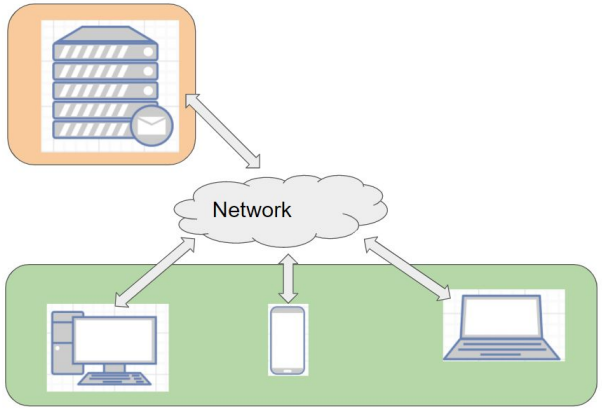
Modern Application Development - I

Revision session (Week 1-6)

Basics of Web

- What is an ‘App’ in general ?
- Categories of Apps and web apps as the main focus of the course
- Web Apps: Heavily network dependent apps that work across OS and devices
- Components of Web App:
  - Storage
  - Computation
  - Presentation
- Application platforms: Desktop, Mobile, Web-based, Embedded
- Web Architectures:
  - Client-server
  - Peer to Peer
- Software Architecture Pattern: MVC (mainly emphasized for this course)



A Client server web Architecture

The Web

- Why is Web the platform of choice for this course
- Historical background of Networks:
  - Telephone Networks
  - Packet switched network
- Historical background of Web:
  - Protocol and ‘Inter’-Network
  - Internet Protocol (IP)
  - Transmission Control Protocol
  - Domain Names
- World Wide Web and its evolution
- Understanding HTTP and simplest server with curl
- Performance parameters of web :
  - Latency
  - Response size

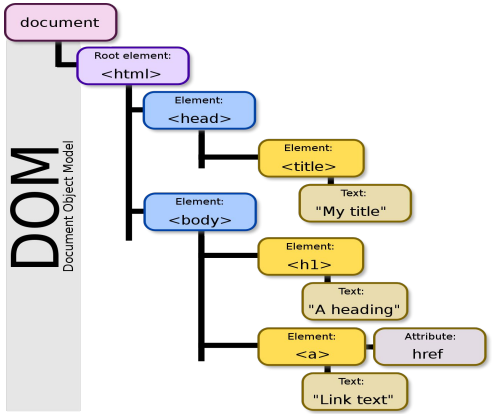
Markup

- **Information Representation:** Interpretation and representation of text using encoding like: ASCII, UTF-8, Unicode etc., Efficiency of encoding
- **Raw Data vs. Semantics:** Mark-up and its types, HTML as an application of SGML
- **Logical structure vs. Styling:** Mark-up vs. style, separation of style, Document Object Model (DOM)
- **HTML5 and CSS:** CSS and its types, Concept of Responsive design

Alternatives to styling?

- Frameworks like bootstrap

JavaScript



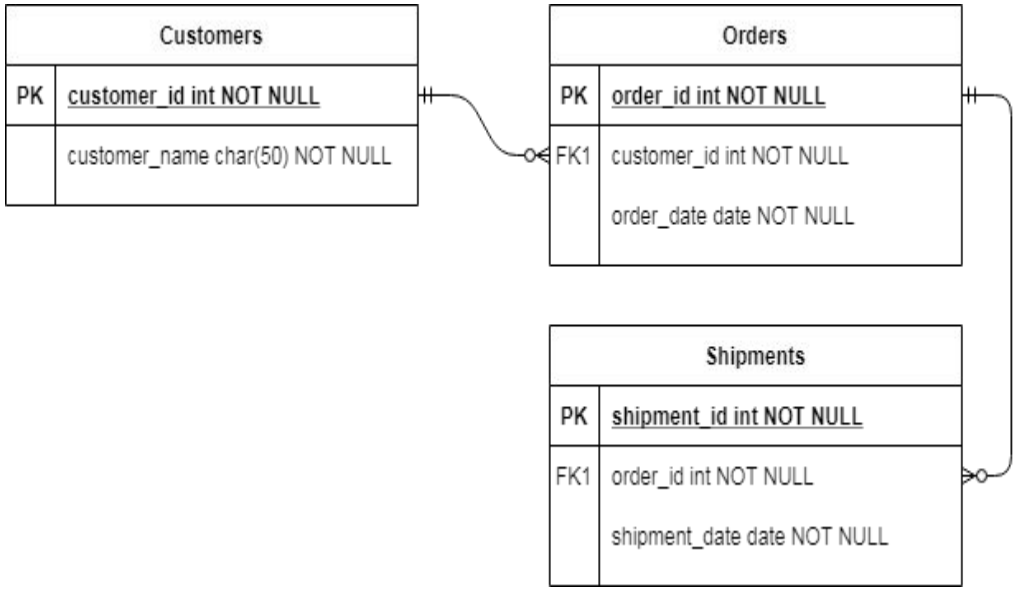
Source: B. Eriksson, Wikipedia

Views

- **Design patterns:** A glimpse of MVC design pattern using a running example of student grade book
- **View** - The ‘V’ in MVC design pattern: User interface design and key differences between user interface and user interaction
- **Types of view:** Fully static, partly dynamic, mostly dynamic
- **User Interface Design:** Goals, Aesthetics, Accessibility
- Guidelines and heuristics defining a systematic process and a light on general principles
- **Tools:**
    - Wireframes
    - HTML generation
    - Templates
- **Accessibility principles:** Perceivable, Operable, Understandable, Robust

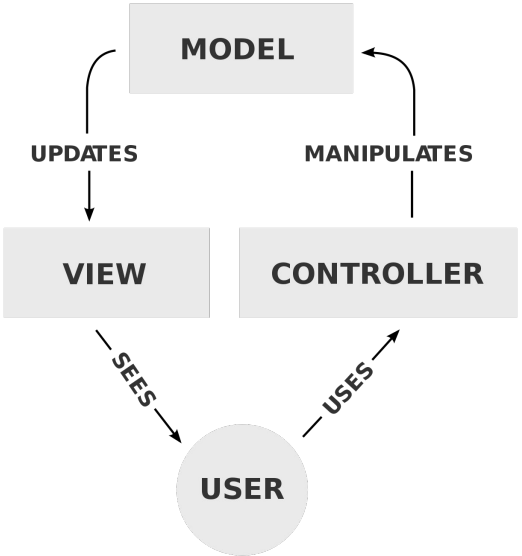
Models

- Persistent storage: Needs and requirements
- Relationships: Between two entities of a table or two separate tables
- Mechanisms for persistent storage:
  - In-memory data structures
  - Relational Databases (SQL)
  - Unstructured Databases (No-SQL)
- Relations: E-R Diagrams
  - one-to-many
  - many-to-many
- Basics of SQL



Controllers

- Origins of design pattern and understanding MVC in detail
- **Controllers:** A link between user and the system
- General concept of taking action in response to the user input
- Concepts of request and response
- **CRUD**
- **Routing:** Mapping URLs to action
- Introduction to **flask** framework



By RegisFrey - Own work, Public Domain, Wikipedia

REST and APIs

- Introduction to **Distributed Software Architecture**
- **REST:** Representational State Transfer: Providing guidelines/constraints by taking into consideration the limitations of Web
- Constraints defining REST:
    - Client-Server Architecture
    - Stateless
    - Layered system
    - Cacheability
    - Uniform Interface
    - Code on demand (Optional)
- HTTP Methods

REST and APIs

- Idempotent Operations
- Encoding Data for transferring complex data types over text based format: JSON
- API data transfer format: Input: text (HTTP)– Output: Complex data types (JSON)
- YAML: for documentation and configuration of API
- REST APIs: Understanding the use and documentation with the help of prominent examples, (CoWin public APIs)
- OpenAPI Specification: Open - public documentation better to identify problems
- Understanding Swagger for OpenAPI documentation