Software Engineering 2: TrackMe

Design Document

Haritha Harikumar, Mohini Gupta, Saloni Kyal
Politecnico di Milano

December 10, 2018

Contents

Chapter 1

Introduction

1.1 Purpose

TrackMe is an android application developed by TrackMe. The application helps to monitor health vitals and location of individuals. TrackMe stands apart from other health monitoring application with its features that enables 3rd party services to monitor the data of individuals for a valid cause. TrackMe also provides other useful services like emergency alerts to ambulances in case an individual's vitals are in critical condition. Apart from that TrackMe comes with a service which offers people or a group of people to organise and visualise races or marathons.

This document is intended to be examined by a group of developers that will implement our system to make the implementation consistent with the system project.

1.2 Scope

Track4Me is a company which provides three software based services. The services are limited to the citizens of Milan. The given problem is to design and develop a software service called "Data4Help", which acts as an intermediary between the individuals and the other companies. It targets two set of users, individuals and third party. The location and health of the user is extracted from a wearable device. The data from the wearable device is extracted with the help of the API of the wearable device. The software tracks the location and health status of individuals or group of individuals and send the information to the third party on their request to monitor the health of individuals by validating the type of requests. The validation of the requests for the information of specific individuals is done by the individual's

self acceptance or rejection. The requests for the information of group of individuals is decided by Track4Me and generalized by allowing the access to the third party, if the number of individuals whose information is to be accessed is greater than 1000.

It builds a new service called "AutomatedSOS" over the top of "Data4Help" after monitoring the demand of requests from third parties for giving non-intrusive SOS service to elderly people, which sends the ambulance to individual's location when the health parameters are below the thresholds. The nearest ambulance is tracked and sent to ensure individual's quick and efficient recovery. A separate mobile application of Track4Me is to be exclusively developed for the ambulance drivers who are informed by push notifications in the mobile application along with the location and other details of the patient in less than 5 seconds to ensure spontaneity in reaction to emergencies. All the hospitals in Milan are informed to ensure all the ambulance drivers installs TrackMe application.

A new service called "Track4Run" is built as a source of revenue which allows organizer to organize an event, athletes to participate in the event and spectators to see the athletes position live in a Map. It exploits the service of "Data4Help" which tracks the location of the athletes to be visible to everyone and helps the organizer to monitor the health status of athletes as well. The organize will be registered as an existing or new third-party. The athletes will be registered as existing or new individual and the spectators can be registered as new/existing individual/third party.

The users can either register as an individual or as an organization/company(third party).Data4Help is a base service which all the individuals and third party will have by default even if the user registers for a different service other than "Data4Help". The third party can register for two services: "Data4Help" and "Track4Run". It uses "Data4Help for receiving information about targeted individuals and "Track4Run to organize events for a particular cause/occasion. The third party is a source of revenue for both cases, as Track4Me provides information about individuals and thereby, provides customers to their organization which brings them profit. Individuals can register for any of the three services. The first service is a free service with a motive to help them to receive specific input from respective third party for their health issues or other factors. The individual can also upgrade to other services if they are an existing a member using one service by making payment. Payment can be made using PayPal or a credit card. Track4Me guarantees the security of the individuals data and only associates with trusted third party companies. Ultimately, it helps both individuals and third party to manage and monitor their personal data and aims at being a robust and efficient software

1.3 Definitions, Acronyms, and Abbreviations

1.3.1 Definitions

1.3.2 Acronyms

1. DD: Design Document
2. API: Application Program Interface
3. DBMS: Data Base Management System
4. SOS: Save our Souls
5. GPS: Global Positioning System
6. GUI: Graphical User Interface
7. RASD: Requirement Analysis and Specification Document

1.3.3 Abbreviations

1. Gn: n-Goals
2. An: n-Assumptions
3. Dn: n-Dependencies
4. Cn: n-Constraints
5. Rn: n-Functional Requirement s

1.4 Document Structure

The DD is composed of 6 parts including references

1. Chapter 1: Introduction
2. Chapter 2: Architectural Design
3. Chapter 3: User Interface Design
4. Chapter 4: Requirements Traceability
5. Chapter 5: Implementation, Integration and test plan

6. Chapter 6: Effort Spent
7. Chapter 7: References

Chapter 2

Architectural Design

2.1 Overview

The TrackMe application has the following architecture.

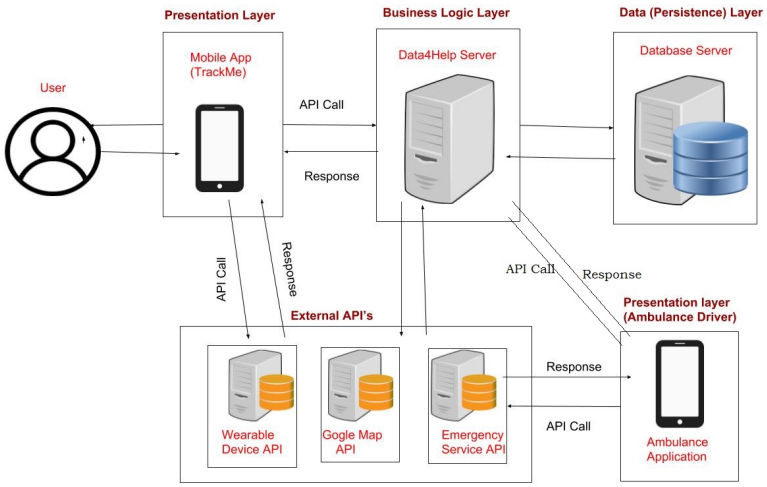

Figure 2.1: Architechtural overview of the system

The architecture contains four layers, *Presentation Layer*, *Business Layer*, *Data Layer* and *External API's*. There are two presentation layer in the ar-

chitecture as one is for the user and one explicit application for the ambulance driver so that they receive the individual's details within 5 seconds and can go to individual's location as soon as possible. All the layers are separately explained below:

- **Presentation Layer:** The presentation layer of the user provides a user interface to the third party and individual where they can perform operations. It also has a map which is used by users using the service of AutomatedSOS and Track4Run. The presentation layer takes the information from external API's by API call to display the information fetched from the external services in the user interface. It also has an API call from business logic layer to display the information from the database server of TrackMe which the user has given during registration.
  The presentation layer of the Ambulance Driver also uses external API's Google MAP API. It also uses the TrackMe server to receive Individual's location in the form of push notification.

- **Business Logic Layer:** The business layer is the application server of the TrackMe application which manages the entire data flow from the application end to the data layer and vice versa. This layer focuses on the business front. and how it will be presented in front of the end users. This includes workflows, business components, and entities beneath the hood of two sub-layers named service and domain model layers.
  While the service layer focuses on defining a common set of application functions that will be available to users, the domain model layer represents expertise and knowledge linked to the specific problem domain. The entire plan is formulated in a way to explore and enhance the future of application.

- **Data (Persistence) Layer:** It contains all the data information which is required by the other layers of the system. It separates the getting and saving of the data from the business layer. The reason we do this is that the business logic (the part of the application that does the heavy lifting for your data manipulation) is not tied to a specific type of data source. The data layer will need to be written to be database specific. This includes data access components, data helpers/utilities, and service agents.

- **External API's:** This layer refers to all the external service used in the application. We use the API's of the external services in order to

extract information from them and use it in the application. *Wearable Device* is provided by the individual at the time of registering which they mention among all the list of compatible devices with the system. The API of the same wearable device is then used to extract the location and health information o the individual in order to provide them to the third party. The *Google Map API* is used to display the map view in the application wherever it is required. The *Emergency Service API* is used by the application to find the nearby hospitals and the ambulance they provide which can be used by the service of AutomatedSOS.

2.2 Component View

In the following diagram, the mentioned components are more closely examined, with the main focus on the application server. While describing the parts, the notation will be the interface they are providing in order to be more general, because there may be one or more different implementations of the same service within the server. The components shown here communicate with each other and work together to complete certain user requests.

DATA4HELP SERVICE

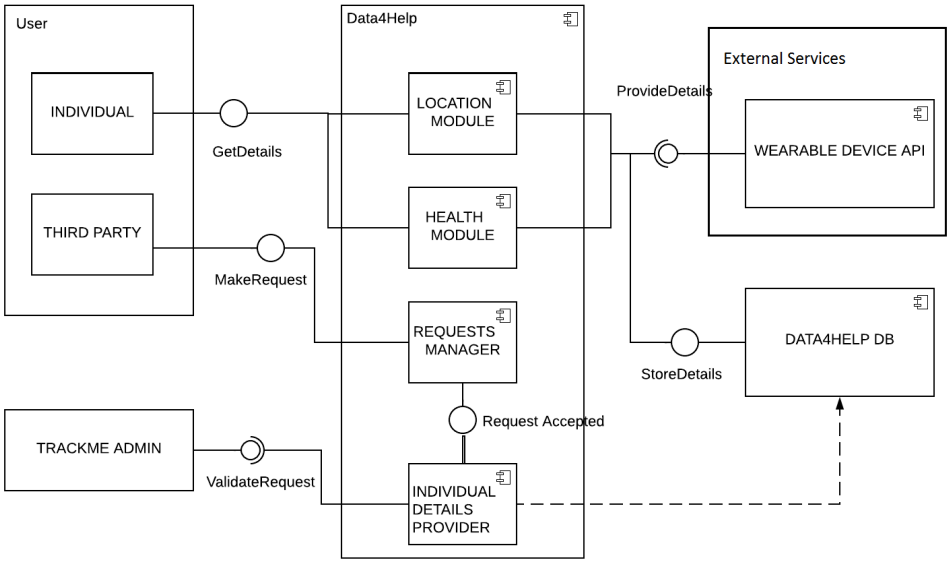

Figure 2.2: Component Diagram for Data4Help Service

- **Location module:** the location fetched from the Wearable Device API provided by an Individual is stored in the Data4Help DB.
- **Health module:** the health values fetched from the Wearable Device API provided by an Individual is stored in the Data4Help DB.
- **Request Manager:** provides an interface to the Third Party to make new request to get the details of a specific Individual or a group of Individual. It also provides an interface to a specific Individual to accept or reject the request.
- **Individual Details Provider:** this uses the help of the Data4Help DB to provide the details to the Third Party after the TrackMe Admin has validated the request.

AUTOMATEDSOS SERVICE

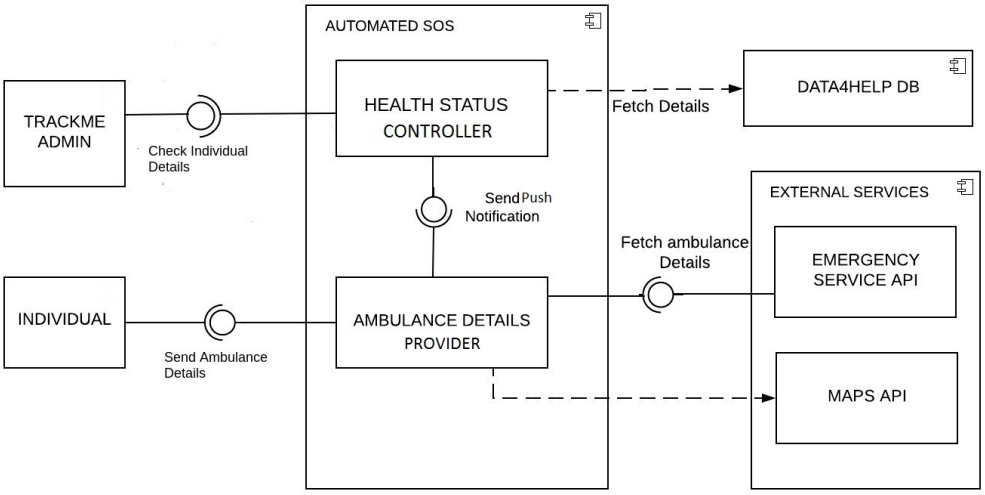

Figure 2.3: Component Diagram for AutomatedSOS Service

- **Health Status Controller:** uses the help of the Data4Help DB to get the details of an Individual. The TrackMe Admin provides an interface check Individual Details to the controller. The controller then, monitors the individual's vitals and when the vital is below the threshold value, it provides an interface named send notification.

- **Ambulance Details Provider:** gets the details of all the Ambulance associated to the major hospitals from the Emergency Service API. It uses the maps API to get the nearby ambulances and send notification to that ambulance driver application. It uses the functionality of Push Notification to send the notification. After a driver accept the request, the details of the driver is send to the Individual.

TRACK4RUN SERVICE

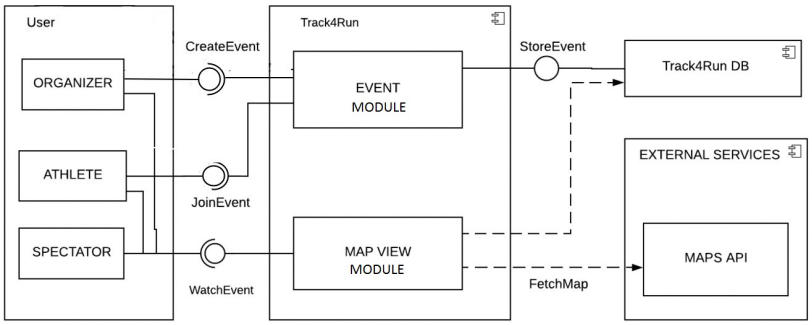

Figure 2.4: Component Diagram for Track4Run Service

- **Event Module:** manages the details of the event and the athletes. It gets all the details of the event from the organizer which is entered after the organizer upgrades to the service. The event details is stored in the Track4Run DB. The athlete joins the event and is then associated with that specific event.
- **Map View Module:** It uses the Map API to provide an interface watch event to the organizer, athlete and the spectator. It checks the validity of the subscribed users. It uses the Track4Run DB to get the location of the athletes participating in an event, and then maps it to the interface.

MAIN COMPONENT

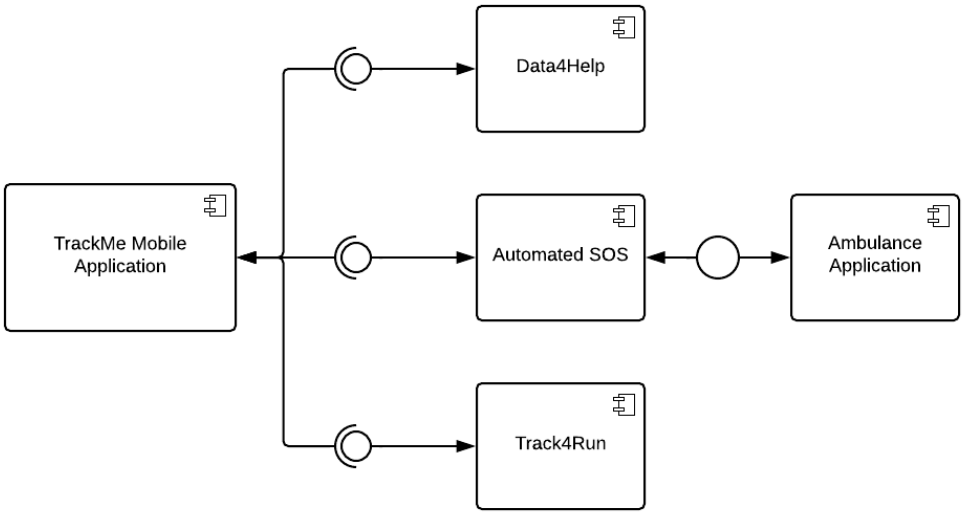

Figure 2.5: Main Component Diagram

The main component is divided into three subcomponents which are shown separately to avoid unreadability. All the three subcomponents are the services which is being provided by the TrackMe Application.
The Ambulance Application is a separate application developed explicitly for the ambulance drivers where they receive individuals details through push notification. It is connected to Automated SOS service for exchange of information of ambulance and individual respectively.

2.3 Deployment View

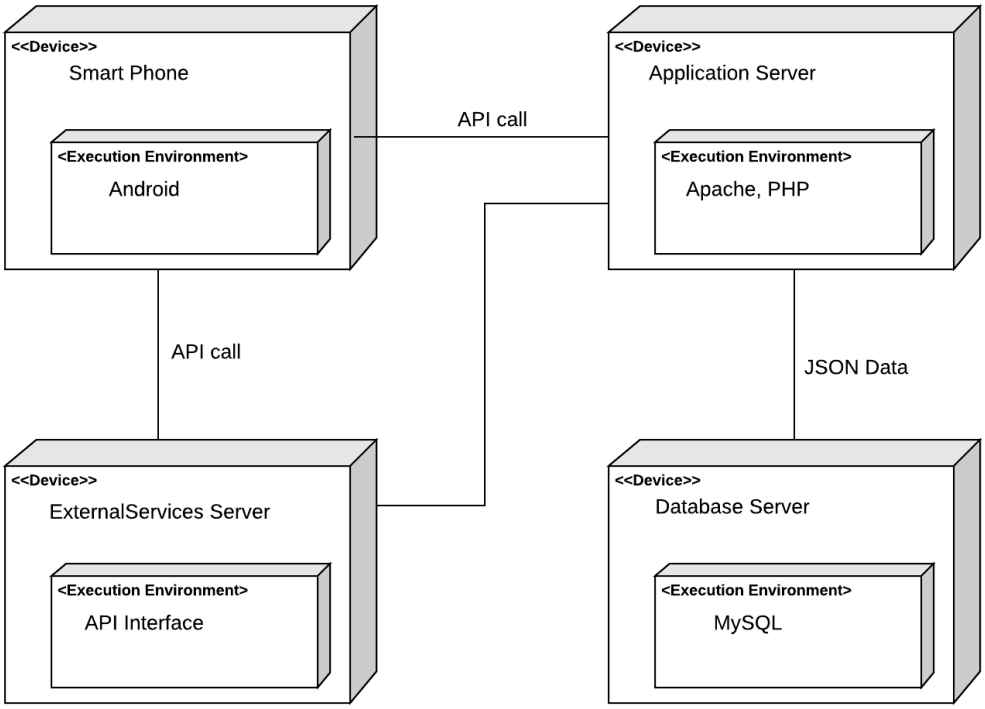

Figure 2.6: Deployment Diagram

It shows architecture of the system as deployment (distribution) of software artifacts to deployment targets. Artifacts represent concrete elements in the physical world that are the result of a development process which are *Android, Apache, PHP, API Interface* and *MySQL*. Deployment target is usually represented by a node which is either hardware device or some software execution environment which are *Smart Phone, Application Server, External Service Server* and *Database Server*. The nodes are connected through communication paths to create networked systems of arbitrary complexity. The nodes an their respective artifacts are explained below:

- **Smart Phone:** The user uses smart phone to view the mobile application of the system. This is implemented in the execution environment Android which is artifact of this device.
  -*Android:* The Android application can be built using any platform. It takes the information from external API's and the database server

of the application to process and show information to the user using XML design techniques and Java Script codes.

- **Application Server:** The Application Server is the host of the application and the database server. It is the central system of the entire application which manages the transfer and retrieval of data. The execution environment here is Apache and PHP.
  -*Apache and PHP:* The Server does all the transaction on the database and manages the retrieval and transfer of data from and to database by using PHP code. We use PHP code to help convert the database into an API in JSON format which can be easily used by the application's front end to use the data.

- **Database Server:** It stores all the data which is used by the application. The execution environment is MySQL.
  -*MySQl:* MySQL database is used to store the details where the data can be easily stored from the form the user fills in the application.

- **ExternalServices Server:** It contains the information of all the external servers required by the application. The information from the server by using the API Interface.
  -*API Interface:* The API provides the data in the JSON format which is fetched in the android platform by API call.

2.4 Runtime View

The runtime view is explained with the help of sequence diagrams, which shows the interaction between the components during runtime in a time flow depicting the order of which function takes place after another. The function names are same as the interface in order to help to relate the component, their interaction and their behavior during runtime. The sequence diagram of the three major services of TrackMe are shown which comprises the entire functioning of TrackMe and integrates to make the complete application.
Here are the three sequence diagrams which briefly summarizes the runtime of the application.

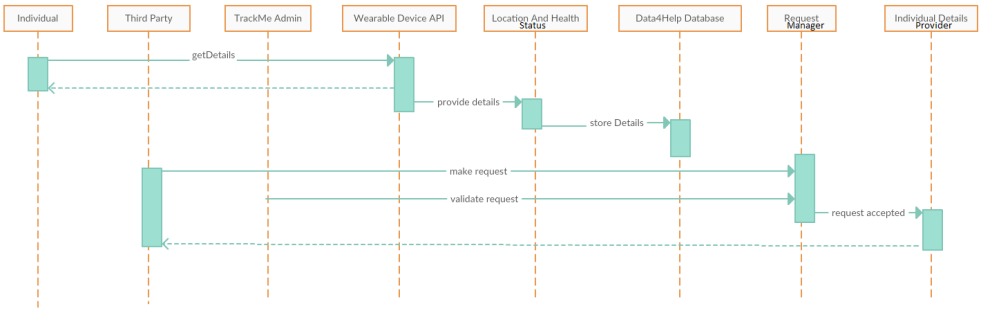

Figure 2.7: Sequence Diagram depicting runtime of Data4Help Service

The location and health is taken from the user by the *GetDetails Interface* with the help of the API of the wearable device by *ProvideDetails Interface*, which the individual selects from the list of compatible devices during registration. The details of the individual is then stored in the Data4Help database using *StoreDetails Interface*.
The thirdparty makes the request to access individual's data which is done by *MakeRequest Interface*.
The TrackMe Admin then validates the third party request and sends it to user to accept/reject or accepts/rejects himself according to the type of requests by *ValidateRequest Interface*.
If the request is accepted the user gets the access of the individual's details which uses the Data4Help Database in order to extract individual's details.

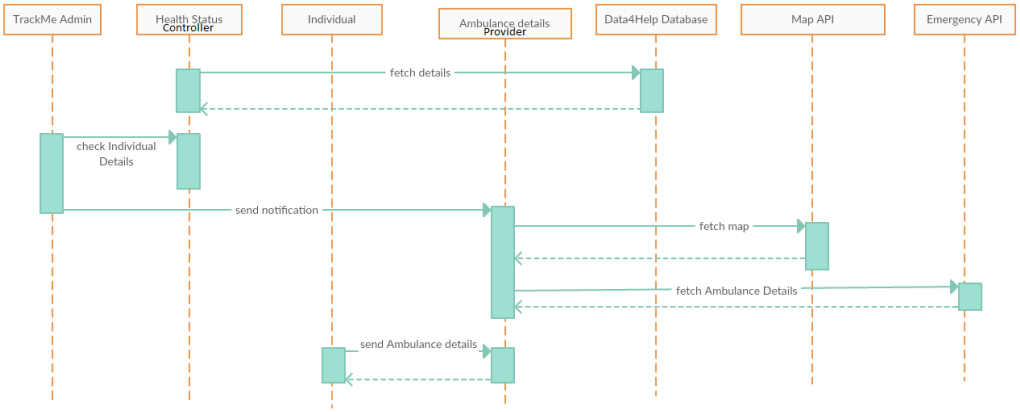

Figure 2.8: Sequence Diagram depicting runtime of AutomatedSOS Service

Trackme checks the age of the user in order to know whether they are eligible for this service and checks the vitals to know about the emergency condition using using *CheckIndividualDetails Interface* which is fetched from the Data4Help Database and updated as Health Status component which contains the emergency vitals of individuals.
If the emergency situation is acknowledged the admin sends notification using *SendPushNotification Interface* which includes the health status and location. The Individual receives the Ambulance Details by *SendAmbulanceDetails Interface* and can track the ambulance whose information is fetched from the emergency Service API using *FetchAmbulanceDetails Interface* and uses Maps API to show the map.

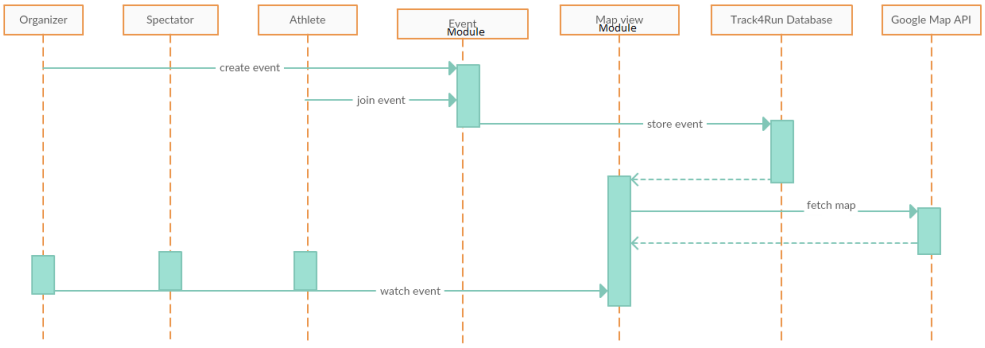

Figure 2.9: Sequence Diagram depicting runtime of Track4Run Service

The organizer creates an event which the athlete joins using *CreateEvent Interface* and *JoinEvent Interface* respectively. The details of the event is store din the Track4Run Daatabase using *StoreEvent Interface* which is used by Map view to show the details on map. The organizer, athlete and Spectator can watch the race in a map using *WatchEvent Interface* which uses Map API to fetch the map.

2.5 Component Interfaces

In the following diagrams, the component interfaces are presented and the dependencies between the parts of the application server are shown. Each Interfaces which is already explained in the runtime view are detailed here with all the functions and methods they include. Like the component diagram, has three parts to implement three services in the same manner this has three separate diagram to explain all the interfaces and there interactions.

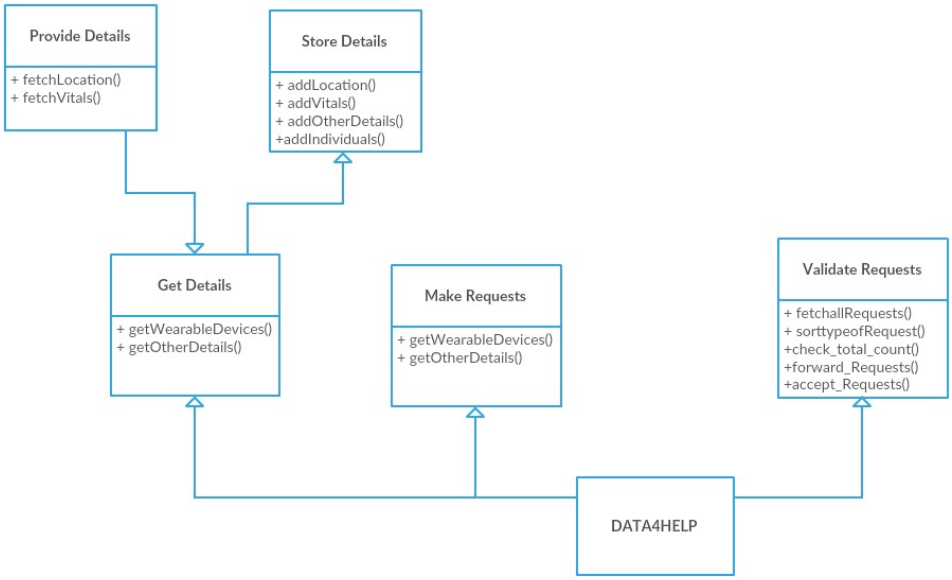

Figure 2.10: Component Interface Diagram for Data4Help

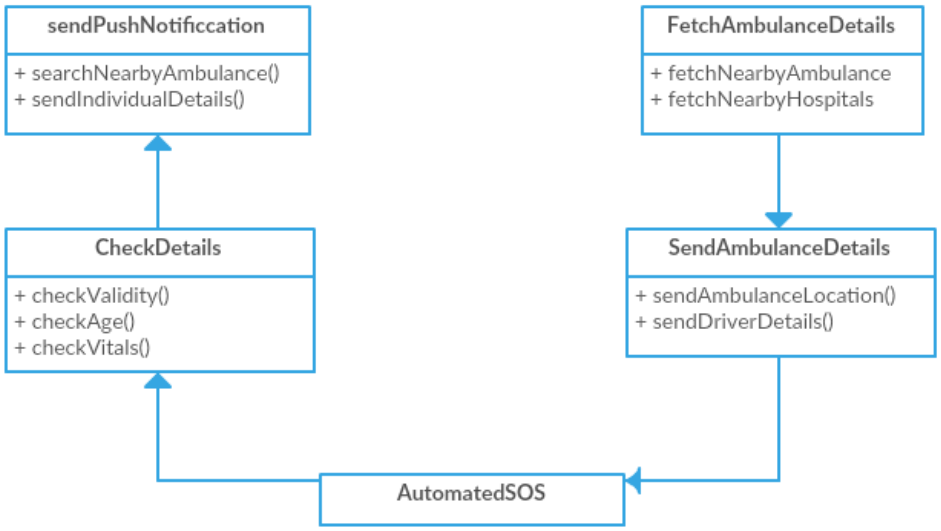

Figure 2.11: Component Interface Diagram for AutomatedSOS

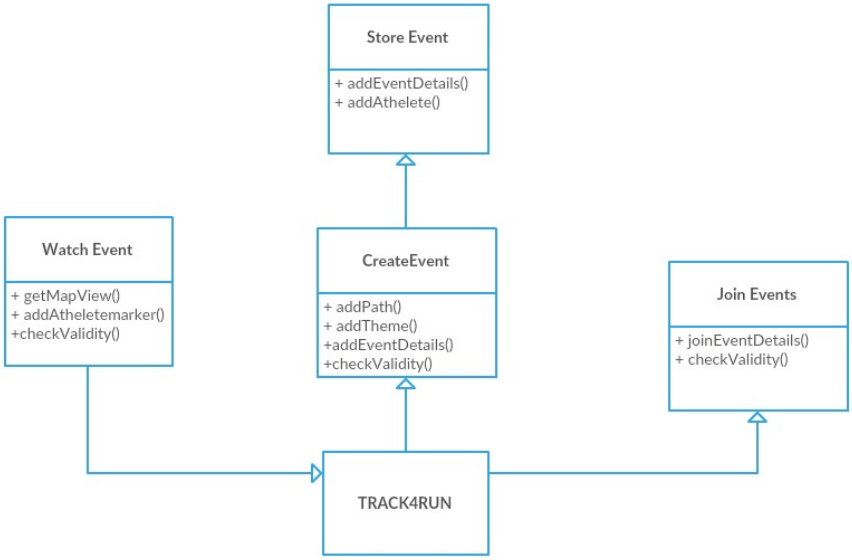

Figure 2.12: Component Interface Diagram for Track4Run

2.6 Selected architectural styles and patterns

2.6.1 Overview

The system essentially uses a variety of the multi-layered architecture i,e is the three-tired architecture for our TrackMe application. The application includes a presentation layer, business layer and data layer. The presentation layer, which is the top most layer represents the display of information related to the services given by the application like checking health status, requesting data, watching a live race etc. The presentation layer directly interacts with the client getting information from the client, with the help of GUI. The business layer also known as the application layer in general, fetches the details form the presentation layer and does detailed processing. For instance it ensures the connection with the corresponding external services required and processing for the data from the services like Google maps, wearable devices etc. The data access layers basically provides an API to the application tier that exposes methods of managing the stored data without exposing or creating dependencies on the data storage mechanisms.

We use a three tier application so that we would be able to update the technology stack of one tier without affecting the other one, and provides an easy of managing the different layers.

2.6.2 Design Patterns and Architectural Choices

1. Push Notification

   We have used the push notifications, which include the location and details of the individuals in need of emergency care are send to the ambulance drivers via the ambulance application. We basically use the application to make sure of the time constraints that the notification should reach within 5 seconds. The choice is made because of the difficulty in coming up with an asynchronous protocol for message exchanging.

2. Robustness

   The principal usage of the system is done through mobile applications which are running on mobile terminals. As we can imagine, in that domain the robustness of the system is an important aspect to keep in mind. The mobile devices are often subject to loss of connectivity and for that reason the communication between the server and a client could be not available in various time points

2.6.3 Other Design Decisions

1. Storage of Passwords

   The passwords are not stored in plain-text but they are hashed and salted with cryptographic hash functions. This provides a last line of defense in case of data theft.

2. Programming Languages

   For the implementation of TrackMe application we choose Java as a programming language. This choice is based on the following considerations:

   (a) Java is a widespread programming language, so we are sure that there will be availability of skilled programmers in this technology.

   (b) The choice of the platform is left to the developer. He/she has wide range of Java platforms to choose from.

3. Ambulance Mobile Application

   The ambulance mobile application has been developed basically for the ambulance drivers to recieve the above mentioned push notification within a time interval of 5 seconds.

4. External Services

   The system uses external services, Google Maps to offload all geo-localization, position tracking and map visualization process. The reasons of the choice are the following:

   (a) Manually developing maps for city of Milan is not a viable option due to tremendous amount of coding and data collection time required.

   (b) Google Maps is a well-established, tested and reliable software component used by millions of people used around the world.

   (c) Google Maps can be used both on the server side and the client side.

   (d) Users feel comfortable using a software which they daily use it.

   (e) Google Maps offers API, enabling programmatic access to features.

   The system also uses external services like wearable device API, which helps us to fetch the health vitals of the patients and the emergency service API to extract all the details of the nearby hospitals. Wearable device are the best choice to get the health vitals of a patient as it is easy to have it on hand always and can be easily connected to the application. Most of the users will be having a prior experience to use a wearable device.

Chapter 3

User Interface Design

The following User-Interface flow diagram enable us to model the high-level relationships between major user interface elements that are described in our RASD documents. It is used to organize the activities of TrackME application into groups and have an overview of the interfaces and interactions.

- The user interface for an individual allows the individual to view the vitals, view the third party details, and accept/reject the request.

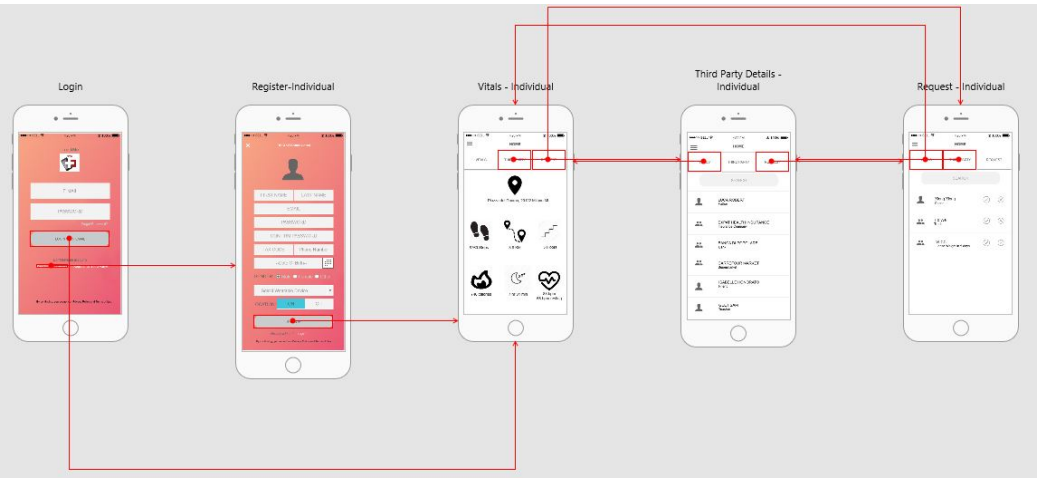

Figure 3.1: User Interface - Individual

- The user interface for a Third Party allows the third party to view the details of the individual viewed in a pop-up box, allows to make new request, and view the status of the request already made.

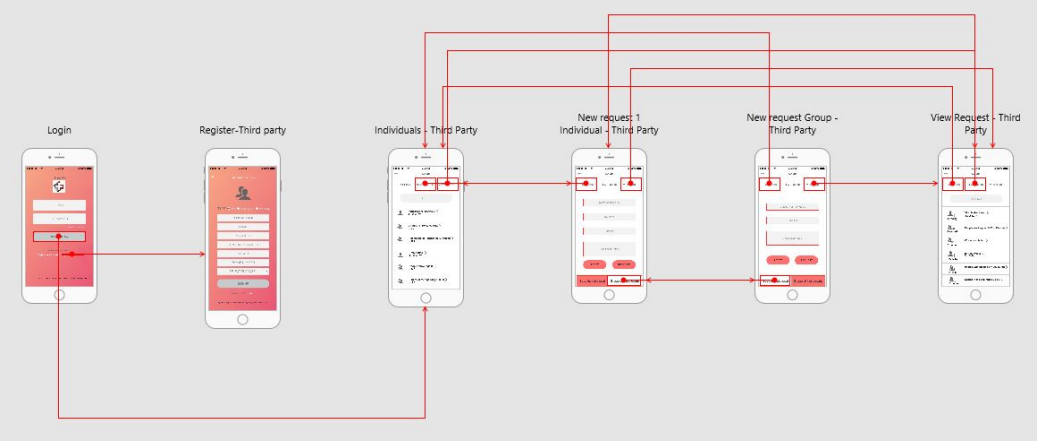

Figure 3.2: User Interface - Third Party

- The Menu Interface allows an user to view their own profile, upgrade to a new service, and then allow to view the new service.

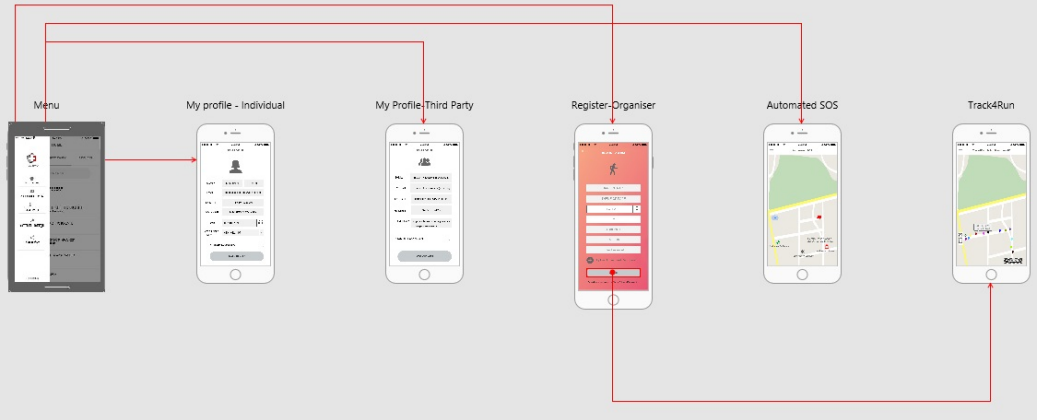

Figure 3.3: User Interface - Menu

Chapter 4

Requirements Traceability

The main objective of determining the design described in this document was to fulfill the requirements and goals mentioned in the Requirement Analysis and Specific Documents.
The following list outline the mapping between the requirement and goals with the design elements mentioned in the DD.

- **[G1]** Providing a list of the wearable devices that an individual can choose from. ([R1]-[R2])
  - Data4Help
    - Location Module
    - Health Module
    - Wearable Device API
    - Data4Help DB
- **[G2]** Expedite the request made by the 3rd party to an individual to access their details. ([R3]-[R4])
  - Data4Help
    - Request Manager
    - Data4Help DB
- **[G4]** Validating the request made by the 3rd party to provide the details of a group of individuals.([R6]-[R8])
  - Data4Help
    - Request Manager

    * Individual Details Provider
    * Data4Help DB

- **[G5]** Quick access of the data to the 3rd party once the request is approved.([R9]-[R10])
  - Data4Help
    * Request Manager
    * Individual Details Provider
    * Data4Help DB

- **[G6]** Immediate update of the data once an individual updates their own details.([R11]-[R12])
  - Data4Help
    * Location Module
    * Health Module
    * Wearable Device API
    * Data4Help DB

- **[G8]** Monitoring the health status of the subscribed individual to the Automated SOS service.([R14]-[R16])
  - AutomatedSOS
    * Health Status Controller
    * Ambulance Details Provider
    * Data4Help DB
    * Emergency Service API
    * Maps API

- **[G9]]** Develops an interface to organize a race for the subscribed 3rd party to the Track4Run service.([R17]-[R19])
  - Track4Run
    * Event Module
    * Track4Run DB

- **[G10]** Give access to the details of the race once an individual participate in it.([R20]-[R21])
  - Track4Run

        * Event Module
        * Track4Run DB

- **[G11]** Live visualization of the race .([R22]-[R23])
    - Track4Run
        * MapView Module
        * Track4Run DB
        * Maps API

Chapter 5

Implementation, Integration and Test Plan

5.1 Implementation

The implementation of the TrackMe system will be done service by service followed with the integration of all the services. The order in which it is be carried out depends on a number of factors like the complexity of the modules and services, the dependence of other modules on the component being implemented and to the system as a whole, and it should also take into account the possibility of discovering flaws with the proposed design. The later should be dealt in a way that, if such an unfortunate event does happen,the flaws should be found and corrected as soon as possible, to limit the cost of the change of design. In this sense, the components of the TrackMe, could be grouped in the following way, with the order specifying the order of implementation:
1. Data4Help Service
2. Automated SOS Service and Ambulance Application
3. Track4Run Service
4. TrackMe (Connecting the above three services)

The first service to be built is Data4Help Service as the further services exploits the features of Data4Help Service. Data4Help is a base service which all the individuals and third party will have by default when they register in the application. Hence, it has to be built first as a separate service. This service does the main function of taking data from user and storing it in database and hence does not require any payment interface with might be required by other services, hence it is implemented separately.

After the first service AutomatedSOS and TrackMe Ambulance Application is implemented which is a service used by only Individuals. This is a service for the elderly individuals whose vitals are monitored and a nearby ambulance is sent whenever emergency situations are encountered. This service exploits the features of the above service and hence it is implemented after that. The details of the individuals with vitals below threshold is sent to the Ambulance Driver. The ambulance drivers have explicit applications in order to ensure fast delivery of individual's data through push notification.Hence the application must be implemented together with this service layer as there is an exchange of information from application to this service layer and vice versa.

The next service implemented is Track4Run service which is basically an event which is organized by Organizers who are basically third party and individuals can perform as athlete. This service is not related to the second service by any means and hence both the services can be implemented parallely and separately and this can be implemented even before AutomatedSOS. The order of these two service wont matter but it must be implemented after Data4Help Service as it exploits the feature of that service as well.

After implementing all the three services of the application, we can merge them and implement the entire working of the application connecting all the three services and how the user upgrades from Data4Help to other services and uses them. Hence this layer is implemented at the end after implementing all the services separately.

5.2 Integration

1. Integration of Data4Help

- Integration of the components of Data4Help of the application server.
  – Request Manager, Individual Details Provider
- Integration of components of Data4Help with the DBMS.
  – Location Module, Data4Help DB
  – Health Module, Data4Help DB
  – Individuals Details Provider, Data4Help DB
- Integration of components of Data4Help with the (other) external services.

– Location Module, Wearable Device API
– Health Module, Wearable Device API

2. Integration of Automated SOS

- Integration of the components of Automated SOS of the application server.
  – Health Status Controller, Ambulance Details Provider
- Integration of components of Automated SOS with the DBMS.
  – Health Status Controller, Data4Help DB
- Integration of components of Automated SOS with the (other) external services.
  – Ambulance Details Provider, Emergency Services API
  – Ambulance Details Provider, Maps API

3. Integration of Track4Run

- Integration of components of Track4Run with the DBMS.
  – Event Module, Track4Run DB
- Integration of components of Track4Run with the (other) external services.
  – Map View Module, Track4Run DB – Map View Module, Maps API

4. Integration of TrackMe

- Integration of the components of TrackMe with Data4Help.
- Integration of components of TrackMe with AutomatedSOS.
- Integration of components of TrackMe with Track4Run.
- Integration of AutomatedSOS with TrackMe Ambulance Application.

5.3 Integration Test Plan

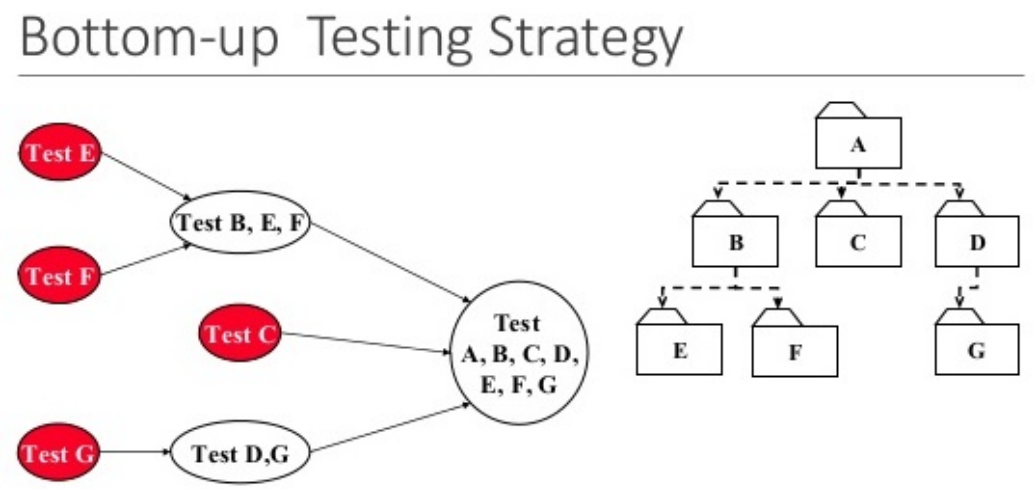

Figure 5.1: Bottom up- strategy for testing integration plan

A bottom-up approach is the piecing together of systems to give rise to more complex systems, thus making the original systems sub-systems of the emergent system. Bottom-up processing is a type of information processing based on incoming data from the environment to form a perception. From a cognitive psychology perspective, information enters the eyes in one direction (sensory input, or the "bottom"), and is then turned into an image by the brain that can be interpreted and recognized as a perception (output that is "built up" from processing to final cognition). In a bottom-up approach the individual base elements of the system are first specified in great detail. These elements are then linked together to form larger subsystems, which then in turn are linked, sometimes in many levels, until a complete top-level system is formed. This strategy often resembles a "seed" model, by which the beginnings are small but eventually grow in complexity and completeness. However, "organic strategies" may result in a tangle of elements and subsystems, developed in isolation and subject to local optimization as opposed to meeting a global purpose.
Considering the implementation plan and the overall architecture of the TrackMe application, the chosen strategy for the integration testing is the *bottom-up strategy*. This allows us to start the integration and it's testing while not waiting for the completion of the development and the unit testing of each component in the system. Considering the integration of two com-

ponents, we would assume that, in best case, they have been implemented fully and that their respectful unit tests pass. However, the integration can, in some cases, start, if necessary, before the implementation has been completed. This can be allowed if the part of the component needed for that specific integration has been completed and tested.

Since the opted solution is to start from the bottom-up, that means that the among the first integrations performed will have the already built external components in them. Since the application rests on these services and the communication with them, this order of integration and testing will enable the earlier detection of errors in these critical parts.

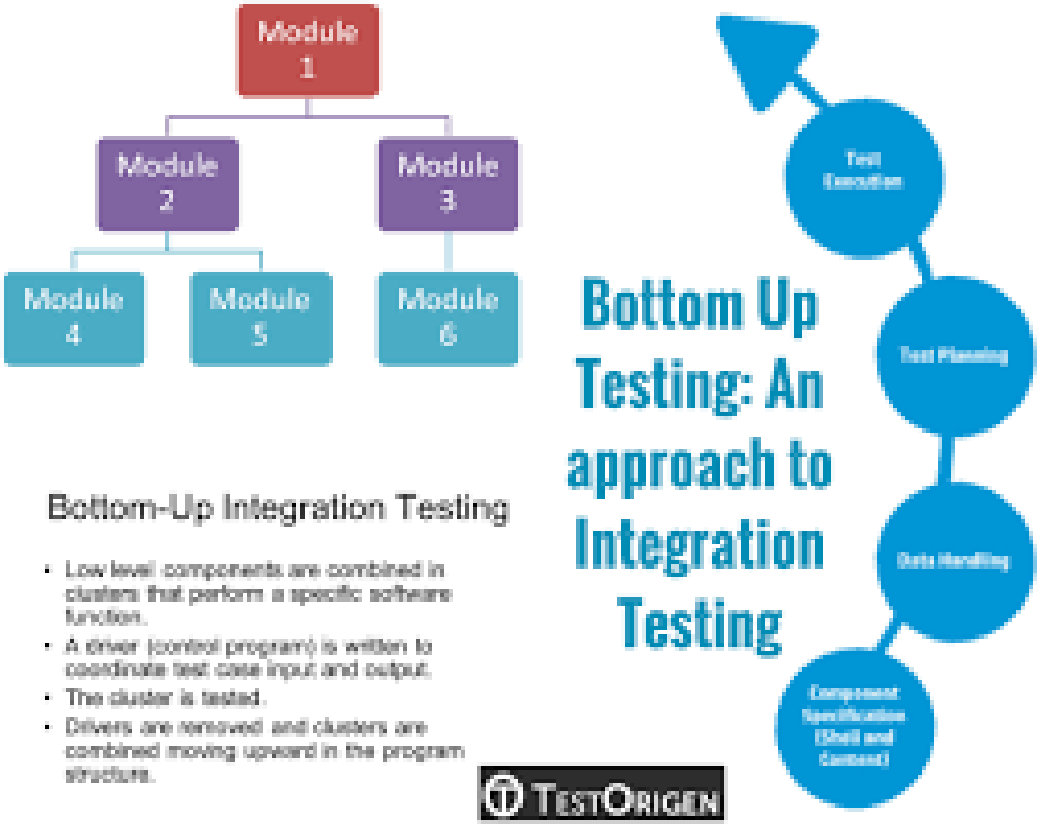

Figure 5.2: Working of bottom up- strategy for testing integration plan

Chapter 6

Effort

6.1 Hours of Work

6.1.1 Saloni Kyal

Date	Task	Hours
22-11-18	Analyze DD Document and Decide Architecture (Group Work)	3
01-12-18	Component Diagram and Runtime View(Group work)	8
02-12-18	Deployment and DD Documentation	3
05-12-18	Implementation plan, integration and testing	4
06-12-18	Component Diagram documentation(Group work)	3
07-12-18	Requirement Traceability and Automated Component interface	1
08-12-18	Documentation and RASD version 2(Group work)	8
	Total	**30**

6.1.2 Mohini Gupta

Date	Task	Hours
22-11-18	Analyze DD Document and Decide Architecture (Group Work)	3
01-12-18	Component Diagram and Runtime View(Group work)	8
03-12-18	User Interface Design	4
05-12-18	Requirement Traceability	1
06-12-18	Component Diagram documentation(Group work)	3
06-12-18	Documentation	3
08-12-18	Documentation and RASD version 2(Group work)	8
	Total	**30**

6.1.3 Haritha Harikumar

Date	Task	Hours
22-11-18	Analyze DD Document and Decide Architecture (Group Work)	3
25-11-18	Introduction, Scope and Details	1
01-12-18	Component Diagram and Runtime View(Group work)	8
03-12-18	Software Component Diagram Final	1
06-12-18	Component Diagram documentation(Group work)	3
07-12-18	Component Interface Diagram	2
08-12-18	Documentation and RASD version 2(Group work)	8
08-12-18	System Architecture and other design constraints	2
09-12-18	Final Draft	2
	Total	**30**

Chapter 7

References

- Specification Document: "Mandatory Project Assignment AY 2018-2019"
- Lucid Chart Tutorials
- Design Slides