Software Architecture

Types of Software

- System Software,Application Software
- Proprietary, open-source and freeware
- Web, Desktop , Mobile

SW Architecture Basics

- Design Patterns – Reusable solutions to common problems
- Principles of Design Patterns : Encapsulation, modularity, reuse
- Benefits of Design Patterns: Code Maintainability, scalability, readability
- Common Design patterns: Singleton, Factory, Observer, Strategy, Adapter
- Anti-Patterns: Commnly failed and bad practices

MVC (Model View Controller)

Core of MVC :

Model : Data and business logic

View : Display data and interacts with the user

Controller: Will manage input, updates the model and renders the view

Advantages

Separated concerns or issues

Easy to test

parallel development

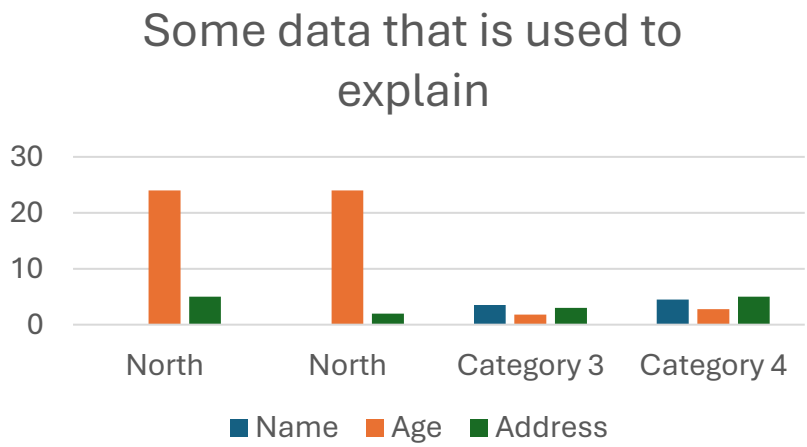
Name	Age	Address

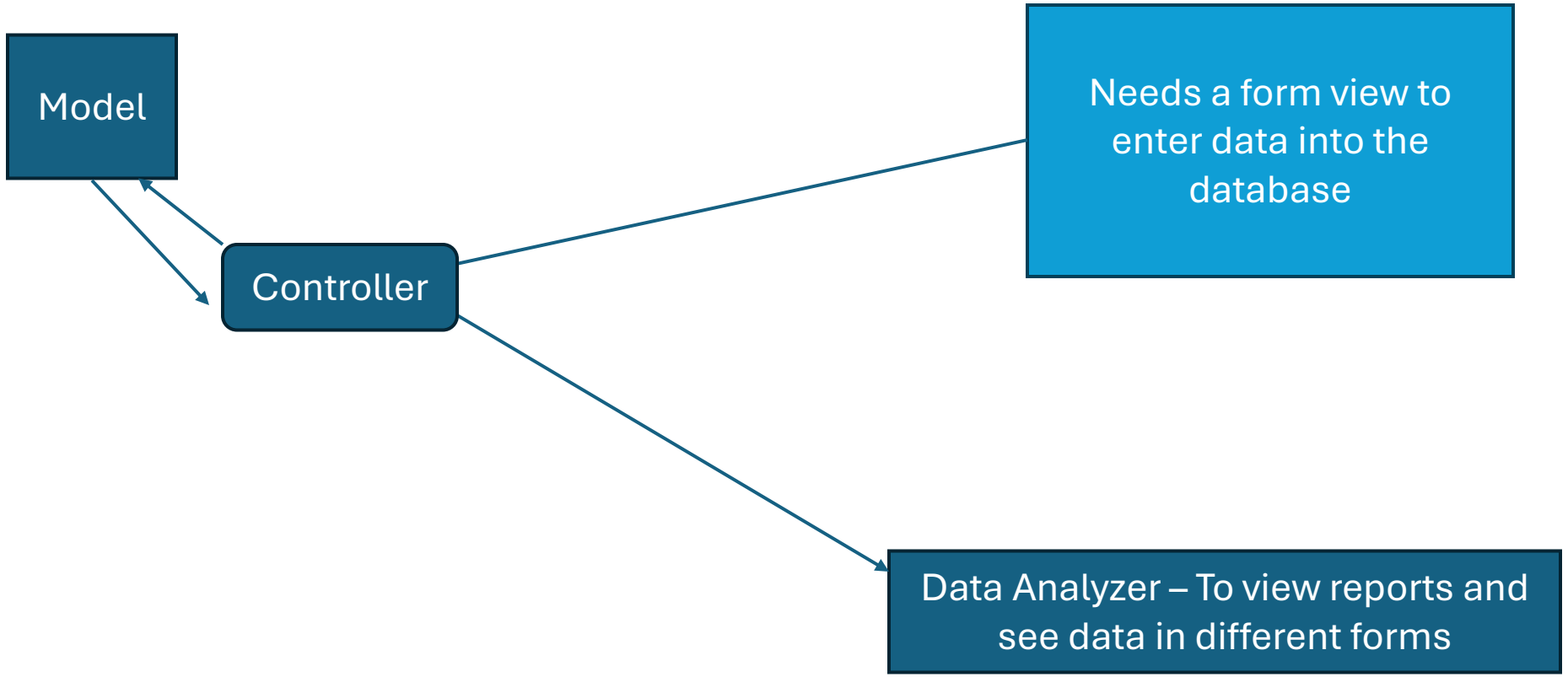
Variants of MVC

MVVM (Model View View Model)
Model View Presenter

Problems :

Complexity
learning time (Curve)
performance of applications




Model
Controller
Needs a form view to enter data into the database
Data Analyzer – To view reports and see data in different forms

Service Oriented Architecture (SOA)

- Core principles of SOA
  - Loose Coupling
  - Resuability
  - Interoperability
- Components of SOA
  - Services
  - Service Contract
  - Service Interface
  - Service Implementation
  - Communication
    - SOAP
    - REST
    - messaging protocol
  - Challenges/Issues
- Governance
- Security
- Performance

SOA

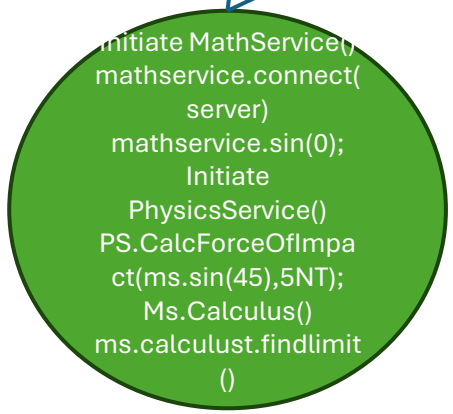
Service
Contract
Connection Point

REMOTE SERVER, on any os

Math Service()
{
sin(angle) { return sin(angle)}
cos(angle)
tan(angle)
sec(angle)
cos(angle)
cot(angle)
}
MS.Calculus{ findlimit() }

Physicserivce
{
CalcForceOfImpact()

}



DrawingPaint()
{
initiateMS
DrawLineATmouseMove(int x, int y){
Ms.Sin(prev.x,prev.y,x, y);
}

Microservice Architectures

- Core of MicS.Arch
  - Many services with a single responsibility
  - Decentralized data management
  - Independent Deployment
- Inter-Service Communication
  - REST
  - gRPC
  - messaging queues

  - Issues/Challenges to look out for
    - Data consistency
    - Service discovery
    - monitoring

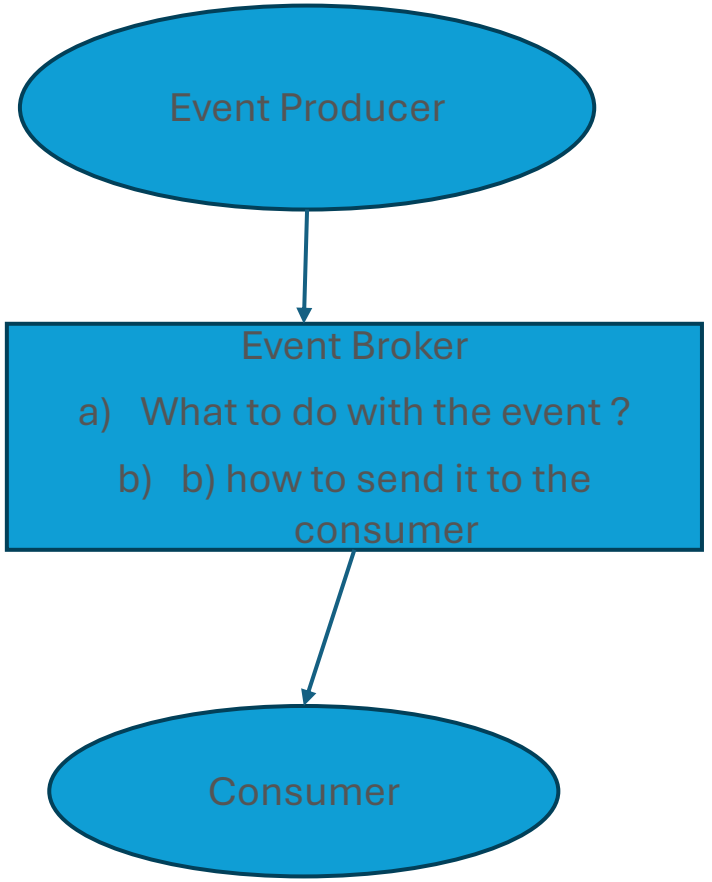
Event Driven Architecture

Components of EDA

- Event producers (service, hardware)
- Event Consumer (Applications)
- Event broker (Middleware)

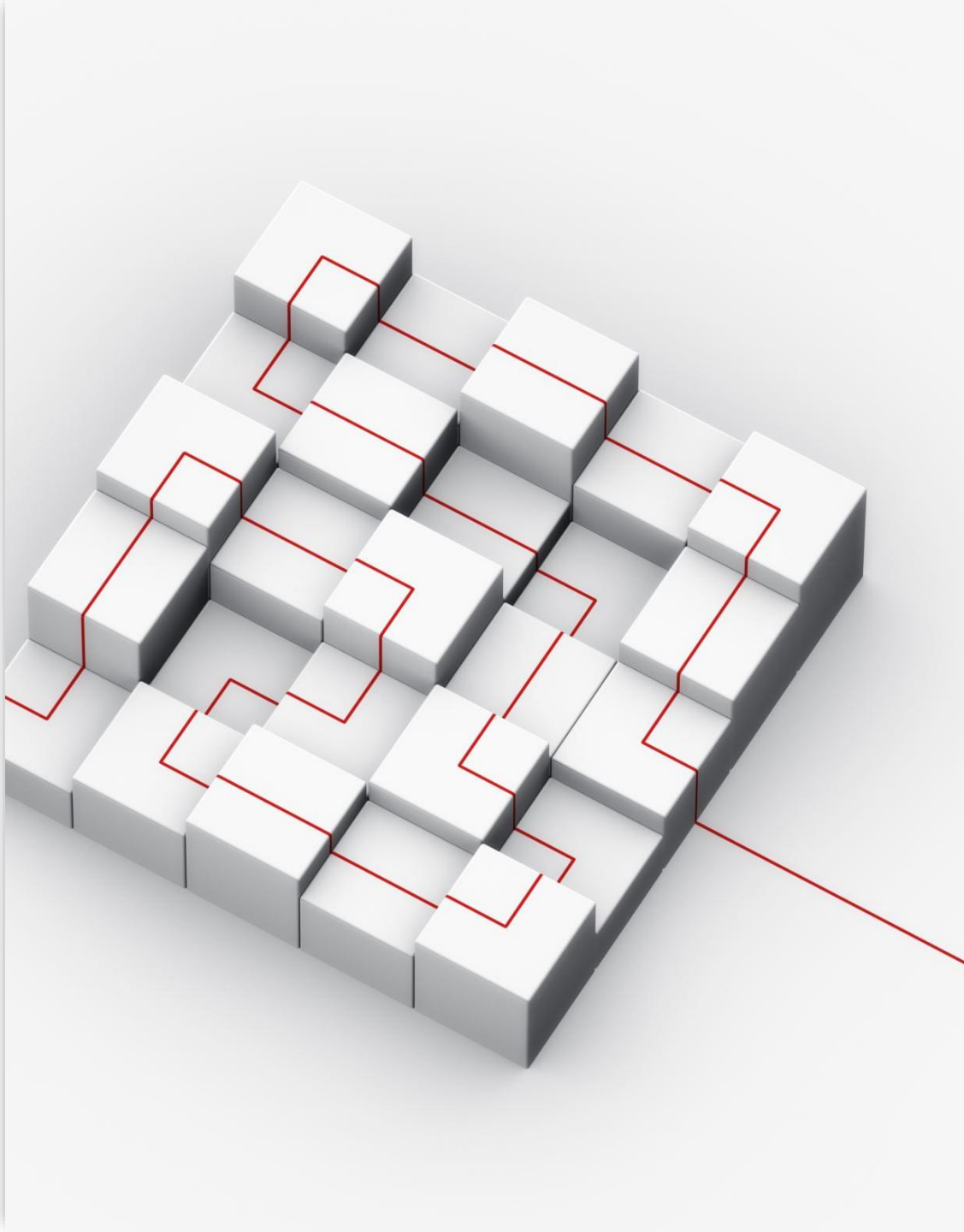
Advantages:

- Scalable
- real time processing
- decoupling



Domain Driven Design (DDD)

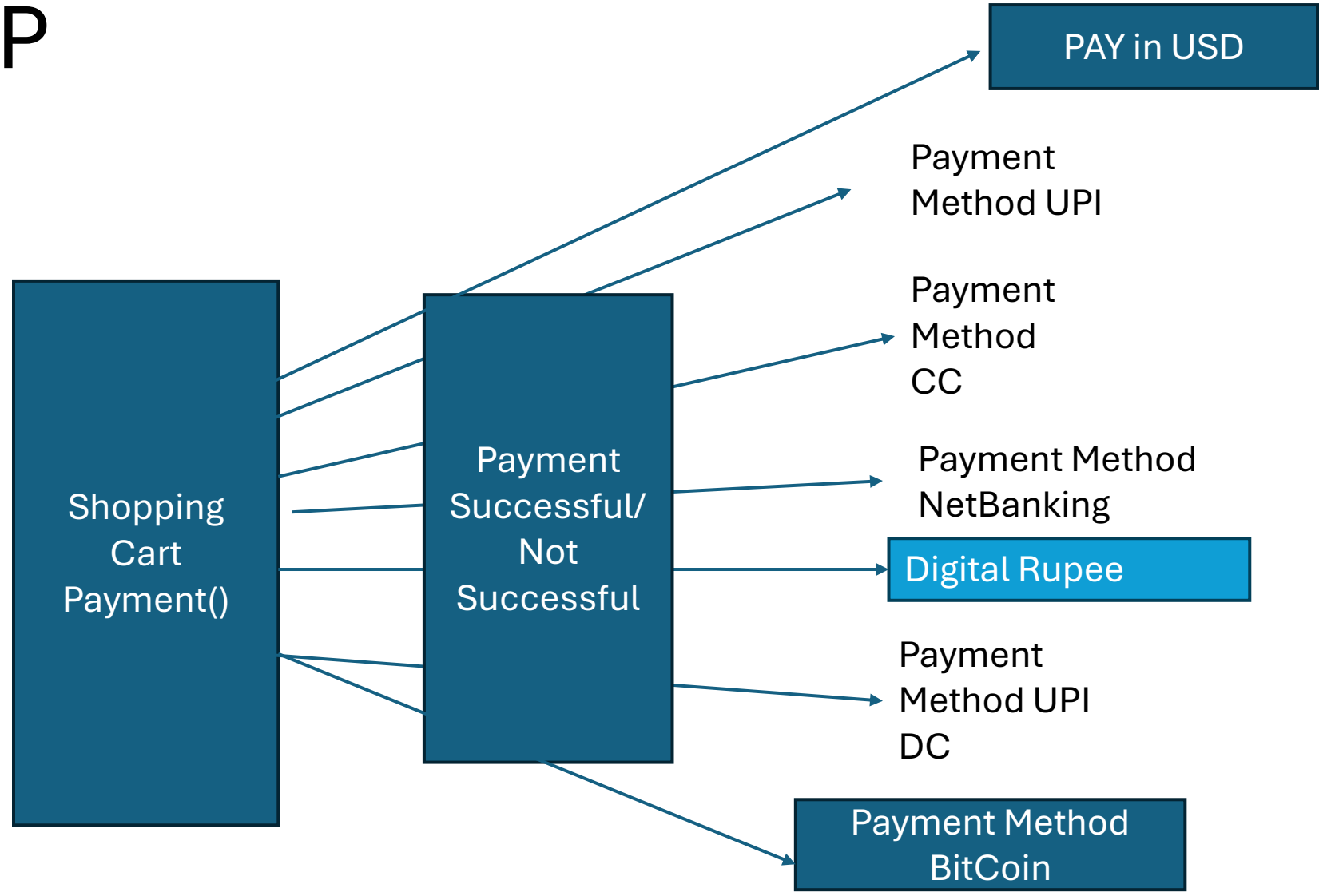
- Building Blocks:
  - Entities , value objects, aggregates, repositories
- Strategic Design with DDD
  - Bounded Contexts
  - Context Maps
- Tactical Design with DDD
  - Domain Models (Retail, eCommerce,)
  - Designing aggregates



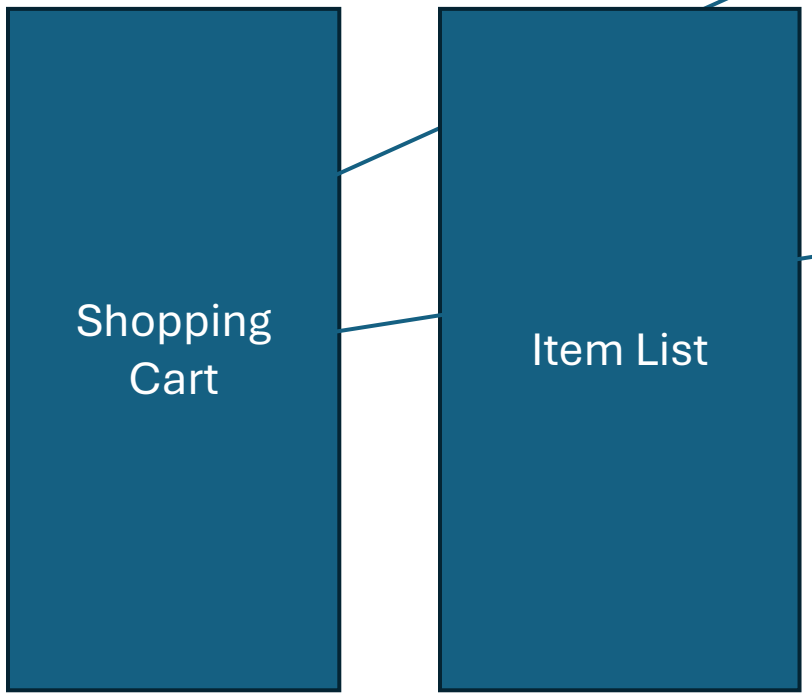
Software Design Principles

- SOLID
  - Single Responsibility Principle – A class should have one and only one reason to change.
  - Open/Closed Principle – SW entities should be open for extension but closed for modification.
  - Liskov Substitution Principle – Subtypes must be substitutable for their base types. float -> money => float currency or money currency;
  - Interface Segregation Principle => Clients should not be forced to depend on interfaces they do not use.
  - Dependency Inversion Principle – Depend on abstractions not on concretions.
    - High Level Modules should not depend on the low level modules
- DRY (DON'T REPEAT YOURSELF)
  - Avoid duplicating code
- KISS
  - Keep designs simple and straightforward
- Code Reviews, Refactoring Techniques

DIP


PAY in USD
Payment
Method UPI
Payment
Method
CC
Payment Method
NetBanking
Digital Rupee
Payment
Method UPI
DC
Payment Method
BitCoin
Shopping
Cart
Payment()
Payment
Successful/
Not
Successful

DRY



next Project

Or

new Module in the same project

DO NOT REWRITE THE CODE FOR SHOPPING CART and ITEM LIST

p1.c

```
main(){}
calctraj(){


}
```

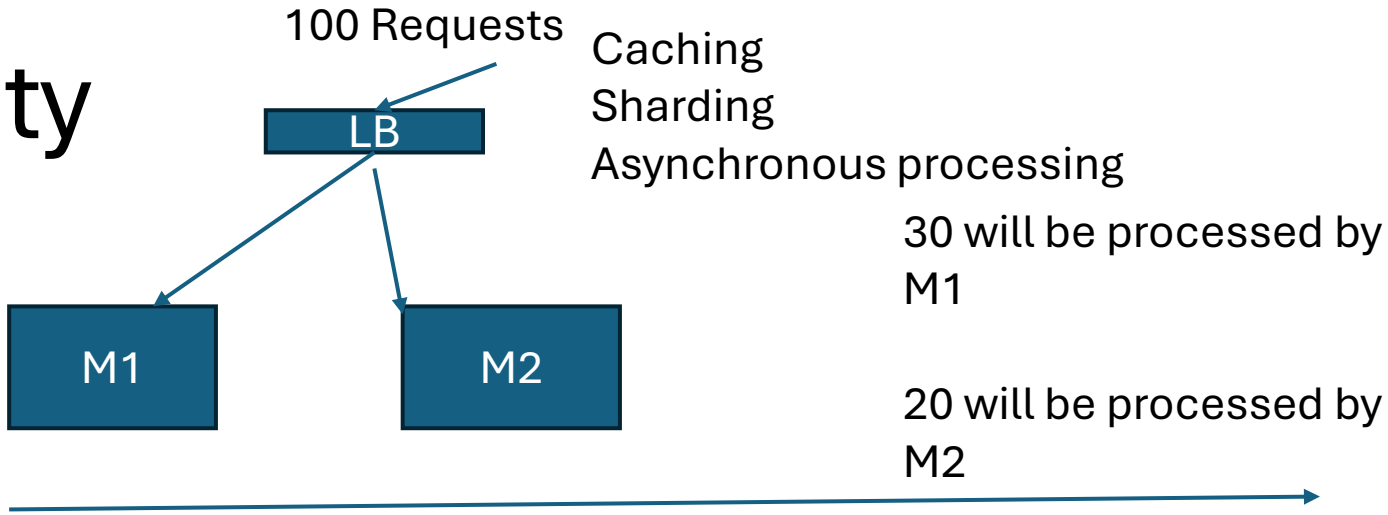
p2.c

```
#include “p1.c”

main(){
calctraj()

}
```

Design for Scalability

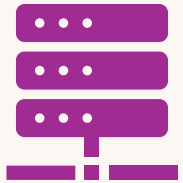


- Two kinds – Horizontal and Vertical
- Load Balancing,
- Statelessness
- Maintainability in Design
  - Code Readability
  - Modularity
  - Documentation

Server 1:
32 GB RAM,
1 TB storage

After Vertical Scaling :
+32 GB RAM =64 GB
+1TB = 2TB

Trends in SArch



Server Less Architecture :

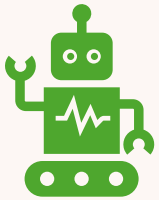
Functions as a service, event driven, automatic scaling

Progressive Web Apps (PWAs)

Offline Capability

Push notifications

Performance



Ai/ML and its impact

Data Pipelines – Data Patterns

Model Serving – Model which works

Edge Computing

Cloud Computing

My Hardware costs are low

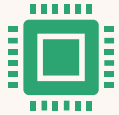
Hosted by someone else

PaaS

SaaS

IaaS

On demand resource availability



Scaling – 2 machines -> 25 machines->1500 machines->1 machine

Cost Efficient

DevOPS and Agile

Develop and Operations

Communication Flow

Development -> Finances->Management

Continuous Integration

Continuous Deployment

Collaboration

CI/CD

Jenkins

GitHub

Microservices and Containerization Architecture

MS – small autonomous services

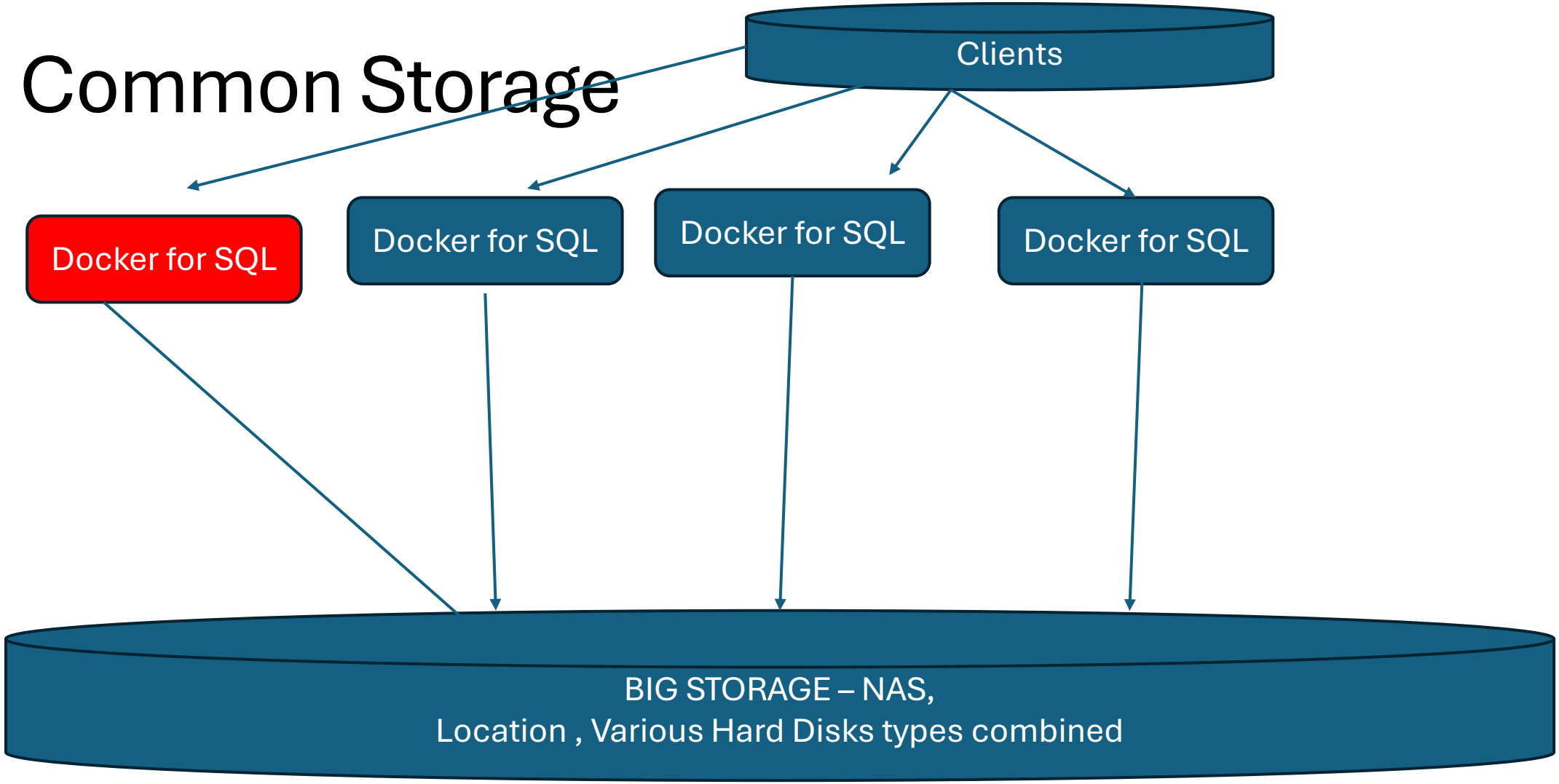
Decentralized Data Management

- Data partitioning
- eventual consistency
- Independent Deployment
  - Continuous Deployment
  - Rollback Strategy

Polyglot Persistence

- Using multiple storage technologies

Common Storage



Containers/COntainerization

- Kubernetes
- Orchestration and Automation
  - To manage containerized application at scale
- Service Discovery and Load Balancing
  - Reliable service access
- Self Healing
  - If a container fails, automatically restart
- Lightweight , portable environments
- Image Management
  - Creating and managing (Docker Images)
- Docker Hub and Registries
  - Storing and Sharing Container images

Agile

- To make software development flexible

Terminologies

Manifesto

Project management

SDLC

Agile SCRUM

Kanban

Daily- Stand up

Agile Design

Product Management

Agile Scale

Release Planning

Iteration Planning

AGILE CRM

Agile Goods and bads

- Focus on customer VALUE
  - Features (Most Important) are made first, with feedback from customer, divide the work into small manageable tasks
  - Banking System : PIN based authentication, Biometric
- Team can decide what to work
- Stakeholder – All the people (Customer, Project Manager, Product Owner, developers) Everybody is in sync.
- Early and Continuous Delivery – Minimum Product that works – Banking – Accounts, Customers we need,-> Transactions -> UPI, Credit Cards, Debit Card, Transactions ON mars, pluto
- We can see issues early and resolve them rather than later
- More time to test, bug free product, less fighting and blaming, all happy with quality of work.

Bads

- Unpredictable – Budget, Time line
- Dependency on customer availability
- Scaling is a nightmare – good for small to medium size teams. More people, more issues, more communication gaps, and getting in co-ordination
- Team Dynamics – People can fall sick, Rainfall can disconnect,
- Increased Overhead: Time, Money
- Project Cost – 100,000 duration 6 months-> people=10
- 5 people quit -> Recruitment cost ->Learning curve->=6 months +(Delay)=>9 months.

AGILE SDLC

- Initiation (Someone came with an idea to be developed)
- Planning (Planning is done with the initiation, changes can happen to the idea during planning also)
- Development Process is called as sprint
  - Time line – 7-15 days
  - Goal
  - This much will be done
  - Next iteration

- Release
- Operations

- RETIRE IT !

Idea – Build a car
Planning to build the car
I want Radar to detect vehicles all around the car
Should have protection against people leaning onto the car when parked -> If the car is locked and someone touches it generate a 110 volt DC shock
In case of crash (when airbags open), the steering should go into a fold mode where it cannot hurt the passenger
If the car cannot stop when braked and radar says it wil crash, eject the driver and occupants with parachutes

Release with first version

PRODUCT

- Faster to goto market
- Software -> Release it (even if there are bugs)->Updates->Patches

- Change Management

Car

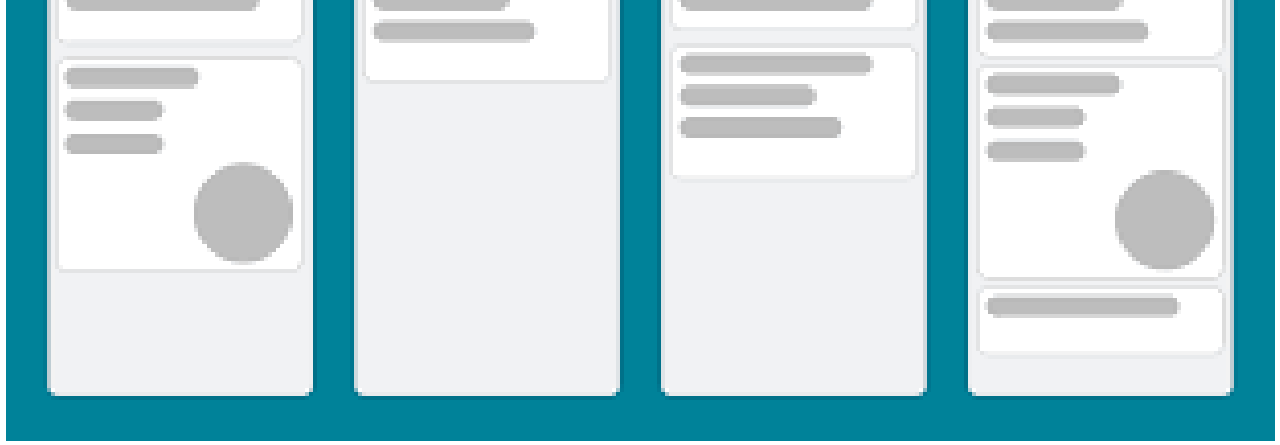
- Engine – Team -> Daily Standup-> What I did yesterday, what will I do today->Issues faced -> How to resolve them (self, or someone else)
- Tyres - Team
- ECU – Team
- Body -> Team

- EngineTL -> I can max speed at 260 KMS per hour
- TYRES TL-> H, I will have to change to V size
- ECU-TL->ECU Cannot handle that speed -> I need to work on it
- BODY->TL->I need to remodel the body to support your wishlists

SCRUM vs AGILE

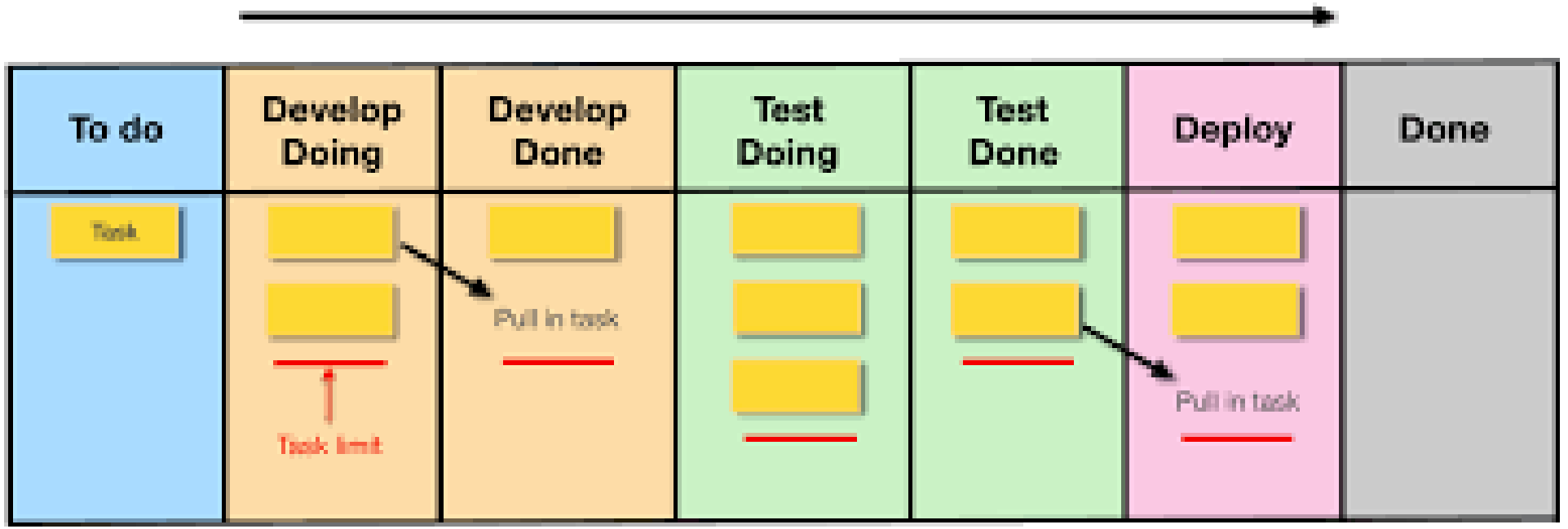
- AGILE is a project management philosophy
- Scrum is a Specific Agile Methodology which is used to facilitate a project

KanBan



Sl. No	Task	Assigned to	Doing	Review	Done
1	Design Kernel Module	Raju			V
2	Go and fix issues in make file	Vishaka		R	

Kanban


To do
Develop Doing
Develop Done
Test Doing
Test Done
Deploy
Done
Task
Pull in task
Task limit
Pull in task


KANBAN Desk
Stories
To Do
In Progr
Testing
Done !