***Assignment 1***

***Section A***

**Team 9 KH - CodeAstra**

**Team Member’s Names:**

* 082-Prashant Chaudhary
* 083-Prathamesh Patidar
* 084-Pratik Pawar
* 085-Pratik Raut
* 086-Pratiksha Barge
* 087-Priyadarshini Meher
* 088-Priyanka Pawar
* 089-Priyanka Mahajan
* 090-Priyanka Aher
* 091-Rahul Darade
* 092-Rahul Somvanshi

***Q1. What are the reasons of a successful and unsuccessful software project?***

***Ans with solutions :***

*A good starting point is by addressing some of the critical reasons software projects fail.*

*Reason 1: Not Enough Time*

*Often, a deadline gets decided before the project starts and is non-negotiable. This deadline results in a headlong rush to get started on the assumption, the sooner you begin coding, the sooner you'll finish the project.*

*A rush to start coding is almost always the wrong approach. It is essential to spend the time to create a good design. Not having a good design leads to continuing changes throughout the development phase. When this happens, time and budget get consumed at a rapid rate.*

*Solution:*

*Don't be tempted to jump straight in and begin coding.*

*Assign sufficient time to create a good design, and the rest of the project will run much better.*

*This approach will improve your reputation when you deliver something that fulfils your customers' expectations and works the first time correctly.*

*Reason 2: Insufficient Budget*

*Many projects have the lowest price, most successful supplier policy, or an unrealistically low budget, not based on the project requirements. When this happens, everything slows down. Resources are slow to arrive or never arrive; corners get cut, and quality suffers.*

*Solution:*

*Be realistic about the budget and base it on the complete requirements.*

*Avoid basing the selection of a supplier solely on the lowest price.*

*Go to a supplier or team with a proven track record of delivering within budget.*

*Use a Supplier Selection Checklist such as the one below to find the right supplier for your project.*

*Reason 3: Poor Communication*

*There's an adage, "never assume anything", which is especially the case for software projects. Good communication with your customer, users, and the development team is critical to project success. Ask yourself three questions:*

*Does everyone in the team understand you?*

*Do they know what you expect of them, or have you assumed they know?*

*Do they communicate well with one another, with users and with other departments?*

*Solution:*

*Find any communication breakdowns now. These can lead to confusion and complications later in the project.*

*Never assume that everyone understands everything that is happening on the project.*

*Take time to create an environment where communication is accessible, open and frequent.*

*Reason 4: Never Reviewing Project Progress*

*As a project progresses, things change, significantly impacting the project. It is important to keep examining the project progress to overcome challenges early and warn stakeholders of possible delays and outcomes changes.*

*Solution:*

*Always set milestones to review progress with your team and stakeholders during the project. Adjust as necessary to stay on course.*

*Stay close to your team to understand what is happening and any challenges they face.*

*Reason 5: Inadequate Testing*

*When the pressure to deliver is on, testing often suffers. The testing gets left until the end of the development cycle with minimal effort expended on testing. Usually, the result is a product filled with bugs and an unhappy customer.*

*Solution:*

*Carry out testing throughout the development lifecycle, testing each module or component as it's developed.*

*Only leave the integration testing until the end of the development lifecycle, which results in less stress and a better product.*

*Reason 6: Testing in the Production Environment*

*It's surprising how many organisations test products in their production environment. Using the production environment is a high-risk strategy that can lead to security breaches and accidental release without testing, disrupting the production systems.*

*Solution:*

*Develop a process for quality assurance and release of new software products.*

*Provide an environment separated from the production environment for testing and bug fixing.*

*Reason 7: Lack of Quality Assurance*

*Often in our haste to deliver the software, quality assurance suffers. Documentation is incomplete for code changes, the design contains flaws, and implementations can be unfinished. These all lead to rework, lost time and eventually unhappy customers.*

*Solution:*

*Take time to quality check and document software before release.*

*Review Michael L Young article 6 Success Factors for Managing Project Quality*

*Reason 8: Not Conforming to Industry Standards*

*World Wide Web Consortium and International Organisation for Standardisation logos Conforming to industry standards in your software projects can prove beneficial by ensuring good accessibility, portability, usability, robustness, and reducing current and future problems. Bodies such as the World Wide Web Consortium (W3C) and International Organisation for Standardisation (ISO) have developed open standards that are hard to challenge.*

*Solution:*

*Take time to introduce a standard approach for your projects.*

*Find what works well and keep doing it.*

*Change anything that isn't working.*

*Review and update your standards regularly.*

*Next time you project manage a software development project, review this list and remind yourself what is needed to ensure success. You'll be surprised; it makes a difference.*

***Q2. What types of problems may arise if a software project is developed on adhoc basis ?***

***Ans.***

*The meaning of word Ad-hoc is something which is not in order or not organised or unstructured. In the similar note the Ad-hoc testing is nothing but a type of black box testing or behavioural testing.*

*Ad-hoc testing is carried out without following any formal process like requirement documents, test plan, test cases, etc. Similarly while executing the ad-hoc testing there is NO formal process of testing which can be documented.*

*Ad-hoc testing is usually done to discover the issues or defects which cannot be found by following the formal process.*

***Q3. Provide three examples of software projects that would amenable to the waterfall model. Be specific.***

***Ans.***

1. *An****Operating System****, as the various specific parts of the OS could be developed as the user requires them*
2. *A****Graphical User Interface****, similar to the OS, the GUI can be created according to the customer’s requirements and approval.*
3. *A****Web Application****, a base application can be developed and delivered, followed by any number of additional plug-ins that the customer would want for additional functionality*

***Q4. Provide three examples of software projects that would be amenable to the prototyping model. Be specific.***

***Ans****.*

*There are three examples of software projects that would be amenable to the incremental model.*

*1) fingerprint voting system*

*2) fingerprint. based ATM system*

*3) weather forecasting system*

*1) Fingerprint voting system- Fingerprint Voting System was implemented with the Arduino technology. In this System a voter can poll his vote easily. In this database server all voters' information was stored to register in this system, the voter should fill a registration form with the help of a user id and password. This information will be checked by the database server. Because all the information about the voter would be already there is anything wrong, the system will not allow the voter to poll his or her vote. This system is helpful to the voter's decreases the time of voting process .*

*2) Fingerprint based ATM system- Fingerprint Based ATM is a desktop application where fingerprint of the user is used as an authentication. The finger print minutiae features are different for each human being so the user can be identified uniquely. Instead of using ATM card Fingerprint based ATM is safer and secure. There is no worry of losing ATM card and no need to carry ATM card in your wallet.*

*3) Weather forecasting system- The global weather forecasting system and solutions market size was valued at USD 2.51 billion in 2016. It is expected to post a CAGR of 7.1% over the forecast period. These systems help enterprises in gaining real-time insights into atmospheric conditions, which in turn, enables enterprises to carefully plan all weather-sensitive operations to ensure security, sustainability, safety, and cost efficiency. Factors such as growing sea and air transportation, increasing stringency of norms pertaining to environmental protection, and high dependency on rainfall for water supply are among the key trends stimulating market growth.*

***Q.5. What process adaptions are required if the prototype will evolve into a delivery system or product?***

***Ans.***

*If a prototype is evolved into a delivery system or product, it begins with communication.*

*The software engineer and customer meet and define the overall objectives for the software,*

*identify whatever requirements are known, and outline areas where further definition is mandatory.*

*The prototype serves as a mechanism for identifying software requirements.*

*If a working prototype is built, the developer attempts to make use of existing program fragments or applies tools*

*(e.g., report generators, window managers, etc.) that enable working programs to be generated quickly.*

***Q6. Provide three examples of software projects that would be amenable to the incremental model. Be specific.***

***Ans:***

*Incremental model is a combination of one or more waterfall model. In these model project requirement are divided into multiple modules and each module is developed separately. Finally developed module is integrated with other modules.*

*Each linear sequence produces deliverable "increments" of the software for example, word-‐processing software developed using the incremental paradigm might deliver basic file management, editing and document production functions in the first increment; more sophisticated editing and document production capabilities in the second increment; spelling and grammar checking in the third increment, and advanced page layout capability in the fourth increment. The process flow for any increment may incorporate the prototyping paradigm. Incremental development is particularly useful when staffing is unavailable for a complete implementation by the business deadline that has been established for the project.*

*Whatsapp, Windows, MS Office all are the examples of incremental model as day to day requirements change people want something better so they again analyse the requirement and repeat the SDLC process*

*When to use the Incremental model:-*

*This model can be used when the requirements of the complete system are clearly defined and understood.*

*Major requirements must be defined; however, some details can evolve with time. There is a need to get a product to the market early. A new technology is being used. Resources with needed skill set are not available. There are some high risk features and goals.*

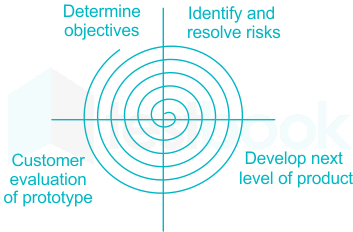
***Q7. As you move outward along the spiral process flow, what can you say about the software that is being developed or maintained?***

***Ans:***

*The product advances to a more complete state as work spirals outward, and the****level of abstraction at which work is conducted decreases****(i.e., implementation specific work accelerates as we move further from the origin).*

***Explanation:***

*One of the most significant models for the****Software Development Life****Cycle that supports****risk handling****is the****spiral model.***

**

*In diagrammatic form, it resembles a****spiral with several loops****. The spiral's precise number of loops is unclear and varies from project to project. A phase of the software development process is referred to as each****spiral loop.***

*The project manager might alter the precise number of phases required to build the product depending on the project's risks. The project manager plays a crucial role in the spiral model of product development since they dynamically set the number of phases.*

*The****waterfall model's****methodical, managed elements are combined with the idea of iterative development in the****spiral model.****Iterative and sequential linear development models, or the waterfall model, are combined to create the spiral model, which places a strong emphasis on risk analysis.*

***Q. 8 Key Differences Between Product-Based and Service-Based Company.***

***Ans.***

*What is a Product-Based Company?*

*Product Based companies are the ones that produce high-end products in order to satisfy their customers’ requirements. These companies are focused on introducing or producing products that have a high market value. These are the companies that keep on improving their products and upgrading their technologies in order or be in the market. Some of the popular*

*product-based companies are Amazon, HP, Microsoft, Facebook, Adobe, Intel, Informatica, Amdocs, CISCO, etc.*

*1)These are the companies that have their own products to sell. Examples are Google, Alphabet, Facebook, Adobe, etc.*

*2)These hire a very limited number of candidates from top colleges like IITs and NITs, and they generally don’t conduct campus placements in tier 3 colleges.*

*3)These provide high salaries and hikes, also the salary is revised based on the cost of living*

*4)These conduct more interview rounds, and cracking interviews is a bit more difficult than Service-Based companies.*

*5)The Learning curve is good, optimized, and efficient means are used. Also, you get to work on the latest technologies.*

*6)Flexible office timings, all you need to do is finish your work in the given time frame.*

*7)Turnaround time is quick as products are bought instantaneously by Veryhikesconsumers.*

*8)The client pays for tangible products after performing various tests physically.*

*What is a Service-Based Company?*

*Service-based companies are the ones that do not have a particular product but their primary business is to provide services. If we talk more clearly then they work for other organizations or have client-based products. The services they provide are intangible and can be provided in the terms of expertise, amenities, or skills. These companies generally provide less salary compared to product-based companies. Some of the popular service-based companies are TCS, Infosys, Wipro, Tech Mahindra, etc.*

*1)These companies may or may not have their products, but their primary business is service. They work for other organizations or we can say they are clients of Product Based companies. Examples are Infosys, Wipro, TCS, Cognizant, etc*

*2)These are the mass recruiters in campus placements.*

*3)These provide comparatively less salary and hikes.*

*4)Cracking interview is relatively easy and has few rounds.*

*5)The learning curve is average. Generally, they put freshers into support projects where the chances are that one does not have the choice to select his/her domain.*

*6)Strict office timings, around 7-9 working hours per day.*

*7)Turnaround time is low and is dependent on the consumer’s requirements.*

*8)There is no physical experience with the product and the client only pays for that.*

*Conclusion:*

*There are many companies in the market and some are service-based and some product based, it depends upon a person’s interest in which company they want to work for. Now when you have gone through the basic difference, you can assess that properly and make up your mind accordingly.*

***Q9. What is a Process framework & framework activities?***

***Ans:***

*The process framework defines a small set of activities that are applicable to all types of projects.*

*It includes framework activities and umbrella activities that are applicable for entire software process.*

*Common framework activites: (CPMCD)*

*1) Communication: Involves communication with customers (project initiation) and gathering requirements.*

*2) Planning: Establishes a plan for project (Estimating, Scheduling, Tracking).*

*3) Modeling: Constructing models (Blueprint) so that customer can understand in a better way.*

*It includes :*

*A) Analysis*

*B) Design*

*4) Construction: Combination of code generation and testing.*

*5) Deployment: Software is delivered to customer.*

*Evaluate software and send feedback.*

***Q10. What are principles of software engineering?***

***Ans:***

*Software engineering principles are a collection of approaches, styles, philosophies, and best practices recommended by world-renown software engineers and authors. As part of software development, these principles serve as guidelines to ensure the final version of a piece of software fulfils its purpose.*

*Principles of Software Engineering are as follows*

*1.SOLID PRINCIPLES*

*SOLID is an acronym for a collection of object-oriented design principles. Each letter in the ”SOLID” stands for one of the following principles:*

*S – SRP (Single Responsibility Principle): According to the Single Responsibility Principle, a class, function, module, or service must have only one reason to change, i.e., it must have only one responsibility. Yet why is this so important?*

*When you write classes or functions that are dedicated to a single functionality, it becomes easier to understand, maintain, and modify your code.*

*If you want to modify the functionality of the system, you would know the exact DRY location where you have to modify the code.*

*It makes the code more organized and readable. It also makes reusing the code easier.*

*O – OCP (Open Closed Principle): In software development, we work in phases. As a team, we implement a bunch of functionalities, test them, and then deliver them to the users. We then move on to implementing the next set of functionalities. When it comes to developing new functionality, the last thing we want to do is to change the existing functionality, which has been tested and is working. Therefore, we try to add new functionality on top of existing ones.*

*This idea is facilitated by the Open-Closed principle. According to it, our functions, classes, and modules should be designed in such a way that they’re open for extension, but closed for modification.*

*Open for Extension: New functionality can be added to classes and modules without breaking the existing code. Composition and inheritance can be used to accomplish this.*

*Closed for Modification: It’s ideal not to make changes that break current functionality, as doing so would require refactoring a lot of existing code and writing several tests to ensure the changes work.*

*L – LSP (Liskov Substitution Principle): According to the Liskov Substitution Principle, all child/derived classes should be replaceable for their parent/base classes without affecting or breaking the program’s correctness. Thus, objects in your subclass (derived/child class) should behave similarly to objects in your superclass (parent/base class). Therefore, you should use inheritance carefully in your projects. While inheritance can be beneficial, it is advisable to use it moderately and contextually. Before you can perform inheritance, you have to consider the postconditions and preconditions of the class.*

*I – ISP (Interface Segregation Principle): According to the Interface Segregation Principle, clients shouldn’t be forced to depend or rely on methods that they don’t use. How can this be achieved? Simple: make your interfaces short or small and focused. An interface with a lot of behaviors is hard to maintain and evolve. Therefore, Separate large interfaces into smaller ones, each focused on a specific set of functions, so that choose to depend or rely only on the functionalities that they require*

*D – DIP (Dependency Inversion Principle): This Principle seeks to eliminate tight coupling between software modules. According to this principle, high-level modules should not depend on lower-level modules, but rather on their abstractions. We can break it down into two parts:*

*A high-level module must be independent of a low-level module. Both should rely on abstractions*

*The abstraction should be independent of the details, while the details should be dependent upon the abstractions.*

*Why do we follow this principle? The reason is that abstractions are pretty stable. As a result, you can easily modify the behavior of your open-source or closed-source code. Therefore, you can improve its future evolution.*

*2. KISS PRINCIPLE (Keep It Simple, Stupid).*

*It’s best if your methods are small, not exceeding 40-50 lines.*

*All the crucial/critical methods should have a commented doc for better understanding for other dev.*

*Methods should only address one problem at a time.*

*Your project has a lot of conditions, right? Break up the codes into smaller blocks as you go.*

*If possible, use simple constructions that solve the problem without a lot of branching, deep nesting, or complex class structures.*

*3. DRY (Don’t Repeat Yourself) PRINCIPLE*

*Make sure not only that your code is error-free, but that it is free of duplicate lines.*

*If a piece of code appears more than twice in the codebase, it should be moved to a separate function.*

*You should create a separate method even if you find that it is repeated a second time.*

*As a bonus, automate any manual processes you can in order to keep the code lean.*