
ETD Archive

2019

Usability Engineering of a Privacy-Aware Compliance Tracking System

Parameswara Reddy Annapureddy
Cleveland State University

Follow this and additional works at: <https://engagedscholarship.csuohio.edu/etdarchive>



Part of the [Computer Sciences Commons](#), and the [Medicine and Health Sciences Commons](#)

How does access to this work benefit you? Let us know!

Recommended Citation

Annapureddy, Parameswara Reddy, "Usability Engineering of a Privacy-Aware Compliance Tracking System" (2019). *ETD Archive*. 1133.

<https://engagedscholarship.csuohio.edu/etdarchive/1133>

This Thesis is brought to you for free and open access by EngagedScholarship@CSU. It has been accepted for inclusion in ETD Archive by an authorized administrator of EngagedScholarship@CSU. For more information, please contact library.es@csuohio.edu.

USABILITY ENGINEERING OF A PRIVACY-AWARE COMPLIANCE TRACKING
SYSTEM

PARAMESWARA REDDY ANNAPUREDDY

Bachelor of Engineering in Electronics and Communication Engineering

Vasavi College of Engineering, Hyderabad, Telangana, India

May 2011

Submitted in partial fulfillment of requirements for the degree

MASTER OF COMPUTER AND INFORMATION SCIENCE

at the

CLEVELAND STATE UNIVERSITY

May 2019

© COPYRIGHT BY PARAMESWARA REDDY ANNAPUREDDY 2019

We hereby approve this for

PARAMESWARA REDDY ANNAPUREDDY

Candidate for the MASTER OF COMPUTER AND INFORMATION SCIENCE degree

for the

Department of Electrical Engineering & Computer Science

and the CLEVELAND STATE UNIVERSITY'S

College of Graduate Studies by

Committee Chairperson, (Wenbing Zhao)

Department of Electrical Engineering and Computer Science, 05/06/2019

Committee Member, (Yongjian Fu)

Department of Electrical Engineering and Computer Science, 05/06/2019

Committee Member, (Lili Dong)

Department of Electrical Engineering and Computer Science, 05/06/2019

Student's Date of Defense: 05/06/2019

DEDICATION

TO MY WIFE, FAMILY, AND FRIENDS WHO STOOD BY ME IN ALL TIMES...

ACKNOWLEDGMENT

Firstly, I would like to express my sincere gratitude to my Prof. Wenbing Zhao for the continuous support of my thesis study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better mentor for my thesis study.

Besides Prof. Wenbing Zhao, I would like to thank my academic advisor Prof. Yongjian Fu for his insightful comments and encouragement, but also for the hard question which incited me to widen my research from various perspectives. I would also like to thank my thesis committee: Prof. Yongjian Fu, Prof. Lili Dong, and Prof. Wenbing Zhao, for being my thesis committee members.

USABILITY ENGINEERING OF A PRIVACY-AWARE COMPLIANCE TRACKING SYSTEM

PARAMESWARA REDDY ANNAPUREDDY

ABSTRACT

Software is useful when it is able to provide useful information to the end user with minimum effort. This thesis is about usability improvements to a privacy-aware human motion tracking system for healthcare professionals. The original system has a number of usability issues: (1) Users need to wear a smartwatch, which will be used to connect to the system; (2) Data are stored in XML, comma-separated-value format which is very difficult to analyze; (3) Data are available only at the local computer and there is no easy way to access them remotely via a Web or mobile interface; (4) Analysis and comparison of different user's performance is very difficult.

In this thesis, we mainly concentrated on solving the above-mentioned usability problems. We introduced a new way of authenticating users for motion tracking using Bluetooth beacons, thereby eliminating the requirement of wearing a smartwatch. We built a Web application which will accept data from Kinect and stores data in the database. This Web application will present the logged data to users via a variety of charts and it can be accessed from anywhere by using a web browser. Along with this, to provide easy access to data, we also provide a mobile application to present users with data analysis and comparison features. Both the Web and the mobile applications are designed to make sure that user privacy is maintained, and proper access controls are imposed on who can see what data.

TABLE OF CONTENTS

	Page
ABSTRACT	vi
LIST OF TABLES	ix
LIST OF FIGURES	x
CHAPTER	
I. INTRODUCTION	1
II. AUTOMATIC USER AUTHENTICATION WITH BLUETOOTH	5
2.1 OUR SOLUTION	5
2.2 DEVELOPMENT	9
III. CLOUD-BASED DATA AGGREGATION	11
3.1 OUR SOLUTION	12
3.2 SYSTEM REQUIREMENTS	13
3.3 SYSTEM DESIGN	13
3.4 DATABASE DESIGN	15
3.5 WEB DEVELOPMENT	17
IV. WEB INTERFACE FOR DATA VISUALIZATION AND ANALYSIS	20
4.1 OUR SOLUTION	20
4.2 SYSTEM REQUIREMENTS	21
4.3 SYSTEM DESIGN	22
4.4 USER STORIES	23
4.5 DEVELOPMENT	24
4.5.1 Login Screen	26
4.5.2 Dashboard Screen	26
4.5.3 Users Management	27

V.	MOBILE APP FOR DATA VISUALIZATION AND ANALYSIS	28
5.1	OUR SOLUTION	28
5.2	SYSTEM REQUIREMENTS	29
5.3	SYSTEM DESIGN	30
5.4	USER STORIES	31
5.5	ENVIRONMENT SETUP	32
5.6	DEVELOPMENT	32
	5.6.1 Login Screen	33
	5.6.2 Dashboard Screen	34
VI.	ANALYSIS OF DATA USING NEW DATA VISUALIZATION SYSTEM	38
VII.	CONCLUSION	45
	BIBLIOGRAPHY	46

LIST OF TABLES

Table	Page
1. User table schema	16
2. Session table schema.....	16
3. Activity table schema.....	17

LIST OF FIGURES

Figure	Page
1. Privacy-aware compliance tracking system architecture	3
2. Sticker beacon by estimate.....	6
3. Bluetooth beacon-based authentication system	8
4. Sample JSON	12
5. ER diagram of our entities	14
6. Data processing system architecture	15
7. Web interface architecture	22
8. Login screen.....	23
9. Dashboard screen	24
10. Administrator dashboard.....	26
11. Android application architecture.....	30
12. Login screen.....	34
13. User selection.....	34
14. Date selection control	35
15. Input selection.....	36
16. Generate chart	37
17. Wrong activity duration trend.....	38
18. Multiple user's wrong activity average duration over time	39
19. Wrong activity duration on each date	40
20. Average wrong activity duration in a day over time.....	41
21. Average wrong activity duration over a weekday	41

22. Number of wrong activities on each weekday	42
23. Wrong activity trend over the month	43
24. Relative Risk Factors	44

CHAPTER I

INTRODUCTION

A software success is mainly dictated by its core features and usability of the software. The definition of usability is in some cases just described as "simple to use". According to ISO 9241-11 [1] "The extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency, and satisfaction in a specified context of use".

- **Effectiveness** is the completeness and precision with which clients accomplish specified objectives. It is decided by looking at whether the user's objectives were met effectively and whether all work is correct. The quality of the client help built into the interface can have a solid effect on effectiveness. The effectiveness of an interface regularly depends on the introduction of choices in a way that's clearly justifiable to the client. The more instructive an interface can be, the better users are able to work in it without issues. Great interface terminology will be within the user's dialect and fitting to the purpose.
- **Efficiency** can be described as the speed (with precision) in which clients can accomplish the assignments for which they utilize the item. ISO 9241 defines effectiveness as the whole assets used in an assignment. Effectiveness

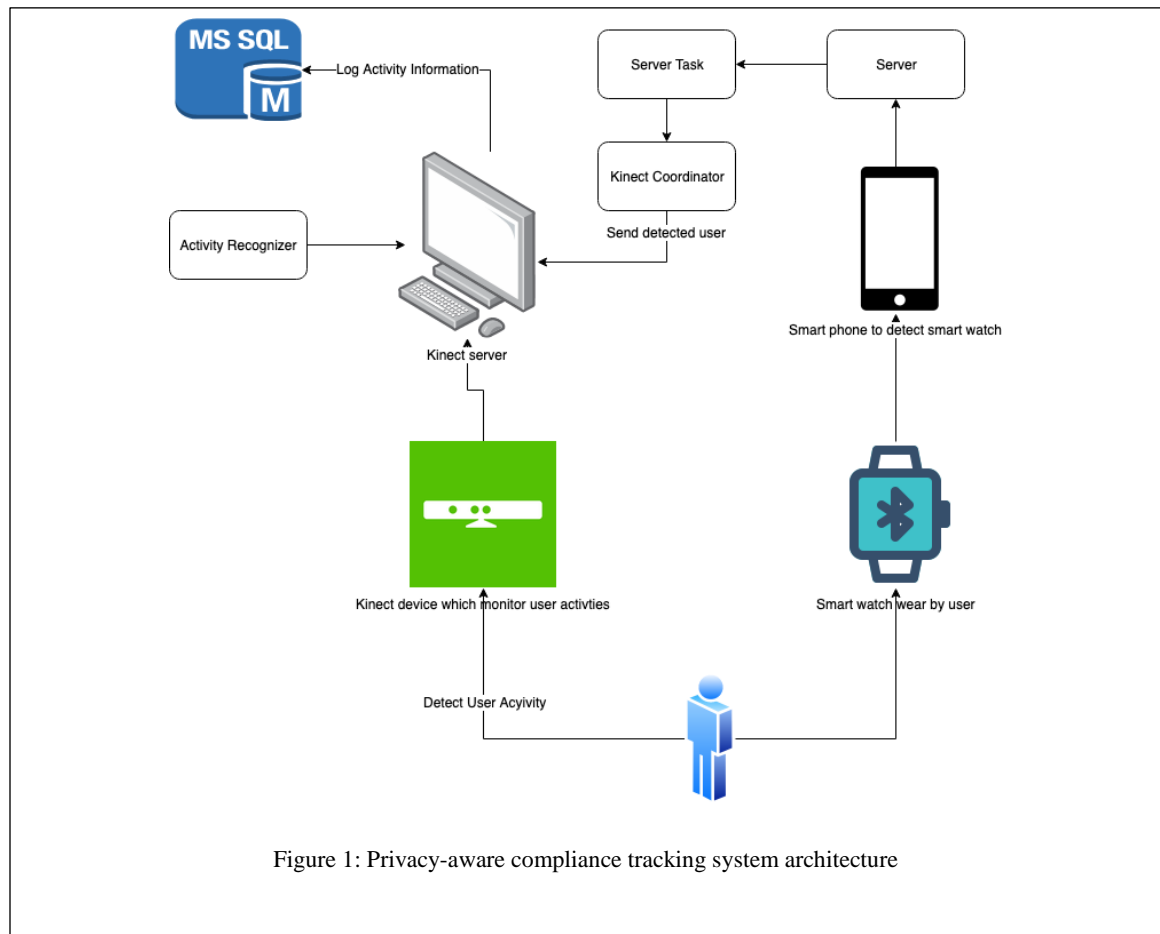
measurements incorporate the number of clicks or keystrokes required or the overall 'time on task'. Making the proper choices for efficient utilization of the software depends on an understanding of the clients and how they want to work.

- **Satisfaction comes when** an interface is pleasant. The visual design is the foremost self-evident component of this characteristic. The style of the visual presentation, the number, capacities, and sorts of realistic pictures or colors, and the utilize of any interactive media components are all portion of a user's quick response.

Usability and client-focused design are iterative. We have to continuously monitor client requirements and provide features accordingly to fulfill their needs. The core idea of usability is any system should provide a complete solution in an effective way.

Privacy-Aware Compliance Tracking System is a software which is built using Kinect [31] sensors to monitor user activities. This system was designed initially to prevent nurse aide back injuries. In the original design of this system, there will be a Kinect server [31] to which Kinect device is connected. A smartphone is also connected to a Kinect server. When nurse aid enters into the room where our Kinect server present, the user has to press the button on their smartwatch to connect to our smartphone. The smartphone will then identify the user and inform to Kinect coordinator in our system. From that moment, our system will track user activities. Activity Recognizer will process Kinect server's skeleton stream and identify whether the activity performed is the wrong activity or not. From that moment our Kinect server will count how much duration that user was in the same posture and then it will log it for our reporting purpose. Along with user activities, this system will also log new user information, when they are connected

and when they are disconnected to our system. This system will log activity information into Amazon Web Services. Figure 1 represents system architecture [2].



Instead of finding user activities by traditional methods, tracking users using Kinect provided a lot of new information and this will be helpful to track nurse aide actions. Even though the current system can produce great insights about user activities, there are usability issues. This great software will be more helpful if we can solve these usability problems. Following will are the usability issues with current software.

- User must wear the smartwatch in order to identify them

- We need to have a smartphone which is connected to our server. This is going to sleep after some time. Because of this we were not able to track users completely and we need to track smartphone status at regular intervals of time.
- All the data is stored in XML [33] format and in MySQL [32] database, but the client interface was not available to access via the web browser.
- There was no way to check user activity status on the go i.e. there is no mobile application which can show user activity information.
- There is no way to compare different user activity information
- There are very few metrics available to analyze activity information

Even though the list is very big, we can consolidate all the list items into the following problems.

- The authentication system is interfering with nurse aide daily routine and there are maintainability problems with hardware used in the authentication system.
- The current system provides only very few types of data analytics.
- User activity analytics are not easily accessible

In the following chapters, we will discuss each problem in detail and propose our solution to each problem and compare it with the existing solution and its usability improvements.

CHAPTER II

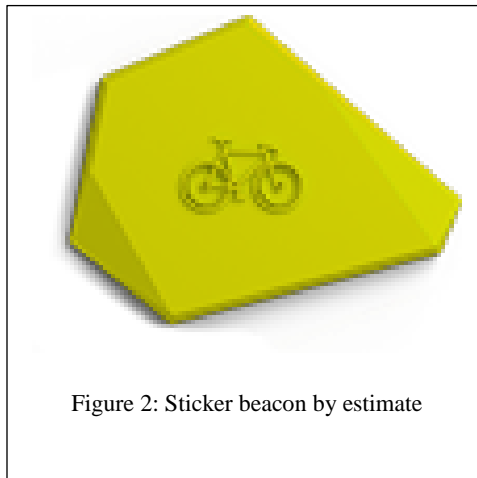
AUTOMATIC USER AUTHENTICATION WITH BLUETOOTH

In the existing system, each user has to wear a smartwatch. If any user forgot to wear a smartwatch, then our system will not recognize the user. As users will wear and remove smartwatch every day, there is a high probability that a user can forget to wear a smartwatch. Along with this, providing smartwatch to each user in a large organization will cost more. Even if we provide smartwatches to each user, it has to be detected by our smartphone placed along with the server. Most of the smartphone devices will be automatically going to sleep mode when they are idle. So, there is a high probability that we might miss user authentication when our smartphone is in sleep mode. In order to track their activities, we have to regularly monitor our smartphone and instruct all our user to wear a smartwatch every day.

2.1 OUR SOLUTION

To overcome this problem, we have to build a solution which is of low cost and it should not interfere with nurse aide daily routine. After doing some research, we found that Low Energy Bluetooth beacons [2] can fit our purpose. Bluetooth beacons are nothing but small Bluetooth devices which can transmit their signals to a very short

distance. They have very long battery life and they can work for years with a small battery. They are very tiny and very cheap when compared to smartwatches. Each Bluetooth beacon will transmit data packets which contain its ID and some other data via its Bluetooth signal. There are currently being used in museums and shopping malls for a better shopping experience. When a user is nearby, based on received Bluetooth signal [3], the smartphone may get associated notifications or web page links. We can even estimate Bluetooth beacon distance from our smartphone which Bluetooth support. Figure 2 will represent one of the short-range beacons available in the market by estimate. There are different types of beacons with different ranges and features. In most of the use cases in e-commerce, health care, and location-based services [4][5], these beacons are placed at a fixed position and when the user comes near to it, the users smart device will detect its signal.



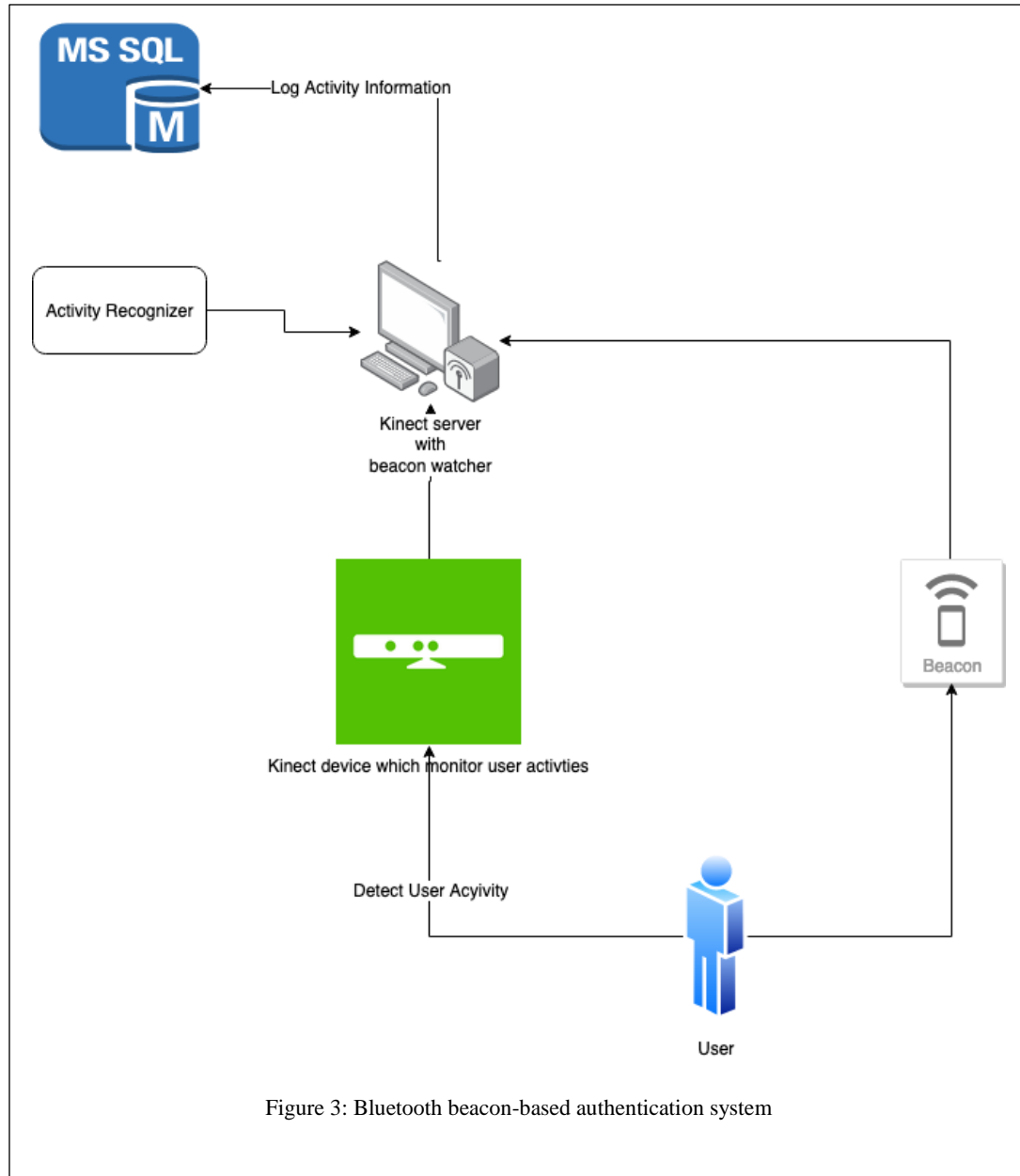
Based on the above features, Bluetooth beacons are a perfect match for our case. We will introduce a new authentication system which will use Bluetooth beacons to identify the user. For this purpose, we will select a small range of Bluetooth beacon as we will only identify users within the small room. As Bluetooth beacons are very

lightweight, we can attach them to user Identity Card. These Bluetooth beacons will be identified by our system. This solves our first problem where the user needs to alter their daily routine. With Bluetooth beacons attached to a user ID card, it is no longer a problem to the user. It will also save a lot of cost to the organization. As these beacons can run over years with a small battery, maintenance cost is also very low.

Our next problem was with the smartphone which identify the user by using beacon ID which is transmitted. As discussed earlier, these smartphones might get into sleep mode after some amount of time. Also, a smartphone is a different hardware than our Kinect server. So, it will add up more maintenance cost to the organization. Luckily Microsoft Windows 8 will support Bluetooth LE protocol and it can identify Bluetooth LE devices. As our Kinect server is also running on Windows 8 or later operating system, we can completely eliminate the need for a smartphone to detect Bluetooth beacons. While scanning for nearby beacons, we should identify beacons whose signal strength is more than threshold only. We should modify this threshold based on conditions. If the threshold is very small, users outside our room will be detected and if threshold is very high, the user in the same room who are little far from server will not be identified.

Figure 3 represents new system architecture with Bluetooth beacon detector. As discussed, user will wear ID card with Bluetooth beacon. Kinect server have Bluetooth LE advertisement watcher which will continuously scan for beacon signals. When it finds new beacon, it will identify unique ID of beacon and identify user based on that unique identifier. From this moment, our system will start tracking user activities and log them to cloud based database against identified user. If we compare Figure 1 and Figure 3, we can clearly see that we have eliminated lot of hardware components like smart watch,

smart phone, server, server task etc. which will reduce overall maintenance effort. Also, as we eliminated smart watch and smart phone components, have reduced significant amount of system cost.



Even though this system works great, there are some serious challenges we faced while implementing this new authentication system. Signal generated by beacons has occasional delays and non-uniform transmission intervals. As these beacons are made by

third-party vendors, we do not have much control over their behavior. Because of irregular intervals of signal transmission by Bluetooth beacons, it will be a problem to identify the actual user in the room. For example, if our system identifies the user who is above the signal strength threshold, there are chances that signal from the intended users is not received by the server because of its irregular signal transmission. Because of this, there are fair chances that our system might report current user activity details to some other user. To overcome this challenge, we introduced time slot-based beacon identification. In this pattern, once we detect a beacon, then our system will not identify any other user within that time slot. After given time slot, when our system receives a new signal, it will try to check the signal strength of all beacons and it will pick last detected beacon if it is present in the received beacon list (We will ignore any beacons with more signal strength). This helps us to solve irregular interval signal transmission problem. This time duration tracking will start when the new beacon is detected by the system and stop tracking for beacons when it lost its signal. We should carefully decide time slot duration until which our system wait to track next beacon signal. If this window is too large and users switched in between this time slot, then our system will report the wrong data. It would be a good idea to put this duration to some minutes, but we need to adjust this duration based on practical use cases.

2.2 DEVELOPMENT

Windows SDK has support to detect Bluetooth LE devices. Windows provide *BluetoothLEAdvertisementWatcher* class. It will provide *Start* method to start scan for Bluetooth devices. This class accepts *Received* event listener. Delegate which we provided to *Received* will be called when new beacon is detected. In *Received* delegate,

we can iterate over all nearby Bluetooth beacons and get meta data related to respective beacon. This meta data includes unique identifier, signal strength etc.

As discussed till now, Bluetooth beacon authentication is very low cost and easily usable alternative to existing smartwatch-based authentication. As technology is progressing at a very faster rate, we are hoping that we can overcome above-mentioned challenges in the near future without compromising our functionality.

CHAPTER III

CLOUD-BASED DATA AGGREGATION

Privacy-Aware Compliance Tracking System will track all user activities and stores in the cloud database. In the existing system, .Net based GUI was provided to visualize data. Using this visualization system, we can visualize data related to one or all users in a specific date range. Even though this system works well, there are many usability problems with this system.

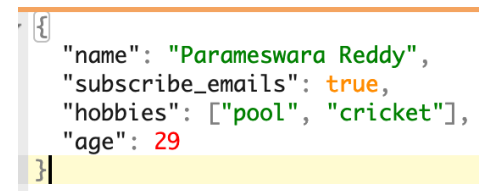
In the current system, data is directly inserted into MySQL database and processed data is exposed via SQL stored procedures. Stored procedures are very good when it comes to performance but in order to access this data, client need to have SQL driver support. For example, if we want to build an interface which cannot connect to the SQL database directly, then it is not possible to visualize data. To support multiple clients, we need to build a data processing component which can be connected from all types of clients and provide data in a format which can be understood by all clients. If we can build such a system, we can build lightweight clients which will only render data received from the data processing component. This leads to more maintainable system.

3.1 OUR SOLUTION

To reuse data processing logic, we will expose our data via web service. A web service is a piece of software which exposes itself via the internet. Web services accept input data and produce output. There are two types of web services [11].

1. SOAP web services: It is XML based protocol for accessing web services. It is independent of platform and language
2. RESTful web services: This is not a protocol, it is just an architectural style. It will support many types of request and response data types like text, HTML, XML, JSON, etc. This is very flexible and is accessed like any other URL.
RESTful web services are simple to consume.

Because of the simplicity and interoperability of RESTful web services we are going to use this as our data exchange medium. As RESTful web services support a variety of data types, we have the option to choose the best data type which suits our purpose. As we want to transfer structured data and we will transfer large amounts of data, it is very important to choose data type which consumes less bandwidth for the same amount of data. The JSON format is a light-weight format which can hold a large amount of data. It is very easy to consume JSON in most of the programming languages. JSON is very easy to write and read. It supports a lot of data types like arrays, numbers, text, etc. Figure 4 will show sample JSON.



```
{  
  "name": "Parameswara Reddy",  
  "subscribe_emails": true,  
  "hobbies": ["pool", "cricket"],  
  "age": 29  
}
```

Figure 4: Sample JSON

3.2 SYSTEM REQUIREMENTS

We need to build a system which accepts activity data as input. It should save input data to appropriate database tables. When the user request data analytics, it should provide data analysis on up to date data. As we are building privacy aware software, we should allow only authorized users to view user's data. One user should be able to access other user data only if he has appropriate authorization. Our system should allow user management. The system should expose processed data via RESTful web services (we will call these web services as Rest API) using JSON format. Each web service should have access restriction. The system should support data logging and data analysis.

3.3 SYSTEM DESIGN

After careful analysis of existing database and requirements, we came up with the following entities for which we need to support create, update, delete and fetch operations.

- User: User is the person who is being tracked by our Kinect sensor or the person who want to access our data analytics. Based on this we defined two types of users
 - Administrator: User who can view data analytics of all users and manage all user information. User with this role will have all permission in the system.
 - Employee: Employee is the person who is being tracked by our sensors. He can only view his own data and he should be able to update his own information in the system.

- Session: When our Kinect server first identifies user, then a session will start and when the user disconnects from our system, then we will end our session. User activities will be tracked by session. User can have multiple session.
- Activity: This represents the intended activity performed by the user. There can be multiple activities in session. Each activity will be performed for some duration. Duration of these activities and the number of times a user performs this kind of activity is our point of interest. We will analyze it in different dimensions over time.

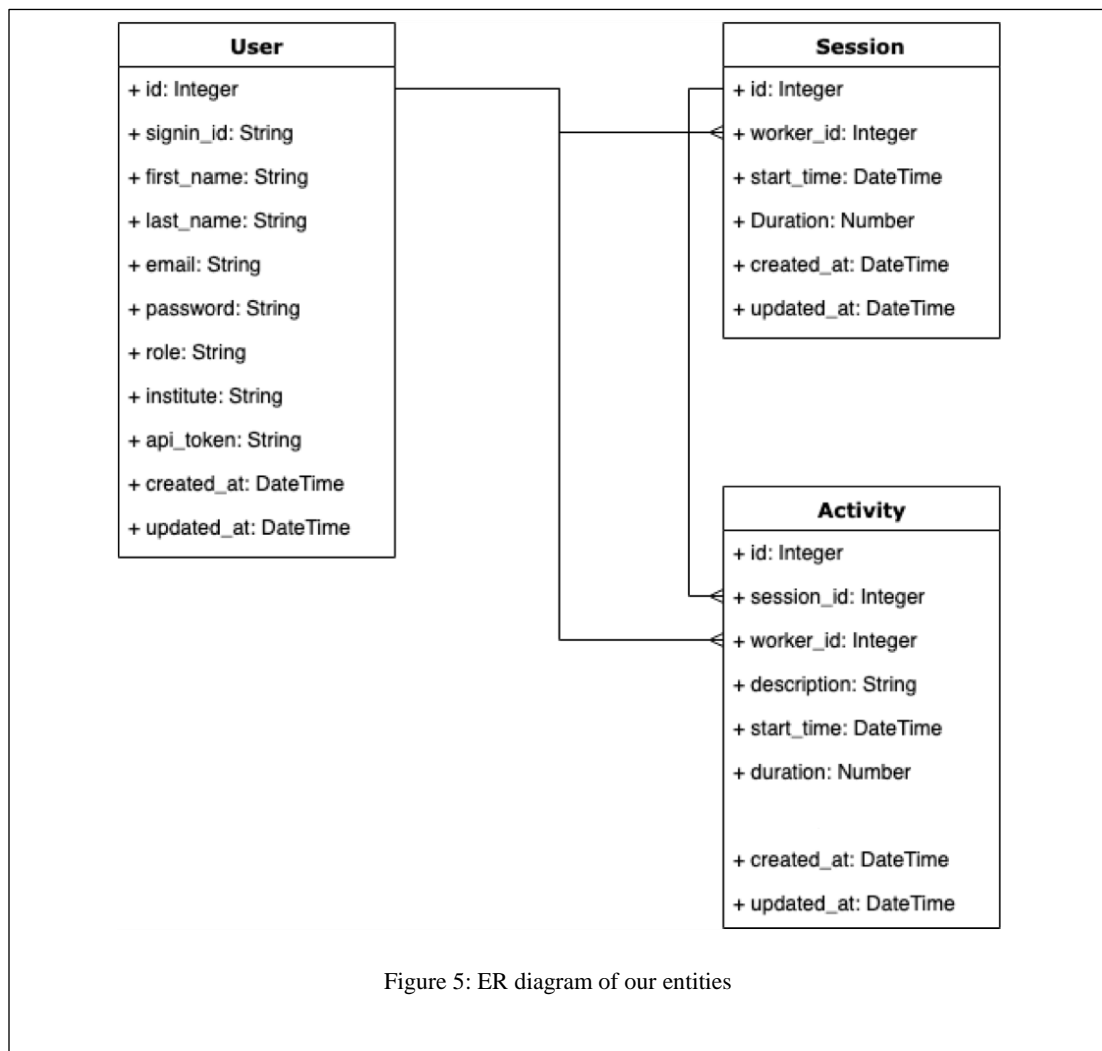


Figure 5 represents our entity relationships.

We need to expose operations via Rest API. These are very important to our system because these APIs will support logging our data analytics into the database. Along with this, we will expose our data analysis as a web service. Each web service will be represented by a URL. We can call these URLs from our clients and pass input data JSON. Our server will process input data and send the response as JSON. Our clients can parse returned response data and display it to end user. Figure 6 will represent system architecture.

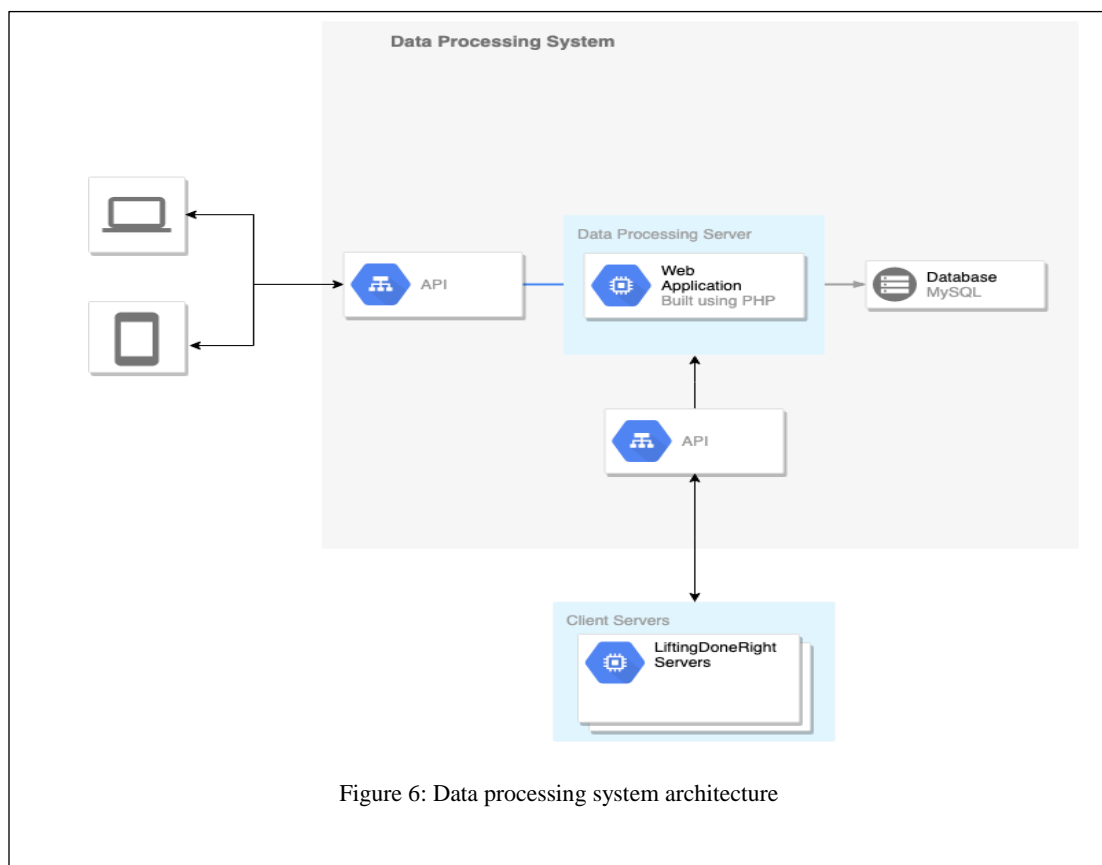


Figure 6: Data processing system architecture

3.4 DATABASE DESIGN

We already discussed that we have 3 entities in our application i.e. user, session, activity.

User Table: Table 1 shows user table schema.

Field	Type	Length
id	INT	10
signin_id	VARCHAR	255
first_name	VARCHAR	255
last_name	VARCHAR	255
email	VARCHAR	255
password	VARCHAR	255
role	VARCHAR	255
institute	VARCHAR	255
api_token	VARCHAR	255
created_at	TIMESTAMP	
updated_at	TIMESTAMP	

Table 1: User table schema

Session Table: Table 2 displays the session table schema.

Field	Type	Length
id	INT	10
worker_id	INT	10
start_time	DATETIME	
duration	DECIMAL	12,2
created_at	TIMESTAMP	
updated_at	TIMESTAMP	

Table 2: Session table schema

Activity Table: Table 3 shows the activity table schema. We are duplicating worker_id in this table to improve the performance of data analytics.

Field	Type	Length
id	INT	10
session_id	INT	10
worker_id	INT	10
description	VARCHAR	255
start_time	DATETIME	
duration	DECIMAL	12, 2
created_at	TIMESTAMP	
updated_at	TIMESTAMP	

Table 3: Activity table schema

3.5 WEB DEVELOPMENT

We are building an application using PHP. PHP is very popular and very easy to develop and deploy web applications. PHP can connect to our MySQL server to fetch data. Instead of directly using PHP to develop an application, we used a framework which is built on top of PHP. The framework which we used to develop our application is Lumen [12]. This is a very lightweight framework which supports easy development and deployment of Rest API using PHP. It will provide support for user authentication and authorization for each API. It also provides an easy way to access data in our database to manage them in the object pattern. For basic operations, we can almost eliminate writing SQL queries. This results in a more understandable and maintainable code. In Lumen, we call each *Entity* as *Model*. This is provided by Eloquent [13] framework. In Eloquent

framework, we can define model by extending *Model* class provided by framework. This class accepts *\$table* variable which hold table name related to our model. Our model will fetch data from the mentioned table when we request data. In this model, we can define another array *\$fillable* which holds array of field names which can be directly mapped from input request. This will be convenient when we want to map large amount of data attributes from request to model. This class also accepts *\$hidden* field which hold array of fields not to be exposed to end user. Usually we do not want to send information like password as response for every user. This is because password is a sensitive data. We can find all properties in its documentation.

Model class expose a lot of utility method on our class to save, update, delete and fetch record of this type. For each *model*, we define a table. In our table, the framework will automatically add two extra columns named *created_at* and *updated_at*. This will help us to track when each record was created or updated. Lumen also provides an interface to execute raw database queries in SQL format. This helps us to run complex SQL queries on the database. This feature is mainly helpful in data analysis.

To expose operations on models as API, we need to create a controller [14] which intern will be exposed via route [15]. While exposing our controller methods as routes, we have to map each route to HTTP request method. The following are the conventions used while developing RestAPI.

- **GET** request type to fetch any data from the server
- **POST** request type to insert any data into the server database like activity, session, etc.
- **PUT** request type to update any existing record like user basic details

- **DELETE** request type to delete any record from our system

Along with these basic controls, we will provide API to login into our system and maintain their session. But Rest API is based on HTTP protocol which does not support session between multiple requests. For this reason, we came up with the following mechanism.

When a user is trying to login into the system, we will check for the validity of email and password. If email or password is wrong, then we will return HTTP 401 error code which indicates that authentication failed. If email and password matches, then we will generate a unique token and save that against the matched profile. We will return this token to the user as a response. User has to send this token in "Authorization" header in an HTTP request when he is requesting access restricted data. This way we can easily identify the user.

With the above features, we provided a mechanism to save activity logs using API and share data analytics to multiple clients by providing appropriate security. Our new design is flexible and extensible. We can build multiple clients on different platforms without worrying about data processing logic. This improves the reachability of our application and this enables us to expose our data to multiple platforms very fast.

CHAPTER IV

WEB INTERFACE FOR DATA VISUALIZATION AND ANALYSIS

In the current system, data is available on the server with .Net client. This limits the usability of our system. If someone want to view data analytics, they have to come to the server and see the data analytics. Also, it does not provide any security which prevents a user from viewing unauthorized data. To monitor their daily activates, they need to take some time out of their busy schedule to go to the server. This will hurt their daily schedule

4.1 OUR SOLUTION

We will build a web interface which supports most of the browsers. By exposing our data analytics via the web interface, users can stay at their own place and check their data. We can even restrict the user from viewing other user's data by role. Along with this, we can also provide user management where an administrator can create or update users.

4.2 SYSTEM REQUIREMENTS

We need a web interface for our users to view their own or their employee's real-time data analytics in a web browser. Our application should support all major browsers.

Following are the required features our application should support.

- User should be able to login into the system using their email and password.
- The administrator should be able to view all users in the system.
- The administrator should be able to update all user related information.
- The administrator should have the ability to change the password of any user.
- Users should be able to update their own profile.
- Each user should only be able to view his own data over a period of time.
- User should be able to view average wrong activity duration by each day.
- User should be able to view average wrong activity duration by each weekday.
- User should be able to view average wrong activity duration by each month.
- User should be able to view average wrong activity duration by each hour in a day.
- User should be able to view the total number of wrong activities performed by each day.
- User should be able to view the total number of wrong activities performed by each weekday.
- User should be able to view the total number of wrong activities performed by each month.
- User should be able to view the total number of wrong activities performed by each hour in a day.

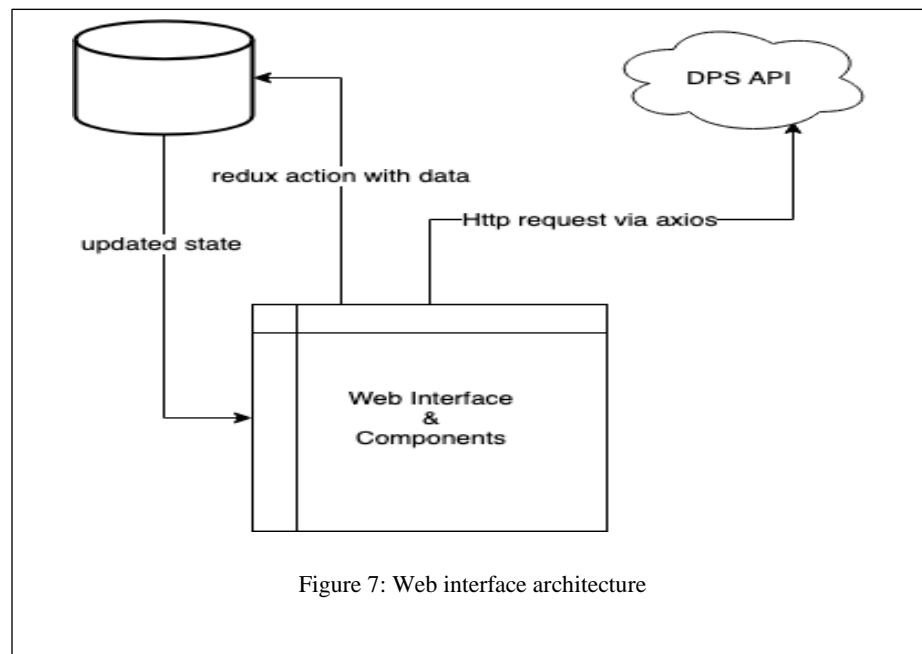
- User should be able to view data as a Bar chart which helps to compare data by a specific date.
- User should be able to view data as Line Chart which helps to compare data over a period of time.

4.3 SYSTEM DESIGN

After careful analysis of system requirements, we should have the following components in our system.

- **Data access service** is needed to send or receive data from the Data Processing System API.
- **Data store service** is needed to save global information for our application and share the data to all pages in our system.
- **Web Interface** to display data to end user.


Figure 7 shows the system architecture.



4.4 USER STORIES

After careful analysis of system requirements, we came up with the following user stories

- Create a screen to provide login functionality to end user. This screen has to take email, password and validate their correctness. After a successful login, the user should be redirected to the Dashboard Page. The screen should look like Figure 8.
- The dashboard should be able to display user analytics based on the start date, end date, analytic type, and chart type. The screen should look like Figure 9
- Create a page to display all users. This should be available only to administrators
- Create a navigation link to access user profile from the menu
- Create an option to log out of the system
- Create a Line Chart to display analytic type
- Create a Bar chart to display analytic type
- Create a service to access data form RestAPI provided by Data Processing System.



The diagram shows a login screen with a title "Login" centered at the top. Below the title, there are two input fields. The first field is labeled "Email" and the second field is labeled "Password". Both labels are positioned to the left of their respective input boxes. The entire login form is enclosed in a rectangular border.

Figure 8: Login screen

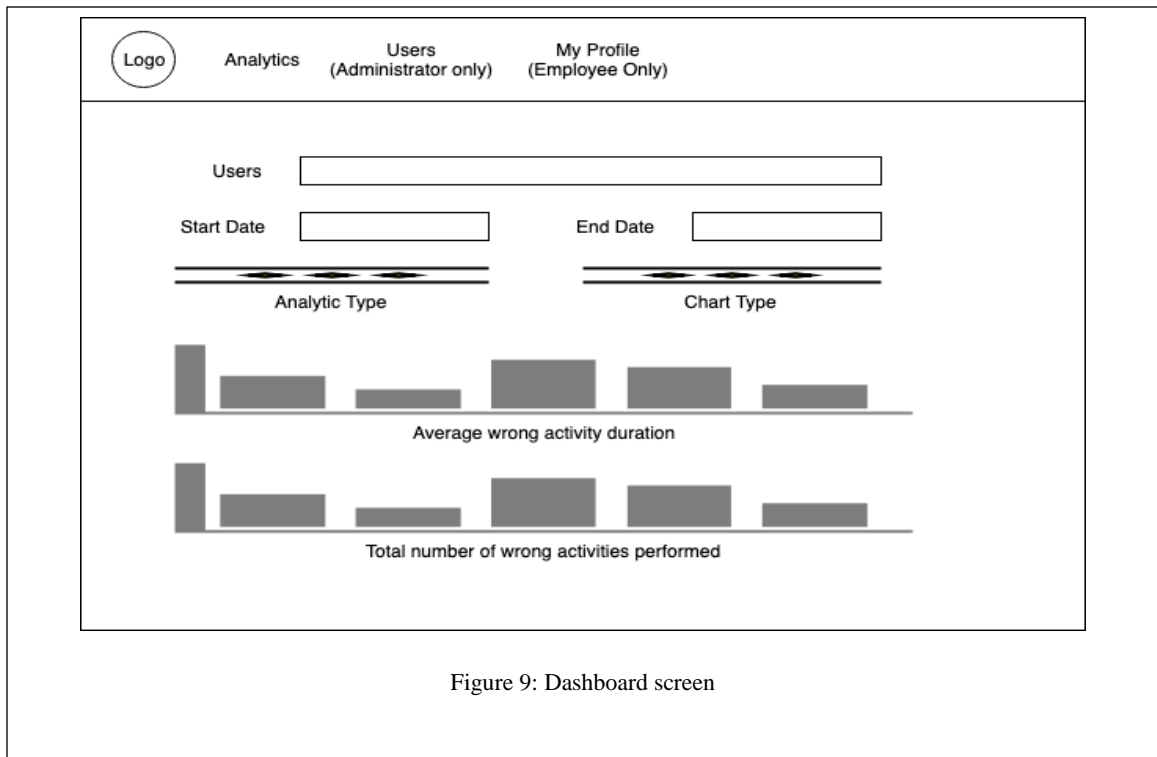


Figure 9: Dashboard screen

4.5 DEVELOPMENT

We need application which run on web browser. Web browser-based applications are built using HTML5, CSS3 and JavaScript. As we have separate data provider, we will build rich client interface. In today's world, we have lot of browsers provided by different operating systems and different vendors. For example, we have google chrome, Microsoft Internet Explorer, Apple Safari etc. As we want to serve large user base, it is very important to support all major browsers. All these browsers almost support most of the common features, but they implement these features in different way. To overcome this, there are lot of wrapper frameworks which will provide uniform way to support multiple browsers. To provide cross-browser compatible we are using CSS [26] Semantic UI CSS framework [20]. We can write our HTML [25] application using plain HTML and

provide interaction to these HTML elements by using Java Script [24]. But to build application using plain HTML and JavaScript will take lot of time. Also, if we want to navigate between multiple screen, we have to reload every page. To eliminate all these problems, we are using ReactJS [16]. This library was developed by Facebook. ReactJS supports component-based architecture. We will create multiple reusable components and use them in appropriate places in our user interface. Using ReactJS, we can also build rich client interfaces which support Single Page Application (SPA) pattern. In this pattern, we will not reload entire web page for each screen. We will only reload necessary areas for each screen by maintaining state of common areas of screen. To create react application, we are using *create-react-app* [30] tool which is provided by Facebook. This will generate application skeleton. To communicate with react we are using Ajax call supported by all browsers. To perform Ajax calls, we are using Axios [18] library which will have request based on promise based architecture. Our main interest of the application is to display user activity analytics inform of charts. So, we have to select good charting library which support lot of charts and easily configurable. This library should support ReactJS based component architecture. After some effort, we found Nivo [19] charting library which support lot of charts and easily configurable. With above base architecture, we have to build following modules.

- Login Screen
- Dashboard screen
- User management
 - List all users
 - Update user profile

- Change password of the user

In the following sections, we will discuss each module in detail.

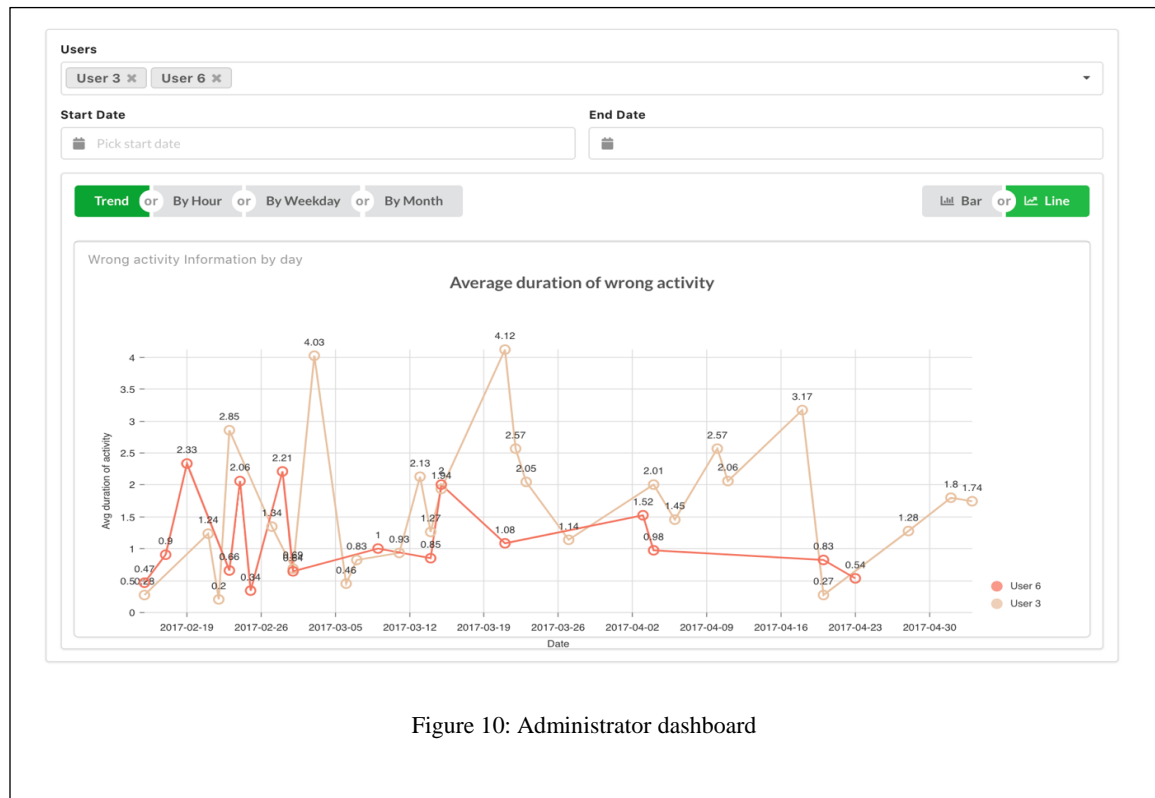


Figure 10: Administrator dashboard

4.5.1 Login Screen

We are creating this screen to allow the user to login into our system. Once the user provides a valid email and password, we will call our API and get api_token for this user. Once the user is able to get api_token, then we will be getting logged in user information using token and redirect to Dashboard.

4.5.2 Dashboard Screen

We are creating this screen to allow users to view data analytics. In this screen, we will allow user to pick the required users. This is only allowed for administrators but for employees, this will be read-only. After user selection, we will allow the user to select a time frame in which he/she interested in. Then we have the option to pick the type of

data analytics he/she interested in. Based on this automatically chart will be populated.

We are proving two different types of charts. A bar chart will allow seeing exact data on each unit and line data will show an overall trend over time.

4.5.3 Users Management

This module is only available to administrators. In this module, we will show all the users in a table. Administrators can edit the user profile and set a password for any user. This will enable the user to login into the system. Administrators can create other administrators or employees.

With this interface, we provided easy access to our data. This will save a lot of time and can help users to extract a lot of information from data.

CHAPTER V

MOBILE APP FOR DATA VISUALIZATION AND ANALYSIS

In CHAPTER 4, we provided an interface to view data analytics by using web browser. The web interface is very good when you have a computer near us. Going to computer to view activity information also takes some time out of the user's schedule. As we already discussed, we try not to disturb the user's schedule.

5.1 OUR SOLUTION

In today's world, the user of smartphones increased very rapidly. If an application provides a native mobile interface, it will reach number of customers as it provides easy access to our features and user can use our application on the go. This is very important for our application as we do not want to interrupt user daily routes for any of our tasks. We can send notifications to the user about their activities. This increases usability and improves our application effectiveness. In the smartphone market, a major section of people is using smartphones based on Android operating system. Android smartphones are less costly, and any developer can publish applications with a very minimal fee. Because of these reasons we started building a mobile application for Android.

5.2 SYSTEM REQUIREMENTS

To develop any application gathering system requirements is very important. In our case, we wanted to expose employee related functionalities via mobile application. So, we are not providing a user administration feature in a mobile application. Also, mobile screen width is very small, it is not a very good idea to display large data using charts.

Based on the in-depth analysis we started with the following system requirements

- Mobile application should support Android 5.1 Operating System as this will support approximately 80% of Android devices in the current market.
- The application should display data to only authorized users
- The employee should be able to see only his own data
- The administrator should be able to view any user data.
- Both employee and administrator should be able to view data between specified dates
- Both the administrator and employee should be able to see the following data analytics
 - Wrong activity duration trend by the user over time
 - Wrong activity duration trend by weekday
 - Wrong activity duration trend by an hour of the day (12 AM, 1 AM, 2 AM, ... 11 PM)
 - Wrong activity duration trend by the month of the year (January 2017, February 2017, etc.)
 - Number of wrong activities performed in the day by user

- Number of wrong activities performed in weekday by the user (Sunday, Monday, Saturday, etc.)
- Number of wrong activities performed an hour of the day (12 AM, 1 AM, 2 AM, ... 11 PM)
- Number of wrong activities performed in the month of the year (January 2017, February 2017, etc.)
- User should be able to log out of the application after use.
- All the information should be up to date with Data Processing Server (DPS)
 - All analytics are presented using a Line Chart

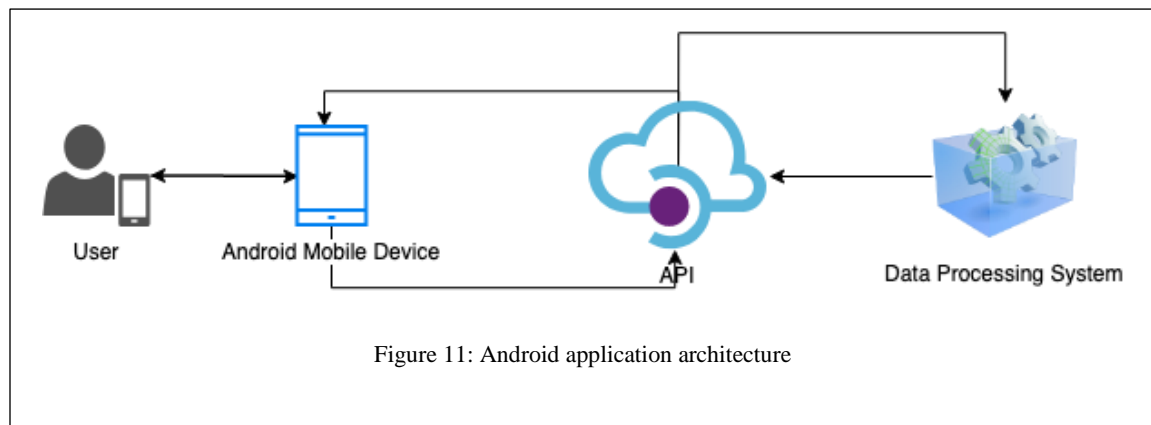
We will not support the Bar Chart because of available screen width on mobile.

5.3 SYSTEM DESIGN

After careful analysis of system requirements, the following components are required to build our Android [28] [29] client

- **API Connector Service** is middleware which will allow our mobile application to connect to API provided by Data Processing System (DPS)
- **Android Application** client which is used by the user to access our data analytics

Figure 11 represents the mobile application architecture of our android application.



5.4 USER STORIES

After analyzing system requirements carefully, we need to work on following user stories

- We need to provide a service to connect to our API provided by Data Processing system. This should automatically convert data received from API into respective beans so that they can be easily consumed
- User should be able to login into our system. Proper validation messages should be displayed to end user
 - We have to take email as input from the user. This is mandatory. The email should contain the '@' symbol
 - We have to take the password as input from the user. This is mandatory
 - User should be able to login by clicking on 'SIGN IN' button
 - Proper messages should be displayed to end user
- User should be able to view his data analytics. Following should be taken care while displaying data analytics by analytics type.
 - Administrators should be able to view data for all users
 - While selecting users, the user should be able to search users by first name or last name
 - The employee should be able to view only his data
 - User should be able to select the start date and end date for data
 - User should be able to select data analytic type

5.5 ENVIRONMENT SETUP

To develop Android mobile application [28] [29], Google provides an Integrated Development Environment called "Android Studio" [34]. This provides rapid application development. To maintain source code versions, we are using git as a version control system. To access our API, we are using retrofit library along with the gson library to convert the response to Java models. Android by default does not provide any multi-select. To support multiple user selection, we are using *MaterialChipsInput* [23].

As our application main aim is to display data analytics as charts, we should select a library which support multiple charts and it should have good community support. By considering all aspects we are using *MPAndroidChart* [21] library. The best feature of this library is it will provide pinch zooming capability so that the user can view more dense data in chart very clearly. This library provides scaling, dragging and animation features

5.6 DEVELOPMENT

We create a project using Android Studio with minimum API level as "API 22: Android 5.1 (Lollipop)". As discussed in design, we are providing two screens in a mobile application.

- Login Screen
- Dashboard Screen

In Android application, each screen is called as "Activity". Each "Activity" contains one layout file and a java file to support its operations. Along with these screens, we need to create a service to connect to our API. This can be done using Retrofit API [22]. This

should automatically add API Token to request headers to access protected routes defined by our Data Processing System.

To access our data securely, we need a service which can automatically fetch and map data to Java models. For this purpose, we need to build an API service. We will create API service using retrofit instance. We will expose this client via a singleton design pattern where we expose one object throughout our application instead of creating an instance for each call. We will define our API Base URL here. We can switch our API from one place to other places by just modifying *BASE_URL* variable at any point of time. To access protected routes we need *api_token* to authenticate. To include *api_token* to request "Authorization" header, we need to add request interceptor. In below class, we will create a class which implements *Interceptor* and override *intercept* method. After our request modification is done, then we will proceed with the next action for request.

5.6.1 Login Screen

This is the main screen of our application. When user launches our application, we will show our login screen. In this screen will take the user's email and password as input. We will validate user email and verify whether email contains '@' character or not. If email is correct, then will check whether user provided password or not. If email and password pass preliminary checks, then we will call our API to validate email and password combination. If they didn't match, then we will show appropriate error message to user on respective input field. If email and password combination match to some user, we will get API token as response from server. Once login is successful, we will fetch logged-in user details and navigate the user to the dashboard screen. Figure 12 shows our mobile interface login screen.

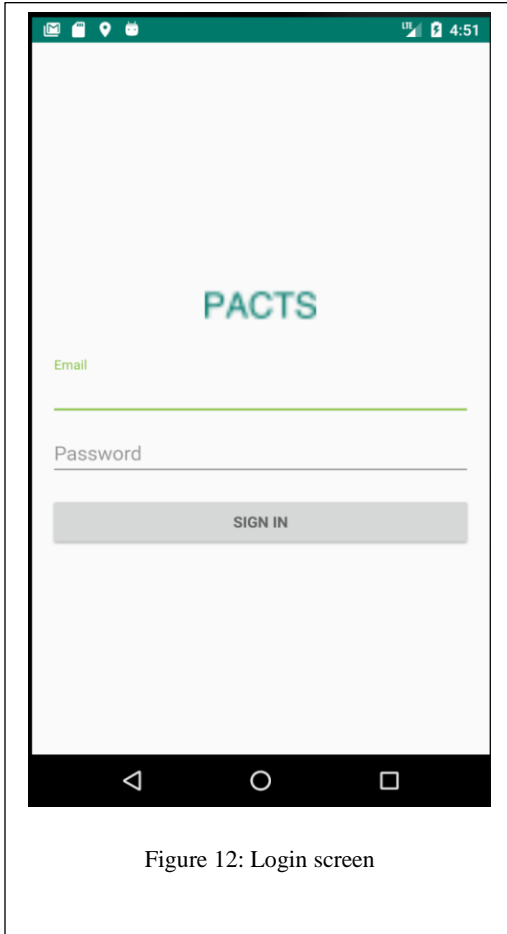


Figure 12: Login screen

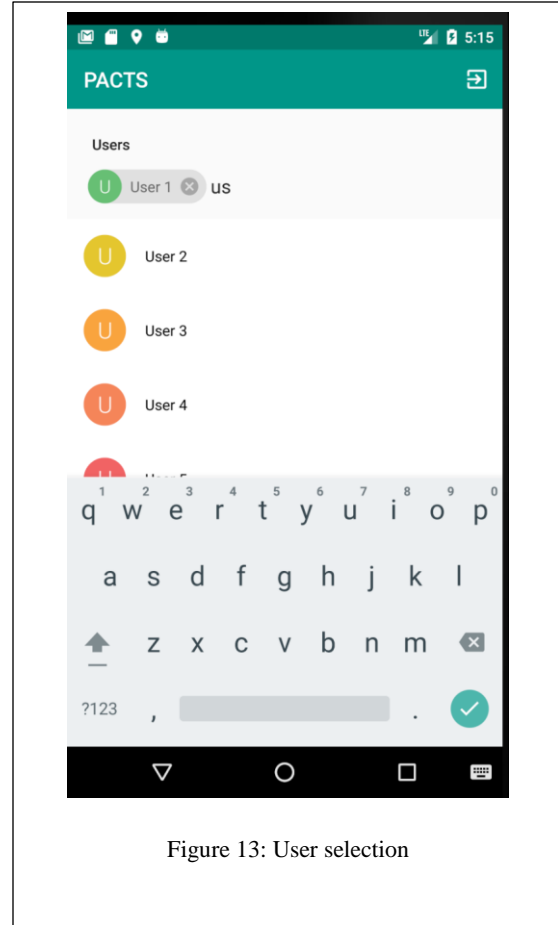


Figure 13: User selection

5.6.2 Dashboard Screen

This will look similar to our web interface. In this screen, only an administrator can select multiple users to compare their analytics. Multiple user selection screens will look like Figure 13.

After this user can select the time frame. When the user wants to select the date, we are providing calendar dialog for them to pick a date. This interface will take less time when compared to the traditional way of entering date as text. This is shown in Figure 14.

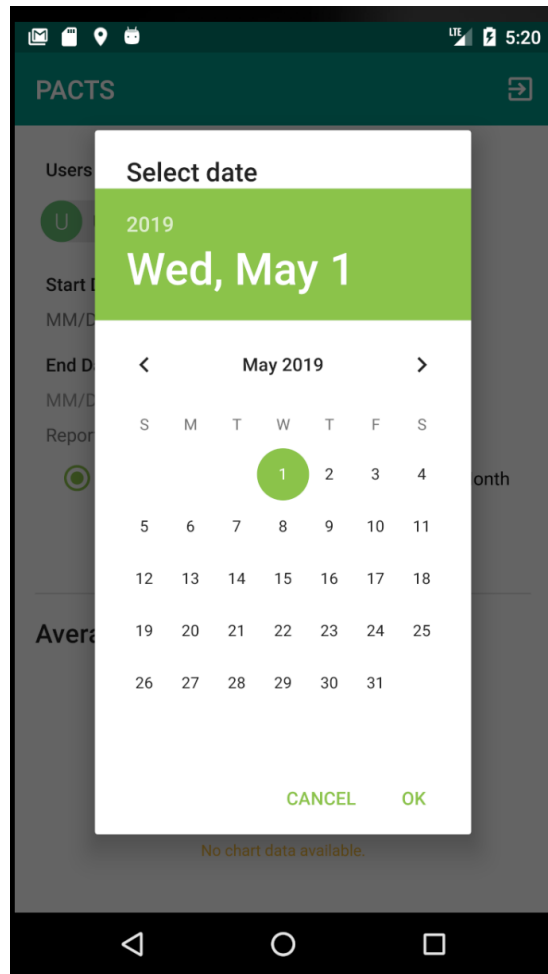


Figure 14: Date selection control

When a user submits a request to the server, the android application will fetch information from API and render as Line Chart. We can zoom our Line Chart for a better view. As we already discussed, we are not supporting the Bar Chart on the mobile screen because of space constraint. Our input selection screen will look like Figure 15.

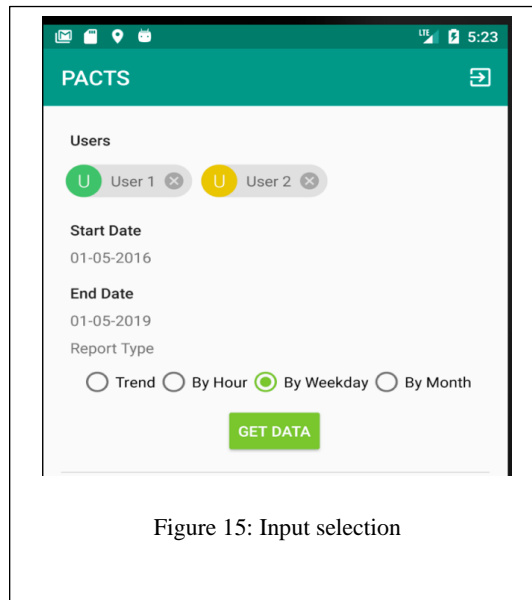


Figure 15: Input selection

Once the user selects input data, they can access data by clicking on "GET DATA" button. This will generate line charts with each color representing different users. Figure 16 will show sample screen of the generated chart.

As our application is very simple and easy to use, it will save a lot of time for users when they are reviewing their analytics. They access their data on the go. Because of this, employee can view their data while travelling back to home or when they have time.

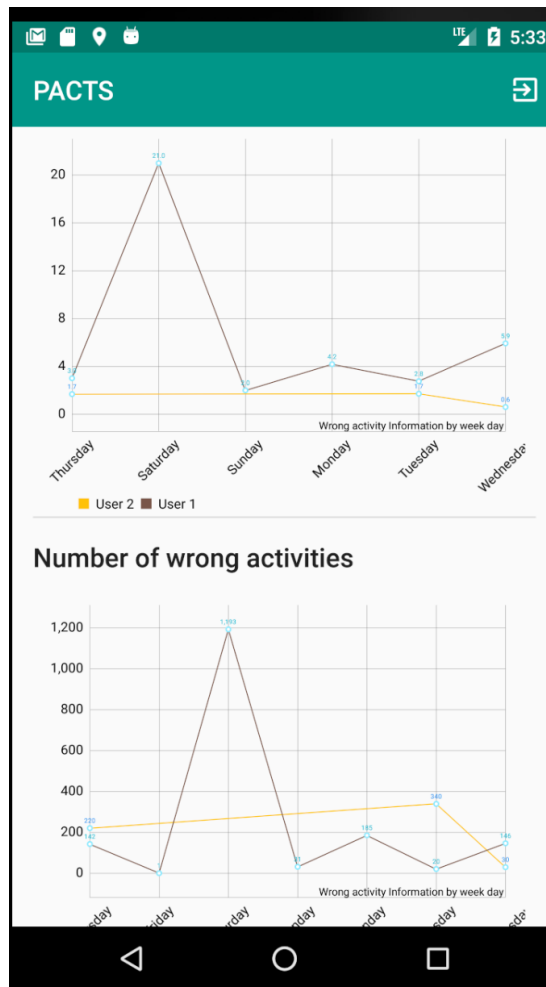
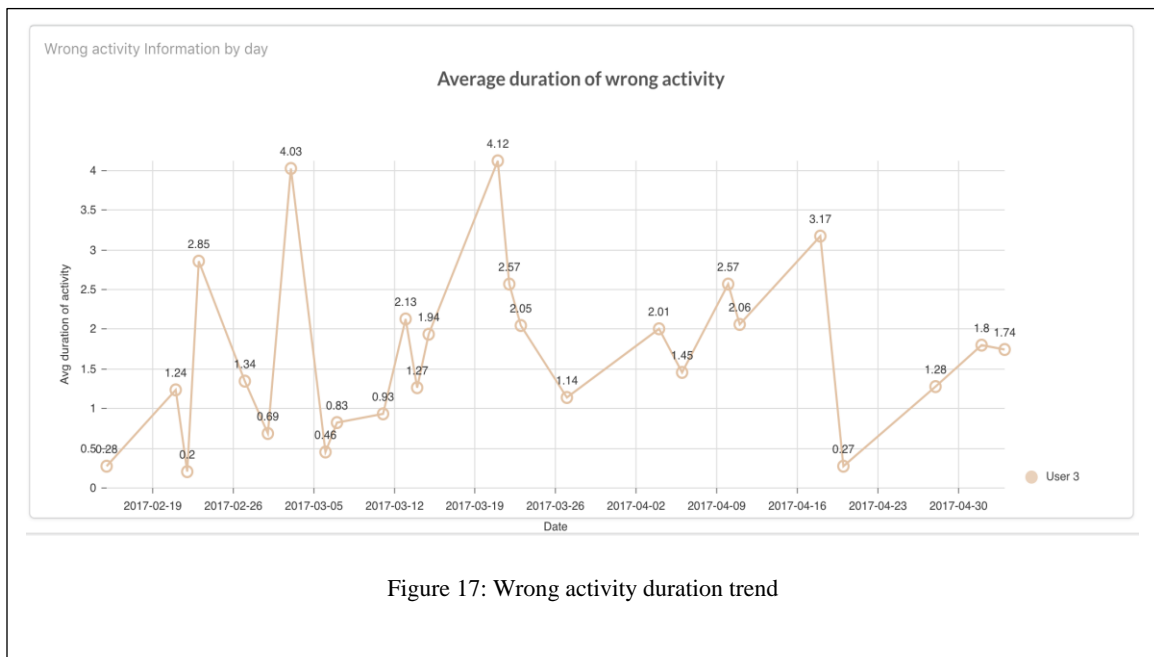


Figure 16: Generate chart

CHAPTER VI

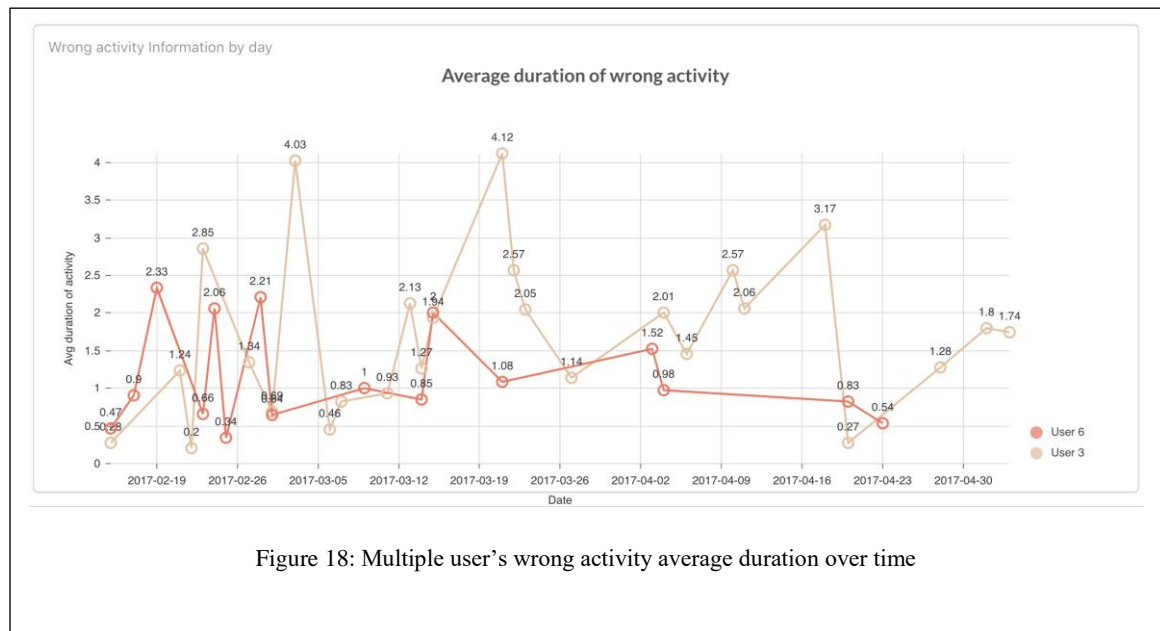
ANALYSIS OF DATA USING NEW DATA VISUALIZATION SYSTEM

Till now, we just build a system to produce analytics and multiple clients on different platforms to display them to end users. We unleash the real power of this system when we are able to get useful information from the analytics provided by the system. As discussed earlier, we are providing multiple types of analytics. We choose to provide these types of data analytics because they can provide insights about data in different dimensions.



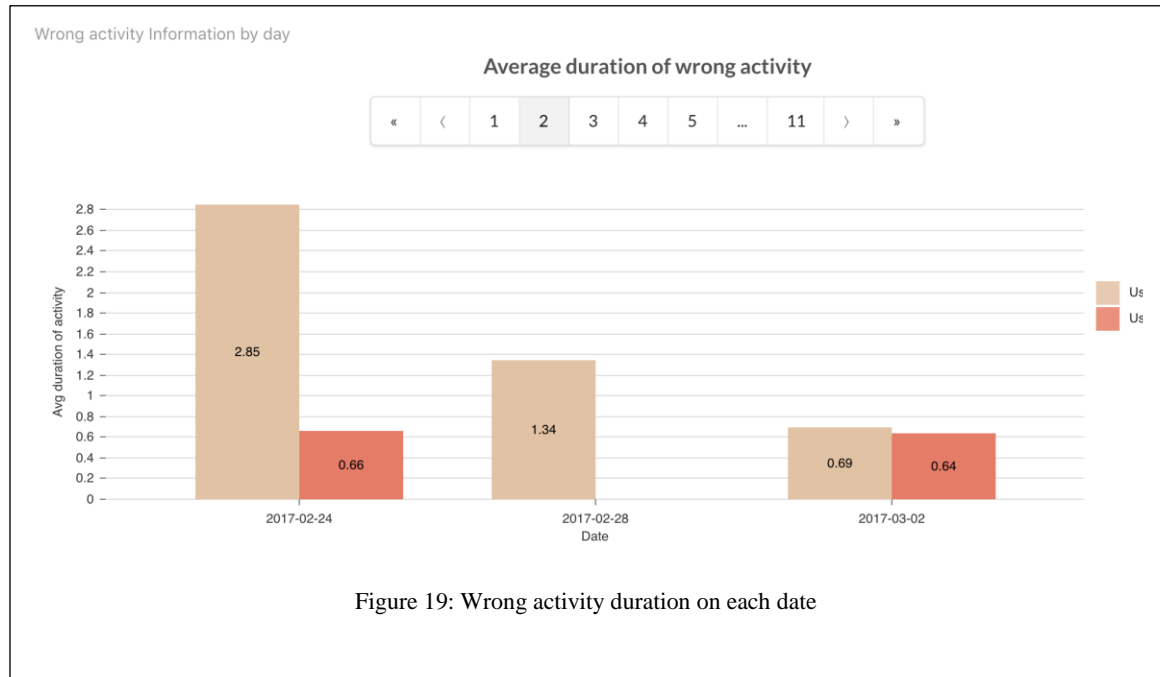
LiftingDoneRight system collected data about different nursing aids for over 3 months. We loaded this data into our data processing system to test our system analytic capability and usability. With our system, we are able to see single person wrong activity duration trend over time which helps the user to review/correct his/her activities in the future. Figure 17 show user wrong activity duration over time.

When we view single user data over time, it will just provides insights about his own data. But we will get actual results and insights when we are able to compare different user analytics over time. Figure 18 shows multiple user's data over time.



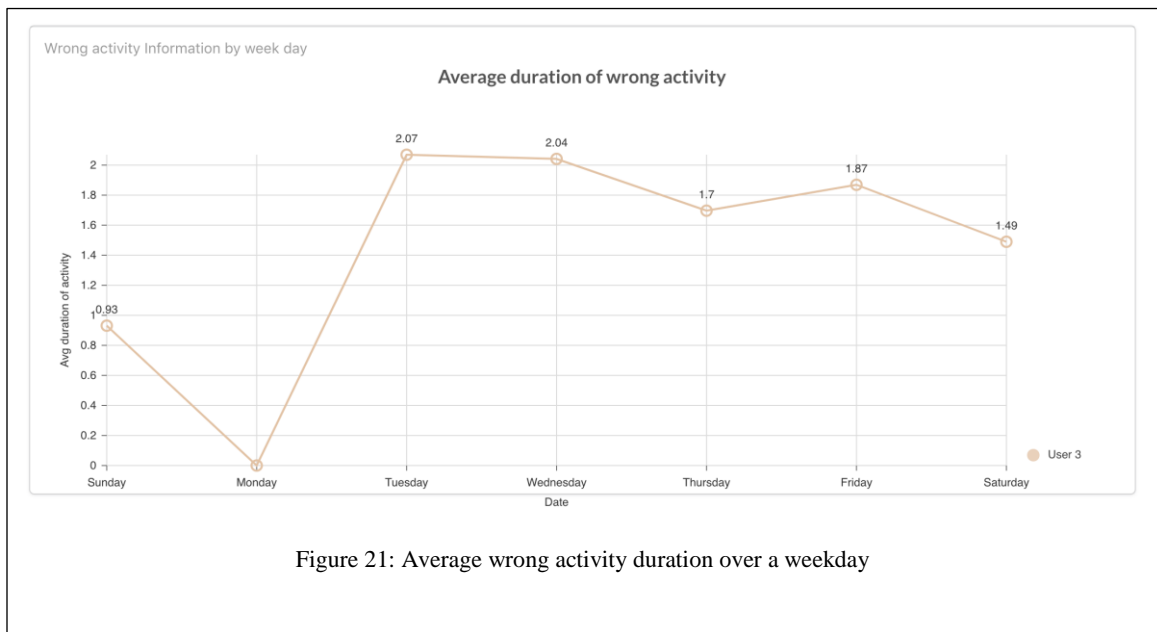
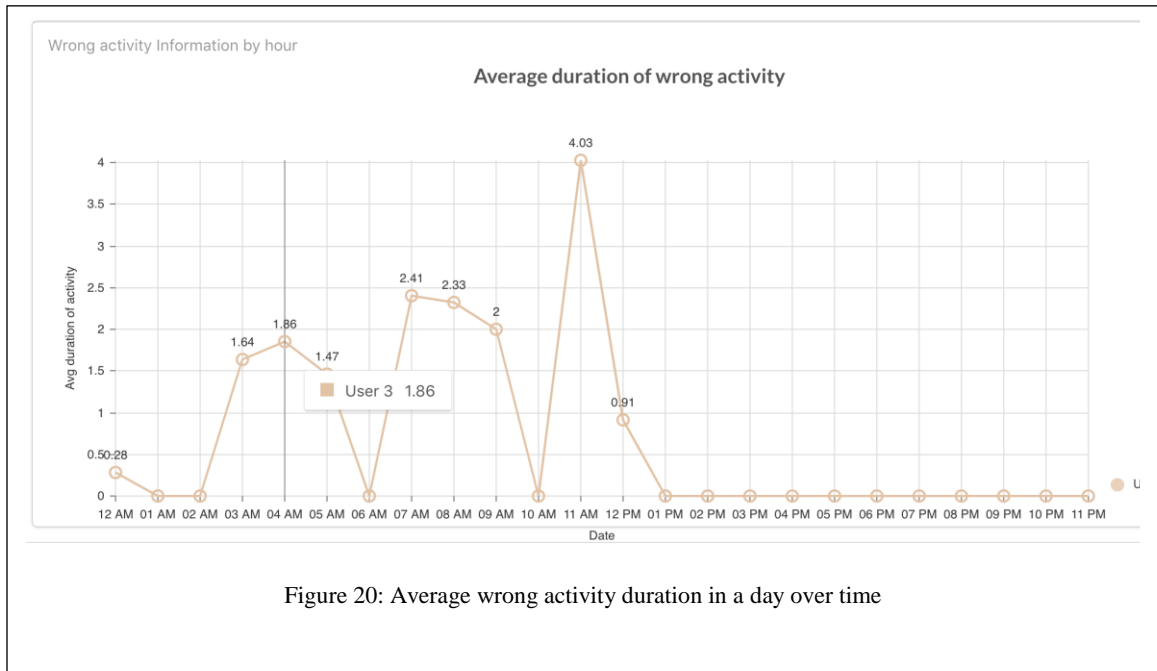
From Figure 18, we can see that User 3 wrong activity duration is relatively more when compared to User 6. With this, we can inform User 3 that he/she should take more care while serving patients. If we want to compare the wrong activity duration by each day, we need a different type of analytics. Our Bar chart represented in Figure 19.

We can see from Figure 19, User 3 has performed the wrong activity for more duration on 2017-02-24. And on 2017-03-02, they almost performed equal duration. This helps us in comparing different user's data at the exact point of time.



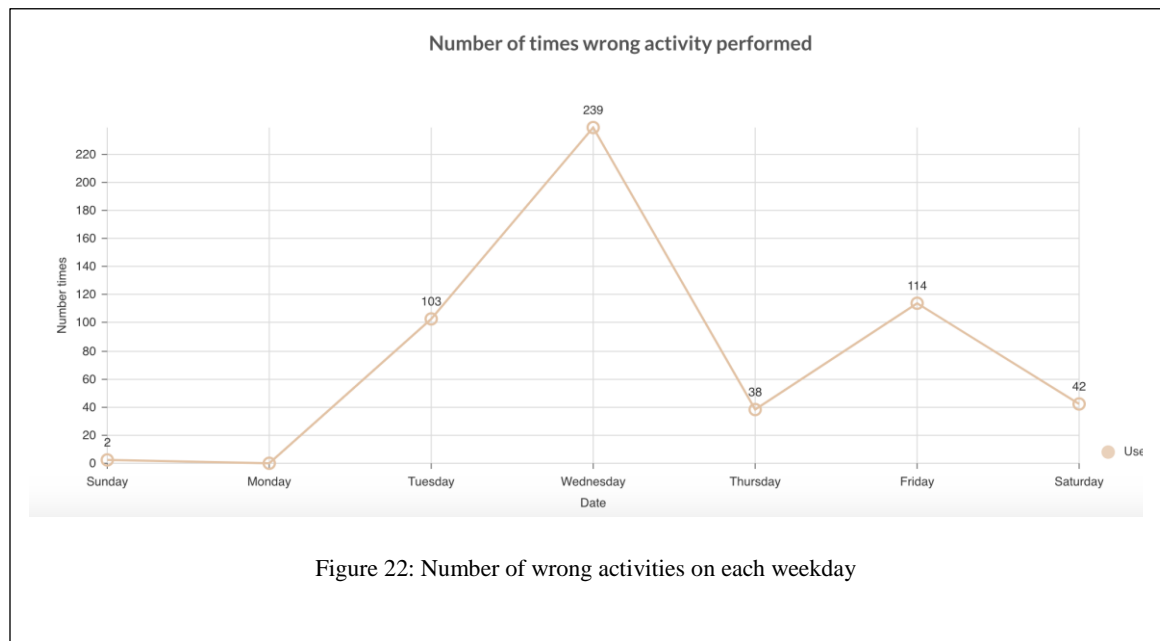
Above analytic is informative but it would be great we can extract more insights from data. Wrong activities might not be distributed equally over time. For any human, energy and enthusiasm will change over time in a day or based on a weekday. User energy and enthusiasm might be varying over time. The user might be very strong in the morning and slowly energy might come down by evening. To find out this, we are providing data analysis over time in a day.

Figure 20 will show User 3 wrong activity duration over time in a day. User 3 is doing more wrong activities at 11 AM. So, we can analyze what is going wrong with User 3 at 11 AM and provide additional assistance at that point of time.



We are further interested in wrong activity duration by weekday. We want to know where our users are having more wrong activity duration. Figure 21 will show average wrong activity duration over a weekday.

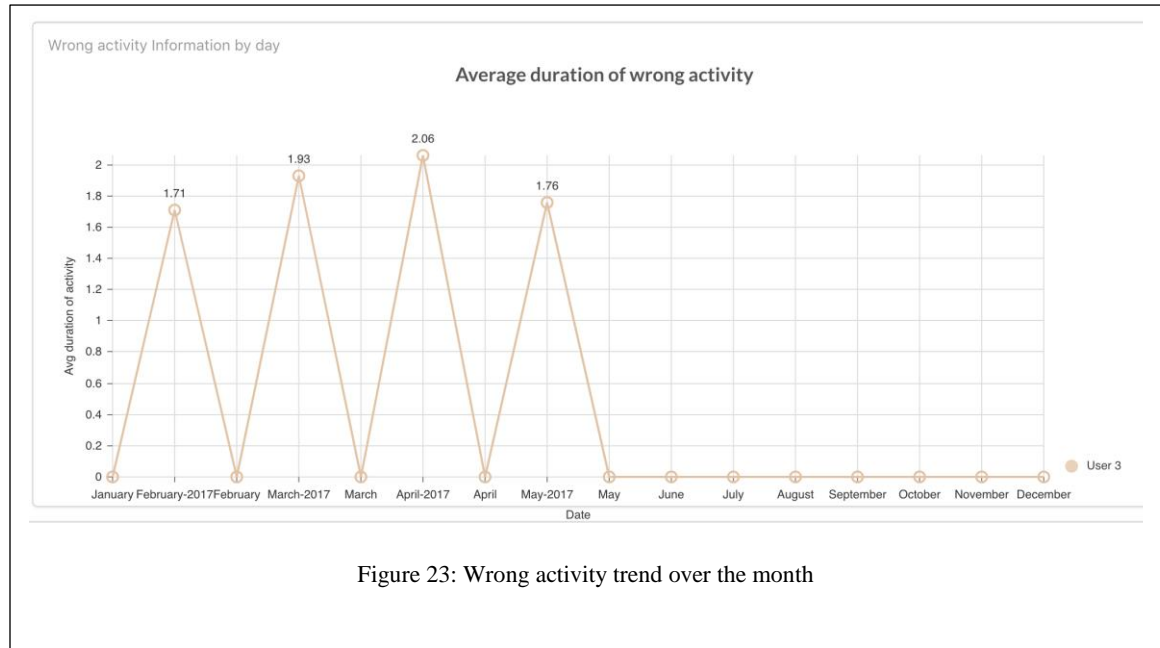
We can see from Figure 21 that Monday average wrong activities are very low. On Tuesday, Wednesday, Thursday, Friday and Saturday, the wrong activity duration is almost the same. To get more insights about the wrong activity trend over a weekday, it would be more helpful if we can see a number of wrong activities performed on each day. This is shown in Figure 22.



From Figure 22, we can see that the number of wrong activities are more on Wednesday. If we observe Figure 22, the average duration of the wrong activity on Wednesday is also high. So, we can conclude that User 3 is doing more wrong activities on Wednesday.

From the above analytics, the Administrator will provide feedback to users continuously and provide the required assistance to users in their daily activities. An

administrator might be interested to see user wrong activity trend over each month so that the administrator can identify how his assistance is helping the user. To provide more insight about this data to the administrator, we provided the duration of wrong activity trend by month. Figure23 shows the wrong activity trend over each month.



We can see from Figure 23 that we user's duration of wrong activity is not decreasing day by day. This help administrator to identify users with this pattern and notify them and provides further assistance if required.

With the above results we can actually compare different user activities graphically. With these charts, we can tell who is performing better but we want to know how much better User 3 when compared to User 6. With the existing information we cannot tell this. For this reason, we are proving normalized data so that we can actually compare two user's performance. To calculate normalized value per user, first we will find average duration of wrong activities by user. Then we will find the maximum value of these averages. In statistical terms, normalization can be calculated with following

formulae. If X is the series with values x_1, x_2, \dots, x_n and x_{maximum} is maximum value of this series and x_{min} is minimum value of this series, then normalized value of x_i is

$$x_i = \frac{x_i - x_{\text{min}}}{x_{\text{maximum}} - x_{\text{min}}}$$

In our case there is no maximum value for average duration. For this reason, we are considering maximum value in average duration as maximum value. Minimum value of our series in best case is 0 i.e. user did not perform any wrong activity. We are calling this normalized value as “Relative Risk Factor” as this represent a value from 0 to 1 based on average duration of wrong activities. We are calling value as relative risk factor because we considered maximum value as maximum value in current series values. In real scenarios, this can be any value. We can see this value in Figure 24.

Name	Reletive Risk Factor
User 1	1.00
User 5	0.31
User 8	0.26
User 3	0.12

Figure 24: Relative Risk Factors

From the above results, we can see that our system provides more insights into data collected and help users to avoid wrong activities. Along with this our system further provides good insights about different users to an administrator. The entire system is built by keeping simplicity in mind and to minimize number click to access data insights.

CHAPTER VII

CONCLUSION

The current application provides a lot of data insights into different dimensions. This helps both users and administrators to get to know their own or their employees working pattern and provide them appropriate suggestions and support. Current system user interface is effective and efficient to all users. It provides a cleaner and faster way to access data in a secured and privacy-aware manner. Providing user data at multiple granularities helps users and administrators to better understand their work routines. We can identify what work our users are doing at the point of time where they are performing more wrong activities for more time, then we can teach them how to do those activities without hurting their spine. We need to provide "Ergonomics and Body Mechanics Training and Education" to all users. This will provide information about how to improve their daily work routines.

BIBLIOGRAPHY

1. ISO 9241-11, Available: <https://www.iso.org/obp/ui/#iso:std:iso:ts:20282:-2:ed-2:v1:en>
2. Getting started with iBeacons and Windows 10, Available: <https://blog.cinlogic.com/2017/02/01/getting-started-with-ibeacons-and-windows-10/>
3. Bluetooth LE explorer, <https://www.microsoft.com/en-us/p/bluetooth-le-explorer/9n0ztkf1qd98?activetab=pivot:overviewtab>
4. S. S. Chawathe. Beacon placement for indoor localization using bluetooth. In The Proceedings of the 11th International IEEE Conference on Intelligent Transportation Systems, pages 980–985. IEEE, 2008
5. Z. Chen, Q. Zhu, and Y. C. Soh. Smartphone inertial sensor-based indoor localization and tracking with ibeacon corrections. IEEE Transactions on Industrial Informatics, 12(4):1540–1549, 2016
6. J. H. Christensen. Using restful web-services and cloud computing to create next generation mobile applications. In Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications, pages 627–634. ACM, 2009
7. D. De, P. Bharti, S. K. Das, and S. Chellappan. Multimodal wearable sensing for fine-grained activity recognition in healthcare. IEEE Internet Computing, 19(5):26–35, 2015
8. N. A. Dudhane and S. T. Pitambare. Location based and contextual services using bluetooth beacons: New way to enhance customer experience. Lecture Notes on Information Theory Vol, 3(1), 2015

9. J. Frisby, V. Smith, S. Traub, and V. L. Patel. Contextual computing: A bluetooth based approach for tracking healthcare providers in the emergency room. *Journal of biomedical informatics*, 65:97–104, 2017
10. Wenbing Zhao, Roanna Lun and Connor Gordon, "The design and implementation of a Kinect-based framework for selective human activity tracking," in *Proceedings of 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, October 2016.
11. Web services, Available: <https://www.javatpoint.com/web-services-tutorial>
12. Lumen: The stunningly fast micro-framework by Laravel, Available: <https://lumen.laravel.com/>
13. Eloquent, Available: <https://laravel.com/docs/5.8/eloquent>
14. Laravel controllers, Available: <https://laravel.com/docs/5.8/controllers>
15. Laravel routing, Available: <https://laravel.com/docs/5.8/routing>
16. ReactJS, <https://reactjs.org/>
17. Redux, <https://redux.js.org/>
18. Axios, <https://github.com/axios/axios>
19. Nivo, <https://nivo.rocks/>
20. Semantic UI React, <https://react.semantic-ui.com/>
21. MPAndroidChart, <https://github.com/PhilJay/MPAndroidChart>
22. Retrofit, <https://square.github.io/retrofit/>
23. MaterialChipsInput, <https://github.com/pchmn/MaterialChipsInput>
24. JavaScript, <https://developer.mozilla.org/en-US/docs/Web/JavaScript>
25. HTML5, <https://developer.mozilla.org/en-US/docs/Web/Guide/HTML/HTML5>

26. CSS 3, <https://developer.mozilla.org/en-US/docs/Web/CSS/CSS3>
27. Signing android application, <https://developer.android.com/studio/publish/app-signing>
28. Android developer portal, <https://developer.android.com/>
29. Android developer guide, <https://developer.android.com/guide>
30. Create-react-app, <https://github.com/facebook/create-react-app>
31. Kinect for Windows, <https://developer.microsoft.com/en-us/windows/kinect>
32. MySQL, <https://www.mysql.com/>
33. XML, <https://www.w3.org/XML/>
34. Android Studio, <https://developer.android.com/studio>