Software Engineering - Internship Assignment

Name: Chandur Dissanayake

Written Questionnaire

1. Explaining what is design pattern and how we can use design patterns in projects.

Design patterns is an idea and best practice software developers follow in object oriented programing, providing reusable solutions to common problems that occur during software designing. The goal is to speed up the development workflow by following recognised design paradigms which are extensively tested. Being independent to programming language the usage of design patterns makes the code reusable, flexible, standardised and maintainable.

But how can we use design patterns for a particular project? First we need to know the types of design pattern and then evaluate what pattern is most suited to our project.

- Creational
    - Builder
    - Factory Method
    - Abstract Factory
    - Singleton
    - Prototype
- Behavioural
    - Template Method
    - Chain of Responsibility
    - Command
    - Strategy
    - Iterator
    - Visitor
    - Mediator
    - Mementor
    - Observer
    - Interpreter
    - State
- Structural
    - Flyweight
    - Composite

- Adapter
- Bridge
- Facade
- Decorator
- Proxy

Creational

This pattern is about how classes and objects are instantiated. Under creational we can have class creational and object creational. Class creational utilises inheritance and also to isolate the system from having any knowledge of how objects are created similar to encapsulation and abstraction while object creational effectively represents how a task is done.

Use case : When a user creates a database connection and desires to access the database from many locations of the code base. Typically what happens is developers would create multiple connection instances from wherever the database connection is needed having many instances being able connected to the same database. To tackle this we can create the database connection class as a singleton in which only one instance of the database connection is created allowing us to easily manage, load balance and omit unnecessary connections.

Structural

Organises classes and objects into large structures and identifies new ways to establish relationships with entities and produce new functionality.

Use case : When there are two interfaces which are required to establish connection through an adapter since they are incompatible we can use what's called adapter design pattern. Adapter pattern converts the interface of the class into another interface or class which the client expects to have.

Behavioural

Identifying common communication patterns between objects and how they interact with each other, the dependencies, how loosely coupled , maintainability and testability.

Use case : The template design pattern contains the structure of an algorithm in a function deferring some aspects to subclasses. Subclasses can re-define certain steps of the algorithm while not changing the structure of it. For example: if we want to extend a module we can do this as per required and meet the needs but we can't modify the structure only add functionality.

2. What is DTO and explain the use of it.

DTO or data transfer objects is an object created to transfer data in object oriented programming.

Let me give a real life example:
When marketing or sales professionals visit us, they would hand over their visiting card(which contains their name and contact information). Do we really care which factory printed the data or how did it happen? We don't, because we are interested in the visiting card information only. Here, a visiting card is a simple data container(data transfer object) for us that is used for passing necessary information to us.

DTO keeps the database/network layer isolated hence separate data related issues from UI or business layers.

Benefits of DTO:

- Hiding the complexity of data layer
- Complements UI or Unit testing by mocking DTOs and injecting data to them as dependency
- No direct dealing with erroneous data related to network or database.

3. How are you going to store secrets in an application without exposing it to the internet?

Usually there are several different ways in which keys are typically stored and all of the reasons I'm about to give you have one common issue and it is unsafe to keep your credentials.

- Storing secrets in your code
- Storing secrets in the environment
- Storing secrets in the database
- Using a secrets syncing service
- Storing secrets in your code as encrypted

All of these methods above are unsafe due to the fact that all these methods still involve having secrets on your server and those could leak or get stolen. For example: if an attacker finds a way into your system using a vulnerability, despite all your efforts in

keeping your secrets safe, the hacker can still browse in the server's memory dump to extract those secrets.

So then where do we store our secret keys? There are different ways which cater to different types of use cases so according to what suits best these methods can be used to store the secrets.

- Source control (github, bitbucket, gitlab)
- CI/CD tools (github actions, gitlab ci, jenkins, circle ci)
    - Suited for multi cloud infrastructure
    - Builds can be self sufficient, run anywhere and can deploy the applications anywhere.
    - Cloud credentials can be stored in CICD tool in order to deploy the builds
- Cloud (AWS secret manager/ Azure Key Vault, GCP secret Manager)
    - Most suited is the secrets are need at run time
    - Volatile secrets
    - Can update secrets at runtime and be used by other services
    - Very accessible (services should be restartable directly from the cloud, without needing to rely on the CI/CD or source control for secrets)
    - Suited for more complex pipelines and we can create environment specific secret managers having their own copy of secrets for all applications
- Third party key vaults (Hashicorp Vault)

So to summarise we can use a combination of all these methods, we can store universal constants in source control, cloud credentials in CI/CD tools and runtime or environment specific secrets in the cloud.

4. What is JWT and how does it work?

JWT or JSON Web Token is an open standard (RFC 7519) for securely transmitting information between endpoints as a JSON object. JWT has 3 components, Header, Payload and Signature each separated by a dot. For example : header.payload.signature

Header

The header specifies the algorithm and the type of the token. Example:

```
const header = {
  "alg": "HS256",
  "typ": "JWT"
};
const encodedHeader = base64url.encode(JSON.stringify(header));
```

Payload

This contains a set of claims and any metadata required by your application to identify the user. For example:

```
const payload = {
  "name": "Chandur",
  "iat": 1591024013,
  "exp": 1892023997
};
const encodedPayload = base64url.encode(JSON.stringify(payload));
```

This example has the standard Issued At Time claim (iat), Expiration Time claim (exp) and a custom claim name. The payload object is then stringified and base64Url encoded to form the second part of the JWT.

Signature

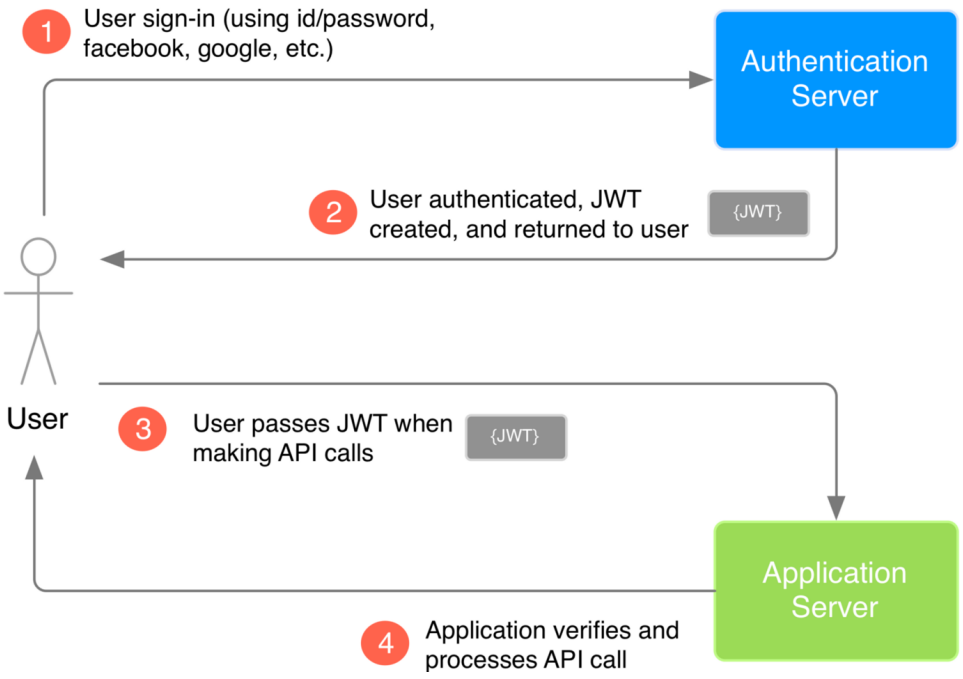
The signature is composed by taking the encoded header, the encoded payload, a secret string, and applying the algorithm specified in the header. If we use the HMAC SHA256 algorithm the signature will be created in the following way:

```
header = {...}
payload = {...}
encodedString = base64UrlEncode(header) + "." + base64UrlEncode(payload);
signature = base64UrlEncode(HMACSHA256(encodedString, 'secret'));
```

By adding a signature to our token string, the ownership and integrity of the provided token can be verified.

The JWT authentication workflow

1. The user enters the username and password to the login auth service.
2. The auth service validates the credentials and generates a JWT token signed with the secret string and a payload containing the user identifier and the expiration or duration timestamp.
3. The browser stores the token in local storage or cookies or anywhere it can be retrieved at a later time.
4. When the user wants to retrieve a protected resource for example : navigate to a protected page), the browser will need to include the token in each HTTP request to our server via the Authentication header.
5. The server checks if the signature is valid and then it generates a new signature from the header and payload of the provided token and then checks if it matches with the signature in the provided token.
6. The server gets the user identifier from the JWT token and starts processing the HTTP request accordingly.



5. What is the difference between SQL and NoSQL databases?

Difference between SQL & NoSQL

	A	B	C
1	**Key Areas**	**SQL**	**NoSQL**
2	Database type	Relational Database	Non-relational database
3	Schema	Predefined	Dynamic
4	Database Categories	Table based	Document based, Key-value stores, Graph stores, Wide column stores
5	Complex Queries	Good	Not good
6	Hierarchical Data Storage	Not good	Good
7	Scalability	Verticle	Horizontal
8	Language	Structured	Unstructured
9	Base Properties	Based on ACID properties	Based on CAP theorem
10	Examples	MySQL, PostgreSQL, Oracle, MS-SQL Server	MongoDB, GraphQL, HBase, Neo4j, Cassandra

6. Suggest a good state management for frontend application and explain why you recommend it.

There are many different state management tools and few of the most popular, used and maintained ones which I compared among each other to come to the conclusion was :

- Redux
- MobX
- Recoil

Recoil is very new and has no community support or ecosystem at the moment. Hence a production level React project cannot depend on a library with weak community support. Recoil is also said to be in an experimental stage making it unreliable.

MobX and Redux on the other hand has many people using it and is widely known in the community. However what makes the difference between the two is the learning curve when using each in the code. MobX requires reactive programming hence if the developer is not skilled there can be many inconsistencies, performance issues, development time and bugs created in the application.

Redux has some scalability and performance issues however there are solutions and improvements being made due to support unlike MobX.

If we discuss about the quality attributes of Redux:

- Usability - very simple, create a slice next pass it to the store and then access it in a component via hooks.
- Maintainability - redux is simple and doesn't require experience to fix something
- Performance - Proven ways to improve speed from official guidelines
- Testability - Redux consist of pure functions (actions and reducers) hence great for unit testing and have mechanisms to write integration tests.
- Scalability - Has redux-dynamic-modules library which enables creation of modular reducers and middleware.
- Modifiability - Supports middleware hence customizable
- Reusability - Framework is agnostic hence reusable
- Ecosystem - helpful addons, libraries and tools
- Community - largest community support
- Pulse - updated and maintained regularly

Below is a QR code of a animation of how react works in simple terms

References:

1. https://www.tutorialspoint.com/design_pattern/design_pattern_overview.htm
2. https://www.geeksforgeeks.org/design-patterns-set-1-introduction/
3. https://www.linkedin.com/pulse/understanding-state-management-front-end-paradigm-jitendrasinh-gohil/
4. https://medium.com/codex/tangled-with-front-end-state-management-use-the-store-86632c20f2ef
5. https://nitin15j.medium.com/frontend-state-management-44169546a2f7
6. https://openexpoeurope.com/en/the-10-best-tools-for-state-management-in-front-end/
7. https://www.integrate.io/blog/the-sql-vs-nosql-difference/#:~:text=SQL%20databases%20are%20vertically%20scalable,data%20like%20documents%20or%20JSON.
8. https://www.geeksforgeeks.org/difference-between-sql-and-nosql/
9. https://www.akana.com/blog/what-is-jwt#:~:text=JWT%2C%20or%20JSON%20Web%20Token,after%20the%20token%20is%20issued.
10. https://jwt.io/introduction
11. https://medium.com/poka-techblog/the-best-way-to-store-secrets-in-your-app-is-not-to-store-secrets-in-your-app-308a6807d3ed
12. https://blog.logrocket.com/best-practices-for-managing-and-storing-secrets-in-frontend-development/
13. https://www.freecodecamp.org/news/4-design-patterns-to-use-in-web-development/#:~:text=A%20design%20pattern%20is%20like,different%20sets%20of%20best%20practices.
14. https://www.tutorialspoint.com/design_pattern/design_pattern_overview.htm
15. https://sourcemaking.com/design_patterns
16. https://www.baeldung.com/java-dto-pattern