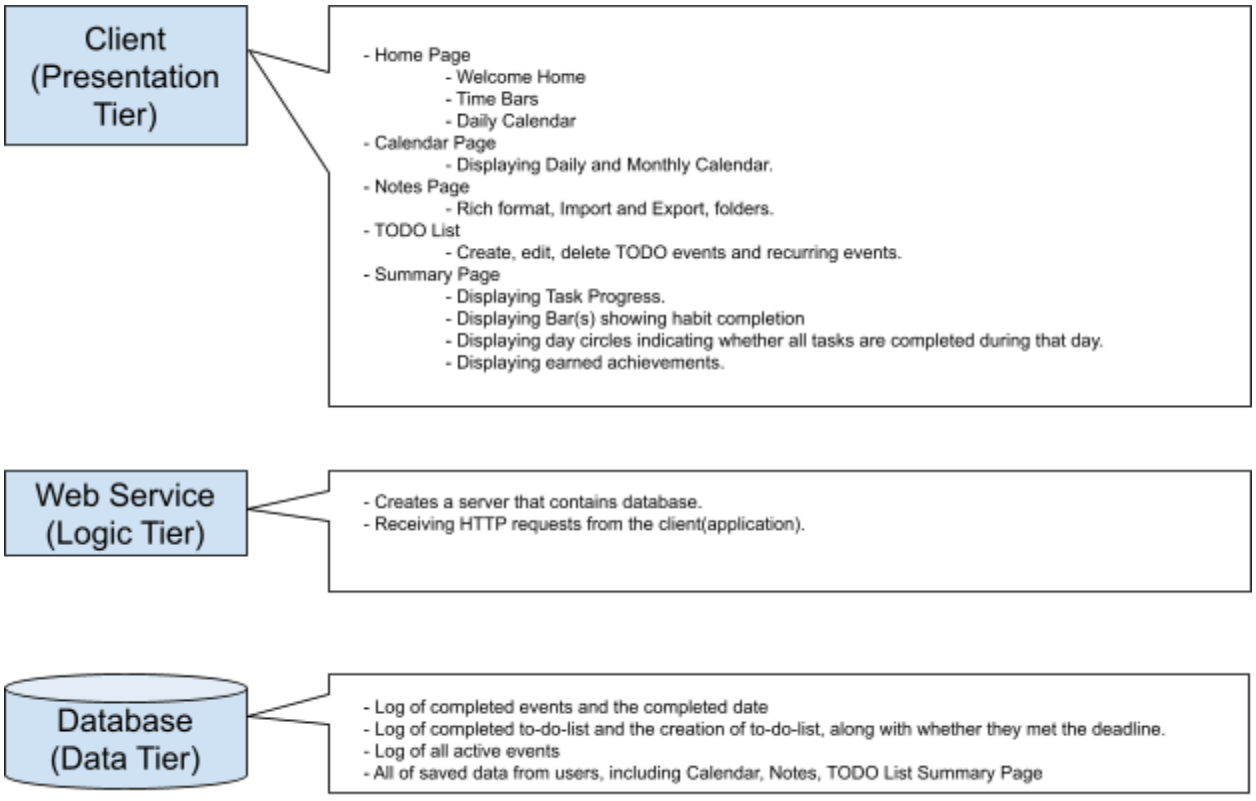
Architecture & Design

Architecture



The architecture style is a three-tier architecture. The presentation tier is the application(client) tier, which is responsible for displaying all of the main features of the application. For example, home display, calendar display, notes display, todo display and summary display are all implemented through the section. The Jetpack Compose Multi Platform is used in the Front-End tier. The Front-End tier, also known as application, communicates with the next tier, which is Logic Tier, also known as Web Service, in order to add, retrieve, modify and delete data that is stored in the database. The Logic tier is mainly responsible for creating a server that contains the database, and receiving HTTP requests from the client(application or Front-End). For

instance, if the application wants to create an event and store it in the database, it first translates the data into JSON format, then a HTTP request is sent to the server created by Web Service through Restful APIs, which then converts the JSON format back into the data and interact data tier which is the database. Then the database stores the data in the table with the corresponding fields. The Web Service is built using the Ktor framework, which is beneficial in various ways. To begin with, it is very suitable for creating Restful APIs, which is essential for interacting with the database. Moreover, the Ktor framework uses coroutine support, which helps to write asynchronous code. The database is the third tier that is responsible for storing, editing, reading and deleting data required by the client application. Data such as active to-do events, completed to-do events as well as statistics of the completed events are all stored in the database. The database is implemented using SQLite and Exposed because there are various benefits of using them. Firstly, SQLite does not require a separate server, so it is very easy to setup and maintain. Secondly, Exposed simplifies database operations so that it is easier to store, read and modify data without using SQL queries. Furthermore, Multi-Project build is used to configure the Gradle project to contain two modules, namely application and server. Each module has its own build.gradle.kts file because each module has its own rules for building. Multi-Project build is beneficial because it makes it easier to maintain and manage the project. Also, Multi-Project allows shared class between application and server so that it is easier to maintain and manage the code when facing modification.

Design

- ### Design patterns

    - Distributed pattern

      Three-Tier architecture style is used, which splits the concerns into presentation, logic and data. This is advantageous because expensive resources are shared.

- Factory method pattern

  When we click a button such as “Home”, “Calendar”, “Notes”, “ToDo lists”, “Summary”, an instance of the class is decided and created by the subclasses MagicHome(), Calendar(), Summary(), Notes(), ToDoList(), and etc. It is beneficial because it defines a way to decide which subclasses to instantiate.

- Design issues and decisions

- Storing recurring events in the database

  It requires too much space to store all of the recurring events in the database, the solution is to only store one of the many same recurring events in the database. Each event class has a field to determine whether it is recurring or not. Therefore, when displaying events, it searches for recurring events and displays them on other days. This method saves space tremendously and it works as intended.

- Displaying events of duration more than one hour in the daily calendar

  When displaying events that have duration more than one hour, the events need to be displayed across multiple hours during the day. The solution is to check if the hour is in the range of the start time and finish time of the event, and only display the event if it is in the time range. This method works as intended.

- Storing notes in the database

  It is difficult to store notes types in the database. The solution is when saving notes, we can convert the notes into html and store it as strings in the database.

Whenever we click the notes, we load the corresponding notes from the database. This method works as intended.

- Retrieving statistics for summary page

  It is very space consuming if we store the statistics in the database. So the solution is to retrieve the events from the database, filter and calculate the statistics in the class so that it saves space.