Practical – 1

Aim: Study about various software development models.

1.Waterfall Model

2.RAD Model

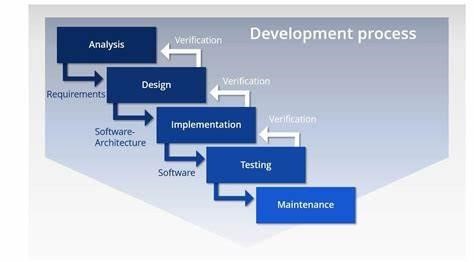
3.Spiral Model

4.Incremental Model

5.Agile Model

# 1. Waterfall Model

The waterfall is a universally accepted SDLC model. In this method, the whole process of software development is divided into various phases.

The waterfall model is a continuous software development model in which development is seen as flowing steadily downwards (like a waterfall) through the steps of requirements analysis, design, implementation, testing (validation), integration, and maintenance.

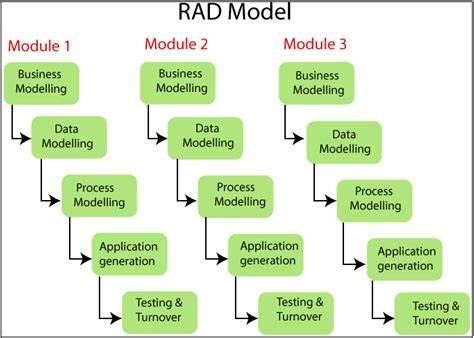
The sequential phases in Waterfall model are −

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

# 2. RAD Model

RAD or Rapid Application Development process is an adoption of the waterfall model; it targets developing software in a short period.

The RAD model is based on the concept that a better system can be developed in lesser time by using focus groups to gather system requirements.



1. Business Modelling
2. Data Modelling
3. Process Modelling
4. Application Generation
5. Testing and Turnover

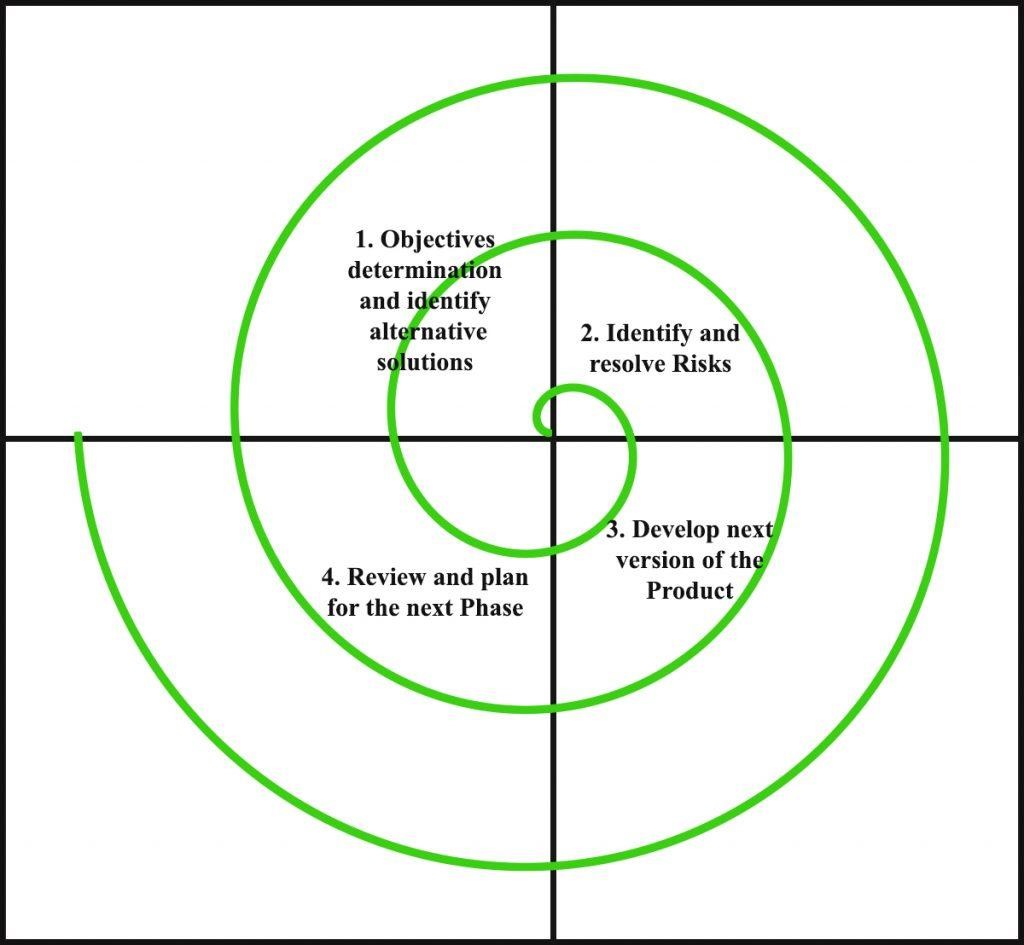
# 3. Spiral Model

The biggest problem we face in the waterfall model is that taking a long duration to complete the product, and the software became outdated. To solve this problem, we have a new approach, which is known as the Spiral model. The spiral model is also known as the cyclic model.

In this model, we create the application module by module and handed over to the customer so that they can start using the application at a very early stage. And we prepare this model only when the module is dependent on each other. In this model, we develop the application in the stages because sometimes the client gives the requirements in between the process.

The different phases of the spiral model are as follows:

1. Requirement analysis
2. Design
3. Coding
4. Testing and risk analysis

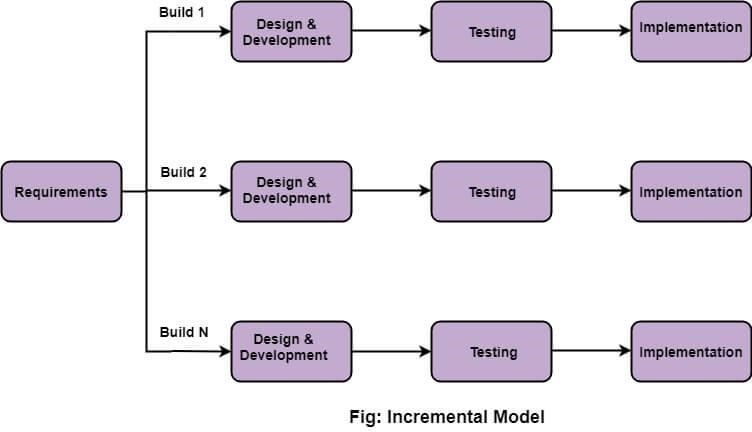


1. Planning objectives or identify alternative solutions: In this stage, requirements are collected from customers and then the aims are recognized, elaborated as well as analysed at the beginning of developing the project.
2. Risk analysis and resolving: As the process goes to the second quadrant, all likely solutions are sketched, and then the best solution among them gets select. Then the different types of risks linked with the chosen solution are recognized and resolved through the best possible approach. As the spiral goes to the end of this quadrant, a project prototype is put up for the most excellent and likely solution.
3. Develop the next level of product: As the development progress goes to the third quadrant, the well-known and mostly required features are developed as well as verified with the testing methodologies. As this stage proceeds to the end of this third quadrant, new software or the next version of existing software is ready to deliver.
4. Plan the next Phase: As the development process proceeds in the fourth quadrant, the customers appraise the developed version of the project and reports if any further changes are required. At last, planning for the subsequent phase is initiated.

## 4. Incremental Model

Incremental Model is a process of software development where requirements divided into multiple standalone modules of the software development cycle. In this model, each module goes through the requirements, design, implementation and testing phases. Every subsequent release of the module adds function to the previous release. The process continues until the complete system achieved.

1. Requirement analysis: In the first phase of the incremental model, the product analysis expertise identifies the requirements. And the system functional requirements are understood by the requirement analysis team. To develop the software under the incremental model, this phase performs a crucial role.
2. Design & Development: In this phase of the Incremental model of SDLC, the design of the system functionality and the development method are finished with



success. When software develops new practicality, the incremental model uses style and development phase.

1. Testing: In the incremental model, the testing phase checks the performance of each existing function as well as additional functionality. In the testing phase, the various methods are used to test the behaviour of each task.
2. Implementation: Implementation phase enables the coding phase of the development system. It involves the final coding that design in the designing and development phase and tests the functionality in the testing phase. After completion of this phase, the number of the product working is enhanced and upgraded up to the final system product

### 5. Agile Model

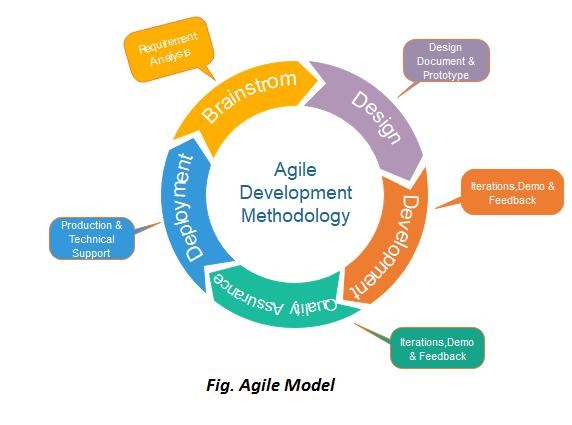
The meaning of Agile is swift or versatile. "**Agile process model**" refers to a software development approach based on iterative development. Agile methods break tasks into smaller iterations, or parts do not directly involve long term planning.

The project scope and requirements are laid down at the beginning of the development process. Plans regarding the number of iterations, the duration and the scope of each iteration are clearly defined in advance.

Phases of Agile Model:

Following are the phases in the Agile model are as follows:

1. Requirements gathering
2. Design the requirements
3. Construction/ iteration
4. Testing/ Quality assurance
5. Deployment
6. Feedback
7. Requirements gathering: In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.
8. Design the requirements: When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the highlevel UML diagram to show the work of new features and show how it will apply to your existing system.
9. Construction/ iteration: When the team defines the requirements, the work begins. Designers and developers start working on their project, which aims to deploy a working product. The product will undergo various stages of improvement, so it includes simple, minimal functionality.
10. Testing: In this phase, the Quality Assurance team examines the product's performance and looks for the bug.
11. Deployment: In this phase, the team issues a product for the user's work environment.
12. Feedback: After releasing the product, the last step is feedback. In this, the team receives feedback about the product and works through the feedback.



**Practical - 2**

Aim: Gathering and analyzing requirements of any twosoftware system.

* How long will our project schedule be?

the length of time an online voting management system takes to complete depends on several factors, such as the size of the voting population, the complexity of the ballot, and the level of security required.

* Who will be involved in the project?

The stakeholders involved in an online voting management system may vary depending on the context and the type of election being held.

* 1. Election officials: These are the individuals or organizations responsible for organizing and overseeing the election process, including setting up the online voting system, verifying voter eligibility, and ensuring the security of the system.
  2. Voters: These are the individuals who are eligible to cast their votes using the online voting system. They may include citizens, members of an organization or association, or shareholders in a company.
  3. IT professionals: These are the individuals who are responsible for developing, testing, and maintaining the online voting system, ensuring that it is secure, reliable, and user-friendly.
  4. Security experts: These are the individuals who specialize in securing computer systems and networks against hacking, cyber-attacks, and other forms of digital threats. They may be involved in designing the security protocols for the online voting system and monitoring the system for any suspicious activity.
  5. Observers and auditors: These are individuals or organizations that monitor the election process to ensure that it is fair, transparent, and free from any irregularities or fraud. They may be given access to the online voting system to verify that the results are accurate and reliable.
* What risks may we face in this project?

Online voting management systems can pose several risks that need to be considered and addressed to ensure the integrity and security of the election process. Here are some of the risks that could be encountered:

* 1. Security breaches: Online voting systems are vulnerable to security breaches, hacking attempts, and other forms of cyber attacks. These attacks can compromise the confidentiality, integrity, and availability of the voting system, leading to fraudulent or manipulated results.
  2. Technical glitches: Technical issues such as system downtime, software bugs, or network failures can disrupt the voting process and cause delays or inaccuracies in the results.
  3. Voter authentication: Ensuring that only eligible voters can cast their votes is a critical aspect of any voting system. However, online voting systems may face challenges in verifying voter identity and preventing duplicate voting or impersonation.
  4. Lack of transparency: Online voting systems can be challenging to audit, leading to concerns about transparency and accountability. Without proper auditing mechanisms, it may be difficult to detect and prevent any manipulation or fraud.
  5. Accessibility and usability issues: Some voters may face challenges accessing or using the online voting system, such as those with disabilities or those without access to reliable internet connectivity.

# Practical - 3

Aim: Prepare SRS for a software system.

 **Functional Requirement**

## User Interfaces

* Voters: The citizens of the country who are eligible for casting vote.
* Register for Online Voting System – Those who already have voter id, they will register themselves for online voting system and they will use their voter id as their user’s name and separate password will be used for secure authentication.
* Cast vote – The citizens will cast their votes for their favourite candidates online through a secure system.
* View own details – The voters will view their own details which they filled up at the time of their registration.

## Hardware Interfaces

* There are no hardware interfaces to this software system.
* The only interfaces are through a computer system.

## Software Interfaces

 The poll server runs on http server that is enabled to handle server pages (e.g. Apache Tomcat for support jsp). It uses a relational database to keep track of the polls, which it connects through standard database connectivity interfaces. In order to run the setup software, the environment needs to have a JVM running on it

 **Non-Functional Requirement**

## Performance Requirements

 The software is expected to have reasonably short response time. It should be able to log-in and feed the voter with new pages on request with a response time of the order of a few seconds.

## Safety Requirements

* The EA should set up his system time appropriately for the election process to start at the correct time.
* In case the EA detects any security lapse in the system, he should able to shut down the server and close all connections immediately while preserving the already polled votes.
* The system should be capable of gracefully recovering from earlier crashes and continuing the voting process.

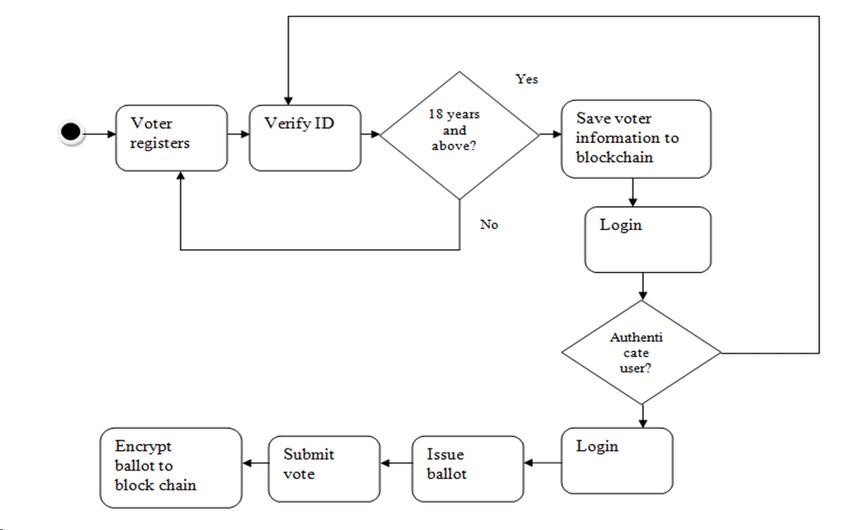
## Security Requirements

* The system should provide basic security features like password authentication and encrypted transactions.
* All the passwords generated and communicated to the users should be stored in the server only in an encrypted form for login management to prevent misuse.
* Serial attacks should be avoided by maintaining a minimum time gap between successive invalid log-in attempts.

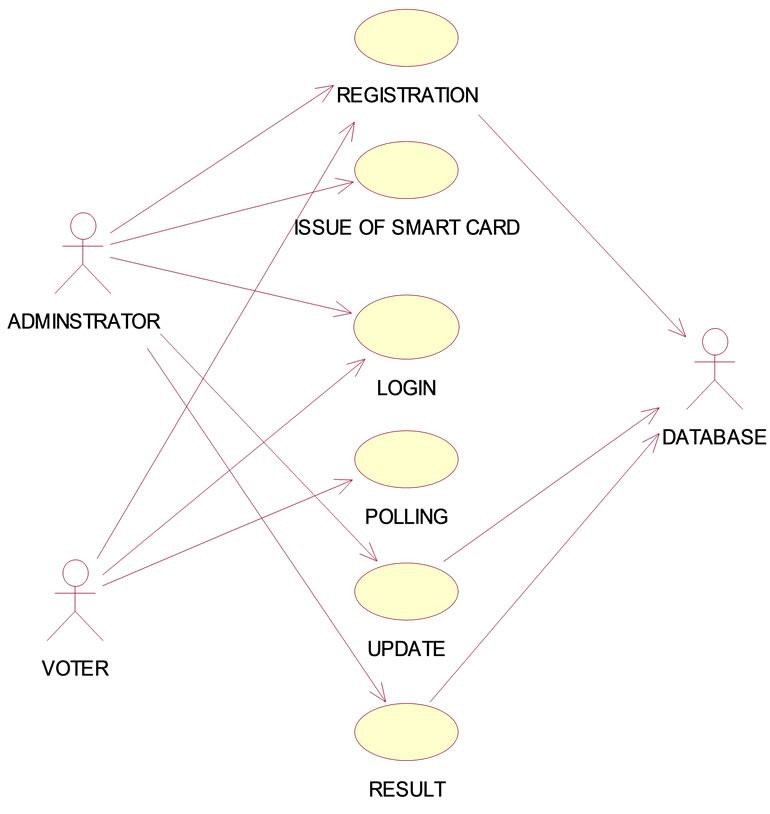
# Practical - 4

Aim: Design activity and use case diagram for your system.

* **Activity diagram**



* **Use case Diagram**



# Practical – 5

Aim: Design Data dictionary for your system.

**Voters Table:**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Description** |
| voter\_id | integer | Unique identifier for each voter |
| first\_name | varchar(50) | First name of the voter |
| last name | varchar(50) | Last name of the voter |
| email | varchar(100) | Email address of the voter |
| password | varchar(255) | Encrypted password for the voter's account |
| status | varchar(20) | Indicates whether the voter's account is active or inactive |
| created\_at | datetime | Timestamp for when the voter's account was created |
| updated\_at | datetime | Timestamp for when the voter's account was last updated |

**Election Table:**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Description** |
| election\_id | integer | Unique identifier for each election |
| election\_title | varchar(100) | Title of the election |
| start\_time | datetime | Timestamp for when the election starts |
| end\_time | datetime | Timestamp for when the election ends |
| status | varchar(20) | Indicates whether the election is active, upcoming, or completed |
| created\_at | datetime | Timestamp for when the election was created |
| updated\_at | datetime | Timestamp for when the election was last updated |

**Candidate Table:**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Description** |
| candidate\_id | integer | Unique identifier for each candidate |
| first\_name | varchar(50) | First name of the candidate |
| last name | varchar(50) | Last name of the candidate |
| election\_id | integer | Identifier for the election that the candidate is running in |
| party name | varchar(50) | Name of the political party that the candidate belongs to |
| candidate info | varchar(255) | Information about the candidate |
| photo\_url | varchar(255) | URL for the candidate's photo |
| created\_at | datetime | Timestamp for when the candidate was added |
| updated\_at | datetime | Timestamp for when the candidate's information was last updated |

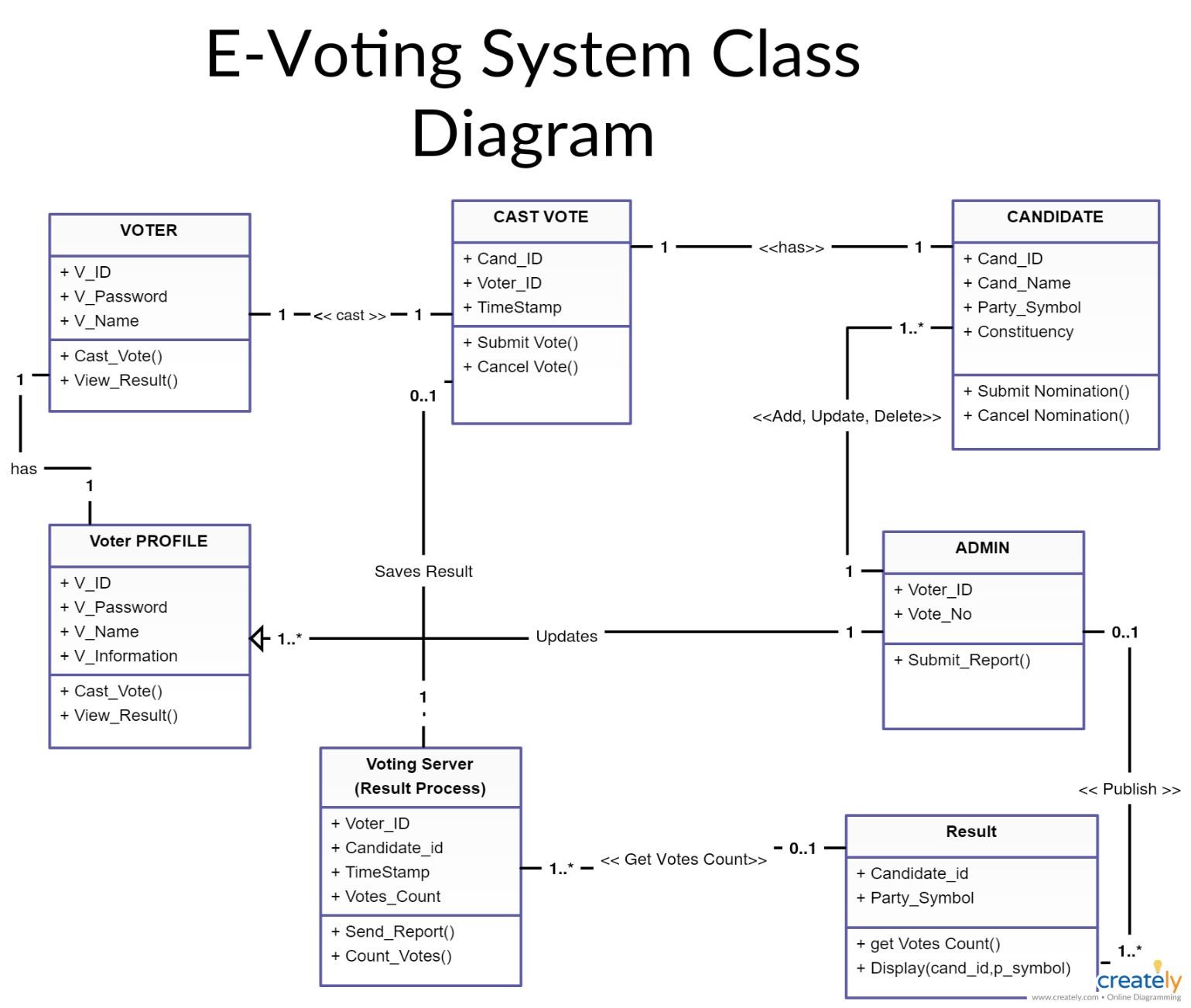
**Vote Table:**

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Description** |
| vote\_id | integer | Unique identifier for each vote |
| voter\_id | integer | Identifier for the voter who cast the vote |
| candidate\_id | integer | Identifier for the candidate who received the vote |
| election\_id | integer | Identifier for the election that the vote was cast in |
| created\_at | datetime | Timestamp for when the vote was cast |
| updated\_at | datetime | Timestamp for when the vote information was last updated |

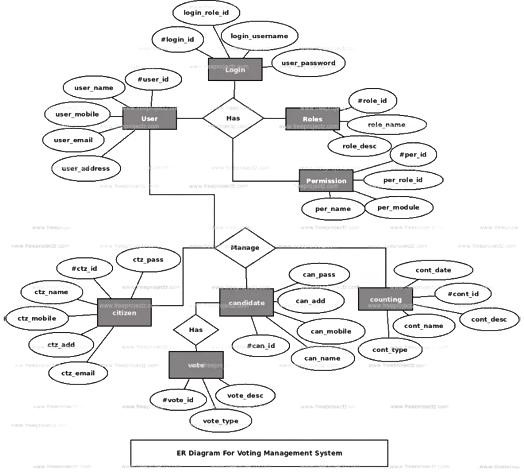
# Practical – 6

Aim: Prepare class diagram and ER diagram for a system.

* **Class diagram**



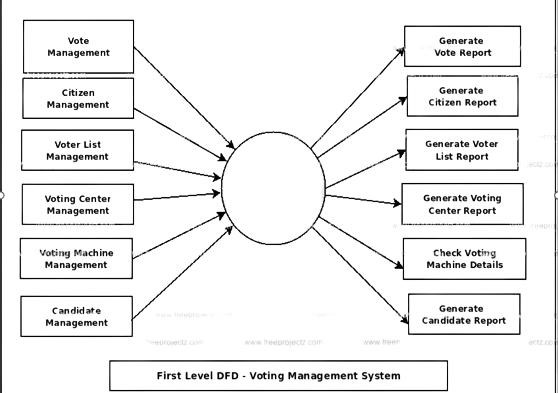
* **ER Diagram**



# Practical -7

Aim: Design Data flow diagram for system.

 **Data Flow Diagram**



# Practical -8

Aim: Study about various testing techniques.

1. Unit Testing
2. Integration Testing
3. System Testing
4. Functional Testing
5. Acceptance Testing
6. Smoke Testing
7. Regression Testing
8. Performance Testing
9. Security Testing
10. User Acceptance Testing

## 1. Unit Testing

Unit testing is a method of testing individual units or components of a software application. It is typically done by developers and is used to ensure that the individual units of the software are working as intended. Unit tests are usually automated and are designed to test specific parts of the code, such as a particular function or method. Unit testing is done at the lowest level of the software development process, where individual units of code are tested in isolation.

**The main advantages of unit testing include:**

1. It helps to identify bugs early in the development process, before they become more difficult and expensive to fix.
2. It helps to ensure that changes to the code do not introduce new bugs.
3. It makes the code more modular and easier to understand and maintain.
4. It helps to improve the overall quality and reliability of the software.

## 2. Integration Testing

Integration testing is a method of testing how different units or components of a software application interact with each other. It is used to identify and resolve any issues that may arise when different units of the software are combined. Integration testing is typically done after unit testing and before functional testing, and is used to verify that the different units of the software work together as intended.

**The main advantages of integration testing include:**

1. It helps to identify and resolve issues that may arise when different units of the software are combined.
2. It helps to ensure that the different units of the software work together as intended.
3. It helps to improve the overall reliability and stability of the software.
4. It’s important to keep in mind that Integration testing is essential for complex systems where different components are integrated together.
5. As with unit testing, integration testing is only one aspect of software testing and it should be used in combination with other types of testing such as unit testing, functional testing, and acceptance testing to ensure that the software meets the needs of its users.

## 3. System Testing

This software is tested such that it works fine for the different operating systems. It is covered under the black box testing technique. In this, we just focus on the required input and output without focusing on internal working. In this, we have security testing, recovery testing, stress testing, and performance testing **Example:**

This includes functional as well as non-functional testing

## 4. Acceptance Testing

Acceptance testing is done by the customers to check whether the delivered products perform the desired tasks or not, as stated in requirements.

## 5. Smoke Testing

This test is done to make sure that the software under testing is ready or stable for further testing

It is called a smoke test as the testing of an initial pass is done to check if it did not catch the fire or smoke in the initial switch on. **Example:**

If the project has 2 modules so before going to the module make sure that module 1 works properly

## 6. Regression Testing

Regression testing is a method of testing that is used to ensure that changes made to the software do not introduce new bugs or cause existing functionality to break. It is typically done after changes have been made to the code, such as bug fixes or new features, and is used to verify that the software still works as intended.

**The main advantages of regression testing include:**

1. It helps to ensure that changes made to the software do not introduce new bugs or cause existing functionality to break.
2. It helps to ensure that the software continues to work as intended after changes have been made.
3. It helps to improve the overall reliability and stability of the software.
4. It’s important to keep in mind that regression testing is an ongoing process that should be done throughout the software development
5. lifecycle to ensure that the software continues to work as intended. It should be automated as much as possible to save time and resources. Additionally, it’s important to have a well-defined regression test suite that covers

## 7. Performance Testing

It is designed to test the run-time performance of software within the context of an integrated system. It is used to test the speed and effectiveness of the program. It is also called load testing. In it we check, what is the performance of the system in the given load. **Example:**

Checking several processor cycles.

**8. Security testing**

Tests the software to identify vulnerabilities and ensure it meets security requirements.

## 9. User acceptance testing (UAT)

Tests the software to determine if it meets the end-user’s needs and expectations.

# Practical -9

Aim: Prepare various test cases for any two system.

## 1. Test case for user registration

* Test whether a user is able to create an account with valid information.
* Test whether a user is not able to create an account with invalid information.
* Test whether a user is not able to create an account with an existing email address.
* Test whether a user is able to login with valid credentials.
* Test whether a user is not able to login with invalid credentials.
* Test whether a user is able to reset their password using their email address.

## 2. Test case for creating an election

* Test whether an election can be created with valid information.
* Test whether an election cannot be created with invalid information.
* Test whether an election can be created with a start time in the future and an end time in the future.
* Test whether an election cannot be created with a start time in the past or an end time before the start time.
* Test whether an election can be edited with valid information.
* Test whether an election cannot be edited with invalid information.

## 3. Test case for adding candidates

* Test whether a candidate can be added to an election with valid information.
* Test whether a candidate cannot be added to an election with invalid information.
* Test whether a candidate can be edited with valid information.
* Test whether a candidate cannot be edited with invalid information.
* Test whether a candidate can be deleted.

## 4. Test case for voting

* Test whether a user can cote for a candidate in an election.
* Test whether a user can only vote once in an election.
* Test whether a user cannot vote if they are not eligible to vote in the election.
* Test whether a user cannot vote if the election has ended.
* Test whether the system records the vote correctly.

## 5. Test case for results

* Test whether the system displays the correct results for an election.
* Test whether the system updates the results in real-time as votes are cast.
* Test whether the system displays the results in a clear and understandable manner.
* Test whether the system handles ties appropriately.
* Test whether the system displays the results only after the election has ended.

# Practical -10

Aim: Study about different types of testing tools and discuss any two in details.

**There are various types of testing tools available for software testing, such as:**

1. Test management tools - these tools are used to manage the testing process, including test planning, test case creation, execution, and reporting.
2. Performance testing tools - these tools are used to measure the performance of an application under various conditions, such as high user loads or heavy data volumes.
3. Security testing tools – these tools are used to detect and prevent security vulnerabilities in an application, such as SQL injection or cross-site scripting (XSS) attacks.

4.Automation testing tools – these tools are used to automate the execution of test cases, reducing the time and effort required for manual testing.

5. Debugging tools – these tools are used to identify and fix software defects, such as memory leaks or race conditions.

## 1. Static Test Tools

Static testing refers to the process of analyzing software or code without actually executing it. This type of testing can be performed manually or using static testing tools. Static testing tools are software applications designed to analyse source code or other software artifacts to identify defects, vulnerabilities, and other issues. Some common types of static testing tools are:

1. **Code review tools** - These tools are used to perform code reviews and identify coding issues, such as syntax errors, coding standards violations, and logic errors. Examples of code review tools include Crucible, Gerrit, and Code Collaborator.
2. **Code analysis tools** - These tools are used to analyse code to identify potential defects and vulnerabilities. They can perform checks for coding standards compliance, code complexity, security vulnerabilities, and other issues. Examples of code analysis tools include SonarQube, Code Sonar, and PMD.
3. **Documentation tools** - These tools are used to generate documentation automatically from source code or other artifacts. Examples of documentation tools include Doxygen and Natural Docs.
4. **Model-based testing tools** - These tools are used to generate test cases automatically based on models of the software or system being tested. Examples of model-based testing tools include Spec Explorer and T-VEC.
5. **Formal verification tools** - These tools are used to prove the correctness of software or systems mathematically. They can verify properties like safety, liveness, and security. Examples of formal verification tools include TLA+ and Alloy.

Static testing tools can help improve software quality by identifying defects early in the development process, reducing the time and cost required for testing and debugging. However, these tools cannot detect all types of defects, and manual testing is still necessary to ensure software quality. Additionally, static testing tools require expertise in the specific tool and the underlying technologies to be effective.

### 2. Dynamic Test Tools

Dynamic testing refers to the process of testing software by executing it and observing its behaviour. This type of testing requires the use of dynamic testing tools, which are software applications designed to automate the testing process, capture and analyse test results, and report defects and issues. Some common types of dynamic testing tools are:

1. **Test management tools** - These tools are used to manage the testing process, including test planning, test case creation, execution, and reporting. They can also integrate with other testing tools to provide a unified testing environment. Examples of test management tools include HP ALM, TestRail, and Zephyr.
2. **Automation testing tools** - These tools are used to automate the execution of test cases, reducing the time and effort required for manual testing. They can simulate user actions, interact with the user interface, and perform checks on system behaviour. Examples of automation testing tools include Selenium, Appium, and Test Complete.
3. **Performance testing tools** - These tools are used to measure the performance of an application under various conditions, such as high user loads or heavy data volumes. They can simulate user behaviour, generate load, and measure response times and throughput. Examples of performance testing tools include JMeter, LoadRunner, and Gatling.
4. **Security testing tools** - These tools are used to detect and prevent security vulnerabilities in an application, such as SQL injection or cross-site scripting (XSS) attacks. They can simulate attacks, perform vulnerability scans, and generate reports on security issues. Examples of security testing tools include OWASP ZAP, Nessus, and Burp Suite.
5. **Debugging tools** - These tools are used to identify and fix software defects, such as memory leaks or race conditions. They can provide information on program execution, variables, and stack traces, and enable interactive debugging. Examples of debugging tools include Eclipse Debugger, Visual Studio Debugger, and gdb.