TABLE OF CONTENT

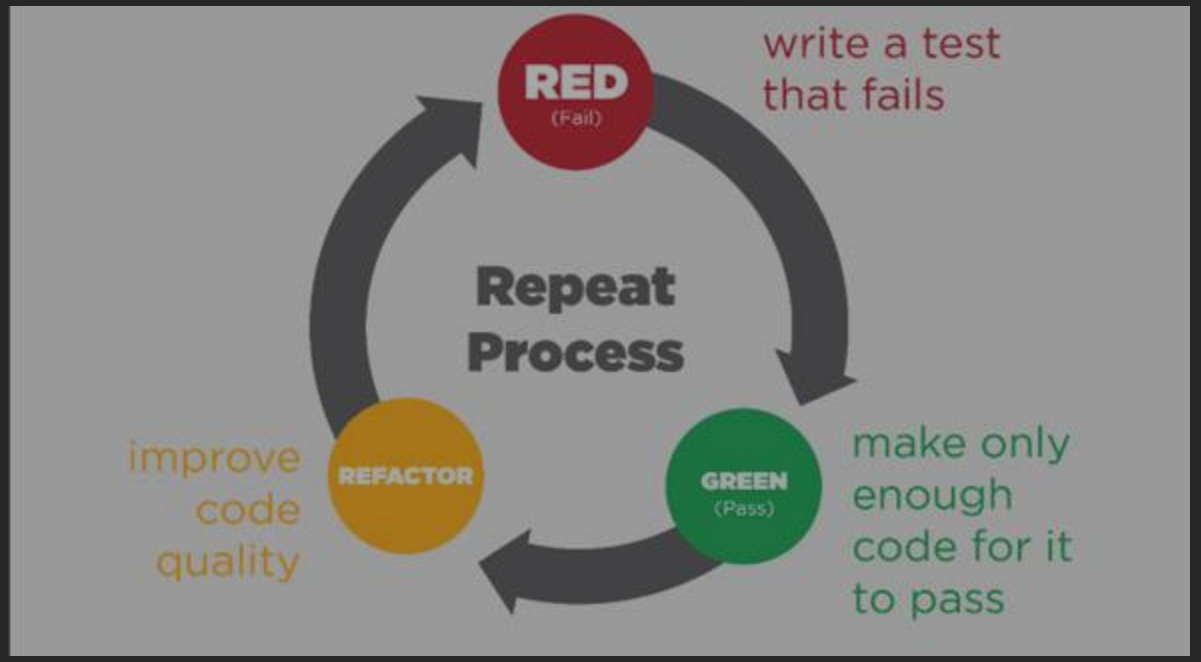
PRESENTED BY: GOPAL RANJAN

TEST DRIVEN DEVELOPMENT

- Test Driven Development ensures code is always tested and functional reducing bugs and improving code quality.
- Test-driven development (TDD) is a method of coding in which you first write a test, and it fails, then write the code to pass the test of development and clean up the code.
- A method in software development where the focus is on writing tests before writing the actual code for a feature.

Steps for TDD :

- Write small focused tests defining desired functionality.
- Write minimum code necessary to pass these tests.
- Refactor code to improve structure and performance.



Process of TDD:

- **Red** – Create a test case and make it fail, Run the test cases.
- **Green** – Make the test case pass by any means.
- **Refactor** – Change the code to remove duplicate/redundancy.

Advantage of TDD

- Unit test provides constant feedback about the functions.
- Quality of design increases further helping in maintenance.
- Test driven development act as a safety net against the bugs.
- Ensures that your application meets the requirements defined for it.
- Have very short development life cycle.

Disadvantage of TDD

- Increased Code Volume
- False Security from Tests
- Maintenance Overheads
- Time Consuming Test Processes
- Testing Environment Set-up

Approaches of TDD:

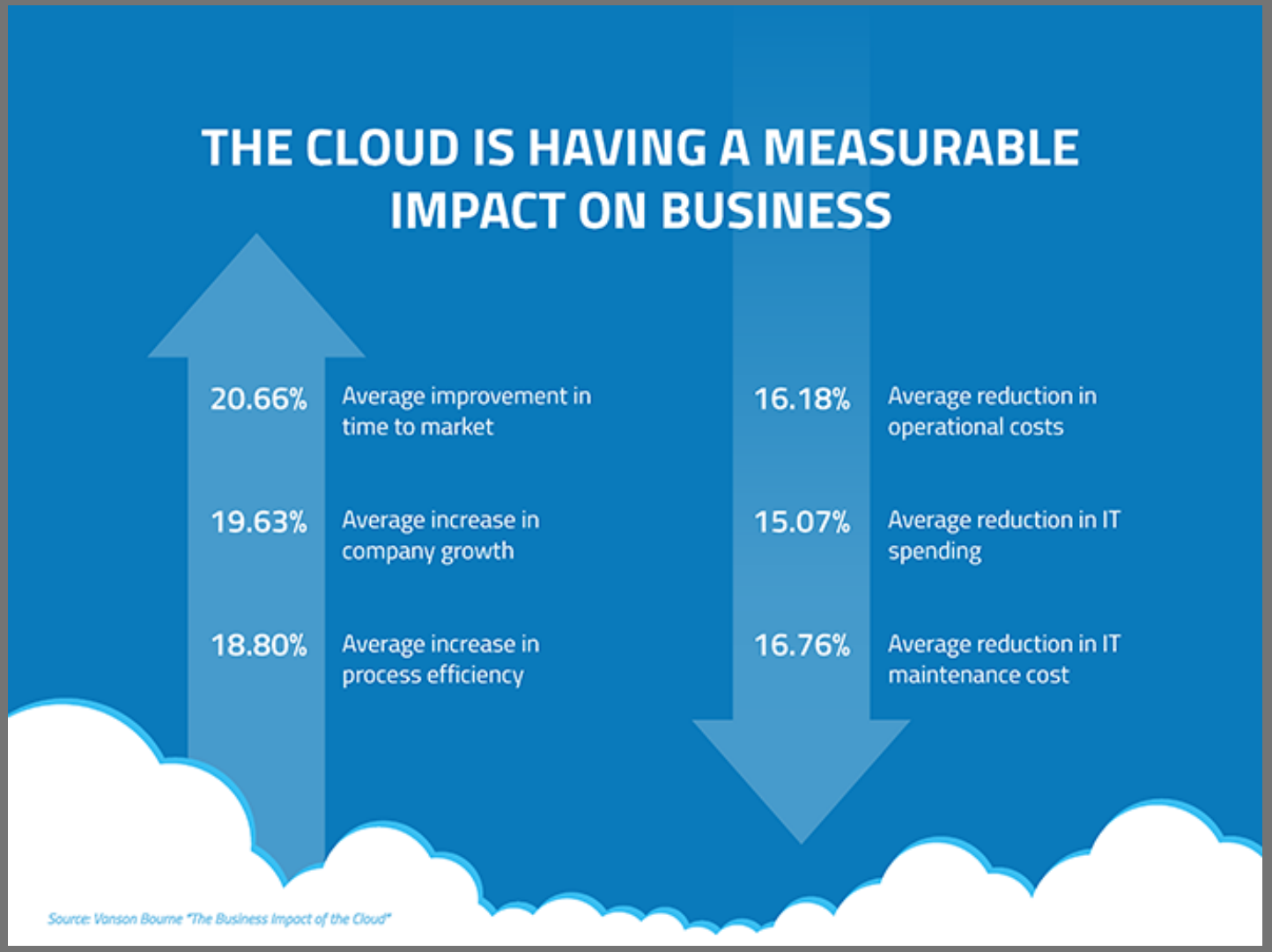
INSIDE OUT (DETROIT SCHOOL OF TDD or CLASSICIST)

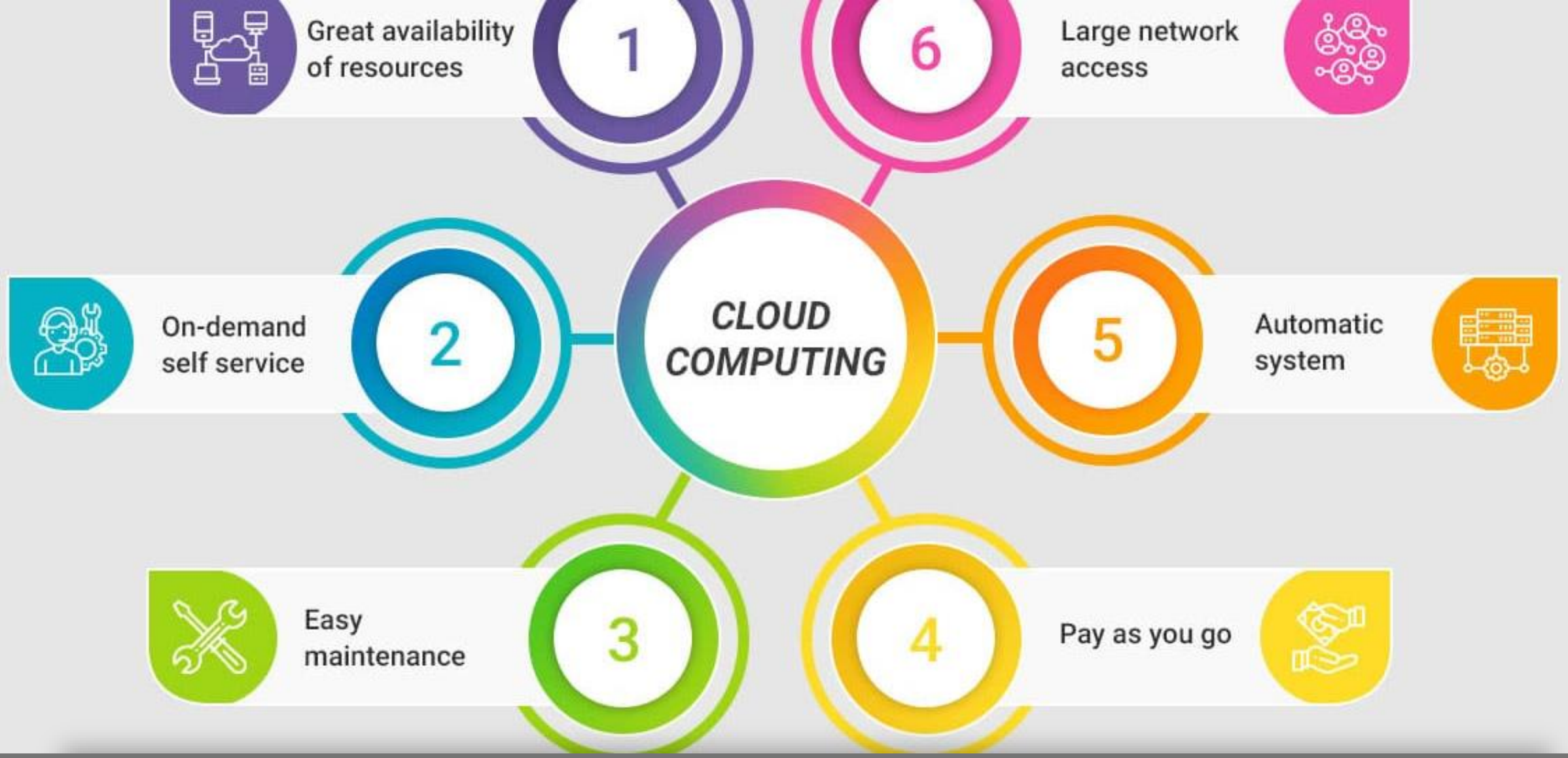
- Focuses on testing the smallest units first and building up from there
- Easier to learn for beginners
- Minimizes the use of mocks
- Helps prevent over-engineering
- Design and architecture are refined during the refactor stage, which can sometimes lead to significant changes

OUTSIDE IN (LONDON SCHOOL OF TDD or MOCKIST)

- Focuses on testing user behaviour and interactions
- Testing starts at the outermost level, such as the user interface, and works inward to the details
- Relies heavily on mocks and stubs to simulate external dependencies
- Harder to learn but ensures the code meets overall business needs
- Design is considered during the red stage, aligning tests with business requirements from the start

CLOUD COMPUTING & BUSINESSES





BENEFITS OF CLOUD COMPUTING

ANALYTICS ROLE IN BUSINESS

PROCESSES INVOLVED IN BUSINESS ANALYTICS

DATA COLLECTION

DATA ANALYSIS

PREDICTIVE MODELING

DATA VISUALIZATION

IMPLEMENTATION & MONITORING

Improved Customer Satisfaction

Increased Sales & Profits

Enhanced Decision Making

Strong Governance

HOW ANALYTICS HELP BUSINESSES GROW?

OWASP

Open Web Application Security Project

The **OWASP Top 10** is a list of the most critical security risks to web applications, published by the **Open Web Application Security Project (OWASP)**. It is updated periodically to reflect emerging threats and industry trends.

1. **Broken Access Control:** Unauthorized users can access restricted data or functionalities due to misconfigured permissions.
2. **Cryptographic Failures:** Weak or missing encryption leads to exposure of sensitive data.
3. **Injection:** Malicious input (e.g., SQL, command, or LDAP injection) is executed by the system.
4. **Insecure Design:** Weaknesses in software architecture leads to security flaws that cannot be patched easily.
5. **Security Misconfiguration:** Improperly configured security settings expose vulnerabilities.
6. **Vulnerable and Outdated Components:** Using outdated libraries, frameworks, or third-party services with known security flaws.
7. **Identification and Authentication Failures:** Weak authentication mechanisms allow attackers to hijack user accounts.
8. **Software and Data Integrity Failures:** Insecure software updates, third-party dependencies, or CI/CD pipeline vulnerabilities.
9. **Security Logging and Monitoring Failures:** Insufficient logging and monitoring make it hard to detect and respond to breaches.
10. **Server-Side Request Forgery:** Attackers manipulate a server into making unauthorized requests.

THE TWELVE FACTOR APP

- Software is commonly delivered as a service in the modern era called web apps or software as a service(SaaS).

THE TWELVE-FACTOR APP

- The 12 Factor App is a methodology for building modern, scalable, and maintainable software-as-a-service (SaaS) applications. It was developed by engineers at Heroku and provides best practices for application development.

i. **Codebase** – A single codebase tracked in version control (e.g., Git), with multiple deployments (staging, production, etc.).

ii. **Dependencies** – Declare all dependencies explicitly in a dependency manager (e.g., package.json, Pipfile) instead of relying on system-wide packages.

iii. **Config** – Store configuration (e.g., database URLs, API keys) in environment variables, not in the codebase.

iv. **Backing Services** – Treat external services (e.g., databases, caches, queues) as attached resources that can be swapped easily.

v. **Build, Release, Run** – Separate build (compilation, dependencies), release (configuration, metadata), and run (execution) stages.

vi. **Processes**– Applications should be stateless and share-nothing, with all persistent data stored in a backing service (e.g., a database).

vii. **Port Binding** – Applications should be self-contained and expose services via ports (e.g., web servers running on a specific port).

viii. **Concurrency**– Scale applications by running multiple processes instead of relying on threads.

ix. **Disposability**– Applications should start and stop quickly, enabling rapid scaling and resilience.

x. **Dev/Prod Parity** – Keep development, staging, and production environments as similar as possible to avoid inconsistencies.

xi. **Logs**– Treat logs as event streams, directing them to a centralized logging service instead of managing files.

xii. **Admin Processes**– Run one-off administrative tasks (e.g., database migrations) as separate processes.

AGILE MANIFESTO

Core Values:

1. **Individuals and interactions** over processes and tools.
2. **Working software** over comprehensive documentation.
3. **Customer collaboration** over contract negotiation.
4. **Responding to change** over following a plan.

12 AGILE PRINCIPLES:

1. **Customer Satisfaction –** Deliver valuable software early and continuously.
2. **Welcome Change –** Adapt to changing requirements, even late in development.
3. **Frequent Delivery –** Deliver working software in short cycles (weeks to months).
4. **Collaboration –** Business stakeholders and developers must work together daily.
5. **Support and Trust –** Build projects around motivated individuals and give them autonomy.
6. **Face-to-Face Communication –** The most efficient way of conveying information is direct conversation.
7. **Working Software –** The primary measure of progress is a working product.
8. **Sustainable Development –** Maintain a constant, manageable pace indefinitely.
9. **Technical Excellence –** Continuous attention to technical quality and good design enhances agility.
10. **Simplicity –** Maximize the work not done by focusing only on what's necessary.
11. **Self-Organizing Teams –** The best designs and solutions emerge from empowered teams.
12. **Reflection and Improvement –** Teams should regularly reflect and adjust their behaviour to improve efficiency.

ASSIGNMENT LINK:

- VIEW TASK

THANK YOU !!!