2.1 Overview

A 3-tier architecture is a type of software architecture which is composed of three "tiers" or "layers" of logical computing which we use it for designing the architecture of our application. They are using in applications as a specific type of client-server system. 3-tier architectures provide many benefits for production and development environments by modularizing the user interface, business logic, and data storage layers. Doing so gives greater flexibility to development teams by allowing them to update a specific part of an application independently of the other parts. That is why we choose this method

• **Presentation Tier:** The presentation tier is the front-end layer in the 3-tier system and consists of the user interface. This user interface is often a graphical one accessible through a web browser or web-based application and which displays content and information useful to an end user. This tier is often built on web technologies such as HTML5, JavaScript, CSS, or through other popular web development frameworks, and communicates with others layers through API calls.

• **Application Tier:** The application tier contains the functional business logic which drives an application’s core capabilities. It’s often written in Java, .NET, C#, Python, C++, etc.

**• Data Tier:** The data tier comprises of the database/data storage system and data access layer. Examples of such systems are MySQL, Oracle, PostgreSQL, Microsoft SQL Server, MongoDB, etc. Data is accessed by the application layer via API calls.

**2.2 High level architecture**

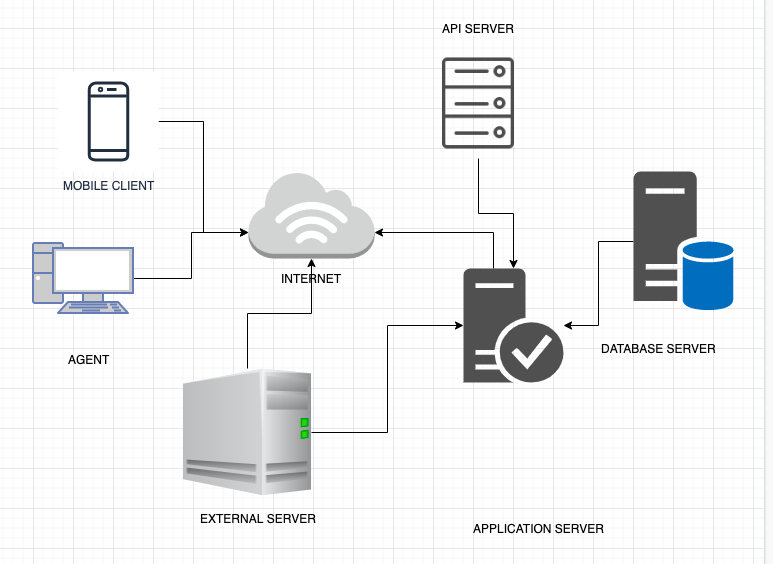


Figure 1: High Level Architecture Diagram.

The figure 1 shows the general architecture of the system. Both Registered User and municipal agent nodes are connected to the Internet in order to communicate with the External Server and Application Server nodes. Registered Userand municipal agent are registered to the application in order to communicate with the application server. Application Server is the part of the program that encodes the business logic that determine how data can be created, stored, and changed. On the other way it is the logical part of the system. Application server uses the services which External Server provides. Application Server is connected to the Database Server to have access to the data. In the Database server all the data, information, pictures and reports which used in the system will be stored, such as the personal information of Registered User.

2.3 Component view

As shown in figure 2, overall architecture of components has been carefully constructed accoring to the requirements. Safestreet is the core component of the system. From the requirements, municipalrity System will be developed on top of Safestreet and it will use for monitoring functionality. Report component will be a sub component of safestreet.

Figure 2: High Level Components Diagram.

* Safestreet: This is the most complex component and this requires more detailed information. Safestreet component is composed of three main services named ReportViolation, StaticsShow and AuthenticationService.
* AuthenticationService: This service will handle authentication related functionalities for both individuals and third parties.
* IndividualService: This component includes individuals Smartphone and Smartwatch devices. Applications which will be running on these devices are stated as modules of this component.
* SmartwatchApp: Smartwatch constantly gathers individuals health data and sends them automat- ically to the individuals Smartphone. Data sent to Smartphone is saved in SmartwatchApp.
* SmartPhoneApp: This module will run a background service which will handle the incoming data requests and also send user data to Data4Help service.
* Database: This component is composed of two different databases and this component is respon- sible of handling data transfer activities of corresponding databases. Data4Help and Track4Run services have different data structures and their data will be stored on different database modules.
* PersonalData: This database will include all health data and identifiable information about the individuals. This data will be used for verification and data transfers for subscribed customers.
* Track4RunData:Thisdatabasewillcontainalldatarelatedtotherunevents.Individuallocations, event times, paths for the run events, organizers and spectators will be located in this database.
* ExternalMaps: This component includes the external map services which will be used by almost all of the core components that require location and map functionalities.
* ThirdPartyService: This component contains SmartphoneApp and WebApplication for third par- ties.
* RunEventService: This service is responsible for handling all run event related functionalities. Organizers can create and define a path for the run event, individuals can search and enroll to run events and finally spectators can search and watch the run events from this service.
* LocationMonitorService: Since location of the athletes has to be tracked, this service will use the location functionality which is provided by the MonitorService which is the component of Data4Help.
* SpectateService: This service will get the location data of athletes from LocationMonitorService for the corresponding run event.

2.4 Deployment View

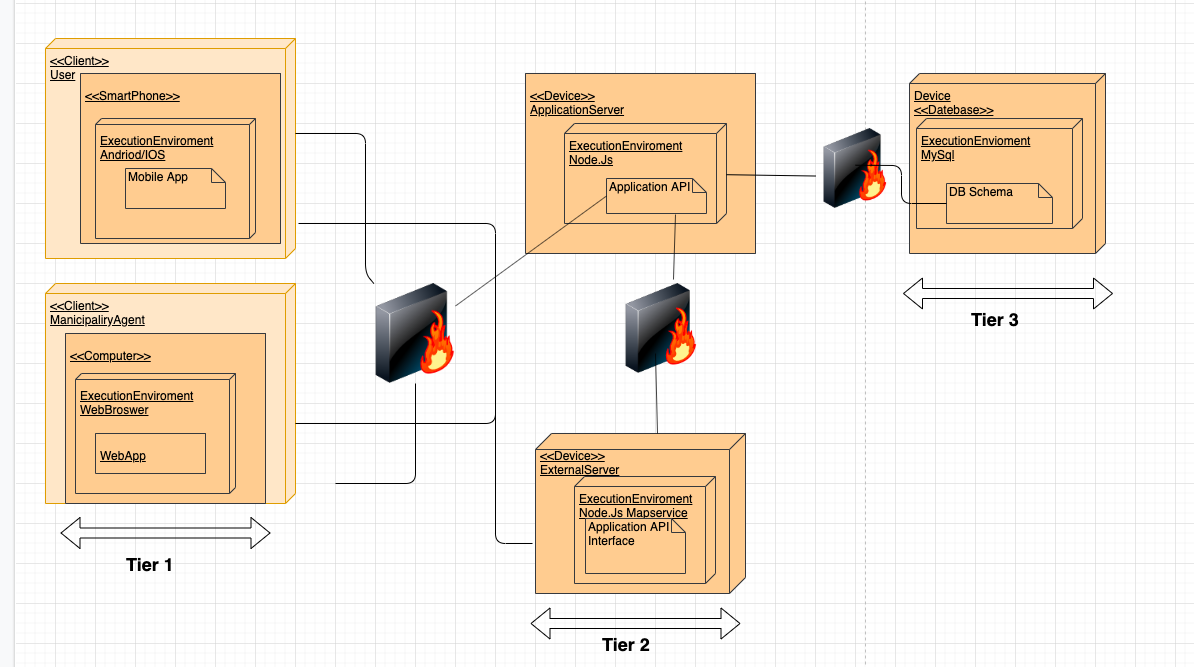


Figure 5: Deployment View Diagram.

This view describes the environment into which the system will be deployed, including the depen- dencies which the system has on its runtime environment which can be seen in the figure 5.

The five-dimensional boxes represent nodes, either software or hardware. Physical nodes should be labeled with the stereotype device, to indicate that it’s a physical device such as a computer or smart- phone.

User: This node consists of one node, Smartphone . Smartphone is a device which used by user. Smartphone is a device which is used by the user and communicates directly to the application layer but since the information is critical, a firewall between the Application Server .

Manicipalrity Agent: Agent are able to use web application by their web browser . Agents are able to use external server which includes Google Maps API and AmbulanceService that are connected to the third party. another server

that is connected to third parties is the application server which includes business logic. Consequently, application server sends the data to DB server.

Application server: It has relation with all other nodes of the system. In this node, all the computation of the system will be done, hence this node is logical part of the server. Application server obtains data from the external services of the system and stored information which are in database server and manages data requests.

External server: This node equips the system with extra services which are essential for the application. For instance, maps service are two nodes which are present in the External servers. Between external server and application server we have firewall since the data which shared are really critical for the system. This means that these type of data is really substantial such as the location of the people, so there should not be any access to the data from outside.

Database server: All the data and information which used in the system will be stored in the Database server such as the personal information of users. Between Application Server and DB Server a firewall will be used to ensure the integrity of the data transfers.

2.7 Selected architectural styles and patterns 2.7.1 Overall Architecture

As it has been explained in high level architecture, application will use three tier architecture ap- proach. This choice will improve the applications maintainability and complement teamwork in high number of people.

Presentation Layer will be the most lightweight tier to ensure that mobile devices and smartwatches which have limited resources will not have any performance issues. Some of the most recent frameworks for both web applications and mobile applications are an excellent choice for this tier. Angular and NativeScript will be used as the main front-end frameworks for both interfaces.

Application layer will handle the core functionalities of this project. Without compromising security in mind, this layer will be able to handle high number of requests and responses with the help of Node.js and Express.js frameworks. With asynchronous task management, the application layer will not require a high specced hardware. This will reduce the maintentance costs significantly as well.

Data Layer is the data storage and data management tier. PostgreSQL database server will be used since it is completely free and this is a big plus for the reduced costs of the project.

Of course the seperation of the layers will introduce performance drawbacks according to the data transfer between layers. But for maintainability and reusability in mind, this layered architecture is a much better choice in the whole development and runtime lifecycle.

**2.7.2 Design Patterns Client-Server Architecture**

This seperation of client and server will improve the performance of the client side. Since all of the business computations will be run on server, client will have a lightweight application. Also this decoupling also improves development process with letting developers work without knowing what is the business logic running on server side.

MVC pattern in Angular

Angular framework introduces a MVC pattern which is very useful for decoupling of the components and request services. Three modules can be used for the development process:

* Components:Thisisthemodulewhichwillmostlycontainthedatawhichispresentedatacertain time. These components can also reused and extended according to the needs of the application. TypeScript is used to develop these modules.
* Views: This type of module is responsible for handling the presentation of the data to user. HTML and CSS is used to develop these modules which is mostly related to design for the user interface.
* Services: This modules are handling the request-response cycle of the web application. All data requests from the components are sent to these modules and data transfer is always going through these modules. The design pattern of this module is Observable. These services can be injected to other components or even other services. For example; client side authentication will be done

through authenticaton service and this service will be injected to every service which handles the data requests from the application server.

Node.js and Express.js

Node.js is a cross platform engine which can be developed with JavaScript language. Express.js is a REST server implementation. With these two tools, business logic of the application will be deveoped. Since Node.js has a single non-blocking thread design, asynchronous development is necessary for the performance of the application server. With properly architectured code, this server is able to handle very high numbered requests and responses with respect to a very low-end hardware.

Mobile Application and Smartwatch Application

NativeScript framework will be used for developing both of mobile and smartwatch applications. Since components and services are developed the same way as Angular, only Views have to be coded with NativeScript. This is an excellent use of reusability of the codebase. After the compilation of the application, all of the views are transofrmed to native components of the corresponding operating systems. Native components are the best choice with respect to both performance and user experience.