Software Engineering Fundamentals

greenwich.edu.vn

UNIVERSITY of GREENWICH

Alliance with FPT Education

Table of Contents

UNIVERSITY of GREENWICH

Alliance with FPT Education

SOFTWARE ENGINEERING

Requirements, Design, Construction, Testing

What is Software Engineering?

Software engineering is the application of a systematic, disciplined and quantifiable approach to the development, operation, and maintenance of software.

Definition by IEEE

Software Engineering

- Software engineering is:
  - – An engineering discipline that provides knowledge, processes, tools, and approaches for:
    - Defining software requirements
    - Performing software design
    - Software construction
    - Software testing
    - Software maintenance tasks
    - Software project management

Software Development Activities

- Software development always includes the following activities (to some extent):
  – Requirements analysis
  – Design
  – Construction
  – Testing
- These activities do not follow strictly one after another (depends on the methodology)!
  – Often overlap and interact

$\left.\begin{array}{l} \\ \\ \\ \end{array}\right\}$ **Software Project Management**

UNIVERSITY of GREENWICH

Alliance with FPT Education

SOFTWARE REQUIREMENTS

Functional & Non-Functional Requirements, Requirements Specification

Software Requirements

- Software requirements describe the functionality of the software
  - Answer the question "what?", not "how?"
  - Define constraints on the system
- Two kinds of requirements
  - Functional requirements
  - Non-functional requirements

Requirements Analysis

- Requirements analysis starts from an idea about the system
  - Customers usually don't know what they need!
  - Requirements come roughly
    - Adjusted during the development
  - Requirements change constantly
- The outcome is some requirements documentation
  - Software Requirements Specification (SRS) / User Stories / UI prototype / informal system description / etc.
- Prototyping is often used, especially for the user interface (UI)

Software Requirements Specification (SRS)

- The Software Requirements Specification **(SRS)** is a formal requirements document
- SRS describes in details:
  – Functional requirements
    - Business processes
    - Actors and use-cases
  – Non-functional requirements
    - E.g. performance, scalability, hardware, integrations, constraints, security, etc.

Software Requirements

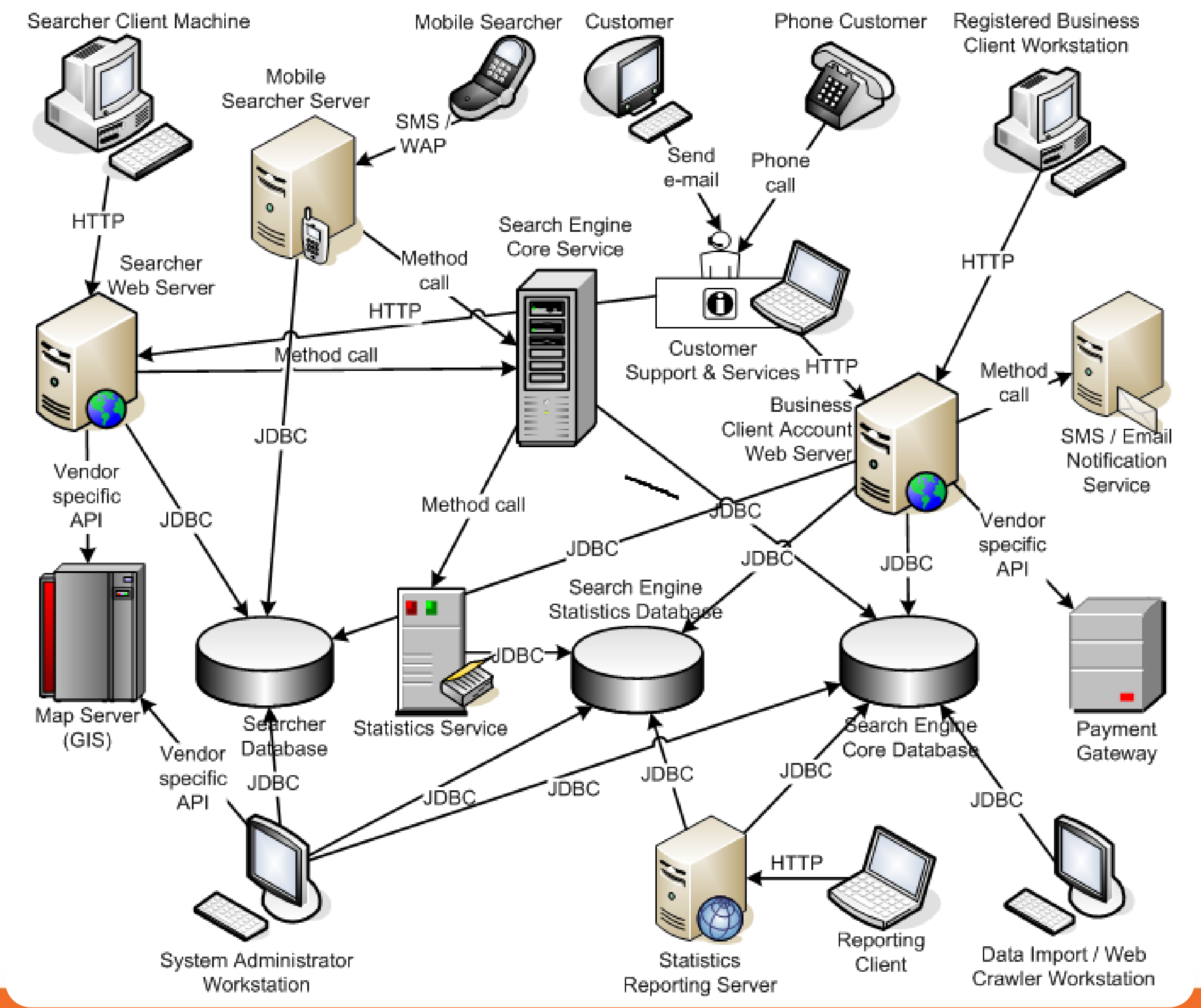
- It is always hard to describe and document the requirements in comprehensive way
  - Good requirements save time and money
- Requirements always change during the project!
  - Good requirements reduces the changes
  - UI prototypes significantly reduce changes

SOFTWARE ARCHITECTURE AND SOFTWARE DESIGN

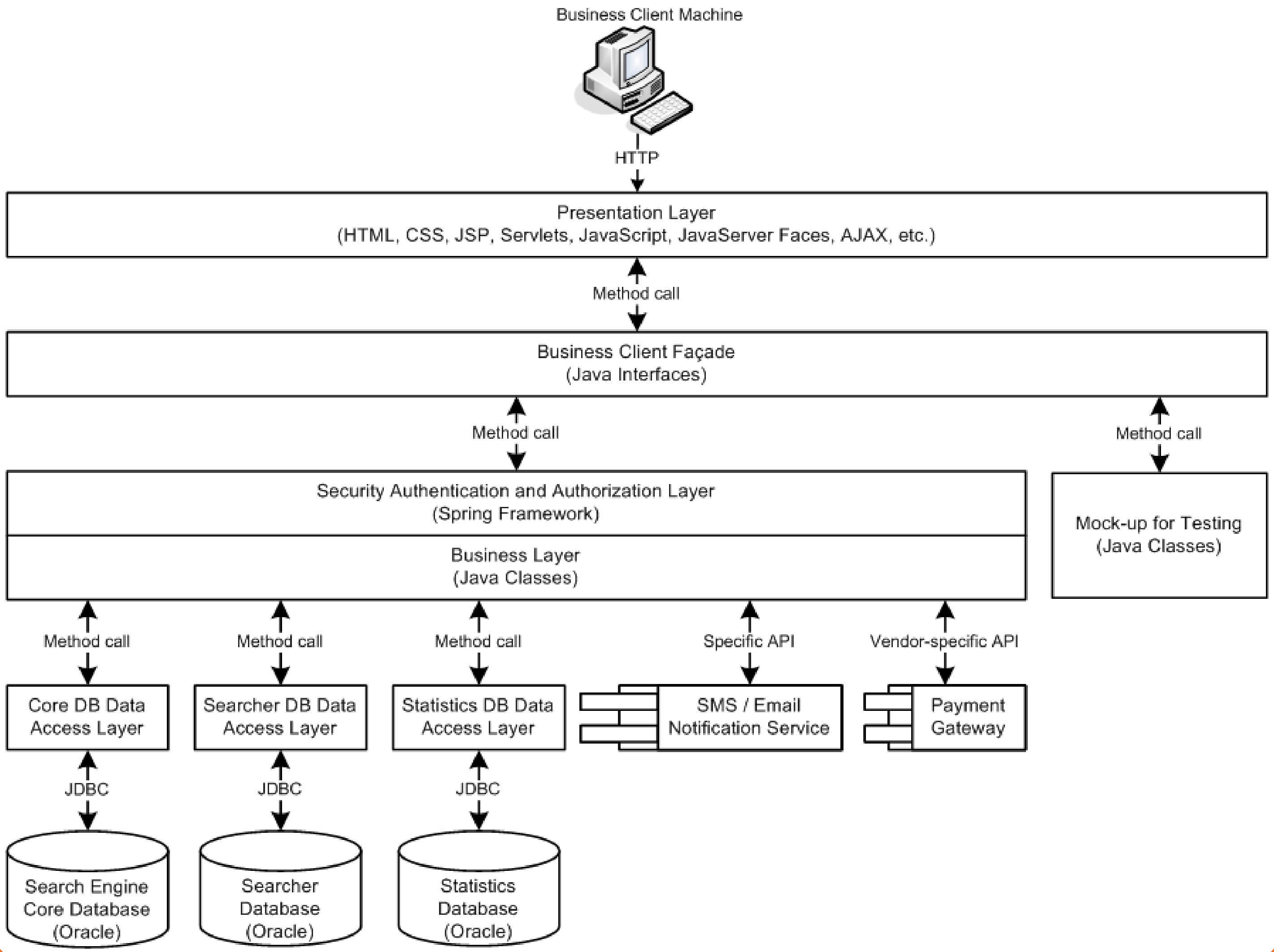
Software Architecture and Software Design

- Software design is a technical description (blueprints) about how the system will implement the requirements
- The system architecture describes:
  - How the system will be decomposed into subsystems (modules)
  - Responsibilities of each module
  - Interaction between the modules
  - Platforms and technologies

System Architecture Diagram – Example

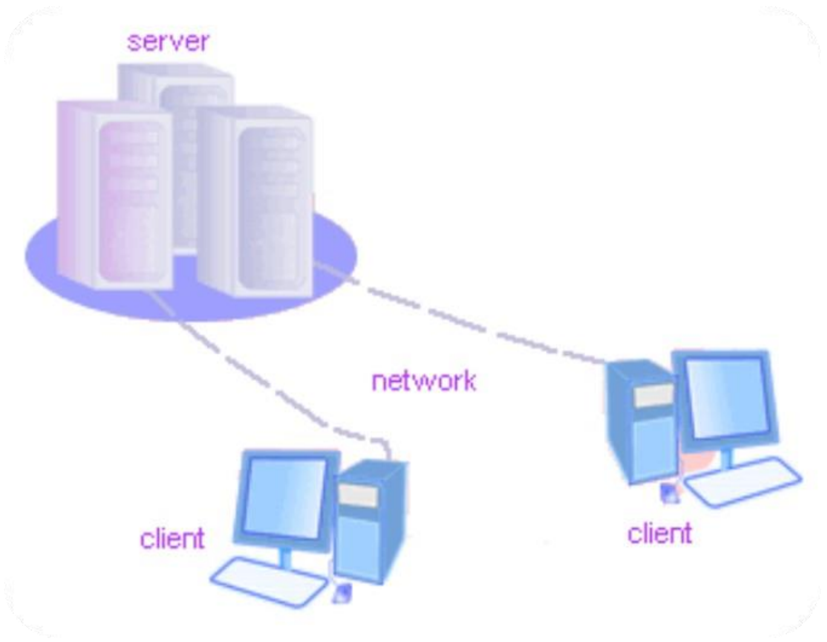


Software Architecture Diagram – Example



Business Client Machine
HTTP
Presentation Layer
(HTML, CSS, JSP, Servlets, JavaScript, JavaServer Faces, AJAX, etc.)
Method call
Business Client Façade
(Java Interfaces)
Method call
Method call
Security Authentication and Authorization Layer
(Spring Framework)
Business Layer
(Java Classes)
Mock-up for Testing
(Java Classes)
Method call
Method call
Method call
Specific API
Vendor-specific API
Core DB Data Access Layer
Searcher DB Data Access Layer
Statistics DB Data Access Layer
SMS / Email Notification Service
Payment Gateway
JDBC
JDBC
JDBC
Search Engine Core Database (Oracle)
Searcher Database (Oracle)
Statistics Database (Oracle)

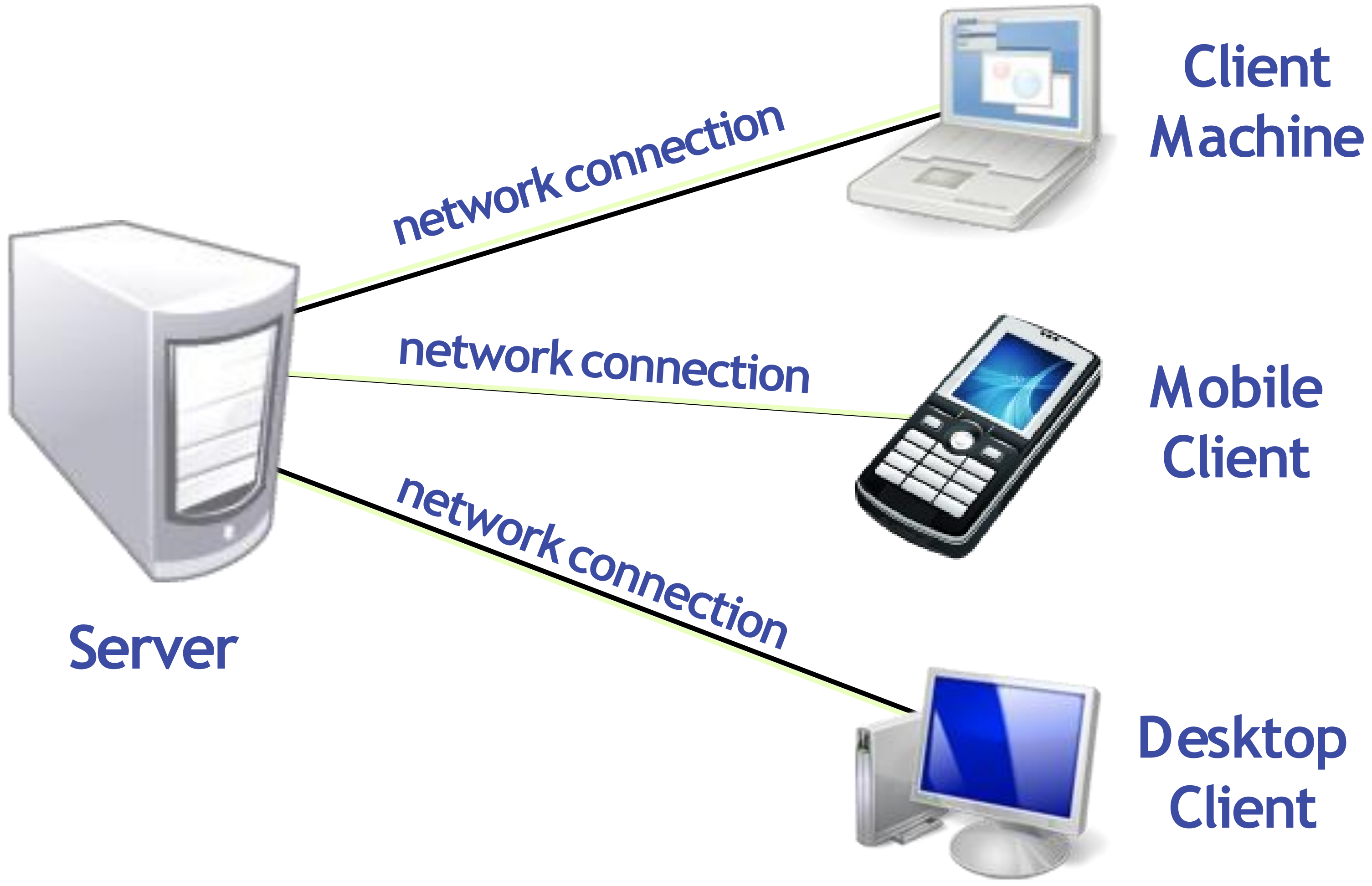
CLIENT-SERVER ARCHITECTURE



Client-Server Architecture

- The client-server model consists of:
- Server – a single machine / application that provides services to multiple clients
- Could be IIS based Web server
- Could be WCF based service
- Could be a services in the cloud
- Clients –software applications that provide UI (front-end) to access the services at the server
- Could be WPF, HTML5, Silverlight, ASP.NET, …

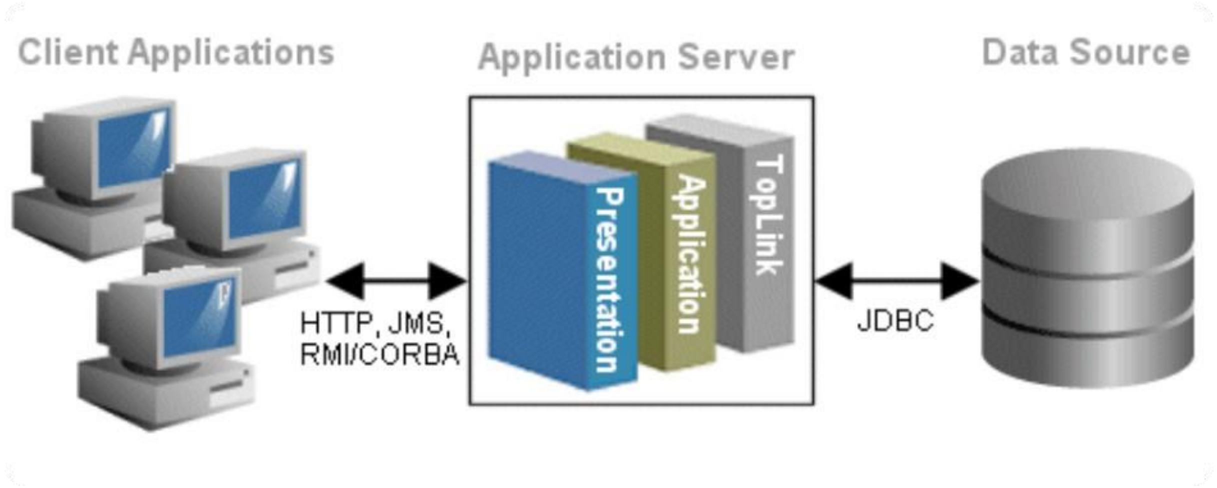
The Client-Server Model



UNIVERSITY of GREENWICH

Alliance with FPT Education

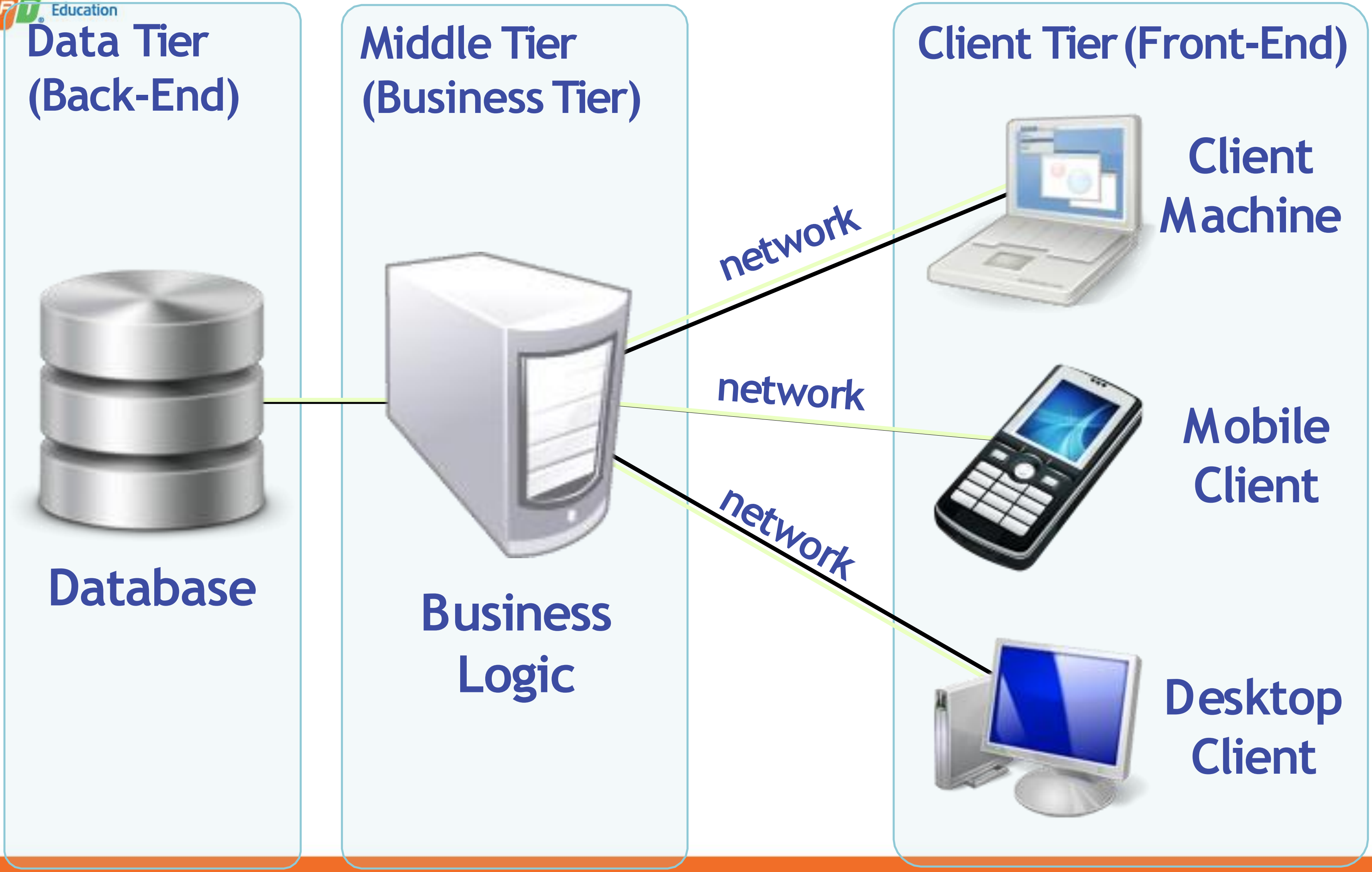
3-TIER / MULTI-TIER ARCHITECTURES



The 3-Tier Architecture

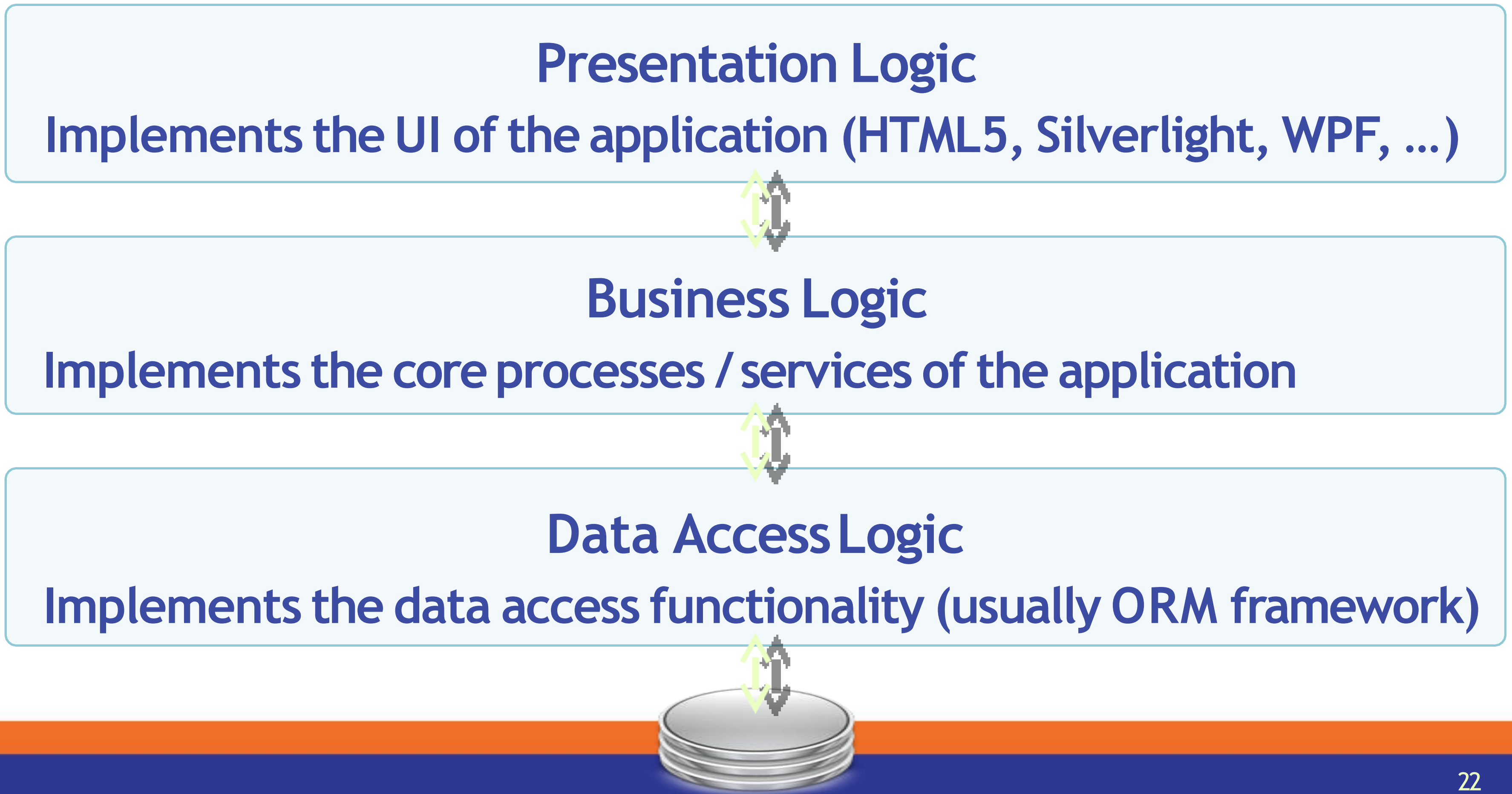
- The 3-tier architecture consists of the following tiers (layers):
  - Front-end (client layer)
    - Client software - provides the UI of the system
  - Middle tier (business layer)
    - Server software - provides the core system logic
    - Implements the business processes / services
  - Back-end (data layer)
    - Manages the data of the system (database /cloud)

The 3-Tier Architecture Model



Typical Layers of the Middle Tier

The middle tier usually has parts related to the front-end, business logic and back-end:



Presentation Logic

Implements the UI of the application (HTML5, Silverlight, WPF, …)

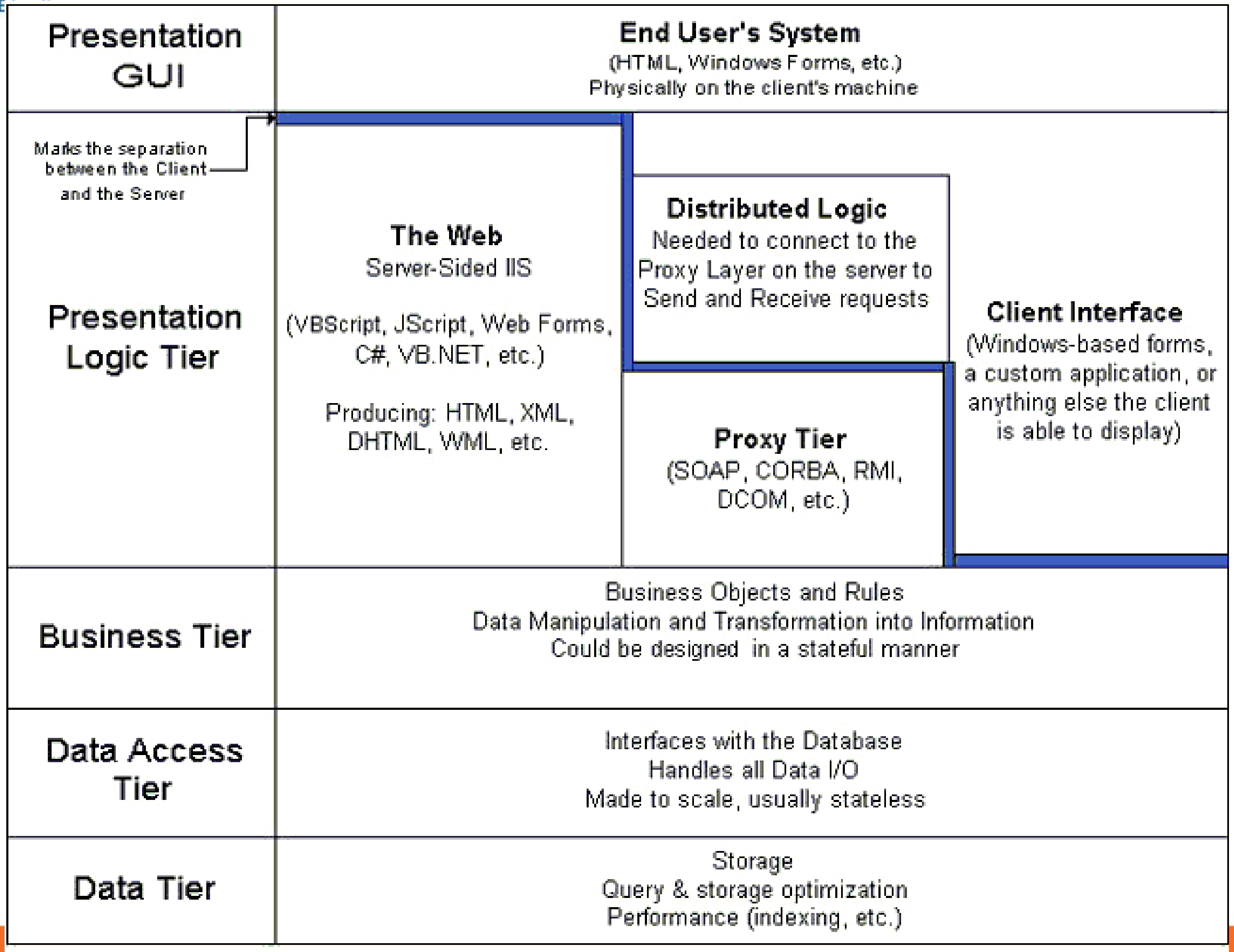
Business Logic

Implements the core processes / services of the application

Data Access Logic

Implements the data access functionality (usually ORM framework)

Multi-Tier Architecture



Presentation GUI	End User's System (HTML, Windows Forms, etc.) Physically on the client's machine		
Marks the separation between the Client and the Server Presentation Logic Tier	**The Web** Server-Sided IIS (VBScript, JScript, Web Forms, C#, VB.NET, etc.) Producing: HTML, XML, DHTML, WML, etc.	**Distributed Logic** Needed to connect to the Proxy Layer on the server to Send and Receive requests **Proxy Tier** (SOAP, CORBA, RMI, DCOM, etc.)	**Client Interface** (Windows-based forms, a custom application, or anything else the client is able to display)
Business Tier	Business Objects and Rules Data Manipulation and Transformation into Information Could be designed in a stateful manner		
Data Access Tier	Interfaces with the Database Handles all Data I/O Made to scale, usually stateless		
Data Tier	Storage Query & storage optimization Performance (indexing, etc.)		

HTML

ASP .NET

WCF

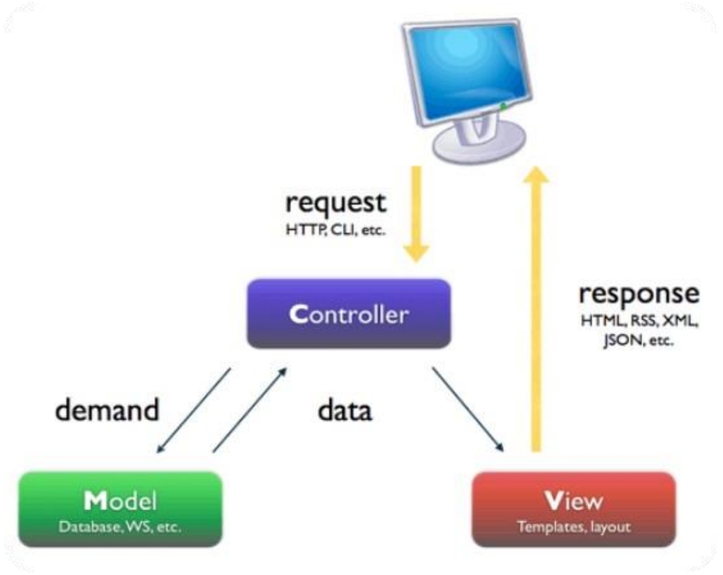
ORM

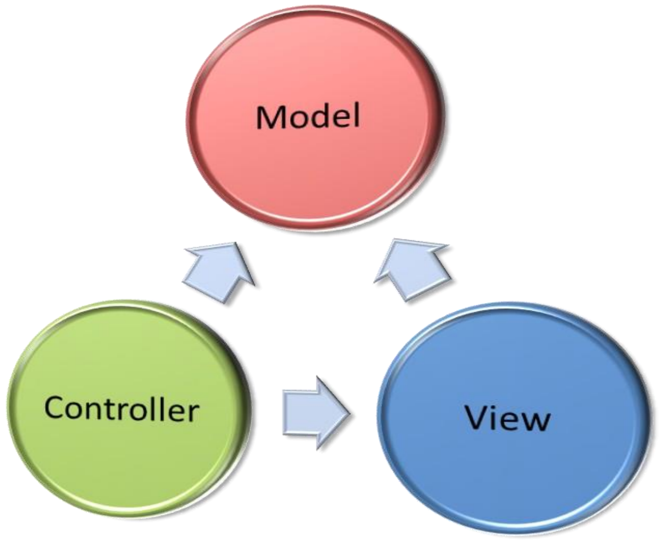
DB

UNIVERSITY of GREENWICH

Alliance with FPT Education

MVC (MODEL - VIEW - CONTROLLER)

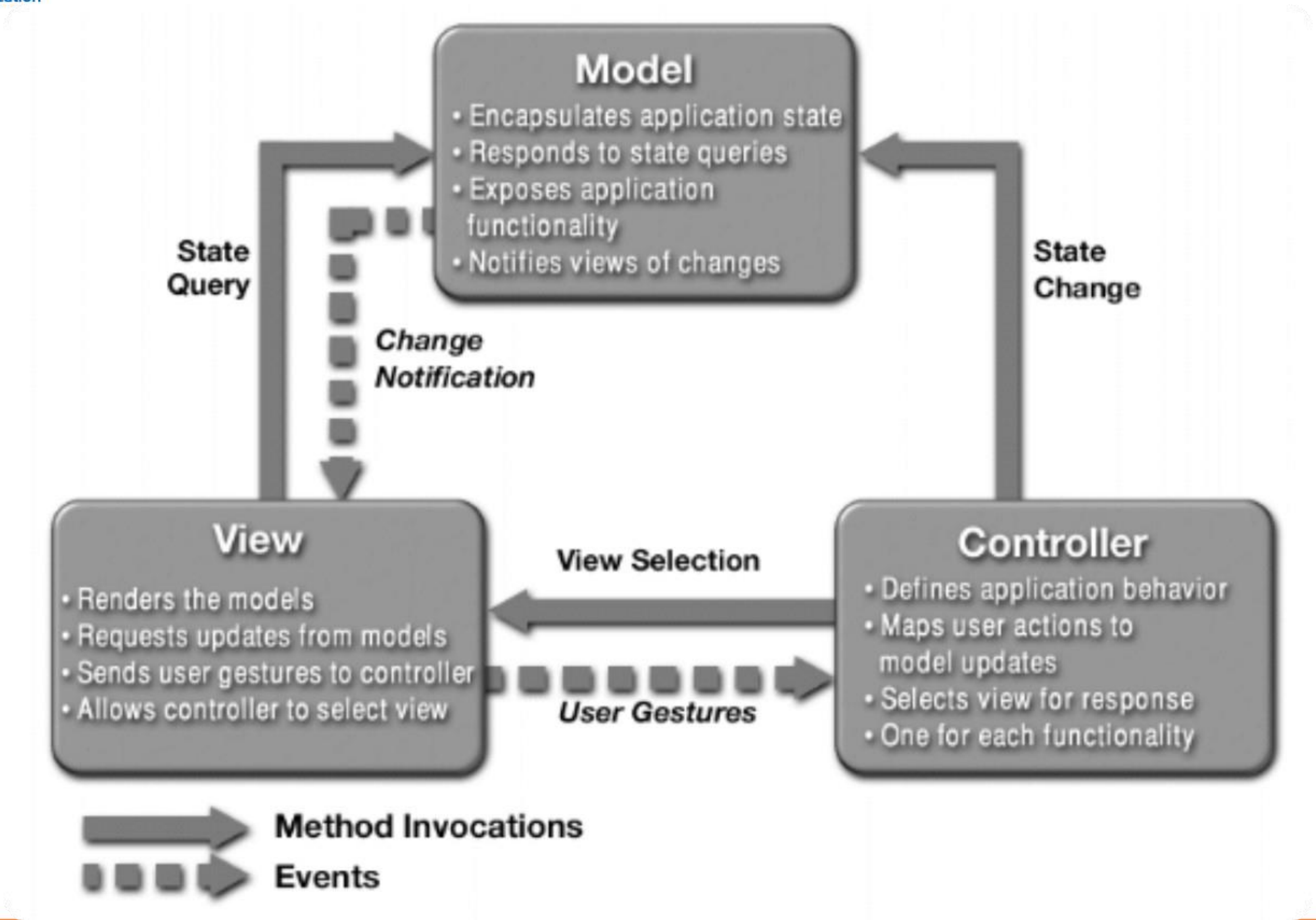




Model-View-Controller (MVC)

- Model-View-Controller (MVC) architecture
  - Separates the business logic from application data and presentation
- Model
  - Keeps the application state (data)
- View
  - Displays the data to the user (shows UI)
- Controller
  - Handles the interaction with the user

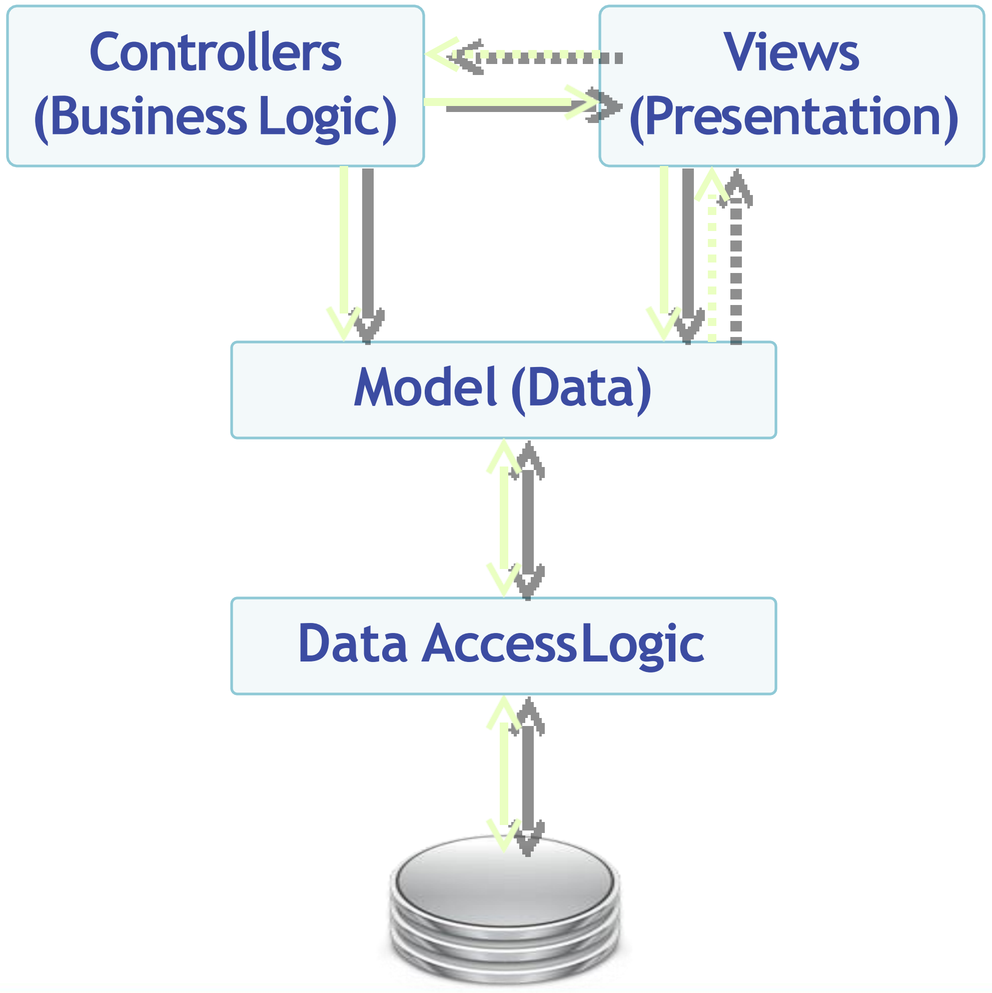
MVC Architecture Blueprint


Model
• Encapsulates application state
• Responds to state queries
• Exposes application functionality
• Notifies views of changes
State Query
State Change
Change Notification
View
• Renders the models
• Requests updates from models
• Sends user gestures to controller
• Allows controller to select view
View Selection
User Gestures
Controller
• Defines application behavior
• Maps user actions to model updates
• Selects view for response
• One for each functionality
Method Invocations
Events

MVC-Based Frameworks

- .NET
  - ASP.NET MVC, MonoRail
- Java
  - JavaServer Faces (JSF), Struts, Spring Web MVC, Tapestry, JBoss Seam, Swing
- PHP
  - CakePHP, Symfony, Zend, Joomla, Yii, Mojavi
- Python
  - Django, Zope Application Server, TurboGears
- Ruby on Rails

MVC and Multi-Tier Architecture



Software Design

- Detailed Design
  - – Describes the internal module structure
  - – Subcomponents, interfaces, process design, data design
- Object-Oriented Design
  - – Describes the classes, their responsibilities, relationships, dependencies, and interactions (usually in UML)
- Internal Class Design
  - – Methods, responsibilities, algorithms and interactions

Software Design Document (SDD)

- The Software Design Document (SDD)
  - Formal description of the architecture and design of the system
- SDD contains:
  - Architectural design
    - Modules and their interaction (diagram)
  - For each module
    - Process design (diagrams)
    - Data design (E/R diagram)
    - Interfaces design (class diagram)

UNIVERSITY of GREENWICH

Alliance with FPT Education

SOFTWARE CONSTRUCTION

Implementation, Unit Testing, Debugging, Integration

Software Construction

- During the software construction phase developers build the software
  - Sometimes called implementation phase
- Software construction includes:
  - Internal method design
  - Writing the source code
  - Writing the unit tests (optionally)
  - Testing and debugging
  - Integration

Writing the Code

- Coding is the process of writing the programming code (the source code)
  - The code strictly follows the design
  - Developers perform **internal method design** as part of coding
- The source code is the output of the software construction process
  - Written by developers
  - Can include unit tests

Testing the Code

- Testing checks whether the developed software conforms to the requirements
  - – Aims to identify defects (bugs)
- Developers test the code after writing it
  - – At least run it to see the results
  - – **Unit testing** works better
    - Units tests are repeated many times
- System testing is done by the QA engineers
  - – Unit testing is done by developers

Debugging

- Debugging aims to find the source of already identified defect and to fix it
  - Performed by developers
- Steps in debugging:
  - Find the defect in the code
    - Identify the source of the problem
    - Identify the exact place in the code causing it
  - Fix the defect
  - Test to check if the fix is working correctly

Software Integration

- **Integration** is putting all pieces together
  - – Compile, run and deploy the modules as a single system
  - – Test to identify defects
- Integration strategies
  - – Big bang, top-down and bottom-up
  - – Continuous integration (CI)

Coding != Software Engineering

- Inexperienced developers consider coding the core of development
  - In most projects coding is only 20% of the project activities!
  - The important decisions are taken during the requirements analysis and design
  - Documentation, testing, integration, maintenance, etc. are often disparaged
- Software engineering is not just coding!
  - **Programmer** != **software engineer**

SOFTWARE VERIFICATION AND TESTING

Software Verification

- What is software verification?
  - Checks whether the developed software conforms to the requirements
  - Performed by the Quality Assurance Engineers (QA engineers)
- Two approaches:
  - Formal reviews and inspections
  - Different kinds of testing
- Cannot certify absence of defects!
  - Can only decrease their rates

Software Testing

- Testing checks whether the developed software conforms to the requirements
- Testing aims to find defects (bugs)
  - Black-box and white-box tests
  - Unit tests, integration tests, system tests, acceptance tests
  - Stress tests, load tests, regression tests
  - Tester engineers can use automated test tools to record and execute tests

Software Testing Process

- Test planning
  - Establish test strategy and test plan
  - During requirements and design phases
- Test development
  - Test procedures, test scenarios, test cases, test scripts
- Test execution
- Test reporting
- Retesting the defects

Test Plan and Test Cases

- The test plan is a formal document that describes how tests will be performed
  - List of test activities to be performed to ensure meeting the requirements
  - Features to be tested, testing approach, schedule, acceptance criteria
- Test scenarios and test cases
  - Test scenarios – stories to be tested
  - Test cases – tests of single function

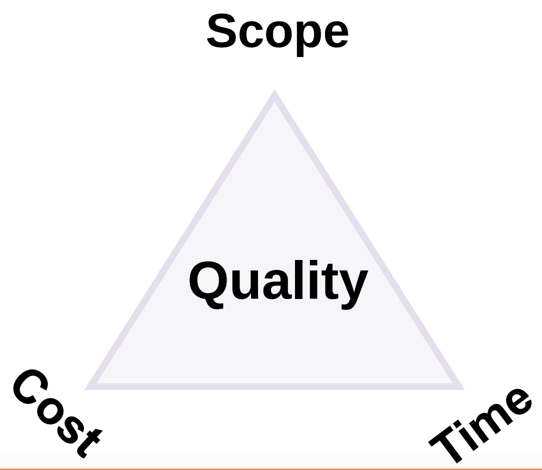
UNIVERSITY of GREENWICH

Alliance with FPT Education

SOFTWARE PROJECT MANAGEMENT

What is Project Management?

- Project management is the discipline of
  - Organizing and managing work and resources in order to successfully complete a project
- Successfully means within defined scope, quality, time and cost constraints
- Project constraints:



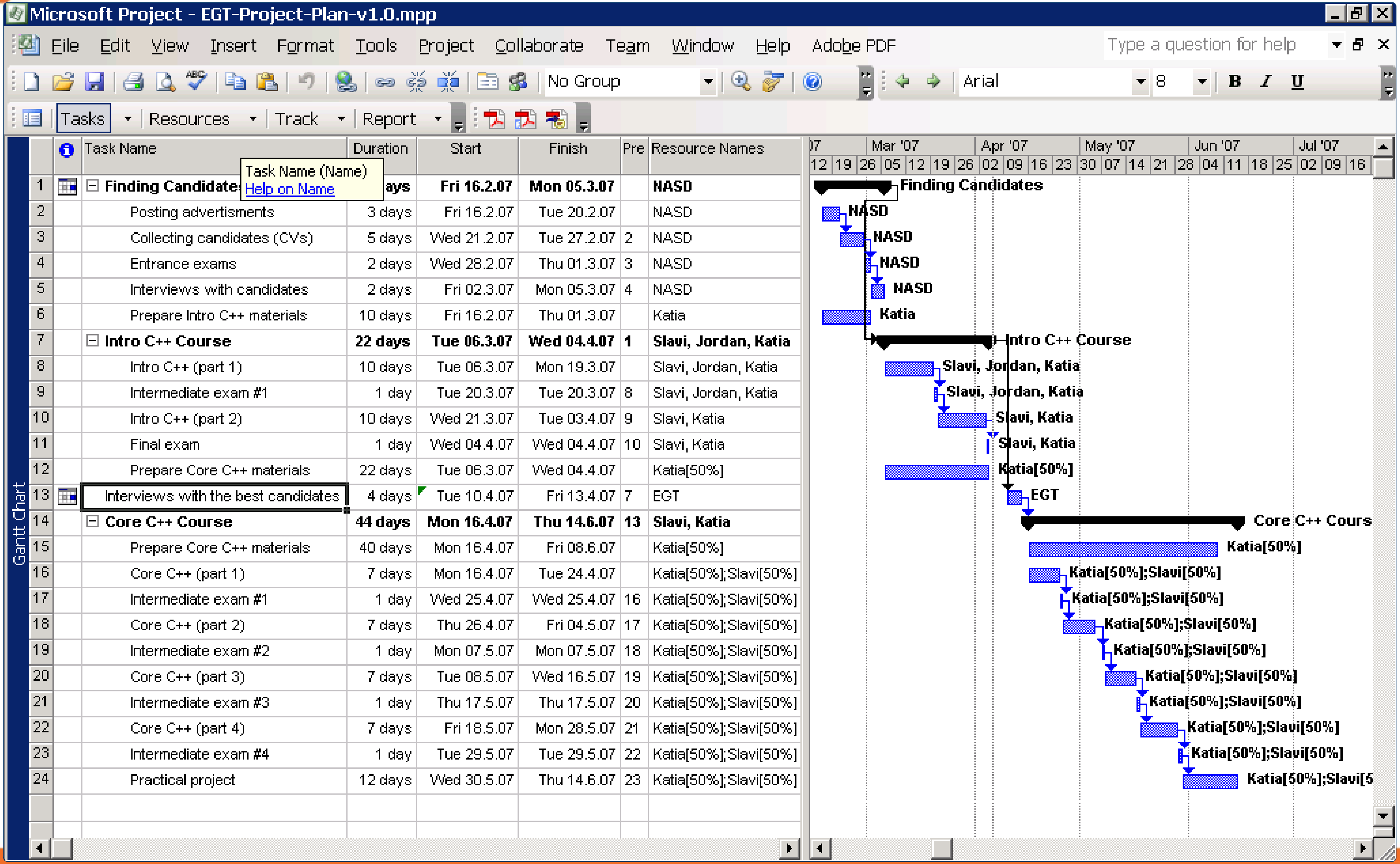
What is Software Project Management?

- Software project management
  - – Management discipline about planning, monitoring and controlling software projects
- **Project** planning
  - – Identify the scope, estimate the work involved, and create a project schedule
- **Project** monitoring **and** control
  - – Keep the team up to date on the project's progress and handle problems

What is Project Plan?

- The project plan is a document that describes how the work on the project will be organized
  - Contains tasks, resources, schedule, milestones, etc.
  - Tasks have start, end, assigned resources (team members), % complete, dependencies, nested tasks, cost, etc.
- Project management tools simplify creating and monitoring project plans

Project Plan – Example



Microsoft Project - EGT-Project-Plan-v1.0.mpp
File Edit View Insert Format Tools Project Collaborate Team Window Help Adobe PDF
Type a question for help
No Group
Arial
8
Tasks Resources Track Report
Gantt Chart
Task Name (Name)
Help on Name
'07 Mar '07 Apr '07 May '07 Jun '07 Jul '07
12 19 26 05 12 19 26 02 09 16 23 30 07 14 21 28 04 11 18 25 02 09 16
Finding Candidates
Intro C++ Course
Core C++ Cours

	Task Name	Duration	Start	Finish	Pre	Resource Names
1	**Finding Candidate**	**ays**	**Fri 16.2.07**	**Mon 05.3.07**		**NASD**
2	Posting advertisments	3 days	Fri 16.2.07	Tue 20.2.07		NASD
3	Collecting candidates (CVs)	5 days	Wed 21.2.07	Tue 27.2.07	2	NASD
4	Entrance exams	2 days	Wed 28.2.07	Thu 01.3.07	3	NASD
5	Interviews with candidates	2 days	Fri 02.3.07	Mon 05.3.07	4	NASD
6	Prepare Intro C++ materials	10 days	Fri 16.2.07	Thu 01.3.07		Katia
7	**Intro C++ Course**	**22 days**	**Tue 06.3.07**	**Wed 04.4.07**	**1**	**Slavi, Jordan, Katia**
8	Intro C++ (part 1)	10 days	Tue 06.3.07	Mon 19.3.07		Slavi, Jordan, Katia
9	Intermediate exam #1	1 day	Tue 20.3.07	Tue 20.3.07	8	Slavi, Jordan, Katia
10	Intro C++ (part 2)	10 days	Wed 21.3.07	Tue 03.4.07	9	Slavi, Katia
11	Final exam	1 day	Wed 04.4.07	Wed 04.4.07	10	Slavi, Katia
12	Prepare Core C++ materials	22 days	Tue 06.3.07	Wed 04.4.07		Katia[50%]
13	Interviews with the best candidates	4 days	Tue 10.4.07	Fri 13.4.07	7	EGT
14	**Core C++ Course**	**44 days**	**Mon 16.4.07**	**Thu 14.6.07**	**13**	**Slavi, Katia**
15	Prepare Core C++ materials	40 days	Mon 16.4.07	Fri 08.6.07		Katia[50%]
16	Core C++ (part 1)	7 days	Mon 16.4.07	Tue 24.4.07		Katia[50%];Slavi[50%]
17	Intermediate exam #1	1 day	Wed 25.4.07	Wed 25.4.07	16	Katia[50%];Slavi[50%]
18	Core C++ (part 2)	7 days	Thu 26.4.07	Fri 04.5.07	17	Katia[50%];Slavi[50%]
19	Intermediate exam #2	1 day	Mon 07.5.07	Mon 07.5.07	18	Katia[50%];Slavi[50%]
20	Core C++ (part 3)	7 days	Tue 08.5.07	Wed 16.5.07	19	Katia[50%];Slavi[50%]
21	Intermediate exam #3	1 day	Thu 17.5.07	Thu 17.5.07	20	Katia[50%];Slavi[50%]
22	Core C++ (part 4)	7 days	Fri 18.5.07	Mon 28.5.07	21	Katia[50%];Slavi[50%]
23	Intermediate exam #4	1 day	Tue 29.5.07	Tue 29.5.07	22	Katia[50%];Slavi[50%]
24	Practical project	12 days	Wed 30.5.07	Thu 14.6.07	23	Katia[50%];Slavi[50%]

SOFTWARE SOLUTION STACKS

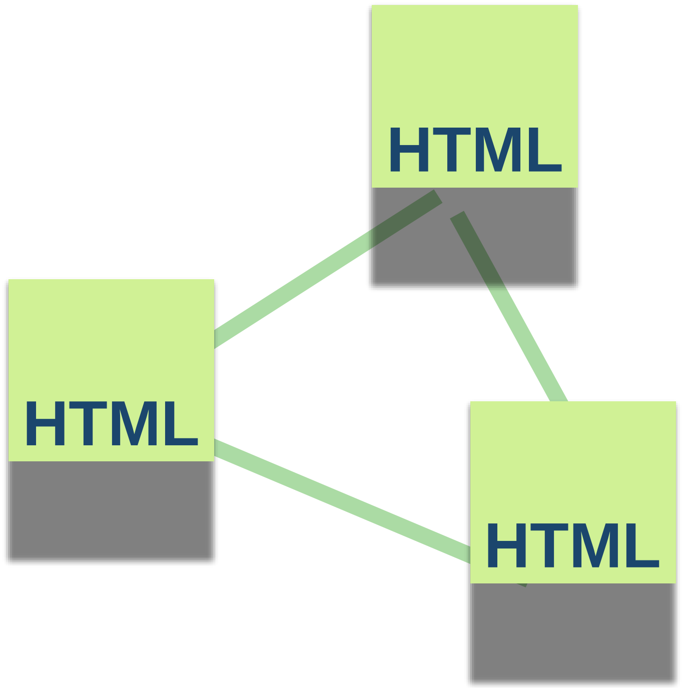
Web 1.0

Static pages of hyperlinked information

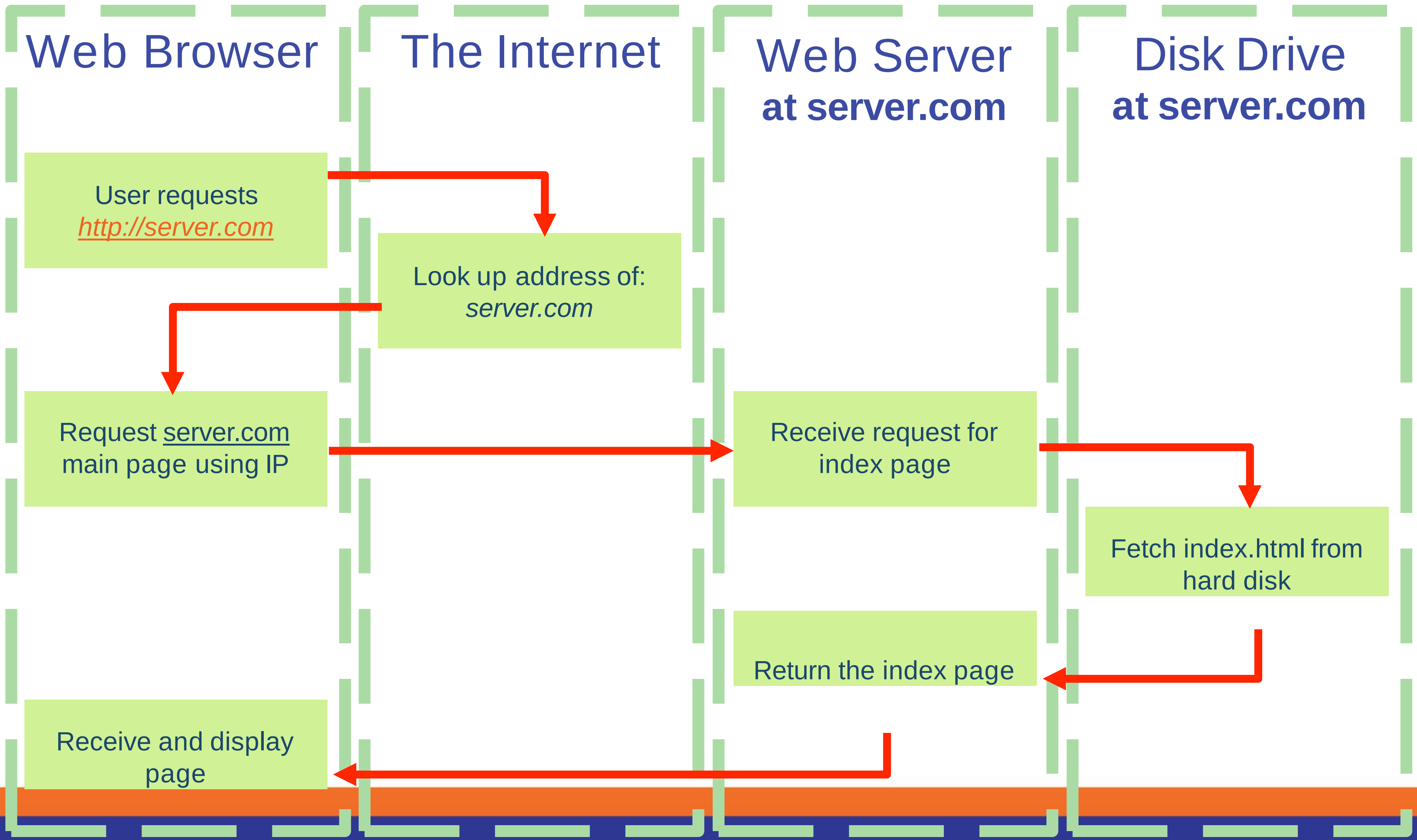
Information revolution

created using HTML

<mark-up language>



Client/Server Request/Response

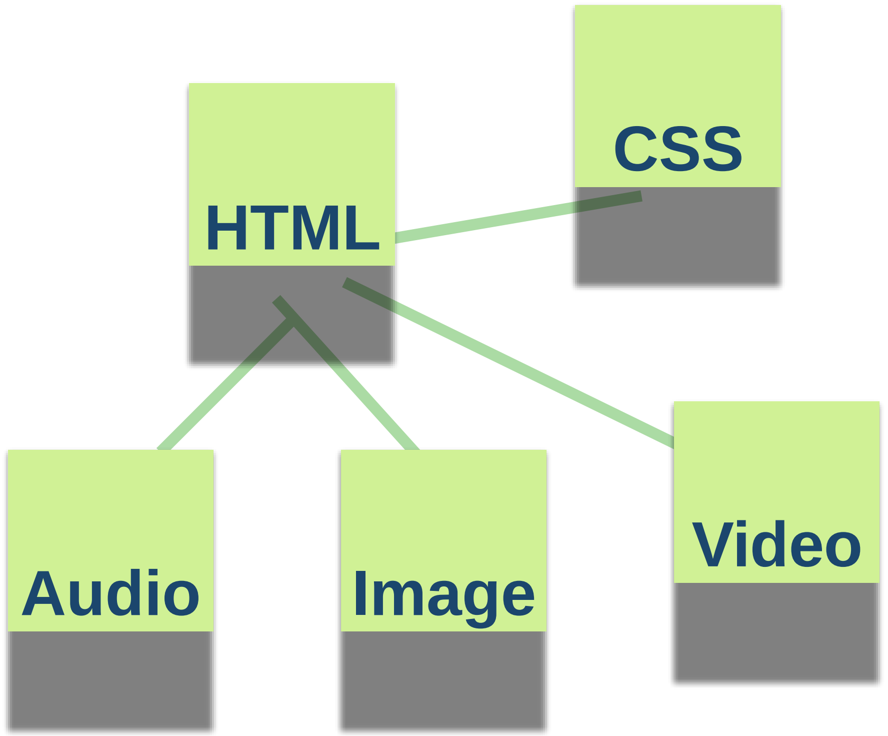


Web Browser
The Internet
Web Server at server.com
Disk Drive at server.com
User requests http://server.com
Look up address of: server.com
Request server.com main page using IP
Receive request for index page
Fetch index.html from hard disk
Return the index page
Receive and display page

Moving Forward

Look and feel improved with style sheets

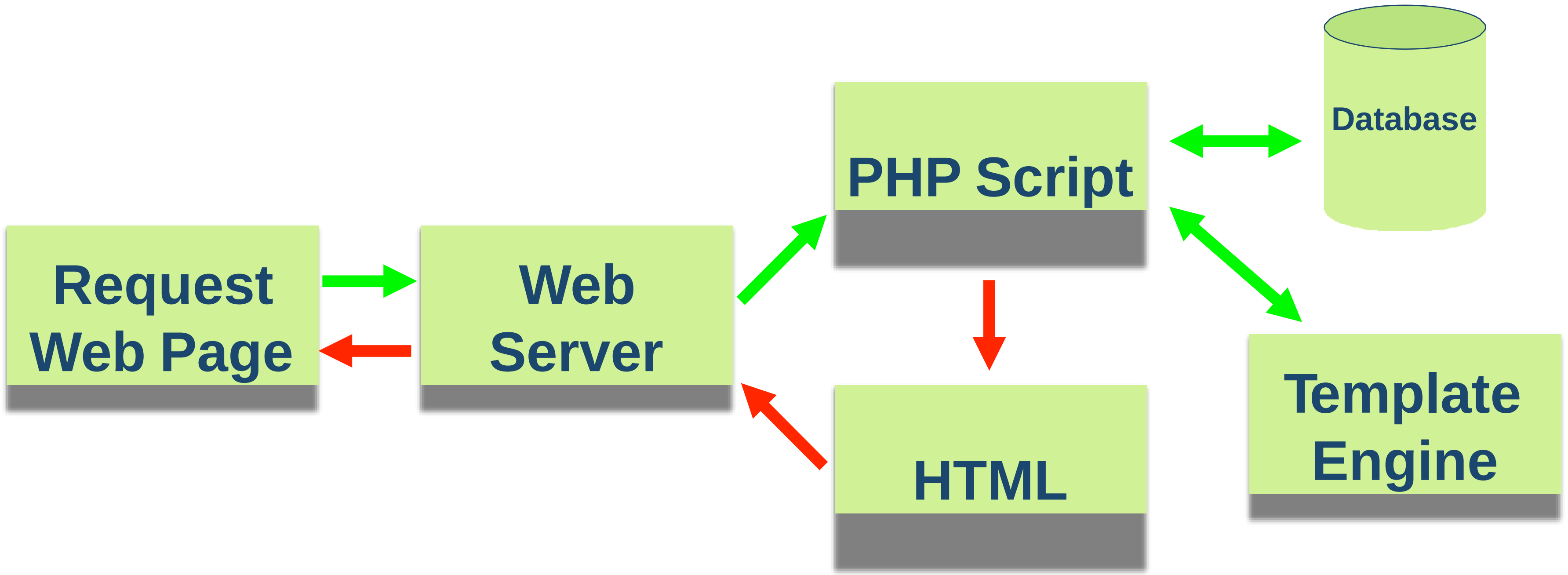
- presentation and style could be separated from the information structure and content
- Cascading Style Sheet (CSS)
- Mixed Media



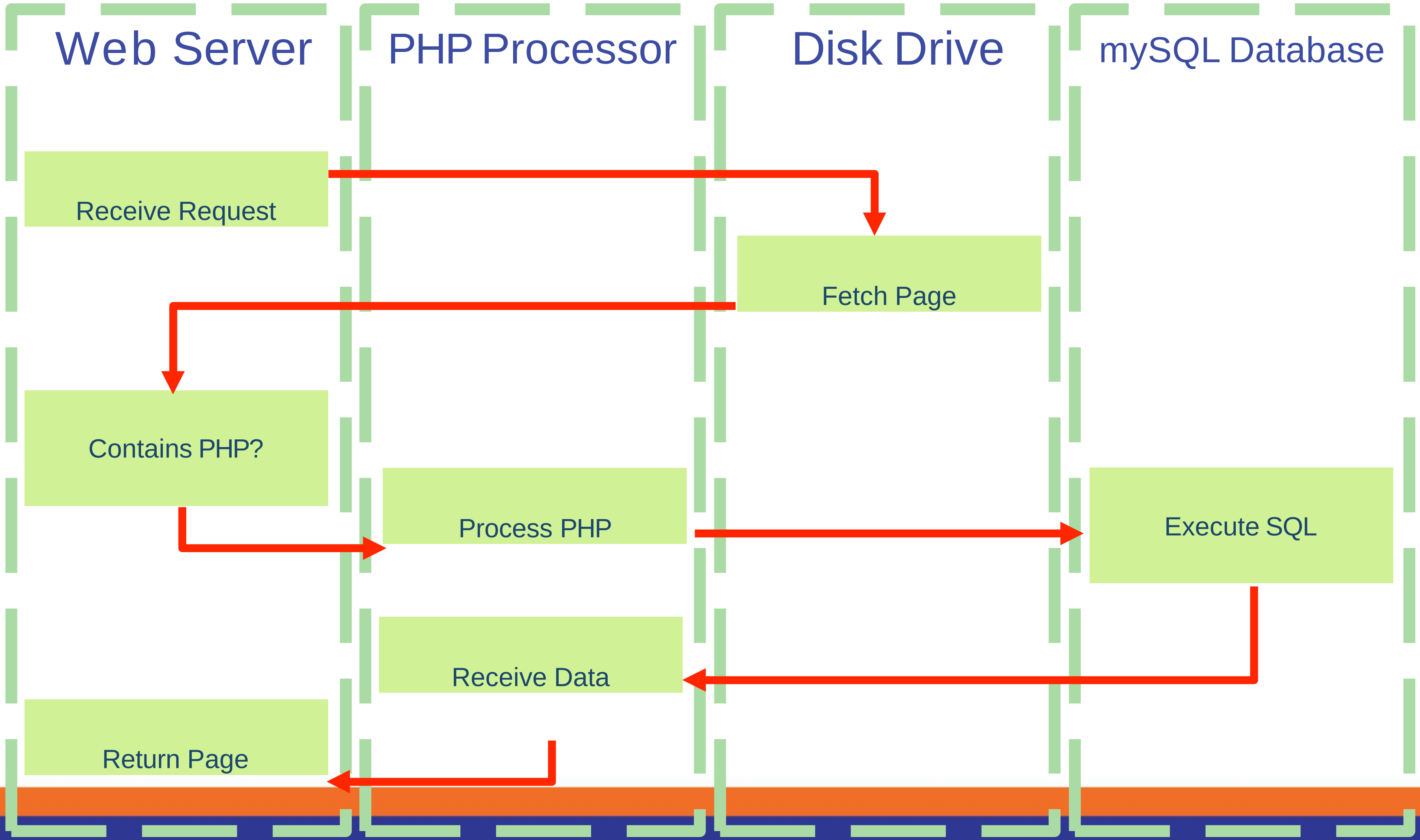
Dynamic Content

Server side dynamic content generation

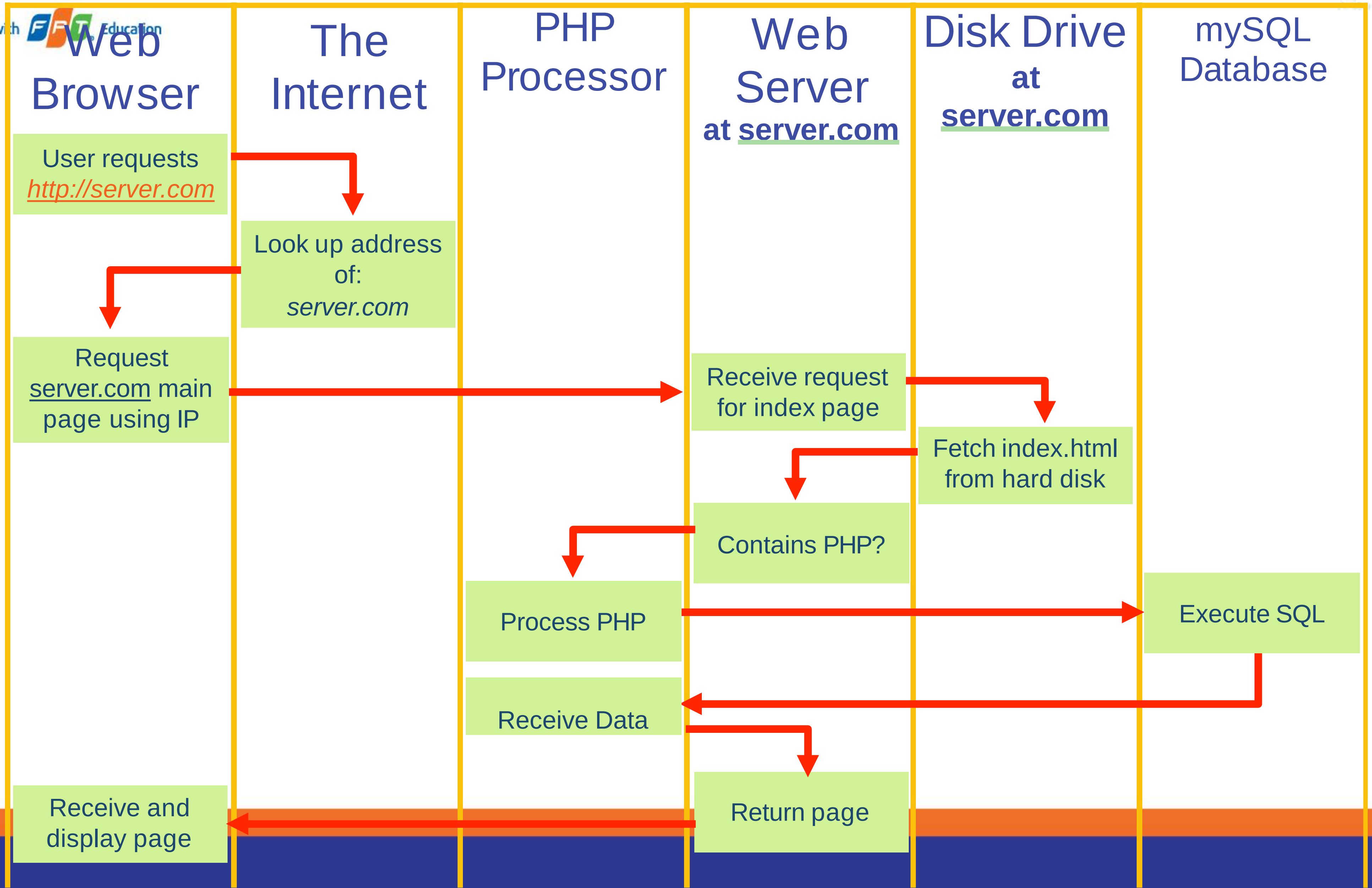
- Mature Web Server Stacks (e.g. LAMP)
- Sites became web services (data not pages)



Dynamic request/response



Client/Server Request/Response

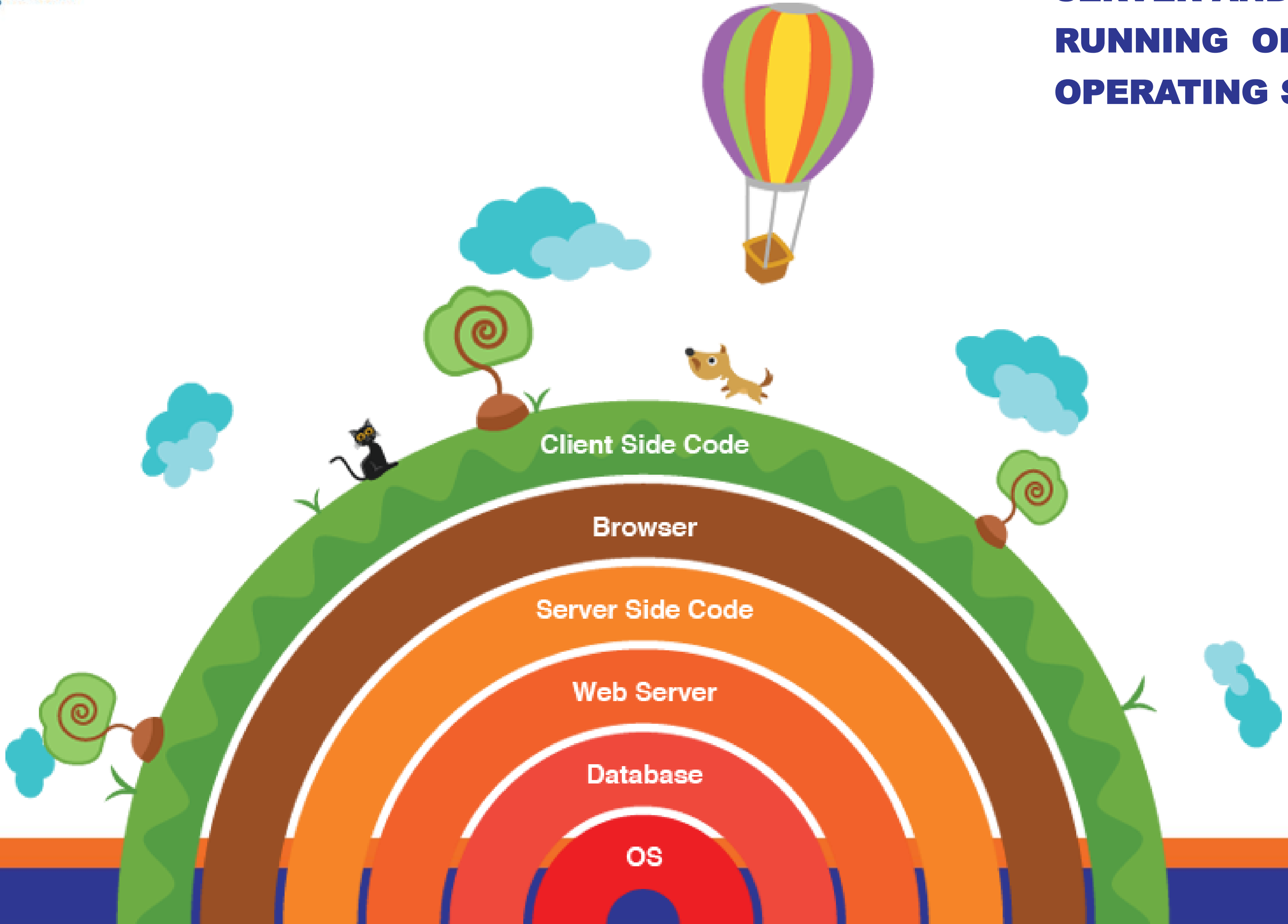


Why are web apps important?

- Nobody wants simple web pages anymore
- Desktop software for common tasks may be coming extinct
- Offsite storage as standard
- Collaboration easily added
- No software installation
- No software updates

A SOFTWARE (OR SOLUTION) STACK: A SET OF COMPONENTS REQUIRED TO MAKE A COMPLETE PLATFORM

WEB STACK: SERVER SIDE CODE UTILIZES WEB SERVER AND DATABASE ALL RUNNING ON AN OPERATING SYSTEM

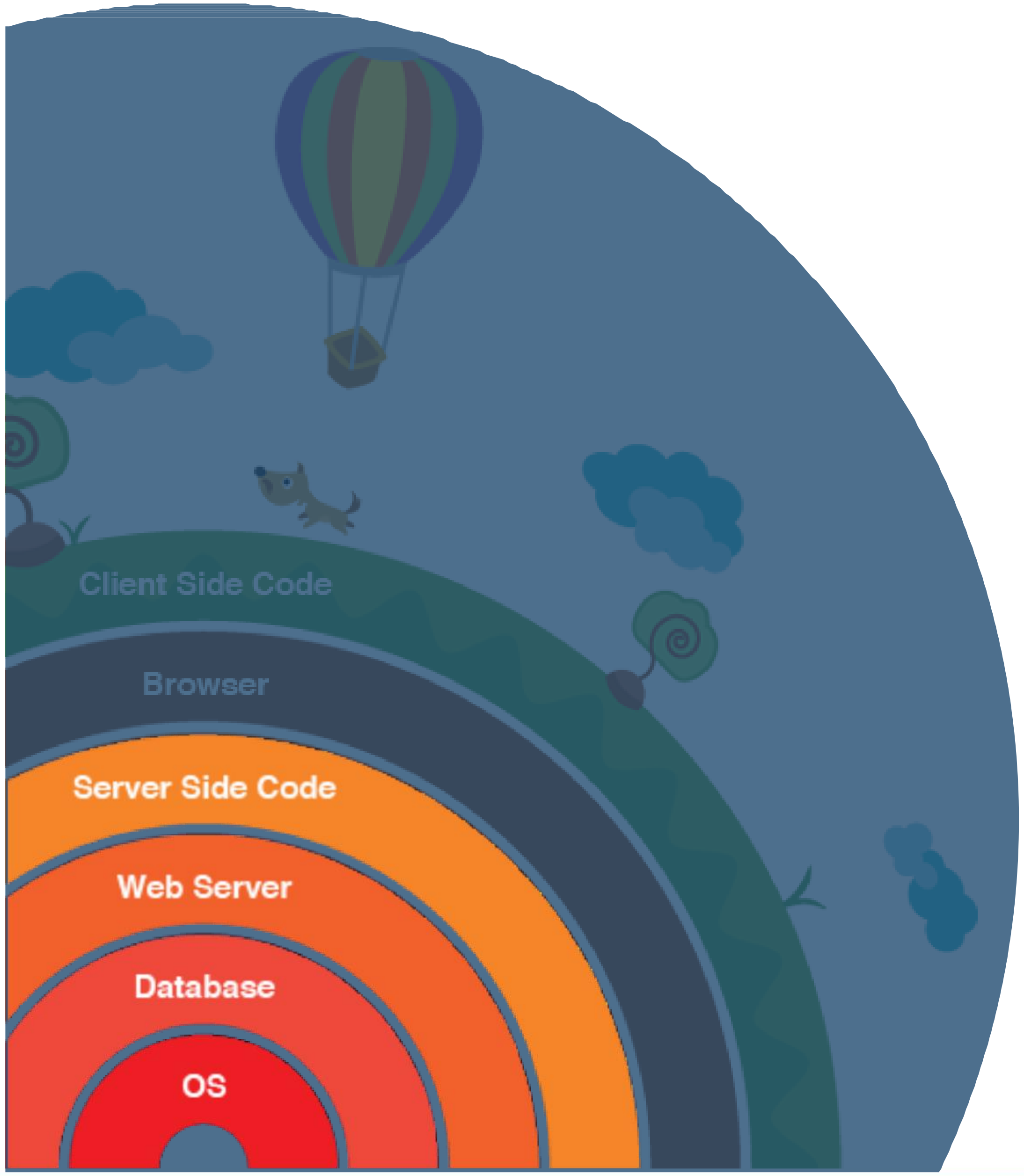


Common Web Stacks

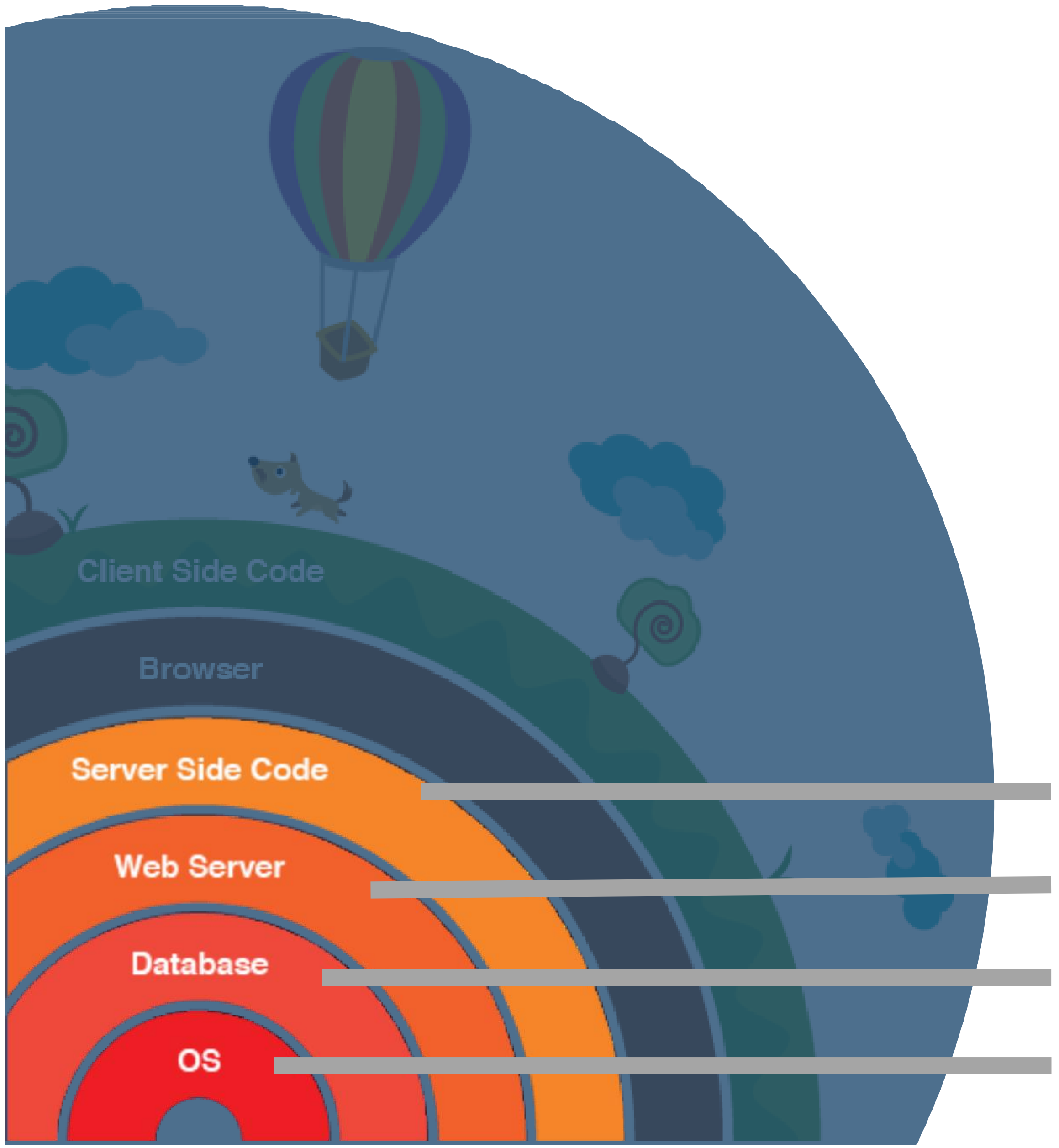
LAMP	WAMP
Linux	**W**indows
Apache	**A**pache
MySQL	**M**ySQL
PHP	**P**HP

Common Web Stacks

LEMP	WIMP	WINS
Linux	Windows	Windows
EnginX	ISS	ISS
MySQL	MySQL	.Net
PHP	PHP	Sql Server

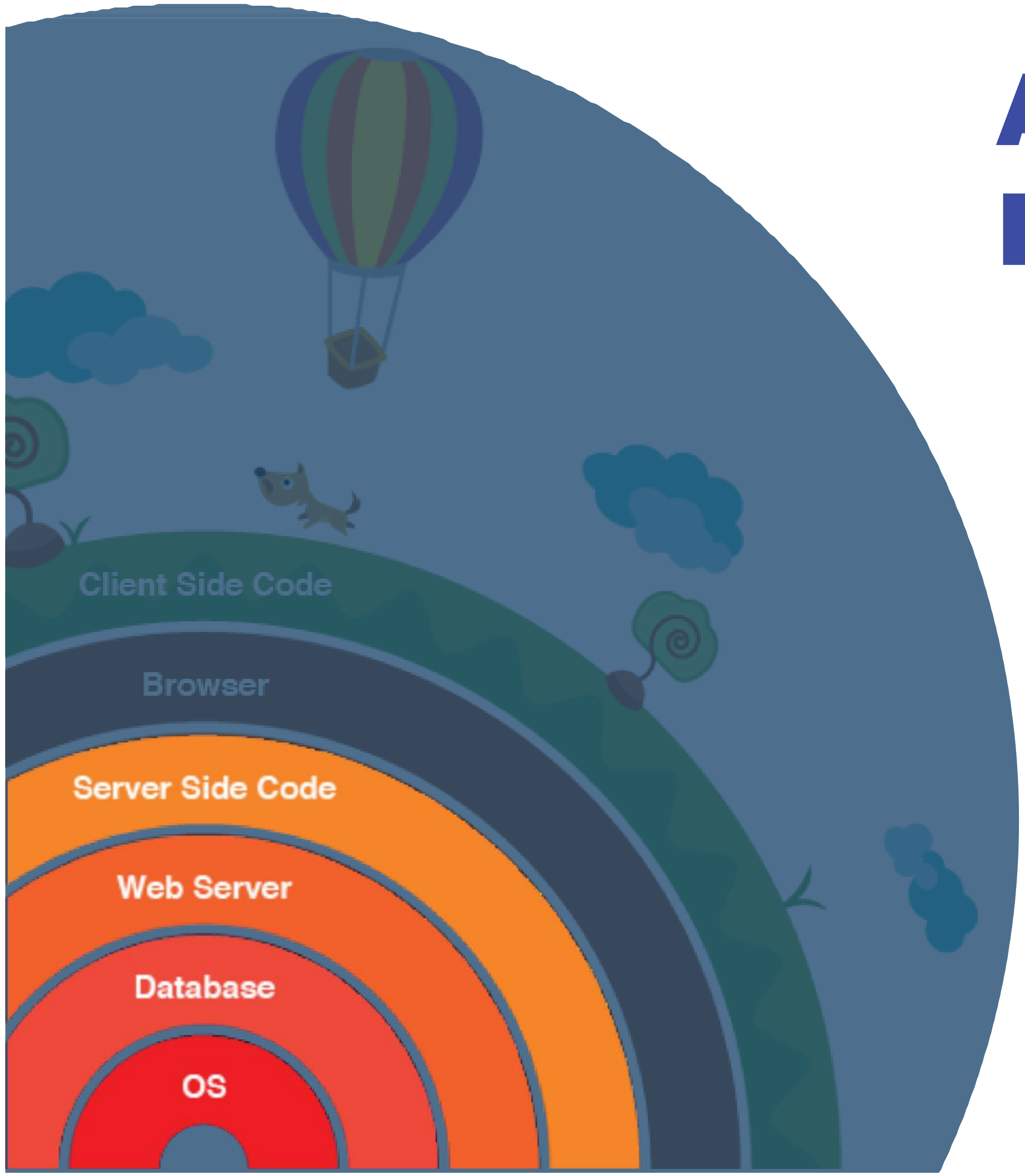


Linux
Apache
MySQL
PHP



PHP
Apache
MySQL
Linux

Alternatives Exist...



	LAMP	MEAN
OS	Linux	no need to mention, node is cross platform
Database	mySQL	mongoDB
HTTP Server	apache	node is its own server, Express makes it easier
Serverside Code	PHP	nodeJS applications, may link to Angular clientside

html	How to structure a webpage
css	How to style a webpage
php	How to make a web page interactive
mySQL	How to load and save information from a webpage

Can be created and viewed on your own machine quite easily

html	How to structure a webpage
css	How to style a webpage
php	How to make a web page interactive
mySQL	How to load and save information from a web page

Needs specialised server in order to work