Software Architecture Document

Knowledge Base Application

Table of Contents

1. Introduction

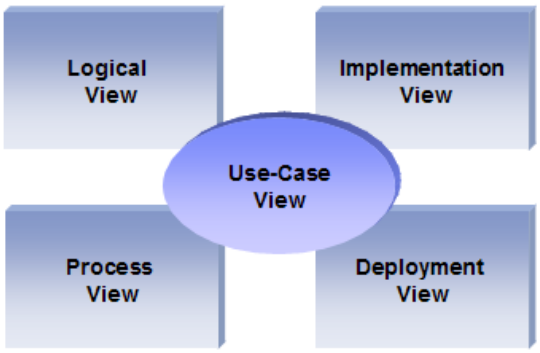
This document provides a high level overview and explains the whole process of Knowledge Base. It explains how an online user will be able to Access Data and gain knowledge on the required component and it includes the underlying architectural process.

The document provides a high-level description of the goals of the architecture, the use cases support by the system and architectural styles and components that have been selected to best achieve the use cases. This framework then allows for the development of the design criteria and documents that define the technical and domain standards in detail.

1.1 Purpose

The Software Architecture Document (SAD) provides a comprehensive architectural overview of Knowledge Base offered by RailsFactory Team. It presents a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

In order to depict the software as accurately as possible, the structure of this document is based on the "4+1" model view of architecture.



The "4+1" View Model allows various stakeholders to find what they need in the software architecture.

1.2 Scope

The scope of this SAD is to depict the architecture of the Knowledge Base online application created by the company RailsFactory.

This document describes the aspects of Knowledge Base design that are considered to be architecturally significant; that is, those elements and behaviors that are most fundamental for guiding the construction of Knowledge Base and for understanding this project as a whole. Stakeholders who require a technical understanding of Knowledge Base are encouraged to start by

reading this document, then reviewing the Knowledge Base UML model, and then by reviewing the source code.

1.3 Definitions, Acronyms and Abbreviations

- **KBA – Knowledge Base Application**
- **ROR -** Hypertext Processor scripting language
- **Mongo** – database
- **HTTP** – Hypertext Transfer Protocol
- **WWW** – World Wide Web
- **Apache** – Web Server
- **RF -** RailsFactory
- **SAD -** Software Architecture Document
- **RUP -** Rational Unified Process
- **UML** – Unified Modeling Language

1.4 Overview

In order to fully document all the aspects of the architecture, the Software Architecture Document contains the following subsections.

Section 2: describes the use of each view

Section 3: describes the architectural constraints of the system

Section 4: describes the functional requirements with a significant impact on the architecture

Section 5: describes the most important use-case realization.

Section 6: describes application's process flow.

Section 7: describes how the system will be deployed.

Section 8: describes Data Model (Entity Relationship Diagram).

2. Architectural Representation

This document details the architecture using the views defined in the "4+1" model, but using the RUP naming convention. The views used to document the Agency Access application are:

Logical view

Audience: Designers.

Area: Functional Requirements: describes the design's object model. Also describes the most important use-case realizations and business requirements of the system.

Process view

Audience: Integrators.

Area: Process Flow: describes the design's concurrency and synchronization aspects.

Implementation view

Audience: Programmers.

Area: Software components: describes the layers and subsystems of the application.

Deployment view

Audience: Deployment managers.

Area: Topology: describes the mapping of the software onto the hardware and shows the system's distributed aspects. Describes potential deployment structures, by including known and anticipated deployment scenarios in the architecture we allow the implementers to make certain assumptions on network performance, system interaction and so forth.

Use Case view

Audience: all the stakeholders of the system, including the end-users.

Area: describes the set of scenarios and/or use cases that represent some significant, central functionality of the system. Describes the actors and use cases for the system, this view presents the needs of the user and is elaborated further at the design level to describe discrete flows and constraints in more detail. This domain vocabulary is independent of any processing model or representational syntax (i.e. XML).

Data view

Audience: Database administrators

Area: Persistence: describes the architecturally significant persistent elements in the data model

3. Architectural Goals and Constraints (non-functional requirements)

This section describes the software requirements and objectives that have some significant impact on the architecture

3.1 Technical Platform

Server side

KBA will be hosted on one of Apache web servers and connecting to Mongo Database servers. Web server will accept all requests from the clients and forward them to the specific KBA server hosting location. All communication with client has to comply with public HTTP, TCP/IP communication protocol standards.

Client Side

Clients will be able to access KBA only on WWW. Clients are requiring using a modern web browser such as Mozilla Firefox, Internet Explorer. The target operating systems are Windows XP and Linux.

Technical Architecture

Framework	Rails
Language	Ruby
Client Scripting	Javascript (jQuery)
Database	Mongo
File Upload	Carrierwave
Web Server	Apache/Mongrel

3.2 Security

The HTTP protocol will be used to facilitate communications between the client and server. Although basic password authentication and role based security mechanisms will be used to protect KBA from unauthorized access, functionality such as money transactions are assumed to be sufficiently protected. The system must be secured, so that a customer can make online payments.

The application must implement basic security behaviors:

- Authentication: Login using a user name and a password
- Authorization: as per our software specifications, web administrator will have enhanced privileges to perform tasks that general user would not be authorized.

3.3 Reliability/Availability (failover)

The scalability and reliability of the system is a key requirement as it pertain the nature of this system.

The server space availability has to be responsive to the increasing data and traffic volumes. The candidate architecture must ensure failover capabilities. Reliability/Availability will be addressed through the server platform.

In the event of the server failing due to an error with one of these applications might result in KBA becoming temporarily unavailable. The expectation for performance is that if any operation will take more than 10 seconds to execute, it should execute as a background operation, freeing the user to perform other tasks. The user should be made aware of all operations with visual feedback.

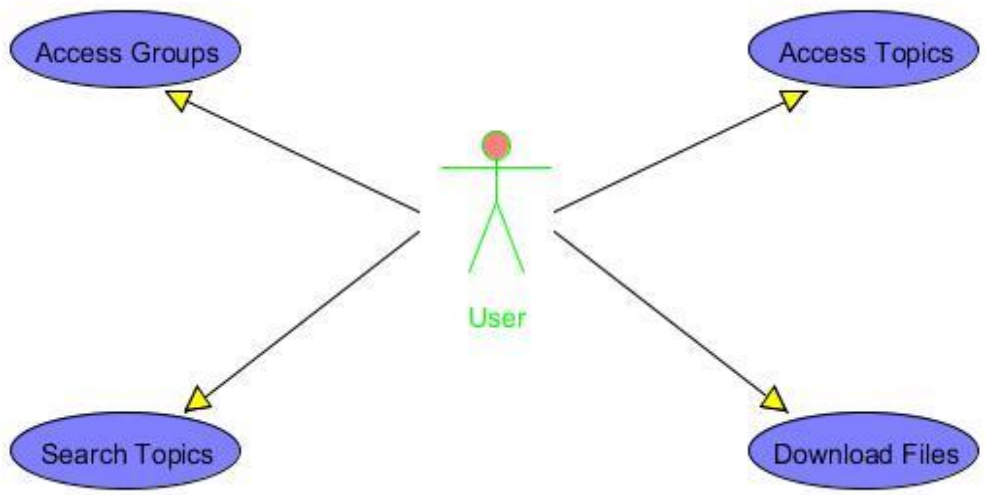
3.4 Performance

Any process must be under 10 seconds on Apache Web server.

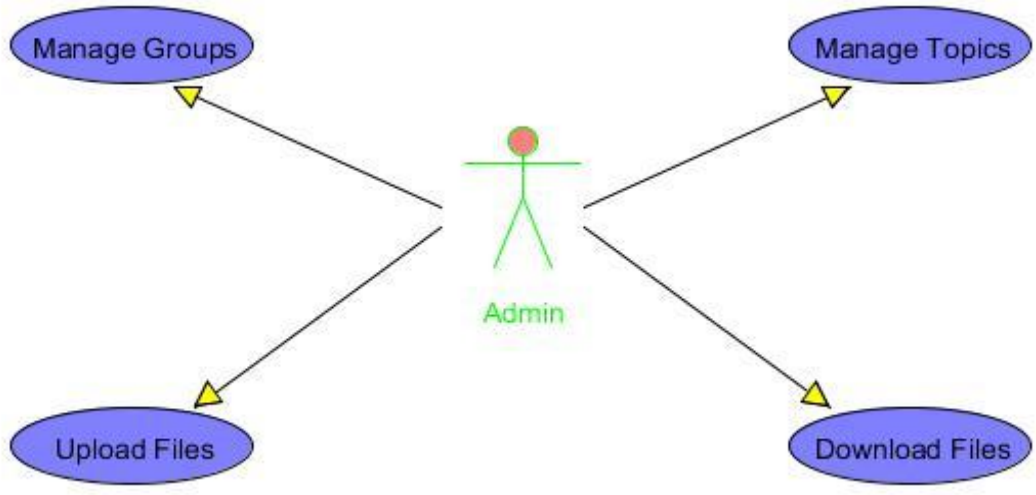
4. Use-Case View

This is a list of use cases from the use-case model that represent significant, central functionality of the final system. The use-cases with a significant impact on the architecture are [UC-Doc]:

User - Use Case View



Admin - Use Case view

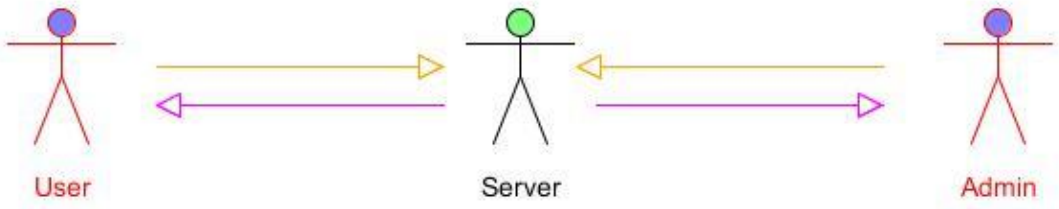


4.1 KBA Actors

As described in the actors' correspondence diagram below, web user could be one of three types:

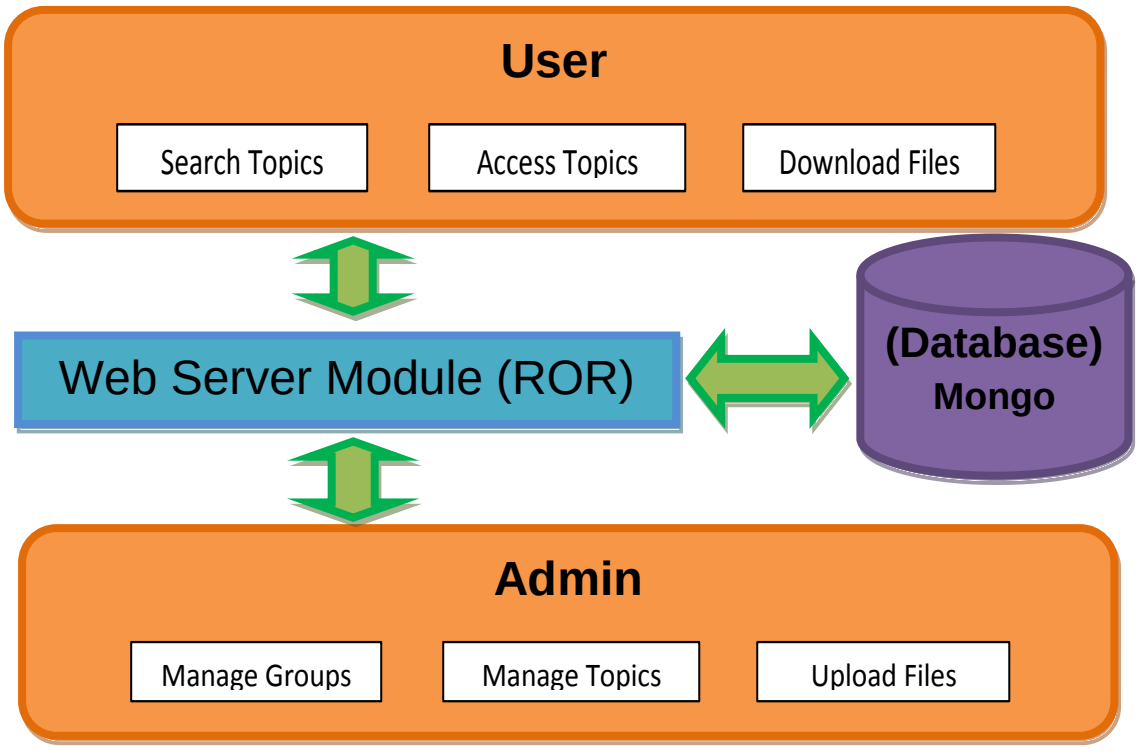
1. **Admin** has enhanced privileges to manage Groups, Topics, Upload Files.
2. **User** are the primary actors, they can access Search & Access Topics, Download Files.

System – Apache Web Server is the third type of actor and is the system itself. It handles all the physical and logical process of the software.



4.2 Use-Case Realizations

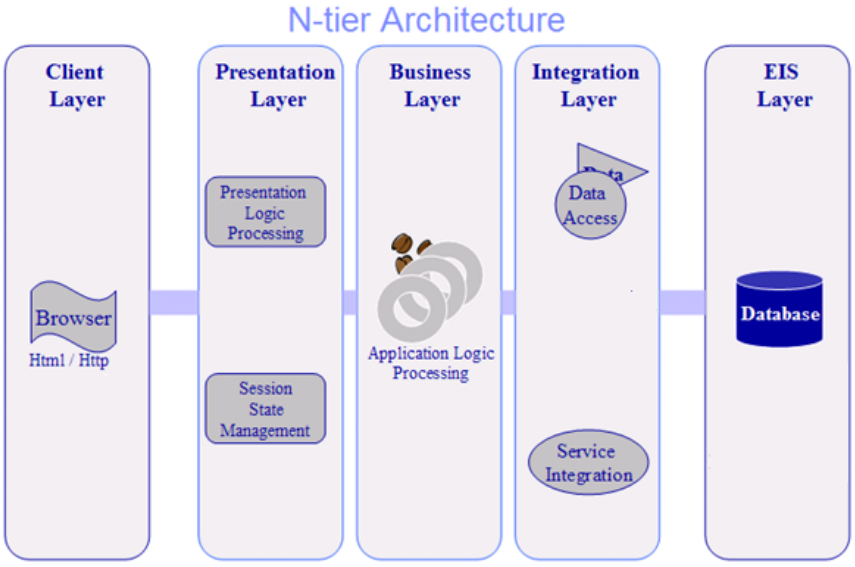
Use case functionality diagram below describes how design elements provide the functionalities identified in the significant use-cases. Use cases are displayed as functionalities for the system. Functionality may enclose more than one use-case. It is assumed that sign–in is default functionality and it has to be implemented before any further functionality will be enabled.



5. Logical View

5.1 Overview

KBA is divided into layers based on the N-tier architecture.



The layering model of the KBA is based on a responsibility layering strategy that associates each layer with a particular responsibility.

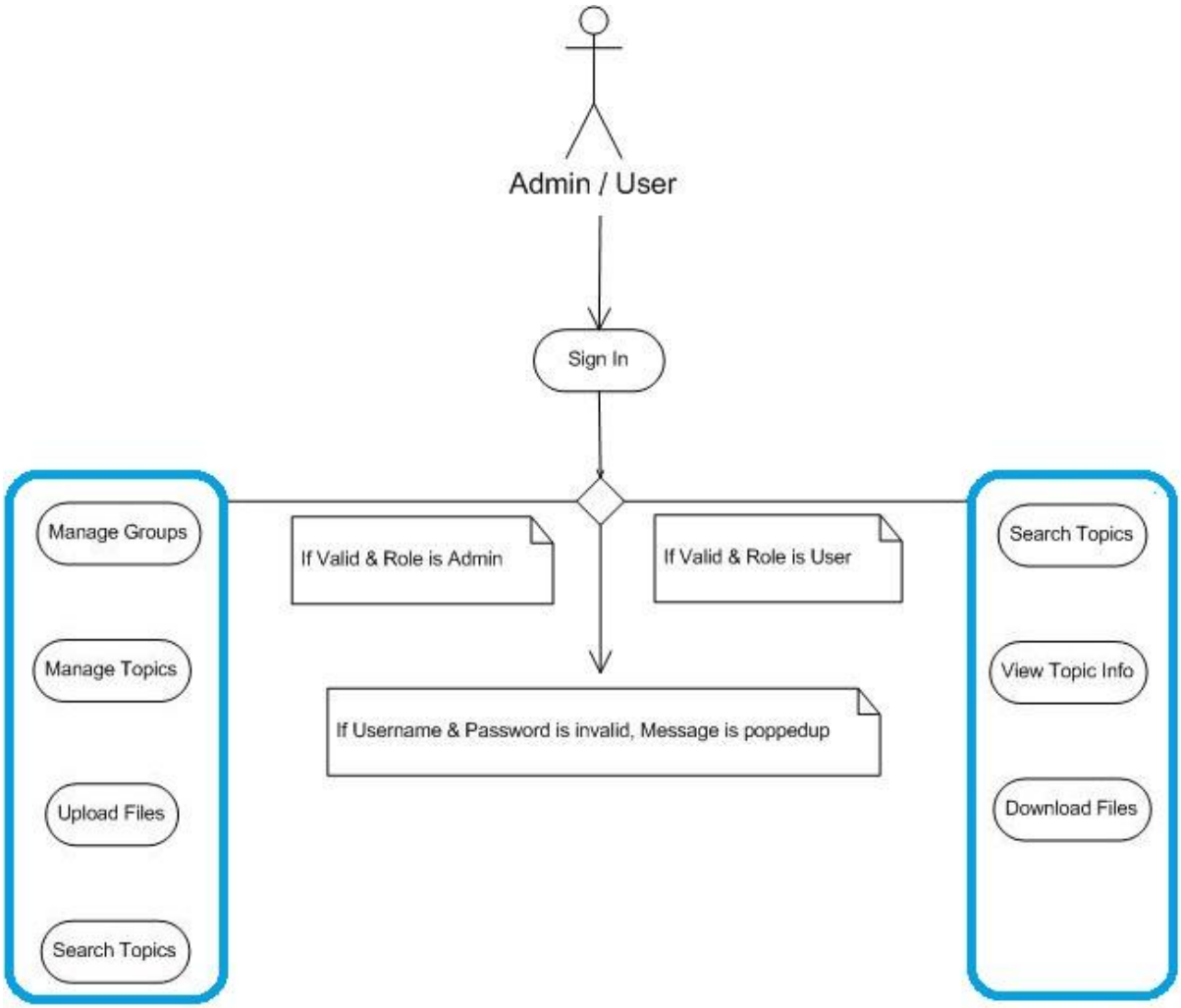
This strategy has been chosen because it isolates various system responsibilities from one another, so that it improves both system development and maintenance.

Each layer's specific responsibilities are as follow:

- The **presentation layer** deals with the presentation logic and the pages rendering on KBA web site. The Presentation layer contains all the components needed to allow interactions with an end-user. It encompasses the user interface.
- The **control layer** manages the access to the domain layer. Control layer contains all the components used to access the domain layer or directly the resource layer when this is appropriate.
- The **resource layer** (integration layer) is responsible for the access to the enterprise information system (databases or other sources of information). Resource layer contains the components needed to enable communication between the business tier and the enterprise information systems: Mongo Database. Refer to data view.
- The **domain layer** is related to the business logic and manages the accesses to the resource layer. The Domain layer contains all the components related to the business logic. It gathers all the subsystems that meet the needs of a particular business domain. It also contains the business object model which is specified in design document.
- The **Common Elements layer** gathers the common objects reused through all the layers like users and assets. Common Element layer contains the components re-used within several layers and may be determined in the future.

6. Process View

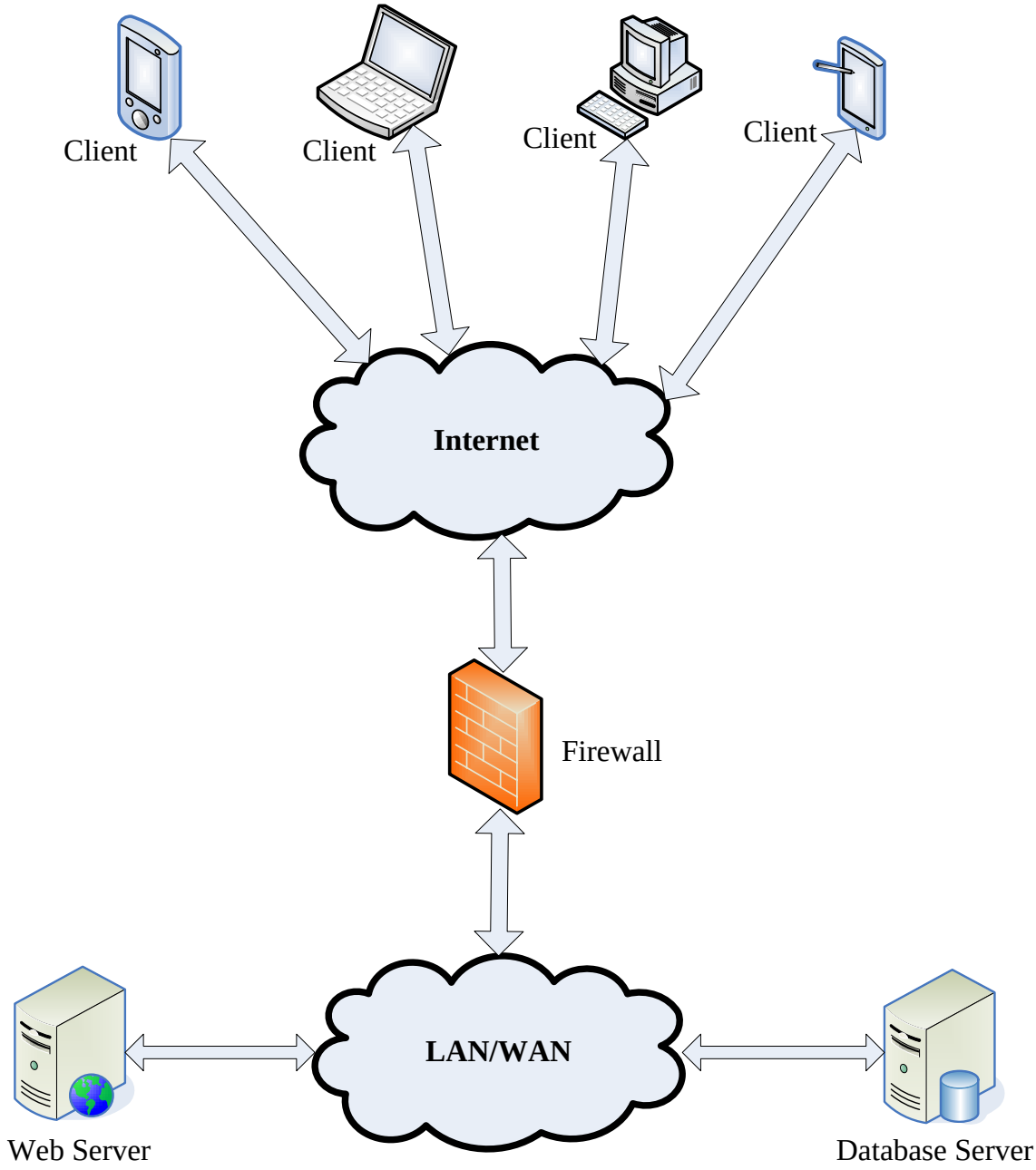
Process Flow:



7. Deployment View

Global Overview

KBA's deployment has not been considered yet. All future code and implementation details will be included in this section. Diagram below shows that all of the ROR code will have a physical location on the Apache's web server and all of the data entities and relationships will be physically located on the Mongo database server.



8. Data View

8.1 Entity Relational Diagram (ERD)