WELCOME TO SOFTWARE ENGINEERING SEMINAR

Software Development Life Cycle (SDLC)

A Structured Approach to Software Development

TEAM MEMBERS

ISHU ANAND MALVIYA	22C6038
ROHIT KEWAT	22C6059
HARSHIT SHARMA	22C6035

INTRODUCTION TO SDLC

INTRODUCTION TO SDLC

What is SDLC?

The Software Development Life Cycle (SDLC) is a structured process used for developing software applications. It outlines the phases or stages involved in the creation, deployment, and maintenance of software. SDLC helps ensure that software is developed in a systematic, efficient, and cost-effective manner, addressing both technical and business requirements.

IMPORTACE OF SDLC

IMPORTANCE OF SDLC

1. Clear Project Structure:

Organised Phases: SDLC breaks down the software development process into distinct phases (Planning, Design, Development, Testing, Deployment, Maintenance). This structure helps teams stay on track and reduces the chances of skipping important steps.

2. Efficient Resource Management:

SDLC helps in the effective allocation of resources (time, budget, personnel) by defining the project requirements and providing clear timelines.

3. Improved Quality:

The testing phase in SDLC ensures that the software meets quality standards and is free from major bugs.The iterative approach in SDLC allows for continuous evaluation of quality, ensuring that issues are addressed before the product is released.

4. Risk Management:

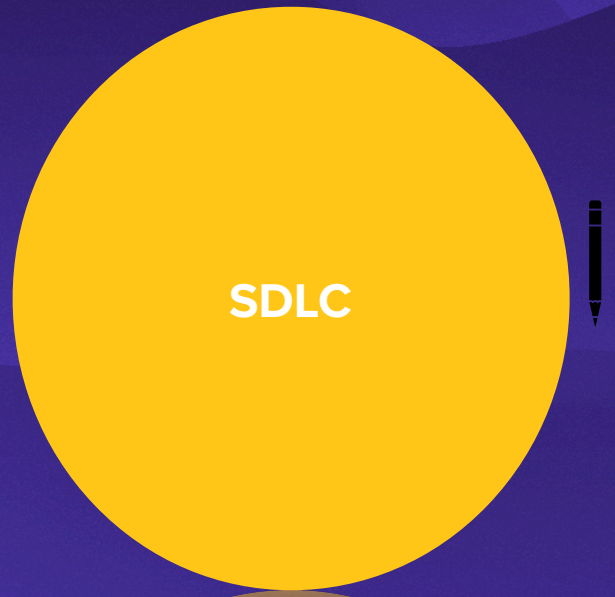
SDLC helps in identifying risks early on, such as feasibility concerns, technical limitations, or schedule delays. By analysing these risks at the beginning of the project (during planning and feasibility studies), teams can take proactive measures to mitigate them.

5. Maintainability:

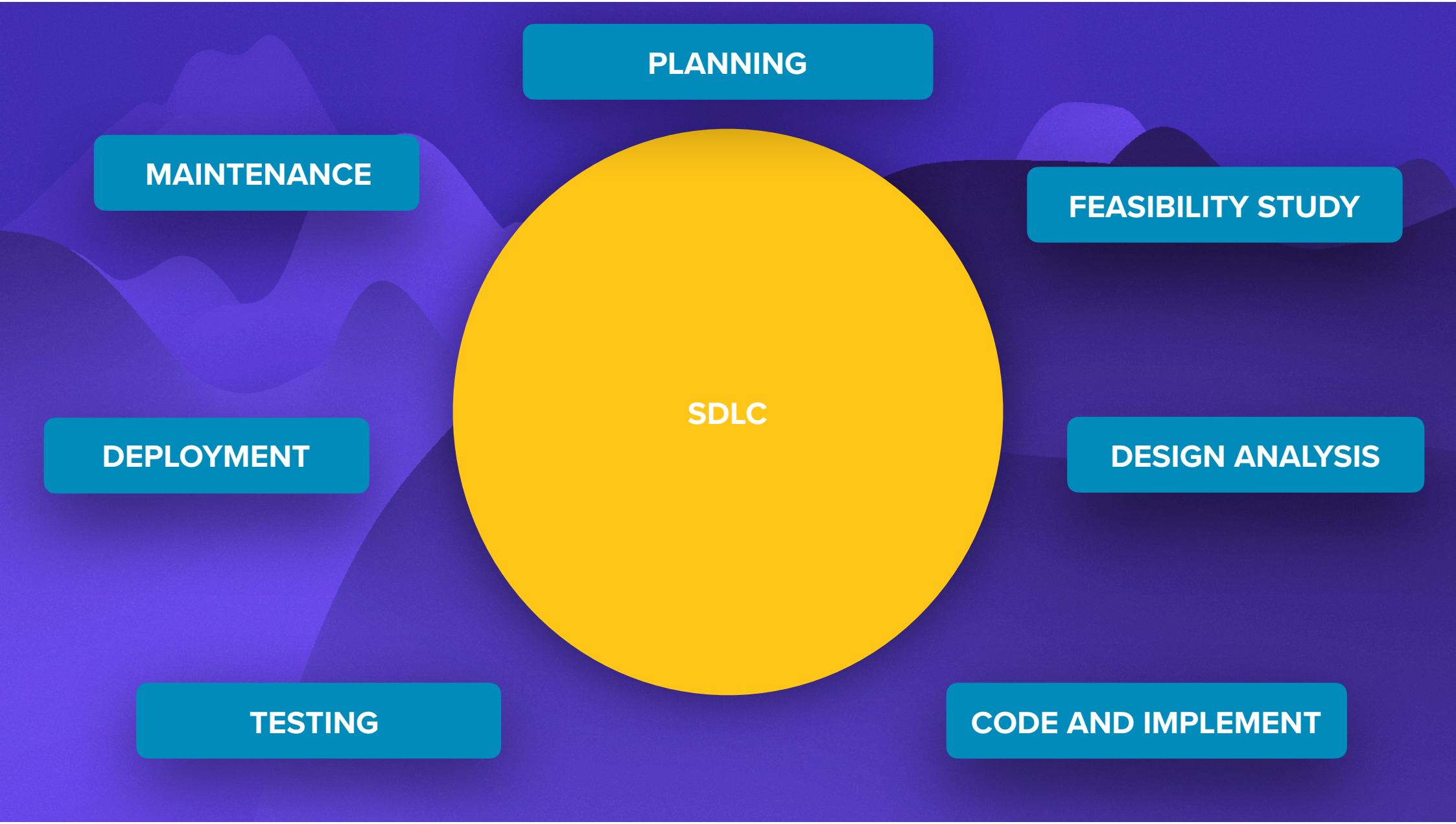
The The structured approach helps to create software that is easier to maintain in the long term.Post-deployment, SDLC's maintenance phase allows for software updates, bug fixes, and upgrades without disrupting the software's core functionality.

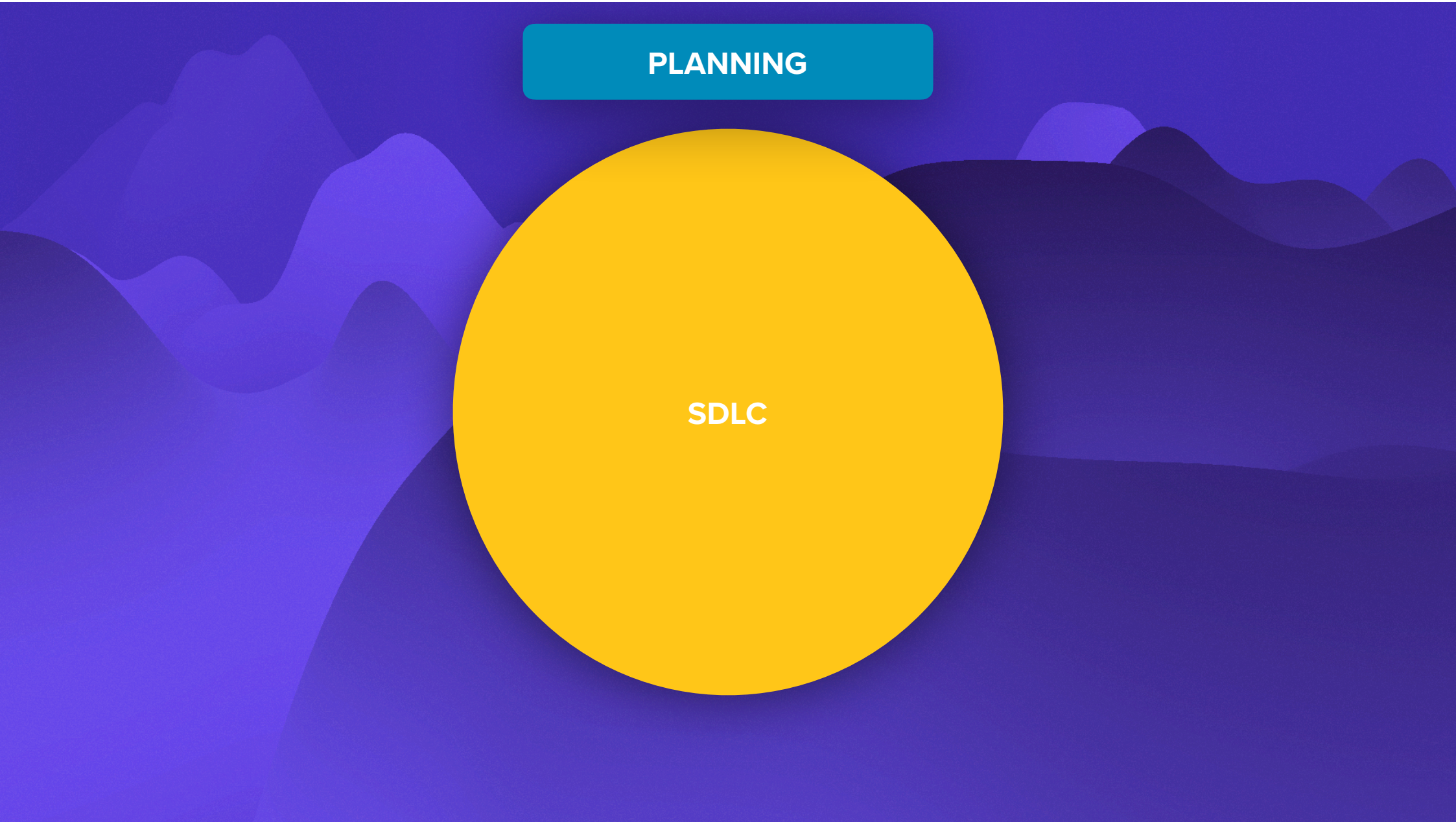
6. Cost Management:

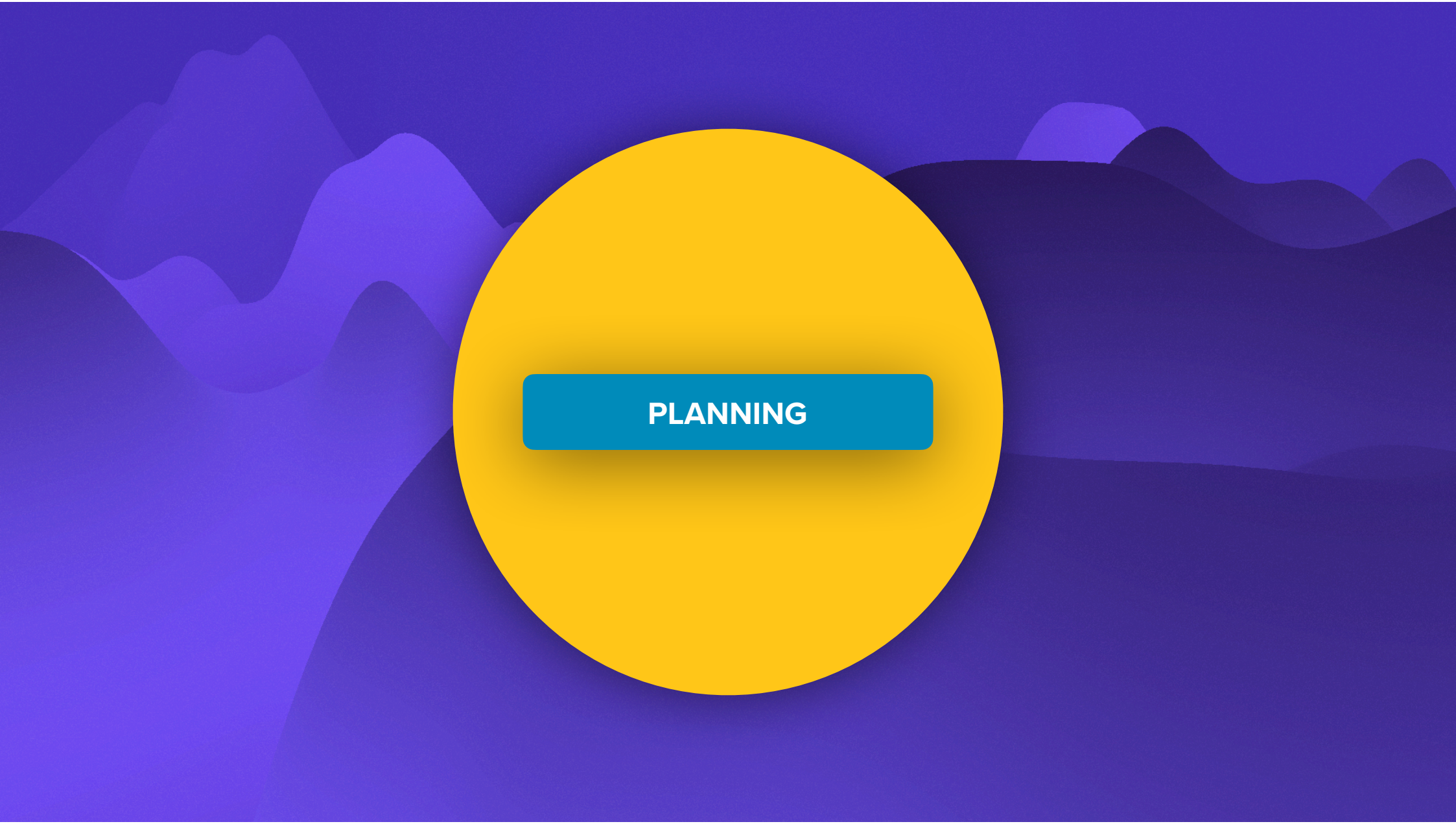
The upfront planning in SDLC helps in budgeting and reduces the likelihood of expensive changes or revisions down the line.If the project follows a defined path, the chances of going over budget or timeline are minimised.




SDLC


PLANNING
FEASIBILITY STUDY
DESIGN ANALYSIS
CODE AND IMPLEMENT
TESTING
DEPLOYMENT
MAINTENANCE
SDLC


PLANNING
SDLC


PLANNING

PLANNING

Planning Phase in the Software Development Life Cycle (SDLC) is the foundation of the entire project, setting the stage for success by aligning the software development process with business objectives and stakeholder expectations. During this phase, the primary goal is to define the project scope, including what the software will and will not do. It involves gathering detailed requirements from stakeholders, including clients, users, and business analysts, to ensure the product meets their needs. For instance, if a company is developing a customer relationship management (CRM) system, they would identify features like customer data management, sales tracking, and reporting as core functionalities.

Budget estimation during the planning phase involves calculating the financial resources required for the entire project, considering various factors such as labor, infrastructure, tools, and external services. For an example, let's consider the development of a Customer Relationship Management (CRM) software. Human Resources: Salaries of project managers, software developers, business analysts, testers, designers, and other team members involved in the project. Example: A software developer's monthly salary may be 40,000, and if the project requires 3 developers for 6 months, that cost will be 1,20,000.

Time estimation during the planning phase involves determining how long each stage of the project will take, ensuring that development progresses smoothly without delays. The planning phase sets the timeline for all the subsequent phases in SDLC.

Requirement Gathering: Typically, 2-4 weeks, depending on the complexity and the number of stakeholders involved. For a CRM system, this may include gathering business and technical requirements from stakeholders and users.


FEASIBILITY STUDY

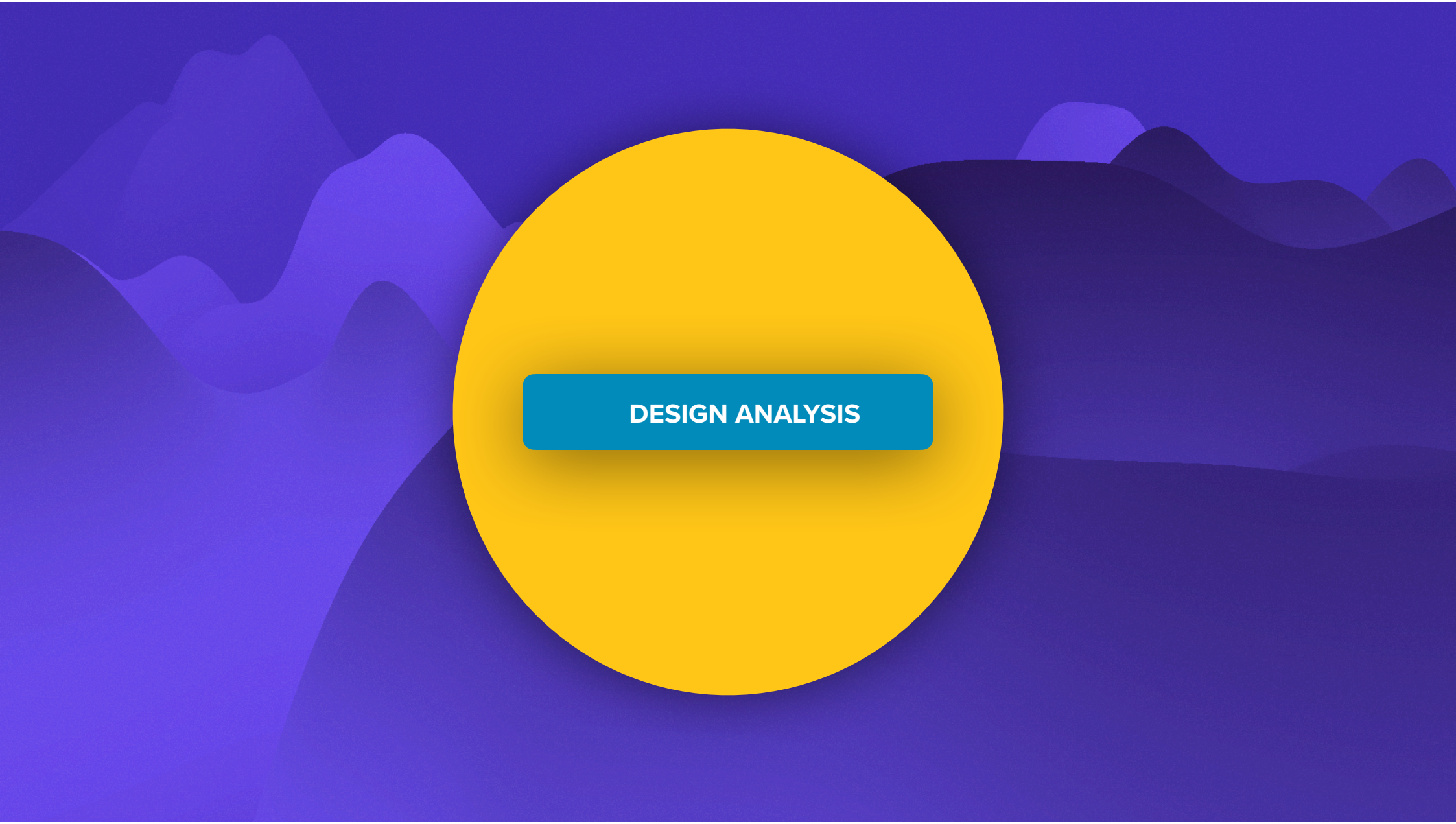
FEASIBILITY STUDY

The Feasibility Study is an essential part of the Planning Phase in the Software Development Life Cycle (SDLC). Its primary purpose is to assess whether a project is viable in terms of technical, financial, and operational factors before proceeding further. This study helps identify potential risks, challenges, and limitations, providing a clear picture of whether the project can be completed successfully within the given constraints, such as time, budget, and available resources. It ensures that the project is worth pursuing, saving time and resources if it's deemed unfeasible early on.

The first aspect of the feasibility study is technical feasibility. This evaluates whether the proposed solution can be built using the existing technology stack, tools, and resources. It also examines if the development team has the necessary skills and expertise to implement the project. For example, if a company is planning to build a Customer Relationship Management (CRM) system that requires integrating with an existing database and incorporating advanced data analytics, the feasibility study will assess whether the current database technology supports these requirements and if the team has the necessary knowledge to work with complex integrations.

Financial feasibility focuses on whether the project can be completed within the allocated budget. This includes estimating development, testing, and deployment costs. If, for example, the CRM system requires expensive third-party tools, this could affect the financial feasibility.

Finally, operational feasibility assesses whether the organization has the resources to support the new system. For a CRM system, this includes ensuring the company's IT infrastructure can handle the new software and that employees are trained to use it effectively.


DESIGN ANALYSIS

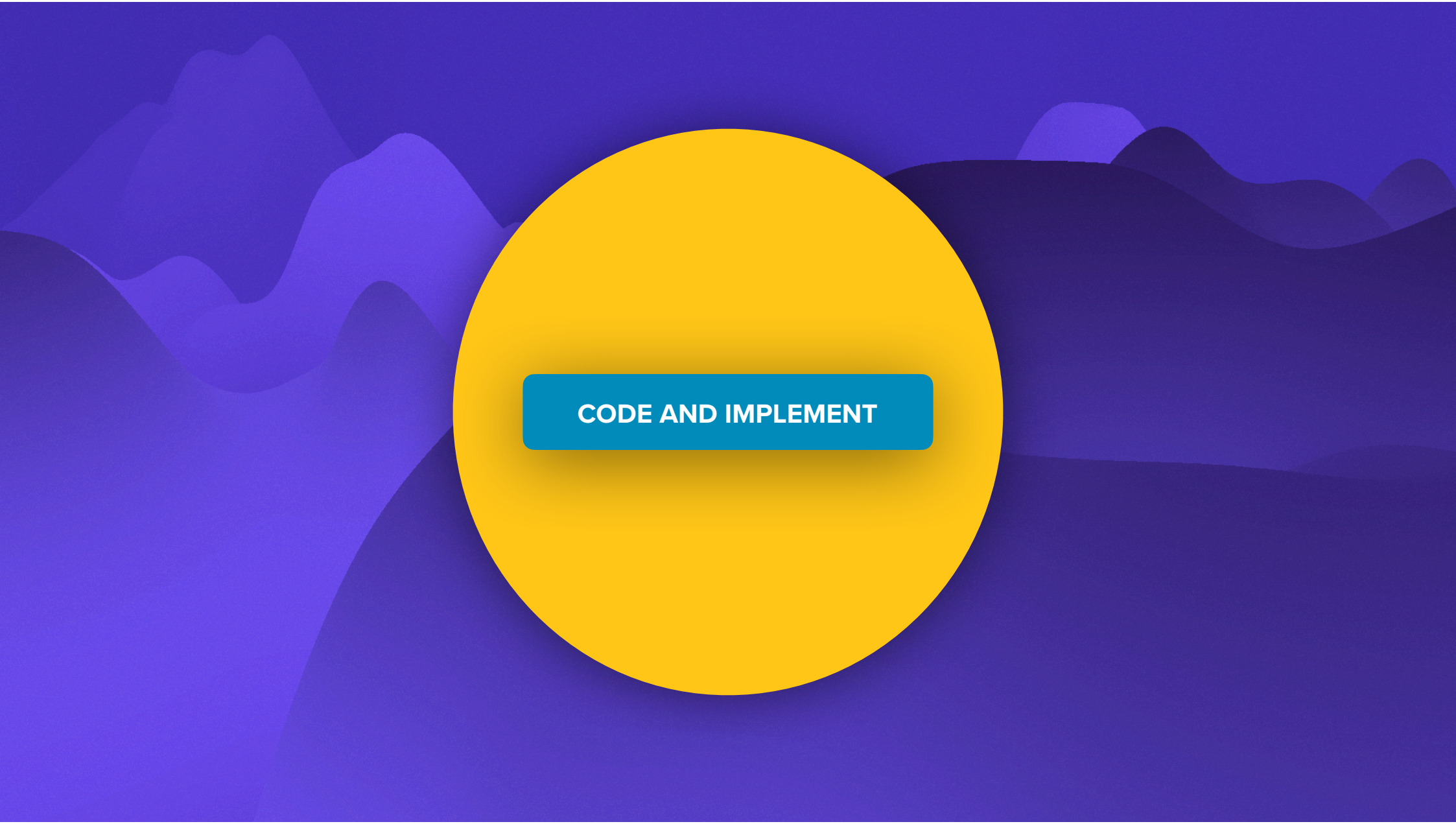
DESIGN ANALYSIS

Design Analysis is a crucial phase in the Software Development Life Cycle (SDLC) that comes after the feasibility study and requirement gathering. During this phase, the software's architecture and system design are planned and outlined in detail to ensure that the final product meets the specified requirements. Design analysis focuses on translating the requirements into a blueprint for the software solution, detailing how the system will be built, how it will function, and how it will interact with other systems.

In the design analysis phase, two main types of design are created: high-level design (also known as architectural design) and low-level design. The high-level design defines the overall architecture of the system, including the software's modules, components, and how they will interact. This stage involves selecting the right technology stack, designing the system architecture, and defining the overall flow of data. For example, when designing a CRM system, this phase will decide how the user interface will interact with the database, define the modules for user management, sales tracking, reporting, etc., and ensure the system's scalability.

The low-level design focuses on the detailed design of individual modules, functions, and databases. This includes defining specific algorithms, data structures, and interactions at a more granular level. In the case of the CRM system, the low-level design would involve creating detailed diagrams for each feature, such as how the sales tracking module will handle data input, processing, and reporting.

Finally, design analysis ensures that the system will be scalable, secure, and maintainable. It involves selecting tools, frameworks, and platforms that will allow the system to grow as required, ensuring the software can accommodate future upgrades or changes without major rework. It also assesses potential risks related to performance, security, and integration, planning for them early on to minimize future issues.


CODE AND IMPLEMENT

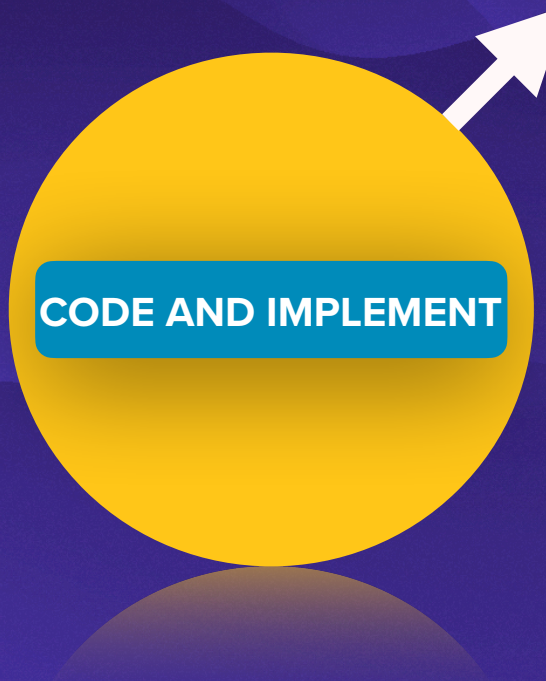
CODE AND IMPLEMENT

Coding and Implementation is a critical phase in the Software Development Life Cycle (SDLC) where the actual development of the software begins. During this phase, developers write the source code based on the design specifications established in the earlier phases. The focus is on translating the design into a working software product, ensuring that all features and functionalities defined in the design phase are properly implemented.

In the coding phase, developers use the chosen programming languages, frameworks, and tools to build the system. For example, in a CRM system, developers might use JavaScript, Python, or Ruby on Rails for the backend, and HTML, CSS, and React for the front end. The coding process involves building out the database, developing business logic, and creating user interfaces. Developers must follow coding standards and guidelines to ensure code readability, maintainability, and consistency.

Implementation refers to the process of deploying the developed software into a live environment. Once the coding is complete and the system is built, it is moved from the development environment to the staging or production environment. In this phase, the software is configured, and the required resources, such as servers and databases, are set up. For example, a CRM system may be deployed on cloud platforms like AWS or Azure, where it will be accessible by end users.

In summary, coding and implementation is where the design turns into a working product. It's the phase where developers write, integrate, and deploy the software, followed by initial tests to ensure functionality.




DEPLOYMENT

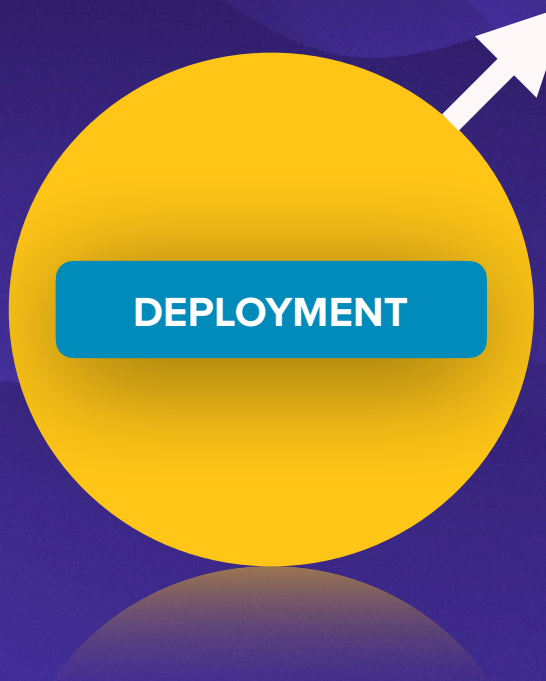
DEPLOYMENT

Deployment is the final phase in the Software Development Life Cycle (SDLC), where the software is made available for end-users. After successful development and testing, the system is installed in the production environment, such as on servers or cloud infrastructure, so that it can be accessed and used. This phase ensures that the software is operational and accessible to users in real-world conditions, and the system is configured for its specific production requirements.

During deployment, the software is integrated with existing systems and databases. For example, a CRM system might need to connect with an organization's email platform or customer database. Proper integration ensures smooth communication between systems and guarantees that the software functions as intended across all components, reducing the risk of operational issues.

Data migration is also an essential part of the deployment phase, particularly if the software replaces an older system or needs to work with pre-existing data. Migrating data accurately ensures that all necessary information is transferred without errors, allowing the new system to function seamlessly. In a CRM system, this might include moving customer records, sales data, and historical interactions into the new system.Requirement Gathering: Typically, 2-4 weeks, depending on the complexity and the number of stakeholders involved. For a CRM system, this may include gathering business and technical requirements from stakeholders and users.

Once deployed, the software undergoes User Acceptance Testing (UAT), where end-users validate that the system works as expected in a real-world environment. This testing ensures that all features and functions are working properly. After deployment, continuous monitoring and maintenance are necessary to address any performance issues, bugs, or required updates, ensuring the software remains effective and secure for its users.




MAINTENANCE

MAINTENANCE

Maintenance is a critical phase in the Software Development Life Cycle (SDLC) that begins once the software has been deployed and is in use by end-users. It involves making updates, fixing bugs, improving performance, and ensuring that the system continues to function effectively over time. The maintenance phase ensures that the software remains secure, reliable, and aligned with evolving user needs and business requirements.

There are different types of maintenance, including corrective maintenance, adaptive maintenance, and perfective maintenance. Corrective maintenance focuses on fixing defects or bugs found after deployment, such as addressing any errors in functionality. Adaptive maintenance involves updating the software to remain compatible with new environments, such as updated operating systems or third-party services. Perfective maintenance aims to improve the software by adding new features or optimising performance, like enhancing user interfaces or increasing the system's speed.

Maintenance is crucial for keeping the software up to date and efficient. It ensures the software adapts to changing technology and continues to meet the needs of users. For instance, if the CRM system needs to support more customers, the software might require updates to handle increased data volume or to integrate with new tools, such as social media platforms.

Overall, the maintenance phase is vital for ensuring that the software remains functional, secure, and relevant throughout its lifecycle. It is an ongoing process that helps address issues, improve the system, and keep it aligned with both business needs and technological advancements.

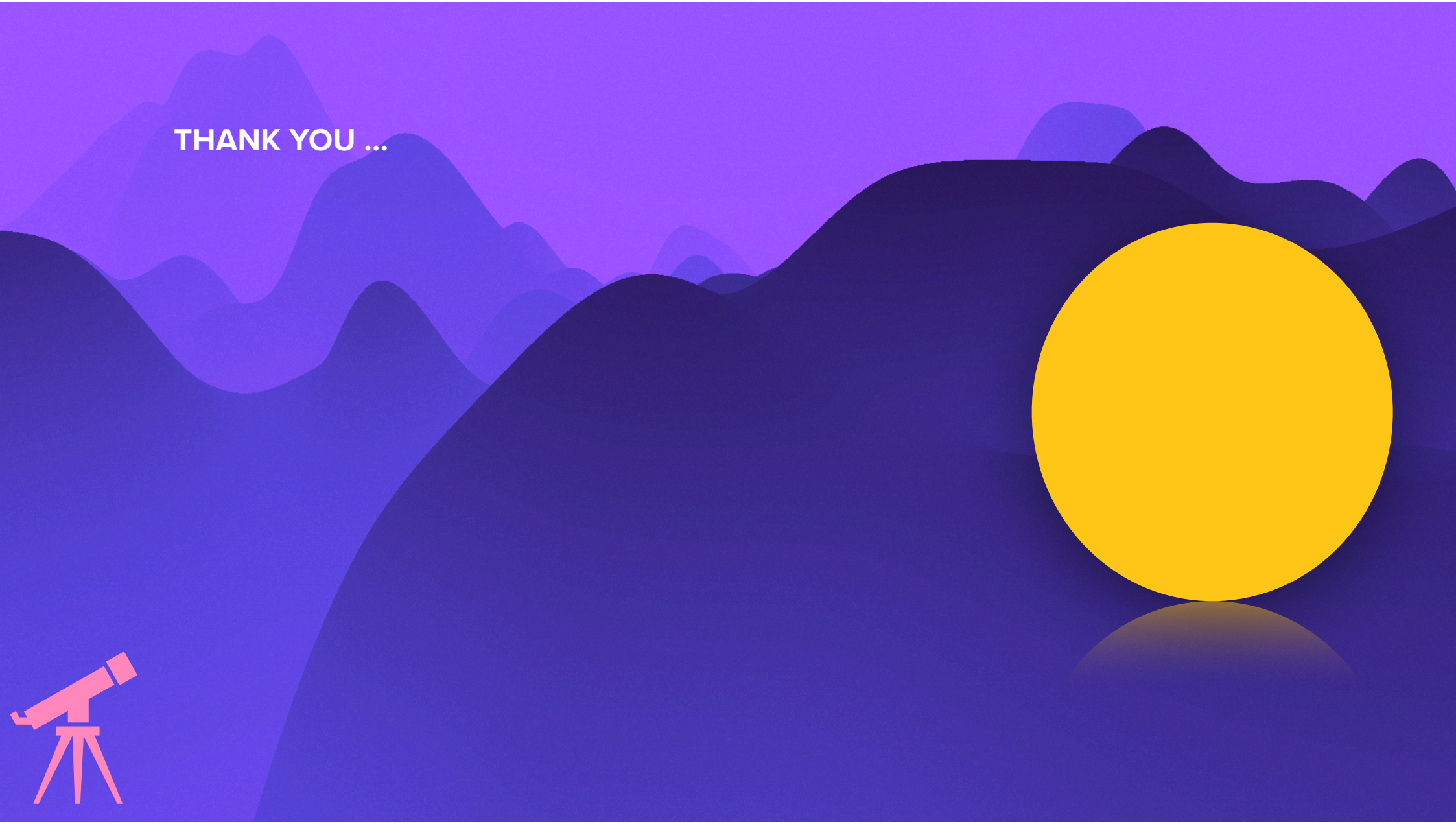
CONCLUSION

CONCLUSION

In conclusion, the Software Development Life Cycle (SDLC) is a comprehensive framework that ensures systematic, efficient, and structured development of software. From the initial planning and feasibility study to the final deployment and ongoing maintenance, each phase plays a crucial role in delivering a high-quality product that meets both technical and business requirements. Through our seminar project, we have explored the different stages of SDLC, including the importance of proper management, design analysis, coding, and testing.

As understanding and applying the SDLC is essential for successfully developing, deploying, and maintaining software. By adhering to SDLC principles, teams can mitigate risks, ensure timely delivery, and provide software that is scalable, secure, and adaptable to future needs.

We hope this seminar has provided valuable insights into the critical processes of software development and highlighted the significance of a structured approach to creating effective and reliable software solutions. Thank you for your attention, and we are open to any questions or discussions.


THANK YOU ...