A project report on

Internship In Société Générale-GSC

(*PAT Internship*)

Submitted in partial fulfillment for the award of the degree of

Bachelor Of Technology

in

Computer Science and Engineering

by

IPPILI AKARSH (18BCE2523)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
VELLORE - 632014

May,2022

Internship In Société Générale-GSC

(PAT Internship)

Submitted in partial fulfillment for the award of the degree of

Bachelor Of Technology

in

Computer Science and Engineering

by

IPPILI AKARSH (18BCE2523)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
VELLORE - 632014

May,2022

DECLARATION

I hereby declare that thesis entitled “Internship in Société Générale - GSC” submitted by me, for the award of the degree BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING, VIT is a record of bonafide work carried out by me under the supervision of Mr. Siva Vangavolu.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Rayagada

Date: 27th May,2022 Signature of the Candidate

Internship Completion Certificate

SOCIETE GENERALE

Date:27th May,2022

CERTIFICATE BY THE EXTERNAL GUIDE

This is to certify that the project report entitled "Internship in Société Générale" submitted by IPPILI AKARSH (18BCE2523) to Vellore Institute of Technology in partial fulfilment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a record of bonafide work carried out by him under my guidance. The project fulfils the requirements as per the regulations of this Institute and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

SIVA VANGAVOLU

Société Générale Global Solution Center Pvt.

Ltd Voyager Building,

International Tech Park

Limited, Bengaluru,

Karnataka 560103

ABSTRACT

There are a number of applications that a company is entitled to build. Some could be for the benefits of a customer, some for internal use itself. At times most development centers consist multiple cross functional engineering teams which are involved in the process of developing features and products for a suite of applications. There are many benchmarks and checks that are required on the application on everyday basis for ensuring that the best quality of service is delivered to the stakeholders of the applications continuously.

Trade finance has a key role to play globally to contribute to the transition to a more sustainable, more inclusive and greener economy. Société Générale have developed innovative sustainability linked trade finance solutions to contribute to our clients' positive impact strategy.

A trade, also called a deal, is an exchange of financial products from one entity to another. The life cycle of a trade is the fundamental activity of exchanges, investment banks, hedge funds, pension funds and many other financial companies.

All the steps involved in a trade, from the point of order placed and trade execution through to settlement of the trade, are commonly referred to as the trade life cycle. Trade life cycle consists of a series of logical stages and steps.

1. Pre-sale stage: Marketing persons from investment banks, brokers and dealers introduce various financial products and vehicles to clients. Investors or institutional fund managers' survey the market and find the most suitable and competitive products.

2. Trade execution: After trading negotiations between seller and buyer, an order is placed and the trade is executed. The completion of a buy or a sell order of a financial product is known as Trade Execution.

3. Trade Capture: After trade execution, the trade is booked in the Front Office system, Middle Office Risk Management system and Back Office system.

4. Trade Validation and Confirmation: Back office validates trade attributes and confirms trade settlement. Risk Management checks the valuation, risks

and limits.

5. Trade Settlement: Any fee or premium needs to be settled. For a periodic cash settlement trade, such as interest rate swaps or bonds, there is a process of simultaneous exchange of cash between parties at each payment date.

6. Trade Termination: A trade may be expired at maturity or terminated early. The early termination could be caused by a position sell or triggered by an early termination provision, such as auto call/cancel, knock-out, etc

During trade life cycle trader makes decisions based on various factors. And the Pact application helps traders with various insights relate to their PnL status. So that they can make good informed decisions in the future. It helps to calculate NBL and to assign to specific groups. Pass the NBI report and other PnL data to downstream applications.

ACKNOWLEDGEMENT

I’m thankful to Mr. Siva Vangavolu for giving me the right environment and showing confidence in me to execute my skills. I would like to thank my team members under whom I received the opportunity to learn new things, for their immense support and guidance throughout. I would like to thank my whole tech tribe who has shared their valuable knowledge and helped me in transitioning from a student life to a professional one.

In jubilant mood I express ingeniously my whole-hearted thanks to our Chancellor Dr. Viswanathan, Dr. RAMESH BABU K (Dean-SCOPE), Dr. VAIRAMUTHU S (HOD- SCOPE), Dr. V. Samuel (Director-Placement and Training) and entire management of VIT all teaching staff and members working as limbs of our university for their enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my internship successfully. I would like to thank my parents and friends for their support

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last, but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Rayagada

Date: 27.05.2022 **Ippili Akarsh**

TABLE OF CONTENTS

LIST OF ACRONYMS

ABBREVATION	FULL FORM
API	Application Programming Interface
REST	Representational State Transfer
DB	Database
SOAP	Simple Object Access Protocol
CD	Continuous Development
DEV	Development
JS	Java Script
HTML	Hyper Text Markup Language
CSS	Cascading Style Sheets
MVC	Model View Controller
DOM	Document Object Model
JDBC	Java Data Base Connectors
XML	Extensible Markup Language
JWT	JSON Web Tokens
JQL	JIRA Query Language
CI	Continuous Integration
NBI	NET Banking Income

CHAPTER 1
INTRODUCTOIN

1.1 INTRODUCTION

The project revolves around Pact decommission. Pact is a middle office application it helps traders about their PnL positions. It achieves it by providing software solutions developed for easy, user friendly and efficient way to calculate NBI and assign to different people / groups.

1.2 PROJECT STATEMENT

The expectation of the project is to deliver a solution for providing the traders with valuable insights about their PnL positions, calculate NBI and allocate it to various groups/people, report NBI and other PNL data into downstream applications.

1.3 OVERVIEW OF THE PROJECT

The project on the whole is pact decommission. It deals with providing the traders insights to their PnL position so that they can make informed decisions for future trading.

Traders book trades in Fidessa. Fidessa will send details to Pact. Pact is a middle office application while helps to calculate NBI and allocate it various groups.

1.4 OBJECTIVES

This project helps traders providing valuable insights related to their PnL status. So that traders can make informed decisions for feature trading. It generates the report in a graphically appealing format on dashboard/UI so that it is well perceived by the users.

CHAPTER 2
BACKGROUND

2.1 INTRODUCTION

Société Générale-GSC is a fast and growing information technology and services company. This line of work involves a lot more than technical skills and business acumen. With the rise in banking infrastructure, the company now follows an agile setup across the teams. The agile needs more evermore IT solutions to manage.

Project monitoring plays a vital part in project management as well as it facilitates fast decision-making processes. However, it is a method often overlooked and only done for the sake of fulfilling the requirements of a project management plan. If put into practice, project monitoring can help project managers and their teams foresee potential risks and obstacles that if left unaddressed, could derail the project. It keeps a check on the project and its proper functioning, reports the progress to the teams and keeps the management aware of the problems which crop up during the implementation of the project. It supports the teams to be more agile and help address the issues faster, is some cases even before they arise.

2.2 MOTIVATION

Société Générale Global Solution Centre is an innovation HUB for Société Générale Group. The company walk the path of innovation. From its inception in 1864, Société Générale has embraced innovation. The company continue this spirit at Société Générale Global Solution Centre with a conscious focus on innovation. With those values at heart, the teams are motivated to innovate and build solutions which are driven towards agility and deliver value.

CHAPTER 3

DESIGN APPROACH AND DETAILS

3.1 INTRODUCTION

In this chapter we shall discuss the design approach in greater detail describing the design patterns used with focus on the code standard and the other components.

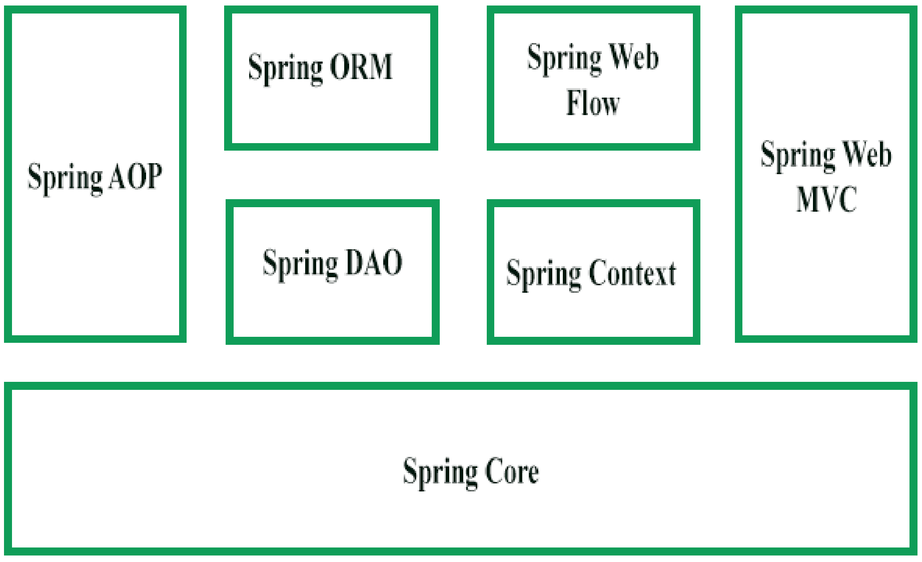
3.2 DESIGN APPROACH AND DETAILS

The very basis of the design comes from leveraging the already existing and widely used libraries and technical stack used across the teams so as to deliver the project with minimal setup requirement and to support the same on existing infrastructure. The project should be easily maintainable and scalable to accommodate other application as well.

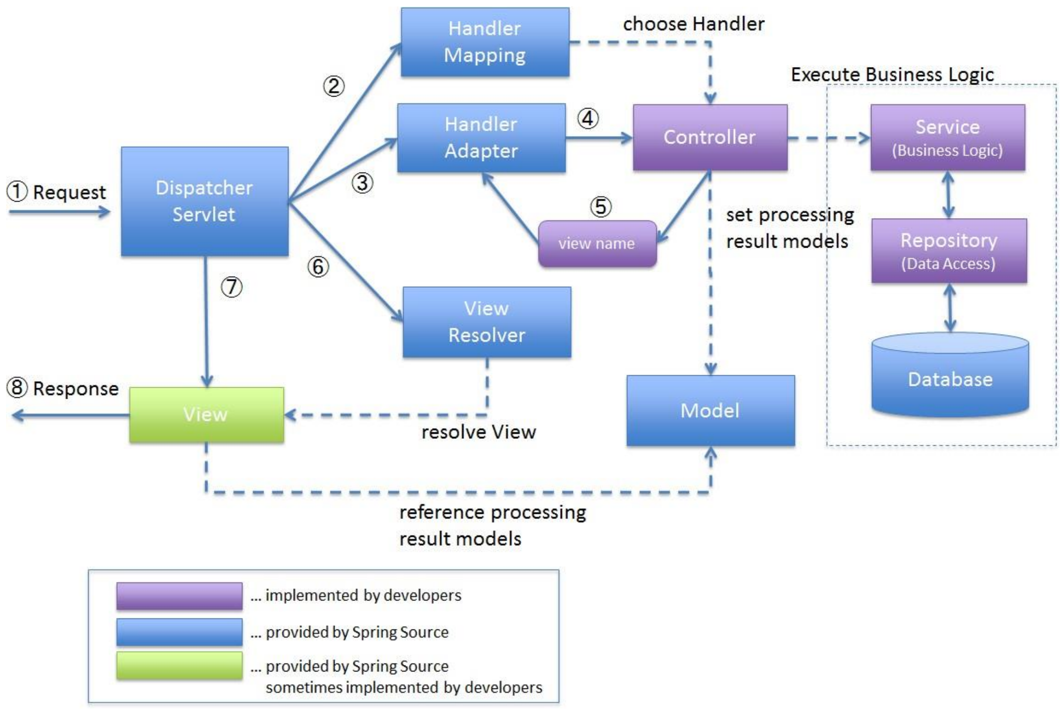
Thus, the project is built on using Spring framework written in JAVA language and build using a project management tool Maven leveraging the rich APIs of JAVA language such as JDBC, JSON and other RESTFUL APIs for the backend part.

The frontend is built using Angular and React using TypeScript for developing single page applications for different use cases.

3.3 SPRING FRAMEWORK

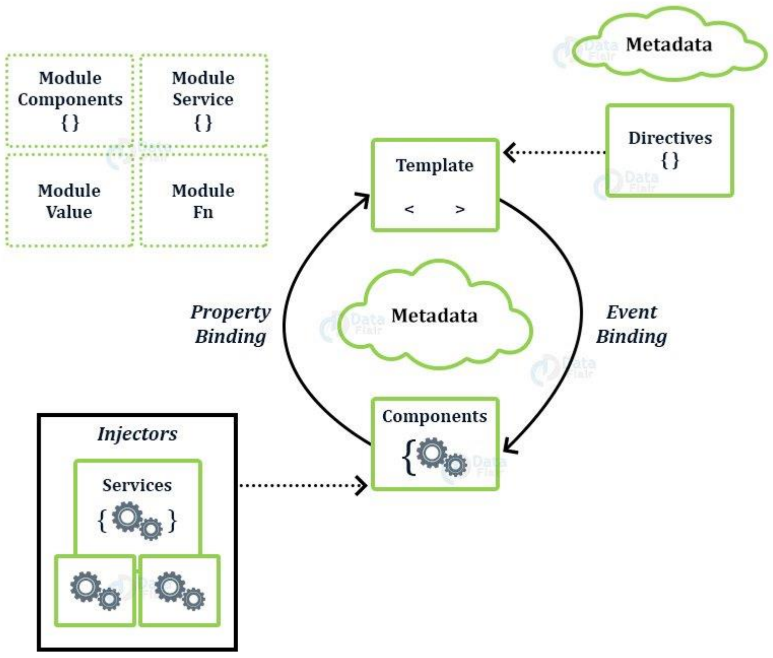


Spring Framework



Request Response Lifecycle

3.4 ANGULAR FRAMEWORK



Building blocks of angular architecture as depicted in the image.

1. Module: An Angular app has a root module, named AppModule, which provides the bootstrap mechanism to launch the application.
2. Components: Each component in the application defines a class that holds the application logic and data. A component generally defines a part of the user interface (UI).
3. Templates: The Angular template combines the Angular markup with HTML to modify HTML elements before they are displayed. There are two types of data binding:
    a. Event binding: Lets your app respond to user input in the target environment by updating your application data.
    b. Property binding: Enables users to interpolate values that are computed from your application data into the HTML.
4. Metadata: Metadata tells Angular how to process a class. It is used to decorate the class so that it can configure the expected behavior of a class.
5. Services: When you have data or logic that isn't associated with the view but has to be shared across components, a service class is created. The class is always associated with the @Injectible decorator.
6. Dependency Injection: This feature lets you keep your component classes crisp and efficient. It does not fetch data from a server, validate the user

input, or log directly to the console. Instead, it delegates such tasks to the services.

3.5 CODES AND STANDARDS

The coding standards involves understanding how to produce clean, maintainable and evolutive code and it’s important for the overall quality of the platform:

- CODE STANDARDS
    - ➢ Clean Code
    - ➢ Test Driven Development (TDD)
    - ➢ Effective JAVA standards:
        - ▪ SOLID principles
        - ▪ FIRST principles for tests
        - ▪ JAVA Unit Test
- REST API STANDARDS

The REST PI standards include the following:

- ➢ Define consistent practices and patterns for all API endpoints across the platform
- ➢ Adhere as closely as possible to company’s API Guidelines Standards List
- ➢ URL structure
- ➢ HTTP methods
- ➢ HTTP Status codes
- ➢ Standard request headers
- ➢ Standard response headers

➢ Response formats

➢ Hierarchical REST structures

➢ Mandatory Endpoints: API health

➢ Tags and Metadata

• CI/CD STANDARDS

Continuous Integration, Continuous Delivery and Continuous Deployment (CI/CD) ensures the principals, patterns to follow and anti-patterns to avoid.

➢ Continuous Integration (CI): The responsibility of CI is to take the code in SCM system and to build artifacts in a reproducible, auditable and automated way

➢ Continuous Delivery & Deployment (CD): The responsibility of CD is to ensure that the artifact has been tested and is registered in the target environment. After that, the service is in the RUN process.

The Components CI/CD include are:

➢ Source Code Management: GIT

➢ Static Code Analysis: SonarLint

➢ CI Server: Jenkins

CHAPTER 4

TECHNICAL SECIFICATIONS

4.1 INTRODUCTION

In this chapter we shall discuss the technical specifications in greater detail describing the technical stack used with focus on each component.

4.2 TECHNICAL STACK

The technical stack includes the following components across the projects:

(A) ANGULAR JS

AngularJS is a JavaScript-based open-source front-end web framework mainly maintained by Google and by a community of individuals and corporations to address many of the challenges encountered in developing single-page applications. It aims to simplify both the development and the testing of such applications by providing a framework for client-side model–view–controller (MVC) and model–view– view model (MVVM) architectures, along with components commonly used in rich Internet applications.

AngularJS is the frontend part of the MEAN stack, consisting of MongoDB database, Express.js web application server framework, Angular.js itself, and Node.js server runtime environment.

The AngularJS framework works by first reading the Hypertext Markup Language (HTML) page, which has an additional custom HTML attribute embedded into it. Angular interprets those attributes as directives to bind input or output parts of the page to a model that is represented by standard JavaScript variables. The values of those JavaScript variables can be manually set within the code, or retrieved from static or dynamic JSON resources.

AngularJS is built on the belief that declarative programming should be used to create user interfaces and connect software components, while imperative programming is better suited to defining an application's business logic. The framework adapts and extends traditional HTML to present dynamic content through two-way data-binding that allows for the automatic synchronization of models and views. As a result, AngularJS de-emphasizes explicit Document Object Model (DOM) manipulation with the goal of improving testability and performance.

AngularJS's design goals include:

- to decouple DOM manipulation from application logic. The difficulty of this is dramatically affected by the way the code is structured.
- to decouple the client side of an application from the server-side. This allows development work to progress in parallel and allows for reuse of both sides.
- to provide structure for the journey of building an application: from designing the UI, through writing the business logic, to testing.

AngularJS implements the MVC pattern to separate presentation, data, and logic components. Using dependency injection, Angular brings traditionally server- side services, such as view-dependent controllers, to client-side web applications.
Consequently, much of the burden on the server can be reduced.

The task performed by the AngularJS bootstrapper occur in three phases after the DOM has been loaded:

1. Creation of a new Injector
2. Compilation of the directives that decorate the DOM
3. Linking of all directives to scope

Why AngularJS?

HTML is great for declaring static documents, but it falters when we try to use it for declaring dynamic views in web-applications. AngularJS lets you extend HTML vocabulary for your application. The resulting environment is extraordinarily expressive, readable, and quick to develop.

Data Binding

Data-binding is an automatic way of updating the view whenever the model changes, as well as updating the model whenever the view changes. This is awesome because it eliminates DOM manipulation from the list of things you have to worry about.

Controller

Controllers are the behavior behind the DOM elements. AngularJS lets you express the behavior in a clean readable form without the usual boilerplate of updating the DOM, registering callbacks or watching model changes.

Directives

Directives are a unique and powerful feature available in AngularJS. Directives let you invent new HTML syntax, specific to your application.

Reusable Components

We use directives to create reusable components. A component allows you to hide complex DOM structure, CSS, and behavior. This lets you focus either on what the application does or how the application looks separately.

Localization

An important part of serious apps is localization. AngularJS's locale aware filters and stemming directives give you building blocks to make your application available in all locales.

Deep Linking

A deep link reflects where the user is in the app. This is useful so users can bookmark and email links to locations within the app. Round trip apps get this automatically, but AJAX apps by their nature do not. AngularJS combines the benefits of deep linking with desktop app-like behavior.

Form Validation

Client-side form validation is an important part of a great user experience. AngularJS lets you declare the validation rules of the form without having to write JavaScript code. Write less code, go have beer sooner.

Server Communication

AngularJS provides built-in services on top of XHR as well as various other backends using third party libraries. Promises further simplify your code by handling asynchronous return of data.

Injectable

The dependency injection in AngularJS allows you to declaratively describe how your application is wired. This means that your application needs no main() method which is usually an unmaintainable mess. Dependency injection is also a core to AngularJS. This means that any component which does not fit your needs can easily be replaced.

Testable

AngularJS was designed from ground up to be testable. It encourages behavior-view separation, comes pre-bundled with mocks, and takes full advantage of dependency injection. It also comes with end-to-end scenario runner which eliminates test flakiness by understanding the inner workings of AngularJS.

Two-way data binding

AngularJS two-way data binding is its most notable feature, largely relieving the server backend of templating responsibilities. Instead, templates are rendered in plain HTML according to data contained in a scope defined in the model. The $scope service in Angular detects changes to the model section and modifies HTML expressions in the view via a controller. Likewise, any alterations to the view are reflected in the model. This circumvents the need to actively manipulate the DOM

and encourages bootstrapping and rapid prototyping of web applications.AngularJS detects changes in models by comparing the current values with values stored earlier in a process of dirty-checking, unlike Ember.js and Backbone.js that trigger listeners when the model values are changed.

$watch

Is an angular method used for dirty checking. Any variable or expression assigned in

$scope automatically sets up a $watchExpression in angular. So assigning a variable to $scope or using directives like ng-if, ng-show, ng-repeat etc. all create watches in angular scope automatically. Angular maintains a simple array of $$watchers in the
$scope objects
Different ways of defining a watcher in AngularJS.

- explicitly $watch an attribute on $scope.
    - $scope.$watch('person.username', validateUnique);
- place an interpolation in your template (a watcher will be created for you on the current $scope).
- ask a directive such as ng-model to define the watcher for you.

```
<input ng-model="person.username" />
```

$digest

is angular method that is invoked internally by angularjs in frequent intervals. In $digest method, angular iterates overall $watches in its scope/child scopes.

$apply

is an angular method that internally invokes $digest. This method is used when you want to tell angular manually start dirty checking (execute all $watches)

$destroy

is both a method and event in angularjs. $destroy() method, removes a scope and all its children from dirty checking. $destroy event is called by angular whenever a $scope or $controller is destroyed.

(B) React

React is a JavaScript library for building user interfaces. React creates a VIRTUAL DOM in memory. Instead of manipulating the browser's DOM directly, react creates a virtual DOM in memory, where it does all the necessary manipulating, before making the changes in the browser DOM. React only changes what needs to be changed.

React adheres to the declarative programming paradigm. Developers design views for each state of an application, and React updates and renders components when data changes. In computer science, declarative programming is a programming paradigm a style of building the structure and elements of computer programs that expresses the logic of a computation without describing its control flow.

Components

React code is made of entities called components. These components are reusable and must be formed in the SRC folder following the Pascal Case as its naming convention (capitalize camelCase). Components can be rendered to a particular element in the DOM using the React DOM library. When rendering a component, one can pass the values between components through "props".

The two primary ways of declaring components in React are through function components and class-based components.

1. Function components are declared with a function that then returns some JSX.

```
const Greeter = () => <div>Hello World</div>;
```

2. Class-based components are declared using ES6 classes.

```
class ParentComponent extends React.Component {
  state = {color: 'green'};
  render () {
    return (
      <ChildComponent color={this.state.color} />
    );
  }
}
```

Where class components are all about the use of classes and the lifecycle methods, functional components have hooks to deal with state management and other problems which arise when writing code in React.

Virtual DOM

Another notable feature is the use of a virtual Document Object Model, or virtual DOM. React creates an in-memory data-structure cache, computes the resulting differences, and then updates the browser's displayed DOM efficiently. This process is called reconciliation. This allows the programmer to write code as if the entire page is rendered on each change, while the React libraries only render subcomponents that actually change. This selective rendering provides a major performance boost. It saves the effort of recalculating the CSS style, layout for the page and rendering for the entire page.

Lifecycle methods

Lifecycle methods for class-based components use a form of hooking that allows the execution of code at set points during a component's lifetime.

- shouldComponentUpdate allows the developer to prevent unnecessary re-rendering of a component by returning false if a render is not required.
- componentDidMount is called once the component has "mounted" (the component has been created in the user interface, often by associating it with a DOM node). This is commonly used to trigger data loading from a remote source via an API.
- componentWillUnmount is called immediately before the component is torn down or "unmounted". This is commonly used to clear resource-demanding dependencies to the component that will not simply be removed with the unmounting of the component (e.g., removing any setInterval() instances that are related to the component, or an "eventListener" set on the "document"

because of the presence of the component)

- render is the most important lifecycle method and the only required one in any component. It is usually called every time the component's state is updated, which should be reflected in the user interface.

React hooks

Hooks are functions that let developers "hook into" React state and lifecycle features from function components. Hooks do not work inside classes — they let you use React without classes.

React provides a few built-in hooks useState, useContext, useReducer, useMemo and useEffect.

Others are documented in the Hooks API Reference. useState and useEffect, which are the most commonly used, are for controlling state and side effects respectively.

Rules of hooks:

There are rules of hooks which describe the characteristic code pattern that hooks rely on. It is the modern way to handle state with React.

1. Hooks should only be called at the top level (not inside loops or if statements).
2. Hooks should only be called from React function components and custom hooks, not normal functions or class components.

Although these rules can't be enforced at runtime, code analysis tools such as linters can be configured to detect many mistakes during development. The rules apply to both usage of hooks and the implementation of custom hooks, which may call other hooks.

(C) Oracle SQL

Flexible Architecture

- Open Source
- Multi-threaded
- Pluggable Storage-Engine
- InnoDB, NDB
- MyISAM

ANSI SQL Standards

- ANSI SQL
- SubQueries, Joins, Cursors
- Prepared Statements
- Views
- Triggers
- Stored Procedures
- User-Defined Functions
- Window Functions and CTEs
- NOWAIT and SKIP LOCK
- Descending Indexes
- Invisible Indexes
- Grouping

Optimizer

- Cost-based Optimizer
- Optimizer Tracing
- JSON Explain
- Optimizer Hints

- Optimizer Histograms

SQL Document Store

- Relational Tables
- JSON Documents
- X Protocol
- X DevAPI
- MySQL Shell

JSON Support

- Native JSON Datatype
- JSON Table Functions
- JSON Aggregation Functions
- JSON Merge Functions
- JSON Partial Update

Replication & High-Availability

- InnoDB Cluster
- Group Replication
- Router
- Built-in Replication Engine
- Master/Slave, Ring, Tree
- Row-based Replication
- Semi-synchronous Replication
- Multi-source Replication
- Time-delayed Replication
- Global Transaction IDs

- Slave Failover, Recovery
- Multi-threaded slaves
- Sharding

MySQL NDB Cluster

- 99.999% Availability
- Distributed architecture
- Synchronous replication
- Real-time transactional performance
- Foreign Keys
- SQL & Non-SQL data access
- Auto sharding of data
- Java, C++, memcached, HTTP

Security

- OpenSSL by Default
- SQL Roles
- Password management

High-Performance

- Performance Schema
- Information Schema
- SYS Schema
- Resource Groups
- Partitioning
- Optimized for high concurrency
- Optimized for Read Only

- Optimized for use with SSD
- Multiple Index Type (B-tree, R-tree, Hash, Full Text, etc.)
- Server-side Thread Pool
- Connection Thread Caching
- Diagnostics, and SQL Tracing

OLTP and Transactions

- ACID Transactions
- Commit, Rollback
- Foreign Keys
- Referential Integrity
- Row-level Locking
- Customizable Lock Isolation Levels
- Distributed Transactions (XA)
- Snapshot Isolation
- Repeatable Reads (readers don't block writers and vice-versa)
- Automatic Deadlock Detection

Data Warehouse Optimized features

- Fast Data Load Utility
- High-Speed Multi-Insert Function
- GROUP BY WITH ROLLUP
- Aggregate UDF
- Analytic SQL Functions
- Multi-Terabyte Scalability

Geo Spatial Support

- InnoDB R-tree Spatial Indexes

- GeoHash
- GeoJSON
- Spatial Reference System (SRS)
- SRID Spatial Data Types
- SRID Spatial Indexes
- SRID Spatial Functions

Character Sets & Collations

- UTF8MB4 as default
- Unicode 9.0
- Case & Accent Sensitive Collations

Manageability and Ease of Use

- Easy Install and Setup
- "3 minutes to Success" with all-in-one Windows Installer

(D) Spring Framework

The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform.

A key element of Spring is infrastructural support at the application level: Spring focuses on the "plumbing" of enterprise applications so that teams can focus on application-level business logic, without unnecessary ties to specific deployment environments.

Features

- **Core technologies**: dependency injection, events, resources, i18n, validation, data binding, type conversion, SpEL, AOP.

  A thorough treatment of the Spring Framework's IoC container is closely followed by comprehensive coverage of Spring's Aspect-Oriented Programming (AOP) technologies. The Spring Framework has its own AOP framework, which is conceptually easy to understand and which successfully addresses the 80% sweet spot of AOP requirements in Java enterprise programming.

  IoC is also known as dependency injection (DI). It is a process whereby objects define their dependencies (that is, the other objects they work with) only through constructor arguments, arguments to a factory method, or properties that are set on the object instance after it is constructed or returned from a factory method. The container then injects those dependencies when it creates the bean. This process is fundamentally the inverse (hence the name, Inversion of Control) of the bean itself controlling the instantiation or location of its dependencies by using direct construction of classes or a mechanism such as the Service Locator pattern.

  The org.springframework.beans and org.springframework.context package s are the basis for Spring Framework's IoC container. The BeanFactory interface provides an advanced configuration mechanism capable of managing any type of object.

ApplicationContext is a sub-interface of BeanFactory. It adds:

- Easier integration with Spring's AOP features
- Message resource handling (for use in internationalization)
- Event publication
- Application-layer specific contexts such as the WebApplicationContext for use in web applications.

In short, the BeanFactory provides the configuration framework and basic functionality, and the ApplicationContext adds more enterprise-specific functionality. The ApplicationContext is a complete superset of the BeanFactory and is used exclusively in this chapter in descriptions of Spring's IoC container.

In Spring, the objects that form the backbone of your application and that are managed by the Spring IoC container are called beans. A bean is an object that is instantiated, assembled, and otherwise managed by a Spring IoC container. Otherwise, a bean is simply one of many objects in your application. Beans,

and the dependencies among them, are reflected in the configuration metadata used by a container.

- **Testing**: mock objects, TestContext framework, Spring MVC Test, WebTestClient.

UNIT TESTING

Dependency injection should make your code less dependent on the container than it would be with traditional Java EE development. The POJOs that make up your application should be testable in JUnit or TestNG tests, with objects instantiated by using the new operator, without Spring or any other container. You can use mock objects (in conjunction with other valuable testing techniques) to test your code in isolation. If you follow the architecture recommendations for Spring, the resulting clean layering and componentization of your codebase facilitate easier unit testing. For example, you can test service layer objects by stubbing or mocking DAO or repository interfaces, without needing to access persistent data while running unit tests.

True unit tests typically run extremely quickly, as there is no runtime infrastructure to set up. Emphasizing true unit tests as part of your development methodology can boost your productivity.

- **Data Access**: transactions, DAO support, JDBC, ORM, Marshalling XML.

  Transaction Management

  Comprehensive transaction support is among the most compelling reasons to use the Spring Framework. The Spring Framework provides a consistent abstraction for transaction management that delivers the following benefits:

  - A consistent programming model across different transaction APIs, such as Java Transaction API (JTA), JDBC, Hibernate, and the Java Persistence API (JPA).
  - Support for declarative transaction management.
  - A simpler API for programmatic transaction management than complex transaction APIs, such as JTA.
  - Excellent integration with Spring's data access abstractions.

    The following sections describe the Spring Framework's transaction features and technologies:

  - Advantages of the Spring Framework's transaction support model describes why you would use the Spring Framework's transaction abstraction instead of EJB Container-Managed

    Transactions (CMT) or choosing to drive local transactions through a proprietary API, such as Hibernate.

    - Understanding the Spring Framework transaction abstraction outlines the core classes and describes how to configure and obtain DataSource instances from a variety of sources.
    - Synchronizing resources with transactions describes how the application code ensures that resources are created, reused, and cleaned up properly.
    - Declarative transaction management describes support for declarative transaction management.
    - Programmatic transaction management covers support for programmatic (that is, explicitly coded) transaction management.
    - Transaction bound event describes how you could use application events within a transaction.

- **Spring MVC** and **Spring WebFlux** web frameworks.
- **Integration**: remoting, JMS, JCA, JMX, email, tasks, scheduling, cache.

Aspect-oriented programming framework

The Spring Framework has its own Aspect-oriented programming (AOP) framework that modularizes cross-cutting concerns in aspects. The motivation for creating a separate AOP framework comes from the belief that it should be possible to provide basic AOP features without too much complexity in either design, implementation, or configuration. The Spring AOP framework also takes full advantage of the Spring container.

The Spring AOP framework is proxy pattern-based, and is configured at run time. This removes the need for a compilation step or load-time weaving. On the other hand, interception only allows for public method-execution on existing objects at a join point.

Compared to the AspectJ framework, Spring AOP is less powerful, but
also less complicated. Spring 1.2 includes support to configure AspectJ
aspects in the container. Spring 2.0 added more integration with AspectJ;
for example,
the pointcut language is reused and can be mixed with Spring AOP-based
aspects. Further, Spring 2.0 added a Spring Aspects library that uses AspectJ to
offer common Spring features such as declarative transaction management and
dependency injection via AspectJ compile-time or load-time weaving.
SpringSource also uses AspectJ AOP in other Spring projects such as Spring
Roo and Spring Insight, with Spring Security also offering an AspectJ-based
aspect library.

Spring AOP has been designed to make it able to work with cross-cutting
concerns inside the Spring Framework. Any object which is created and
configured by the container can be enriched using Spring AOP.

The Spring Framework uses Spring AOP internally for transaction
management, security, remote access, and JMX.

Since version 2.0 of the framework, Spring provides two approaches to
the AOP configuration:

- schema-based approach[16] and
- @AspectJ-based annotation style.[17]

```
<beans xmlns="http://www.springframework.org/schema/beans"

  xmlns:mvc="http://www.springframework.org/schema/mvc"

  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

  xmlns:aop="http://www.springframework.org/schema/aop"

  xmlns:context="http://www.springframework.org/schema/cont

  ext"

  xsi:schemaLocation="http://www.springframework.org/schem

  a/beans
```

```
http://www.springframework.org/schema/beans/spring-
beans.xsd http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-
context.xsd http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-
mvc.xsd http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-
aop.xsd">
```

The Spring team decided not to introduce new AOP-related terminology; therefore, in the Spring reference documentation and API, terms such as aspect, join point, advice, pointcut, introduction, target object (advised object), AOP proxy, and weaving all have the same meanings as in most other AOP frameworks (particularly AspectJ).

Data access framework

Spring's data access framework addresses common difficulties developers face when working with databases in applications. Support is provided for all popular data access frameworks in Java: JDBC, iBatis/MyBatis, Hibernate, Java Data
Objects (JDO), Java Persistence API (JPA), Oracle TopLink,
Apache OJB, and Apache Cayenne, among others.

For all of these supported frameworks, Spring provides these features

- Resource management - automatically acquiring and releasing database resources
- Exception handling - translating data access related exception to a Spring data access hierarchy
- Transaction participation - transparent participation in ongoing transactions
- Resource unwrapping - retrieving database objects from

connection pool wrappers

- Abstraction for binary large object (BLOB) and character large object (CLOB) handling

All these features become available when using template classes provided by Spring for each supported framework. Critics have said these template classes are intrusive and offer no advantage over using (for example) the Hibernate API directly. In response, the Spring developers have made it possible to use the Hibernate and JPA APIs directly. This however requires transparent transaction management, as application code no longer assumes the responsibility to obtain and close database resources, and does not support exception translation.

Together with Spring's transaction management, its data access framework offers a flexible abstraction for working with data access frameworks. The Spring Framework doesn't offer a common data access API; instead, the full power of the supported APIs is kept intact. The Spring Framework is the only framework available in Java that offers managed data access environments outside of an application server or container.

While using Spring for transaction management with Hibernate, the following beans may have to be configured:

- A DataSource

  like com.mchange.v2.c3p0.ComboPooledDataSource or org.apache.commons. dbcp.BasicDataSource

- A SessionFactory
  like org.springframework.orm.hibernate3.LocalSessionFactoryBean with a DataSource attribute

- A HibernateProperties
  like org.springframework.beans.factory.config.PropertiesFactoryBean

- A TransactionManager
  like
  org.springframework.orm.hibernate3.HibernateTransactionManager

with a SessionFactory attribute

Remote access framework

Spring's Remote Access framework is an abstraction for working with various RPC (remote procedure call)-based technologies available on the Java platform both for client connectivity and marshalling objects on servers. The most important feature offered by this framework is to ease configuration and usage of these technologies as much as possible by combining inversion of control and AOP.

The framework also provides fault-recovery (automatic reconnection after connection failure) and some optimizations for client-side use of EJB remote stateless session beans.

Spring provides support for these protocols and products out of the box

- HTTP-based protocols
    - Hessian: binary serialization protocol, open-sourced and maintained by CORBA-based protocols
    - RMI (1): method invocations using RMI infrastructure yet specific to Spring
    - RMI (2): method invocations using RMI interfaces complying with regular RMI usage
    - RMI-IIOP (CORBA): method invocations using RMI-IIOP/CORBA
- Enterprise JavaBean client integration
    - Local EJB stateless session bean connectivity: connecting to local stateless session beans
    - Remote EJB stateless session bean connectivity: connecting to remote stateless session beans
- SOAP

    - Integration with the Apache Axis Web services framework

Apache CXF provides integration with the Spring Framework for
RPC-style exporting of objects on the server side.

Both client and server setup for all RPC-style protocols and products supported by the Spring Remote access framework (except for the Apache Axis support) is configured in the Spring Core container.

There is alternative open-source implementation (Cluster4Spring) of a remoting subsystem included into Spring Framework that is intended to support various schemes of remoting (1-1, 1-many, dynamic services discovering) …

Spring Boot

Spring Boot is Spring's convention-over-configuration solution for creating stand-alone, production-grade Spring-based Applications that you can "just run".[23] It is preconfigured with the Spring team's "opinionated view" of the best configuration and use of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need very little Spring configuration. Features:

- Create stand-alone Spring applications
- Embed Tomcat or Jetty directly (no need to deploy WAR files)
- Provide opinionated 'starter' Project Object Models (POMs) to simplify your Maven configuration
- Automatically configure Spring whenever possible
- Provide production-ready features such as metrics, health checks and externalized configuration
- Absolutely no code generation and no requirement for XML configuration.

(E) JDBC

Java Database Connectivity (JDBC) is an application program interface (API) packaged with the Java SE edition that makes it possible to standardize and simplify the process of connecting Java applications to external, relational database management systems (RDBMS). Fundamentally, applications written in Java perform logic. The Java language provides facilities for performing iterative logic with looks, conditional logic with if statements and object-oriented analysis through the use of classes and interfaces. But Java applications do not store data persistently. Data persistence is typically delegated to NoSQL databases such as MongoDB and Cassandra, or to relational databases such as IBM's DB2 or Microsoft's SQL Server or the popular open source database MySQL.

JDBC interfaces, classes and components

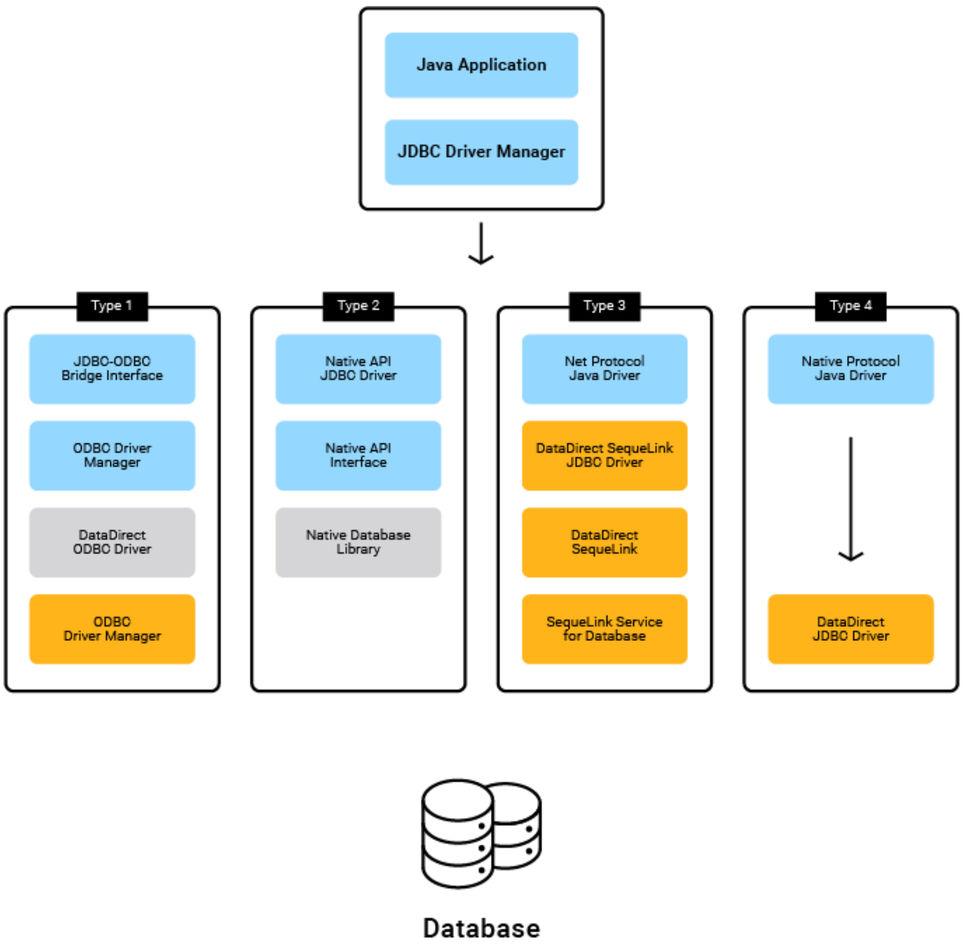
The JDBC API is composed of a number of interfaces and classes that represent a connection to the database, provide facilities for sending SQL queries to a database and help Java developer process the results of relational database interactions. JDBC

and SQL compared Structured Query Language (SQL) is an ISO specification that defines how applications can query, update and just generally interact with a relational database. The JDBC API does not perform any functions that could otherwise be performed through a SQL query. The goal of the JDBC API is to provide a connection to a relational database through with SQL queries can be performed, and the results from those queries can be processed within a Java program. JDBC is a connectivity API. SQL remains the language used to actually talk to the database. The basic steps to connect to a JDBC database are:

Load the right JDBC driver

- Obtain the database URL
- Use the JDBC DriverManger to connect to the database
- Create a SQL based Statement or PreparedStatement object
- Execute the statement against the database

- Process the results and handle any SQL exceptions



(F) MAVEN

Maven is a tool that can now be used for building and managing any Java-based project. We hope that we have created something that will make the day-to-day work of Java developers easier and generally help with the comprehension of any Java-based project.

Maven's primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period of time. In order to attain this goal, Maven deals with several areas of concern:

- Making the build process easy
- Providing a uniform build system
- Providing quality project information
- Encouraging better development practices
- Making the build process easy

While using Maven doesn't eliminate the need to know about the underlying mechanisms, Maven does shield developers from many details.

Maven builds a project using its project object model (POM) and a set of plugins. Once you familiarize yourself with one Maven project, you know how all Maven projects build. This saves time when navigating many projects.

Maven provides useful project information that is in part taken from your POM and in part generated from your project's sources. For example, Maven can provide:

- Change log created directly from source control
- Cross referenced sources
- Mailing lists managed by the project
- Dependencies used by the project

- Unit test reports including coverage

Third party code analysis products also provide Maven plugins that add their reports to the standard information given by Maven.

Maven aims to gather current principles for best practices development

and make it easy to guide a project in that direction.

For example, specification, execution, and reporting of unit tests are part

of the normal build cycle using Maven. Current unit testing best practices

were used as guidelines:

Maven also assists in project workflow such as release and issue

management. Maven also suggests some guidelines on how to layout your

project's directory structure. Once you learn the layout, you can easily

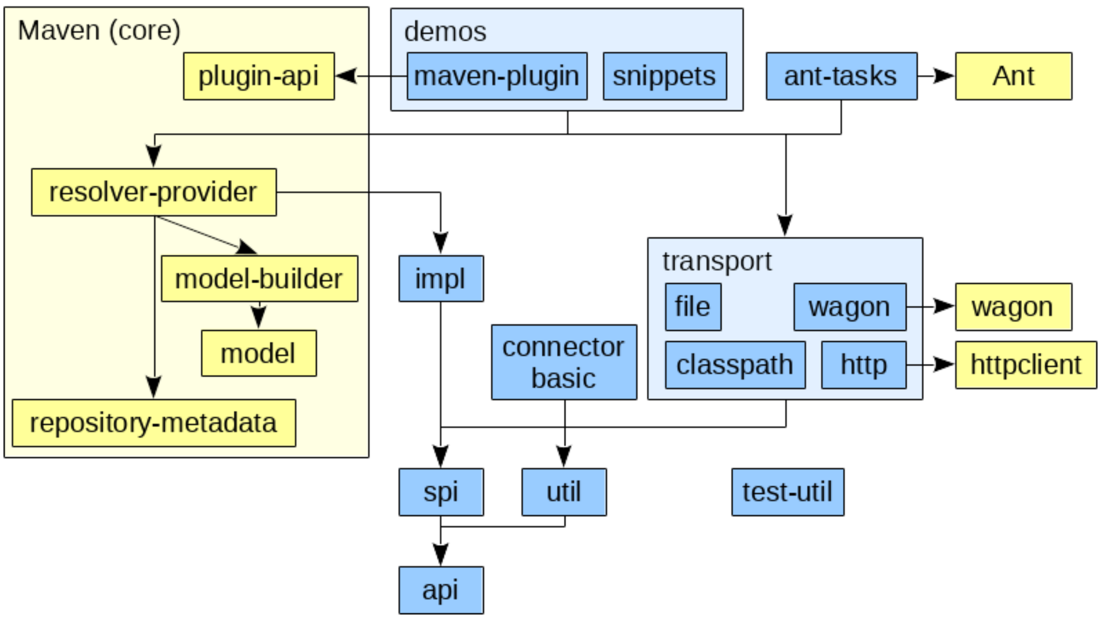
navigate other projects that use Maven.

While takes an opinionated approach to project layout, some projects may not fit

with this structure for historical reasons. While Maven is designed to be flexible to

the needs of different projects, it cannot cater to every situation

without compromising its objectives.



(G) GIT

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

Branching and Merging

The Git feature that really makes it stand apart from nearly every other SCM out there is its branching model.

Git allows and encourages you to have multiple local branches that can be entirely independent of each other. The creation, merging, and deletion of those lines of development takes seconds.

This means that you can do things like:

- **Frictionless Context Switching**. Create a branch to try out an idea, commit a few times, switch back to where you branched from, apply a patch, switch back to where you are experimenting, and merge it in.
- **Role-Based Codelines**. Have a branch that always contains only what goes to production, another that you merge work into for testing, and several smaller ones for day to day work.
- **Feature Based Workflow**. Create new branches for each new feature you're working on so you can seamlessly switch back and forth between them, then delete each branch when that feature gets merged into your main line.
- **Disposable Experimentation**. Create a branch to experiment in, realize it's not going to work, and just delete it - abandoning the work—with nobody else ever seeing it (even if you've pushed other branches in the meantime).

Notably, when you push to a remote repository, you do not have to push all of your branches. You can choose to share just one of your branches, a few of them, or all of them. This tends to free people to try new ideas without worrying about having to plan how and when they are going to merge it in or share it with others.

There are ways to accomplish some of this with other systems, but the work involved is much more difficult and error-prone. Git makes this process incredibly easy and it changes the way most developers work when they learn it.

Distributed

One of the nicest features of any Distributed SCM, Git included, is that it's distributed. This means that instead of doing a "checkout" of the current tip of the source code, you do a "clone" of the entire repository.

Multiple Backups

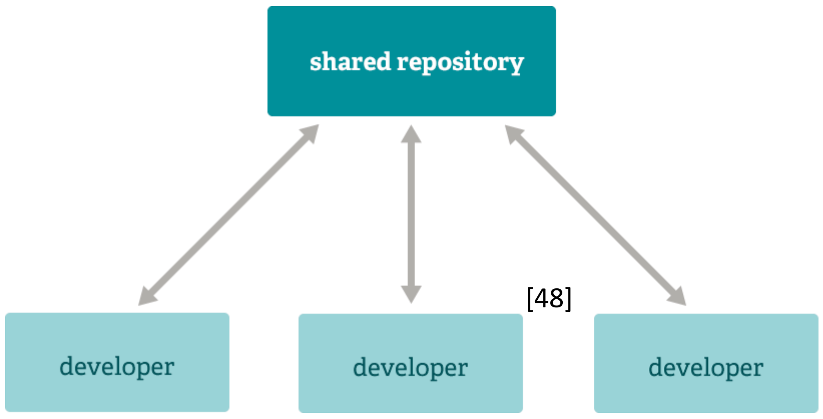
This means that even if you're using a centralized workflow, every user essentially has a full backup of the main server. Each of these copies could be pushed up to replace the main server in the event of a crash or corruption. In effect, there is no single point of failure with Git unless there is only a single copy of the repository.

Any Workflow

Because of Git's distributed nature and superb branching system, an almost endless number of workflows can be implemented with relative ease.

Subversion-Style Workflow

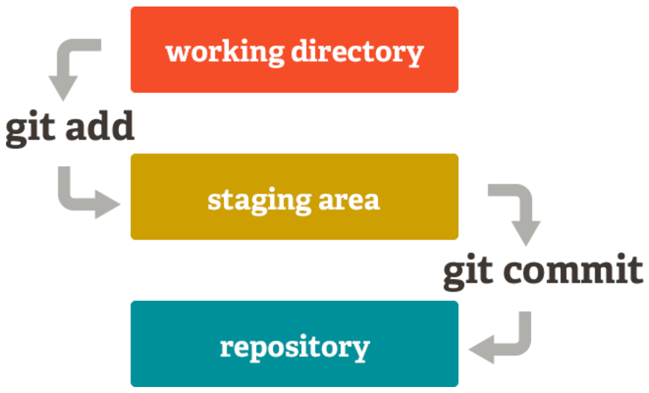
A centralized workflow is very common, especially from people transitioning from a centralized system. Git will not allow you to push if someone has pushed since the last time you fetched, so a centralized model where all developers push to the same server works just fine.



Staging Area

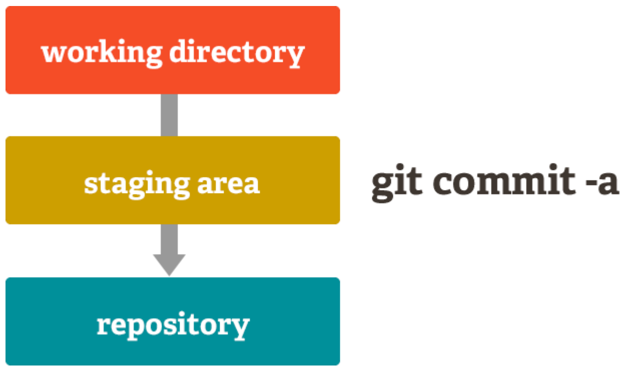
Unlike the other systems, Git has something called the "staging area" or "index". This is an intermediate area where commits can be formatted and reviewed before completing the commit.

One thing that sets Git apart from other tools is that it's possible to quickly stage some of your files and commit them without committing all of the other modified files in your working directory or having to list them on the command line during the commit.



This allows you to stage only portions of a modified file. Gone are the days of making two logically unrelated modifications to a file before you realized that you forgot to commit one of them. Now you can just stage the change you need for the current commit and stage the other change for the next commit. This feature scales up to as many different changes to your file as needed.

Of course, Git also makes it easy to ignore this feature if you don't want that kind of control — just add a '-a' to your commit command in order to add all changes to all files to the staging area.



Free and Open Source

Git is released under the GNU General Public License version 2.0, which is an open source license. The Git project chose to use GPLv2 to guarantee your freedom to share and change free software---to make sure the software is free for all its users.

However, we do restrict the use of the term "Git" and the logos to avoid confusion. Please see our trademark policy for details.

(H) JENKINS

Jenkins is a self-contained, open source automation server which can be
used to automate all sorts of tasks related to building, testing, and
delivering or deploying software.

Jenkins can be installed through native system packages, Docker, or
even run standalone by any machine with a Java Runtime Environment
(JRE) installed.

Jenkins is a free and open source automation server. It helps automate
the parts of software development related to building, testing, and
deploying,
facilitating continuous integration and continuous delivery. It is a server-based
system that runs in servlet containers such as Apache Tomcat. It supports
version
control tools,
including AccuRev, CVS, Subversion, Git, Mercurial, Perforce, ClearCase
and RTC, and can execute Apache Ant, Apache Maven and sbt based projects
as well as arbitrary shell scripts and Windows batch commands. The creator of
Jenkins
is Kohsuke Kawaguchi. Released under the MIT License, Jenkins is free
software.

Builds can be triggered by various means, for example by commit in a version
control system, by scheduling via a cron-like mechanism and by requesting a
specific
build URL. It can also be triggered after the other builds in the queue have
completed. Jenkins functionality can be extended with plugins.

Continuous Integration and Continuous Delivery

As an extensible automation server, Jenkins can be used as a simple CI
server or turned into the continuous delivery hub for any project.

Easy installation

Jenkins is a self-contained Java-based program, ready to run out-of-the-box, with packages for Windows, Mac OS X and other Unix-like operating systems.

Easy configuration

Jenkins can be easily set up and configured via its web interface, which includes on- the-fly error checks and built-in help.

Plugins

With hundreds of plugins in the Update Center, Jenkins integrates with practically every tool in the continuous integration and continuous delivery toolchain.

Extensible

Jenkins can be extended via its plugin architecture, providing nearly infinite possibilities for what Jenkins can do.

Distributed

Jenkins can easily distribute work across multiple machines, helping drive builds, tests and deployments across multiple platforms faster.

Plugins

Plugins have been released for Jenkins that extend its use to projects written in languages other than Java. Plugins are available for integrating Jenkins with most version control systems and bug databases. Many build tools are supported via their respective plugins. Plugins can also change the way Jenkins looks or add new functionality. There are a set of plugins dedicated for the purpose of unit testing that generate test reports in various formats (for example, JUnit bundled with

Jenkins, MSTest, NUnit, etc.) and automated testing that supports automated tests.

Builds can generate test reports in various formats supported by plugins (JUnit support is currently bundled) and Jenkins can display the reports and generate trends and render them in the GUI.

Mailer

Allows configuring email notifications for build results. Jenkins will send emails to the specified recipients whenever a certain important event occurs, such as:

1. Failed build.
2. Unstable build.
3. Successful build after a failed build, indicating that a crisis is over
4. Unstable build after a successful one, indicating that there's a regression

Credentials

Allows storing credentials in Jenkins. Provides a standardized API for other plugins to store and retrieve different types of credentials.

Monitoring external jobs

Adds the ability to monitor the result of externally executed jobs.

SSH agents

This plugin allows managing agents (formerly known as slaves) running on *nix machines over SSH. It adds a new type of agent launch method. This launch method will

1. Open a SSH connection to the specified host as the specified username,
2. Check the default version of Java for that user,
3. [not implemented yet] If the default version is not compatible with Jenkins's agent.jar, try to find a proper version of Java
4. Once it has a suitable version of Java, copy the latest agent.jar via SFTP (falling back to scp if SFTP is not available),
5. Start the agent process.

Javadoc

This plugin adds Javadoc support to Jenkins. This functionality used to be a part of the core, but as of Jenkins 1.431, it was split off into separate plugins.

The plugin enables the selection of "Publish Javadoc" as a post-build action, specifying the directory where the Javadoc is to be gathered and if retention is expected for each successful build.

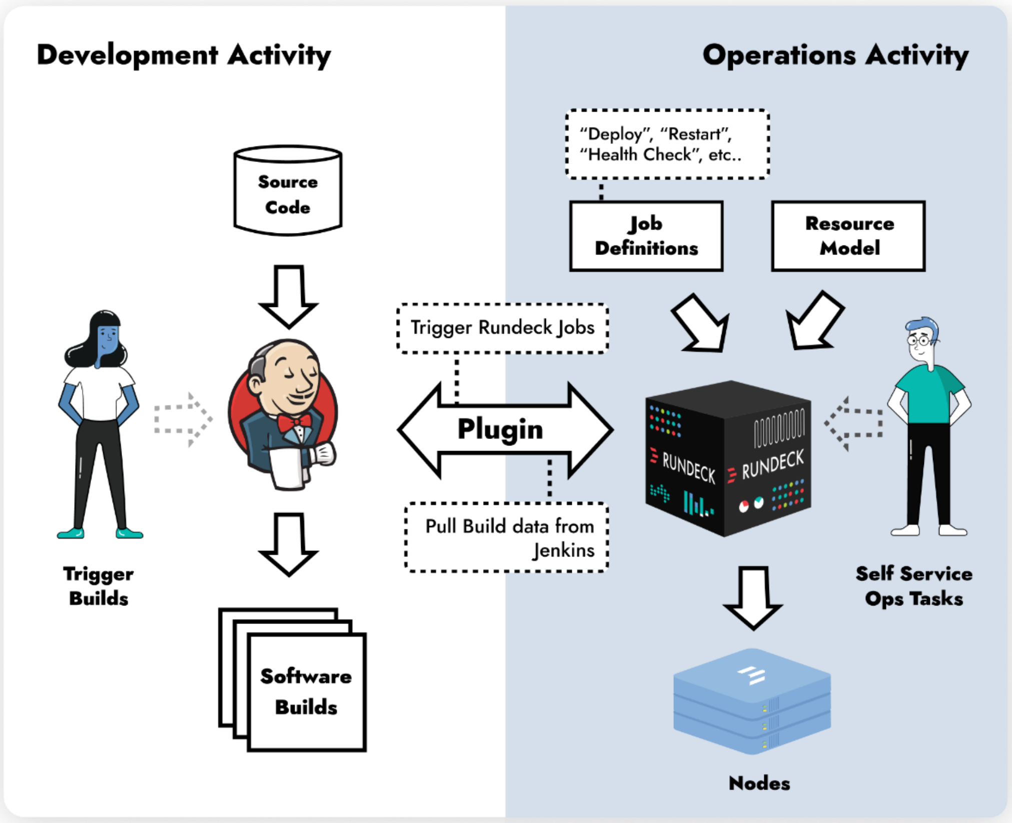
Oneline explanation

Jenkins can be used to schedule and monitor the running of a shell script via user interface instead of command prompt .

Security

Jenkins' security depends on two factors: access control and protection from external threats. Access control can be customized via two ways: user authentication and

authorization. Protection from external threats such as CSRF attacks and malicious builds is supported as well.



(I) JIRA

Jira is a proprietary issue tracking product developed by Atlassian that allows bug tracking and agile project management.

Plan

Create user stories and issues, plan sprints, and distribute tasks across your software team.

Track

Prioritize and discuss your team's work in full context with complete visibility.

Release

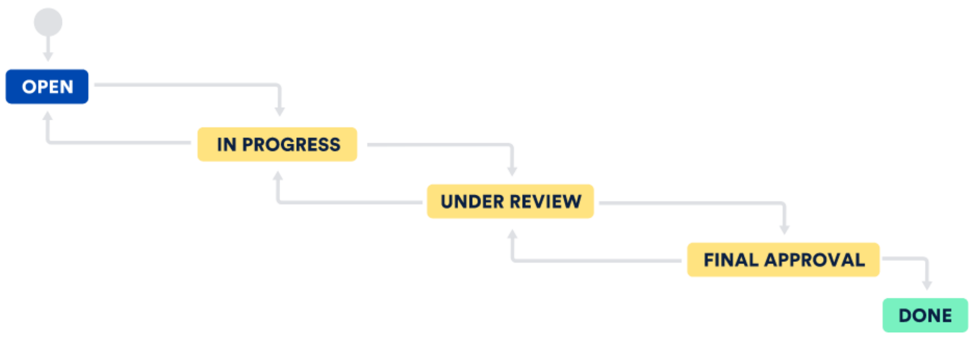
Ship with confidence and sanity knowing the information you have is always up-to- date.

Report

Improve team performance based on real-time, visual data that your team can put to use.

Choose a workflow, or make your own

Every team has a unique process for shipping software. Use an out-of-the-box workflow, or create one to match the way your team works.



Jira Scrum Boards

If Scrum is the framework that guides teams and their roles, events, artifacts, and rules, then the Jira Scrum Board is the visual display of its progress during the development cycle. Much more than a task board, a Jira Scrum Board functions to:

- **Increase communication and transparency**
  The Jira Scrum Board is the single source of truth for the all the work a team needs to complete. Since it can be accessed by any team member at any time, everyone clearly understands what's on his or her plate, and can quickly identify any blockers.

- **Promote sprint planning and iterative development**
  At the heart of the Scrum framework is the Sprint, a designated amount of time (typically two weeks) for teams to build a potentially releasable product increment. The Jira Scrum board is designed so teams can organize their work around the Sprint timeframe.

- **Improve team focus and organization**
  Teams miss project deadlines when they're over-committed on their workload or lose track of key milestones. Jira Scrum Boards provide transparency into the team's work by slicing work into stages and utilizing burndown and velocity reports.

A Jira Scrum Board for every team

Although Jira Scrum Boards are ideal for highly technical teams who practice agile project management, teams of all types can take advantage of the key concepts of scrum and use the Jira Scrum Board to facilitate smooth project management. Here are a few ideas.

- **Software Development**
  When thinking about the scrum methodology, most people think of software teams that build and release code as its main adopters. This is true, and when used with a Jira Scrum Board, teams can also view

information like code commits, branches, pull requests, and deployment status - all in a single view.

- **Marketing**
  Marketers manage large, complex projects - such as product launches and events - that involve multiple teams and skills set. The Jira Scrum Board is a great way to slice up work related to web development, asset and content creation, design needs, and more, so every project launches on time.

- **Business & HR**
  Hiring, staffing, and informing employees of the latest company news is an internal marketing project in itself. When it comes to updating career and company information on the website, creating internal communication hand- outs, and more, Jira Scrum Boards is an excellent tool to track the status of every piece of the project.

Key terms and concepts to know

If you're new to the Jira Scrum Board, there are a handful of terms to familiarize yourself with before you get started.

Sprint

A time-boxed period of time, typically two weeks, during which a "Done," useable, and potentially releasable product increment is created.

Backlog

Owned by the Product Owner, this is a list of features, defects, enhancement, and experiments that need to be done. Each item has a description, rank, size estimate, and value.

User story

Backlog items can be written as stories about a user's experience with a product or feature. A popular format is: As a <role>, I want to <action> so that <value or justification>

Issue

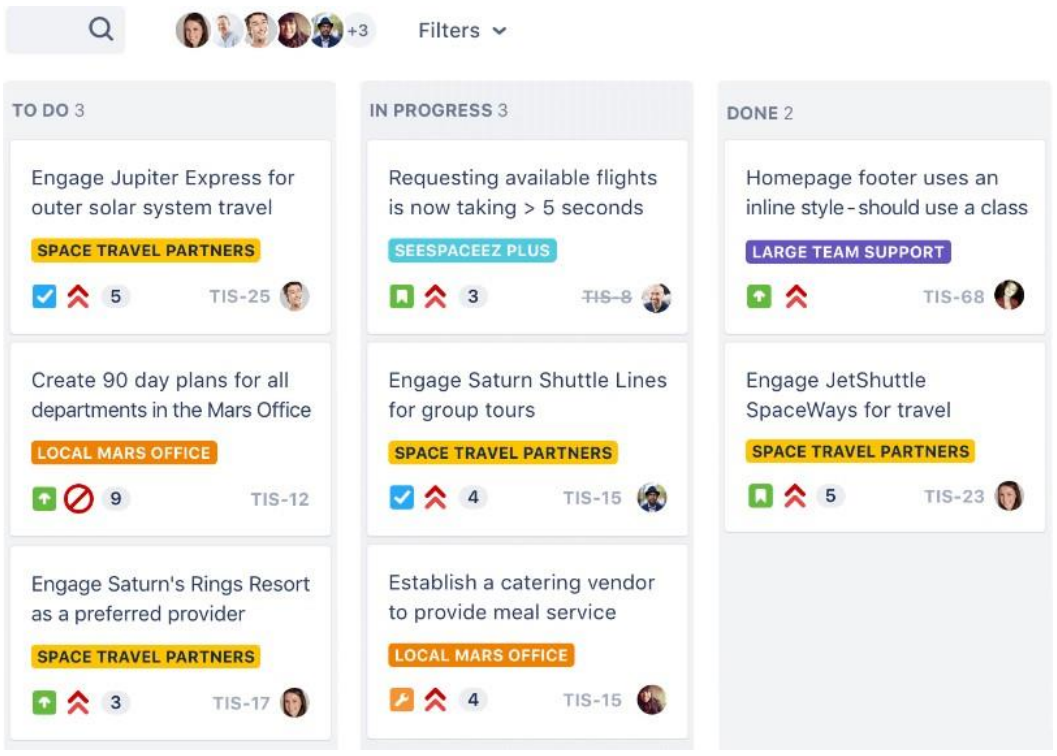
Also known as the User Story, in a Jira Scrum Board the issue contains all tasks, dependencies, and relevant information related to a single item of work.

Epic

An epic captures a large body of work. It is essentially a large user story that can be broken down into a number of smaller stories. It may take several sprints to complete an epic.

Swimlane

A swimlane is a means of categorizing issues on the Jira Scrum Board so that agile teams can easily see issues grouped by different criteria, say by user.


+3 Filters

TO DO 3

Engage Jupiter Express for outer solar system travel
SPACE TRAVEL PARTNERS
5 TIS-25

Create 90 day plans for all departments in the Mars Office
LOCAL MARS OFFICE
9 TIS-12

Engage Saturn's Rings Resort as a preferred provider
SPACE TRAVEL PARTNERS
3 TIS-17

IN PROGRESS 3

Requesting available flights is now taking > 5 seconds
SEESPACEEZ PLUS
3 ~~TIS-8~~

Engage Saturn Shuttle Lines for group tours
SPACE TRAVEL PARTNERS
4 TIS-15

Establish a catering vendor to provide meal service
LOCAL MARS OFFICE
4 TIS-15

DONE 2

Homepage footer uses an inline style - should use a class
LARGE TEAM SUPPORT
TIS-68

Engage JetShuttle SpaceWays for travel
SPACE TRAVEL PARTNERS
5 TIS-23

CHAPTER 5
CONCLUSION

Results

The following enlist the key areas on which I have made some impact while working on the project:

- Developed User Friendly UI Components using angular and react.
- Developed a feature of create and search adjustments users made.
- I developed few APIS to added user authorization using spring boot.
- Designed some UI screens to present the data to the end user.
- Developed React components to request different reports.
- Fixed some bugs arising in the project.
- Developed Unzip Utility tool.
- Integrated continuous changes in user requirement using agile principles.

My Key Learnings

With the opportunity I got to work at the Startup in SGGSC I leant some valuable lessons which will be of great help in my career ahead.

I got to learn some lifelong lessons involving co-operating with each other, having patience while getting stuck at some point, taking responsibilities specially of my work and being motivated throughout to achieve the goal.

Besides these, I feel to have made some progress on my technical aspect majorly in Java frameworks, Angular and react. I will try to further enhance my skills on the foundations I have built while working for the project.

REFERENCES

1] https://git-scm.com/about/distributed

2] https://www.atlassian.com/software/jira/features

3] https://angular.io/

4] https://reactjs.org/

5] https://www.jenkins.io/

6] https://spring.io/

7] https://junit.org/junit5/docs/current/user-guide/

8] https://www.progress.com/faqs/datadirect-jdbc-faqs/how-does-jdbc-work

9] https://maven.apache.org/

10] https://www.atlassian.com/git/tutorials/what-is-git

11] https://docs.oracle.com/javase/tutorial/jdbc/overview/index.html