A short description to explain the meaning of Software Development Life Cycle (SDLC). Do ensure that you include three (3) software development models in your explanation. (Nicholas)
The systems development life cycle (SDLC) is a conceptual model used in software development projects. It has a 5 part framework which is utilized for the planning, requirements, design, coding, testing, and deployment of software development projects.

Stage 1. Planning and Requirements
During this phase, the objective and the requirements to produce the product are carefully considered. An estimate of resources, such as manpower and funds is being put together. All of the information is then analysed to see if there is an alternative solution of creating the project. If there is no other viable alternative, the information is assembled into a project plan and presented to management for approval.

Stage 2. Design and Analysis
The design phase is the look and feel of the system design. The flow of data processing is mapped into charts, and the project team determines the most logical design that they all agree for the user interface. The project team designs the layouts that the developers uses to write the codings for the actual interface.

During the analysis stage the project team determines the end-user requirements. Often this is done with the assistance with the sponsors, which provide an explanation of their needs and what their expectations are for the finished software. The project team will put into consideration all of the user requirements and gets a sign-off from the client and management to move forward with the system design.

Stage 3. Coding.
Detailed designs that are created by the software designers are being converted into actual interfaces done by the programmers.

Stage 4. Testing.
During the test phase all aspects of the system are tested for functionality and performance. The system is tested for integration with other products as well as any previous versions with which it needs to communicate. The testing phase is to verify that the system contains all the end user requirements laid out in the analysis phase and all the functions are accurately processing data working with all other systems and if it meets the expectations.

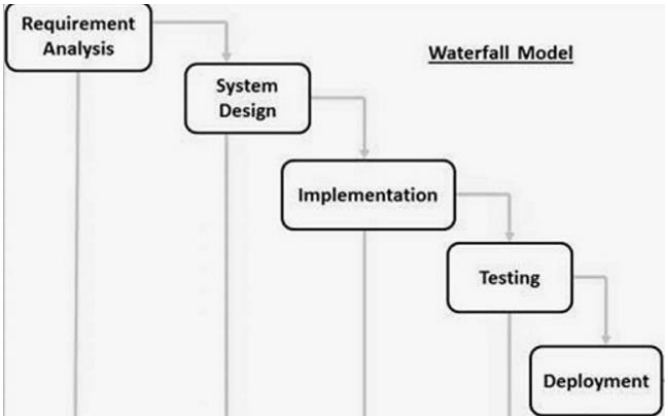
Stage 5. Deployment
The application will first be only distributed among a group of selected customers prior to official release to test the software and to see if there is any problems with it. They will receive feedback from these group of people and make necessary tweaking and changes for any issue before officially releasing it.

Agile Modelling
Agile SDLC model is a combination of iterative and incremental process models with focus on purpose adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small increment builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks. Every iteration involves cross functional teams working simultaneously on various areas like design, coding, etc.

Advantages
- Promotes teamwork
- Resource requirements are minimum
- Suitable for fix or changing environments
- Easy to manage

Disadvantages
- Not suitable for complex dependencies
- More risk of sustainability, maintainability and extensibility.

Waterfall Model
It is the first model to be ever introduced. It is very simple to understand and use. In the waterfall model, each step have to be completed before moving on to the next step or phase. The requirements are very clear and fixed, there is no ambiguous requirements. The product definition is clear and the project duration is rather short.



Illustrate the step by step in the waterfall model.

Advantages
- Phases are processed and completed one at a time
- Clear defined stages
- Easy to arrange tasks
- Process and results are well documented

Disadvantages
- No working software is produced until the end
- High amount of risk and uncertainty
- Not a good model for complex projects

Iterative Model

In the Iterative model, it starts with a small set of the software requirements and iteratively enhances and evolving the software until the complete system is implemented and ready to be deployed.

An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.

Unlike other SDLC models, Iterative model is unique and is used when

- Requirements of the system is clearly defined and understood
- There is plenty of time
- A new technology is being used and learnt by the developing team while working on the project
- Resources that required specific skill set are not available
- There might be high risk changes in future.

Advantages

- Some working functionality can be developed early in the life cycle
- Results are obtained systematically and periodically
- Progress can be measured
- Less costly to change the requirements
- Testing and debugging is much easier
- Easier to manage risk (High risk functionalities are done first)
- Supports changing requirements
- Better suited for large and mission-critical projects

Disadvantages

- More resources are required
- Not suitable for small projects
- Management complexity is more
- Highly skilled resources are needed for analysis
- End of project date is not definite
- Project progress relies heavily on the risk analysis phase

(Xin Yi)
SDLC define as the steps in software development process. It consists of gathering client's requirements, analysis and design the software, implement programming language according to client's requirements, test the software to ensure error-free and lastly, deploy the software to the client. There are a few models to develop the software.

Waterfall Model
This process completes each phase of the original SDLC method successfully before going to the next phase. It is possible to schedule a deadline for each phase since Waterfall method proceeds in strict order and is able to complete on time. Although it is the easiest method to use, however once an error occurs on the previous phase, it will be difficult to amend since the flow of this model is in sequence.

Prototyping Model
This process start with a prototype whereby developer draft a prototype for the client before implementing. First, developer will need to understand the basic product requirements. Secondly, the prototype is developed according to the requirements. Thirdly, the developed prototype will be handover to the client for feedback. Lastly, developer will make amend according to client feedback and develop a new prototype again. This cycle will keep repeating until client expectations are met. There are two type of software prototyping:

- Throwaway Prototyping
  This prototype requires minimal effort and requirements to build a prototype. The prototype will be discard once the requirements are finalized and the system will develop accordingly to client's requirements.

- Evolutionary Prototyping
  Developer build a functional prototype that is well-understood and show client for comment. This process repeat until the full system is complete

  The pros of this model are that errors can be detect earlier and client can gain better understanding of the system this is going to be developed. However the cons are, it increase complexity of the system and developers might use existing prototypes to build the system

Agile Model
This process is more flexible, which uses an adaptive approach whereby planning is not required and the phase does not flow accordingly unlike the traditional SDLC. It is most suitable for client who are constantly changing the requirements. This model is easy to manage and requires teamwork. However, this method requires lots of changes and if there is a lack of documentation, it may be quite challenging to carry on the work. Since this method is highly dependent on client's instructions, communication is very important or else the team might deliver the end product wrongly.

(Joseph)
SDLC is a procedure used to guide in software designing, developing and testing. Its main purpose is to develop top-graded software that satisfy client expectations within the deadline and budget. SDLC comes in 5 phases:

- Requirements
    - User requirements and expectations
- Analysis/Design
    - Hardware design, UI design, Database design, Program coding design
- Coding
    - Coding Implementation (E.g. CRUD)
- Testing
    - Making sure no errors or bugs occurs before deployment
- Deployment
    - Launch the software

An example of software development models that practice SDLC are waterfall model, prototyping model and agile model.

Waterfall model is a sequential approach which progresses to the next phase after completion of the previous which is easy to control and monitor as only one phase is being dealt with at a time. However, if a problem was later to be discovered at an earlier stage, it would be almost impossible to resolve.

Prototyping model is another approach that develops the system or product and is reworked until it is finally approved by the client. This can serve as a leverage for the client as they can change their mind to add or remove requirements. However, doing so tend to corrupt software structure which can be costly in the long run.

Agile model is another approach that is adaptive and independent in nature. It believes that different and specific methods are required to suit the project requirements. Not only does it provide more flexibility to the developers but it also promotes teamwork. However, such model is best utilized for small or medium projects while huge projects still do require some form of fixed agreement.

(Ron)
Software Development Life Cycle, SDLC for short, is a normally 6-part framework which is utilized for the planning, defining, designing, building, testing, and deployment of software development projects. The framework specifies the executed tasks throughout the stages involved for the software development process, which are as follows:

Stage 1: Planning and Requirement Analysis
The foundation and most crucial part of SDLC, requirement analysis is performed by team members of higher ranking along with sales department, domain expert, market survey, and customer contribution. Using analysis results, a simple project approach is planned along with product feasibility research throughout operational, technical, and economical areas, which allows for the definition of technical approaches which can be utilized for minimal risk project implementation. Additionally, Quality assurance requirement planning and project risk identification is also done in this stage.

Stage 2: Defining Requirements
Product requirements are defined and documented in this step, which must then be sent for customer and market analysts approval using a Software Requirement Specification (SRS) document consisting of all the to-be designed and developed product requirements. Problems may occur in this stage due to pitfalls such as difficulty specifying requirements accurately, incomplete and/or unstable requirements, as well as misunderstanding of user needs.

Stage 3: Designing the Product Architecture
Using requirements stated in the SRS, the proposal and documentation of one or more design approaches done in a Design Document Specification (DDS), which is then reviewed by all significant stakeholders. Following which, the best design approach is chosen based on several parameters such as risk assessment, design modularity, budget, and time constraints. Difficulty fulfilling this step might occur due to pitfalls such as cost/time issues, inability to meet the requirements, multiple design options comparison, poor and/or overly specific design, and substantial amounts of derived requirements.

Stage 4: Building or Developing the Product
Development of the product begins in this stage, whereby code befitting to the software type is produced according to the DDS, which can be efficiently done provided a thorough and orderly design is present. Furthermore, predefined coding guidelines must be adhered to during development. This stage, as with the rest, has several pitfalls to watch out for such as technology limitations, lack of enhancement consideration, multiple developers/teams, deadline issues, and conflicting choices such as speed against performance.

Stage 5: Testing the Product
In this stage, SRS defined quality standards are attempted to be met through repeated cycles reporting, tracking, and resolving of product defects found through testing. Some pitfalls of this stage include non-developer, late, and/or insufficient testing.

Stage 6: Deployment in the Market and Maintenance

In this stage, the product can be either released to a small user-base within a real business environment for UAT (User Acceptance Testing) or deployed into the market as a whole or in stages, depending on the organization's business strategy.

If UAT is performed, the product will be either released as is or with further enhancements depending on feedback. While in the market, maintenance is performed for the customer base. Pitfalls in this stage include version control, production and development environment mismatch, lack of production environment testing, and wrong expectation of production environment.

With these 6 steps, the production of high-quality software meeting /exceeding customer expectations while meeting time and cost estimates is targeted by the SDLC. 3 examples of important and/or popular SDLC models are as such:

Waterfall Model

- The first Process Model introduced
- Also known as linear-sequential life cycle model
- Requires each phase to be completed before beginning of next phase

Appropriate situations for usage:

- Well documented, clear, and fixed requirements
- Stable product definition
- Understood, non-dynamic technology
- Unambiguous requirements
- Short project
- Abundant availability of resources along with required expertise for product support

Pros:

- Ease of understanding and usage
- Ease of management due to model rigidity
- Well-illustrated stages
- Comprehensive milestones
- Ease of task arrangement
- Well documented process and results

Cons:

- Production of working software in late parts of life cycle
- High risk and uncertainty
- Poor model for complex and object-oriented projects
- Unsupportive of changing requirements
- Progress measurement difficult within stages
- Technological and/or business challenges and/or constraint identification unavailable early

Iterative Model

- Iterative enhancement (design modifications and added capabilities) of evolving versions through reviews, beginning with partial specification and implementation of software requirements
- Several iterations of software development cycle at a given moment is possible
- Whole requirement divided into various builds
- successful through meticulous validation of requirements as well as verification and testing of each software version against said requirements in each model cycle

Appropriate situations for usage:

- Well defined and understood requirements of complete system
- Definite significant requirements
- Usage of technology foreign to developers
- Unavailable resources with required expertise which will likely be used on contract basis
- High-risk features and risk of changing requirements present

Pros:

- Early and periodical results
- Measurable progress
- Parallel development possible
- Less scope/requirement change cost
- Ease of testing and/or debugging during smaller iteration
- Identification and resolving of risk during iteration
- High risk parts are done first
- Operational product available after every increment
- Superior risk analysis
- Lower initial operating time
- Suitable for mission-critical and large projects

Cons:

- Possibly more resource consuming
- Not favourable of changing requirements
- Increased management attention/complexity
- Risk of system architecture and/or design issues
- Unsuited for smaller projects
- Risk of unclear project conclusion
- Risk analysis requires skilled resources
- Extremely dependent on risk analysis

Prototyping Model

- Display of in-development product through application prototype development
- Rising in popularity due to early understanding of customer requirements via feedback
- 4 types:
    - Throwaway (Rapid/Close Ended) - Low effort and analysis requirement. The prototype discarded once actual requirements are understood, followed by development of actual system
    - Evolutionary (breadboard) - Development of prototype using initially understood requirements with the addition of more when understood.
    - Incremental - Development of subsystems into multiple functional prototypes, which are later integrated into a complete system
    - Extreme - Applied in web development domain. Development process involves prototyping of all existing pages in HTML (HyperText Markup Language) format, usage of a prototype services layer to simulate data processing, and implementation + integration of services to final prototype.

Appropriate situations for usage:

- Development of highly user interactive systems such as online systems
- Layout management of systems requiring form filling and/or scrolling of pages before data processing
- Unbeneficial to high data processing and low UI (User Interface) software

Pros:

- User involvement present before implementation
- Early detection of defects, decreasing time and cost
- Early user feedback
- Easy identification of absent and/or problematic functionality

Cons:

- Highly dependent on prototype, leading to risk of inadequate requirement analysis
- Confusion of prototype with actual systems
- Risk of increased system complexity due to expansion of original scope
- Risk of unfeasible usage of existing prototype to build system by developers
- Risk excessive effort for prototype building due to improper monitoring of process
- Difficulty estimating, planning, and managing prototype projects due to lack of regular deliverables

References

(Nicholas)
http://smallbusiness.chron.com/steps-system-development-life-cycle-43241.html
https://www.tutorialspoint.com/sdlc/sdlc_waterfall_model.htm
https://www.tutorialspoint.com/sdlc/sdlc_iterative_model.htm

(Xin Yi)
https://www.tutorialspoint.com/sdlc/sdlc_agile_model.htm
http://searchsoftwarequality.techtarget.com/definition/waterfall-model
https://www.tutorialspoint.com/sdlc/sdlc_software_prototyping.htm

(Ron)
https://www.tutorialspoint.com/sdlc/sdlc_overview.htm