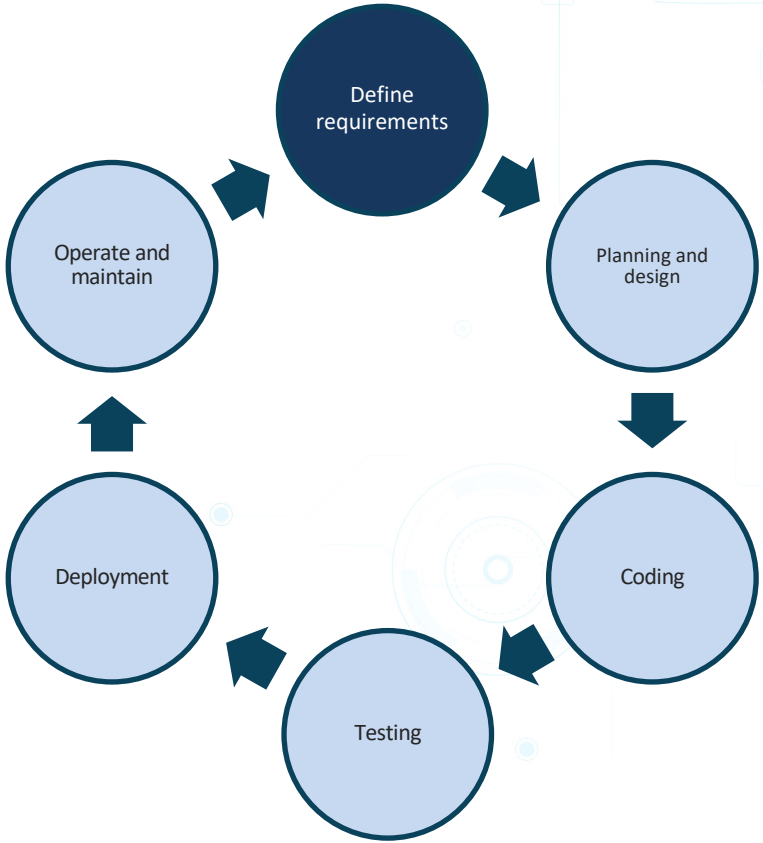
Software Development Life Cycle (SDLC)

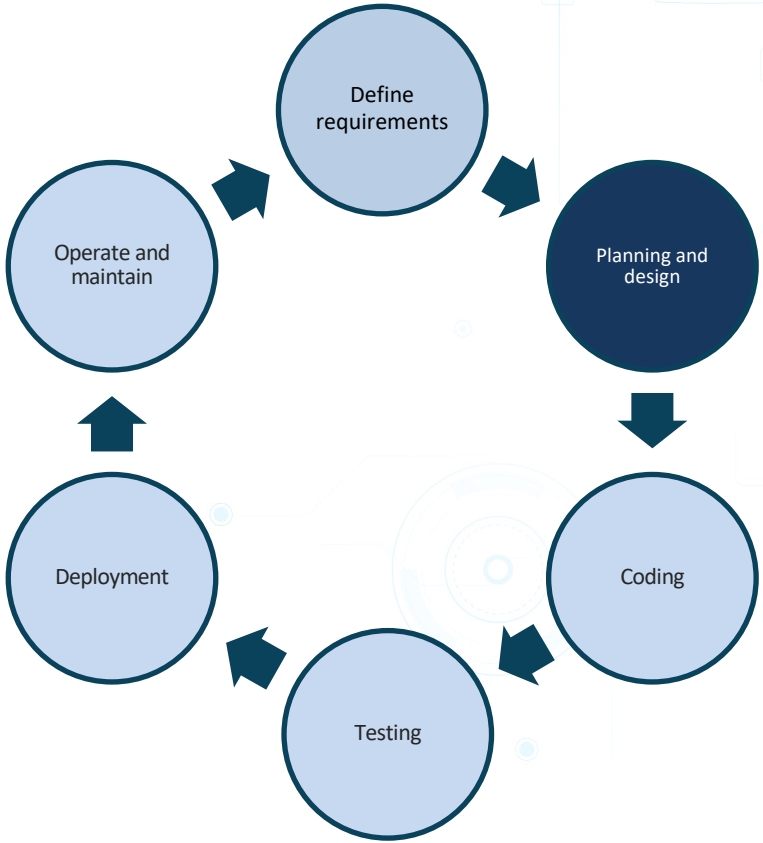
- Used to manage complex projects
- Assess secure coding practices
  - OWASP Top 10
  - SANS Institute
  - Center for Internet Security

SDLC Phases



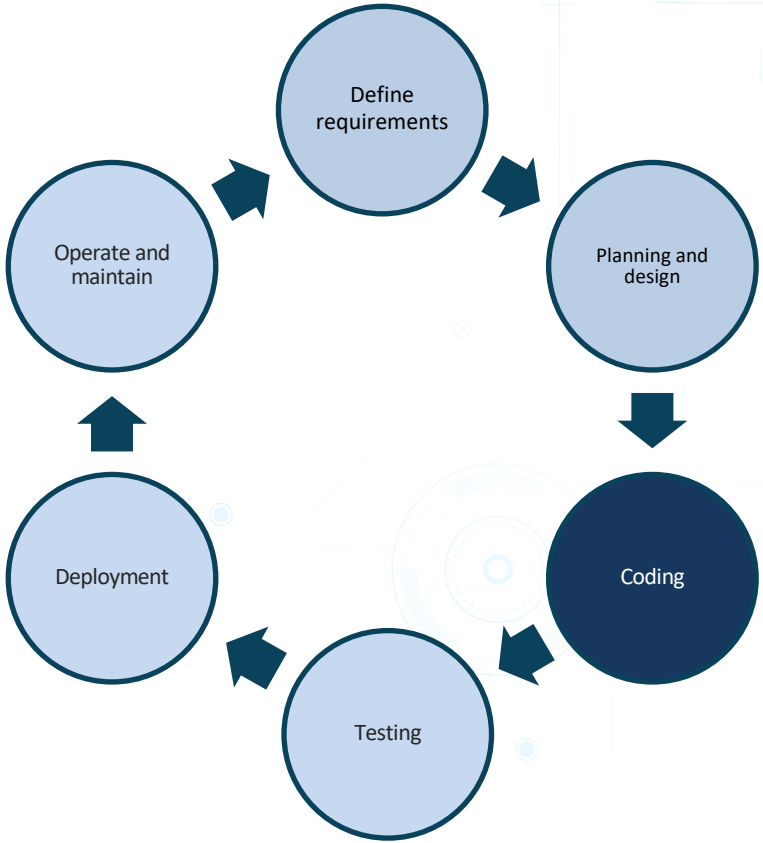
- Clearly define business need
- Risk assessment
- Define inputs and outputs

SDLC Phases



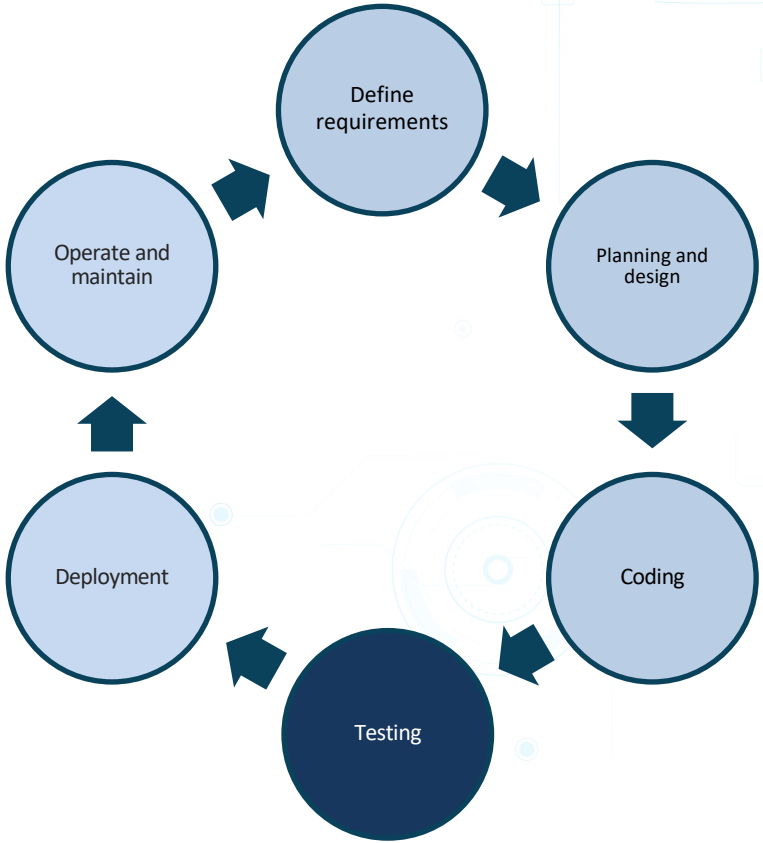
- Mobile device app
- Where the solution may be used
- Legal/regulatory compliance
- High availability
- Assemble team

SDLC Phases



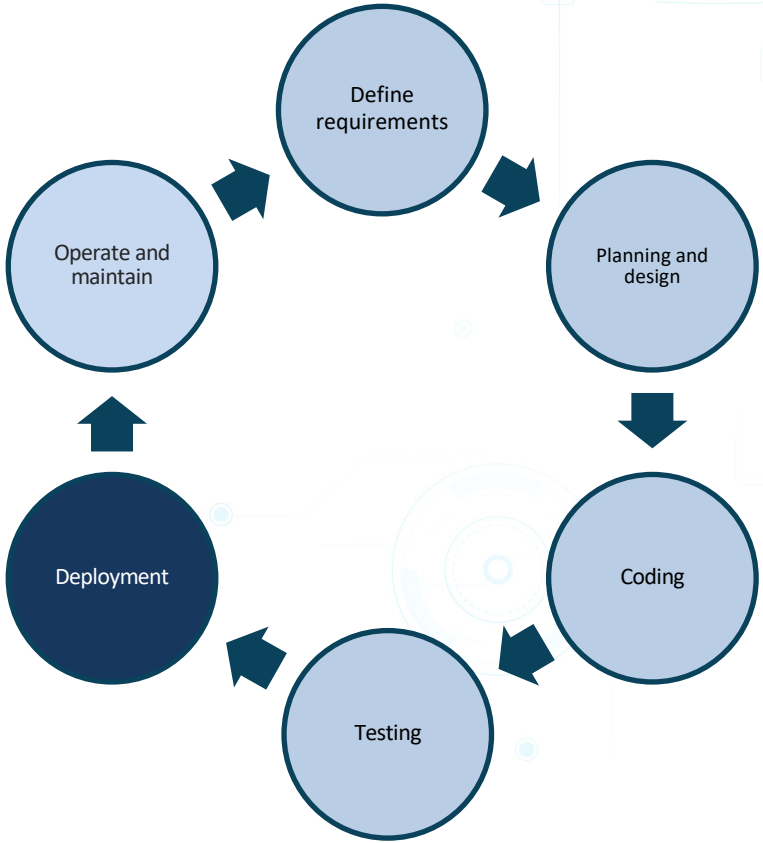
- Secure coding practices
- Peer review

SDLC Phases



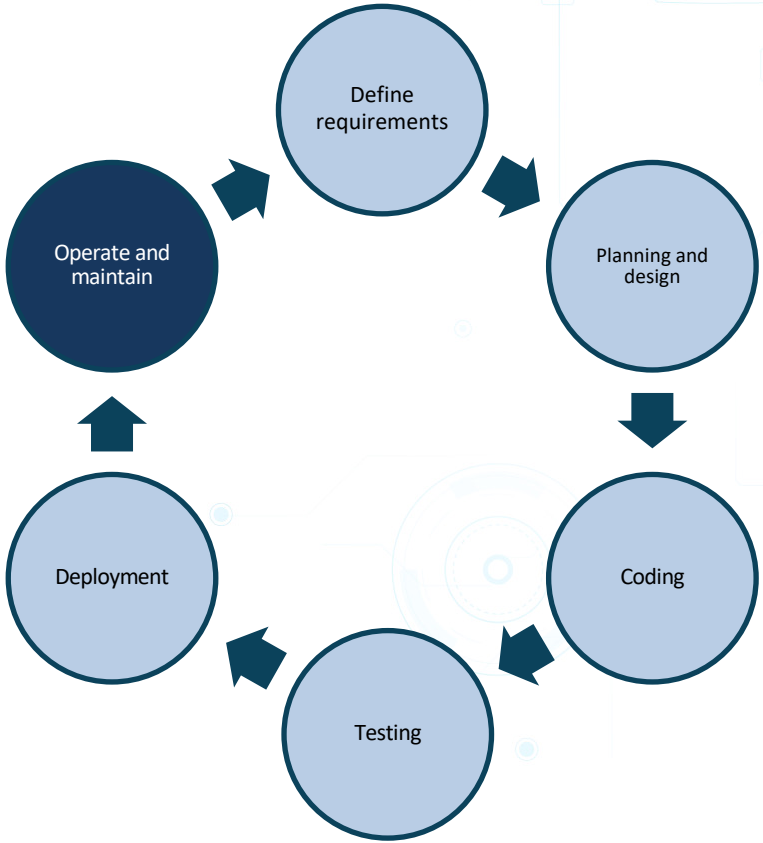
- Testing types
  - Unit, functional, user acceptance, regression

SDLC Phases



- Pilot testing for a small group
- Continuous integration and deployment

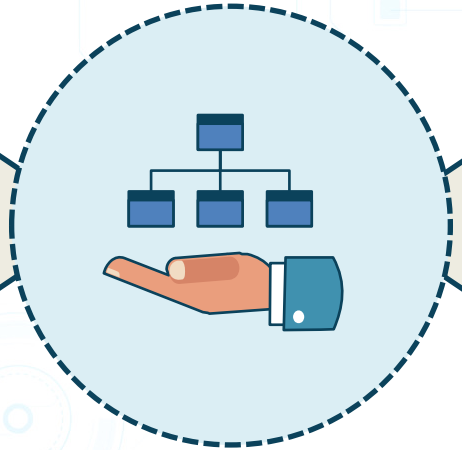
SDLC Phases



- Help desk tickets
- Adding users
- Updates
- Security control review

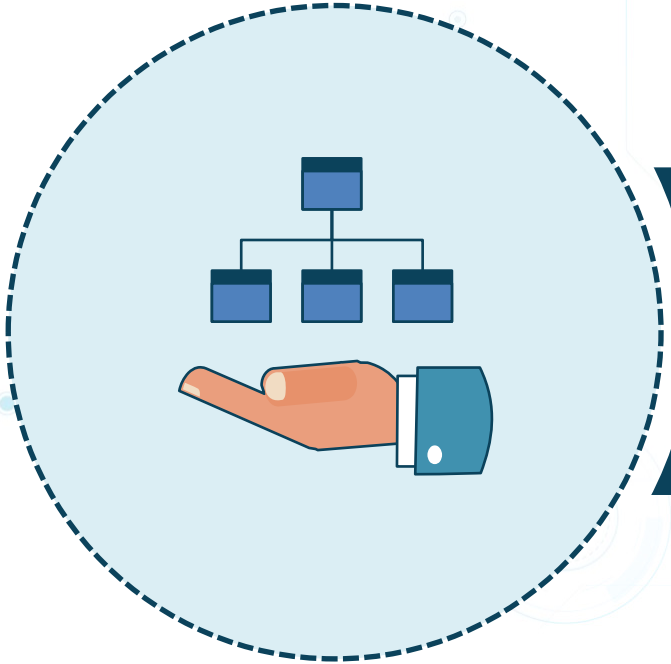
IT Project Management (ITPM)

Structured approach



IT alignment with business goals

IT Project Management

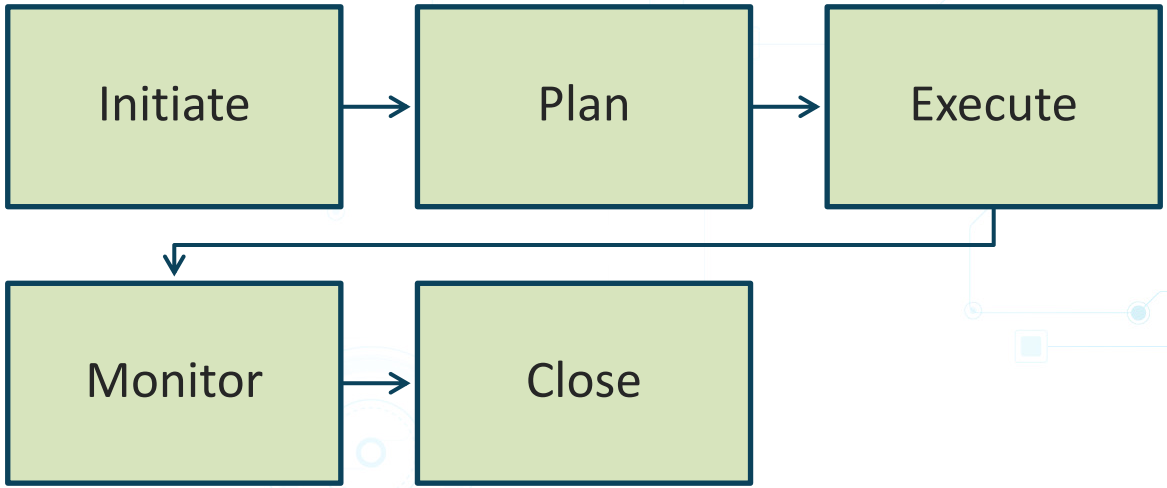


- On time
- Compliance
- On budget

Project Management Considerations

- Risk management
- Team and resource management
- Supply chain and outsourcing management
- Manage stakeholder expectations

The Project Management Process



Secure Coding

- Security must be a part of each SDLC phase
- Must have a clear definition of security requirements

Secure Coding

- Peer review
  - Another set of eyes
- Input validation
- Best practices
  - OWASP Top 10
  - SANS
  - Center for Internet security

Secure Coding

Continuous integration and delivery (CI/CD)

Be wary of third-party code

Code analysis tools (FindBugs, Androwarn)

Software developer security awareness and training

Software Testing

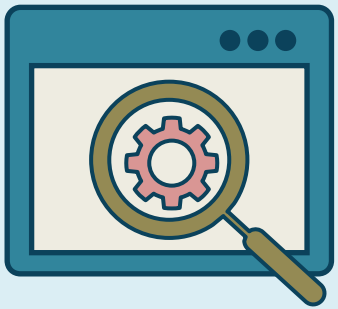
Quality assurance

Security

Software Testing

- Static code analysis
- Run-time testing (dynamic testing)
  - Set security and performance baseline first
- Testing environment
  - Sandbox
  - Network and host virtualization
  - Testing tools such as fuzzers

Software Testing

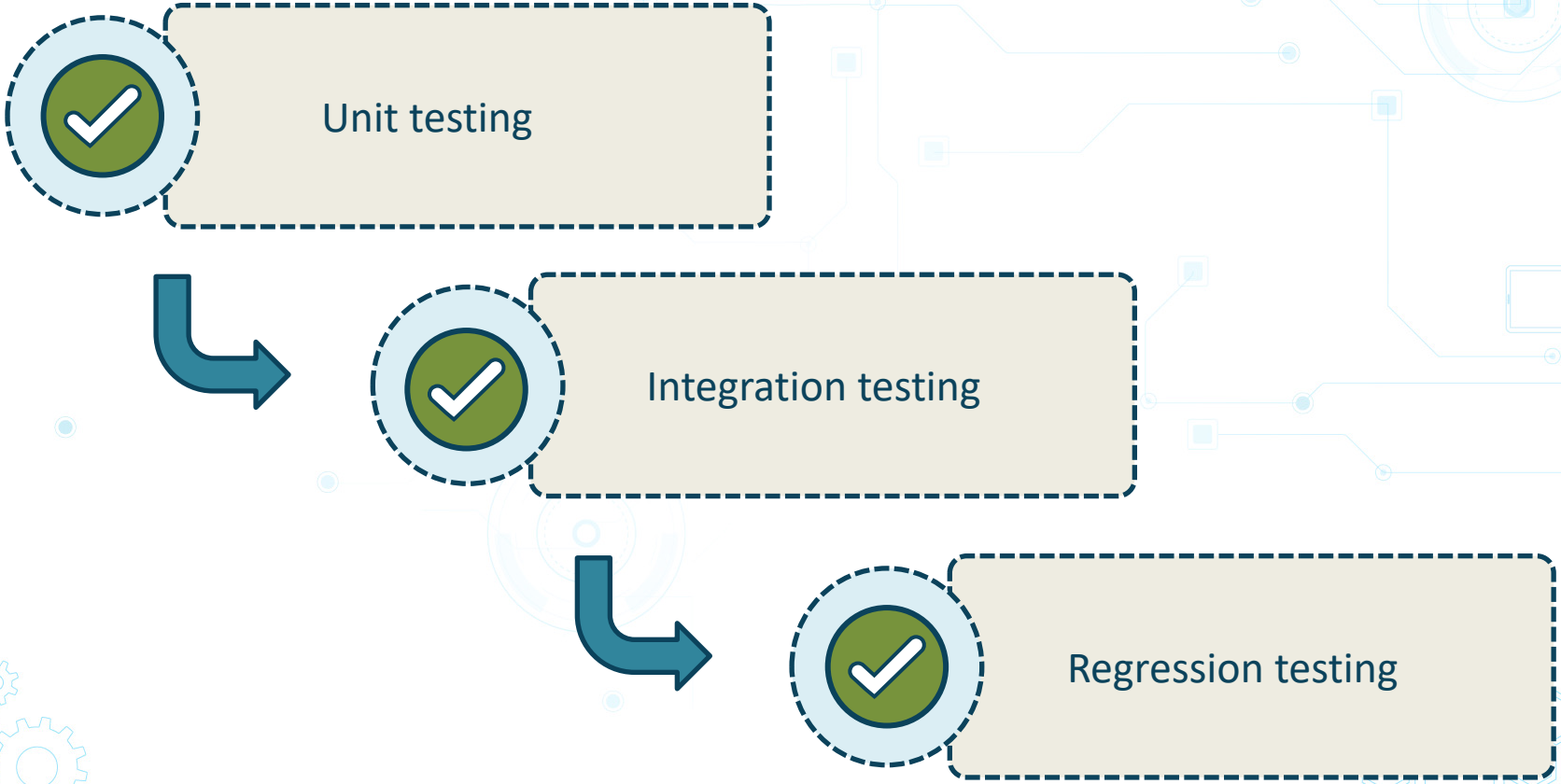


- Web application vulnerability scan
- Manual or automated tools
- Check for
  - Misconfiguration
  - Directory traversal
  - SQL injections
  - Remote command execution

Unit Testing

- Evaluation of code segments
  - Sub routines
  - Code modules
  - Object class, method, and property
  - Component such as a web browser plug-in

Unit Testing Position



Unit Testing

- No dependencies on other external code components
- Testing is performed by the software developer
- Automated tools such as RSpec

Regression Testing

- Triggered by changes or integration testing
- Can be automated
- Have changes introduced new problems?

Regression Testing

- Run tests again
  - Unit testing
  - Integration testing
- Selective testing
  - Prioritize critical software components or features
  - Items that users will be most impacted by

Regression Testing

Prevent code changes during testing

Re-testing

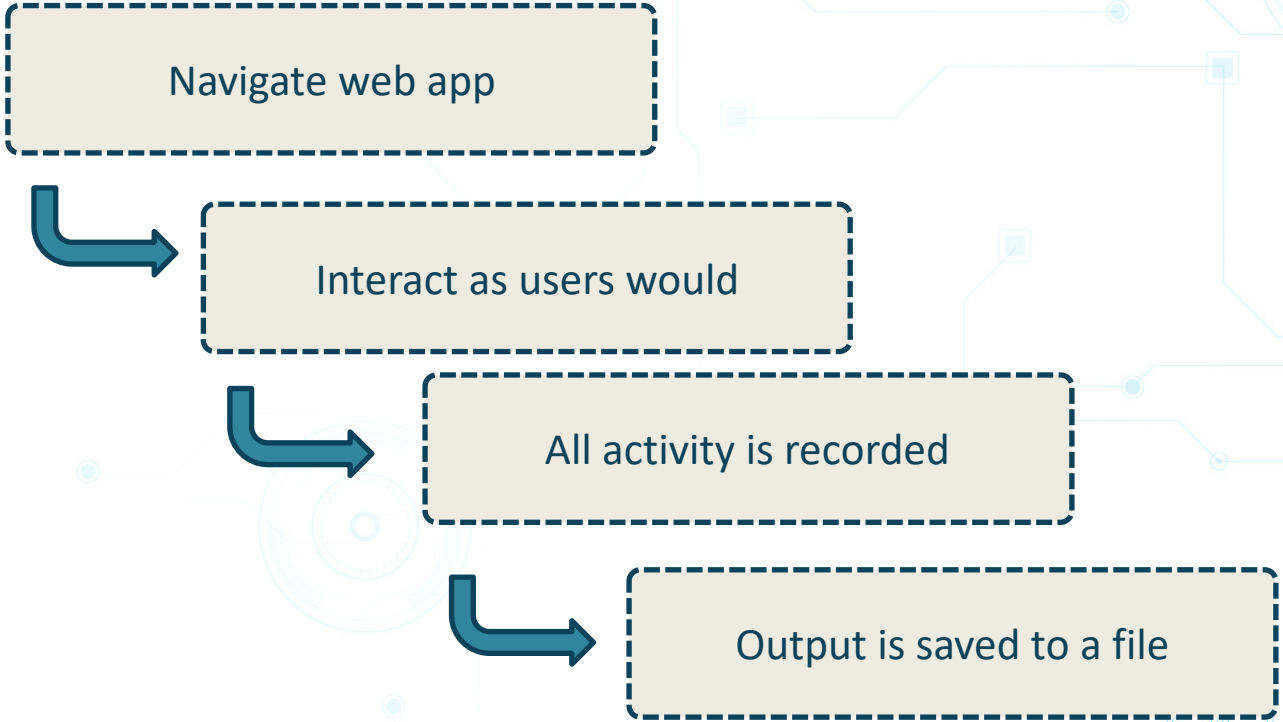
Acceptance Testing

- Does solution behavior align with design requirements?
- Are user needs addressed?
- Is the requirement implemented correctly?

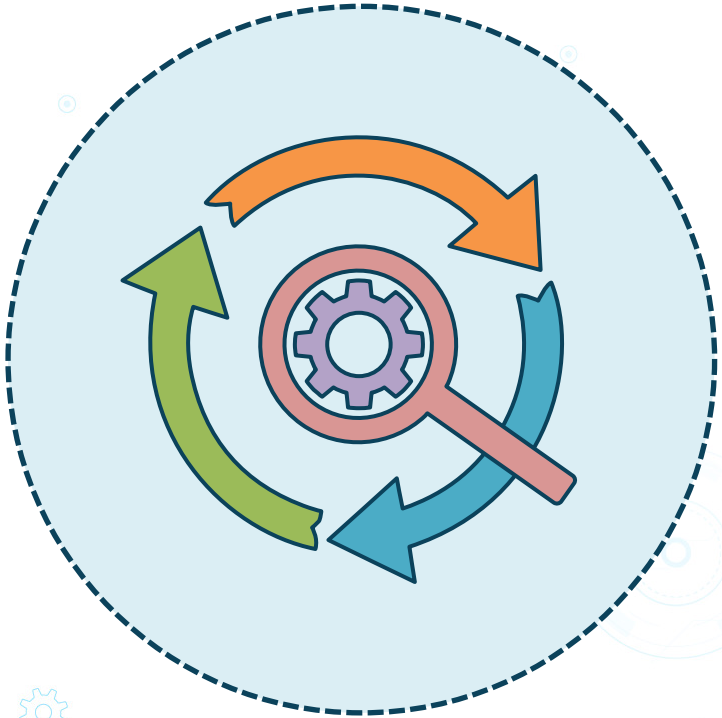
Acceptance Testing

- Legal and regulatory compliance
- Contractual compliance
- Reasonable performance
- SME testing

Automated Acceptance Testing Example

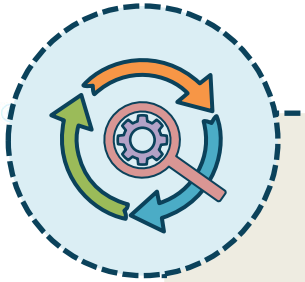


Integration Testing



- Do all components interact together properly?
- Much more encompassing than unit testing

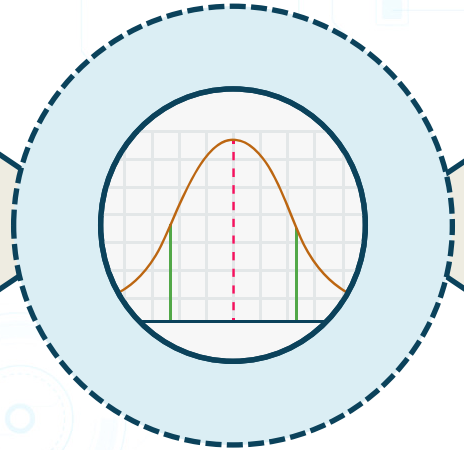
Integration Testing



- Gather documentation regarding unit interaction
- Performed after unit testing is completed
- Performed by developers are a testing team, not users
- Use automation

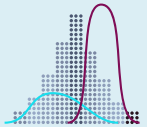
Performance Testing

Simulate realistic conditions



Identify baseline performance

Load Testing



- Maximum amount of data at a time
- Maximum number of concurrent users
- Performance during peak workloads
  - App and supporting infrastructure
  - Historical peak workload data
  - Users leave a site that responds too slowly

Stress Testing

- Tests solution reliability
- Load conditions far beyond normal activity
- What is the breaking point?
- Identifies how to
  - Configure high availability
  - Plan for alternate solutions when required

Fuzzing

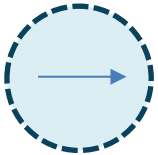
- App testing
- Unexpected data
- Observe app behavior

Fuzzing

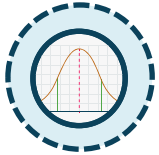
- Automated testing
- Fuzz data
  - Random
  - Invalid data inputs for app
  - Malformed packets

Fuzzing

Supplying incorrect data types



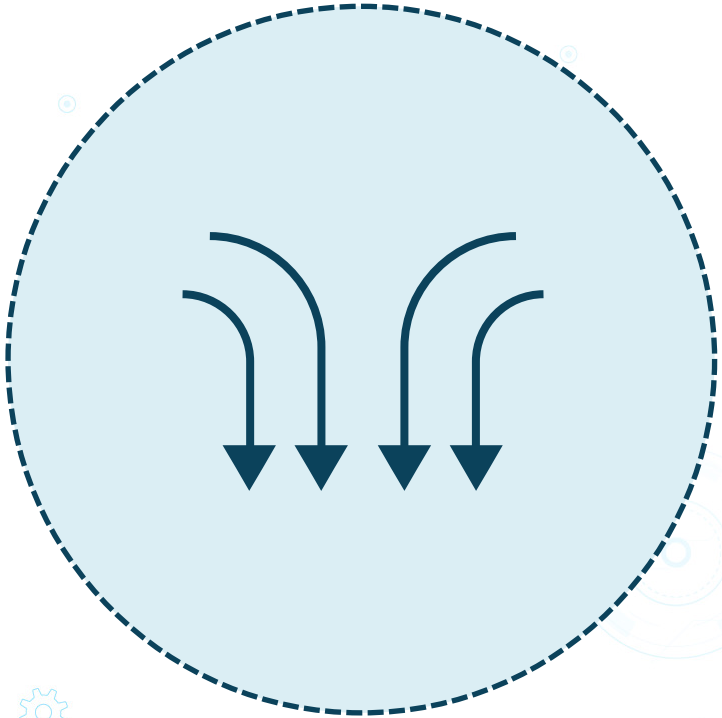
Writing beyond defined memory boundaries (buffer overflow)



Large volume of data in small amount of time

Malformed URLs

Continuous Integration and Delivery (CI/CD)



- This is a type of DevOps control
- Software changes
- Rapid deployment

Continuous Integration

- Developer code modifications are merged into the main code branch
  - Immediate
  - Synchronize copied repository
- Due to time constraints, testing automation is a must

Continuous Delivery

- Software changes *can be* made available quickly
  - Continuous deployment *does* make changes available
- After successful testing, automated delivery
- Automation can save time and money

System Migration and Data Conversion

- Migration
  - Movement of systems and data between dissimilar systems
- Conversion
  - Data transformation for consumption in the target system
- Common with cloud adoption and company mergers

System Migration

- Determine suitable cloud replacements
  - Cloud readiness tools
- Service Level Agreements (SLAs)
- Run systems in parallel

Data Conversion

- Data privacy laws and regulations
- Data masking
- Filter out unnecessary data
- Data validation
- File formats
- Data type transformation

In this exercise, you will

- List three common secure coding practices
- Explain the relationship between unit and regression testing
- Describe the purpose of fuzzing
- Provide examples of load testing

Secure Coding

- Peer review
- Input validation
- Limited use of third-party code

Unit and Regression Testing

- Unit
  - Test small chunks of code
- Regression
  - Test for unwanted negative effects of code changes

Fuzzing

- Observe app behaviour when fed
  - Random data
  - Invalid data inputs for app
  - Malformed packets

Load Testing

- Maximum amount of data at a time
- Maximum number of concurrent users
- Performance during peak workloads