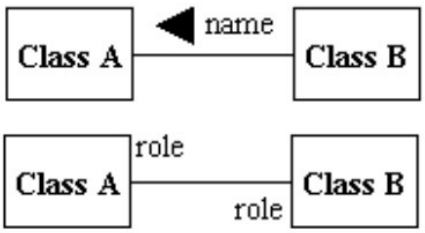
Visibility

Use visibility markers to signify who can access the information contained within a class. Private visibility hides information from anything outside the class partition. Public visibility allows all other classes to view the marked information. Protected visibility allows child classes to access information they inherited from a parent class. .



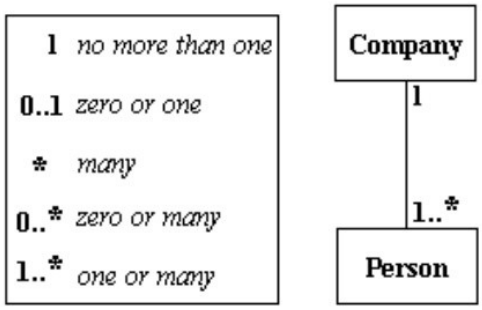
Associations

Associations represent static relationships between classes. Place association names above, on, or below the association line. Use a filled arrow to indicate the direction of the relationship. Place roles near the end of an association. Roles represent the way the two classes see each other. Note: It's uncommon to name both the association and the class roles.



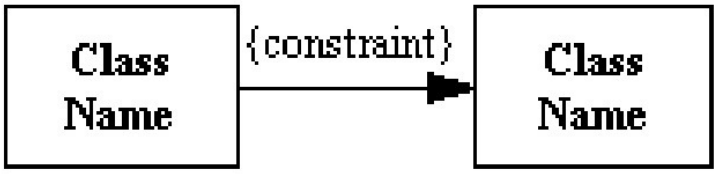
Multiplicity (Cardinality)

Place multiplicity notations near the ends of an association. These symbols indicate the number of instances of one class linked to one instance of the other class. For example, one company will have one or more employees, but each employee works for one company only.

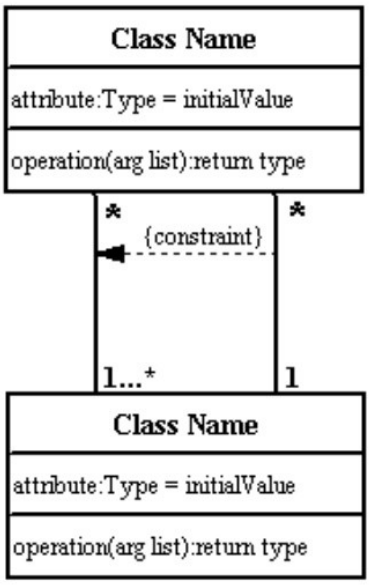


Constraint

Place constraints inside curly braces {}.

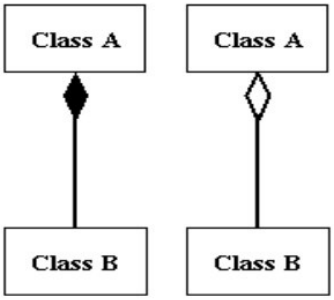


Simple Constraint



Composition and Aggregation:

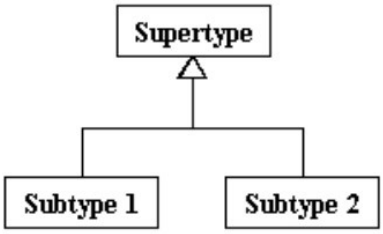
Composition is a special type of aggregation that denotes a strong ownership between Class A, the whole, and Class B, its part. Illustrate composition with a filled diamond. Use a hollow diamond to represent a simple aggregation relationship, in which the "whole" class plays a more important role than the "part" class, but the two classes are not dependent on each other. The diamond end in both a composition and aggregation relationship points toward the "whole" class or the aggregate



Generalization:

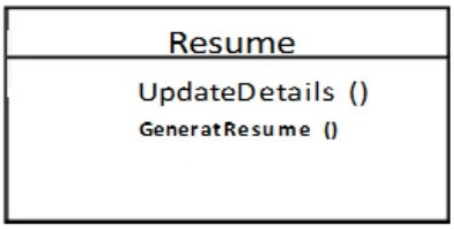
Generalization is another name for inheritance or an "is a" relationship. It refers to a relationship between two classes where one class is a specialized

version of another. For example, Honda is a type of car. So the class Honda would have a generalization relationship with the class car.

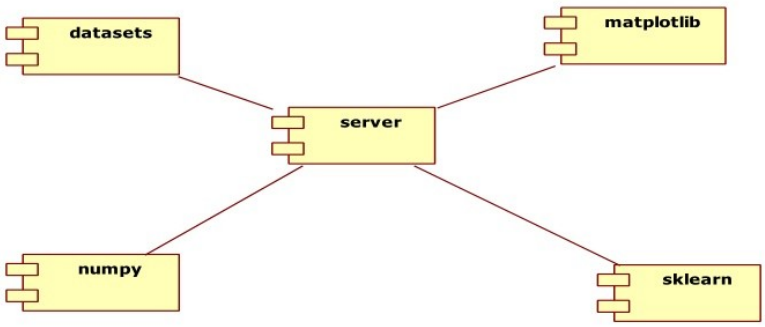


In real life coding examples, the difference between inheritance and aggregation can be confusing. If you have an aggregation relationship, the aggregate (the whole) can access only the PUBLIC functions of the part class. On the other hand, inheritance allows the inheriting class to access both the PUBLIC and PROTECTED functions of the super class.

Class diagram



COMPONENT



4.6 use-case diagrams

Use case diagrams model the functionality of a system using actors and use cases. Use cases are services or functions provided by the system to its users.

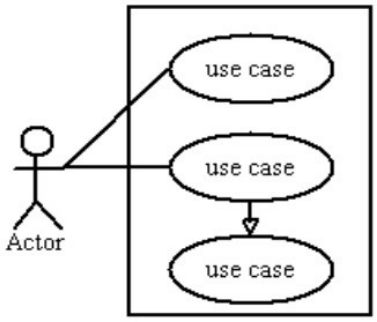
Basic Use Case Diagram Symbols and Notations

System

Draw your system's boundaries using a rectangle that contains use cases. Place actors outside the system's boundaries.

Use cases are services or functions provided by the system to its users.

Basic Use Case Diagram Symbols and Notations.

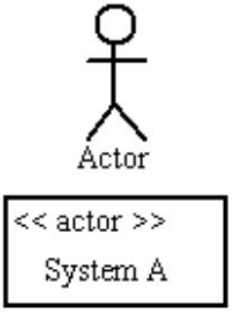


Use Case:

Draw use cases using ovals. Label with ovals with verbs that represent the system's functions.

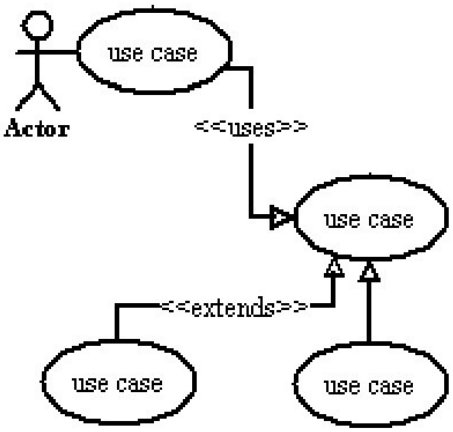
Actors:

Actors are the users of a system. When one system is the actor of another system, label the actor system with the actor stereotype.



Relationships:

Illustrate relationships between an actor and a use case with a simple line. For relationships among use cases, use arrows labelled either "uses" or "extends." A "uses" relationship indicates that one use case is needed by another to perform a task. An "extends" relationship indicates alternative options under a certain use case. In use case diagrams, there are several types of relationships that depict interactions between actors and use cases



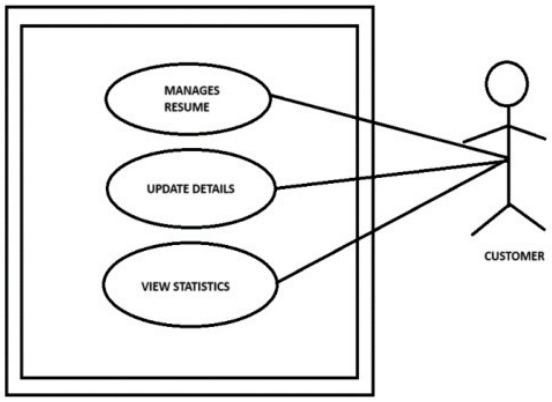


Figure: Use-Case Diagram

4.7 Sequence Diagram

Sequence diagrams describe interactions among classes in terms of an exchange of messages over time.

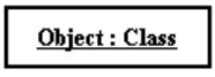
Basic Sequence Diagram Symbols and Notations:

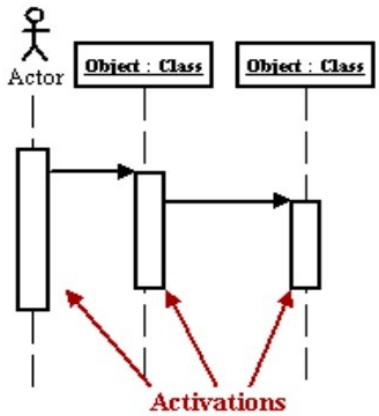
Class roles

Class roles describe the way an object will behave in context. Use the UML object symbol to illustrate class roles, but don't list object attributes.

Activation:

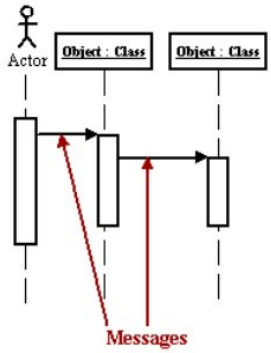
Activation boxes represent the time an object needs to complete a task.





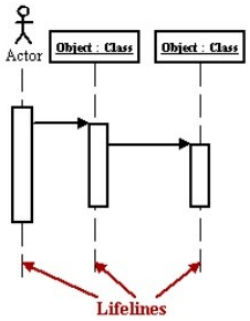
Messages:

Messages are arrows that represent communication between objects. Use half-arrowed lines to represent asynchronous messages. Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its tasks figures & diagrams are below.



Arrow	Message type
→	Simple
→	Synchronous
→	Asynchronous
↩	Balking
→	Time out

Various message types for Sequence and Collaboration diagrams Lifelines. Lifelines are vertical dashed lines that indicate the object's presence over Time.

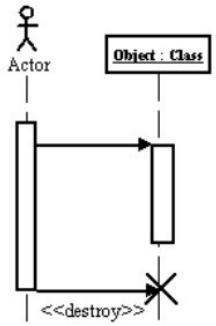


Asynchronous messages are sent from an object that will not wait for a response from the receiver before continuing its Lifelines are vertical dashed lines that indicate the object's presence over time are below.

messages represent communication or interaction between objects or actors within a system. Messages convey information about the flow of control and data between different components or entities involved in a particular scenario. These messages indicate a direct and immediate interaction between sender and receiver, where the sender waits for a response from entities involve in the receiver before proceeding.

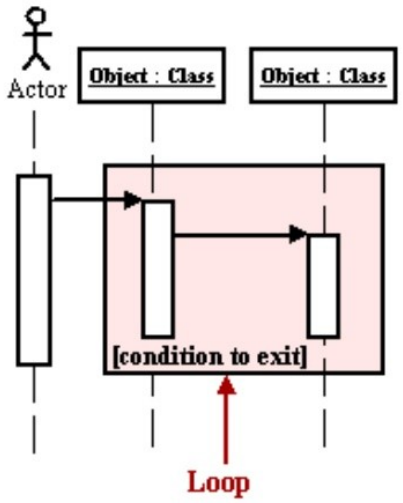
Destroying Objects:

Objects can be terminated early using an arrow labeled "<< destroy >>" that points to an X.



Loops:

A repetition or loop within a sequence diagram is depicted as a rectangle. Place the condition for exiting the loop at the bottom left corner in square brackets [ ].



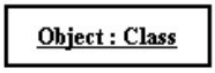
Collaboration Diagram:

A collaboration diagram describes interactions among objects in terms of sequenced messages. Collaboration diagrams represent a combination of information taken from class, sequence, and use case diagrams describing both the static structure and dynamic behavior of a system. Collaboration diagrams focus on the structural organization of objects and the messages exchanged between them to achieve a particular behavior or functionality.

Basic Collaboration Diagram Symbols and Notations:

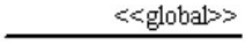
Class roles:

Class roles describe how objects behave. Use the UML object symbol to illustrate class roles, but don't list object attributes.



Association roles:

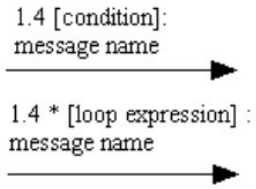
Association roles describe how an association will behave given a particular situation. You can draw association roles using simple lines labeled with stereotypes.



Messages:

Unlike sequence diagrams, collaboration diagrams do not have an explicit way to denote time and instead number messages in order of execution. Sequence numbering can become nested using the Dewey decimal system. For example, nested messages under the first message are labelled 1.1, 1.2, 1.3, and so on. A condition for a message is usually placed in square brackets immediately following the sequence number. Use a * after the sequence number to indicate a loop.

Learn how to add arrows to your lines.



4.8 Activity diagrams

An activity diagram illustrates the dynamic nature of a system by modelling the flow of control from activity to activity. An activity represents an operation on some class in the system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation. Because an activity diagram is a special kind of state chart diagram, it uses the same modelling conventions.

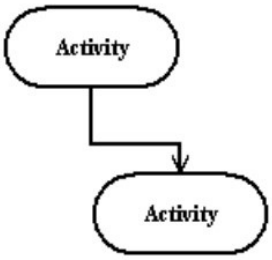
Basic Activity Diagram Symbols and Notations:

Action states:

Action states represent the non-interruptible actions of objects. You can draw an action state in Smart Draw using a rectangle with rounded corners.

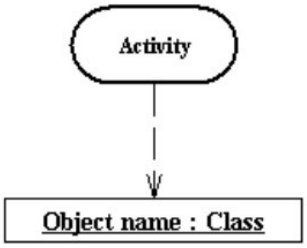
Action Flow:

Action flow arrows illustrate the relationships among action states.



Object Flow:

Object flow refers to the creation and modification of objects by activities. An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.



Initial State:

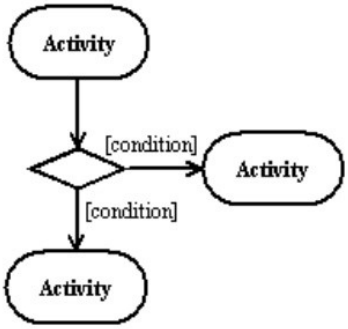
A filled circle followed by an arrow represents the initial action state.

Final State:

An arrow pointing to a filled circle nested inside another circle represents the final action state.

Branching:

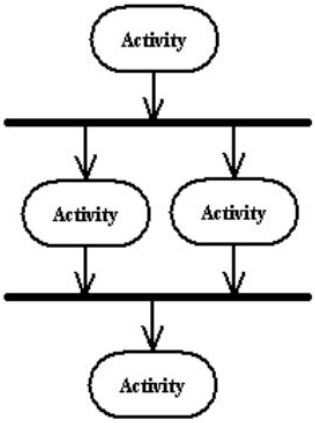
A diamond represents a decision with alternate paths. The outgoing alternates should be labelled with a condition or guard expression. You can also label one of the paths "else."

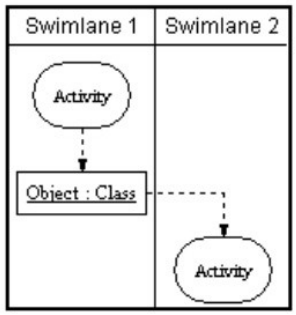


Synchronization

A synchronization bar helps illustrate parallel transitions. Synchronization is also called forking and joining.

An object flow arrow from an action to an object means that the action creates or influences the object. An object flow arrow from an object to an action indicates that the action state uses the object.





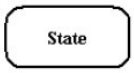
State chart Diagram:

A state chart diagram shows the behavior of classes in response to external stimuli. This diagram models the dynamic flow of control from state to state within a system.

Basic State chart Diagram Symbols and Notations:

States:

States represent situations during the life of an object. You can easily illustrate a state in Smart Draw by using a rectangle with rounded corners.



Transition:

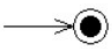
A solid arrow represents the path between different states of an object. Label the transition with the event that triggered it and the action that results from it.

Initial State:

A filled circle followed by an arrow represents the object's initial state.

Final State:

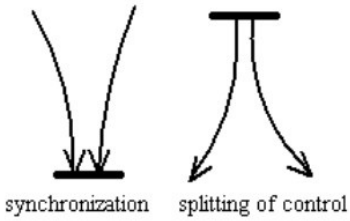
An arrow pointing to a filled circle nested inside another circle represents the object's final state.



Synchronization and Splitting of Control:

A short heavy bar with two transitions entering it represents a synchronization of control. A short heavy bar with two transitions leaving it represents a splitting of control that creates multiple states. Links represent the communication pathways or associations between objects in the system.

Links are represented by lines connecting objects, often labeled with the message or interaction that occurs between them.



STATE CHART DIAGRAM:

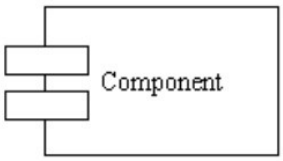
What is a UML Component Diagram?

A component diagram describes the organization of the physical components in a system.

Basic Component Diagram Symbols and Notations:

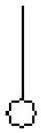
Component:

A component is a physical building block of the system. It is represented as a rectangle with tabs that are all used in the data flow diagrams for better. Learn how to resize grouped objects like components.



Interface:

An interface describes a group of operations used or created by components.

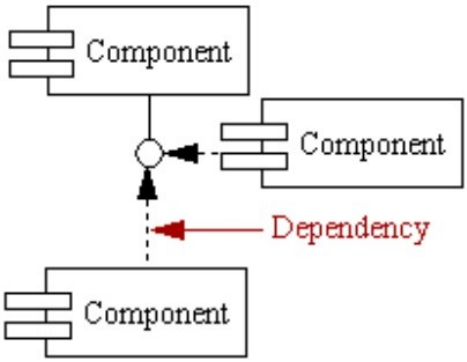


Dependencies:

Draw dependencies among components using dashed arrows. Learn about line_styles in Smart Draw. Messages indicate the interactions or communications between objects within the system.

Messages may include method calls, data exchanges, or other forms of communication between objects.

Dependies are depicted along the links connecting objects, typically labeled with the relevant method or operation.



COMPONENT DIAGRAM

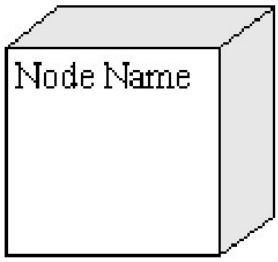
What is a UML Deployment Diagram?

Deployment diagrams depict the physical resources in a system including nodes, components, and connections.

Basic Deployment Diagram Symbols and Notations

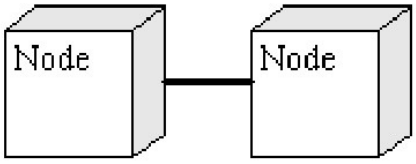
Component

A node is a physical resource that executes code components. Learn how to resize grouped objects like nodes.



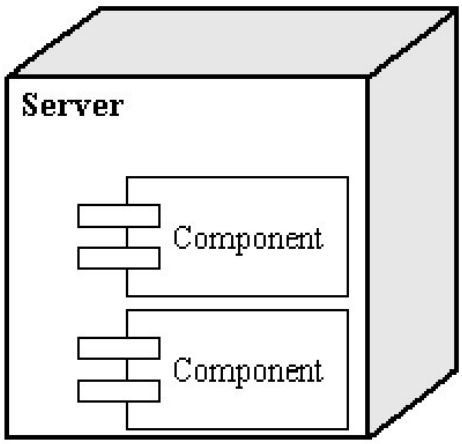
Association:

Association refers to a physical connection between nodes, such as Ethernet. Learn how to connect two nodes. depicted along the links connecting objects, typically labeled with the relevant method or operation.

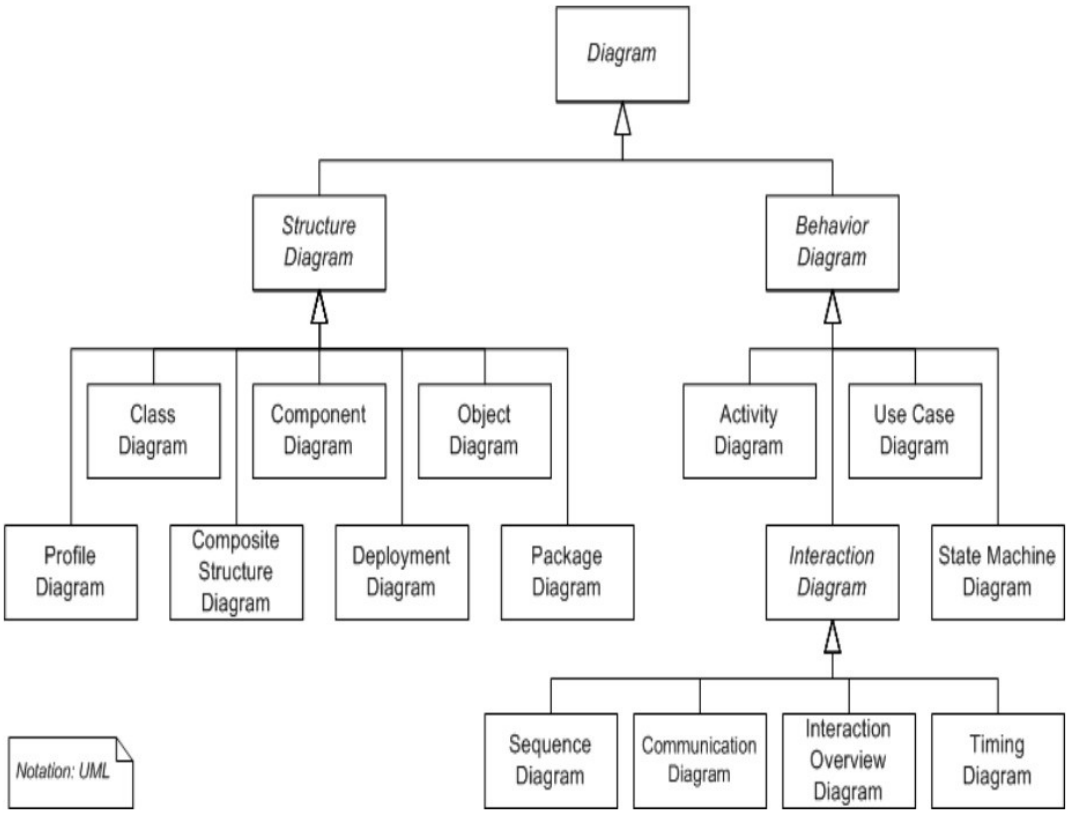


Components and Nodes:

Place components inside the node that deploys them.



4.9 UML Diagrams Overview



UML combines best techniques from data modelling (entity relationship diagrams), business modelling (work flows), object modelling, and component modelling. It can be used with all processes, throughout the software development life cycle, and across different implementation technologies. UML has synthesized the notations of the Booch method, the Object-modelling technique (OMT) and Object-oriented software engineering (OOSE) by fusing them into a single, common and widely usable modelling language.

CHAPTER-5

IMPLEMENTATION

5.1 MODULE & DESCRIPTION

Modules

1. **User**
2. **Resume builder**

Modules description

1. **User:** Users on selecting the format will be given an online form to be filled. The form includes questions like academics, personal, interests, hobbies, skills, courses, experience and so on.
2. **Resume builder:** On submitting the form the system stores the data and within a short period of time generates resume in a format.

5.2 TECHNOLOGY DESCRIPTION

Introduction of Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted_language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. Python, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. Python is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms .

including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library

What is Python

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

1. web development (server-side),
2. software development,
3. mathematics,
4. system scripting.

What can Python do

1. Python can be used on a server to create web applications.
2. Python can be used alongside software to create workflows.
3. Python can connect to database systems. It can also read and modify files.
4. Python can be used to handle big data and perform complex mathematics.
5. Python can be used for rapid prototyping, or for production-ready software development.

Why Python

1. Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
2. Python has a simple syntax similar to the English language.
3. Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
4. Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.

5. Python can be treated in a procedural way, an object-orientated way or a functional way.

Good to know

1. The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
2. In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, PyCharm, NetBeans or Eclipse which are particularly useful when managing larger collections of Python files.

Python Syntax compared to other programming languages

1. Python was designed for readability, and has some similarities to the English language with influence from mathematics.
2. Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
3. Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

DJANGO

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

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes reusability and "pluggability" of components, rapid development, and the principle of don't repeat yourself. Python is used throughout, even for settings files and data models. . Built by experienced developers, it takes care of much of the hassle of Web development, so you

can focus on writing your app without needing to reinvent the wheel.

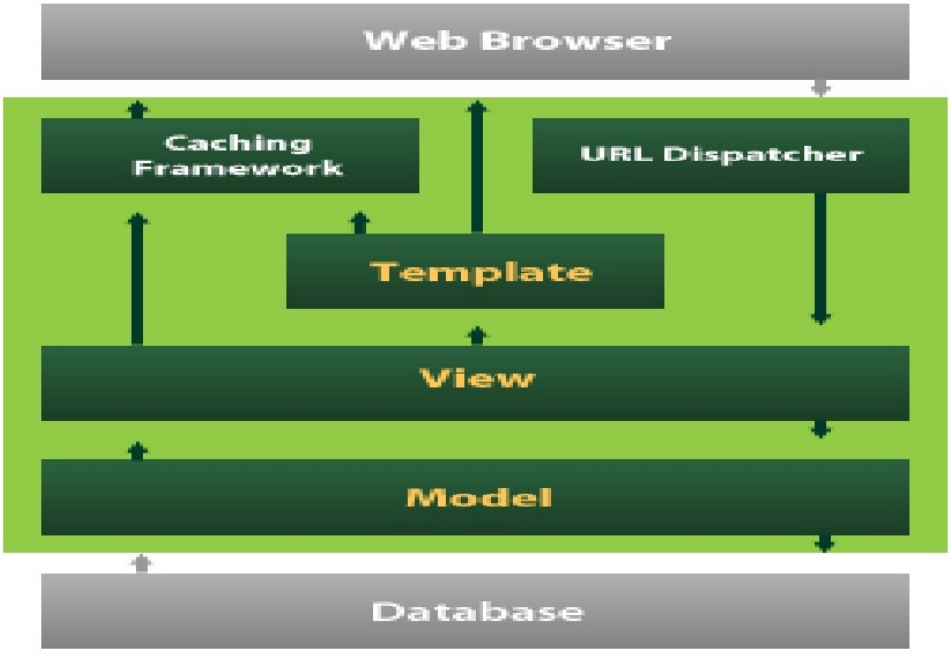


Figure: Model-view-controller

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

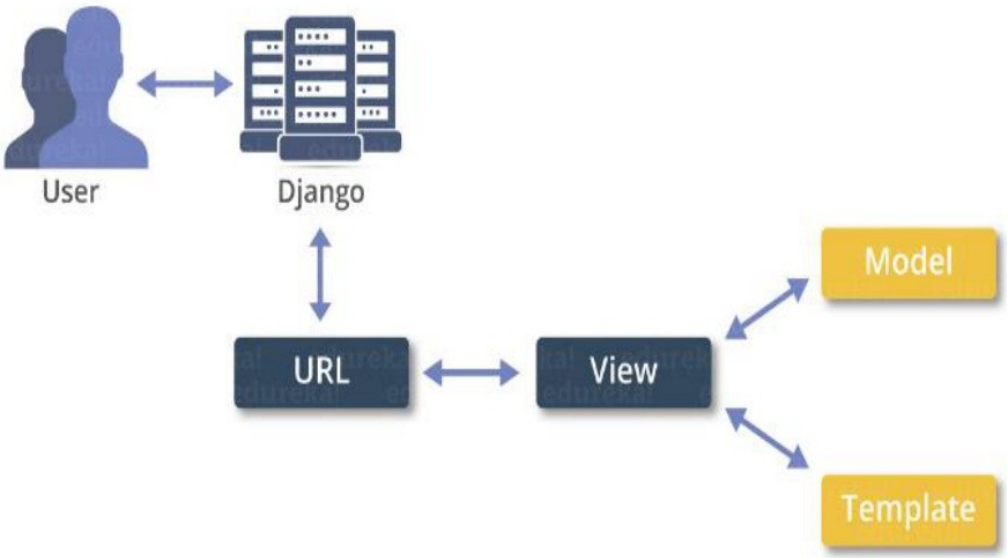


Figure: Django Framework

5.3 CODING

```
from django.shortcuts import render 2 from .forms import ContactForm 3 4 def
home(request): 5 return render(request, 'home.html', {}) 6 7 def info(request): 8
form=ContactForm() 9 if request.method == 'POST': 10
form=ContactForm(request.POST) 11 if form.is_valid(): 12
name=form.cleaned_data['name'] 13 email=form.cleaned_data['email'] 14 15
skills_1=form.cleaned_data['skills_1'] 16 skills_2=form.cleaned_data['skills_2'] 17
skills_3=form.cleaned_data['skills_3'] 18 skills_4=form.cleaned_data['skills_4'] 19
20 mobile=form.cleaned_data['mobile'] 21 address=form.cleaned_data['address']
22 23 experience_1_title=form.cleaned_data['experience_1_title'] 24
experience_1_dur=form.cleaned_data['experience_1_dur'] 25
experience_1_desc=form.cleaned_data['experience_1_desc'] 26 27
experience_2_title=form.cleaned_data['experience_2_title'] 28
experience_2_dur=form.cleaned_data['experience_2_dur'] 29
experience_2_desc=form.cleaned_data['experience_2_desc'] 30 31
education_1=form.cleaned_data['education_1'] 32
education_1_dur=form.cleaned_data['education_1_dur'] 33
education1_score=form.cleaned_data['education1_score'] 34 35
education_2=form.cleaned_data['education_2'] 36
education_2_dur=form.cleaned_data['education_2_dur'] 37
education2_score=form.cleaned_data['education2_score'] 38 39
data={'name':name} 40 data['email']=email 41 data['skills_1']=skills_1 42
data['skills_2']=skills_2 43 data['skills_3']=skills_3 44 data['skills_4']=skills_4 45
46 data['mobile']=mobile 47 data['address']=address 48 49
data['experience_1_title']=experience_1_title 50
data['experience_1_dur']=experience_1_dur 51
data['experience_1_desc']=experience_1_desc 52 53
```

data['experience_2_title']=experience_2_title 54

data['experience_2_dur']=experience_2_dur 55

data['experience_2_desc']=experience_2_desc 56 57

data['education_1']=education_1 58 data['education_1_dur']=education_1_dur 59

data['education1_score']=education1_score 60 61 data['education_2']=education_2

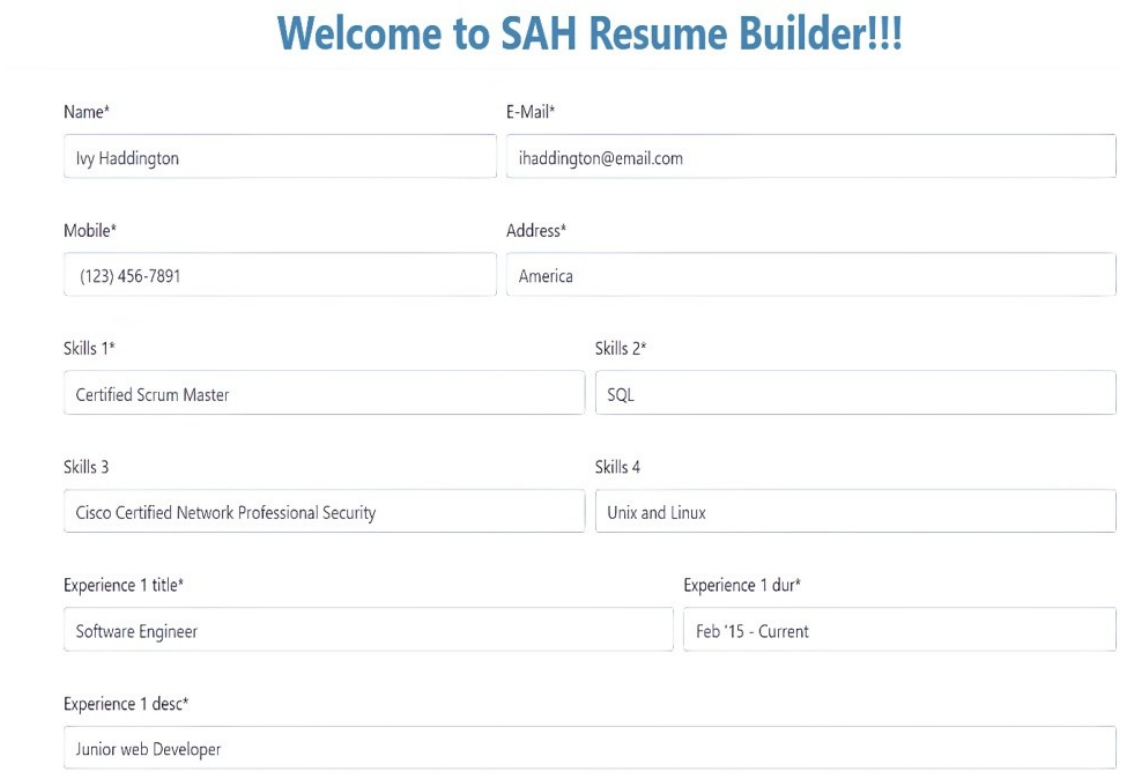
62 data['education_2_dur']=education_2_dur 63

data['education2_score']=education2_score 64 return

render(request,'home.html',data) 65 #to add more go to : forms.py 66 #

print(name,email) 67 return render(request,'info.html',{'form':form}

5.4 OUTPUT SCREENS



Experience 1 desc*

Junior web Developer

Experience 2 title

Experience 2 dur

Experience 2 desc

Education 1*

Masters in Computer

Education 1 dur*

2 years

Education1 score*

Average of 9 Pointer

Education 2*

Bachelor of Science in Computer Science

Education 2 dur*

Aug '99 - Dec '02

Education2 score*

Average of 9.5 Pointer

Submit

Ivy Haddington
SOFTWARE DEVELOPER

e: ihaddington@email.com
m: (123) 456-7891

Personal Profile Address: America

Work Experience

Software Engineer
Feb '15 - Current
Junior web Developer

Key Skills

Certified Scrum Master
SQL
Cisco Certified Network Professional Security
Unix and Linux

Education

Masters in Computer
2 years
Score : Average of 9 Pointer

Bachelor of Science in Computer Science
Aug '99 - Dec '02
Score : Average of 9.5 Pointer

CHAPTER-6

SYSTEM TESTING

6.1 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.

6.2 TYPES OF TESTS

6.2.1Unit 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.

6.2.2 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.

6.2.3Functional 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 cantered on the following items:

1. Valid Input : identified classes of valid input must be accepted.
2. Invalid Input : identified classes of invalid input must be rejected.
3. Functions : identified functions must be exercised.
4. Output : identified classes of application outputs must br 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.

6.2.4System 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. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test.

6.2.5 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.

6.2.6 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.

6.2.7Unit 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.

6.3 Test strategy and approach

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

6.3.1 Test objectives

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

6.3.2 Features to be tested

1. Verify that the entries are of the correct format
2. No duplicate entries should be allowed
3. 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.

6.3.3 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.

CHAPTER-7

CONCLUSION

The project Online Resume Builder is for computerizing the working of building resumes. Resume Builder Application is really an interesting & new topic can be used by freshers or experienced persons to create resume. Using Resume Builder Application users can create a resume with standard format. A Resume Builder Application can run on any operating system of windows. It can be used by freshers or experienced persons to create a resume.

CHAPTER-8

FUTURE SCOPE

In the future, the "Resume Builder" project could be enhanced with several advanced features to further streamline the resume creation process and provide added value to users. One potential improvement could involve integrating machine learning algorithms to analyze job descriptions and suggest personalized content for resumes based on industry trends and employer preferences. Additionally, the platform could offer interactive resume templates with customizable design elements, allowing users to create visually appealing resumes tailored to their preferences. Another valuable addition could be the incorporation of natural language processing capabilities to provide real-time feedback on resume content, helping users optimize their resumes for readability, clarity, and relevance. Furthermore, the platform could expand its functionality to include job search and application management features, enabling users to seamlessly apply for positions directly from their created resumes and track their application progress within the same interface. Overall, these future enhancements would elevate the "Resume Builder" project into a comprehensive career management platform, empowering users to create compelling resumes and navigate the job search process more effectively.

CHAPTER-9

REFERENCES

1. Catano, V. M., Wiesner, W. H., & Hackett, R. D. (2016). Recruitment and selection in Canada (6th ed.). Toronto, ON: Nelson Education Ltd.
2. Derous, E., & Ryan, A. M. (2012). Documenting the adverse impact of resume screening: Degree of ethnic identification matters. International Journal of Selection and Assessment, 20, 464-474. https://doi.org/10.1111/ijsa.12009
3. "Language-Check 0.8: Python Package Index," Pypi.python.org. N.p., 2016. Web. 17 Apr. 2016.
4. Chen, C., Huang, Y., & Lee, M. (2011). Test of a model linking applicant resume information and hiring recommendations. International Journal of Selection and Assessment, 19, 374-387.