**CONTENTS**

1. **INTRODUCTION**
   1. 1.1 INTRODUCTION TO PROJECT
   2. EXISTING SYSTEM & ITS DISADVANTAGES

1.3 PROPOSED SYSTEM & ITS ADVANTAGES

1. **SYSTEM ANALYSIS**

2,1 STUDY OF THE SYSE CINPUT & OUTPUT REPRESENTATION

* 1. PROCESS MODELS USED WITH JUSTIFICATION
  2. SYSTEM ARCHITECTURE

1. **FEASIBILITY STUDY**
   1. TECHNICAL FEASIBILITY
   2. OPERATIONAL FEASIBILITY
   3. ECONOMIC FEASIBILITY
2. **REQUIREMENT SPECIFICATIONS**
   1. FUNCIONAL REQUIREMENTS
   2. PERFORMANCE REQUIREMENTS
   3. SOFTWARE REQUIREMENTS
   4. HARDWARE REQUIREMENTS
      1. INTRODUCTION TO PYTHON
      2. Django
      3. Sqlite
      4. Html, Java Script
      5. Pycharm
3. **SYSTEM DESIGN**
   1. . INTRODUCTION
   2. DATA FLOW DIAGRAMS
   3. UML DIAGRAMS
   4. E-R DIAGRAM
   5. NORMALIZATION
4. **IMPLEMENTAION CODING**
5. **OUTPUT SCREENS**
6. **SYSTEM TESTING** 
   1. INTRODUCTION TO TESTING
   2. TESTING STRATEGIES
7. **SYSTEM SECURITY**

8.1 INTRODUCTION

8.2 SECURITY IN SOFTWARE

**10. BIBLIOGRAPHY**

**1.1 INTRODUCTION & OBJECTIVE**

**Abstract:**

The objective of this application is to develop a system to enable interaction between employers and applicants. The purpose is to allow communication between the interested parties and complete the task of recruitment quickly.

**1.2 Existing system:**

In the present system both job seekers and job providers depend on the services offered by consultancies. Or employers have to directly notify the jobs and conduct interviews by themselves. This procedure results in more time and effort.

**1.3 Proposed System:**

The proposed system is a web application which provides an efficient way to manage the information for various users to cater (provide what is needed) to their needs. It is a complete Portal for Job seekers and employers. It is an exclusive career portal aimed just for the service of job seekers. It is a common platform where corporate recruiters and job seekers come under a single roof.

**2.1 STUDY OF THE SYSTEM**

To provide flexibility to the users, the interfaces have been developed that are accessible through a browser. The GUI’S at the top level have been categorized as

1. Administrative user interface
2. The operational or generic user interface

The ‘administrative user interface’ concentrates on the consistent information that is practically, part of the organizational activities and which needs proper authentication for the data collection. These interfaces help the administrators with all the transactional states like Data insertion, Data deletion and Date updation along with the extensive data search capabilities.

The ‘operational or generic user interface’ helps the end users of the system in transactions through the existing data and required services. The operational user interface also helps the ordinary users in managing their own information in a customized manner as per the included flexibilities

**2.2 INPUT & OUTPOUT REPRESENTETION**

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**INPUT AND OUTPUT DESIGN**

**INPUT DESIGN**

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

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

**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

**OUTPUT DESIGN**

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

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

2. Select methods for presenting information.

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

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

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

**2.3 PROCESS MODEL USED WITH JUSTIFICATION**

**SDLC (Spiral Model):**



SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gathering** **stage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application, define

operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title and

textual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term *requirements traceability*.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator

**Analysis Stage:**

The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.



When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Test Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.



The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category.

For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**2.4 SYSTEM ARCHITECTURE**

**Architecture flow:**

Below architecture diagram represents mainly flow of requests from users to database through servers. In this scenario overall system is designed in three tires separately using three layers called presentation layer, business logic layer and data link layer. This project was developed using 3-tire architecture.

**Business Logic Layer**

**Presentation Layer**

**Request**

**Response**

**Data Link**

**Layer**

**Data Base**

**Feasibility Study**

**Feasibility Study:**

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* Technical Feasibility
* Operation Feasibility

Economical Feasibility

**3.1 TECHNICAL FEASIBILITY**

The technical issue usually raised during the feasibility stage of the investigation includes the following:

* Does the necessary technology exist to do what is suggested?
* Do the proposed equipments have the technical capacity to hold the data required to use the new system?
* Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
* Can the system be upgraded if developed?

Are there technical guarantees of accuracy, reliability, ease of access and data security?

**3.2 OPERATIONAL FEASIBILITY**

**OPERATIONAL FEASIBILITY**

**User-friendly**

Customer will use the forms for their various transactions i.e. for adding new routes, viewing the routes details. Also the Customer wants the reports to view the various transactions based on the constraints. Theses forms and reports are generated as user-friendly to the Client.

**Reliability**

The package wills pick-up current transactions on line. Regarding the old transactions, User will enter them in to the system.

**Security**

The web server and database server should be protected from hacking, virus etc

**Portability**

The application will be developed using standard open source software (Except Oracle) like Java, tomcat web server, Internet Explorer Browser etc these software will work both on Windows and Linux o/s. Hence portability problems will not arise.

**Availability**

This software will be available always.

**Maintainability**

The system called the ewheelz uses the 2-tier architecture. The 1st tier is the GUI, which is said to be front-end and the 2nd tier is the database, which uses My-Sql, which is the back-end.

The front-end can be run on different systems (clients). The database will be running at the server. Users access these forms by using the user-ids and the passwords.

**3.3 ECONOMIC FEASILITY**

The computerized system takes care of the present existing system’s data flow and procedures completely and should generate all the reports of the manual system besides a host of other management reports.

It should be built as a web based application with separate web server and database server. This is required as the activities are spread throughout the organization customer wants a centralized database. Further some of the linked transactions take place in different locations.

Open source software like TOMCAT, JAVA, Mysql and Linux is used to minimize the cost for the Customer.

**4.1 FUNCTIONAL REQUIREMENTS SPECIFICATION**

## Modules:

The system is proposed to have the following modules:

**1. Administrator**

**2. HR head**

**3. Interviewer**

**4. Applicant**

**5. Manager**

**Administrator Module:**

This module provides an interface to add new jobs in Job Street. This module provides functionalities to notify job and walk-in details. This module also includes job search functionality which is very helpful for job seekers to search for jobs. Also it provides facility to filter the jobs based on criteria like Location, Experience and Functional Area. Keyword search is also available. Administrator generates reports like number of Applicants in the web site and number of HR and managers.

**Applicant module:**

Applicant submits his/her Resume to an identified e-mail id with an opportunity id. On receiving the email, extract the mail contents & store it. Based on the opportunity id send it to respective HR (Finance, IM, etc.).Applicant will respond with Acceptance/Rejection subject line. On Rejection process will be ended. On Acceptance new employee ID/mail-id will be generated & process will be ended.

**HR Module**:

HR will verify the resume; if shortlisted, forward it to respective Managers. On rejection, an email will be sent to the Applicant with reason. Manager will update based on the results.

**Manager module:**

Manager will schedule the interview & assign a team member to perform interview. Manager will update based on the results.

**Interviewer module:**

Interviewer conducts the interview & updates the interview results.

**FUTUREWORK:**

There are many open problems that are left for future research. For example, it is both interesting and challenging to investigate how to predict the extremely large values and how to deal with missing data (i.e., breach incidents that are not reported). It is also worthwhile to estimate the exact occurring times of breach incidents. Finally, more research needs to be conducted towards understanding the predictability of breach incidents (i.e., the upper bound of prediction accuracy).

**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.

**REQUIREMENT SPECIFICATION**

**Functional Requirements**

* Graphical User interface with the User.

**Software Requirements**

For developing the application the following are the Software Requirements:

1. Python
2. Django
3. Mysql
4. Wampserver

**Operating Systems supported**

1. Windows 7
2. Windows XP
3. Windows 8

**Technologies and Languages used to Develop**

1. Python

**Debugger and Emulator**

* Any Browser (Particularly Chrome)

**Hardware Requirements**

For developing the application the following are the Hardware Requirements:

* Processor: Pentium IV or higher
* RAM: 256 MB
* Space on Hard Disk: minimum 512MB

**4.4.1. INTRODUCTION TO PYTHON**

**PYTHON**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](https://en.wikipedia.org/wiki/Interpreted_language), Python has a design philosophy that emphasizes code [readability](https://en.wikipedia.org/wiki/Readability) (notably using [whitespace](https://en.wikipedia.org/wiki/Whitespace_character) indentation to delimit [code blocks](https://en.wikipedia.org/wiki/Code_block) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code) than might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B)or [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [CPython](https://en.wikipedia.org/wiki/CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source) software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation). Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library).

Python is a [multi-paradigm programming language](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language). [Object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) and [structured programming](https://en.wikipedia.org/wiki/Structured_programming) are fully supported, and many of its features support [functional programming](https://en.wikipedia.org/wiki/Functional_programming) and [aspect-oriented programming](https://en.wikipedia.org/wiki/Aspect-oriented_programming) (including by [meta programming](https://en.wikipedia.org/wiki/Metaprogramming) and [meta objects](https://en.wikipedia.org/wiki/Metaobject) (magic methods)). Many other paradigms are supported via extensions, including [design by contract](https://en.wikipedia.org/wiki/Design_by_contract) and [logic programming](https://en.wikipedia.org/wiki/Logic_programming).

Python uses [dynamic typing](https://en.wikipedia.org/wiki/Dynamic_typing), and a combination of [reference counting](https://en.wikipedia.org/wiki/Reference_counting) and a cycle-detecting garbage collector for [memory management](https://en.wikipedia.org/wiki/Memory_management). It also features dynamic [name resolution](https://en.wikipedia.org/wiki/Name_resolution_(programming_languages))([late binding](https://en.wikipedia.org/wiki/Late_binding)), which binds method and variable names during program execution.

Python's design offers some support for [functional programming](https://en.wikipedia.org/wiki/Functional_programming) in the [Lisp](https://en.wikipedia.org/wiki/Lisp_(programming_language)) tradition. It has filter(), map(), and reduce() functions; [list comprehensions](https://en.wikipedia.org/wiki/List_comprehension), [dictionaries](https://en.wikipedia.org/wiki/Associative_array), and sets; and [generator](https://en.wikipedia.org/wiki/Generator_(computer_programming)) expressions.[[47]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-59-47) The standard library has two modules (itertools and functools) that implement functional tools borrowed from [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)) and [Standard ML](https://en.wikipedia.org/wiki/Standard_ML).

The language's core philosophy is summarized in the document *The*[*Zen of Python*](https://en.wikipedia.org/wiki/Zen_of_Python) , which includes [aphorisms](https://en.wikipedia.org/wiki/Aphorism) such as:[[49]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-PEP20-49)

* Beautiful is better than ugly
* Explicit is better than implicit
* Simple is better than complex
* Complex is better than complicated
* Readability counts

Rather than having all of its functionality built into its core, Python was designed to be highly [extensible](https://en.wikipedia.org/wiki/Extensibility). This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with [ABC](https://en.wikipedia.org/wiki/ABC_(programming_language)), which espoused the opposite approach.[[31]](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-venners-interview-pt-1-31)

While offering choice in coding methodology, the Python philosophy rejects exuberant syntax (such as that of [Perl](https://en.wikipedia.org/wiki/Perl)) in favor of a simpler, less-cluttered grammar. As [Alex Martelli](https://en.wikipedia.org/wiki/Alex_Martelli) put it: "To describe something as 'clever' is *not* considered a compliment in the Python culture." Python's philosophy rejects the Perl "[there is more than one way to do it](https://en.wikipedia.org/wiki/There_is_more_than_one_way_to_do_it)" approach to language design in favor of "there should be one—and preferably only one—obvious way to do it".

Python's developers strive to avoid [premature optimization](https://en.wikipedia.org/wiki/Premature_optimization), and reject patches to non-critical parts of the [CPython](https://en.wikipedia.org/wiki/CPython) reference implementation that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use [PyPy](https://en.wikipedia.org/wiki/PyPy), a [just-in-time compiler](https://en.wikipedia.org/wiki/Just-in-time_compilation). [Cython](https://en.wikipedia.org/wiki/Cython) is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name—a tribute to the British comedy group [Monty Python](https://en.wikipedia.org/wiki/Monty_Python)—and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a [famous Monty Python sketch](https://en.wikipedia.org/wiki/Spam_(Monty_Python))) instead of the standard [foo and bar](https://en.wikipedia.org/wiki/Foobar).

A common [neologism](https://en.wikipedia.org/wiki/Neologism) in the Python community is *pythonic*, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called *unpythonic*.

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as *Pythonists*, *Pythonistas*, and *Pythoneers*.[[](https://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-AutoNT-27-55)

The largest new feature in Python 3.7 is a new fundamental data type: Unicode strings. Unicode uses 16-bit numbers to represent characters instead of the 8-bit number used by ASCII, meaning that 65,536 distinct characters can be supported.

**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](https://en.wikipedia.org/wiki/Reusability) and "pluggability" of components, rapid development, and the principle of [don't repeat yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself). Python is used throughout, even for settings files and data models.



Django also provides an optional administrative [create, read, update and delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete) interface that is generated dynamically through [introspection](https://en.wikipedia.org/wiki/Introspection_(computer_science)) and configured via admin models



Despite having its own nomenclature, such as naming the callable objects generating the [HTTP](https://en.wikipedia.org/wiki/HTTP) responses "views", the core Django framework can be seen as an [MVC](https://en.wikipedia.org/wiki/Model-view-controller) architecture. It consists of an [object-relational mapper](https://en.wikipedia.org/wiki/Object-relational_mapping) (ORM) that mediates between [data models](https://en.wikipedia.org/wiki/Data_modeling)(defined as Python classes) and a [relational database](https://en.wikipedia.org/wiki/Relational_database) ("**M**odel"), a system for processing HTTP requests with a [web templating system](https://en.wikipedia.org/wiki/Web_template_system)("**V**iew"), and a [regular-expression](https://en.wikipedia.org/wiki/Regular_expression)-based [URL](https://en.wikipedia.org/wiki/Uniform_Resource_Locator) dispatcher ("**C**ontroller").

Also included in the core framework are:

* a lightweight and standalone web server for development and testing
* a form serialization and validation system that can translate between [HTML](https://en.wikipedia.org/wiki/HTML) forms and values suitable for storage in the database
* a template system that utilizes the concept of [inheritance](https://en.wikipedia.org/wiki/Inheritance_(object-oriented_programming)) borrowed from object-oriented programming
* a [caching](https://en.wikipedia.org/wiki/Web_cache) framework that can use any of several cache methods
* support for [middleware](https://en.wikipedia.org/wiki/Middleware) classes that can intervene at various stages of request processing and carry out custom functions
* an internal dispatcher system that allows components of an application to communicate events to each other via pre-defined signals
* an [internationalization](https://en.wikipedia.org/wiki/Internationalization_and_localization) system, including translations of Django's own components into a variety of languages
* a [serialization](https://en.wikipedia.org/wiki/Serialization) system that can produce and read [XML](https://en.wikipedia.org/wiki/XML) and/or [JSON](https://en.wikipedia.org/wiki/JSON) representations of Django model instances
* a system for extending the capabilities of the template engine
* an interface to Python's built-in [unit test](https://en.wikipedia.org/wiki/Unit_test) framework

Django's configuration system allows third party code to be plugged into a regular project, provided that it follows the reusable app conventions. More than 2500 packages are available to extend the framework's original behavior, providing solutions to issues the original tool didn't tackle: registration, search, [API](https://en.wikipedia.org/wiki/Application_Programming_Interface) provision and consumption, [CMS](https://en.wikipedia.org/wiki/Content_Management_System), etc.

This extensibility is, however, mitigated by internal components dependencies. While the Django philosophy implies loose coupling, the template filters and tags assume one engine implementation, and both the auth and admin bundled applications require the use of the internal [ORM](https://en.wikipedia.org/wiki/Object-relational_mapping). None of these filters or bundled apps are mandatory to run a Django project, but reusable apps tend to depend on them, encouraging developers to keep using the official stack in order to benefit fully from the apps ecosystem

Django can be run in conjunction with [Apache](https://en.wikipedia.org/wiki/Apache_HTTP_Server), [Nginx](https://en.wikipedia.org/wiki/Nginx) using [WSGI](https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface), [Gunicorn](https://en.wikipedia.org/wiki/Gunicorn), or [Cherokee](https://en.wikipedia.org/wiki/Cherokee_(Webserver)) using flup (a Python module). Django also includes the ability to launch a [FastCGI](https://en.wikipedia.org/wiki/FastCGI)server, enabling use behind any web server which supports FastCGI, such as [Lighttpd](https://en.wikipedia.org/wiki/Lighttpd) or [Hiawatha](https://en.wikipedia.org/wiki/Hiawatha_(web_server)). It is also possible to use other [WSGI](https://en.wikipedia.org/wiki/Web_Server_Gateway_Interface)-compliant web servers. Django officially supports four database backends: [PostgreSQL](https://en.wikipedia.org/wiki/PostgreSQL), [MySQL](https://en.wikipedia.org/wiki/MySQL), [SQLite](https://en.wikipedia.org/wiki/SQLite), and [Oracle](https://en.wikipedia.org/wiki/Oracle_Database). [Microsoft SQL Server](https://en.wikipedia.org/wiki/Microsoft_SQL_Server) can be used with django-mssql on [Microsoft](https://en.wikipedia.org/wiki/Microsoft) [operating systems](https://en.wikipedia.org/wiki/Operating_systems), while similarly external backends exist for [IBM Db2](https://en.wikipedia.org/wiki/IBM_Db2), [SQL Anywhere](https://en.wikipedia.org/wiki/SQL_Anywhere) and [Firebird](https://en.wikipedia.org/wiki/Firebird_(database_server)). There is a [fork](https://en.wikipedia.org/wiki/Fork_(software_development)) named django-nonrel, which supports [NoSQL](https://en.wikipedia.org/wiki/NoSQL_(concept)) databases, such as [MongoDB](https://en.wikipedia.org/wiki/MongoDB) and [Google App Engine](https://en.wikipedia.org/wiki/Google_App_Engine)'s Datastore.

Django may also be run in conjunction with [Jython](https://en.wikipedia.org/wiki/Jython) on any [Java EE](https://en.wikipedia.org/wiki/Java_EE) application server such as [GlassFish](https://en.wikipedia.org/wiki/GlassFish) or [JBoss](https://en.wikipedia.org/wiki/JBoss). In this case django-jython must be installed in order to provide [JDBC](https://en.wikipedia.org/wiki/JDBC) drivers for database connectivity, which also can provide functionality to compile Django in to a .war suitable for deployment.

[Google App Engine](https://en.wikipedia.org/wiki/Google_App_Engine) includes support for Django version 1.x.x as one of the bundled frameworks.

For developing a Django project, no special tools are necessary, since the source code can be edited with any conventional [text editor](https://en.wikipedia.org/wiki/Text_editor). Nevertheless, editors specialized on [computer programming](https://en.wikipedia.org/wiki/Computer_programming) can help increase the productivity of development, e.g., with features such as [syntax highlighting](https://en.wikipedia.org/wiki/Syntax_highlighting). Since Django is written in Python, text editors which are aware of Python syntax are beneficial in this regard.

[Integrated development environments](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) add further functionality, such as [debugging](https://en.wikipedia.org/wiki/Debugging), [refactoring](https://en.wikipedia.org/wiki/Code_refactoring), and [unit testing](https://en.wikipedia.org/wiki/Unit_testing). As with plain editors, IDEs with support for Python can be beneficial. Some IDEs that are specialized on Python additionally have integrated support for Django projects, so that using such an IDE when developing a Django project can help further increase productivity.

Django is a free and open source Python web framework that helps to develop maintainable and secured websites rapidly. It deals with the difficulties of web development, so you can completely focus on building your application. The Django project's performance, community, and stability have grown excessively over the past decade.

Django is compatible with all the projects irrespective of its size, whether it is small or big one. Because of its scalability, it is ideal for startups and also serves extraordinary for sites with hundreds of users a moment, like or Disqus.

Some of the popular websites functioned by Django web framework are Pinterest, Instagram, Mozilla Support, Reddit Gifts, Coordinato, Chess etc.

**SQLite**

SQLite is an in-process library that implements a [self-contained](https://www.sqlite.org/selfcontained.html), [serverless](https://www.sqlite.org/serverless.html), [zero-configuration](https://www.sqlite.org/zeroconf.html), [transactional](https://www.sqlite.org/transactional.html) SQL database engine. The code for SQLite is in the [public domain](https://www.sqlite.org/copyright.html) and is thus free for use for any purpose, commercial or private. SQLite is the [most widely deployed](https://www.sqlite.org/mostdeployed.html) database in the world with more applications than we can count, including several [high-profile projects.](https://www.sqlite.org/famous.html)

SQLite is an embedded SQL database engine. Unlike most other SQL databases, SQLite does not have a separate server process. SQLite reads and writes directly to ordinary disk files. A complete SQL database with multiple tables, indices, triggers, and views, is contained in a single disk file. The database [file format](https://www.sqlite.org/fileformat2.html) is cross-platform - you can freely copy a database between 32-bit and 64-bit systems or between [big-endian](http://en.wikipedia.org/wiki/Endianness) and [little-endian](http://en.wikipedia.org/wiki/Endianness)architectures. These features make SQLite a popular choice as an [Application File Format](https://www.sqlite.org/appfileformat.html). SQLite database files are a [recommended storage format](https://www.sqlite.org/locrsf.html) by the US Library of Congress. Think of SQLite not as a replacement for [Oracle](http://www.oracle.com/database/index.html) but as a replacement for [fopen()](http://man.he.net/man3/fopen)

SQLite is a compact library. With all features enabled, the [library size](https://www.sqlite.org/footprint.html) can be less than 600KiB, depending on the target platform and compiler optimization settings. (64-bit code is larger. And some compiler optimizations such as aggressive function inlining and loop unrolling can cause the object code to be much larger.) There is a tradeoff between memory usage and speed. SQLite generally runs faster the more memory you give it. Nevertheless, performance is usually quite good even in low-memory environments. Depending on how it is used, SQLite can be[faster than direct filesystem I/O](https://www.sqlite.org/fasterthanfs.html).

SQLite is [very carefully tested](https://www.sqlite.org/testing.html) prior to every release and has a reputation for being very reliable. Most of the SQLite source code is devoted purely to testing and verification. An automated test suite runs millions and millions of test cases involving hundreds of millions of individual SQL statements and achieves [100% branch test coverage](https://www.sqlite.org/testing.html#coverage). SQLite responds gracefully to memory allocation failures and disk I/O errors. Transactions are[ACID](http://en.wikipedia.org/wiki/ACID) even if interrupted by system crashes or power failures. All of this is verified by the automated tests using special test harnesses which simulate system failures. Of course, even with all this testing, there are still bugs. But unlike some similar projects (especially commercial competitors) SQLite is open and honest about all bugs and provides [bugs lists](http://www.sqlite.org/src/rptview?rn=1) and minute-by-minute [chronologies](http://www.sqlite.org/src/timeline) of code changes.

The SQLite code base is supported by an [international team](https://www.sqlite.org/crew.html) of developers who work on SQLite full-time. The developers continue to expand the capabilities of SQLite and enhance its reliability and performance while maintaining backwards compatibility with the [published interface spec](https://www.sqlite.org/c3ref/intro.html), [SQL syntax](https://www.sqlite.org/lang.html), and database [file format](https://www.sqlite.org/fileformat2.html). The source code is absolutely free to anybody who wants it, but [professional support](https://www.sqlite.org/prosupport.html) is also available.

The SQLite project was started on [2000-05-09](https://www.sqlite.org/src/timeline?c=2000-05-29+14:26:00). The future is always hard to predict, but the intent of the developers is to support SQLite through the year 2050. Design decisions are made with that objective in mind.

We the developers hope that you find SQLite useful and we entreat you to use it well: to make good and beautiful products that are fast, reliable, and simple to use. Seek forgiveness for yourself as you forgive others. And just as you have received SQLite for free, so also freely give, paying the debt forward.

# Features Of SQLite

* [Transactions](https://www.sqlite.org/transactional.html) are atomic, consistent, isolated, and durable (ACID) even after system crashes and power failures.
* [Zero-configuration](https://www.sqlite.org/zeroconf.html) - no setup or administration needed.
* [Full-featured SQL](https://www.sqlite.org/fullsql.html) implementation with advanced capabilities like [partial indexes](https://www.sqlite.org/partialindex.html), [indexes on expressions](https://www.sqlite.org/expridx.html), [JSON](https://www.sqlite.org/json1.html), [common table expressions](https://www.sqlite.org/lang_with.html), and [window functions](https://www.sqlite.org/windowfunctions.html). ([Omitted features](https://www.sqlite.org/omitted.html))
* A complete database is stored in a [single cross-platform disk file](https://www.sqlite.org/onefile.html). Great for use as an [application file format](https://www.sqlite.org/appfileformat.html).
* Supports terabyte-sized databases and gigabyte-sized strings and blobs. (See [limits.html](https://www.sqlite.org/limits.html).)
* Small code [footprint](https://www.sqlite.org/footprint.html): less than 600KiB fully configured or much less with optional features omitted.
* Simple, easy to use [API](https://www.sqlite.org/cintro.html).
* Fast: In some cases, SQLite is [faster than direct filesystem I/O](https://www.sqlite.org/fasterthanfs.html)
* Written in ANSI-C. [TCL bindings](https://www.sqlite.org/tclsqlite.html) included. Bindings for dozens of other languages available separately.
* Well-commented source code with [100% branch test coverage](https://www.sqlite.org/testing.html#coverage).
* Available as a [single ANSI-C source-code file](https://www.sqlite.org/amalgamation.html) that is [easy to compile](https://www.sqlite.org/howtocompile.html) and hence is easy to add into a larger project.
* [Self-contained](https://www.sqlite.org/selfcontained.html): no external dependencies.
* Cross-platform: Android, \*BSD, iOS, Linux, Mac, Solaris, VxWorks, and Windows (Win32, WinCE, WinRT) are supported out of the box. Easy to port to other systems.
* Sources are in the [public domain](https://www.sqlite.org/copyright.html). Use for any purpose.
* Comes with a standalone [command-line interface](https://www.sqlite.org/cli.html) (CLI) client that can be used to administer SQLite databases.

## Suggested Uses For SQLite:

* **Database For The Internet Of Things.** SQLite is popular choice for the database engine in cellphones, PDAs, MP3 players, set-top boxes, and other electronic gadgets. SQLite has a small code footprint, makes efficient use of memory, disk space, and disk bandwidth, is highly reliable, and requires no maintenance from a Database Administrator.
* **Application File Format.** Rather than using fopen() to write XML, JSON, CSV, or some proprietary format into disk files used by your application, use an SQLite database. You'll avoid having to write and troubleshoot a parser, your data will be more easily accessible and cross-platform, and your updates will be transactional. **Website Database.** Because it requires no configuration and stores information in ordinary disk files, SQLite is a popular choice as the database to back small to medium-sized websites.
* **Stand-in For An Enterprise RDBMS.** SQLite is often used as a surrogate for an enterprise RDBMS for demonstration purposes or for testing. SQLite is fast and requires no setup, which takes a lot of the hassle out of testing and which makes demos perky and easy to launch.

**PyCharm**

**PyCharm** is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) used in [computer programming](https://en.wikipedia.org/wiki/Computer_programming), specifically for the [Python](https://en.wikipedia.org/wiki/Python_(programming_language)) language. It is developed by the Czech company [JetBrains](https://en.wikipedia.org/wiki/JetBrains).[[2]](https://en.wikipedia.org/wiki/PyCharm#cite_note-2) It provides code analysis, a graphical debugger, an integrated unit tester, integration with [version control systems](https://en.wikipedia.org/wiki/Revision_control) (VCSes), and supports web development with [Django](https://en.wikipedia.org/wiki/Django_(web_framework)).

PyCharm is [cross-platform](https://en.wikipedia.org/wiki/Cross-platform), with [Windows](https://en.wikipedia.org/wiki/Windows), [macOS](https://en.wikipedia.org/wiki/MacOS) and [Linux](https://en.wikipedia.org/wiki/Linux) versions. The Community Edition is released under the [Apache License](https://en.wikipedia.org/wiki/Apache_License),[[3]](https://en.wikipedia.org/wiki/PyCharm#cite_note-community-3) and there is also Professional Edition with extra features, released under a [proprietary license](https://en.wikipedia.org/wiki/Proprietary_software).

**Features**

* Coding assistance and [analysis](https://en.wikipedia.org/wiki/Code_analysis), with [code completion](https://en.wikipedia.org/wiki/Autocomplete), syntax and error highlighting, linter integration, and quick fixes
* Project and code navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages
* Python [refactoring](https://en.wikipedia.org/wiki/Refactoring): including rename, extract method, introduce variable, introduce constant, pull up, push down and others
* Support for web frameworks: [Django](https://en.wikipedia.org/wiki/Django_(web_framework)), [web2py](https://en.wikipedia.org/wiki/Web2py) and [Flask](https://en.wikipedia.org/wiki/Flask_(web_framework))
* Integrated Python [debugger](https://en.wikipedia.org/wiki/Debugger)
* Integrated [unit testing](https://en.wikipedia.org/wiki/Unit_testing), with line-by-line [code coverage](https://en.wikipedia.org/wiki/Code_coverage)
* [Google App Engine](https://en.wikipedia.org/wiki/Google_App_Engine) Python development
* Version control integration: unified user interface for [Mercurial](https://en.wikipedia.org/wiki/Mercurial), [Git](https://en.wikipedia.org/wiki/Git_(software)), [Subversion](https://en.wikipedia.org/wiki/Apache_Subversion), [Perforce](https://en.wikipedia.org/wiki/Perforce) and [CVS](https://en.wikipedia.org/wiki/Concurrent_Versions_System) with changelists and merge

It competes mainly with a number of other Python-oriented IDEs, including [Eclipse](https://en.wikipedia.org/wiki/Eclipse_(software))'s [PyDev](https://en.wikipedia.org/wiki/PyDev), and the more broadly focused [Komodo IDE](https://en.wikipedia.org/wiki/Komodo_IDE).

## Intelligent Coding Assistance

PyCharm provides smart code completion, code inspections, on-the-fly error highlighting and quick-fixes, along with automated code refactorings and rich navigation capabilities.

### Smart Code Navigation

Use smart search to jump to any class, file or symbol, or even any IDE action or tool window. It only takes one click to switch to the declaration, super method, test, usages, implementation, and more.

### Fast and Safe Refactorings

Refactor your code the intelligent way, with safe Rename and Delete, Extract Method, Introduce Variable, Inline Variable or Method, and other refactorings. Language and framework-specific refactorings help you perform project-wide changes.

## Built-in Developer Tools

PyCharm’s huge collection of tools out of the box includes an integrated debugger and test runner; Python profiler; a built-in terminal; integration with major VCS and built-in database tools; remote development capabilities with remote interpreters; an integrated ssh terminal; and integration with Docker and Vagrant.

## Web Development

In addition to Python, PyCharm provides first-class support for various Python web development frameworks, specific template languages, JavaScript, CoffeeScript, TypeScript, HTML/CSS, AngularJS, Node.js, and more.

## Scientific Tools

PyCharm integrates with IPython Notebook, has an interactive Python console, and supports Anaconda as well as multiple scientific packages including Matplotlib and NumPy.

## Customizable and Cross-platform IDE

Use PyCharm on Windows, Mac OS and Linux with a single license key. Enjoy a fine-tuned workspace with customizable color schemes and key-bindings, with VIM emulation available.

**About PYTHON**:

* + 1. **Django**

**INTRODUCTION**

**4.4.3 SQLite**

**Dr.E.F.Codd’s Rules**

These rules are used for valuating a product to be called as relational database management systems. Out of 12 rules, a RDBMS product should satisfy at least 8 rules + rule called rule 0 that must be satisfied.

**RULE 0: Foundation Rule**

For any system to be advertised as, or claimed to be relational DBMS should manage database with in it self, with out using an external language.

**RULE 1: Information Rule**

All information in relational database is represented at logical level in only one way as values in tables.

**RULE 2: Guaranteed Access**

Each and every data in a relational database is guaranteed to be logically accessibility by using to a combination of table name, primary key value and column name.

**RULE 3: Systematic Treatment of Null Values**

Null values are supported for representing missing information and inapplicable information. They must be handled in systematic way, independent of data types.

**RULE 4: Dynamic Online Catalog based Relation Model**

The database description is represented at the logical level in the same way as ordinary data so that authorized users can apply the same relational language to its interrogation as they do to the regular data.

**RULE 5: Comprehensive Data Sub Language**

A relational system may support several languages and various models of terminal use. However there must be one language whose statement can express all of the following:

Data Definitions, View Definitions, Data Manipulations, Integrity, Constraints, Authorization and transaction boundaries.

**RULE 6: View Updating**

Any view that is theoretical can be updatable if changes can be made to the tables that effect the desired changes in the view.

**RULE 7: High level Update, Insert and Delete**

The capability of handling a base relational or derived relational as a single operand applies not only retrieval of data also to its insertion, updating, and deletion.

**RULE 8: Physical Data Independence**

Application program and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access method.

**RULE 9: Logical Data Independence**

Application programs and terminal activities remain logically unimpaired whenever any changes are made in either storage representation or access methods.

**RULE 10: Integrity Independence**

Integrity constraints specific to particular database must be definable in the relational data stored in the catalog, not in application program.

**RULE 11: Distributed Independence**

Whether or not a system supports database distribution, it must have a data sub-language that can support distributed databases without changing the application program.

**RULE 12: Non Sub-Version**

If a relational system has low level language, that low language cannot use to subversion or by pass the integrity rules and constraints expressed in the higher level relational language.

* + 1. **HTML**

Hypertext Markup Language(HTML), the languages of the world wide web(WWW), allows users to produces web pages that included text, graphics and pointer to other web pages (Hyperlinks).

HTML is not a programming language but it is an application of ISO Standard 8879,SGML(Standard Generalized Markup Language),but

Specialized to hypertext and adapted to the Web. The idea behind Hypertext one point to another point. We can navigate through the information based on out interest and preference. A markup language is simply a series of items enclosed within the elements should be displayed.

Hyperlinks are underlined or emphasized works that load to other documents or some portions of the same document.

Html can be used to display any type of document on the host computer, which can be geographically at a different location. It is a versatile language and can be used on any platform or desktop

HTML provides tags(special codes) to make the document look attractive.

HTML provides are not case-sensitive. Using graphics,fonts,different sizes, color, etc.. can enhance the presentation of the document. Anything

That is not a tag is part of the document it self.

**Basic Html Tags**:

<!-- --> Specific Comments.

<A>………</A> Creates Hypertext links.

<B>………</B> Creates hypertext links.

<Big>……..</Big> Formats text in large-font

<Body>…….</Body> contains all tags and text in the Html-document

<Center>……</Center> Creates Text

<DD>………..</DD> Definition of a term.

<TABLE>……</TABLE> creates table

<Td>………..</Td> indicates table data in a table.

<Tr>………..</Tr> designates a table row

<Th>……….</Th> creates a heading in a table.

##### A D V A N T A G E S:-

* + A HTML document is small and hence easy to send over the net.It is small because it does not include formatted information.
  + HTML is platform independent

HTML tags are not case-sensitive.

* + 1. **JAVA SCRIPT**

The Java Script Language

JavaScript is a compact , object-based scripting language for developing client and server internet applications. Netscape Navigator 2.0 interprets JavaScript statements embedded directly in an HTML page. and Livewire enables you to create server-based applications similar to common gateway interface(cgi) programs.

In a client application for Navigator, JavaScript statements embedded in an HTML Page can recognize and respond to user events such as mouse clicks form

Input, and page navigation.

For example, you can write a JavaScript function to verify that users enter valid information into a form requesting a telephone number or zip code . Without any network transmission, an Html page with embedded Java Script can interpret the entered text and alert the user with a message dialog if the input is invalid or you can use JavaScript to perform an action (such as play an audio file, execute an applet, or communicate with a plug-in) in response to the user opening or exiting a page.

**5.1 INTRODUCTION**

Systems design

**Introduction: Systems design** is the process or art of defining the architecture,

components, modules, interfaces, and data for a system to satisfy specified requirements. One

could see it as the application of systems theory to product development. There is some

overlap and synergy with the disciplines of systems analysis, systems architecture and

systems engineering.

DFD’ s

?????????????

**5.3 UML DIAGRAMS**

**Unified Modeling Language**:

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

* + User Model View
    1. This view represents the system from the users perspective.
    2. The analysis representation describes a usage scenario from the end-users perspective.
  + Structural model view
    1. In this model the data and functionality are arrived from inside the system.
    2. This model view models the static structures.
* Behavioral Model View

It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

* Implementation Model View

In this the structural and behavioral as parts of the system are represented as they are to be built.

* Environmental Model View

In this the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

UML is specifically constructed through two different domains they are:

* UML Analysis modeling, this focuses on the user model and structural model views of the system.
* UML design modeling, which focuses on the behavioral modeling, implementation modeling and environmental model views.

Use case Diagrams represent the functionality of the system from a user’s point of view. Use cases are used during requirements elicitation and analysis to represent the functionality of the system. Use cases focus on the behavior of the system from external point of view.

Actors are external entities that interact with the system. Examples of actors include users like administrator, bank customer …etc., or another system like central database.

Diagrams

**5.5. NORMALIZATION**

A Database is a collection of interrelated data stored with a minimum of redundancy to serve many applications. The database design is used to group data into a number of tables and minimizes the artificiality embedded in using separate files. The tables are organized to:

* Reduced duplication of data.
* Simplify functions like adding, deleting, modifying data etc..,
* Retrieving data
* Clarity and ease of use
* More information at low cost

# Normalization

Normalization is built around the concept of normal forms. A relation is said to be in a particular normal form if it satisfies a certain specified set of constraints on the kind of functional dependencies that could be associated with the relation. The normal forms are used to ensure that various types of anomalies and inconsistencies are not introduced into the database.

**First Normal Form:**

A relation R is in first normal form if and only if all underlying domains contained atomic values only.

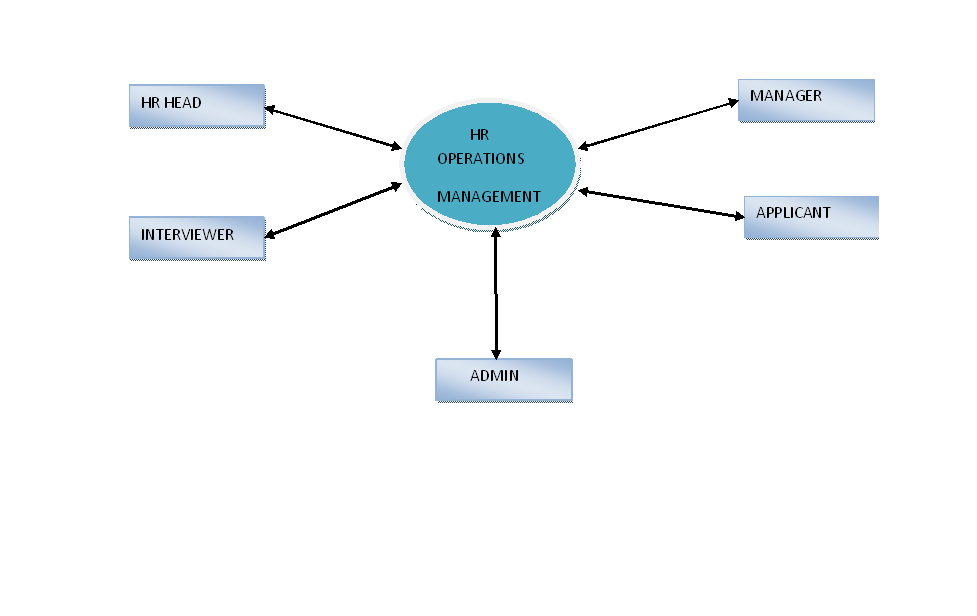
**Second Normal Form:**

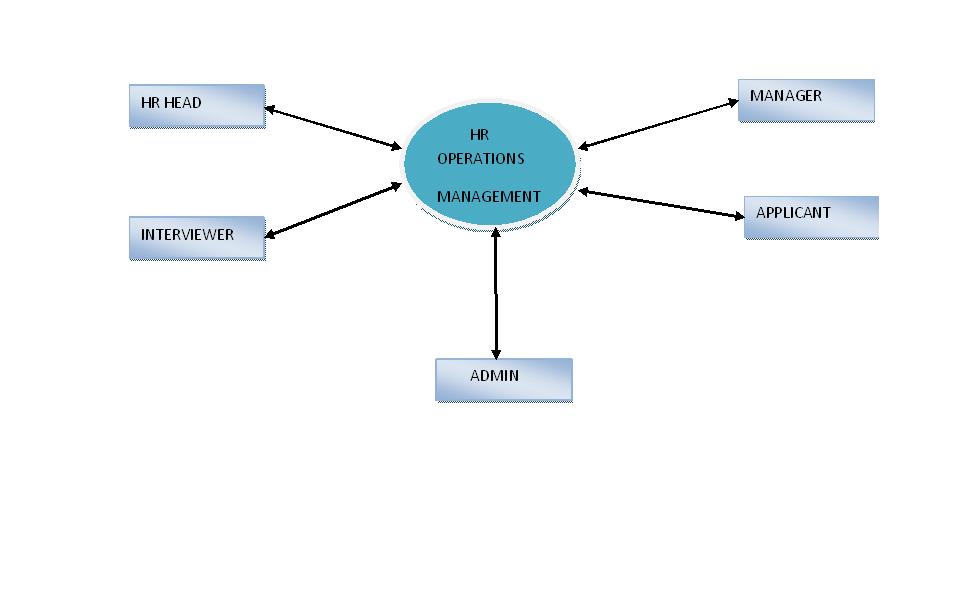
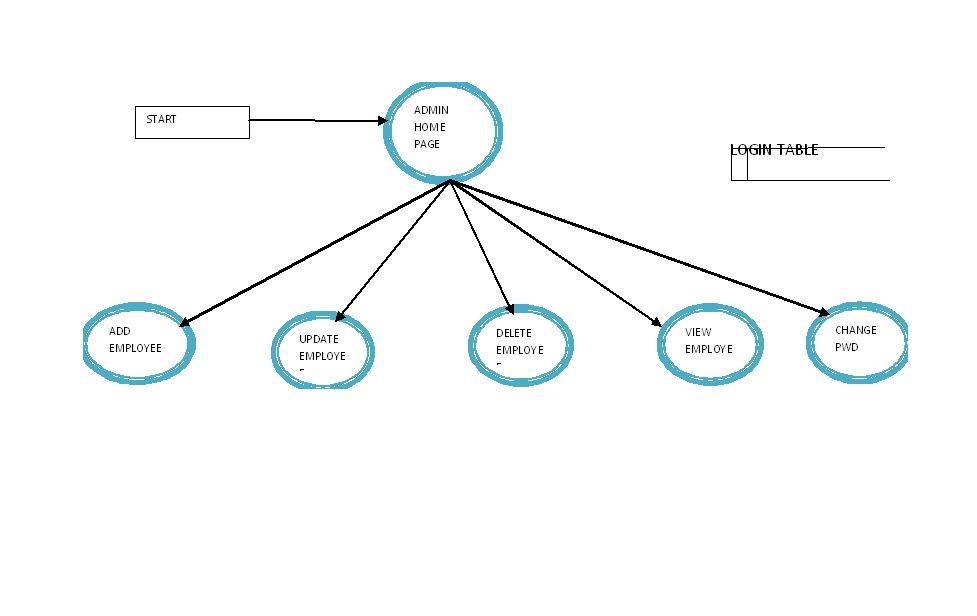
A relation R is said to be in second normal form if and only if it is in first normal form and every non-key attribute is fully dependent on the primary key.

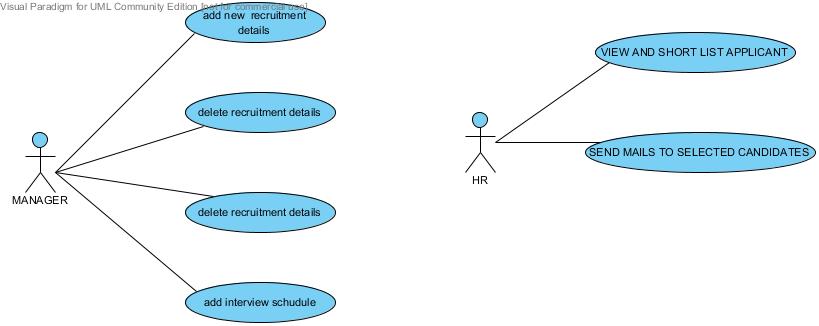
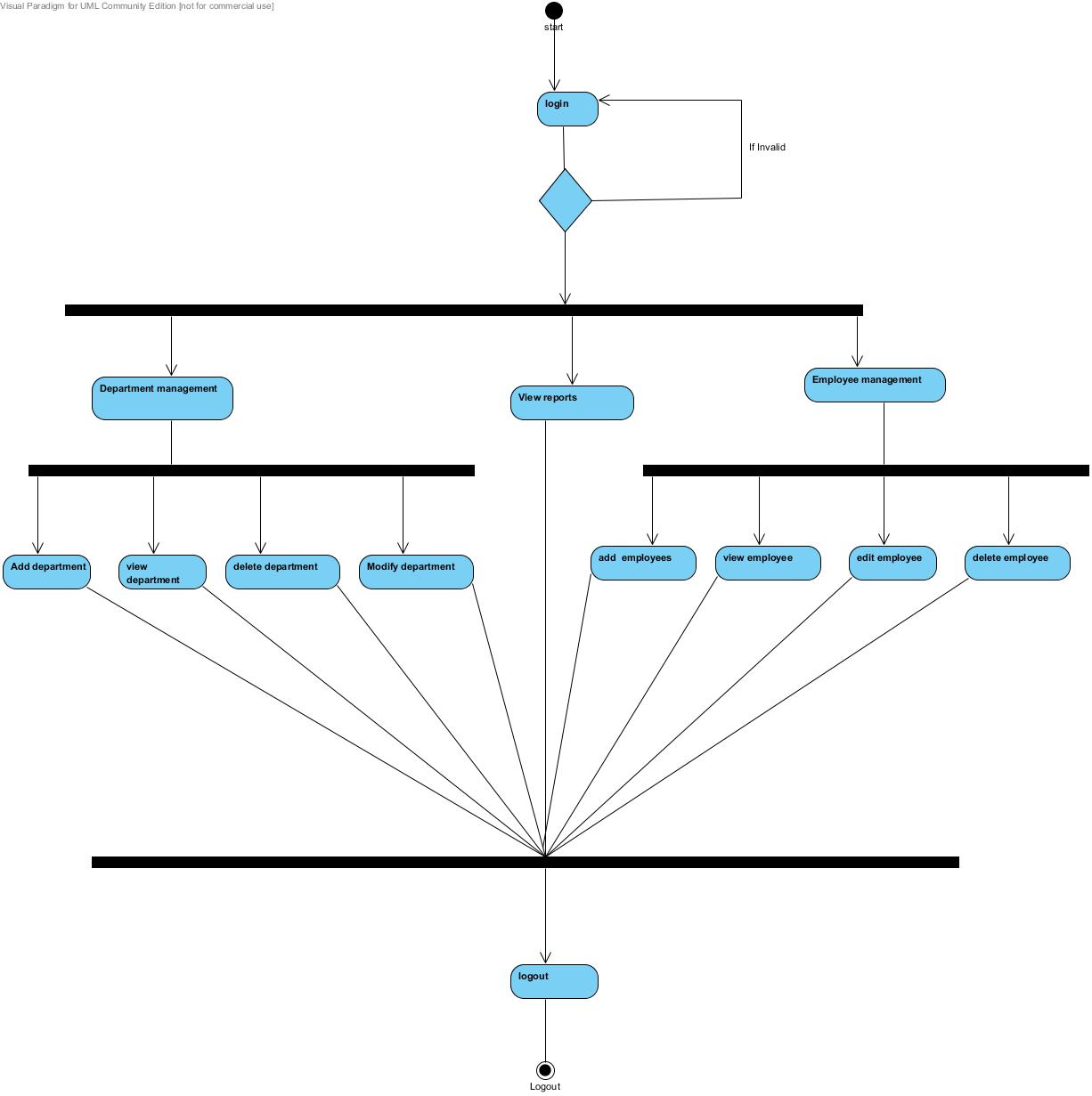
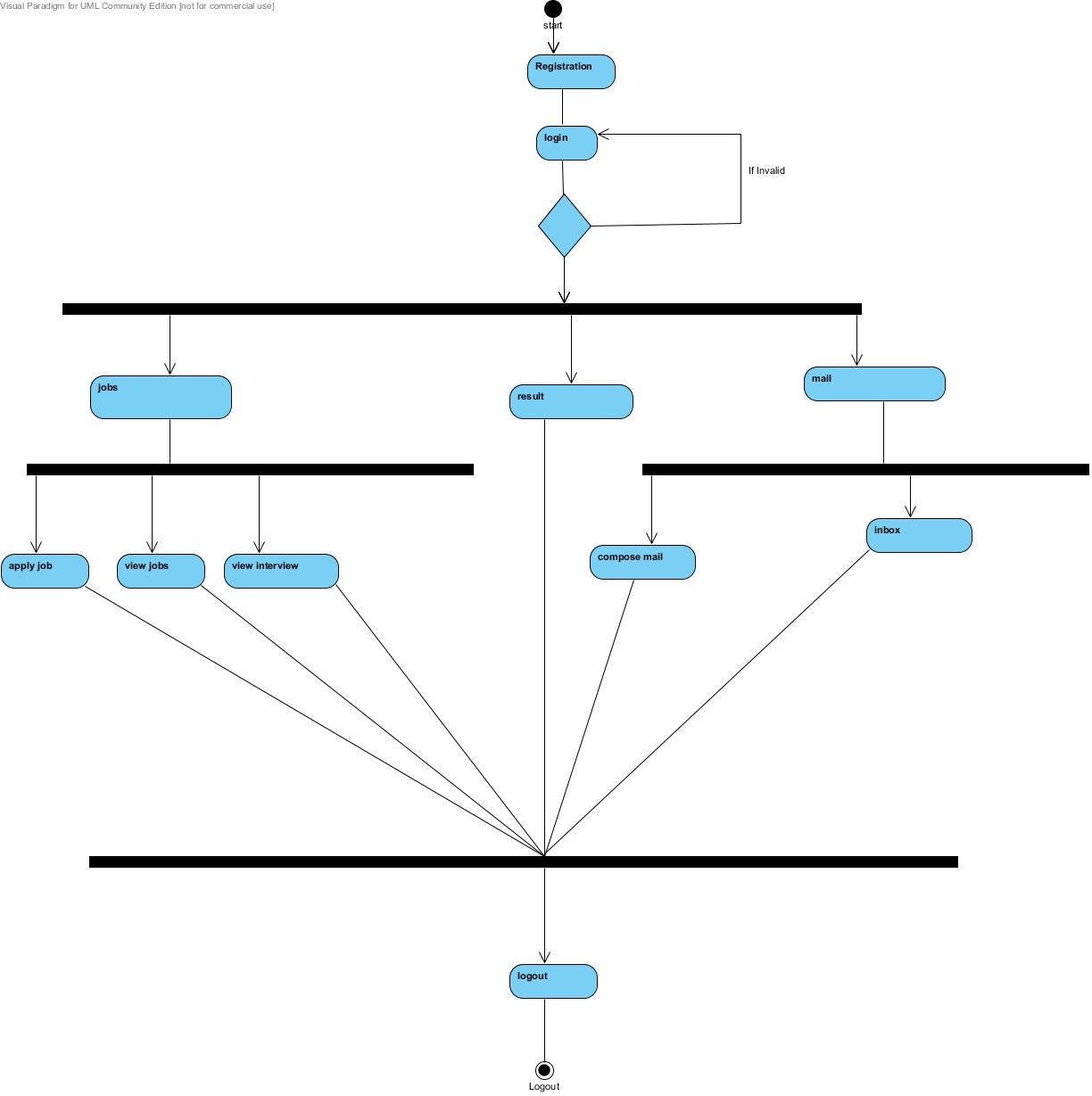
**Third Normal Form:**

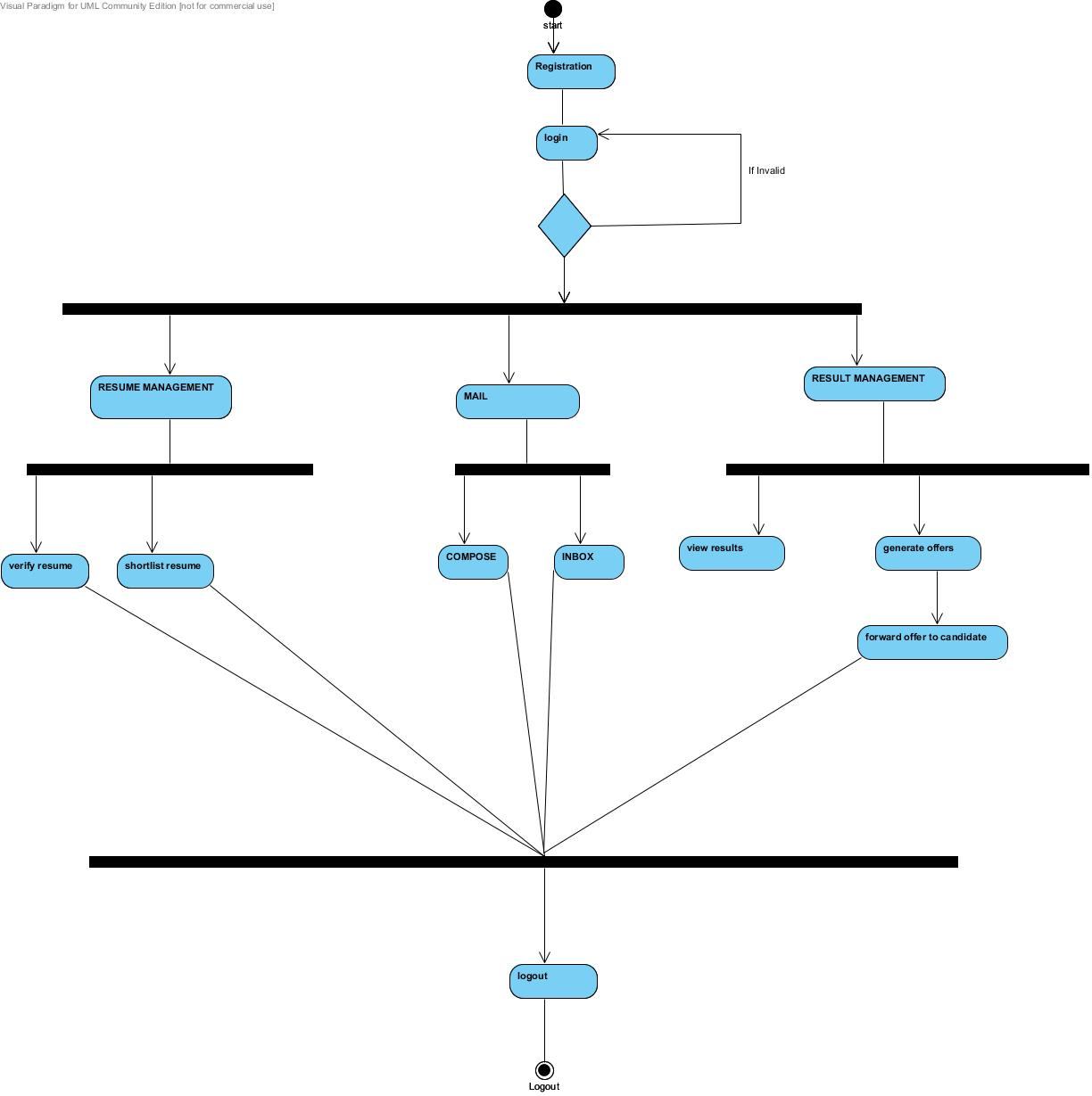
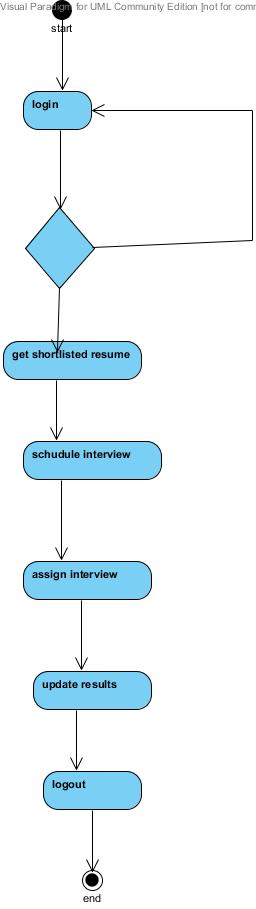
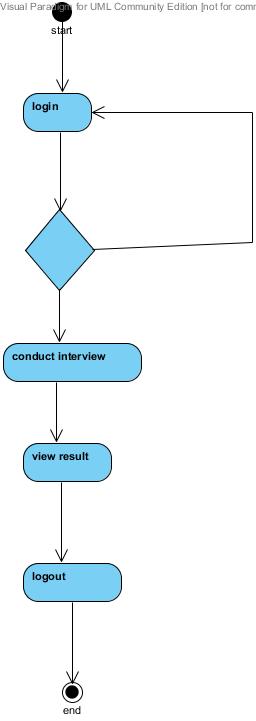
A relation R is said to be in third normal form if and only if it is in second normal form and every non key attribute is non transitively depend on the primary key.

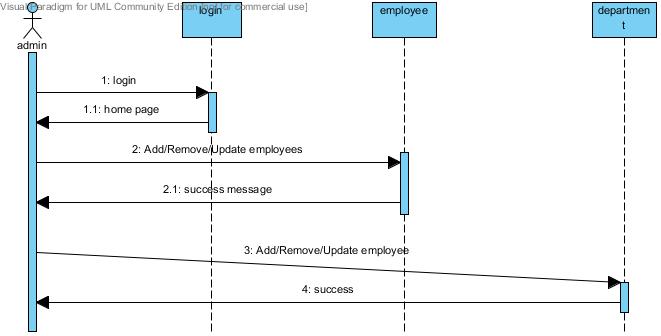
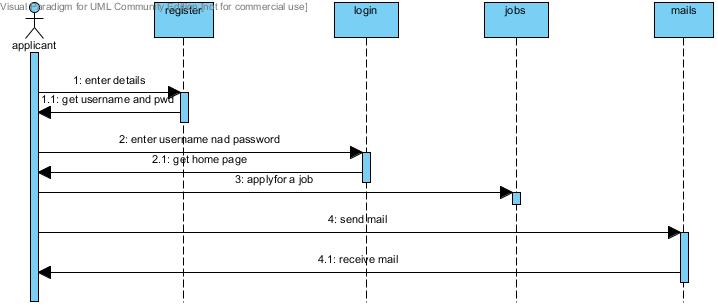
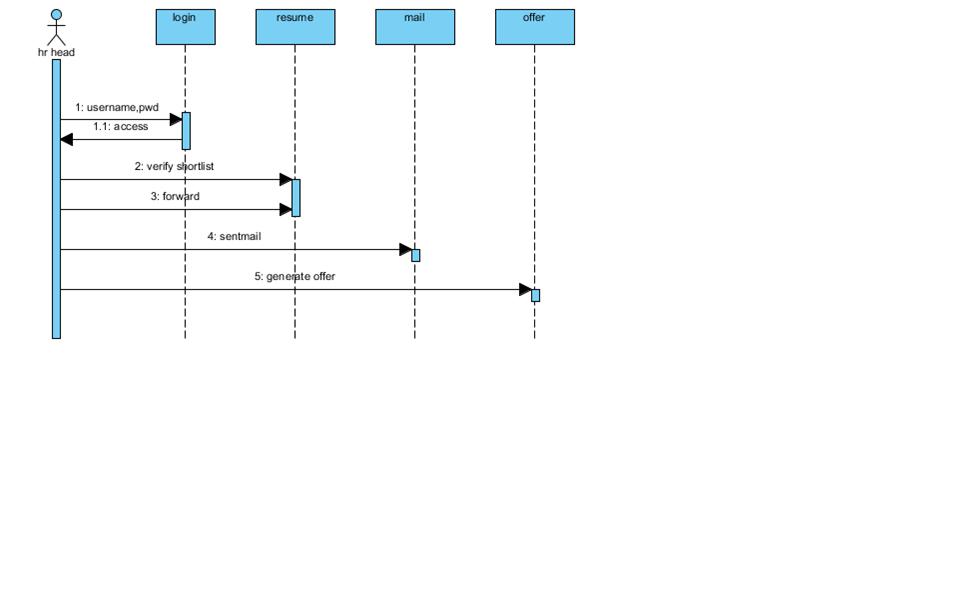
ER-Diagram

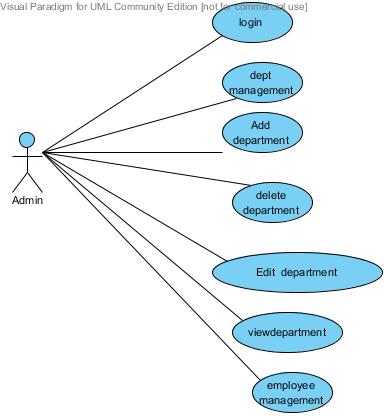
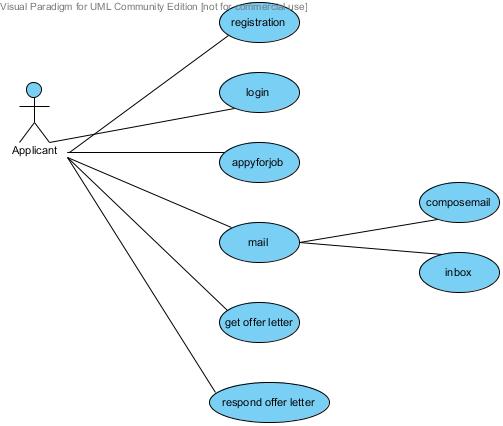
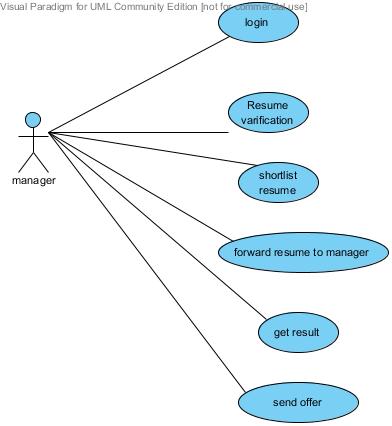
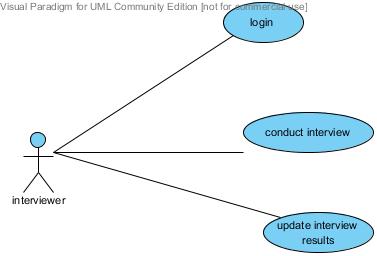
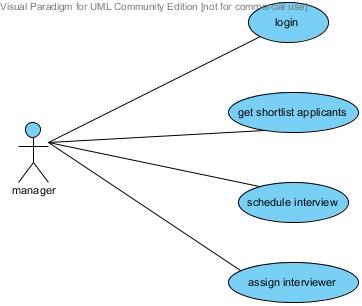
****

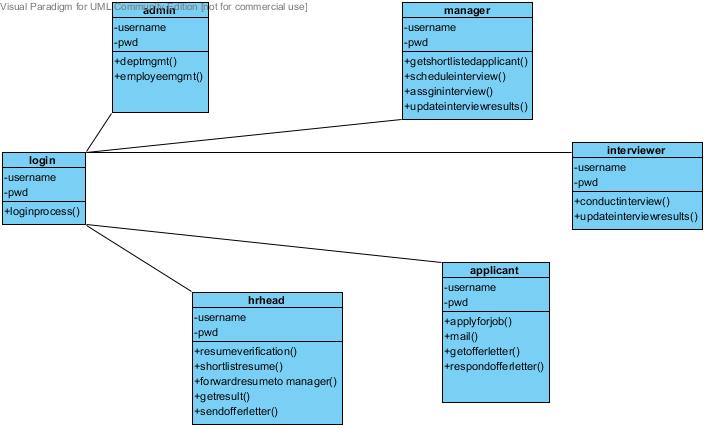
 

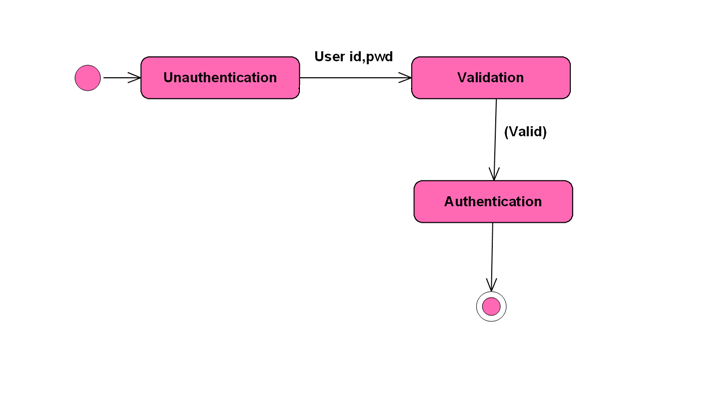
  









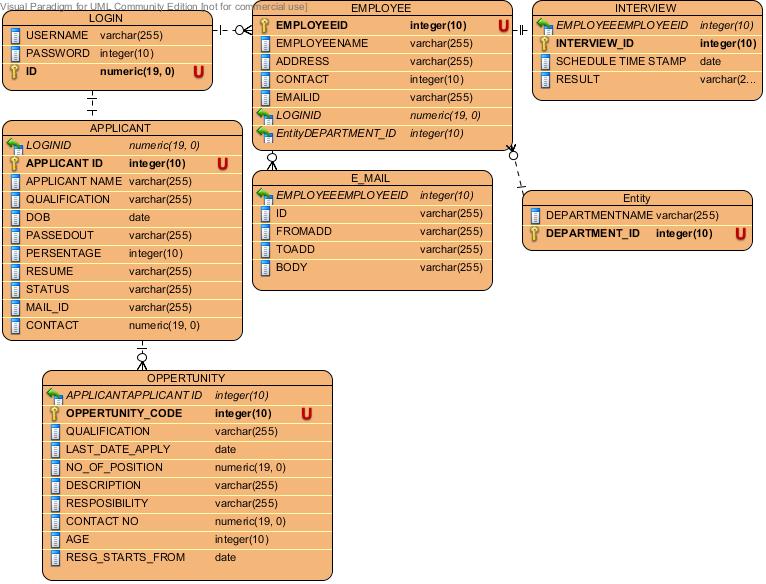
START

IF INVALID

YES

START

LOGIN TABLE



Testing

**INTRODUCTION TO TESTING**

**Introduction to Testing:**

Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During software development. During testing, the program is executed with a set of test cases and the output of the program for the test cases is evaluated to determine if the program is performing as it is expected to perform.

**7.2 TESTING IN STRATEGIES**

In order to make sure that the system does not have errors, the different levels of testing strategies that are applied at differing phases of software development are:

**Unit Testing:**

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

**Each module can be tested using the following two Strategies:**

**Black Box Testing:**

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

* Incorrect or missing functions
* Interface errors
* Errors in data structure or external database access
* Performance errors
* Initialization and termination errors.

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

**White Box testing:**

In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases:

* Guarantee that all independent paths have been Executed.
* Execute all logical decisions on their true and false Sides.
* Execute all loops at their boundaries and within their operational bounds
* Execute internal data structures to ensure their validity.

**Integrating Testing :**

Integration testing ensures that software and subsystems work together a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

**System Testing :**

Involves in-house testing of the entire system before delivery to the user. It's aim is to satisfy the user the system meets all requirements of the client's specifications.

**Acceptance Testing :**

It is a pre-delivery testing in which entire system is tested at client's site on real world data to find errors.

#### Test Approach :

**Testing can be done in two ways:**

* Bottom up approach
* Top down approach

**Bottom up Approach:**

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded with in the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower level ones they are tested individually and then linked with the previously examined lower level modules.

**Top down approach:**

This type of testing starts from upper level modules. Since the detailed activities usually performed in the lower level routines are not provided stubs are written. A stub is a module shell called by upper level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower level module.

**Validation:**

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

**TESTING**

**Test case** : **Login module**

**Purpose:** Login module is being tested here. The Application has login module for all the three (i.e) Admin module, faculty module, Student module.

**Prereq:** The login pass contains two labels, email id and password with the corresponding text fields.

**Test Data:** Providing Invalid email id for admin and testing is performed to check Admin

Homepage is accessed or not.

**Steps:** 1.Go to login Page

2. Enter no email id and password

3. Click on Submit button and checking the validations

4. Enter correct email id and password (i.e) admin and admin

5. Homepage is shown

6. verifying the homepage is Admin Homepage

**Test Screens:** The following screens represent the above test cases

**8.1 INTRODUCTION**

System Security:

Setting Up Authentication for Web Applications

Introduction:

To configure authentication for a Web Application, use the <login-config> element of the web.xml deployment descriptor. In this element you define the security realm containing the user credentials, the method of authentication, and the location of resources for authentication.

**8.2 SECURITY IN SOFTWARE**

To set up authentication for Web Applications:

1. Open the web.xml deployment descriptor in a text editor or use the Administration Console. Specify the authentication method using the <auth-method> element. The available options are:

**BASIC**

Basic authentication uses the Web Browser to display a username/password dialog box. This username and password is authenticated against the realm.

**FORM**

Form-based authentication requires that you return an HTML form containing the username and password. The fields returned from the form elements must be: j\_username and j\_password, and the action attribute must be j\_security\_check. Here is an example of the HTML coding for using FORM authentication:

<form method="POST" action="j\_security\_check">

   <input type="text" name="j\_username">  
   <input type="password" name="j\_password">

</form>

The resource used to generate the HTML form may be an HTML page, a JSP, or a servlet. You define this resource with the <form-login-page> element.

The HTTP session object is created when the login page is served. Therefore, the session.isNew() method returns FALSE when called from pages served after successful authentication.

**9. BIBLIOGRAPHY**

HTML

HTML Black Book by Holzner

JDBC

Java Database Programming with JDBC by Patel moss.

Software Engineering by Roger Pressman



**SYSTEM STUDY**

**FEASIBILITY STUDY**

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

**Three key considerations involved in the feasibility analysis are,**

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

**ECONOMICAL FEASIBILITY**

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

### TECHNICAL FEASIBILITY

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

**SOCIAL FEASIBILITY**

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

**SYSTEM TEST**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

### TYPES OF TESTS

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

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

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

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

**Black Box Testing**

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

**Unit Testing**

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

**Test strategy and approach**

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

**Test objectives**

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

**Features to be tested**

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

# Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

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

**Acceptance Testing**

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

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