**Course Project:** Family Tracker Application (Android)

by

**Group – 4**

**BSCS14035** – FizzaTauqeer

**BSCS14059** – MamoonaRiaz

**BSCS14072** – SiddiquaNayyer

**BSCS14004** – AwabAqib

* **Incorporate the feedback that you have been given in the previous assignment:-**

We were advised by the course instructor to consider various dimensions while answering how we would propose to build our system incrementally, rather than secluding to a singular dimension. Prior to this advice, we answered the question by propositioning to build the system with empty classes comprised of predefined functions, while adding functionality and overriding in later, incremental stages. After getting the above mentioned feedback, we could comprehend our previous viewpoint along the lines of other dimensional factors of incrementality (for example, tracking location of one family member, then two family members and thus extending to the entire family in each phase). Similarly, in building such a system, one feature would be added and tested, then the next and so on - in each incremental stage.

* **Identify which of the following techniques do you think are useful for requirements elicitation in your project. Make sure that you justify your answer. More than one technique may be applicable in your project.**

1. **Interviews**

Interviews, both of closed and open-ended nature, will be viable for numerous aspects of our requirements intended - or therefore elicited - for the project. Our system demands constant (and at times precise) communication and correlation of family members’ location or their location tracks, and interviewing such intended stakeholders will obviously reflect and alert us of the needs and expectations of the majority of them from our system, and how it can serve to be maximally useful to these intended users. For example, interviews could direct us to adding features of location alerts, perimeter scaling/threshold distances and emergency buttons into our system – these features could be unusual in such a family tracking set-up but if they aid in the purpose of the system and are demanded by the users of such a system then we would obviously strive to include them. Moreover, getting to know the preferences of the intended market of the system will ensure its better success – it is simply the difference between deciding whether the stakeholders, for example, opt more for alerts through SMS or through Email, thus strengthening itself as a requirement, albeit elicited, of the system. Another example of such a differentiating factor would be if the bulk of the intended users would appreciate location tracking of about an hour, under an hour or more than an hour – an aspect that is bound to alter the requirements of the system.

1. **Scenarios**

When you are building a system, there are numerous opportunities for unexpected events and scenarios to occur – like the system application crashing unpredictably, or skipping an entire module/phase, however important, while running. We cannot account for every possible reaction or activity to such events, but we can handle some pivotal scenarios to avoid precarious future situations by setting up requirements that may help in evading them. For example, when we initially thought of building our system, there was no requirement of authentication and verification of members who could track a certain member - anyone with your metadata from the system application could end up tracking you. This highly-prone-to-occur scenario led us to ensure that two-way verification and authentication must be an essential requirement of our system, so that a user’s location does not end up in the wrong hands or end up with someone without the user’s prior knowledge. Hence, some forethought of brainstorming requirements for scenarios that will inevitably occur or that have a high probability of occurring can handle the system better and more efficiently since those requirements will specifically address them. Scenarios can also help us in devising the system flow better and handle errors or bugs that can otherwise needlessly materialize.

1. **Use Cases**

Use Cases in our system set-up will not be as useful as the other techniques in eliciting requirements, since the environment of use cases mainly interacts with the system after it has been built, rather than participating in features that must build the system up. A use case is a list of actions or steps of events typically defining the interactions between an actor and a system to achieve a goal or goals. Those goals are intended to be requirements of that system, hence use cases cannot add to that set of requirements – rather it can ‘test’ that set of requirements. It is a type of technique that if it were to add to the requirements set, it would converge entirely in the ‘Scenarios’ technique, thus leaving no room for its singular application of technique. The application of a use case, for example, may lead us to a scenario where a requirement or goal is not achievable at all; therefore it may necessitate a new requirement to take its place while achieving the envisioned functionality. Such events more than usually fall in the application of a ‘Scenario’ technique, hence dedicating an entire technique that is a subset of another larger and more useful technique will be wasteful and inefficient of our system. However, if we plan to add the feature of supporting multiple sub-families, then Use Cases becomes an essential technique for identifying requirements that heavily depend on multiple interactions of the system and each of those multiple sub-families.

1. **Ethnography**

Ethnography will prove to be a useful technique for eliciting requirements of our system (more so if we plan to build it incrementally), since often users and consumers are not as eloquent and concise in extending their needs and expectations (in interviews, for example) and thus cannot portray an accurate representation of these needs and expectations as types of requirements due to these communicational gaps. Often, these requirements also seem to change over time. Ethnography can help in identifying the real need of a user rather than the one he is presumably communicating correctly but not sufficiently, and helps to avert the likely possibility of the system not meeting a user’s envisioned set of requirements and hopes from it. This ensures that user satisfaction from the system is maintained till the very end of the product. For example, the requirement of pinpoint location of a member through our system will regardless require location tracking of that member from the point he has activated his location services. However, the requirement demands a precise location rather than a track – hence, requirement conflict can occur when differentiating between the two – does the user want a location track or does the user want a pinpointed location? Here, Ethnography comes into play – the technique knows that the requirement is to achieve a precise location but it also realizes that to achieve this requirement another requirement (that of keeping a track of the user’s location) is needed to be fulfilled first. This distinction between what a user demands and what he actually requires is made clear only through the technique of Ethnography. Thus, it is quite useful for the process of requirement elicitation in our proposed system.

* **Game Developer:-**

Our system is not a game dependent one – hence, the question is not applicable to it.

* **Requirements Grouping**

1. **Categorize your system requirements into functional and nonfunctional requirements.[[1]](#footnote-1)**

* **Functional Requirements:**

1. Reliable location tracking of registered family members.
2. Exact and a pinpointed location of the member to be tracked.
3. Emergency buttons for alerting family members in matters of danger (for example, if you are kidnapped).
4. SMS Alerts - that are sent when the emergency button is clicked/pressed - to family members that are accompanied by embedded links of your current location (or last known location).
5. Concept of alarms upon entering restricted locations, violating set perimeters or upon receiving alerts.
6. Multiple family members tracking.
7. Push notifications of the application – for prompting user to turn on GPS, WiFi, Mobile Data, any alerts, etc.

* **Non-Functional Requirements:**

1. Entire location track of the member to be tracked, given time constraints (for example, the entire location track of the member since the past hour or two).
2. Authentication before tracking.
3. Two-way verification/authentication approach (Family members must verify other ‘watcher’ members for tracking).
4. Thresholds for distances – location tracking of even a minute difference in location and location tracking of generalized or expansive differences in location.
5. Multiple watchers – those who can track you.
6. **For each of the non-functional requirements, also identify its type.**

* **Non-Functional Requirements:**

1. Product Requirements -> Usability Requirements

Product Requirements -> Efficiency Requirements -> Performance Requirements

1. Product Requirements -> Dependability Requirements, Security Requirements
2. Product Requirements -> Security Requirements
3. Product Requirements -> Dependability Requirements

Product Requirements -> Efficiency Requirements -> Performance Requirements

1. Product Requirements -> Usability Requirements, Dependability Requirements, Security Requirements
2. **For each non-functional requirement, define metrics that can be used to measure it.**

* **Non-Functional Requirements:**

1. **Speed** – Given time constraints, this metric will be essential in correlating speedy location tracks to its intended users while ensuring that user/event response time can be easily quantified and updated vigilantly. Screen refresh time will enable constant updates in the location track to be relayed to the user. For realistic tracking of location, speed should take a correlation rate of just over a minute (90 seconds at best).

**Reliability** – This metric will ensure that the location track being provided is unfailing and correct.

**Portability** – This metric will ensure that each of the target systems (for example, in the case of multiple family ‘watchers’ requiring location track of the same family member) receives the location track efficiently and in due time.

1. **Speed** – This metric will ensure that the user/event response time is swift for authentication correspondence.

**Reliability** – The authentication process of enabling users to track you (also an intended user) requires dependability on providing that tracking right and access to the correct users - a quality this metric can provide.

1. **Speed** – This metric will ensure that the user/event response time is swift for authentication and verification correspondence.

**Reliability** – The authentication process of enabling users to track you (also an intended user) requires dependability on providing that tracking right and access to the correct users - a quality this metric can provide.

**Portability** – This metric will ensure that each of the target systems (for example, in this case of family ‘watchers’ requiring location track of a family member) receives the authentication and verification correspondence efficiently and in due time.

1. **Speed** – This metric will be essential in correlating speedy location tracks to its intended users while ensuring that user/event response time can be easily quantified and updated vigilantly. Screen refresh time will enable constant updates in the location track to be relayed to the user, with even minute differences in location showing up in near-constant time.

**Size** – This metric will ensure that the constantly updating and expanding location tracks are saved and tallied to its users in near-constant time.

**Reliability** – The process of enabling users to track you (also an intended user) constantly and keep track of each minute change in location requires dependability on providing that tracking correctly in the first place - a quality this metric can provide.

1. **Speed** – This metric will ensure that the user/event response time is swift for such hierarchical correspondence between multiple watchers and the user to be tracked. Screen refresh time will enable constant updates in the location track to be relayed to these users.

**Reliability** – This metric will ensure that the location track being provided to the multiple watchers is unfailing and correct.

**Portability** – This metric will ensure that each of the target systems (for example, in the case of multiple family ‘watchers’ requiring location track of the same family member) receives the location track efficiently and in due time.

**Note**: These metrics can be verified and measured through and by applying procedures of mathematical verification of correctness to distance scaling and location relay, speed-testing tools like accelerometers, and prediction analysis of past trials of the system, when it is launched, through statistical tools and languages (like MATLAB and R).

* **Requirements Validation**

1. **Identify conflicts in your requirements, if any. Document the conflicts and the mechanism you used to fix them.**

* **Conflict 1, between requirements:**

1. Reliable location tracking of registered family members.
2. Entire location track of the member to be tracked, given time constraints (for example, the entire location track of the member since the past hour or two).

To ensure reliable and efficient location tracking, ‘ideally’ one would not add any time constraints into the tracking mechanism. However, interfaces like the ‘LocationListener’ - that are primarily used for updating location and location tracking in Android-based Java - demand some limitation of tracking time in their modules. Such a mechanism would drain the computing power and battery of the device the system would be installed on as well. Thus, realistically, we cannot satisfy both requirements at the same time – one requirement has to be undermined and compromised. We limit this reliable location tracking to a time constraint because constant tracking since the installation of the system to its presumed end is along the lines of uncertainty. The mechanism we used to fix this conflict was simply increasing our time constraint to one that was ideally suitable, large and satiable for reliable and efficient location tracking.

* **Conflict 2, between requirements:**

1. Emergency buttons for alerting family members in matters of danger (for example, if you are kidnapped).
2. SMS Alerts - that are sent when the emergency button is clicked/pressed - to family members that are accompanied by embedded links of your current location (or last known location).

If location-providing services like GPS or WiFi are not enabled or permissible while using the system (at some given instance i.e.), the embedded location links that get sent as SMS alerts upon clicking the ‘Emergency’ button will not be able to get the location of the proposed user at all. Some combining factor between the two requirements is needed in such a situation to ensure some type of reliable location to be sent along the embedded link, hence the mechanism we used to fix this was by proposing to store the last known location of the user indefinitely (when his location providing services were activated i.e.), concatenate it along the embedded link and send it forth as a usual SMS alert, if such a set-up was to occur.

1. **Can you actually build all the requirements using the current technology?**

Using current technology, all our requirements seem feasible enough to be established and achieved. Our initial aim is to develop and launch it in Android Studio using Java Language, which houses many in-built libraries and modules (stated in Homework 1 and in above passages as well) that can be used to ease any build of the system and its requirements. On the contrary, current technology such as location providing services through cellular networks and Global Positioning Systems (GPS) may make it easier for all requirements to be fulfilled.

1. **If there are any requirements that you cannot build, how can you possibly modify them to make them realistic?**

24/7 tracking of location is not an intended requirement of our system, but one that is unrealistically expected from a family tracker application by the majority of its intended users. This is neither feasible nor possible; with limited storage and easily drainable batteries of most devices nowadays we cannot comprehend such a facility in the targeted devices at hand. To make such a requirement genuine and a concrete part of our system, we would have to scale the feature back to enable as much time constraints as can possibly be handled efficiently by these targeted devices and systems. This way, tracking of location that is physically possible by the system can be achieved and yet will still have flavors of the ‘idealistic’ expectation.

1. **Which of the following techniques you think could be used to check verifiability of your system requirements and why:**
2. **Requirements reviews**

This technique would be an indispensable part of checking the verifiability of our system requirements, since it is the procedure of going through the system requirements with the intended stakeholders of the system and looking at different aspects of each requirement. Such reviews bring forth (or clear out) issues like whether each requirement is realistically testable, whether each requirement has been properly understood by all interested stakeholders of the system, whether each requirement’s origin is clearly stated and/or whether each requirement is in harmony with the other requirements or is in conflict. This technique would help in weeding out requirements that are not in accord with the system and help in keeping those requirements that are in unity with the system. Each requirement of our proposed system has a heavy impact and influence on the overall system; hence this process would provide the system a more efficient curve of growth comparatively.

1. **Prototyping**

The main focus of our proposed system is to successfully track the location of our users – it is a pivotal requirement of our system upon which other requirements have laid foundation on. If we plan on building a system that can track an entire family, we must implement a successful system that can atleast track a singular user or provide his pin-pointed location. A prototype of our family tracker system, modeling the tracking of a singular user and a single ‘watcher’ would massively help in checking the verifiability of our system’s main requirement without giving rise to complicated structures that may be likely prone to fail.

1. **Test-Case Generation**

Generating and implementing test cases can help in checking whether each system requirement is being met or not. It can also help in finding dubious bugs and patterns that builders of the system could not otherwise have found. Mathematical verification of correctness of the system can also be achieved by applying such test cases to such verification techniques as aids of environmental information for testing purposes.

* **Introduction**

The Family Tracker application will serve to keep and show location track, while also providing concise location, of each registered family member. Each family member will be allotted a ‘watcher’ family member that they can authenticate and verify for tracking them – a methodology to provide essential security and knowledge of allowing people to access your location. With social issues like elopement, child kidnapping, ransom, and the general lack and quality of security and safety, a family tracking system would assist in providing consistent and constant knowledge of wellbeing and whereabouts of each member of the family to every other member of the family. Features like SMS alerts, alarms and emergency buttons will also aid in notifying and warning concerned members and the general populace of any imminent danger that any member could be in with relevant location information.

Given below is a high-level overview of the system and a rudimentary UML diagram for reference:-

If already user?

Login Activity

Sign Up Activity

Menu Activity:

Alerts Activity

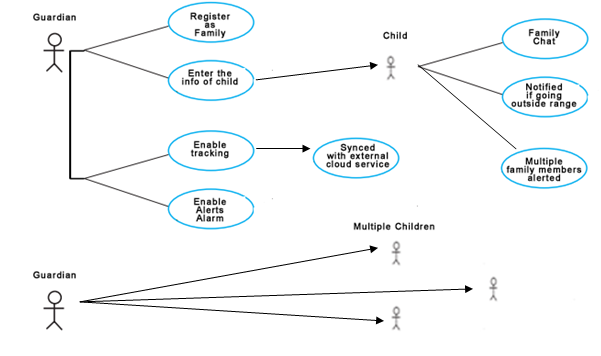
Alerts

Maps Activity

Location Activity

Track

Exact Location

****

* **Hardware and Software Requirements**

1. Android phones/devices with an Operating System more than 4.2.2.
2. Minimum RAM (Random Access Memory) of 1 GB.
3. A preferred processor of 1.2 GHz dual core processor.
4. Minimum storage of 1 GB to keep backups of the tracked positions and map data.
5. WiFi/3G connectivity.
6. Global Positioning System (GPS) services and permissions to access instant/track of location.
7. Global System for Mobile (GSM) connections that are preferably local e.g. Warid, Telenor, etc.
8. Updated Android Studio software for building.
9. Testing tools like JTest, JUnit and Litmus.
10. Cookies and Javascript must be enabled when using outsourced and aiding applications, like Google Maps.
11. Broadcast Receiver and service functionalities for outsourced and aiding applications, like Google Maps.
12. Android devices that have Intel or NVIDIA Graphics card in their models.

**Note:** Our functional and non-functional requirements are a constituting part of the software requirements of our system – they have been stated originally in this document prior to this discussion. Reasoning to incorporate the above mentioned requirements is mainly the affect over performance and a reliable, fall-back system - known to provide quality service and experience.

1. <https://stackoverflow.com/questions/16475979/what-is-functional-and-non-functional-requirement>

   **Note:** Non-functional requirements were categorized under this answer’s basis and the course instructor’s provided slides. [↑](#footnote-ref-1)