Data management component of ManDown application

Arshan Shafiei

***Abstract*—ManDown is a fun social app which aims to monitor and improve user's drinking pattern. This is done by diagnosing alcohol consumption of the user and use machine learning techniques to evaluate the user's intoxication state. In order to meet this, one of the main components needed is the Data management module. In this report, I explain fully what this module does in detail, what are the main design challenges by referring to related work carried by others, how it connects to other parts of the system, the implemented functionality so far and further actions need to be taken.**

I. Introduction

ManDown is designed to diagnose and help user's drinking habit based on the data being passed to it. Just like any other mobile app, data management is a crucial module in the overall design of this application. The back-end database is responsible for collecting the data from the user, storing data in an appropriate way while considering available memory and bandwidth constraints to eventually transfer data back to the front-end user. Furthermore, it forms the foundation of machine learning aspect of the system.

"A database is a collection of information that is organized so that it can easily be accessed, managed, and updated" [13]. Databases are managed by Database Management Systems (DBMS) which allow fetching ,updating,storing and removing data. For our case, Our database is a mobile database, which differ from other database types in many aspects. For example, mobile databases are single user systems, and therefore a concurrency control mechanism is not a problem [3]. The mobile database required for ManDown is responsible for storing the result collected from the sources and passing it on to the machine learning component for classification.

The input devices used in this project are Android mobile phone and smart watch. Data are gathered from these device's using two methods:

1) Active method: These data are collected directly from the user. In particular, our application consists of games that are designed to ask how much the user has consumed alcohol while keeping them entertained . Furthermore, these games will measure the user's reaction time as another symptom of intoxication.
2) Passive method: The passive observation constantly monitors the behavior of the user using embedded sensors available on the phone and watch. These sensors aim to measure signs of alcohol intoxication such as loss of control.

II. Requirements

In order to manage the database,it is worth explaining what data are intended to capture from the sensors and how they relate to the corresponding alcohol intoxication symptom. Our system will be designed to capture three main symptoms of alcohol intoxication. These symptoms include [8]:

1) Ataxia: This involves loss of coordination and appears clearly in the walking pattern [11]. It is often classified as one of the most accurate symptom of alcohol intoxication.
2) Slow reaction time: As accurate as Atexia, time to react to an action is significantly affected.
3) Loss of motor control: This involves the inability to control the body and in extreme cases results in crashes and damage.

Other symptoms like slurred speech or rapid eye movement may be useful but are not included in our database due to lower indicator of intoxication. In order to capture the described symptoms, our system uses the sensors found in the smart phone and smart watch. Our data collection input (based on the described symptoms) can be categorized to the following:

1) Body motion data: These data are collected by the accelerometer and gyroscope in both the watch and mobile device to target Atexia and loss of motor control.
2) Touch input: These data are gathered from user's reaction time to gamification tools developed and can be used to target slow reaction time symptoms.

While gathering data from the sensors,developing the games and the user interface is done by other group members, my role is to contribute to the data management aspect of the project in collaboration with Max Poynton ,which focuses more on the machine learning sub module.

A. Android applications

Since our data is stored on supported Android databases, it is crucial to have a basic understanding of how Android operating system works as a requirement of data management. Android is an open source operating system developed by Google based on Linux Kernel. An important part in Android operating systems is the Dalvik virtual which is similar to Java virtual machine and executes the generated bytecode.

An android application overall is consisted of an *activity,service, broadcast receiver* and *content provider* [6]. An *activity* is simply a single screen of the application. On the other hand, a *service* and *broadcast receiver* are not visible and runs in the background. For Data management purposes, The important component of the system is the *content provider* which provides access to the data from Android software point of view. The connection of the described components are done by an Extensible Markup Language(XML) file called *Android manifest* [6].

B. Sub goals in Data management module

The overall Data management module role is categorized in to several tasks for this application as described in the following list:

1) Data collection: Ensuring the data is smoothly collected from the gyroscope, accelerometer and gamification input and are stored the correct format.
2) Data transfer: Ensuring the communication channel used for data transformation is suitable for our ManDown application.
3) Data safety and security: Ensuring data is stored safely with necessary security encryption implemented. This a fundamental part of data management we are dealing with confidential data from the users.
4) Data consistency: All devices see the same version of data with no inconsistency involved.
5) Data availability: The system always responds within fixed upper limits of time considering the constraints in bandwidth.
6) Partition Tolerance: The system is always available even when messages are lost or even Internet connection is lost.

The latter three (*Consistency, Availability and Partition Tolerance*) are known as *CAP* theorem in databases which states that no distributed system can maintain all three of those properties simultaneously [5]. Details of how the above goals are achieved is later described in the implementation section.

III. Existing Data management research

Data management has been an ongoing research from over a hundred year ago. it would be useful to have a basic understanding of the underlying principles used in order to choose which one best fits our purpose. Data models are categorized to four types:

1) Unstructured data: These data as it appears from their name are not organized in a predefined manner. They include multimedia files, documents, spreadsheets, news, emails,memorandums, reports and web pages are difficult to capture and store in the common database storage due to the complexity of their search algorithm [2].
2) Flat files: Data are stored in plain text format and separated by commas or tabs like CSV files. Each line of the text file is one record However, It does not have a structured relational format. These models is widely used in Data warehousing projects.
3) Semi-structured data: These type of data do not have columns and tables like relational databases however, they have some semantics to explain their fields. Good examples could be Java Script Object Notation (JSON) or XML files commonly used in Non-relational databases.
4) Structured data: These data are stored in relational columns and tables. Each table is given a key (primary key) which acts as a unique column for each entry. Furthermore,tables can be joined using specific attributes as foreign keys if needed.

Having the above data models in place, databases structures are described using their different levels of schema's. The *internal schema* of a database describes it's physical layout of data. This schema for example explains what page sizes are used and what method is used for indexing. Another type of database schema is the *conceptual schema* of a database. While the internal schema focuses on describing the physical layer, conceptual schema describes the tables and their attributes(columns). It also manages query processing using databases *data manipulation languages* [4].

The most common type of data manipulation language (*DML*) is Structured Query Language (SQL). It performs create, read, update and delete (*CRUD*) functionality in relational databases. There has been a few standards for the SQL language provided by ANSI, but many database vendors tend to provide more functionality by adding extra extensions and releasing their own version [7]. The SQL syntax is based on relational algebra and is consisted of clauses, expressions, predicates and statements enabling many query processing features. When a SQL query (usually starts with *SELECT* statements) is executed, the relational database management system (RDBMS) performs planning and optimization to handle the query and produce the result. Examples of RBDMS SQL databases are MySQL, dBase, Oracle, Microsoft SQL server and PostgreSQL.

In contrast, Due to the overhead introduced by providing full relational DBMS, *No-SQL* databases do not follow the traditional relational model and are designed for handling non structured or semi structured data. These databases are useful for managing huge amount of data as a result have seen major advancements in recent years. In particular, referring back to the CAP theorem,in return to not having organized structured, they perform better than relational databases in maintaining *ACID* properties (Atomicity, Consistency, Isolation and Durability) of transactions in databases [7]. Moreover, they have a better handling of network partitions and replication of data in large number of servers. Most popular No-SQL databases are MongoDB, CouchDB, Amazon DynamoDB and Hypertable.

A. Existing mobile databases

Mobile databases are a distributed database which operate on smart phones and are responsible for managing mobile user's handheld data. While they resemble to large-scale commercial databases, they differ from them in many aspects. Local mobile databases have lower CPU power for query processing, and consume battery power, which is essential for embedded platforms like mobile phones and smart watches [1]. Mobile databases also need to satisfy network transmission speed, available bandwidth, geographical location and data consistency (known as *replication in mobile computing* when accessed from multiple devices .

In general, mobile databases follow two architectural models when it comes to interfacing with the application:

1) Local databases: These type of databases store data locally on either the internal built-in memory or on an external memory attached to the device. Their capacity as a result is limited to the available local memory. Furthermore, the stored is data lost if the device is destroyed or stolen. On the other hand, they do not requite Internet connectivity for data access and are not constrained in bandwidth. The most popular embedded databases in the market are currently BerkeleyDB, CouchBase Lite and SQLite supporting both iOS and Android platforms. For the purpose of our ManDown application, it is useful to know benefits and drawbacks of common Android local databases.

   BerkeleyDB: Outweighs from other Android local mobile databases in terms of data management support as it allows storing data in any relational, key-value, Java objects or XML format [12].

   Couchbase Lite: Classified as No-SQL database, data are stored as JSON format. This database also offers cloud storage on Couchbase servers and is known to be very efficient at synchronization between it's cloud and local storage. [12].

   SQLite: This database is contained in an open source C library, which supports SQL and gets embedded in to the program (not a client-server engine). It is also commonly used to store and retrieve information within the browsers.

2) Cloud databases: With the growing market of cloud computing, these databases are widely used in mobile applications in today's world. They store data on cloud and require an Internet connectivity to transfer/fetch data from the cloud storage. Most popular Android cloud storage's in the market are:

   Amazon Web services: This service provides access to both SQL and No-SQL databases. "Amazon Relational Database (RDS) run either MySQL, Oracle or SQL Server instances, while Amazon SimpleDB is a schema-less database meant for smaller workloads"[9].

   MongoLab: popular No-SQL cloud database, this database runs on other cloud providers like Amazon, Google and Microsoft Azure and integrates with Platform as a service(Paas) tools.

   Firebase: Again a No-SQL database, data are stored as JSON format and synchronized in real time to all the devices. This database allows data to be available when the app goes offline. It also gives access to the storage data via HTTP request on web.

Many of the described database vendors use a combination of both local and cloud databases to improve user experience. In other words, data are stored locally and synchronized to the cloud if device goes on line.

In cloud databases, The android app connects to the web service via HTTP (HTTPS could be used for extra security) which can then enable access to the back-end web database as shown in the diagram below. These type of web services are called RESTful web services and can be built using PHP or Node. js. The back end database could be any of the introduced cloud databases.

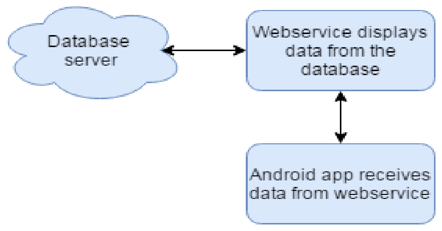

Database server
Webservice displays data from the database
Android app receives data from webservice

Figure 1. Remote database architecture

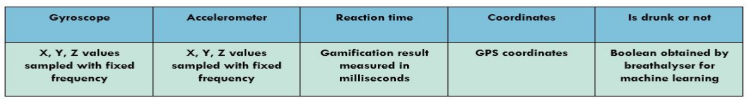
IV. Implementation

In order to have a working prototype as quickly as possible and to get experience with Android databases, Different methods of storing data have been practiced on Android Studio and it's suitability has been assessed for ManDown application. Due to the arguments raised before (applying machine learning, data loss etc. .), we are targeting cloud databases as it's suit most to our application However,each user's data can optionally be saved on the local device as well.

A. *Local storage*

There are three way of storing data locally on Android devices:

1) key-value sets: This method has been practiced is for storing very small amount of data as key-value pair in files. Since each user's has large amount of data, this method is not applicable to our application.
2) Saving files: The *File* API allows android developers to save files on internal or external storage subject to availability on devices. Data stored on internal disk space is private and only accessible by the associated application [10]. While this method is suitable for large data, it does not support query processing which is crucial for ManDown data,as we are dealing with statistical drinking patterns which require relational algebra operations for data management.
3) SQL databases: By default, Android stores data in SQLite database available in *android. database. sqlite* package. A table for each user has been created in SQL with columns of sensor readings and gamification result filled with dummy data for the moment. Queries have been written in SQL to get the get statistical data for practice in Android Studio. Figure bellow illustrates how our table created looks like.



Gyroscope	Accelerometer	Reaction time	Coordinates	Is drunk or not
X, Y, Z values sampled with fixed frequency	X, Y, Z values sampled with fixed frequency	Gamification result measured in milliseconds	GPS coordinates	Boolean obtained by breathalyser for machine learning

Figure 2. Data capture of ManDown application

B. Cloud storage

Comparing introduced databases, the most suitable cloud database for the purpose of our application is Firebase. This database abstract away the design of communication channel, creating tables etc. . and provides an encrypted No-SQL cloud database for free (5GB maximum). Data are persisted to the disc which makes the app responsive even offline. This is a useful feature for ManDown users as we do not want to lose our data if Internet connection is lost. It allows data access via HTTP requests as well, which enables flexibility of design for data to be passed to the machine learning component. It is also provides many data security rules to encrypt the database.

C. Data security

As mentioned before, Since many people do not want their drinking habits to be available online, data security plays an important role in our data management. The data stored locally on the internal memory of the device can only be accessed from the app and can further be encrypted on the SQLite database. From the point of view of cloud storage, Firebase allows HTTPS instead of HTTP which provides an extra level of data communication security. From an application point of view, Firebase also offers creating security rules like users identification methods such as user-name/password to grant access to the identified user to improve the overall security of the system.

V. System design

The data stored will be passed to the machine learning module which does the binary classification of data (intoxication of the user). The data management overall, collects the data from the sensors, while acting as an intermediate node to the user interface. Figure bellow shows the overall system design of our application.

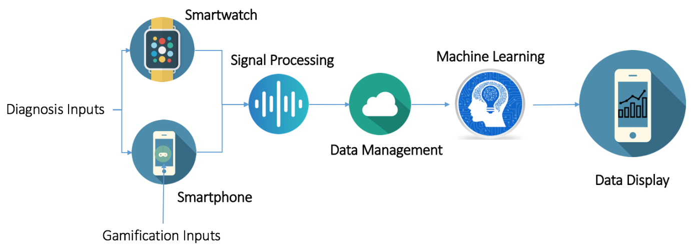


Figure 3. ManDown system design [8]

VI. Conclusion

Data management module plays a crucial role in the performance and evaluation of ManDown application. It involves storing data in the correct, Managing data transfers while considering data privacy and security. I will continue to capture data on the cloud database, implement the security rules and users authentication while focusing more on how to connect the outputs to the machine learning and eventually the user interface of the system.

References

[1] W. Li, H. Yang, and P. He. “The Research and Application of Embedded Mobile Database”. In: *2009 International Conference on Information Technology and Computer Science* 2 (July 2009), pp. 597–602. DOI: 10.1109/ITCS.2009.304.

[2] S. Z. Z. Abidin, N. M. Idris, and A. H. Husain. “Extraction and classification of unstructured data in WebPages for structured multimedia database via XML”. In: *2010 International Conference on Information Retrieval Knowledge Management (CAMP)* (Mar. 2010), pp. 44–49. DOI: 10.1109/INFRKM.2010.5466948.

[3] LING LIU and M. TAMER ÖZSU. “Encyclopedia of Database Systems”. In: *Springer Science Business Media, LLC 2009* (2010), pp. 1–2. DOI: 10.1007/978-0-387-39940-9_1362.

[4] X. Cuiling. “Research on the architecture of spatial data manipulation language”. In: *2011 IEEE 3rd International Conference on Communication Software and Networks* (May 2011), pp. 796–798. DOI: 10.1109/ICCSN.2011.6015009.

[5] S. Gilbert and N. Lynch. “Perspectives on the CAP Theorem”. In: *IEEE Computer Society* 45.2 (Feb. 2012), pp. 30–36. ISSN: 0018-9162. DOI: 10.1109/MC.2011.389.

[6] C. Szabó et al. “Database refactoring and regression testing of Android mobile applications”. In: *2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics* (Sept. 2012), pp. 135–139. ISSN: 1949-047X. DOI: 10.1109/SISY.2012.6339502.

[7] Y. Li and S. Manoharan. “A performance comparison of SQL and NoSQL databases”. In: *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)* (Aug. 2013), pp. 15–19. ISSN: 1555-5798.

[8] Diyar Alyasiri et al. *Our First design report*.

[9] Brandon Butler. *10 of the most useful cloud databases*. URL: http://www.networkworld.com/article/2162274/cloud-storage/cloud-computing-10-of-the-most-useful-cloud-databases.html.

[10] Android developers. *Saving Data in SQL Databases*. URL: https://developer.android.com/training/basics/data-storage/databases.html.

[11] National Ataxia Foundation. *DIAGNOSIS OF ATAXIA*. URL: http://www.ataxia.org/learn/ataxia-diagnosis.aspx.

[12] KATERINA ROUKOUNAKI. *Five popular databases for mobile*. URL: https://www.developereconomics.com/five-popular-databases-for-mobile.

[13] Margaret Rouse. *database*. URL: http://searchsqlserver.techtarget.com/definition/database.