Tony Le Huynh - Work Book A

GCAB012201

Q1.

One of the most common industry accepted frameworks for typical Flask Web Applications, is called the MVC paradigm. This is where web applications are separated into different modules of an app, known as modularisation, based on the tasks that they complete (Reference 7).

The three main components which MVC architecture separates a Flask application are (Reference 25):

1. **Model** - Where data is managed
2. **View** - How the app's data should be displayed
3. **Controlle**r - Routes commands to the model and view

As an example, a user will request to view a page of a web application. This means a HTTP request is made resulting in a request route that is attached directly to a controller, and is pattern matched by the server (Reference 7).

The controller is responsible for business logic, is essentially the link between the view and model and also handles user interactions. This is where user's requests are handled and the model and/or view components are updated accordingly in response to input from the users of the app (Reference 25). For Flask web applications, the controller is implemented using routes and route handlers.

Though the controller can sometimes update the view component directly, it will typically first interact with the model component.

The model is the component with a direct connection to a database, thus being responsible for managing data. Interaction with a database involves retrieving, inserting and updating data (Reference 25). The data managed can also come from an API or JSON object (Reference 26). For Flask applications, an ORM (Object Relational Mapping) library will be used to interact between the web application and database, with a popular ORM library being SQLAlchemy (Reference 7).

Once all the necessary data is retrieved and organised, the model component then interacts with the view component.

The view component defines how the app's data is displayed to the user (Reference 25). Based on the data provided by the model component, the view renders the requested HTTP page. It is then the controller's responsibility to return the HTTP response sent back to the user, which is displayed to the user in their browser (Reference 27).

Q2.

A database management system (DBMS) is system software for users to create, manage and manipulate databases, as well as store data (Reference 1). One such DBMS that is utilised in web applications, including Flask frameworks, is MySQL. MySQL is a popular relational database management system (RDBMS) that is developed and supported by Oracle Corporation (Reference 2).

There are many reasons why MySQL is commonly used in web applications and also included below are the pros of this DBMS.

MySQL is a RDBMS, meaning that it is a relational database that stores data in separate tables to be used in relation to other stored datasets (Reference 6). A huge benefit of MySQL is that it is scalable, open-source and free, meaning that it is an ideal cost-effective option for startups and small businesses, as opposed to using the company budget on DBMSs with paid commercial licenses (Reference 3). Another advantage of MySQL is its ease of use and simplicity, which means it is more forgiving to use as opposed to other DBMS (Reference 5). With MySQL being open-source, this means there are numerous third-party tools which can be utilised to easily expand MySQL, and there are also large numbers of users that have made resources online to help solve any potential encountered problems whilst using MySQL (Reference 5).

Despite the numerous advantages of MySQL, there are a few cons to note.One such disadvantage is that MySQL is not very efficient in handling huge databases, and as such, not as powerful as other DBMS with large volumes of data (Reference 3). Because transactions aren't handled efficiently, MySQL is also prone to data corruption. Paid databases have developing and debugging tools, as well as official tech support, in place whereas it can be a challenge to debug stored procedures in MySQL. Lastly, MySQL is not fully SQL compliant, meaning some developers can find it difficult to adjust to the syntax of SQL while using MySQL (Reference 4).

Q3.

Agile is a type of project management methodology commonly utilized for managing software development projects. Agile methodology places an emphasis on flexibility, iteration and collaboration, where the focus is to deliver working software frequently throughout the implementation and completion of a project. It consists of a series of tasks which are first created, executed and then adapted to suit any situation demands (Reference 7).

In essence, Agile project management methodology is about iteration. Large projects are broken up into smaller tasks and chunks, and the developer team can iterate on these tasks. Once a certain task or goal is completed, the team meets together with any key stakeholders to review the task and make any required iterations. At these regular intervals, the team can also make reflections on how to make the next iteration of the Agile cycle more effective, as well as return to any previous older features of software that need updating due to new requirements (Reference 7). This is different to other methodologies, such as waterfall, where once a goal has been met the team moves on and doesn't return back to the previous step.

As discussed above, Agile methodology is a very useful approach for dynamic environments with evolving requirements which developer teams are faced with, and should certainly be used to create the web application project.

To start the project, the Team must first come together to define what is required for the software. This is where the concept of user stories comes into play. By utilizing user stories, the Team can think from the point of view of a customer to consider what the customer is looking for to be able to use the software, as well as the characteristics of these target customers. This customer-centric approach allows the Team to define what requirements must be met to deliver working software to the customer and ensure a great customer/user experience. This is what a common structure for a user story would look like: “As a _____, I want _____, so that _____.” (Reference 7).

Once the requirements of the software application are defined, the Team can come together to break down the project into smaller manageable tasks to be completed. This is where another aspect of Agile methodology comes in, which is the use of Kanban boards, such as tools like JIRA or Trello. Kanban is a tool that allows for easy visualization for all the team members in the project to see all tasks to complete (Reference 7). Tasks can represent specific features or functionalities of the project which need to be delivered. Each task is then sorted into various columns. While each project would have different requirements, an example of columns that would be used for the Kanban board would include (Reference 7):

1. Backlog - This column would contain a list of all tasks that have not yet been started. Any new tasks that come up throughout the project are added to this column as well.
2. In progress - This column contains the list of tasks that are currently being worked on in the current sprint/cycle.

3. Needs review - This column would contain any tasks that require reviewing from other team members/stakeholders
4. Done - This column contains the list of tasks that have been completed. This allows for the team to easily see what has been done, as well as revisit any previously completed tasks.

With the use of user stories and kanban, the developer team can define what the initial requirements of the web application are, as well as the smaller tasks which need to be completed to finish the project. It is important to clarify that requirements can be subject to change, which means the team may be required to revisit previous tasks that require updating.

Delivering working software frequently is the primary measure of progress for Agile. This is where the core Agile concept of sprints is utilized to achieve this. Large software projects are broken up into cycles, or sprints, which typically have a duration of 2 weeks (Reference 7). This is so that tasks can be completed, reviewed and reflected upon in a timely manner.

Task prioritization is key to ensure the project is completed efficiently, ensuring that the essential features of the application are done first. Things to take into account would be identifying what are the important tasks to ensure a “minimum viable product” of the project is done, the length of time that a task may take, which tasks are dependent on others being done first and which tasks which are not as essential though they may be “nice to have” for a project (Reference 5).

At the start of each sprint, the team comes together to plan what the objectives and goals are for the sprint. The team then prioritizes each task based on importance and complexity, which are then assigned to individual team members. The team can do this by using the kanban boards mentioned previously, where tasks to be completed for the particular sprint will be moved to the “In progress” column.

Each team member will then work on their assigned tasks during the sprint, with a focus on delivering working software at the end of the sprint. During this time, code is committed frequently as well as the automation of testing of software (Reference 9).

Communication and collaboration is also a key part of the execution phase of sprints. This is where another key Agile concept of stand-ups comes in (Reference 7). Stand-up meetings, which are often daily, allow for teams to check in with each other regarding the progress of their work for the sprint. This holds team members accountable by verbalizing any achievements, as well as allowing for team members to understand what each individual is currently working on. Stand-ups are also an opportunity for team members to raise any concerns or problems that they are currently facing, which allows for others to provide assistance and solutions where needed. Stand-ups are to be brief, and serve as an excellent opportunity for the team members to communicate with each other effectively (Reference 7).

At the end of each sprint entails a retrospective phase or sprint review, where the team can check-in with each other, review their work, reflect on the past sprint, make any adjustments,

receive feedback and identify any areas for improvement to take into account for the next sprint. These are different to stand-ups as they serve to reflect on the sprint as a whole. Questions to discuss for team members during this retrospective would include (Reference 7):

1. What worked well?
2. What could be improved?
3. What will we commit to doing in the next sprint?

Feedback from other team members, key stakeholders as well as customers can be take into account and analyzed at the end of the sprint.

With incremental project iterations, called sprints (consisting of the three phases - planning, execution and review) the team can ensure that each continual iteration of the Agile project management cycle keeps improving. This also displays how not all the planning needs to occur at the beginning of the project which is typical of traditional approaches, where Agile project management methodology allows for teams to constantly review, adapt and adjust to deliver working software while meeting user needs and evolving requirements.

Q4.

Source control, or version control, is a vital process utilized for developer teams. Source control allows for multiple developers to back up, manage and track changes to a project's software code over time (Reference 7). Source control management systems can allow developers to see the history of code development as well as for the merging of developer's changes and contributions from multiple sources into a centralised project codebase (Reference 8).

With access to the revision history of code, developers can trouble-shoot any issues or mistakes by identifying who made certain changes and what those changes were. The developer team can then utilise source control systems to revert to earlier versions of a project when required as well as test different versions of the project source code (Reference 9). This shows how source control can allow for developer teams to easily collaborate whilst protecting the source code of a project from any issues such as human error.

Developers usually organise the code for a project into a folder structure or "file tree", which allows for individual developers on a team to make changes to different features or parts of the file tree at the same time (Reference 9).

Git is a commonly used source control management system used by developer teams. Here is an example of a simple standard source control workflow using Git (Reference 10):

1. The main codebase or “master” branch used to track the source code for the project is first established
2. Individual members of the developer team will make their own local copy (or clone) of the source code and then create a new “branch”, where they will work on a specific feature or aspect of the project code
3. The developers will commit any changes into that branch
4. When they have finished working on the feature, the developer can then test and review their changes locally and in a test environment
5. After the testing has been complete and the changes are ready to be merged into the main source code, the developer can then push their local branch to the remote repository and also create a “pull request”
6. This “pull request” is then reviewed by other team, who can suggest any edits and provide feedback
7. Once all changes or edits are approved by the team, the changes are then merged into the main codebase, or “master” branch.

Q5.

Testing is crucial, as part of the software development process, to ensure that your code and the software application that you are working on is functioning as intended and meet the planned requirements. Testing is also important to identify any issues, bugs or any areas that need further work, so that these can be corrected so that quality working software is the final output (Reference 7).

A testing process for software development will also involve (Reference 7):

1. Test case: A situation or scenario that is being tested
2. Test suite: A collection of test cases
3. Test setup: Tasks which are executed before or after testing such as creating some data to be utilised for the testing. This also includes creating test cases and test scripts which will be used to test the code.
4. Test result: The output of a test case. The test case should display the results of the testing and also reveal the information and reasons behind if a test fails.

There are two types of testing that are utilised in software development, which are manual testing and automated testing. Manual testing is where a person is manually performing tasks such as a person manually testing a website’s features are working as intended. Automated testing is where programmed tests are written to automatically test code and perform tasks without requiring human involvement (Reference 7).

Unit testing is a type of standard automated software testing process. It involves specific and focussed testing of individual units or components of software (Reference 11). These small units of code are things such as a function or a method, and are tested in isolation from the rest of the application (Reference 11). Unit testing is typically executed frequently and early in the

development process to ensure that any changes to code don't break existing functionality, and that the individual software units/components being tested are working as intended without any bugs (Reference 11).

An example of unit testing can be where a unit test is written to test a function that calculates the sum of two numbers when given inputs. The unit test would verify to ensure that the function returns the expected result, which is the correct sum of the two numbers, when given a valid input. The unit test should also fail as predicted when provided an invalid input. This unit test case can be run when code is updated to ensure that the function continues working properly.

Q6.

Information system security is a critical aspect of any project, especially when building and running a web application. Cyber crime is an issue faced by many organizations, where hackers can gain unauthorised access to data and systems and cause damage. This shows how it's more important than ever in this day and age to have robust security measures in place to protect sensitive information and prevent unauthorised access (Reference 13).

Information system or application security is the practice of minimising vulnerabilities in software through establishing security measures. Web applications are exposed to a variety of risks as they must allow for connections from clients across unsecured networks. API security flaws can also include insufficient authentication and unintended data disclosure (Reference 12). Thus, it is important to ensure that security measures are followed from across all stages from the initial design, software building, deploying and maintaining a web application.

Ensuring the security of the application can prevent malicious parties exploiting any vulnerabilities and accessing data or code. Failing to protect the software and systems can result in loss of finances, trust, brand image, data, intellectual information as well as potential penalties due to breach of regulations (Reference 12)..

Three types of obligations to consider for information system security are - business, regulatory and customer obligations (Reference 14).

1. **Business obligations** are the security commitments of the company. In this instance, ACME corporation would have certain guidelines in place to ensure that information in the business - such as employee files, sensitive company data, intellectual information - are kept secure.
2. **Regulatory obligations** are the legal obligations that the company must meet. An example of this is the Privacy Act 1999 which is an Australia that details how companies should approach handling and protecting personal information about individuals (Reference 15). Another such regulation is the General Data Protection Regulation (GDPR) which is a set of enforceable and uniform requirements for protecting personal information and data of users (Reference 16).

3. **Customer obligations** are the security commitments that the company has to the customer/user to ensure that their information is protected securely.

There are also 6 pillars that should be considered for information system security of web applications (Reference 13):

1. **Authentication** - Identifying the the correct users are utilising the system through means such as login details via username and password
2. **Authorisation** - Ensuring that only certain users have the necessary authorisation and permissions for their use within the system
3. **Data processing** - Ensuring that data from user input are in the intended data formats as opposed to malicious data
4. **Encryption** - Sealing and locking sensitive data through means of data encryption
5. **Logging** - Monitoring things such as user activity to identify any potential malicious activities
6. **Application security testing** - Employing testing procedures to ensure that the security of the application, code and systems are able to handle break-in attempts. Types of security tools that can also assist with this are Software Composition Analysis (SCA), Static Application Security Testing (SAST), Dynamic Application Security Testing (DAST) and Interactive Application Security Testing (IAST).

By taking into consideration the three areas of obligations as well as the 6 pillars of application security, the developer team can incorporate these considerations into the development of the web application and information system security.

Further practices and requirements that the developer team can follow for the information system security of the web application include (Reference 12):

- Gaining knowledge and appreciation for top cyber threats. OWASP (Open Web Application Security Project) identify 10 globally recognised cyber threats and update this list regularly.
- Establish security requirements of the web application at the start of the Software Development Life Cycle which are to be followed throughout
- Detail in reports arrears of potential vulnerabilities and how to minimise or mitigate risks
- Ensure security at various levels such as enforcing secure coding practices for input validation, authentication and authorisation as well as a code review process which takes into account security

The Australian Cyber Security Centre also provides their Information Security Manual (ISM) which outlines a framework which organisations can utilise and adhere to to prevent cyber attacks and protect their systems (Reference 17). The developer team can also refer to the ISM when implementing measures for information system security.

Based on the points discussed above, there are a few measures which can be utilised in the Flask application that ACME will build, in order to cover the 6 pillars of information security - Authentication, Authorisation, Data processing, Encryption, Logging and Application Testing.

Firstly it is important to cover authorisation by limiting and controlling user permissions in cloud or source control services used by ACME.

In terms of security within the application itself, there are a number of Flask packages that are available and can be utilised to cover off on the security pillars. These should be used in conjunction with each other to cover specific security needs of ACME and the application.

In terms of authentication and authorisation, Flask-Security is a viable package to be used for these purposes (Reference 32). This is because Flask-Security allows for these security mechanisms to be added to the web application - session based user authentication, role management for authorisation, password hashing and two-factor authentication via token based authentication (Reference 32). Other packages such as Flask-Login (user session management), passlib (Python password hashing library) and Flask-Principal should be used in conjunction with Flask-Security in order for these comprehensive security options to be available. This shows how Flask-Security is a great option as a package to be used for the web app built by ACME to help cover the security pillars of authentication and authorisation.

There are a few options for covering the data processing pillar of information security - preventing malicious input into the application. A great option to consider is the use of a third-party Web Application Firewall (WAF) to protect the ACME web application. A WAF is a security solution that can filter incoming HTTP traffic, helping to protect the application from potential attacks such as a SQL injection (Reference 20). Flask-Talisman is also a Flask Extension that works in a similar fashion where HTTP headers are set to help protect against common web application security concerns (Reference 33), and should certainly be used for the ACME web application.

Flask-WTF is a package that is also a great option to help with this aspect of security. The extension uses the python WTForms library which provides data validation, which validates the data the user submits to meet the criteria defined by the developers. Furthermore, WTForms offers protection for the web application from CSRF (Cross-Site Request Forgery) attacks (Reference 34). Therefore, Flask-WTF is a package that should be used for the ACME application.

Security requirements for data encryption can be covered by Python packages such as passlib and cryptography. Passlib is a password hashing library, which provides password hashing for applications and cryptography is a package that allows you to use cryptographic algorithms, including symmetric and asymmetric encryption, to encrypt data (Reference 33).

Flask-Session and Flask-Login are packages which would help cover the security requirements of logging and monitoring user activity. Flask-Login is a package that provides the feature of

user session management for flask, and handling the task of saving users' sessions over extended time periods (Reference 35). Flask-Session adds support for Server-side Session in a web application (Reference 11).

In terms of the security pillar requirement of application testing, there are a few options. Pytest is a popular python package that can be used by the developer team to test the code written (Reference 36). Features from the pytest package such as being able to run unit tests, will help ensure that functions and code are working as intended for the web application. Also, Flask-WebTest is a package that should be used, as it is an extension that helps test web applications (Reference 37). Flask-Unsign is also a package that can be used by the developer team, as it is a penetration testing utility (Reference 33). Aside from these Python packages, ACME Corporation should consider using third-party services for the security testing of the web application.

Q7.

There are many common methods of protecting information and data to ensure that they are protected from any cyber attacks.

The 6 pillars to integrate in robust information system security are (Reference 13):

1. Authentication
2. Authorisation
3. Data processing
4. Encryption
5. Logging
6. Application security testing

Applying these 6 pillars can inform on the methods used for information system security in the web application for ACME Corporation.

Authentication involves identifying users within a system. The first method of security for authentication would involve ensuring that all the system users (whether that be the developer team or other stakeholders) have secure login credentials with usernames and robust passwords. Business.gov.au recommends that passwords be a secure passphrase that are long, complex, unpredictable and unique. This means being at least 14 characters, having a mixture of capital and lowercase letters, numbers, special characters, not being reused for other accounts and consisting of a group of unrelated words (Reference 18). Other forms of authentication include session authentication, where a user will prove their identity by supplying a cookie saved by the user's browser (Reference 7). There is also Bearer authentication where a token is stored by the user and used to authenticate their identity (Reference 7). These authentication methods can be used to ensure that the correct users are verified by the web application. Covering the security requirement of authentication can be achieved in the web application through using the package Flask-Security.

The next layer of protection would be to use multi-factor authentication which is a verification security process requiring users to provide two or more proofs of their identity before accessing the system (Reference 19). This would involve things such as entering a second code generated on the user's personal mobile device or sent to their email, as well as fingerprint or facial recognition. By having multi-factor authentication in place, the team can ensure there is an extra layer of security to minimise any instances of unauthorised access (Reference 19). Multi-factor authentication can be achieved in the web application through using the package Flask-Security.

Authorisation involves determining user permissions within a system or application. An important guideline for teams to follow is ensuring that access to information, data, programs or certain aspects of a system should only be given to those who actually need it when they require them (Reference 12). This means limiting access privileges where only authorised users have access to sensitive information or certain programs. Limiting and controlling user permissions can help minimise any unwarranted access of data as well as instances of data being leaked. If any user accounts become compromised by hackers, they can also be limited due to limited access privileges being in place. The security requirement of authorisation can be achieved in the web application through using the package Flask-Security. Additionally, user permissions for cloud platforms, source control services and database management systems used by the web application should be configured by ACME for security.

Data processing is a principle to follow when considering security. There should be a system in place, as well as considerations whilst developing and coding the web application, to ensure that input of data from users or external sources are as expected, as opposed to potentially malicious data. For example, a coding error could allow unverified inputs resulting in SQL injection attacks by hackers (Reference 13). The security pillar of data processing can be covered by packages such as Flask-Talisman and Flask-WTF, which help protect the application from attacks and security concerns.

An additional option for protection would be for the implementation of a web application firewall (WAF). A WAF is a security measure that helps protect web applications by filtering and monitoring HTTP traffic between the web application and the internet, thus protecting the web application from potential attacks (Reference 20). Having antivirus and firewall software installed on devices used by ACME Corporation can also help prevent any devices becoming compromised.

Encryption involves sealing and locking data, where data is transformed into an incomprehensible format as secret code, known as ciphertext, and the only way to access this data is to decrypt it with a cryptographic key (Reference 19). Examples of encryption algorithms used to alter data into ciphertext include Triple Data Encryption Standard (3DES), Advanced Encryption Standard (AES), Rivest, Shamir and Adleman encryption (RSA), Twofish encryption and RC4 encryption (Reference 21). It is important that encryption is utilised by ACME Corporation to ensure that sensitive data and information cannot be stolen or read by those with

malicious purposes. The python package ‘cryptography’ is a great package that allows the use of encryption algorithms for data in the ACME web application.

Logging involves having a system in place that will monitor and track user activity (Reference 12). A security measure can include having automated alerts in place based on certain thresholds or activity. This can help the developer team to identify malicious users as well as any security issues within the web application. Flask-Session as well as Flask-Login are packages that can be used by the ACME web application that can help monitor user activity.

Application security testing involves the testing of the web application and software to ensure they are safeguarded from break-in or hacking attempts. Four different types of security tools can assist with this process:

1. SCA: Software Composition Analysis
2. SAST: Static Application Security Testing
3. DAST: Dynamic Application Security Testing
4. IAST: Interactive Application Security Testing

SCA tools analyse external libraries and software components utilised by the application and monitor these for vulnerabilities. SAST tools perform a security scanning of code as it is written. DAST tools analyse running code and identify issues with requests and responses, any interfaces, scripts, injections, authentication and sessions. IAST tools scan code while it is running and while users are using the application, and perform testing on application and data flow (Reference 12). Packages such as pytest can be used to test code written while packages such as Flask-WebTest can assist with testing the application. It is also recommended to consider third-party services for testing the security of the ACME web application.

On top of these 6 information system security pillars mentioned above, there are also some methods recommended by the Australian Government to protect the company from cyber threats (Reference 19):

- Backing up the company’s data regularly so that information can be recovered if there are any incidents of cyber attacks or computer issues
- Ensuring software is kept up to date as obsolete software is a security risk
- Having spam filters in place to reduce any malicious spam or phishing emails received by the company
- Keeping track of all computer equipment and software used by the team to prevent unathorised access
- Implementing cyber security policies and protocol to help train and guide team members
- Secure all wireless networking used by the team

Q8.

The primary legal obligation to comply with, in regards to handling user data, is the Privacy Act 1988, which is the national privacy law in Australia enforced by the Office of the Australian Information Commissioner (OAIC). The Privacy Act 1988 has guidelines which businesses and organisations must adhere to in regards to the collection, use and storage of personal information of their customers & users (Reference 19). Under the Privacy Act 1988, ACME corporation may be obligated to protect user data from theft, misuse, interference, loss, unauthorised access or modification, and when the business no longer has use for the user data, then their personal information must be destroyed or de-identified (Reference 19).

If ACME Corporation has annual turnover more than $3 million, then the company must comply with the Privacy Act (Reference 19). However, even if the annual turnover is less than this amount, ACME Corporation may be still required to comply with the Privacy Act 1988 depending on the business type and business activities.

Under the Privacy Act 1988, the business must (Reference 19 & 24):

1. Comply with the Australian Privacy Principles (APPs) provided by the OAIC. An example of an APP would be to ensure strong security measures are in place to protect user personal data handled by ACME Corporation. Another is that the company can only use the user data for marketing purposes only if certain conditions are met (Reference 24).

2. Have a clear privacy policy which users will consent to and see, which outlines the information collected as well as how it will be used and protected by ACME Corporation. The privacy policy must also be regularly reviewed and updated where necessary. The OAIC also has guidelines which ACME Corporation can follow when developing its privacy policy.

3. Comply with the Notifiable Data Breaches scheme - where if a data breach occurs involving user personal data and is likely to cause harm to the user, then ACME Corporation must notify both the individual involved and also the OAIC.

As a side note, the location of the company as well as of its users, and also the types of user data being collected, will determine what legal obligations will apply. For example, if the personal information collected is from users in California, then the California Consumer Privacy Act (CCPA) will apply whereas if the users are from the European Union, then the General Data Protection Regulation (GDPR) will need to be adhered to (Reference 22). In Australia, there is

also legislation based on the state or territory, such as the Information Privacy Act 2009 for Queensland, and Privacy and Personal Information Protection Act 1998 for New South Wales (Reference 23). There are also sector-specific legislations outlined for industries that the business operates under, such as telecommunications, health and banking (Reference 23).

To meet these legal requirements for the project -

1. It is important for ACME Corporation to seek legal advice to ensure the company is complying with all legal obligations that pertain to ACME Corporation. This involves following the guidelines established by the Privacy Act 1988, as well as any other international, state-specific or sector-specific regulations that apply.

2. Also ACME must consult with legal counsel to form a privacy policy, as per the guidelines provided by the OAIC.

3. Lastly, ACME must take the necessary steps to have strong security measures in place which protect user personal data.These include all of the security requirements and methods for web application security discussed in the previous sections. The main security pillars are authorisation, authentication, data processing, encryption, logging and application security testing. Having protocols, third-party applications, systems and installed packages in place which cover these aspects of security will assist ACME with ensuring user data protection. For example, the use of a Web Application Firewall will help prevent malicious attacks. User data can be encrypted through using Python packages such as ‘cryptography’. Also packages such as Flask-Security can be in place which cover user authorisation and authentication, helping ensure that no unauthorized access of user personal data occurs.

Q9.

The relational database model involves data that is organised and structured into one or more tables (or “relations”), with pre-defined relationships between them (Reference 8). Tables play the role of holding information about objects represented in a database, and are used to show the relations between the objects.

Each table has a unique name and contains rows and columns, where a row is also known as a record or tuple and columns can also be called fields or attributes (Reference 28). Rows in the table represent an instance of the relation, and can also be thought of as a collection of related values or instances of an object or entity, while each column is representative of an attribute of the relation or characteristic of the object (Reference 8).

Keys are also used to establish the links between tables in the relational database model. As a table cannot contain duplicate rows according to the model, each table will have a column or set

of columns which represent the primary key (Reference 29). Primary keys uniquely identify rows within a table.

Foreign keys are also used in tables, acting as a reference to a primary key of another existing table (Reference 28). They create relationships between tables by linking them with each other. Through the use of these foreign and primary keys, the relational database model can represent complex relationships between objects.

Q10.

It is important that data remains accurate, consistent and reliable in relational databases. Data integrity is enforced in the relational database model through integrity constraints. Integrity constraints are a set of rules that ensure any processes performed in the database, such as the insertion, deletion or updating of data, is done in such a way that data integrity is not affected (Reference 31). These include (Reference 30):

1. **Domain Integrity Constraint**: This can be defined as a rule that restricts the kind of values or attributes that a column can store in a table. For example, a column must only accept a specific and specified data type and only a range of values. If, as an example, there is a column (attribute) that contains integer values, then you will be unable to insert string values into it as it only accepts integer values as its domain. This ensures consistency of data in the tables of the relational database.

2. **Entity Integrity Constraint**: This is where every table in a relational database contains a unique primary key, and this primary key value cannot be null. This is because the primary key is used to identify individual rows in relation, meaning that if the primary key was null then the individual rows in a table would not be able to be identified. Entity Integrity Constraint ensures that each of the data records in a table are able to be identified and accessed individually.

3. **Referential Integrity Constraint**: This integrity constraint is a restriction on how foreign keys are used and is enforced when a foreign key of one table references the primary key of another table. If a foreign key in Table 1 refers to the primary key in Table 2, then every value of the foreign key in Table 1 must be null or be available in Table 2. This helps to maintain consistency between related tables.

4. **Key Integrity Constrain**t: This is where a column or set of columns of a table is designated as a key, ensuring that each row in a table is unique. This ensures that no two rows in the table for that column or set of columns contain the same duplicate

values. The Key Integrity constraint helps ensure the accuracy of data by preventing the insertion of duplicate values into a table.

These types of data integrity constraints can be enforced in a relational database. Database management systems often have default settings that enforce these integrity constraints, and the user can also define further rules by using specific SQL commands.

Q11.

As part of the relational database model, there are various ways in which data can be manipulated within a relational database - which include adding, changing, removing and retrieving data.

Structured Query Language (SQL) is the most commonly used relational database language which programmers utilise to manipulate data in relational databases. The four main operations used for data manipulation in SQL are (Reference 7):

1. SELECT - To query/retrieve data in the database
2. INSERT - To insert/add data into a table in the database
3. UPDATE - To update/change data in a table in the database
4. DELETE - To delete/remove data from a table in the database

Here are some examples of performing these above SQL commands to manipulate data (Reference 7):

1. **Retrieve data**: Using the “SELECT” statement to retrieve data from a table in the relational database.
2. **Add data**: Using the “INSERT INTO” SQL statement to add new data records or rows to a table in the relational database.
3. **Change/Update data**: Using the “UPDATE” statement to modify existing data records or rows in a table from the relational database.
4. **Remove data:** Using the “DELETE FROM” statement to delete data records or rows from a table in the relational database.

There are also further ways to manipulate data in relational databases. This includes using operations to join rows from multiple tables together, filter data records that meet certain criteria, sorting data in a particular desired order as well as performing calculations, such as averages

and sums on subsets of data (Reference 30). Here are the SQL statements used to achieve these operations (Reference 7):

- JOIN - To combine rows from two or more tables, based on a related column
- WHERE - Filters rows based on certain criteria
- GROUP BY - Forms groups of rows from tables with the same column value
- ORDER BY - This command is used to sort the resulting output by a specific order

Q12.

The chosen company for the conducted research is Uber.

a)

For the back-end of the web application, Uber primarily uses the programming languages Python and Golang (Reference 43).

Initially, the Uber web application was built using Python and Tornado for the back end. Tornado is a Python web framework and asynchronous networking library (Reference 46). Python is a programming language that is popularly used to build websites and software, for automation and for data science and analysis (Reference 47).

Over time, Uber split their codebase into microservices, thus replacing older Python code that was initially written (Reference 43). These older segments of Python code were rewritten in the programming language Golang, due to their native support for concurrency being ideal for most new performance-critical services and due to Golang providing efficiency, simplicity and runtime speed (Reference 43). Golang, or Go, is an open-source, compiled and statically typed programming language designed by Google (Reference 48). Similar to Python, Golang is also a programming language that can be used for multiple purposes such as web-development (for Uber's back-end of their web application), data science and cloud computing. Uber also utilises Amazon Web Services, which supports Golang.

Uber primarily uses the programming languages Java, Swift and Objective-C for the front-end of their web application, especially for their mobile app development (Reference 43). Most of Uber's user base as well as their drivers, primarily use the Uber application on their Android or iOS devices. Java is a commonly used programming language in coding web applications as well as coding for Android Mobile App Development (Reference 49). Uber utilises Swift as it is a compiled programming language designed by Apple to be used for the development of iOS applications (Reference 51). Objective-C is also a primary programming language, and a superset of the C programming language, and is used by developers when writing software for iOS (Reference 53). Objective-C is also used by Uber's iOS engineers for writing the development of the Uber iOS app.

Uber developers also use JavaScript to build data products as their core tools as well as for web front-end applications(Reference 43). JavaScript is a programming language that allows developers to implement complex features on web pages (Reference 52). Along with JavaScript are some JavaScript libraries and frameworks used by Uber. These include React, Flux, Node.js, Backbone.js and D3.js (Reference 43). React is a JavaScript library utilised for building user interfaces (UI) and their UI components (Reference 52). React is what is used by Uber for the front-end of their web application, with some React components also being used by Uber's Android and iOS engineers. Flux, D3.js and Backbone.js are also used in the development of

Uber's web front-end application (Reference 43). Flux is a JavaScript architecture designed to build client-side web apps and for UI, which runs on a unidirectional data flow and has a centralised dispatcher (Reference 54). D3.js is a JavaScript library for manipulating documents based on data (Reference 55). Backbone.js is a library for structuring JavaScript code (Reference 54).

Uber also states that the core of their web tech stack is built on top of Node.js (Reference 43). Node.js allows developers to create both front-end and back-end applications using JavaScript (Reference 48). It is an open-source, cross-platform JavaScript runtime environment and library for running web applications outside the client's browser, thus allowing Uber to share JavaScript code between client and server for the development of their web application (Reference 56).

Initially, Node.js was primarily used by Uber's Marketplace team (Reference 43). Marketplace is the frontmost end of the Uber engine, which funnels real-time requests and locations as well as housing logic for products such as UberRUSH and UberEATS (Reference 43). Uber states that its core trip execution engine was also originally written in Node.js due to its asynchronous primitives and simple, single-threaded processing, as well as giving the Uber developers the ability to manage large quantities of concurrent connections (Reference 43). Uber's frontline API for their mobile apps is written in Node.js, where the API routes incoming requests from their mobile clients to other APIs or services (Reference 43). Uber also uses NGINX for SSL termination, some authentication and also proxies to Uber's frontline API, where NGINX is a software used for features such as web serving, reversing proxying and caching (Reference 59). Uber has also stated that the systems that process pings from Uber riders and drivers in real-time are written in Node.js and Golang (Reference 43).

Stats are also generated using StatsD by Uber (Reference 43). StatsD is a network daemon that runs on the Node.js platform and is used as an industry-standard solution for monitoring applications and gathering custom metrics (Reference 49).

Another software previously used by Uber for their front-end is Mapbox. A primary function of the Uber application is for users and drivers utilising the map feature. Mapbox is an advanced map service API that allows developers to have access to comprehensive location information to create and customise dynamic and static maps (Reference 57). This map service is integrated into Uber's web and mobile applications (Reference 43). In the current information that is available, Uber is using Google Maps for this map feature, allowing users to enter their desired destinations, see their route as well as estimates of arrival times (Reference 58).

In Uber's infancy, they had used PostgreSQL as their database management system, however they transitioned away from PostgreSQL as they required increased data storage and lower system response times (Reference 43).
Uber has stated that they run on a hybrid cloud model, utilising a mix of both cloud providers and multiple active data centers (Reference 43). Currently, Uber uses MySQL, Riak, Redis, MongoDB, Hadoop and Cassandra (Reference 43).

Uber's application is also deployed in AWS (Reference 43). AWS, or Amazon Web Services, is a comprehensive cloud computing platform provided by Amazon, which allows Uber to store data on the AWS Cloud (Reference 56).

MySQL is a relational database management system and Uber has developed "Schemaless" which is their data storage architecture built on top of MySQL (Reference 43). The Schemaless approach which Uber developed for their data storage has allowed for better flexibility and scalability to manage Uber's rapidly increasing data storage needs, using a format that didn't require a predefined database scheme. Schemaless/MySQL is used by Uber for long-term data storage.

Riak and Cassandra meet Uber's high-availability and low-latency demands. NoSQL is an approach to designing databases which enables the storage and querying of data outside the traditional structures found in relational databases (Reference 61). Cassandra and Riak are both NoSQL distributed databases (Reference 60 & 62). MongoDB and Redis are also NoSQL databases, where MongoDB stores data on disk and Redis is an in-memory store with Redis being used by Uber for caching and queuing (Reference 62). As mentioned above, Uber uses a number of NoSQL databases such as Riak, Cassandra, Redis and MongoDB for their comprehensive data storage and manage requirements as a company.

Uber has also mentioned that they use a Hadoop warehouse for distributed storage and analytics for complex data (Reference 43). Hadoop is a framework that allows for the distributed processing of large data sets across clusters of computers (Reference 62).

As Uber's interact with each other and mobile devices, these interactions are valuable for internal uses such as debugging (Reference 43). For logging, Uber utilises multiple Kafka clusters. Kafka is a distributed system consisting of servers and clients that communicate via a TCP network protocol (Reference 63).

Uber also uses machine learning software for its web application and various services. Uber has stated that they use TensorFlow, PyTorch, Keras (Reference 43). Machine learning is integral for Uber's application as it offers things such as creating personalise experiences for riders based on their previous trip data and behaviour patterns, being able to use machine learning algorithms to for dynamic pricing which adjusts based on real-time supply and demand, for fraud detection as well as for analysing driving behaviour (Reference 43). TensorFlow is an open-source library developed for machine learning applications and Keras is a machine-learning API (Reference 56). Pytorch is also another open source machine learning framework based on the Python programming language and the Torch Library (an open-source machine learning library) (Reference 1).

From the information publicly available, Uber has used the payment gateways Braintree and Stripe for the processing of their payment transactions (Reference 42). Braintree is a payments platform that allows Uber to accept payments in their web application and Stripe is very much similar where Stripe is an online payment processing platform (Reference 64). These payment

platforms/gateways allow Ubers to accept payments from their customers who use credit cards, debit cards as well as other payment methods such as PayPal.

Uber has stated that they use the software Aurora and Mesos to run their microservices with consistent configurations scalably (Reference 43). Aurora is a Mesos framework that runs applications and is responsible to keep them running where in the case of failure, Aurora will reschedule the applications onto other machines (Reference 65).

In terms of continuous integration, Uber has used tools such as Jenkins to assist with its software development process (Reference 43). Jenkins is an open source automation server that provides plugins to aid in building, deploying and automating projects (Reference 66).

b)

Uber has stated in their Tech Blog that they use a hybrid cloud model, utilising a mix of both cloud providers and multiple active data centers to host their application (Reference 43).

It is worth noting that it has been recently announced in 2023 that Uber signed agreements with Oracle (Oracle Cloud Infrastructure) and Google (Google Cloud Platform) to migrate most of Uber’s data center contents onto the cloud (Reference 67).

A data center is a physical facility that contains the IT infrastructure for running and delivering applications and services, as well as for storing and managing the data that is associated with the applications (Reference 68). As a large company, Uber would own and operate multiple data centers, which would contain Uber's servers, storage devices as well as other hardware components.

In this case, servers are high-powered computers that store, process and manage network data and systems - thus playing a role in hosting in Uber’s application (Reference 69). Servers accept and respond to requests made from clients across a network. Servers contained in Uber’s data centers would host various systems and components of the Uber app.

Uber also utilises cloud platforms such as Amazon Web Services (AWS), and now Google Cloud Platform (GCP) and Oracle Cloud Infrastructure (OCI) to host its application (Reference 43). Hosting Uber’s application on cloud platforms helps UBer manage its computing resources and data storage without just relying on physical servers (Reference 70). Services such as AWS, GCI and OCI provide a network of virtual and physical cloud servers which can host Uber’s application. These cloud platform services also offer a range of other services such as computer, storage, databases, analytics, networking, developer tools and enterprise applications (Reference 49).

c)

Firstly, Uber is primarily used as a mobile app running on both Android and iOS devices. Users use the Uber app to request rides from Uber drivers, and are provided a trip to their desired destination with the app also providing information such as estimated travel time and the cost of their ride fare.

The map service as well as real-time GPS tracking of the Uber drivers is done through software such as Google Maps and MapBox, which are integrated with Uber's application (Reference 43).

Payment for trips would be processed through payment gateways that Uber utilises such as Stripe and Braintree.

Uber's backend servers play an integral role for the use of the app, as the application will communicate with the backend servers to retrieve data and perform actions. As mentioned previously, these backend servers would be hosted on cloud platforms that Uber uses such as AWS, GCP or OCI, as well as physical servers located in Uber's data centers (Reference 43). Things such as processing requests made by the application, ride requests, dispatching Uber drivers, processing payments and the storing of data would be managed through these backend servers of the Uber application.

An API gateway is a server that acts as a single point of entry for a set of microservices. It works by receiving client requests and then forwarding them to the appropriate microservice. After, it returns the server's response to the client. Therefore, API gateways are responsible for processes such as routing requests, authentication, enforcing rate limiting, caching, monitoring and logging (Reference 71). It is through the frontline API gateway of Uber's infrastructure, that the Uber web application will communicate with Uber's backend servers, as well as Uber's microservices which can also communicate with each other (Reference 43). Requests can also be routed to other APIs or services. The way Uber's infrastructure is designed is that Uber's application consists of many microservices which are responsible for specific processes. These processes can include things such as ride dispatching and ride fare calculation (Reference 43). Each of these microservices will communicate with each other as well as with the application through APIs, as part of the functioning of the Uber application.

As discussed previously, Uber uses multiple different databases to store its different types of data. Uber uses the relational database MySQL as well as the NoSQL databases such as Riak, Cassandra, Redis and MongoDB (Reference 43). The backend servers of Uber would play the role of communicating with these databases and retrieve the appropriate data from them.

Data analytics and machine learning are also integral to the functions of the Uber application. Kafka is a software that would be responsible for logging. Machine learning software used by Uber such as TensorFlow, PyTorch and Keras would also be utilised in the functioning of the

application. For example, the machine learning algorithms would play a role in dynamically setting pricing based on real-time demand and supply, as well as things such as predicting rider demand. These machine learning algorithms and software would be trained on the data stored in Uber's databases.

d)

Uber uses multiple different databases to store different types of data. Uber uses the relational database MySQL as well as the NoSQL databases such as Riak, Cassandra, Redis and MongoDB (Reference 43).

Uber has stated that they use MySQL for long term data storage. MySQL is a relational database management system, where data is stored in separate tables with pre-defined relations between them (Reference 6). Tables represent objects or entities in relational databases, and store data.

Given these qualities of relational databases, Uber would store the following types of data in MySQL for its application.

1. **User data**: Information about Uber's user base would be stored in a relational database. Data such as the user's details, name, home address, email address, password, phone number and payment information would be stored in the tables of MySQL
2. **Driver data**: This would be information on Uber's drivers. This would include data such as their name, contact details, vehicle type, licence plate number and payment details.
3. **Trips data**: This would be data on things such as pick-up and drop-off locations, trip durations, distance travelled and the trip fare amount.
4. **Payments data**: This would include data on payments made by users, payments made to Uber drivers and revenue generated for Uber
5. **Feedback data:** Uber has a driver and rider rating system, so this data based on 5 star ratings and written feedback would be stored

Uber has stated that they use Hadoop for distributed storage and analytics of complex data. Redis is a NoSQL database that is an in-memory store, and is thus used by Uber for caching and queuing (Reference 62).

They also stated that Riak and Cassandra meet high-availability, low-latency demands. This means that for these NoSQL databases that Uber uses, including MongoDB, they would handle unstructured data and store data that might not be used for long term storage.

NoSQL databases are non-relational databases, where rather than using tables with predefined rows and columns, data is instead stored differently than relational tables (Reference 62).Major types of NoSQL databases include document databases, where data is stored in documents

similar to JSON objects, key-value databases, where each item contains keys and values and graph databases, where data is stored in nodes and edges (Reference 62).

Real-time user and driver data would be such data types stored in these NoSQL databases. Examples of this data would be real-time locations of both users and drivers and traffic congestions and weather hazards such as flooding which would affect routes. Real-time events such as sporting events could also be stored as these would affect Uber trips. Therefore real-time calculations for estimated trip durations would be data stored that is unstructured. System log information about the Uber app such as bugs could also be stored in these NoSQL databases. Also support requests from both riders and drivers of the Uber app can also be stored as this unstructured data.

e)

There are a few main entities which need to be tracked by the Uber application.

Firstly the riders or users. This would include information such as their name, password, email address, phone number and payment information.

Secondly, the Uber drivers. Information would include driver details such as their name, contact information, age, vehicle information, payment information (BSB and Account number for payments from their Uber trips).

Another entity would be the trips themselves, such as data about trip routes (arrival and destination addresses), trip durations, distance traveled and trip fare.

Another entity would be the vehicles of the drivers. This would include vehicle details such as the make, model, license plate number, year and last date the vehicle was serviced.

Another entity could be a payment, which would have data such as the payment method used, date of transaction and amount paid.

Another entity type could also be location where data would be the addresses for pick-up and destinations. This would include details such as street address, city, state and country.

Other entities would include the ratings, which is the data about ratings regarding both the driver and the rider. This would include rating value (e.g. 5 stars), the trip ID and the driver and riders associated with that rating.

Another entity could include promotional offers, such as the promotional code, the discount percentage applied for the particular trip, and expiration date.

f)

Here are the relationships identified for entities previously mentioned.

Rider - Rating
An Uber rider could have multiple ratings from all their different trips, however only one rating can be associated with one rider.

Driver - Rating
Similar to the above, an Uber driver can have multiple ratings associated with the, however only one rating can be associated with one Uber driver.

Trip - Rating
Only one trip can be associated with one rating for the trip, and vice versa.

Trip - Promotion
Assuming that Uber only allows one promotion to be applied at a time, only one trip can be associated with one promotion at a time and vice versa.

Rider - Trip
A rider can have many trips, and each trip can be associated with many riders (as Uber has a rideshare feature)

Driver - Trip
A driver can have many trips, however each trip is only associated with one driver.

Rider - Payment
A rider can have many payments and payments can be associated with many riders (as Uber has a rideshare feature).

Driver - Payment
A driver can have many payments made to them, however each payment is only associated with one driver.

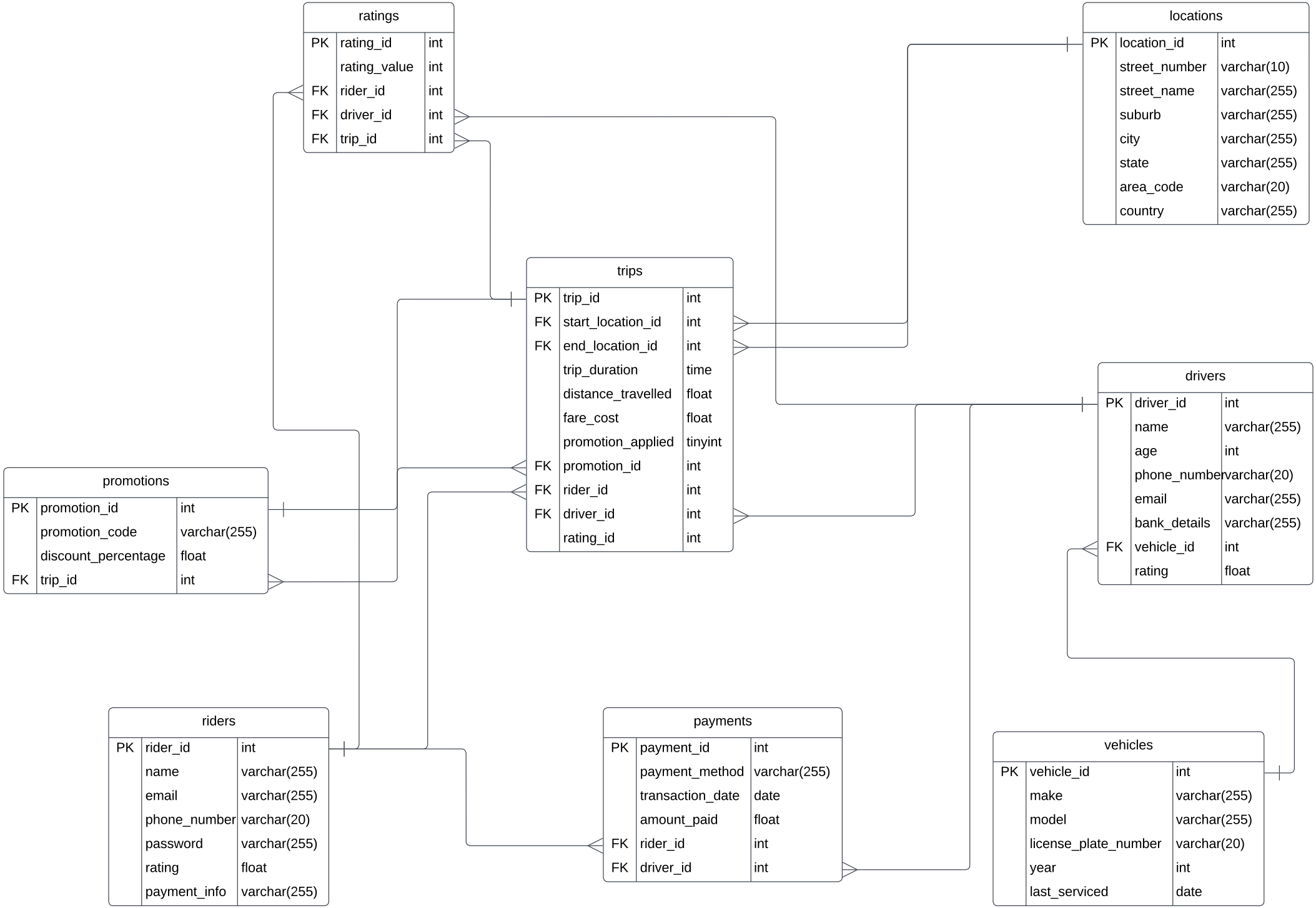
Rider - Driver
For each trip, there can be many riders associated with a driver (due to rideshare) however there can only be one driver associated per rider for the trip.

Rider - Location
A rider can have many locations (many different pickup and drop off locations) and these locations can also have many different riders. It is not a one-to-one relationship as there can be many riders at the same location.

Driver - Location

A driver can also have many locations (different pick up and drop off addresses), and these locations can also have many different drivers. It is not a one-to-one relationship as there can be many drivers at the same location.

g)


ratings
PK rating_id int
rating_value int
FK rider_id int
FK driver_id int
FK trip_id int
locations
PK location_id int
street_number varchar(10)
street_name varchar(255)
suburb varchar(255)
city varchar(255)
state varchar(255)
area_code varchar(20)
country varchar(255)
trips
PK trip_id int
FK start_location_id int
FK end_location_id int
trip_duration time
distance_travelled float
fare_cost float
promotion_applied tinyint
FK promotion_id int
FK rider_id int
FK driver_id int
rating_id int
drivers
PK driver_id int
name varchar(255)
age int
phone_number varchar(20)
email varchar(255)
bank_details varchar(255)
FK vehicle_id int
rating float
promotions
PK promotion_id int
promotion_code varchar(255)
discount_percentage float
FK trip_id int
riders
PK rider_id int
name varchar(255)
email varchar(255)
phone_number varchar(20)
password varchar(255)
rating float
payment_info varchar(255)
payments
PK payment_id int
payment_method varchar(255)
transaction_date date
amount_paid float
FK rider_id int
FK driver_id int
vehicles
PK vehicle_id int
make varchar(255)
model varchar(255)
license_plate_number varchar(20)
year int
last_serviced date

REFERENCE LIST

1. [Tech Target](https://www.techtarget.com/searchdatamanagement/definition/database-management-system)
2. [MySQL](https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html)
3. [W3schools](https://www.w3schools.blog/mysql-advantages-disadvantages)
4. [Simplilearn](https://www.simplilearn.com/tutorials/sql-tutorial/postgresql-vs-mysql)
5. [LinkedInLearning](https://www.youtube.com/watch?v=x6WQ9TTCD70&ab_channel=LinkedInLearning)
6. [SiSense](https://www.sisense.com/glossary/relational-database/)
7. [Ed Coder Academy](https://edstem.org/au/courses/10081/lessons/27591/slides/194990)
8. [AWS](https://aws.amazon.com/devops/source-control/)
9. [Atlassian](https://www.atlassian.com/git/tutorials/what-is-version-control)
10. [Build5Nines](https://build5nines.com/introduction-to-git-version-control-workflow/)
11. [Geeksforgeeks](https://www.geeksforgeeks.org/unit-testing-software-testing/)
12. [EC Council](https://www.youtube.com/watch?v=qqdMShR-FWw&ab_channel=ECCouncil)
13. [Vmware](https://www.vmware.com/topics/glossary/content/application-security.html)
14. (ProserveIT)[https://www.proserveit.com/blog/information-security-requirements]
15. (Find and Connect)[https://www.findandconnect.gov.au/ref/australia/biogs/FE00127b.htm]
16. [Nixora Group](https://www.dfat.gov.au/sites/default/files/nixora-group-eufta-submission.pdf)
17. [Australian Cyber Security Centre](https://www.cyber.gov.au/acsc/view-all-content/ism)
18. [Insurance Business Australia](https://www.insurancebusinessmag.com/au/news/breaking-news/ten-ways-to-protect-your-business-from-cyber-attacks-249955.aspx)
19. [Business.gov.au](https://business.gov.au/online/cyber-security/protect-your-business-from-cyber-threats#ensure-you-use-multi-factor-authentication-mfa)
20. [Cloudflare](https://www.cloudflare.com/en-gb/learning/ddos/glossary/web-application-firewall-waf/)
21. [Kaspersky](https://www.kaspersky.com.au/resource-center/definitions/encryption)
22. [Legal Vision](https://legalvision.com.au/privacy-laws-your-obligations-as-a-business/)
23. [ICLG](https://iclg.com/practice-areas/data-protection-laws-and-regulations/australia)
24. [OAIC](https://www.oaic.gov.au/privacy/australian-privacy-principles/australian-privacy-principles-quick-reference)
25. [Developer Mozilla](https://developer.mozilla.org/en-US/docs/Glossary/MVC)

26. [freeCodeCamp](https://www.freecodecamp.org/news/the-model-view-controller-pattern-mvc-architecture-and-frameworks-explained/)
27. [RealPython](https://realpython.com/the-model-view-controller-mvc-paradigm-summarized-with-legos/)
28. [Google Cloud](https://cloud.google.com/learn/what-is-a-relational-database)
29. [Nanyang Technological University](https://www3.ntu.edu.sg/home/ehchua/programming/sql/Relational_Database_Design.html)
30. [Techopedia](https://www.techopedia.com/definition/811/data-integrity-databases)
31. [Computer Science Explained Easy](https://www.youtube.com/watch?v=iwR9XNuLojs&ab_channel=ComputerScienceExplainedEasy)
32. [Python hosted](https://pythonhosted.org/Flask-Security/)
33. [Pypi](https://pypi.org/project/talisman/)
34. [DigitalOcean](https://www.digitalocean.com/community/tutorials/how-to-use-and-validate-web-forms-with-flask-wtf)
35. [Flask Login](https://flask-login.readthedocs.io/en/latest/)
36. [pytest](https://docs.pytest.org/en/7.2.x/)
37. [Flask WebTest](https://flask-webtest.readthedocs.io/en/latest/)
38. [stackshare](https://stackshare.io/uber-technologies/uber)
39. [Uber](https://www.uber.com/en-AU/blog/tech-stack-part-one-foundation/)
40. [emizentech](https://www.emizentech.com/blog/uber-tech-stack-software-architecture.html)
41. [theappsolutions](https://theappsolutions.com/blog/development/uber-tech-stack/)
42. [stackshare](https://stackshare.io/uber-technologies/uber)
43. [Uber](https://www.uber.com/en-AU/blog/tech-stack-part-one-foundation/)
44. [emizentech](https://www.emizentech.com/blog/uber-tech-stack-software-architecture.html)
45. [theappsolutions](https://theappsolutions.com/blog/development/uber-tech-stack/)
46. [tornado](https://www.tornadoweb.org/en/stable/)
47. [coursera](https://www.coursera.org/articles/what-is-python-used-for-a-beginners-guide-to-using-python)
48. [freecodecamp](https://www.freecodecamp.org/news/what-is-go-programming-language/)
49. [aws](https://aws.amazon.com/what-is/java/)
50. [AbhiAndroid](https://abhiandroid.com/java/)
51. [altexsoft](https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-swift-programming-language/)
52. [developerMozilla](https://developer.mozilla.org/en-US/docs/Learn/JavaScript/First_steps/What_is_JavaScript)
52. [w3schools](https://www.w3schools.com/whatis/whatis_react.asp)
53. [developer Apple](https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html)
54. [geekforgeeks](https://www.geeksforgeeks.org/javascript-architecture-flux-components/)

55.[d3js](https://d3js.org/)
56.[simplilearn](https://www.simplilearn.com/tutorials/nodejs-tutorial/what-is-nodejs)
57. [uptech](https://www.uptech.team/blog/mapbox-vs-google-maps-vs-openstreetmap)
58.
[cnbc](https://www.cnbc.com/2019/04/11/uber-paid-google-58-million-over-three-years-for-map-services.html)
59. [NGINX](https://www.nginx.com/resources/glossary/nginx/)
60. [Cassandra Apache](https://cassandra.apache.org/_/index.html)
61. [IBM](https://www.ibm.com/au-en/topics/nosql-databases)
62. [riak](https://riak.com/)
62.[MongoDB](https://www.mongodb.com/compare/mongodb-vs-redis#:~:text=MongoDB%20and%20Redis%20are%20modern,is%20an%20in%2Dmemory%20store)
62.[Apache Hadoop](https://hadoop.apache.org/)
63. [Kafka](https://kafka.apache.org/intro)
64. [Braintree](https://www.braintreepayments.com/au/faq)]
65. [Aurora](https://aurora.apache.org/)
66. [Jenkins](https://www.jenkins.io/)
67.
[Techspot](https://www.techspot.com/news/97603-uber-shut-down-own-data-centers-strikes-cloud.html)
68. [IBM](https://www.ibm.com/au-en/topics/data-centers)
69. [Serverwatch](https://www.serverwatch.com/guides/what-is-a-server/_)
70. [Business News Daily](https://www.businessnewsdaily.com/5030-what-is-cloud-hosting.html)
71.
[Medium](https://medium.com/geekculture/system-design-basics-what-is-an-api-gateway-b858e9491608)