**G. *Selected* *architectural* *styles* *and* patterns:**

**G.1 Design Patterns**

Since we are using 3 Tier architecture separation of concerns is very important when implementing software. Separation of concerns is a design principle for separating a computer program into distinct sections, such that each section addresses a separate concern. A concern is a set of information that affects the code of a computer program. Each component is concern only one concern. Also, separation of concerns ease developers to reuse and maintain components.

MVC is one of the architectures which ensures separation of concerns. There are three main parts of MVC architecture. These are Model, View and Controller:

**Model** is the component corresponds to all the data-related logic that the user works with. This can represent either the data that is being transferred between the View and Controller components or any other business logic-related data. In TrackMe, Model includes all data related logic.

**View** is a component which is used for all the UI logic of the application. In TrackMe, View includes user interfaces which clients interact with.

**Controller** act as an interface between Model and View components to process all the business logic and incoming requests, manipulate data using the Model component and interact with the Views to render the final output. In TrackMe, Controller includes router which directs request to related component.

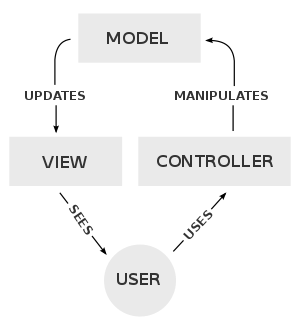


Diagram of interactions within the MVC pattern.

In order to increase reusability of components, ease maintenance and ensure separation of concerns MVC is the main architecture of TrackMe.

In addition to MVC pattern following patterns is used in some components for specific purposes:

**Factory Pattern:** is a creational pattern that uses factory methods to deal with the problem of creating objects without having to specify the exact class of the object that will be created. In our system in order to support different types of database server factory pattern is as DB Factory.

**Builder Pattern**: is a design pattern designed to provide a flexible solution to various object creation problems in object-oriented programming. The intent of the Builder design pattern is to separate the construction of a complex object from its representation. Builder Pattern is used in creating Anonym and Individual Data Requests by third parties.

**Observer Pattern**: is a software design pattern in which an object, called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods. Observer pattern is used SOS service to implement checking threshold values and send notification.

**Facade Pattern**: is a structural pattern which hides the complexities of the system and provides an interface to the client using which the client can access the system. For routing requests of web server to application server face pattern is used in TrackMe.

**G.2 Other Design Choices**

TrackMe uses Map services in order to display users’ location on map, especially Track4Run service. For this reason, external map service should be integrated. Google Maps is our design choice of to be integrated services since it provides accurate and maintainable location services.

In order to increase reliability application server should be integrated. Load balancer will be used to distribute requests according to servers’ workloads. In any case of any shortage on a server, this load balancing will prevent inaccessibility to services.

References

<https://en.wikipedia.org/wiki/Factory_method_pattern>

<https://en.wikipedia.org/wiki/Observer_pattern>

<https://en.wikipedia.org/wiki/Builder_pattern>

<https://www.tutorialspoint.com/design_pattern/facade_pattern.htm>

<https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>