

# Darren Small T2A1-A, WORKBOOK PART A

---

## Question 1 Answer:

The typical architecture of a API project, such as a flask application with Python, would be using the MVC design pattern. MVC stands for Model - View - Controller and divides an applications architecture into these 3 main components. Looking at each of these components and their specific roles: - Model: The model component contains all the logic relating to data, so the user can interact with the database. It is the only component able to connect to the database and the model only communicates with the controller and responds with the requested data from the database to the controller as required. It can provide all the data related actions from the user being transferred between the view and controller. - View: The view component provides all the UI interactions of the application for the users inputs and display outputs. It only communicates with the controller to provide a view of the data from the model. - Controller: The controller component contains all the core logic and is the main processor of the application. It connects the view and model components and is the interface between them for taking requests from the view to manipulate data through the model. It then has the logic to provide the view with the correct rendered output to display to the user.

An example of a application with this type of architecture could be a weather app:

- 1 - The user opens up a web browser and goes to the weather applications main page. This is the View or the UI component. User interacts with the View and enters the locations and days they would like to know the weather outlook for.
- 2 - The View sends this request to the Controller to get the weather information.
- 3 - Then the Controller requests the Model to find this weather information from the database.
- 4 - The Model finds all the information from the database and returns this to the Controller.
- 5 - Then the Controller decides how to display this information to the user and advises the View on how to render this data for display.

## References:

Pratap M., 2024. How does the Python MVC Framework Work [Online] Available at: <https://supersourcing.com/blog/how-does-the-python-mvc-framework-work-what-are-the-benefits/>

GeeksforGeeks., 2024. MVC Design Pattern [Online] Available at: <https://www.geeksforgeeks.org/mvc-design-pattern/>

Kumar N., 2021. How the Model View Controller Architecture Works [Online] Available at: <https://www.freecodecamp.org/news/model-view-architecture/>

## Question 2 Answer:

A commonly used database in an API project, such as a flask application with Python, would be using a PostgreSQL database. The pros and cons of using this type of database are:

- Pros:

- It is a RDBMS system compliant with SQL standard making it easy to use for developers and users familiar with SQL.

- Postgres became ACID compliant in 2001, meaning it has the 4 properties of atomicity, consistency, isolation & durability, making it well known for its reliability. It can handle a large amount of data while being highly stable and not crashing or losing data. This maintains data integrity and does offer write ahead logging, to prevent data loss, as well as crash recovery procedures.

- Postgres is highly extensible & flexible. What this means is the database can easily be extended, it is not fixed, and supports a wide range of inbuilt data types as well as any new custom data types, functions, operators and languages can easily be added as required.

- It is highly scalable. Postgres can be run on a single server or multiple servers to manage large amounts of data. It can also handle a large number of concurrent users making it the ideal choice for large applications requiring both these features.

- Has MVCC - Multi Version Concurrency Control which enables the use of multiple transactions accessing the same data simultaneously with out any issues.

- Postgres security is very high with role based access controls, secure connections and data encryption available.

- Postgres can support data warehousing and business intelligence tasks with advanced analytics & data visualization.

- It is open source and freely available, allowing the freedom of use to implement and modify according to the project needs. Because of this active and open source community it is continually updated and improved on for better solutions.

- Cons:

- Postgres performance can be slower than other database systems when performing certain types of data structures or queries.

- For new users and developers it can be quite complicated and hard to learn. Because the database is highly configurable, this type of complexity is a potential drawback.

- The Postgres documentation can be hard to navigate, as being open source there are a large volume of resources available and many tools support MySQL and not Postgres.

- Migrating data from other databases is challenging and difficult, especially for large amounts of data.

### References:

BrainerHub Solutions. 2023. PostgreSQL: A Practical Guide—Features and Advantages. [Online] Available at: <https://www.linkedin.com/pulse/postgresql-practical-guidefeatures-advantages-brainerhub-solutions>

Quest. What is PostgreSQL and how does it compare to other database management systems? [Online]

Available at: <https://www.quest.com/learn/what-is-postgresql.aspx>

Nguyen, H. 2024 What is PostgreSQL and Everything You Need to Know. [Online] Available at: <https://techvify-software.com/what-is-postgresql/>

Vojak, J. 2022. Exploring the pros and cons of SQL databases — MySQL, Postgres, Oracle, Microsoft SQL, and Amazon Aurora. [Online] Available at: <https://josipvojak.com/exploring-the-pros-and-cons-of-sql-databases-mysql-postgres-oracle-microsoft-sql-and-amazon-3c8de880b8d4>

## Question 3 Answers:

The implementation of an Agile project management methodology for an API project would be to use either the SCRUM approach or the KANBAN approach. But the approach to choose ultimately depends on the type of API project being undertaken, the development team in place and the project outcomes for the client. Below is a description of each Agile approach and example(s) of a real world example where businesses have applied this to their own software/api projects:

- SCRUM:

Scrum methodology for projects is where the team promises to achieve and deliver a piece of project work by the end of a certain amount of time, called a sprint. Sprints usually last for a length of time between 1 to 4 weeks. The team usually consists of a Product Owner, Scrum Master and the development team members. The task for the sprint will be broken down into smaller tasks called 'stories' where each development team member will be assigned their 'story' to work on for that particular sprint.

The scrum master works together with all dev team members to help them stay focused, resolve any issues and keep the sprint task to its designated timeframes. This is usually done by also running regular daily team meetings.

The product owner prioritizes the tasks for the dev team and the backlog of the project. They also liaise with the client in regards to the overall progress of the project and any changes or concerns they have for their overall outcomes of the project.

At the end of each sprint task goal, the whole team will come together to review, plan ahead for future sprints and retrospectively analyse what went well and not so well for this task.

Scrum real world example:

- Salesforce:

- Salesforce embraced a scrum methodology for its cloud based products back in late 2006. Before this, and the reason for their change, was because they had seen their seasonal releases slip from 4 per year, when first started, go down to just one per year. So they knew they had to change their development methodology. Since adopting a scrum methodology in late 2006, by 2010 each major release had been successfully delivered on the exact day as scheduled and their customer satisfaction indicator was at 94%. Another scrum advantage they saw was an increase in productivity, measuring by features

produced per developer, they saw an increase of 38%. The case study white paper done on this in 2010 can be reviewed from this link:

- [The inside story of salesforce.coms transformation from waterfall to agile](#)

- KANBAN:

Kanban methodology for projects is more a visual signaling mechanism where the workflow is represented on a kanban board by tasks that will flow from one stage of the workflow to the next stage. For example, the most common workflow or columns of each stage are 'To Do', 'In Progress', 'In Review' and 'Done' but these are most often customised by the development team for each specific project.

When each task has been reviewed / accomplished by a development team member at a stage on the kanban board, they will move the task to the next stage until the outcome of 'Done' is achieved. Depending on the size of the project and the development team, each stage could be assigned and achieved by the same team member or different team members.

With kanban, releases or updates are done when they are ready, not based on predetermined dates or a regular schedules like scrum. The whole development team works together and owns the kanban board to achieve the projects outcomes. Because of this, the most common practice to deal with issues or blockages, is to implement Work In Progress (WIP) limits, where caps are placed on the number of tasks that can be placed in any particular column. When the cap or limit is reached the whole team can work together to resolve the tasks that stalled the workflow process.

Kanban real world example:

- Microsoft:
  - Microsoft Commerce and Ecosystems division were having issues with most engineers still using a mixture of both the waterfall and scrum methodologies. They were constantly missing sprint conclusions and had a need to visualize their work, which scrum could not provide. So they decided to transition across to the kanban method. The kanban method and its visual nature made it easier for the entire team to see what needed to be done, whether it was a newcomer to the team, someone back from vacation or a part timer, all team members could visually identify what's next on the kanban board. The results were a smoother process, happier customers and a team that worked well together. The article and reference can be viewed from this link:
  - [Deploying Kanban at Microsoft leads to engineering excellence](#)

References:

Rehkopf, M. Atlassian. Kanban vs. scrum: which agile are you? [Online] Available at:  
<https://www.atlassian.com/agile/kanban/kanban-vs-scrum>

GeeksforGeeks. 2024. Kanban vs. Scrum : Top Differences You Should Know [Online] Available at:  
<https://www.geeksforgeeks.org/difference-between-scrum-and-kanban/>

## Question 4 Answers:

A standard source control process for an API project will be first the project development team needing to use / deciding to use a version control systems software. This VCS software is used by the development team to manage changes in the projects source code over the length of time the project takes to complete. The development team process for the VCS of the project will be applying a number of standard practices, these are:

- Repository setup: organising the structure of the code into clear and consistent repositories.
- Branching strategies: to manage code changes a good branch strategy is required. This could be having a 'main' branch for production ready code, a 'dev' branch for ongoing development and testing, then creating new branches as required for future version releases or bug fixes where these can be merged back into the main branch when finished.
- Commit messages & frequency: when team members are committing changes they need to write concise and informative messages. This way other members of the team will know and understand what they have done. Members should also be instructed to make regular frequent small commits rather than infrequent larger commits. This makes tracking changes and identification of bugs easier for all team members.
- Using pull requests: team members should use pull requests for all code changes. These pull requests can be used for reviewing code, testing and code quality reviews before merging them back into the 'main' branch.
- CI/CD processes: a Continuous Integration/Continuous Deployment process should be implemented. This involves a pipeline to trigger the automation of the building, testing and deployment of code changes to the dev environment.
- Resolving conflicts: all team members should work together to resolve any merging conflicts quickly when they happen. These conflicts happen when different members have modified the same part of a file.
- .gitignore: To specifically have any file or directories ignored by the VCS software, member should use a .gitignore file.
- Security: access control measures should be used by the dev team to protect the main production branch. This security should help prevent things like deletions, direct pushing to production and unauthorised access.
- Cleanup process: a branch cleanup process should be regularly done to remove old & unused branches. This will maintain a manageable and clean repository.
- Documentation: The repository should include a documentation file or README file to guide any new or current dev team members of the processes of building, running, testing and cloning any part of the software project.

Examples of VCS software tools available in the market are:

1. Git: Git is a open source VCS software that tracks the project changes, staging areas and commits these to the repository. It is the most widely used VCS system in the world and has become a standard in the software development industry.
2. GitHub: GitHub is a cloud based hosting service tool that hosts the Git repositories. GitHub provides a wide variety of different VCS features for collaboration and access control. It is owned by Microsoft and tends to be used for more open source community public code access.
3. BitBucket: is a very similar VCS Git repository management tool with all the

same features as GitHub but tends to more widely used for the private business sector. It is owned by Atlassian.

4. AWS CodeCommit: this is AWS (Amazons) version of a VCS Git repository management tool.

Examples of projects using VCS via GitHub:

1. This link provides a basic GitHub repository template & example and can be used to create a well documented README file:

[GitHub Call-for-Code / Project-Sample](#)

2. This link is the Ruby on Rails platform project. This has 5000+ contributors & 92380 commits to date:

[GitHub rails / rails](#)

3. This link is the facebook / react platform project. This has 1646 contributors & 19245 commits to date:

[GitHub facebook / react](#)

References:

Atlassian, 2024. What is version control? [Online] Available at: <https://www.atlassian.com/git/tutorials/what-is-version-control>

Masal, B. 2023. Best Practices for Effective Source Control (version control) in Software Development. [Online] Available at: <https://www.linkedin.com/pulse/best-practices-effective-source-control-version-software-balaji-masal-agxpf>

Indeed. 2024. Source control: definition, importance and examples. [Online] Available at: <https://uk.indeed.com/career-advice/career-development/source-control>

mattfarina. 2024. Large Projects on GitHub [Online] Available at: <https://gist.github.com/mattfarina/7627cb5ebb8fc01bfd62f4a6942fce04>

## Question 5 Answers:

For a standard testing process for a API project, the development team will first have to either write their own framework or choose from a number of API software testing tools. Some of the most common API testing tools available are:

- Postman API Platform.
- Amazon API Gateway.

- Google Apigee.
- Insomnia by Kong.
- Swagger from SmartBear.
- Microsoft Azure API Management.

The development team will then decide on the types of tests required for their application by verifying the functionality & performance of the API. The most common types of API tests are:

- UI testing: not specific to an API but this testing is important for the applications integration between the user interface (front end experience) and the API (back end functionality). This type of testing is usually a manual process.
- Functional testing: this testing ensures the API functions in the exact way it is designed to for the application. It is done by verifying that the correct responses and data formats are returned by the API from specific requests. These tests should include:
  - status code verification: the correct status codes are returned eg; 200, 404 etc.
  - response data validation: the responses include all the correct data fields & values.
  - date accuracy: data returned is as expected & accurate.
  - error handling: all errors are handled correctly and display meaningful messages.
  - operates using CRUD: all create, read, update and delete functions works as expected.
- Performance testing: this testing ensures the API can perform and handle large volumes of data and traffic. It is done by replicating multiple users with multiple requests so the API will be under load and then measuring its performance. These tests should include:
  - load test: replicating high user traffic to measure speed & responsiveness times.
  - response time: measuring the time to respond between different loads eg; normal & peak loads.
  - scalability: when increasing the load and users, testing how it scales.
- Security testing: this testing ensures the API is secure. It must be able to prevent any unauthorized access and protect the applications sensitive data. This is done by executing tests to identify any vulnerabilities in the API system. These tests should include:
  - authentication: verifying the proper authentication and manages invalid credentials correctly.
  - authorization: verifying all users can only have access to the data they can modify or view.
  - input validation: tests like SQL & command injection to validate all inputs.
  - data encryption: testing the data during transmission is encrypted.
- API documentation testing: this type of testing is used to determine that the API documentation accurately reflects its capabilities and all its features work as expected for the projects application. Similar to functional testing but could be incorporated with tests that include:
  - ease of use: verifying the API integrates well and is easy to use.
  - naming conventions: checking the naming conventions of all endpoints and parameters are consistent.
  - document accuracy: verify the documentation is up to date and accurate for the API.

Examples of real world API testing:

- a airline booking site application like skyscanner, needs to display to the user all the correct airline flight details, dates and rates in a timely manner. To do this it must effectively communicate with all the different airlines API's. The skyscanner development team must regularly do the above API tests to ensure their website application is successfully communicating with all the different airlines API services and returning the correct results in a acceptable timeframe. Should the user also then decide to

purchase a certain airline product they must ensure the users personal details are effectively secure and correctly sent to the airlines boooking system.

- another example is the company i work for called myPak Solutions. MyPak provides a packing application for pharmacies around Australia to print their patients prescribed medication details on a blisterpack packaging supplied by the pharmacy, so they can effectively take their medications in a safe manner, reducing the chance of dependency or overdosing. To do this the myPak application must have access to the pharmacies medication dispensing programs data, which is done via their API's. There are currently 6 dispensing medication software companies in Australia and 4 of them provide access to their data via their own API's. MyPak's development team regularly use the same above API tests to ensure that patients medication data its packing application is accessing is correct, secure and achieved in a timely manner.

#### References:

Yasar K, Gillis A. 2024. What is API testing? Everything you need to know. [Online] Available at: <https://www.techtarget.com/searchapparchitecture/definition/API-testing>

Codeheart I. 2023. API Testing: A Guide for Developers. [Online] Available at: <https://caisy.io/blog/api-testing-for-developers>

Testsigma. 2024. API Testing : What It is, How to Test & Best Practices. [Online] Available at: <https://testsigma.com/guides/api-testing/>

## Question 6 Answers:

The 3 principles of information system security are explained below and known as the CIA triad:

1. Confidentiality: this principle is to ensure that unauthorised users cannot access sensitive / personal information & data. Measures must be put in place to ensure only authorised users can access this type of information and any other user is restricted , so that this type of data will remain confidential. One of these measures would be to check the authorisation levels of all users that have access to the applications data.
2. Integrity: this principle is to ensure the data is accurate, consistent and trustworthy. Measures are required so that no modifying of the data is done in transmission either accidentally or intentionally by malicious attacks, or by any unauthorised users.
3. Availability: this principle is to ensure that data is available to the authorised users of the application within a given timeframe. The timeframe availability is determined by the use case of the application but all technical infrastructure & hardware must be maintained, online and backed up to meet these expectations.

#### References:

GeekforGeeks. 2022. Principle of Information System Security. [Online] Available at: <https://www.geeksforgeeks.org/principle-of-information-system-security/>

Milkovski I. 2022. What are the 3 Principles of Information Security? [Online] Available at: <https://www.linkedin.com/pulse/what-3-principles-information-security-igor-milkovski-m-a>

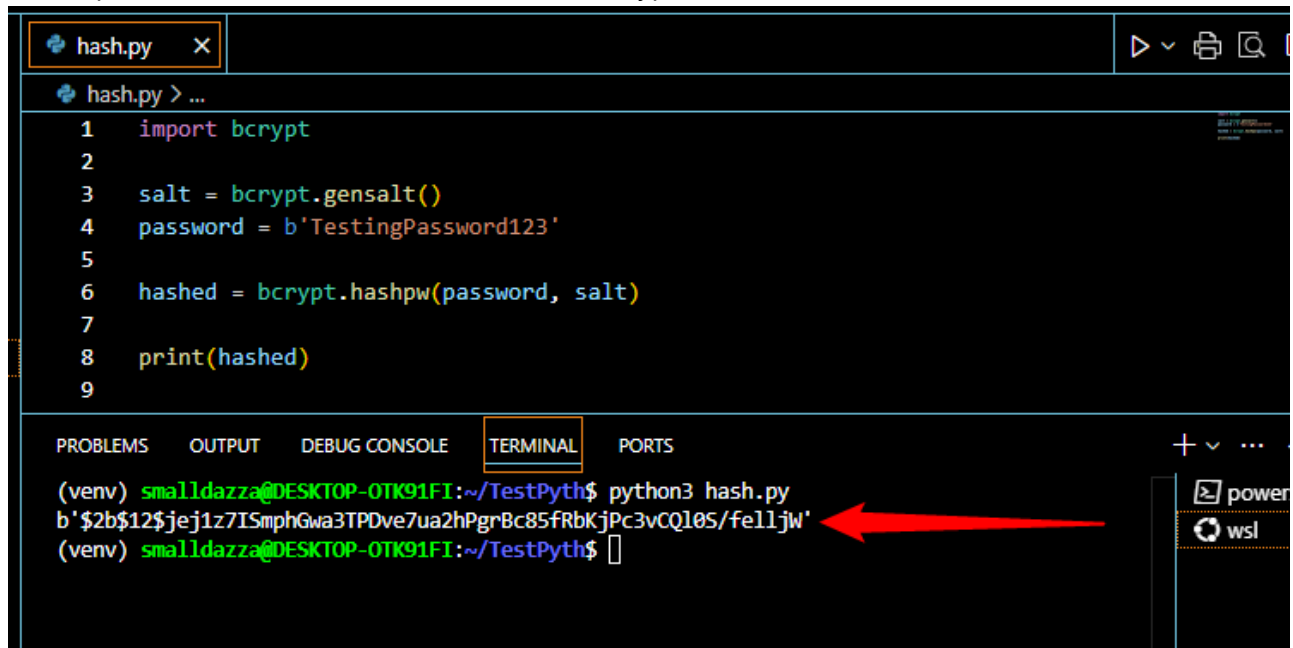


Hashemi-Pour C. CIA triad (confidentiality, integrity and availability). [Online] Available at: <https://www.techtarget.com/whatis/definition/Confidentiality-integrity-and-availability-CIA>

## Question 7 Answers:

The implementation of the 3 CIA triad principles of information system security in an API project is essential and in most cases must be done to meet local and international privacy laws. To achieve all 3 principles in any API project the following measures should be implemented:

- data encryption mechanisms: to maintain the integrity of data during transmission data encryption can be achieved by using secure protocols such as HTTPS. HTTPS will encrypt HTTP requests & responses with a technology called TLS. If an attacker were to access these HTTPS requests and responses they would only see random characters instead of the text being transmitted.
- password encryption techniques such as hashing: one process called hashing can help achieve confidentiality. It is widely used in cryptography and is the process of converting any data into a fixed length hash that can never be reversed. This is why it is widely used for encrypting passwords. There are many different ways in coding to hash a password but one example, in the image below, is in Python to hash a password we can use the module called 'bcrypt':



```
hash.py
hash.py > ...
1 import bcrypt
2
3 salt = bcrypt.gensalt()
4 password = b'TestingPassword123'
5
6 hashed = bcrypt.hashpw(password, salt)
7
8 print(hashed)
9

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
(venv) smalldazza@DESKTOP-OTK91FI:~/TestPyth$ python3 hash.py
b'$2b$12$je1z7ISmPhGwa3TPDve7ua2hPgrBc85fRbkjPc3vCQ10S/felljw'
```

The red arrow indicates the encrypted or hashed password generated. The 'salt' variable in this code is a random input of additional one way data added to the hash for an extra layer of security.

- the use of checksum methods: to check the accuracy of data transmission over a system or network and that no changes, damage or errors have occurred between the sender & receiver, a process called checksum can be used. The checksum method in coding is an algorithm that derives a value from the data, then compares this value at both the sender and receiver's end. If there are any changes in the value it indicates the unreliability of the data transmitted. This process verifies the integrity and authenticity of the API data.
- keeping authorised user permissions & access control lists up to date: in keeping API access & data confidential, application administrators should regularly inspect that the list of users and permission levels are up to date. Especially if the API application is a large project with many team members/users joining or leaving, as this will not allow any unauthorised access from team members that have left.

- using two-factor authentication or a OTP for access to an application or API: the use of either a 2FA or OTP process for users when requiring access is a second layer of security to ensure both the confidentiality & integrity of the API data , so only users with these keys are allowed to create, read, update & delete. The most common of these used for API security is generating security tokens or One Time Passcodes (OTP) to be entered and verified before the users access or request is granted. Using python there are a number of modules & libraries to help with both of these. Below are a couple of examples applied in code:

- security tokens using the Json Web Tokens (JWT) library:

```
JWTtoken.py X
JWTtoken.py > ...
1 import jwt
2
3
4 key = "secret"
5 token_encoded = jwt.encode({"key": "my_password"}, key, algorithm="HS256")
6
7 print(token_encoded)
8
9 decode_token = jwt.decode(token_encoded, key, algorithms="HS256")
10
11 print(decode_token)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

(venv) smalldazza@DESKTOP-0TK91FI:~/TestPyth\$ python3 JWTtoken.py  
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpzZW50L3Npd29yZCJ9.2gBH0xihax0p75N65-r\_u09NmH5xx8oHGJwnRRFin6I

(venv) smalldazza@DESKTOP-0TK91FI:~/TestPyth\$

The green arrow indicates the password entered (as a dictionary), then the red arrow indicates the jwt token generated to encode this password. The blue arrow indicates the token decoded to display and verify it is the same correct password.

- One Time Passcode (OTP) using the pyotp library:

```
OTP.py 1 X
OTP.py > ...
1 import pyotp
2
3
4 secret_key = pyotp.random_base32()
5
6 hotp = pyotp.HOTP(secret_key)
7 otp_code = hotp.at(0)
8
9 print("Your OTP number is: ", otp_code)
10
11 user_otp = input("Confirm your OTP number: ")
12 if (hotp.verify(user_otp, 0)):
13     print("Access granted!!")
14 else:
15     print("Incorrect OTP. Denied access. :( ")
16
```

PROBLEMS (1) OUTPUT DEBUG CONSOLE TERMINAL PORTS

(venv) smalldazza@DESKTOP-0TK91FI:~/TestPyth\$ python3 OTP.py  
Your OTP number is: 334707  
Confirm your OTP number: 334707  
Access granted!!

(venv) smalldazza@DESKTOP-0TK91FI:~/TestPyth\$ python3 OTP.py  
Your OTP number is: 800694  
Confirm your OTP number: 800693  
Incorrect OTP. Denied access. :(

(venv) smalldazza@DESKTOP-0TK91FI:~/TestPyth\$

Here are 2 examples of an OTP number being generated. The green arrow indicates when the user enters the correct matching OTP number and access is granted. The red arrow indicates when the user enters an incorrect OTP number and access is denied.

- a VCS (version control system) for application releases: it is essential that the development team use a version control system for the life cycle of an API project. A VCS system ensures all changes to code are monitored, saved and merged so that the application can have scheduled update releases and the integrity of the code is ensured for reliability.
- backing up of data, redundancy and failover measures: backing up of data is an important measure to ensure the availability of the applications data. Regular backing up of the database and storing these backups in a secure and stable environment, so if a failure were to happen a database backup restoration procedure could be initiated to have the API application back online as soon as possible. Failover mechanisms like a backup webserver that can immediately take over the responsibilities should the main web server have a failure, preventing DDOS attacks by using load balancers to restrict & control the flow of traffic to certain ports or areas of the application, these are all options to ensure the availability of an API's application to the users when required.

#### References:

Alrazihi T. 2023. Enhancing Software Security: Implementing the CIA Triad in Your Application. [Online] Available at: <https://www.linkedin.com/pulse/enhancing-software-security-implementing-cia-triad-your-alrazihi-pw0ae>

Hashem-Pour C. 2023. CIA triad (confidentiality, integrity and availability). [Online] Available at: <https://www.techtarget.com/whatis/definition/Confidentiality-integrity-and-availability-CIA>

freeCodeCamp. 2020 The CIA Triad — Confidentiality, Integrity, and Availability Explained. [Online] Available at: <https://www.freecodecamp.org/news/the-cia-triad-confidentiality-integrity-and-availability-explained/>

Krishna A. 2023. How To Generate OTPs Using PyOTP in Python. [Online] Available at: <https://blog.ashutoshkrish.in/how-to-generate-otps-using-pyotp-in-python>

Padilla J. 2022. Welcome to PyJWT [Online] Available at: <https://pyjwt.readthedocs.io/en/stable/index.html#>

## Question 8 Answers:


The legal obligations for a social media application to consider in regards to handling their user personal data if operating in Australia will be the Australian Government's Privacy Act 1988. This Privacy Act applies to all Australian organisations with an annual turnover of more than \$3 million or factors in if an organisation operates in Australia, including carrying on a business or presence in Australia, so will include all social media applications like Facebook, Twitter, Instagram etc.

The Privacy Act 1988 was introduced to protect & promote the privacy of individuals and how these organisations handle personal information. Specifically regarding social media applications and the legal obligations of the Privacy Act they would be required to meet, some would be the following (Note = the term 'organisation' below will refer to a social media developer / application):


- Consideration of a Privacy Policy: an organisation covered under the Privacy Act 1988 must have a privacy policy in place which can be printed out or viewed on a website or mobile device. This privacy

policy must cover and disclose the following:


- organisations name and contact details.
- the kind of personal information they collect and save.
- how this information is collected and where they save it.
- their reasons for collecting this personal information.
- how they use this personal information.
- how users can access their own personal information and change it.
- how complaints can be lodged and handled regarding personal information.
- if your information will be disclosed outside of Australia and the other countries they will disclose it to.
- may also include the time frame they hold onto your personal information.

Reference:  Privacy Act 1988, Schedule 1, Part 1.

- Personal & sensitive information collected by a organisation:
  - Personal information will refer to a individuals name, address, phone number, DOB, signature, photographs, biometrics and location information from a mobile device.
    - however if a photo or video taken by someone, without permission and was done in a personal capacity, then the Privacy Act 1988 does not apply to these.
  - Sensitive information refers to a individuals race or ethnicity, political & religious beliefs and sexual orientation.
  - the organisation must not collect personal information unless it is reasonably necessary, or directly related to one or more of the organisations activities.
  - the organisation must not collect sensitive information about an individual unless the individual consents to the collection and it is reasonably necessary, or directly related to one or more of the organisations activities.
  - the organisation must only collect personal information by lawful & fair means.
  - the organisation must collect personal information about that individual only from the individual.


Reference:  Privacy Act 1988, Schedule 1, Part 2, Principle 3.

- Unsolicited Personal Information: if the organisation receives personal information it did not ask for / solicit, then the organisation needs to determine if they would have collected the information using the same methods in above Principle 3:
  - If not = they must destroy the unsolicited personal information.
  - If so = they must notify the individual as per Principle 5 details and to make them aware of the collection of their personal information and hold it following the same privacy act guidelines.


Reference:  Privacy Act 1988, Schedule 1, Part 2, Principle 4.

- Notification of collecting personal information: when an organisation collects an individuals personal information it must notify the individual that it has done so and :
  - the organisation must identify itself and its contact details.
  - how it received this information.
  - the purposes for which they will use this information, including if they share this information with any other organisations.

- the organisation must disclose how the individuals can access their privacy policy and allow them access to their personal information and / or make complaints about their use of this personal information.

Reference:  Privacy Act 1988, Schedule 1, Part 2, Principle 5.

- Dealing with personal information: if an organisation has collected / holds the personal information of an individual for a particular primary purpose then it must not use or disclose this information for any other secondary purposes unless:
  - they receive the consent of the individual to do so.
  - individuals would expect their personal information to be used for the secondary purpose if it is related to the primary purpose.
  - required by an Australian law or court order.

Reference:  Privacy Act 1988, Schedule 1, Part 3, Principle 6.

## Question 9 Answers:

A relational database allows data to be stored in tables that are related to each other using defined relationships. The structural aspects of these types of databases are:

- Tables: the main building block of these databases which are sometimes called relations. Tables will each represent a particular entity of data to be stored and are structured into rows and columns. Each of these rows in the table will be a data instance of the entity. Each of the columns in a table will define certain attributes or characteristics of these instances of data.
- Rows: also called records will each contain instances of data for the entity being stored in the table. Each row will contain individual data sets that correspond to each table column/attribute.
- Columns: also called attributes and these will define the type of data that can be stored in the table. Each column will be named relevant to the specific entity data it will contain and the data type that it will be eg; dates, integer, text. They will also define data constraints such as if it will be a unique value or a null value.
- Primary keys: these are a certain column in most tables that uniquely identifies each row of data. They ensure that each record is a distinct data set and can be directly referenced when required. Primary keys enable different tables in a database to have relationships via referencing another table's foreign key.
- Foreign keys: also a certain column in a table that can reference a primary key in another table. This also enables relationships between different tables so data in one table can be linked to data in another table.
- Constraints: these are the rules to enforce the integrity and consistency of the data being stored. There are primary key constraints, foreign key constraints, unique constraints which will help prevent invalid data being entered into the database tables.
- Indexes: these are data structures to speed up the querying of database table data. They contain all the information required to quickly and efficiently access the data requested.

References:

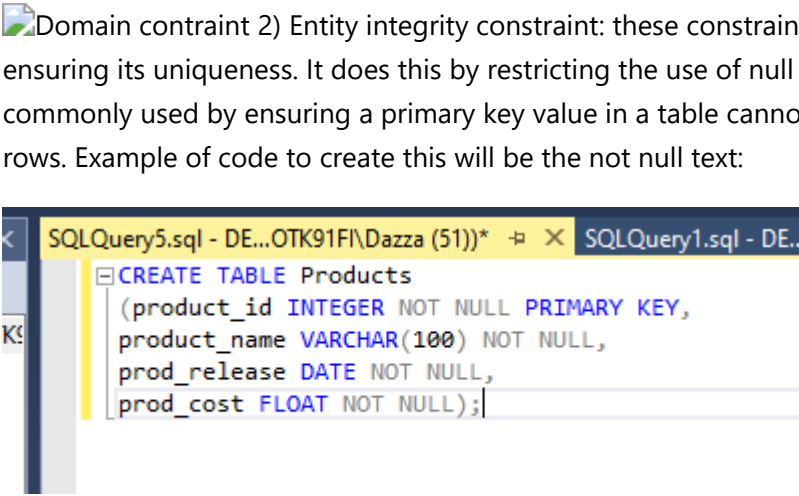
Gillis A. 2024. Definition relational database [Online] Available at:  
<https://www.techtarget.com/searchdatamanagement/definition/relational-database>

SQL Savvy. 2024. The structure and components of relational databases [Online] Available at: <https://www.linkedin.com/pulse/structure-components-relational-databases-suchc>

## Question 10 Answers:

In a relational database the integrity aspects are known as 'Integrity Constraints'. These constraints are a set of rules that help to ensure the data quality, consistency and security by being applied to the CRUD processes on the DBMS. The types of integrity rules are:

1. Domain integrity constraint: these constraints are applied to each attribute in the database to ensure and check the value being entered meets the correct data type set by the constraint. For example, these constraints can be integer, character, date, time, string, etc. An example of these being used to create a table & its columns will be the blue code text:

Domain constraint 2) Entity integrity constraint: these constraints enforce the integrity of a table's data by ensuring its uniqueness. It does this by restricting the use of null values into the database attributes. Most commonly used by ensuring a primary key value in a table cannot be null, as this is a unique identifier for all rows. Example of code to create this will be the not null text:

```
SQLQuery5.sql - DE...OTK91FI\Dazza (51))* X SQLQuery1.sql - DE...
CREATE TABLE Products
(product_id INTEGER NOT NULL PRIMARY KEY,
product_name VARCHAR(100) NOT NULL,
prod_release DATE NOT NULL,
prod_cost FLOAT NOT NULL);
```

3) Referential integrity constraint: these constraints establish a relationship and enforce data integrity between 2 tables also preventing inconsistencies when referring to data across different tables. When a foreign key in a table 2 which references a primary key in another table 1, these constraints make sure every foreign key value exists in the table 1 primary key value. In code an example of the foreign key references:

```
SQLQuery8.sql - DE...OTK91FI\Dazza (72))* X SQLQuery7.sql - DE...OTK91FI\Dazza (71))* X
CREATE TABLE Orders
( OrderID INTEGER NOT NULL PRIMARY KEY,
CustID INTEGER FOREIGN KEY REFERENCES Customer(CustID),
OrderDate DATETIME );
```

4) Key integrity constraint: these constraints are used to uniquely identify one or more columns in a table and restrict duplicate values. The primary key constraint is a good example of this where it is used to uniquely identify each row in the database ensuring no records will have the same primary key value. In code using the primary key:

```
SQLQuery9.sql - DE...OTK91FI\Dazza (75))* X SQLQuery8.sql - DE...OTK91FI\Dazza (72))* X
CREATE TABLE Students (
StudentID INT PRIMARY KEY,
FirstName VARCHAR(50),
LastName VARCHAR(50)
);
```

References:

W3 schools. 2024. SQL Constraints [Online] Available at: [https://www.w3schools.com/sql/sql\\_constraints.asp](https://www.w3schools.com/sql/sql_constraints.asp)

Gupta V. 2024. Integrity Constraints in DBMS - Types and Examples [Online] Available at: <https://www.almabetter.com/bytes/articles/integrity-constraints-in-dbms>

Suganghi A. 2024. What are Integrity Constraints in DBMS? Types, Examples [Online] Available at: <https://www.knowledgehut.com/blog/database/integrity-constraints-in-dbms>

## Question 11 Answers:

The manipulative aspects of a relational database using SQL has 2 aspects, one being Data Definition Language (DDL), the second being Data Manipulation Language (DML). Both DDL & DML are commands used to create, access, transform, remove and analyse the data in the database tables.

- Data Definition Language (DDL) are a set of commands used to define the schema of a database. Such commands will specify the structure of the database objects like the tables, indexes, views and constraints. Types of DDL commands are CREATE, DROP, ALTER, TRUNCATE AND RENAME and once these have been executed they automatically commit the changes, so cannot be undone. Examples in SQL:

```
SQLQuery8.sql - DE...OTK91FI\Dazza (52))*  
CREATE TABLE Patients (  
    PatientID INT,  
    FirstName VARCHAR(200),  
    LastName VARCHAR(200),  
    Department VARCHAR(100)  
);  
  
ALTER TABLE Employees  
ADD Salary INT;  
  
DROP TABLE Employees;
```

- Data Manipulation Commands (DML) are a set of commands used to manipulate data inside of database tables. The operations they perform on the database data are transactional to add, alter, retrieve or remove and so do not automatically commit when executed, so can be rolled back if necessary should any errors happen. The types of DML commands used are SELECT, INSERT, UPDATE, MERGE and DELETE. Example in SQL:

```
SQLQuery9.sql - DE...OTK91FI\Dazza (73))* SQLQuery8.sql - DE...OTK91FI\Dazza (52))*  
INSERT INTO Employees (Employee_id, FirstName, LastName, Department)  
VALUES (10, 'John', 'Smith', 'IT');  
  
UPDATE Employees  
SET Salary = 50000  
WHERE employee_id = 10;  
  
SELECT * FROM Employees;  
  
DELETE FROM Employees  
WHERE employee_id = 10;
```

## References:

Panigrahi K. 2023. Difference between DDL and DML in DBMS [Online] Available at:  
<https://www.tutorialspoint.com/difference-between-ddl-and-dml-in-dbms>

GeeksforGeeks. 2024. Difference Between DDL and DML in DBMS [Online] Available at:  
<https://www.geeksforgeeks.org/difference-between-ddl-and-dml-in-dbms/>

Bytes A. 2024. DML and DDL Commands in SQL [Online] Available at:  
<https://www.almabetter.com/bytes/tutorials/sql/dml-ddl-commands-in-sql>

## Question 12 Answers:

Conducting research into a web application: I have choesn the web app Notion.so, the url for this web app is:  
<https://www.notion.so/>

### 1. The tech stack used by the Notion application:

- Frontend:
  - Frontend foundational browser technologies of html & javascript will be used for the applications UI to be displayed in users web browsers. Javascript is the programming language of choice for Notions app.
  - React: for a responsive user interface experience, notion is using components from the React javascript library. React allows developers to build application user interfaces by combining different design components or creating custom components using javascript functions, for their application UI screens, web pages and apps.
  - Redux: for javascript apps this is used to manage the 'state' of the application and handle complex data flow to keep it behaving consistently in different environments (client, server and native).
  - Typescript: is used to ensure developer type safety and a better experience. This is a syntactic superset that is added on top of javascript as a syntax, so the developers know exactly what type of data is being used in the code.
  - Webpack: to help with the applications performance webpack processes the applications javascript modules into bundles, which is usually one file, for better loading time of web browsers.
- Backend:
  - Node.JS: notion builds its backend services with this javascript runtime environment. This allows both the front end and backend to be built using javascript and allows the application to run on any operating system software.
  - Express.JS: this is used to handle their server side routing and build notions API endpoints. Express.js is a framework that builds RESTful API's for use with Node.js.
  - PostgreSQL databases: notion uses this as their primary databases for managing and storing the user data.
- Hosting:
  - AWS: notion are using AWS RDS and S3 servers for hosting their databases and other services for effective data efficiency and management.

## References:



Michael. 2023. Breaking Down Notion's Tech Stack. [Online] Available at: <https://slashdev.io/-breaking-down-notions-tech-stack>

The SaaS Talk. 2024. Exploring Notion: The Tech Stack Behind the Popular Tool [Online] Available at: <https://medium.com/@TheSaaS Talk/exploring-notion-the-tech-stack-behind-the-popular-tool-c3250326303d>

Slim. 2019. I'm the engineer at Notion who rebuilt search. AMA! [Online] Available at: [https://www.reddit.com/r/Notion/comments/f0lb23/im\\_the\\_engineer\\_at\\_notion\\_who\\_rebuilt\\_search\\_ama/](https://www.reddit.com/r/Notion/comments/f0lb23/im_the_engineer_at_notion_who_rebuilt_search_ama/)

Umen D. 2024. Best technology stack for SaaS applications [Online] Available at: <https://brights.io/blog/saas-technology-stack>

2. Based on Notions own technology blogg and other sources (referenced below), Notions app infrastructure is being hosted by AWS services. In particular they are using the Amazon RDS service for use with PostgreSQL and AWS boast this is the easiest way to deploy, operate and scale PostgreSQL databases in the cloud. Notion are also using a number of other AWS services being integrated into their infrastructure for helping with large data management, efficiency and security purposes. The following AWS infrasturcture services are being utilised by Notions app:

- Amazon RDS for PostgreSQL cloud service: PostgreSQL databases used for storing users data entered such as texts, headings, rows, lists, images and pages.
- Amazon S3 object storage servers: S3 servers are used for their ability to store, scale effectively for large amounts of data and support data processing engines at low cost.
- Apache Spark via Amazon EMR services: Spark is a open source distribution processing system for managing large data workloads that can optimize query execution for fast analytic queries against data of any size. Notion use this as their main data processing engine.

#### References:

Notion tech blogg. 2024. Building and scaling Notion's data lake [Online] Available at: <https://www.notion.so/blog/building-and-scaling-notions-data-lake>

Labs Relbis. 2024. Examining Notion's Backend Architecture [Online] Available at: [https://labs.relbis.com/blog/2024-04-18\\_notion\\_backend](https://labs.relbis.com/blog/2024-04-18_notion_backend)

Notion Help page. 2024. Security practices [Online] Available at: <https://www.notion.so/help/security-and-privacy>

AWS services referenced. [Online]: <https://aws.amazon.com/rds/postgresql/> , <https://aws.amazon.com/s3/> , <https://aws.amazon.com/what-is/apache-spark/>

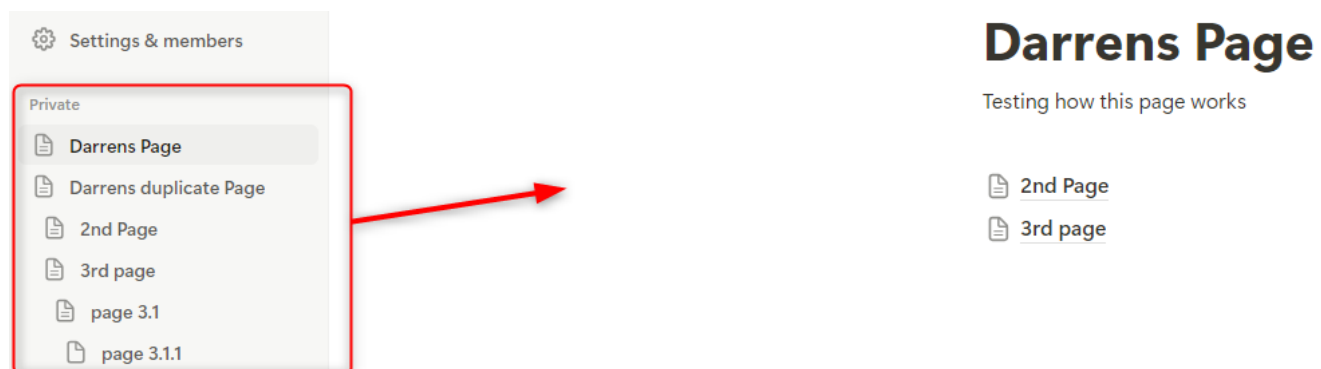
3. The interaction between Notions app technologies starts with the front end and users inputing their data into Notions user interface. Their user interface is primarily built using React and its javascript libraries, which are combined to create dynamic and interactive web app pages. I cant exactly find out the way Notion have deployed Redux to interact within their apps environment as it can have many uses, but most use Redux when you have a complex state management scenario, with many components that need access to a shared state. It's particularly helpful in large-scale applications where the patterns and tools provided by Redux make it easier to understand when, where, why, and how the

state in your application is being updated, and how your application logic will behave when those changes occur.

Then the interface will use Notions dedicated APIs which will be a webserver(s) using the Express.js framework sitting on top of the Node.js cross platform runtime environment. Node is used to construct Notions API endpoints with Express managing the server side routing. This setup allows the application to handle large scale requests in the background without any lag, ensuring real time updates and a smoother experience for the users. Node.js ability to scale up very quickly also ensures that as Notions use grows in popularity the application can still deliver reliability and performance.

Next is the Notion PostgreSQL databases with its high performance abilities to store and manage millions of users data in the forms of blocks, files, pages and spaces efficiently. Again Notion have chosen these types of databases for their scalability aspects and from one of there tech bloggs have demonstrated their database sharding strategy. Because their block data in Postgres was doubling every 6-12 months they started horizontally sharding each Postgres physical instance. In 2023 they increased their physical instances to 96 with 5 logical shards per instance, thus the Notion data lake comprised of 480 logical shards.

Note for Answers 4,5,6 & 7 = Notions application has many different use cases for multiple different individuals or business environments. For these answers I have only specifcily researched the use case of a individual user accessing one workspace environment to create their own pages and content. Example:

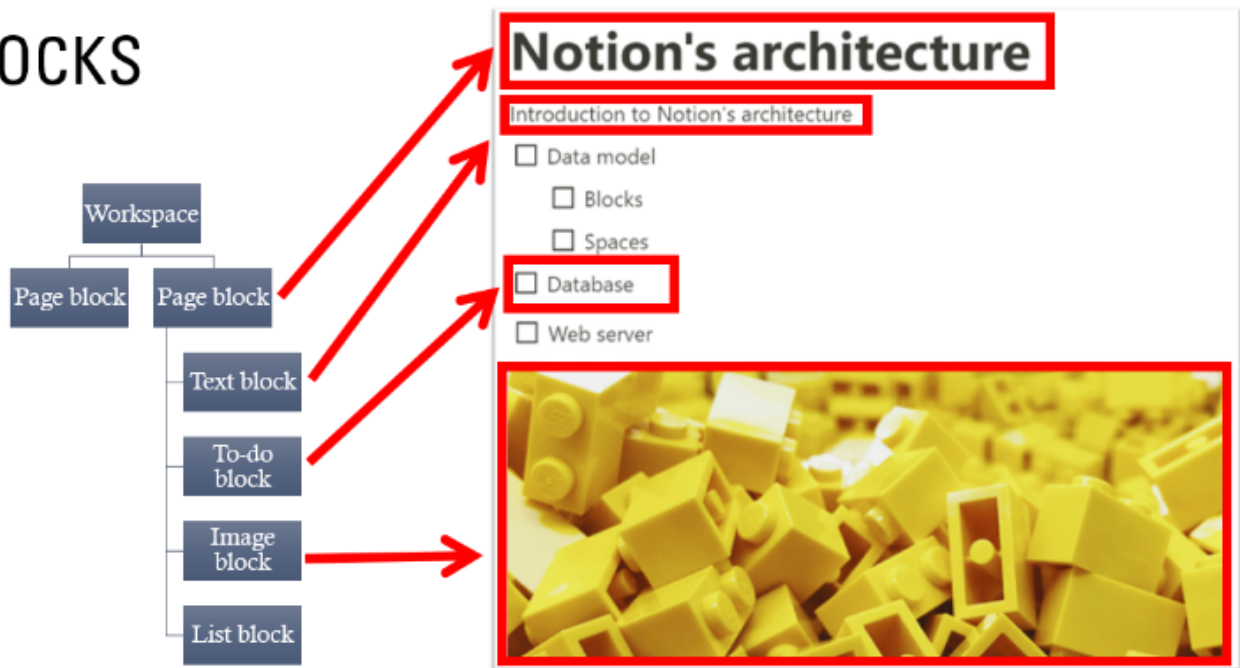


4. Notion have structured their data inside the databases using what they call a block model. From their Notion tech blogg page, they have written a number of blogg posts to explain / describe this data structure (references below). Everything users see or create in the Notion app like pages, text, to do points, images and lists is modeled as a different blocks. These 'blocks' are entities in the backend stored in a specific Postgres database.

The data model structure of the entities within a database are a versatile hierarchical structure similar to a tree model from where it starts with a main / 'Parent' entity. From their a 'Child' entity is linked to the main / 'Parent' entity, then below this multiple sub entities are linked or not to the 'Child' entity depending on the use case of each user and their data entry inputed. Each of these entities represents certain users pages and inputed data from the front end and are explained in question 5 answers.

Notion have designed this block model structure so that all their users information can stand on its own and have the ability to customize how it is organized and shared. A basic diagram image of Notions entity data structure model is here:

# BLOCKS



Reference:

Tech Blogg. 2024. Building and scaling Notion's data lake [Online] Available at: <https://www.notion.so/blog/building-and-scaling-notions-data-lake>

Tech Blogg. 2024. The data model behind Notion's flexibility [Online] Available at: <https://www.notion.so/blog/data-model-behind-notion>

Labs Relbis. 2024. Examining Notion's Backend Architecture [Online] Available at: [https://labs.relbis.com/blog/2024-04-18\\_notion\\_backend#data-model](https://labs.relbis.com/blog/2024-04-18_notion_backend#data-model)

5. The entities in a Notion database are specifically designed to reflect the users workspace design. Notion have allocated each entity to be a specific block of user entered data. For example in the above data model image, each entity will represent and most likely named as users\_workspace, pages, text, to\_do\_points, images and lists. All these entities and their attributes are listed below:

- The first would be the main / 'Parent' entity allocated as the users workspace (users\_workspace) and will contain attributes reflecting the users details like: user\_id, user\_name, user\_email, user\_login, user\_password.
- The next entity would be the users pages created (pages) and will contain attributes reflecting the users pages and the content added like: id, type, title, parent\_id.
- The following sub entities will contain user entered information from all the users pages created:
  - text entity will contain the text the user has added as content for each page to be displayed, attributes like: text\_id, text\_content, page\_id.
  - to\_do\_points entity will contain the text of multiple points the user has assigned to each page, attributes like: todo\_id, todo\_text, page\_id.
  - images entity will contain the saved image assigned to a page to be displayed, attributes like: image\_id, image, page\_id.
  - lists entity will contain any list items saved in a page to be displayed, attributes like: list\_id, list\_item, page\_id.

6. Because as mentioned above Notion have structured their databases as a versatile hierarchical structure similar to a tree model, this means the relationships between the database entities is very specific. One users\_workspace entity can have a relationship with multiple user pages and then each of the individual pages created can have multiple attributes added to them from the sub entities. These relationships require each entity to have a primary key and a foreign key. These keys in each entity are as follows:

- users\_workspace entity is the main 'Parent' entity and contains:
    - primary key = user\_id.
  - pages entity as a 'Child' of the user\_workspace will contain:
    - primary key = id.
    - foreign key = parent\_id.
  - the sub entities will contain:
    - text entity:
      - primary key = text\_id
      - foreign key = page\_id
    - to\_do\_points entity:
      - primary key = todo\_id
      - foreign key = page\_id
    - images entity:
      - primary key = image\_id
      - foreign key = page\_id
    - lists entity:
      - primary key = list\_id
      - foreign key = page\_id
- So to define each of the relationships between all entities would be like this:
- users\_workspace to pages : One to Many
  - pages to text: One to Many
  - pages to to\_do\_points: One to Many
  - pages to images: One to Many
  - pages to lists: One to Many

7. I have designed the following normalised ERD diagram of the Notion application researched in Answers 12.1,2,3,4,5,6 above:

