VISVESVARAYA TECHNOLOGICAL UNIVERSITY
Jnana Sangama, Belgavi-590018, Karnataka, INDIA

PROJECT REPORT
On

"Managing the Spread of Diseases in a Smart City Through Data Analysis: A Novel Approach"

Submitted in partial fulfillment of the requirements for the VIII Semester

Bachelor of Engineering
In
INFORMATION SCIENCE AND ENGINEERING

For the Academic year
2016-2017

BY

Chaithra P	1PE13IS401
Nandini Prasanna B P	1PE14IS405
Shreesha S	1PE13IS084
Syed Amjad Ali	1PE13IS100

Under The Guidance Of
Mr. Anand M S
Professor
Dept. of Information Science & Engineering, PESIT-BSC.

Department of Information Science and Engineering
PESIT Bangalore South Campus
Hosur Road, Bengaluru-560100

PESIT Bangalore South Campus

Hosur Road, Bengaluru-560100

Department of Information Science and Engineering

CERTIFICATE

*This is to certify that the project work entitled **"Managing the Spread of Diseases in a Smart City Through Data Analysis: A Novel Approach"** is a bonafide work carried out by **Chaithra P, Nandini Prasanna B P, Shreesha S, Syed Amjad Ali** and bearing USNs **1PE13IS401, 1PE14IS405, 1PE13IS084 and 1PE13IS100** respectively in partial fulfillment for the award of Degree of Bachelors (Bachelors of Engineering) in Information Science and Engineering of Visvesvaraya Technological University, Belagavi during the year 2016-2017.*

It is certified that all corrections/suggestions indicated for internal assessment have been incorporated in the Report. The project report has been approved as it satisfies the academic requirements in respect of project work prescribed for said degree.

Signatures:

Project Guide	Head, Dept. of ISE	Principal/Director
Mr. Anand M S	**Dr. Annapurna D**	**Dr. Surya Prasad J**
Professor, Dept. of ISE	PESIT-BSC, Bengaluru	PESIT-BSC, Bengaluru
PESIT-BSC, Bengaluru		

External Viva

Name of the Examiners	Signature with date
1.	
2.	

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of this project would be incomplete without the mention of the people who made it possible, without whose constant guidance and encouragement would have made our efforts to go in vain.

Our sincere gesture towards the **Management of PESIT, Bangalore South Campus** for fulfilling all our requirements throughout these years and providing us the right path to a better future.

We would like to thank **Dr. J Surya Prasad, Principal, PESIT Bangalore South Campus** for laying the foundation of this curriculum for us.

We would also like to thank **Dr. Annapurna D, HoD, Department of Information Science and Engineering** for projecting her able guidance, regular source of encouragement and assistance throughout this project.

Our heartfelt thanks to Project Coordinators, **Prof. Karthik S, Associate Professor,** and **Mrs. Kakoli Bora, Assistant Professor, Department of Information Science and Engineering** for having supported the work related to this project. Their contributions and technical support in preparing this report are greatly acknowledged.

We convey our sincere thanks to our project guide **Prof. M S Anand, Professor, Department of Information Science and Engineering** for providing encouragement, constant support and guidance which was of great help to complete this project successfully.

We also extend our gratitude to all the faculty members of Department of ISE, who have been encouraging throughout. My sincere thanks to my friends and family for their inputs and support.

Chaithra P (1PE13IS401)
Nandini Prasanna B P (1PE14IS405)
Shreesha S (1PE13IS084)
Syed Amjad Ali (1PE13IS100)

ABSTRACT

Vast amount of data is being generated every time we visit a hospital or a healthcare centre. This data has so far been largely unused to forecast or predict, using machine learning algorithms, the trends and number of cases that could be reported in future. It is important to forecast the cases to be reported so that the hospitals, municipal corporations as well as the citizens are cautioned about the disease spread and necessary actions can be taken/planned efficiently. In this project, we use Facebook's open sourced library 'Prophet' to train the regression model and predict the possible number of patients that are going to complain of a particular disease in future. We also compute a 'Location Vulnerability Index' for each area in a city to measure the relative risk of disease spread. We also provide a platform for effective communication among hospitals, civic authorities and citizens about risks, precautions, measures to curb the spread and community surveys. This novel approach aids a smart city in managing the spread of diseases through historical data analysis and effective communication among all stakeholders of the society.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1.1. Introduction

An era of open information in healthcare is now under way. We have already experienced a decade of progress in digitizing medical records, as pharmaceutical companies and other organizations aggregate years of research and development data in electronic databases. The government and other public stakeholders are also accelerating the move towards transparency by making decades of stored data usable, searchable, and actionable by the healthcare sector as a whole [1]. Healthcare data generated over a period of time plays a critical role in predicting the intensity and spread of a disease such as Malaria or Viral Fever in a particular geographic region. Managing such data is central to a smart city's capability to resile. A smart city can leverage the data produced by different stakeholders of the city to prevent the spread and prepare for the cure of the disease. The city's approach in case of an epidemic has to be systemic and systematic – it cannot be fragmented and incremental. The combinatorial complexity of managing and analyzing information is a major source of the difficulty.

The release of big data is transforming the discussion of what is appropriate or right for a patient and right for the healthcare ecosystem. In keeping with these changes, McKinsey & Company, in their study, have created a holistic, patient-centered framework that considers five key pathways to value, based on the concept that value is derived from the balance of healthcare spend (cost) and patient impact (outcomes):

(i) *Right living*: Patients can build value by taking an active role in their own treatment, including disease prevention,
(ii) *Right care*: This pathway involves ensuring that patients get the most timely, appropriate treatment available through a coordinated approach across settings and providers,
(iii) *Right provider*: This pathway proposes that patients should always be treated by professional experts that are best matched to the task so as to achieve the best outcome,
(iv) *Right value*: To fulfill the goals of this pathway, providers and payors will continuously enhance healthcare value while preserving or improving its quality,
(v) *Right innovation*: This pathway involves the identification of new therapies and approaches to delivering care, across all aspects of the system, and improving the innovation engines themselves.

1.1.1. Purpose of The Project

Although there have been tremendous advancements in the field of technology over the past couple of decades, we've largely ignored leveraging technology in healthcare until recently. This provides us with an opportunity to come up with products that help in predictive analysis of epidemic outbreak or disease spread.

As the saying goes "Prevention is Better Than Cure", we plan to build a product that uses machine learning and statistical models to find patterns in disease spread and potential epidemic outbreak. By doing so, we improve the Right living, Right care, Right provider, Right value and Right innovation ecosystem. This in-turn will improve the efficiency of disease prevention and cure by doctors, enable patients take appropriate precautionary measures, help governments and municipal corporations invest money wisely in appropriate areas and schemes.

Another purpose of this project is to equally involve the various stakeholders of a smart city, i.e., the public authorities, doctors, and citizens in creating awareness of the diseases spreading in the locality, providing precautionary measures. The concerted effort in improving the locality has both economic and social impacts.

1.1.2. Scope

The system primarily consists of a mobile application which will be used by the government/municipal authorities, and the citizens for information sharing amongst themselves as well as with us for data processing.

Apart from the mobile application, we have a Machine Learning algorithm predicting the number of cases which could occur in the next few days at a particular hospital and/or the vulnerability of a locality where an epidemic disease could spread. Before the data is consumed for learning, we would use automated scripts to clean the data and explore features.

The mobile application will a platform to disseminate insights to concerned stakeholders and thus allowing them to prepare for the prediction and/or educate them about the precautionary measures to avoid critical situations.
The hospitals would get access to the forecast of the possible number patients to approach them in the next few days for each disease through a web interface.

1.1.3. Impact

General impact would be realization of a healthy, aware and an engaged society with a potential for our product to predict and avert an epidemic breakout. From the citizens perspective, the most visible impact would be improved knowledge about diseases spreading in the locality and education on steps needed to be taken for its prevention. This general awareness by the public can hugely supplement the initiatives driven by the local authorities and cure prescribed by the doctors. From the healthcare provider's (Doctor) perspective, the impact would be the efficient data management to stock the required hospital resources (pharmacy, beds, staff, etc.) before the breakout of an epidemic. Also, the precautions taken by the informed citizen would help fasten the process as well as improve the effectiveness of treatment. Thus, the success rate of the treatment significantly improves.

From the government authority's perspective, the major impact would be in the success of their initiatives of generating awareness among the masses about the prevention and cure of the diseases and general dos-and-don'ts. The government authorities can also conduct surveys by directly connecting with affected/vulnerable citizens through the mobile app. It also saves huge amount of spending both by the government and the individual. The economic saving can thus have a cascading positive effect on various sectors of the society. To develop a measure of the economic gains that could come through the new value pathways, McKinsey & Company evaluated a range of healthcare initiatives undertaken by the US Government and assessed their potential impact as total annual cost savings, holding outcomes constant, using a 2011 baseline. Scaling these early successes out to system-wide impact, they estimated that the pathways could account for $300 billion to $450 billion in reduced healthcare spend.

1.2. Literature survey

[1] Peter Groves, Basel Kayyali, David Knott, Steve Van Kuiken, "The 'big data' revolution in healthcare", McKinsey & Company, January 2013

[2] Matthew Liotine, Arkalgud Ramaprasad, Thant Syn, "Managing a Smart City's Resilience to Ebola: An Ontological Framework", 2016 49th Hawaii International Conference on System Sciences[3] Jiwei Li, Claire Cardie, "Early Stage Influenza Detection from Twitter", November 2013, ACM

[4] Donal Simmie, Nicholas Thapen, Chris Hankin, "DEFENDER: Detecting and Forecasting Epidemics using Novel Data-analytics for Enhanced Response", April 2015
[5] Weng-Keen Wong, Andrew Moore, Gregory Cooper, Michael Wagner, "What's Strange About Recent Events (WSARE): An Algorithm for the Early Detection of Disease Outbreaks", Journal of Machine Learning Research 6 (2005)
[6] Raúl Isea, "A Potential Correlation Between the Temperature of the Pacific Ocean and Data from Google Trends May Yield a Warning Sign for the Outbreak of Zika", American Journal of Social Sciences 2016

1.3. Existing System

Though many academicians and analysts have carried out extensive research on using Big Data Analytics in healthcare to predict the spread of epidemics, there have been no citizen centric product released to the public. Also, there has been no release of any system as large scale as our proposed product so far in the public domain. This is a pilot project and built on top of the research of distinguished researchers and analysts.

1.4. Proposed System

We are proposing a system with three main components with continuous feedback and system improvement. The entire product can be disintegrated to three sub-systems, which are:

(i) *Data Collection*: This component will consist of a web application that will be used primarily by doctors/hospitals to update the number of cases reported for each disease every day. The app will have a simplistic design and a simple user interface to allow doctors (or hospital data operators) to update data fast and with ease. The data will be collected every day from each hospital from all localities of a smart city about the nature of the cases treated by the doctors who use the application and the anonymized patient profile. The data thus collected will be cleaned, organized well and stored in our servers.

(ii) *Data Analysis*: This component will run in the background without any customer facing application. The data procured in the previous component will be run against various algorithms [4][5] which we develop in the course of the project. The main aim of the algorithm will be:

- To predict the number of cases likely to be of a particular disease in the next '*x*' days for each locality based on historical data, where *x* is configurable.

- To predict the Vulnerability Index (VI) of a locality which signifies the risk that a person holds of contracting a specific disease if the person stays/visits that particular locality. The VI is a metric to measure the intensity of the disease spread in a locality on real-time basis (Updated every day).

(iii) *Ecosystem Engagement*: Based on the insights obtained in the second component, relevant information will be shared with the local municipal authority to take appropriate action to prevent further spread of disease or to enable them to plan accordingly for the precautionary steps to be taken. We will also enlighten the citizens in each area about the kind of health diseases spreading in their community and suggest them precautionary actions to be taken. The authorities can store appropriate pharmacy and drugs to minimize the effect of an epidemic. The municipal authorities will also be able to create survey forms and target specific citizen group for improved governance delivery. We will also include a feedback mechanism to improve the accuracy of the predictions made by the algorithm. The seasonal changes affecting the predictions shall also be considered for improved accuracy.

1.5. Statement of The Problem

Healthcare is one of the sectors which is at the verge of a huge technological disruption. People from the industry haven't experimented much in this sector as the problems are largely complex and extremely risky (E.g., Handling patient data, Accuracy of the analysis). The problem that we are aiming to solve is no doubt complex, but comparatively has a lower order of risk as we will not be dealing with the anonymized patient health records. At present, most governments and municipal authorities do a damage control after an epidemic breaks out in the locality, thus spending large amounts of money in rehabilitation of patients and reverting back to normalcy. Generally, there is a reactive approach towards handling the spread of diseases rather than a proactive approach. The doctors often don't see the epidemic spreading at the early stage, and even if they do recognize an epidemic spread, the latency in doctors informing the right authorities prove costly. In most cases the people are less educated about the right practices to prevent an epidemic breakout and are helpless when an epidemic strikes. Thus, we are trying to solve the problem of lack of information of epidemic breakout and thus bridging the three main stakeholders of a smart city for an effective living.

1.6. Summary

There is a moderately simple solution to this seemingly difficult problem of lack of information and coordination. The three stakeholders of a smart city (authorities, doctors and citizens) can be visualized as the three dots in the society. They lack coordination and seem random. But, if we connect the three dots such that each dots connects the other two with a single edge to form a triangle, there will be perfect coordination among themselves. This philosophy drove us to create a product that connects the local authorities, the doctors and the citizens. We built predictive models to estimate the number of cases that might be registered for a each/particular disease in each locality allowing micro-focus on problems and its solutions. We also predict the potential spread of disease to unaffected localities based on the weather data obtained from various points in the smart city. Ultimately, we would like to see better healthcare delivery and an improved disease-spread management framework, thus saving several resources for the individual and the government.

2.1. System Requirements Specification

Requirement specification is the activity of translating the information gathered during analysis into a requirement document. It gives a comprehensive description of the intended environment for the development of the software under consideration.

2.2. Operating Environment

The project consists of two major parts: a server backend that collects the data and performs analysis on it, and a mobile app that is the interface for end users to interact with the system. The requirements for both parts are mentioned in the following sections.

2.2.1. Hardware Requirements

- **Server:**
    - Processor: Intel Core i3 or better
    - RAM: 4 GB or higher
    - Disk space: 500 GB
    - A powerful GPU for executing machine learning tasks efficiently.
- **Mobile app:**
    - Processor: 1 GHz
    - RAM: 1 GB
    - Disk space: 100 MB

2.2.2. Software Requirements

Following are the software requirements of the proposed system, for the different environments:

❖ **Development Machine:**

- Operating system: Ubuntu 14.04 LTS
- Android SDK
- Android Studio 2.x IDE
- Oracle JDK 1.8
- nginx web server
- Apache Spark 2.1.1
- Node.js 6.x
- NativeScript 2.5
- MySQL

- Python 2.7.x or Python 3.4.x
- *Python packages:*
    - pyspark
    - fbprophet
    - numpy
    - matplotlib
    - Django
    - virtualenv

❖ **Server:**

- Operating system: Ubuntu 14.04 LTS
- nginx web server
- Apache Spark 2.1.1
- MySQL
- Python 2.7.x or Python 3.4.x
- *Python packages:*
    - pyspark
    - fbprophet
    - numpy
    - matplotlib
    - Django
    - virtualenv
    - gunicorn

❖ **Mobile Application:**

- Operating system: Android 4.1 (Ice-cream Sandwich) or newer
- Java

2.3. Functional Requirements

The system has the following functional requirements:

- Data needs to be collected from various sources such as hospitals, public health officials and doctors (using the web app).
- The system needs to store historical data for each hospital from all areas of a city/town, along with details of the municipal authority's details, in order to predict its occurrence in the future.

- Use the data thus stored to predict the possible number of patients to be diagnosed in next 10, 30, 60 days for each hospital.
- Predict the vulnerability of each geographical area to be affected by a potential epidemic.
- The system should provide a common platform for exchange of information and ideas between citizens and the local municipality corporation.

2.4. Non-functional Requirements

The system has the following non-functional requirements:

- The system needs to be reliable.
- The system should have high availability. It should remain operational at all times.
- The mobile app should have a simple interface and should be user friendly as it may be used by people with varying levels of technical know-how.
- The system should be designed with scalability in mind. It needs to handle a large volume of data and a high number of users.
- The mobile app should be supported on a range of devices, from low-end spectrum to those on the higher end.

2.5. User Characteristics

There are primarily three kinds of users who will engage with our system:

1. *Hospitals / Doctors:* The hospitals will use the web application to upload their daily patient data and view the predictions, trends, etc.
2. *Government Bodies:* This is any government body responsible for a geographical area such as municipal corporation, municipality, etc. The municipal corporation will use the mobile app to communicate with the citizens and educate them about various government initiatives with regard to disease spread prevention and immunizations. They will also have complete control on who they interact with (i.e., targeted citizen segment with a similar characteristics). The municipal corporations will also be able to host surveys and retrieve responses from the citizens through the mobile application to facilitate more appropriate governance delivery. It is necessary for the municipal corporations to get real-time data of the reported cases to assess the effectiveness of their initiatives and improve further. Hence, the consolidated area-wise data will be available for the municipal corporations through the web application.

3. *Citizens:* The citizens will be able to view the Vulnerability Index of their location (area) and other areas for each disease spreading in the respective areas.

2.6. Applications of the System

The system can be applied at each and every smart city in a country to drive smart governance and a data driven approach towards preparing for the healthcare needs of the citizens. Being a novel approach, this system needs to be implemented on a pilot basis and evaluated.

2.7. Advantages of the System

- The system helps the hospitals prepare themselves for unprecedented spurt in number of cases of a particular disease by stocking the pharmacies and medical equipment.
- The hospitals can access trends and analysis of the reported cases and undertake suitable research and development of specialized centers of treatment.
- The municipal corporations get exact data of the number of cases being reported for each disease on a daily basis.
- The corporations can devise appropriate initiatives to counter the spread of diseases and safeguard the citizens from potential epidemic breakouts.
- The corporations can create surveys and get feedback directly from the citizens as and when necessary.
- The citizens benefit from this system by getting daily updates on the vulnerability of an area (locality) in spreading a particular disease.
- The citizens get direct updates from the municipal corporations on new initiatives taken, general tips in keeping themselves safe from contracting the disease.

2.8. Summary

This chapter gives all the specific details about the working and development environment used to develop this project. The system implements the functional requirements such as the prediction algorithm, the information hub, etc. This chapter also outlines the three different user profiles: hospitals, government bodies and citizens and the advantages they reap from the implementation of the system.

3.1. High Level Design

A high-level design provides an overview of a solution, platform, system, product, service, or modules. Such an overview is important in a multi-system project development to make sure that each supporting component design will be compatible with its neighboring component design and the entire system as a whole. High-level design will usually include a high-level architecture diagram depicting the different components, interfaces and networks that need to be further specified or developed.

3.2. Design Considerations

This section describes many of the issues which need to be addressed or resolved before attempting to devise a complete design solution.

3.2.1. Assumptions and Dependencies

Since the domain of this project is healthcare, it is extremely difficult to procure historical health records and statistics from different hospitals without proper authorization and credible identity. Hence, we have generated close to 4.5 GB of data emulating the data supposed to be sent by various hospitals. For the generation of data, we have assumed the presence of 391 hospitals spread across Bengaluru. The data is generated for each day from January 1st, 2014. A suitable random number is assumed for the patient details while generating the master data.

Ideally, the data procurement is dependent on uninterrupted supply of data every day from all hospitals of a smart city under consideration.

3.2.2 Goals and Constraints

The goals of this system are to:

- Enable the hospitals to obtain predictions, trends and analysis of the patients complaining of particular diseases.
- Facilitate the municipal corporation to connect with the citizens directly and also obtain real-time healthcare statistics and analysis.
- Provide the citizens with the vulnerability index and opportunity to directly connect with the municipal corporation.

The constraints include the following:

- The data to be infused by the hospitals need to be of a specific format for easy integration at the server.

- The hospitals need to plug the data to our systems every day for accurate results.
- The municipal corporation representatives need to be adequately trained to use the mobile app and web app to effectively communicate with the people digitally.

3.3. System Architecture

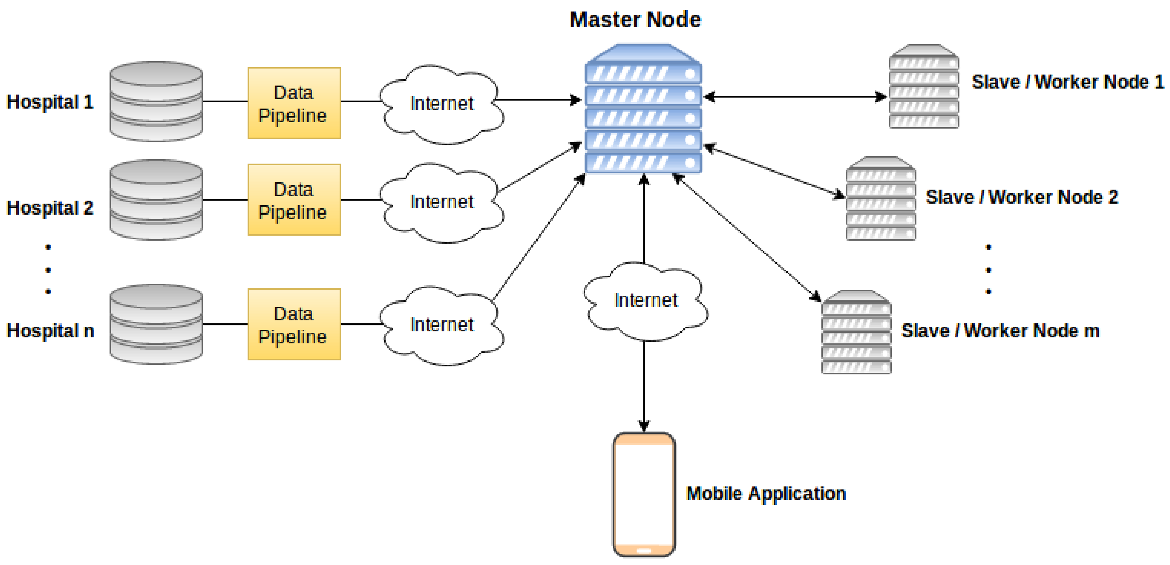


Fig. 3.1. System Architecture

Data from different hospitals will be collected and stored in the master node. The data pipeline denotes the transformations required, if necessary, to the data to clean it and bring to a uniform form. The data can be procured using API endpoints or .csv file uploads through web portal. Master node is the server which does all the processing and runs all the algorithms. The master node uses the help of the slave/worker nodes for faster execution and scalability. The mobile application uses the internet to access the vulnerability index and the broadcasts sent by the municipal corporations.

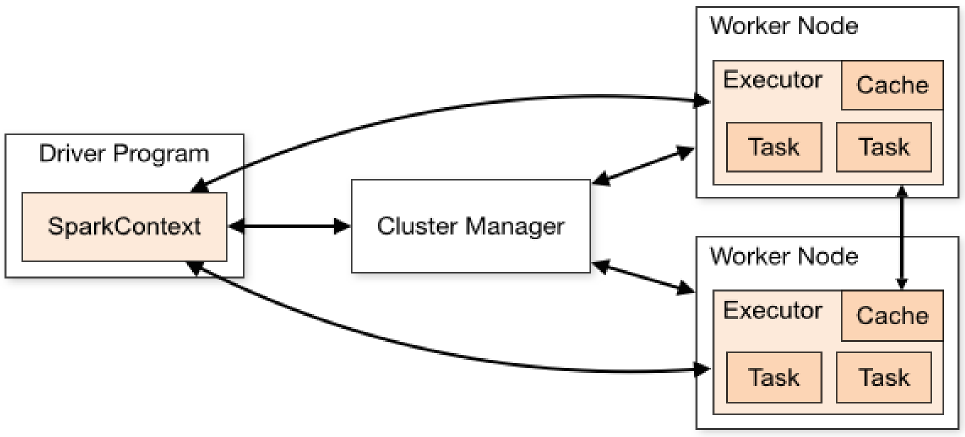


Fig 3.2. Apache Spark Architecture and Workflow

The above diagram represents the general workflow of a spark job. The cluster manager manages the allocation of the spark job to specific cluster nodes as appropriate at the time of executing the driver program. The driver program is split into several worker nodes and the result of each worker node is later combined and returned to the SparkContext.

3.4. Data Flow Diagrams

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system; where the data will come from and go to; and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel (as shown in a flow-chart).

3.4.1. Data Flow Diagram - Level 0

- ***Component 1***: This component predicts/forecasts the future number of patients that are expected to visit a particular hospital complaining of a particular disease. For this component, the patient data and the data of cases reported at each hospital is taken on a daily basis as input. Once the model is run on the input data, the output containing the predictions for next 'n' days (where 'n' can be configured easily. Eg. 10, 30, 60, etc.) is returned.

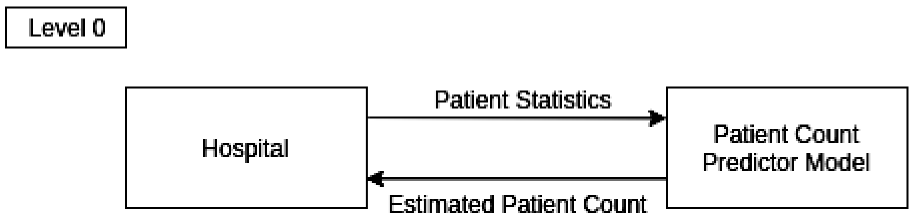


Fig. 3.3. Data Flow Diagram - Level 0, Component 1

- ***Component 2***: This component computes the Location Vulnerability Index by taking the consolidated area-wise hospital data as input and providing the normalized area-wise vulnerability index as the output which is then fed to information hub (mobile app) for the citizens to view.

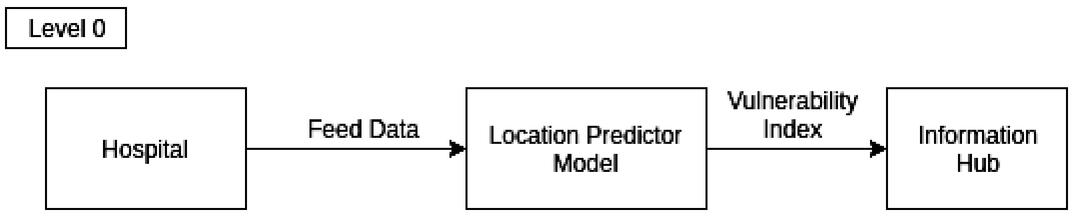


Fig. 3.4. Data Flow Diagram - Level 0, Component 2

- ***Component 3***: This component (Information Hub) acts as a platform for exchange of information between the citizens and the municipal corporation. The hospitals engage with the Information Hub's backend by uploading the patient statistics and receive the forecast of patients for next 'n' days, trends in the disease spread, etc. On the other side, the municipal corporation interacts with the citizens directly by sending them announcements of new initiatives, asking them to take online surveys and giving them general guidelines.

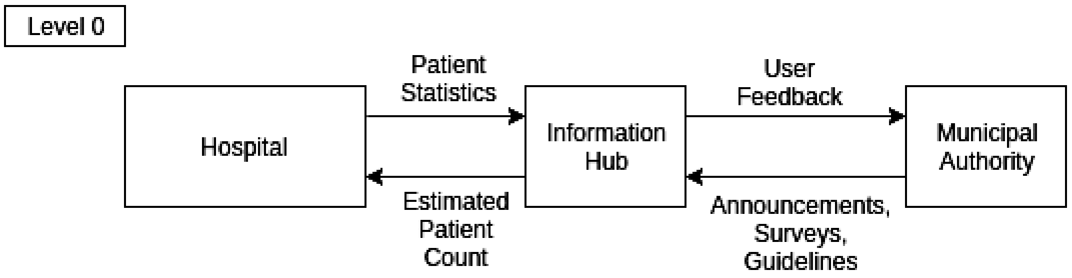


Fig. 3.5. Data Flow Diagram - Level 0, Component 3

3.4.2. Data Flow Diagram - Level 1

- ***Component 1***: While taking the input from hospitals, the system includes a data pipeline which feed transforms the data to suit it the algorithm implemented. This is an important step to maintain uniformity in the structure of the data received and thus ensure the sanctity of the output (results) produced.

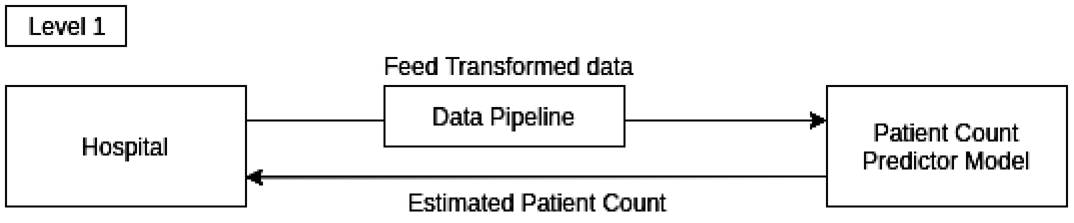


Fig. 3.6. Data Flow Diagram - Level 1, Component 1

- ***Component 2***: The data pipeline for this component slightly differs from the data pipeline outlined in the previous component. The feed data mentioned here is aggregated data from the cleaned data obtained in the previous component. This aggregation is done based on the area in which each hospital is situated in.

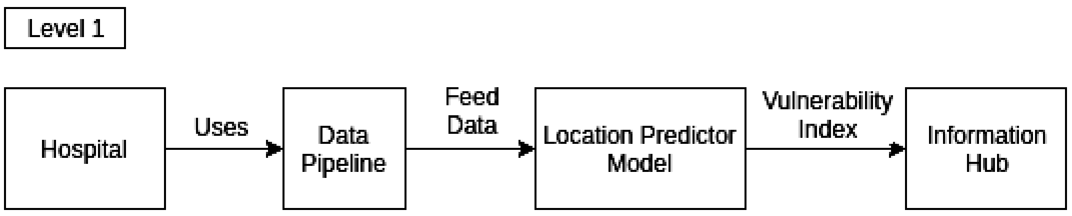


Fig. 3.7. Data Flow Diagram - Level 1, Component 2

- ***Component 3***: For level 1 of Data Flow Diagram, we can split the third component into two sub-components. The first sub-component is where the citizens can send survey responses, complaints, feedback etc. to the municipal corporation. In return, they can be contacted directly by the municipal corporation based on the nature of the query sent. The second sub-component comprises of the municipal corporation sending announcements, survey forms and general guidelines to the citizen through the Information Hub app.

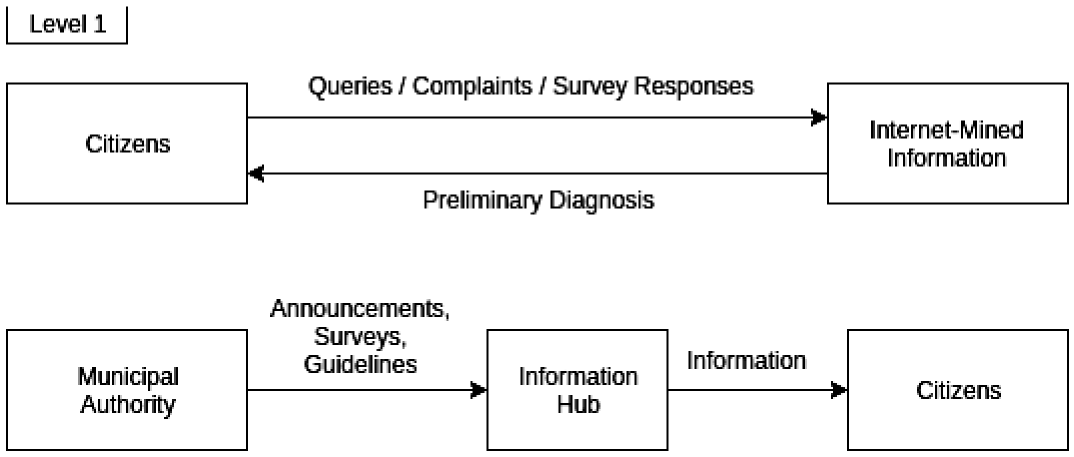

Level 1
Citizens
Queries / Complaints / Survey Responses
Internet-Mined Information
Preliminary Diagnosis
Municipal Authority
Announcements, Surveys, Guidelines
Information Hub
Information
Citizens

Fig. 3.8. Data Flow Diagram - Level 1, Component 3

3.4.3 Data Flow Diagram - Level 2

- ***Component 1***: Once the feed transformed data is obtained, the data is appended to the existing data we have (with respect to each hospital) and updated on our server. Since all patient data would be anonymized, the data security is not of utmost importance in this case. Once the data is stored, it is subject to a regression algorithm (*fbprophet*) and the results are collected.

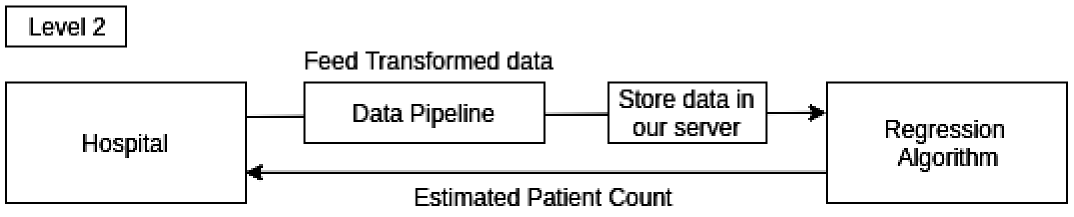


Fig. 3.9. Data Flow Diagram - Level 2, Component 1

- ***Component 2***: After the aggregated data is procured, the rate of change in the number of cases reported for the past 5 days is computed and the normalized rate is assigned to each area as the vulnerability index. This index is further sent to the Information Hub as and when requested by the user.

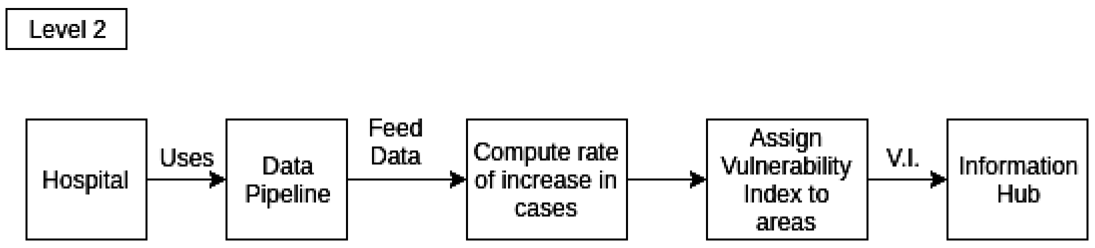

Level 2
Hospital
Uses
Data Pipeline
Feed Data
Compute rate of increase in cases
Assign Vulnerability Index to areas
V.I.
Information Hub

Fig. 3.10. Data Flow Diagram - Level 2, Component 2

3.5. Use case diagram

A use case diagram is a graphic depiction of the interactions among the elements of a system. A use case is a methodology used in system analysis to identify, clarify, and organize system requirements. The use cases, which the specific roles are played by the actors within and around the system.

As shown in the diagram below, the use case diagram consists of actors and the actions they do. The different actors are Hospital, Municipal Authority and Citizens. Each of these actors engage in some (exclusive or nonexclusive) activity.

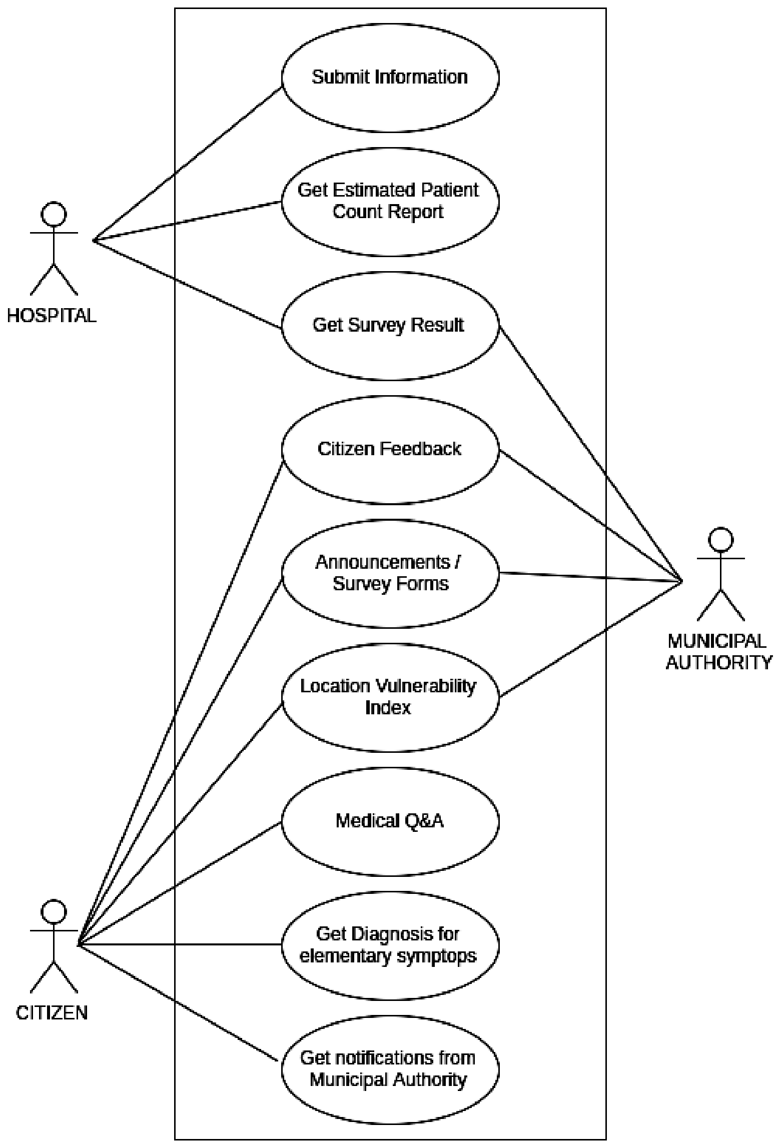

Submit Information
Get Estimated Patient Count Report
Get Survey Result
Citizen Feedback
Announcements / Survey Forms
Location Vulnerability Index
Medical Q&A
Get Diagnosis for elementary symptops
Get notifications from Municipal Authority
HOSPITAL
MUNICIPAL AUTHORITY
CITIZEN

Fig. 3.11. Use Case diagram for the system

3.6. Sequence Diagram

Sequence diagram in UML is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of Message Sequence Chart. Sequence diagrams are sometimes called Event diagrams, Event Scenarios and Timing diagrams.

Component 1 involves the Hospital and the Server. Once the patient data is uploaded, the data pipeline procedure is run (preprocessed in batch). Later, the processed data is sent to the server for storage and processing. The stored data is processed with the algorithms defined later in this chapter to obtain the prediction (forecast). Once the results are computed (and the graphs plotted), they are returned to the hospitals for their reference.

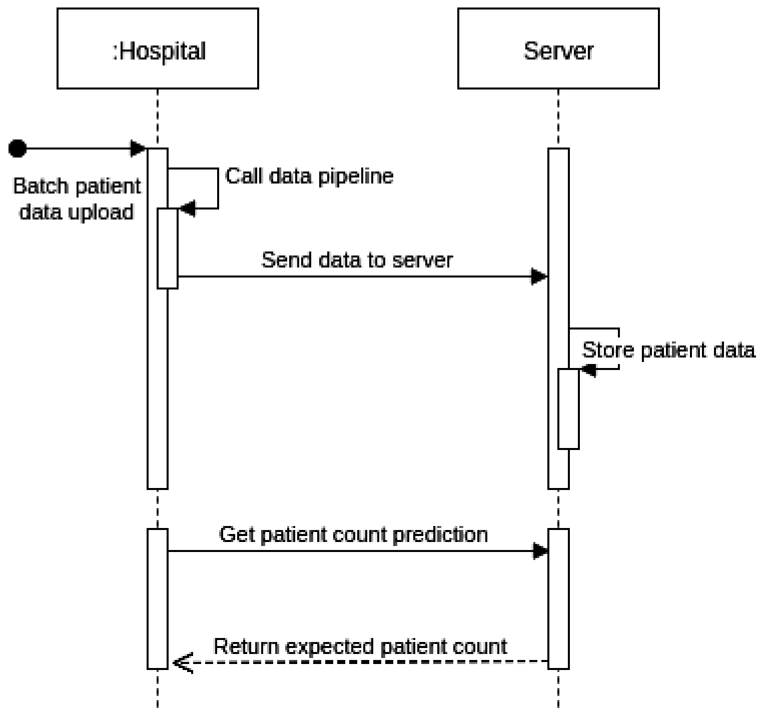

:Hospital
Server
Batch patient data upload
Call data pipeline
Send data to server
Store patient data
Get patient count prediction
Return expected patient count

Fig. 3.12. Sequence diagram of Patient Count Predictor Module (Component 1)

The sequence diagram for the Location Vulnerability Index (Component 2) is also the same as that of component 1 as the two components differ only by the algorithms used to compute them.

Component 3 involves three entities - Citizens, Hospitals and the Server. The sequence diagram below consists of 4 independent user interactions with the system. The citizen can query the server about particular details about the municipal corporation or the municipal officials about specific disease details. The citizen can also request the Vulnerability Index of particular area. The direct exchange of information between the citizen and the server is enabled through APIs.

The Municipal Authority can create notifications/announcements targeted at filtered demographic pool or all the citizens in general. The notifications are routed to the citizens through the central server.
The Municipal Authority can also create Survey Forms which can be accessed by the citizens when notified by the server. The responses recorded are further submitted to the Municipal Authority by the server after the expiry of the survey.

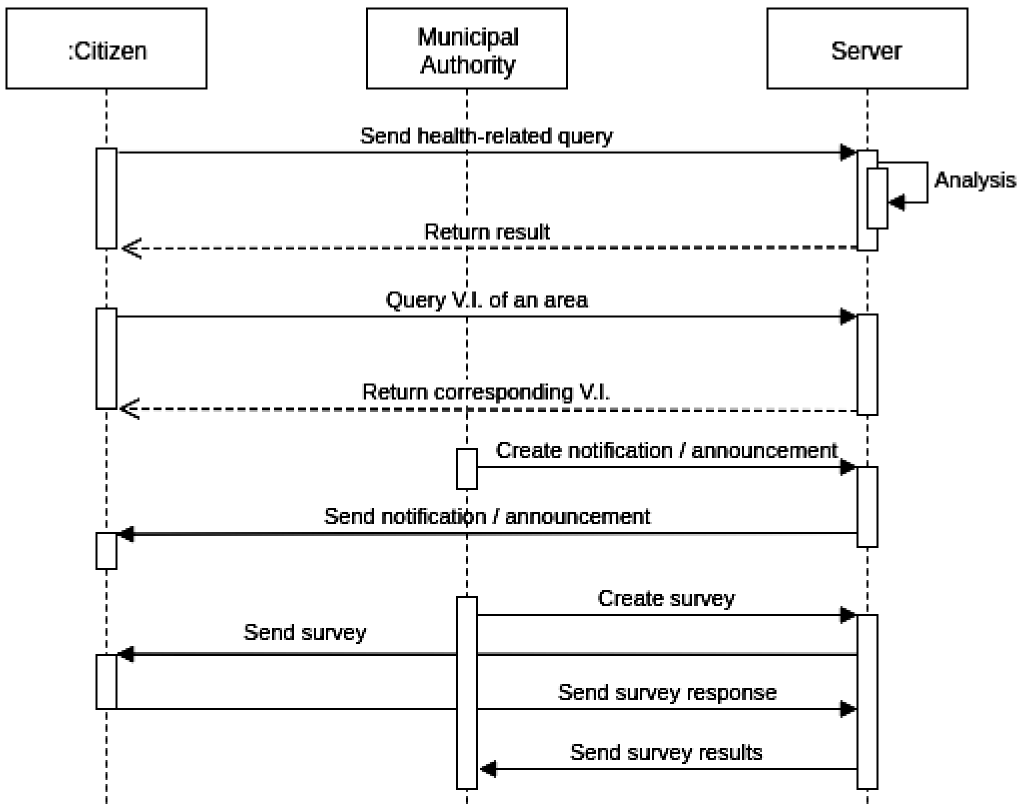


Fig. 3.13. Sequence diagram of Information Hub (Component 3)

3.7. Summary

This chapter gives us the detailed description of the user interaction with the system. To describe this, we have used interaction models such as use case diagrams and data flow diagrams. This chapter also provides the overall system architecture along with the assumptions to be made for the system and its constraints.

4.1 Purpose

The purpose of detailed design is to give clear details about the respective modules and interfaces of the system. It includes the components of the system, data structures and algorithms, and design of classes. These details once formally listed down form the design document, which acts as a blueprint for future development. This aids the developer in the implementation phase, as it is relatively easier to begin coding once all the decisions with regard to the design and structure are formally specified.

4.2 User Interface Design

A user interface is a system through which users interact with a machine. It includes both software (logical) and hardware (physical) components. Every device or application needs an interface that exposes the required functionality to the user, while keeping the internal details hidden.

This project consists entirely of software components, hence the user interface offered is a software graphical user interface. It is described in the subsection that follows.

4.2.1 Software Graphical User Interfaces

The system consists of three types of users – Hospitals, Government Bodies and Citizens. Each of these users has a separate interface to interact with the system that exposes functionality pertinent to the user.

Hospitals

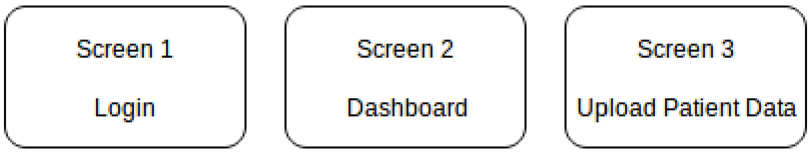


Fig. 4.1. Web GUI screens for Hospitals

Hospitals have access to a web-based graphical user interface (GUI) that supports the following functionality:

- Upload patient data for a day.
- View patient count predictions for each disease for the next *n* days.

The first screen that a hospital representative sees while accessing the interface is Login. After successful authentication, the user is taken to the Dashboard page. Dashboard gives them quick access to the predicted patient counts for the hospital for different diseases in the next *n* days.

Government Bodies

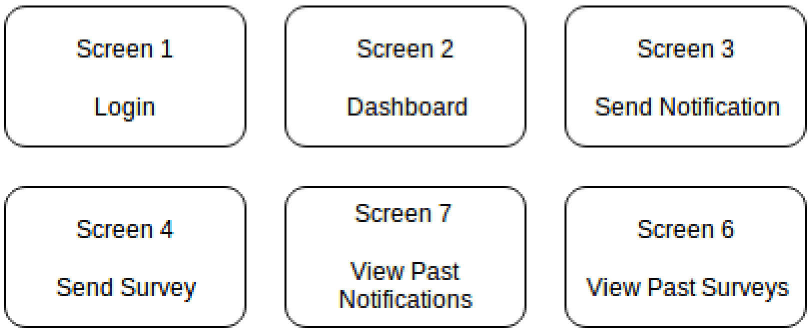


Fig. 4.2. Web GUI screens for Government bodies

Government bodies too have access to a web-based GUI that affords them the following functionality:

- View statistics of areas under their jurisdiction.
- Send targeted notifications.
- View past notifications.
- Send targeted surveys.
- View past surveys.

After successful authentication, a representative of a government body can access the other screens and perform the supported operations.

Citizens

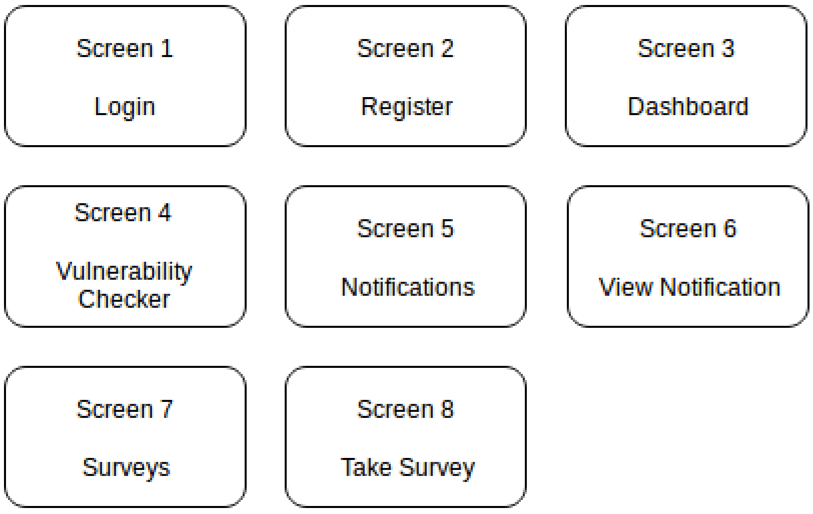


Fig. 4.3. Mobile application GUI screens for Citizens

The information generated by the system is made available to citizens through a user-friendly mobile application. It consists of the following features:

- Find diseases spreading in current location, along with their severity (Vulnerability Index).
- Check the Vulnerability Index of any disease for a particular state and city.
- Receive push notifications from government bodies about urgent alerts, informational messages, and so on.
- Take health-related surveys created by government bodies to get information from citizens.

Like the other two modules, the mobile application too starts with the Login screen which also allows new users to register themselves. After authentication, the Dashboard opens and the user can easily navigate between the different screens using a side menu.

4.3. Server Module

The server module forms the back-end of the system. It implements the following parts:

1. Web Application
2. API

4.3.1. Web Application

The web application provides web-based interfaces for hospitals and government bodies to use the system. It can be accessed from any location or device, using internet and a web browser. The functionality of the web application has been described in the previous sections discussing the GUI for hospitals and government bodies.

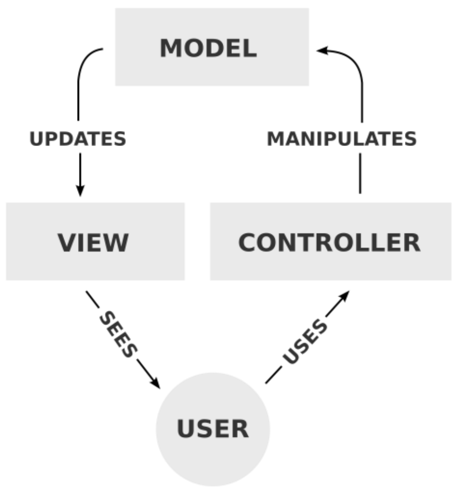


Fig. 4.4. MVC architectural pattern

It is based on the MVC (Model-View-Controller) architectural pattern. A Model-View-Controller pattern is made up of the following three parts:

- **Model** - The lowest level of the pattern which is responsible for maintaining data.
- **View** - This is responsible for displaying all or a portion of the data to the user.
- **Controller** - Software Code that controls the interactions between the Model and View.

The class design for the models used in the application are as follows:

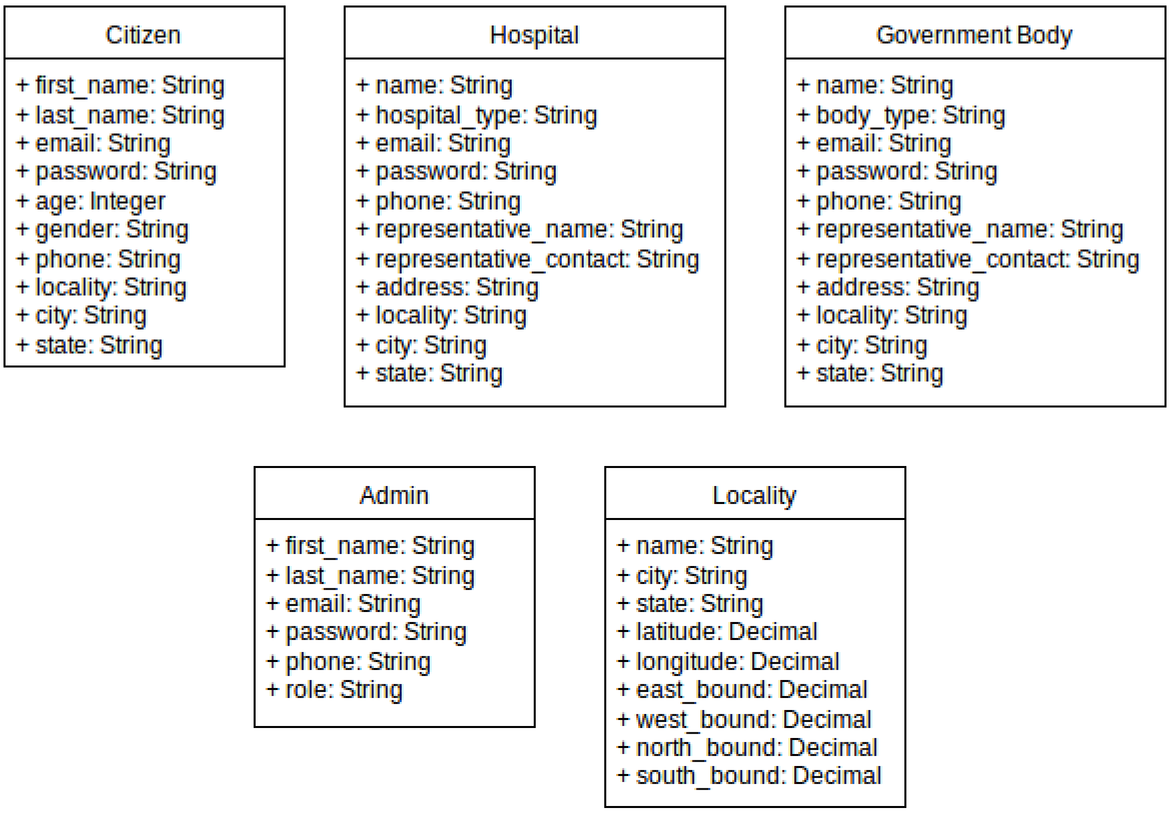


Fig. 4.5. Models for the web application

4.3.2. API

The HTTP API connects the server module with the mobile application. It allows the mobile application to programmatically send and retrieve data from the server. The request is sent from the application to a URL. If the request requires some data to be sent along, they are added to its parameters. Additional information such as authentication details are attached to the request's metadata. The responses from the server are sent in a form that can be easily parsed in the application.

4.4 Analysis Module

The analysis module consists of two sub-modules:

1. Patient Count Predictor Module
2. Location Vulnerability Index

4.4.1. Patient Count Predictor Module

This module is based on the pipes and filters architecture to sequentially process data and transfer the processed data to the next filter. One of the main algorithms used in this module is the *fbprophet*.

fbprophet [4] is a library for time-series forecasting open sourced by Facebook in February 2017. It takes into consideration the seasonality bias and the regression model is improved through constant feedback.

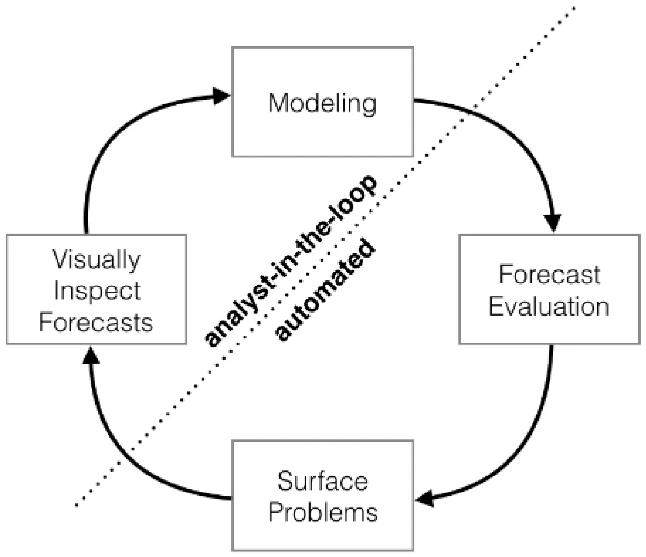


Fig 4.6. Facebook Prophet forecasting process

4.4.2. Location Vulnerability Index

This module is based on the custom developed algorithm. It follows the pipes and filter architecture. The different filters included are the area-wise data aggregation, filtering data to retain the last 5 days' data, compute the vulnerability_value, normalize the vulnerability_value to obtain the Vulnerability Index as shown in the figure 4.7 below.

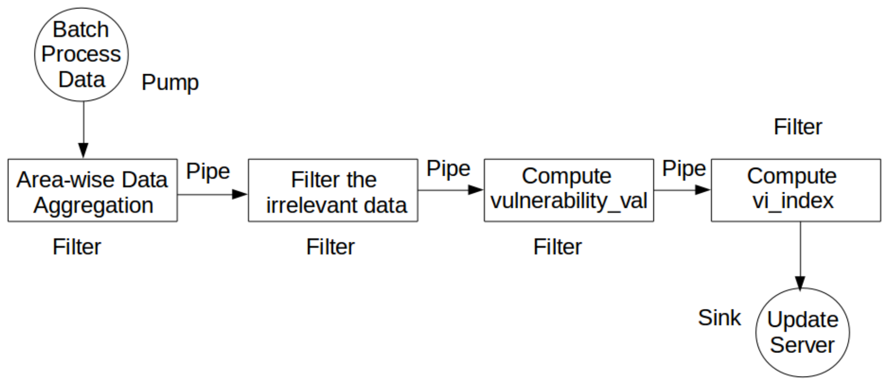


Fig. 4.7. Location Vulnerability Index Module Architecture

4.5 Mobile Application

Mobile application is the primary front-end of the system for the citizens. It uses the client-server architecture, where the mobile application is the client. The app makes calls to the server through internet in order to receive the data needed for rendering the different screens.

The screens of the application are as listed below. The functionality for these screens have been discussed in the earlier subsections.

- *Dashboard*
- *Vulnerability Checker*
- *Notifications*
- *View Notification*
- *Surveys*
- *View Survey*

Internally, each screen uses the MVC (Model-View-Controller) architectural pattern, just like the web application. In addition, we also make use of the Observer design pattern, which allows support for data binding. In Observer pattern, a one-to-many dependency between objects is established so that when one object changes state, all its dependents are notified and updated automatically. This allows, for instance, a variable in the logic side of a screen to be “binded” to an element on the screen, such as a TextView. When the variable changes value, the TextView is automatically updated, and vice versa. This fits in the “View” part of MVC.

4.6 Summary

This chapter describes in detail the design decisions we took with regard to all the modules in the system. It also discusses the functionality of the various modules and how users interact with them. This lays down a strong foundation for the upcoming stages of development and makes it easier for us to carry out the implementation with the help of the detail design.

5.1. Implementation

The implementation phase of any project development is the most important phase as it yields the final solution, which solves the problem at hand. The implementation phase involves the actual materialization of ideas, which are expressed in the analysis document and development in a suitable programming language in order to achieve the necessary final product. Often the product is ruined due to incorrect programming language chosen for implementation or unsuitable method of programming. It is better for the coding phases to be directly linked to the design phase in some sense that the design is done with object oriented approach.

The implementation stage involves careful planning, investigation of existing systems and constraints on implementation, designing of methods to achieve a changeover and evaluation of changeover methods.

The following steps are taken during this phase:

- Installation of Software Utilities
- Implementation and Testing of independent modules and sub-modules
- Integration and System Testing
- Iterative Testing and Enhancements

The factors concerning the programming language and platform chosen are described next.

5.2. Programming Language Selection

While implementing a software system, programming language selection is one of the key decisions that have to be made. Since there are so many programming languages to choose from, it is easy to get lost in the intricacies of each. The choice of programming language however depends on a variety of factors. Some of the factors that should be kept in mind while selecting the programming language are:

- Platform support and portability
- Applicability to the problem domain
- Ease of syntactical implementation
- Programming language skill of each team member working on the project

Considering the above factors, we have chosen Python as the primary language for web application and the server backend. However, the mobile application is developed using the NativeScript framework.

- *Python*: Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy which emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax which allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java. The language provides constructs intended to enable writing clear programs on both a small and large scale. Python's rich library (internal as well as external) help us implement various algorithms and applications with utmost ease and precision.
- *TypeScript*: TypeScript is a free and open-source programming language developed and maintained by Microsoft. It is a strict syntactical superset of JavaScript, and adds optional static typing and class-based object-oriented programming to the language. TypeScript may be used to develop JavaScript applications for client-side or server-side (Node.js) execution. TypeScript is designed for development of large applications and transcompiles to JavaScript. As TypeScript is a superset of JavaScript, any existing JavaScript programs are also valid TypeScript programs.

5.3. Platform Selection

Django: Django is a free and open source web application framework, written in Python. A web framework is a set of components that helps in developing websites faster by reducing the amount of repeated code. Django lets programmers use the Python language to build web applications and takes care of the commonly needed functionality such as authentication, form handling, website administration, etc. [2]

NativeScript: NativeScript is an open-source framework to develop apps on the Apple iOS and Android platforms. NativeScript apps are built using platform agnostic programming languages such as *JavaScript or TypeScript* for the logic and *CSS and XML* for the styling and layout . NativeScript directly supports the AngularJS framework.

Mobile applications built with NativeScript result in fully native Apps, which use the same APIs as if they were developed in Xcode or Android Studio [3].

NativeScript and all the required plugins are installed using the package manager npm. Projects are created, configured and compiled via the command line. Platform independent user interfaces are defined using XML files. NativeScript then uses the abstractions described in the XML files to call native elements of each platform. Application logic developed in Angular2 and TypeScript can be developed independent of the target platform as well. A NativeScript mobile application is built using the node.js runtime and tooling

5.4. Code Conventions

Throughout the project, we have followed the PEP8 Style Guide for Python. PEP8 gives coding conventions for the Python code comprising the standard library in the main Python distribution. [4]

5.4.1. Naming Conventions

- *Packages*: Package names are lowercase strings, optionally separated by a ‘.’. E.g., from pyspark.sql import SparkSession
- *Class Names*: Class names should normally use the CapWords convention. E.g., import SparkSession
- *Function Names*: Function names should be lowercase, with words separated by underscores as necessary to improve readability. E.g., def compute_predicted_patient_count():
- *Variables/Objects*: Variables or objects use lowercase words separated by underscores.
  E.g., trend_graph, daily_count = 0.
- *Constants*: Constants are usually defined on a module level and written in all capital letters with underscores separating words. E.g., NUM_HOSPITALS, PREDICT_FOR_DAYS
- *Comments*: Single line comments starts with ‘#’ and multiline comments starts with ‘“““’ (triple quotes) and ends with ‘”””’ (triple quotes again). E.g., # This is a single line comment.

5.4.2. File Suffix and File organization

5.4.2.1. Commonly used file suffixes:

1. .py : Python Source files
2. .ts : TypeScript Source files
3. .css : CSS style files
4. .xml : Mobile screen Layouts files
5. .png : Image Files
6. .csv : Comma Separated Values - Data files required for processing

5.4.2.2. File Organization:

1. *Analysis Module*

- Data/
    - generate_data.py - Contains Python code to programmatically generate simulated data
    - hosp_<hospital_ID>.csv - Contains the simulated data generated for each hospital
- Graphs/
    - <disease>_predict_<hospital_ID>.png - Contains the graph of predicted data
    - <disease>_trend_<hospital_ID>.png - Contains the graph of trend analysis
- Location VI/
    - aggregate_data_<date>.csv - Master file containing VI for all possible disease-area pairs
    - <disease>_location_VI.csv - Contains disease specific VI for all areas
- Predictions/
    - <disease>_hosp_<hospital_ID>.csv - Contains the predictions for '*n*' days (as specified)
- spark/ - Contains the Spark Server related resources
    - bin
    - conf
    - data
    - licenses
    - python

    - sbin
    - yarn
    - RELEASE
    - LICENSE
- spark-warehouse/ - Contains all the temporary files created while running Spark jobs
- location_vi.py - Contains Python code to compute the Vulnerability Index for each area
- patient_predictor.py - Contains Python code to forecast the number of patients for next ‘*n*’ days.

2. *Python/Django Web Application*

   The following hierarchy shows the file organization for the web application:

- *accounts/* - App for authentication of users.
    - *admin.py*
    - *forms.py*
    - *models.py*
    - *tests.py*
    - *urls.py*
    - *views.py*
- *backend/* - Project app, containing the main URLs and settings
    - settings.py
    - urls.py
    - wsgi.py
- *core/* - Core app, containing all the logic
    - admin.py
    - forms.py
    - models.py
    - tests.py
    - views.py
- *static/* - Directory to store all static files such as images, stylesheets and JavaScript files.
- *templates/* - Directory to store HTML templates for user-facing views.
- *manage.py* - Script to perform administrative actions such as starting the app and migrating database.

3. *Mobile Application*

   The following hierarchy shows the file organization for the mobile application:

- *app/*
  - *App_Resources/* - Directory for static resources such as images, styles and values
  - *fonts/* - Directory for external fonts used in the app
  - *models/* - Directory for models used in the app
  - *shared/* - Directory for shared modules and configuration
  - *views/*
    - *<view-name>/*
      - *<view-name>.ts* - File containing the logic for the view
      - *<view-name>.xml* - File defining the layout of the view
      - *<view-name>.css* - File containing CSS rules specific to the view
  - *app.css* - Application-wide CSS stylesheet
  - *app.ts* - TypeScript file that serves as the starting point of the app
- *node_modules/* - Directory that contains all the external modules used in the app.
- *platforms/* - Directory that contain platform specific code.
- *package.json* - File containing meta information for the app

5.5. GUI Design

- *Web Application*: The web application's Graphical User Interface is implemented using HTML/CSS. Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications. With Cascading Style Sheets (CSS) and JavaScript it forms a triad of cornerstone technologies for the World Wide Web. Web browsers receive HTML documents from a web-server or from local storage and render them into multimedia web pages. HTML describes the structure of a web page semantically and originally included cues for the appearance of the document. HTML elements are the building blocks of HTML pages. With HTML constructs, images and other objects, such as interactive forms, may be embedded into the rendered page. It provides a means to create structured documents by denoting structural semantics for text such as headings, paragraphs, lists, links, quotes and other items. CSS is a cornerstone technology used by most websites to create visually engaging web pages, user interfaces for web applications, and user interfaces for many mobile applications.

- *Mobile Application*: NativeScript, the framework used to develop the mobile application comprises of the following modules to help implement better interfaces [3]:
    - *StackLayout*: This layout arranges its children horizontally or vertically. The direction is set with the orientation property.
    - *GridLayout*: This layout defines a rectangular layout area that consists of columns and rows.
    - *ScrollView*: The ScrollableView component allows you to display a scrollable area in your application, which has content that is larger than its bounds. The ScrollView has an orientation property, which allows you to set different orientations to the view.

      The possible values of orientation are:
        - horizontal
        - vertical
    - *ActionBar*: The ActionBar is NativeScript's abstraction over the Android ActionBar and iOS NavigationBar. It represents a toolbar at the top of the activity window, and can have a title, application-level navigation, as well as other custom interactive items.
    - *Label*: Provides a text label that shows read-only text.
    - *TextView*: The TextView widget provides an editable multiline text view.
    - *Button*: Provides a standard button widget that reacts to a tap event.
    - *MapView*: MapView is part of the NativeScript Google Maps SDK which supports views that can embed Google maps and its related functionalities.
    - *ListView*: The ListView shows items in a vertically scrolling list. We can set an *itemTemplate* to specify how each item in the list should be displayed.
    - *SideDrawer*: It is a component can show a hidden view that contains navigation UI or common settings. The hidden view can be displayed with a flick gesture and can be shown from any of the four edges of the screen. The view is also displayed with a transition which can be chosen from a set of predefined transitions.

5.6. Algorithms

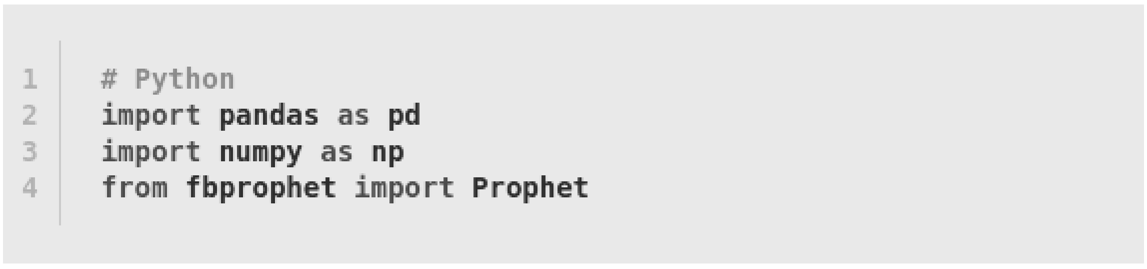
5.6.1. Patient Count Predictor

The patient Count Predictor forecasts the number of patients that are expected to visit a particular hospital complaining of a specific disease. This forecasting is based on historical data, where an additive regression model is used to plot a model and forecast. To ease the implementation and improve the accuracy, we have used an open sourced library *fbprophet*.

Prophet [5] is a forecasting tool open sourced by Facebook in February 2017 available in Python and R. At its core, the Prophet procedure is an additive regression model with four main components:

1. A piecewise linear or logistic growth curve trend. Prophet automatically detects changes in trends by selecting changepoints from the data.
2. A yearly seasonal component modeled using Fourier series.
3. A weekly seasonal component using dummy variables.
4. A user-provided list of special days/seasons.

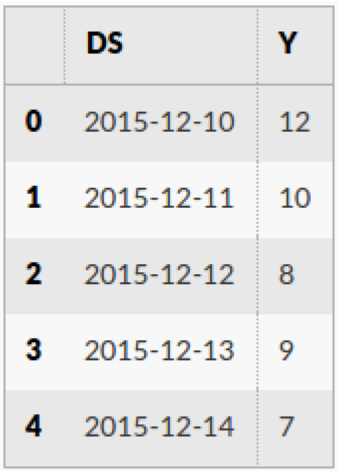
The Prophet library and its dependencies can be imported as shown below:



```
# Python
import pandas as pd
import numpy as np
from fbprophet import Prophet
```

Fig. 5.1. Import Prophet library

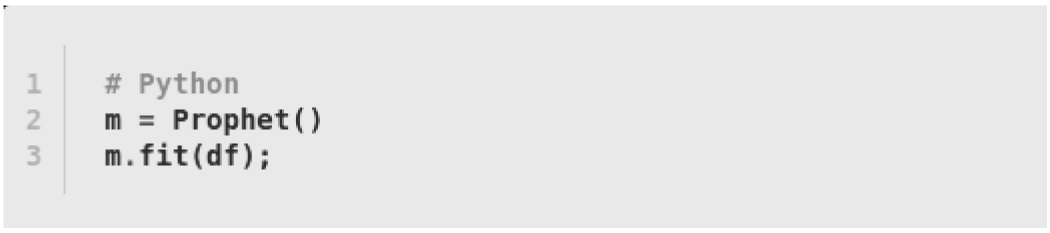
A DataFrame need to be readied with the time series and the corresponding total patient count reported as shown in the below figure to fit the model and predict future values:



	DS	Y
0	2015-12-10	12
1	2015-12-11	10
2	2015-12-12	8
3	2015-12-13	9
4	2015-12-14	7

Fig. 5.2. DataFrame which is a parameter to the model fit function.

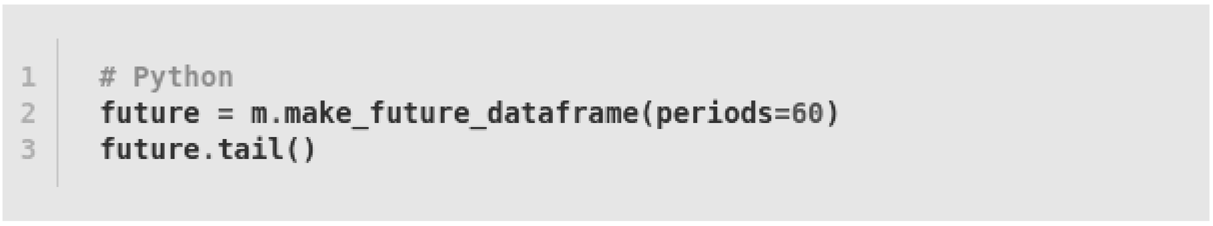
We fit the model by instantiating a new Prophet object. Any settings to the forecasting procedure are passed into the constructor. Then we call its fit method and pass in the historical DataFrame. Fitting should take 1-5 seconds.



```python
# Python
m = Prophet()
m.fit(df);
```

Fig. 5.3. Code to instantiate a Prophet object and fit the model.

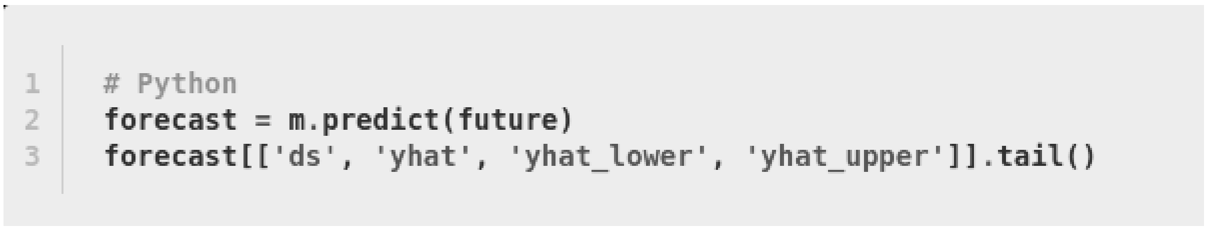
Predictions are then made on a DataFrame with a column ds containing the dates for which a prediction is to be made. You can get a suitable DataFrame that extends into the future a specified number of days using the helper method Prophet.make_future_dataframe. By default, it will also include the dates from the history, so we will see the model fit as well.



```python
# Python
future = m.make_future_dataframe(periods=60)
future.tail()
```

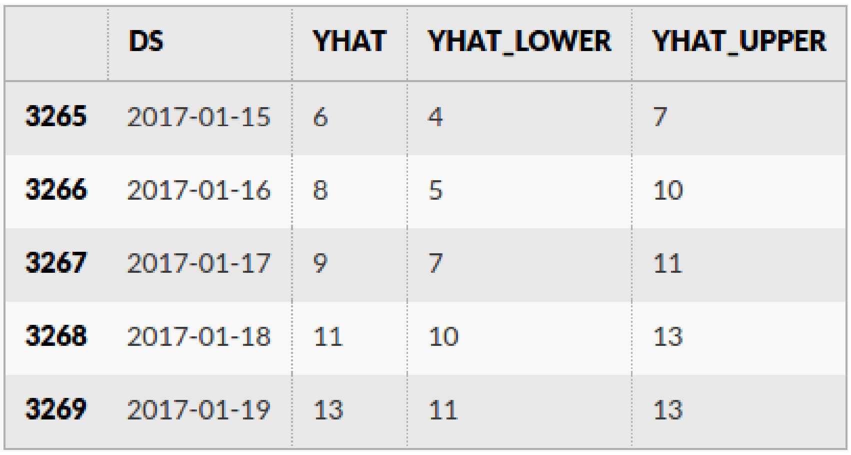
Fig. 5.4. Code to forecast the patient count for the next 60 days and print the last 5 days' results produced.

The predict method will assign each row in future a predicted value which it names yhat. If you pass in historical dates, it will provide an in-sample fit. The forecast object here is a new DataFrame that includes a column yhat with the forecast, as well as columns for components and uncertainty intervals (lower and higher limit of the forecasted value).



```
# Python
forecast = m.predict(future)
forecast[['ds', 'yhat', 'yhat_lower', 'yhat_upper']].tail()
```

Fig. 5.5. Code to forecast the values as well as include the uncertainty levels



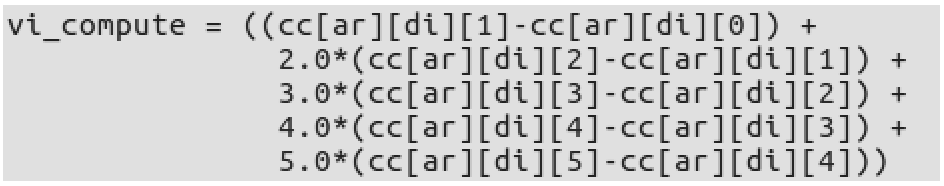
	DS	YHAT	YHAT_LOWER	YHAT_UPPER
3265	2017-01-15	6	4	7
3266	2017-01-16	8	5	10
3267	2017-01-17	9	7	11
3268	2017-01-18	11	10	13
3269	2017-01-19	13	11	13

Fig. 5.6. DataFrame forecasted containing the forecasted values and the limits.

5.6.2. Location Vulnerability Index

This algorithm is used to compute the Location Vulnerability Index which indicates the risk a person holds on visiting/living in that particular area of contracting the specific disease. This metric measures the intensity of spread of the disease and the rate of spread which are both critical in measuring the risk. By this metric, we hope to raise awareness in the people who are travelling to high risk area and imply to take necessary precautions to keep themselves safe from contracting the communicable disease.

Once the data is aggregated for each area specific to a disease, the vulnerability value is computed by giving differential weights to the most recent days of the past five days.



```
vi_compute = ((cc[ar][di][1]-cc[ar][di][0]) +
              2.0*(cc[ar][di][2]-cc[ar][di][1]) +
              3.0*(cc[ar][di][3]-cc[ar][di][2]) +
              4.0*(cc[ar][di][4]-cc[ar][di][3]) +
              5.0*(cc[ar][di][5]-cc[ar][di][4]))
```

Fig. 5.7. Formula to compute the VI value.

The VI values computed for each area is then normalized to uniformly distribute the values on a scale of 0 to 10 so as to easily represent it on a heatmap on the mobile application.

5.7. Summary

This chapter constitutes the implementation details as followed in the project. The chapter starts with the details of programming language selection and the platform selection. This is followed by the naming convention followed during the coding phase (PEP8 for Python and related frameworks) and the file organization required for smooth placement of input, output files and easy access to the relevant scripts. The User Interface elements used in the project is outlined which provides a preview of why the respective element was used. The chapter ends with the description of the two main algorithms implemented in the project.

6.1. Software Testing

Software testing is an examination accompanied to deliver evidence about the excellence of the product or system under test. It establishes the quality of the product and the extent to which it meets the specifications from an unbiased view. Test methods include the procedure of probing a program or application with the intent of finding software bugs (errors or other defects), and verifying that the software product is fit for use.

Software testing typically tries to execute a program or application with the objective of finding software bugs or errors or other defects. The task of testing is an iterative process as when one bug is fixed; it can illuminate other, deeper bugs, or can even create new ones.

As the number of possible examinations for any simple software components is practically infinite, all software testing uses some strategy to select tests that are feasible for the available time and resources. As the outcome, software testing typically tries to execute a program or application with the resolute of finding software bugs or errors or other defects. The task of testing is an iterative process as when one bug is fixed, it can illuminate other, deeper bugs, or can even create new ones.

6.2. Testing Process

The process begins by testing each independent module first, then testing multiple modules at a time to check integrity of the application. This project used Top down decomposition method for Unit testing and Big bang approach for Integration testing.

6.2.1. Levels of Testing

They are normally perceived by 3 types: unit testing, integration testing and system testing. Examination is centrally continuous by where they are added in software development process, or by the level of specificity of the test.

There are two different levels of tests from the view of customers: low-level testing (LLT) and high-level testing (HLT). HLT is a group of tests for the whole software application or product. LLT is a group of tests for different level components of software application or product.

6.3. Test environment

Test environment refers to the environment which is needed to execute the tests. The test environment to be used varies with the technological stack. In this project, we used Python's unittest module to test the Server and Analysis modules. For the mobile application, we used Mocha test framework with Chai assertion library.

6.4. Unit Testing of Main Modules

6.4.1. Unit Testing of Server Module

Execution of the Server Module is tested for various conditions and the test cases are tabulated as follows:

Test Case ID	1
Description	Negative Test Case for Login method
Input	A username containing special characters
Expected Output	Login should fail because the username is invalid
Actual Output	Login fails because username is invalid
Remarks	Test Case Passed

Table 6.1. Unit Test Case 1 for Server Module

Test Case ID	2
Description	Negative Test Case for Get diseases in location
Input	A decimal number and an alphabetical string
Expected Output	Error because a coordinate can only be a decimal number
Actual Output	Error because one of the coordinates is invalid
Remarks	Test Case Passed

Table 6.2. Unit Test Case 2 for Server Module

Test Case ID	3
Description	Negative Test Case for Get diseases in location method
Input	A decimal number and an alphabetical string
Expected Output	Error because a coordinate can only be a decimal number

Actual Output	Error because one of the coordinates is invalid
Remarks	Test Case Passed

Table 6.3. Unit Test Case 3 for Server Module

Test Case ID	4
Description	Positive Test Case for Get VI for disease and location method
Input	A disease, city and state
Expected Output	List of VIs for that disease in the given city and state
Actual Output	List of VIs for that disease in the given city and state
Remarks	Test Case Passed

Table 6.4. Unit Test Case 4 for Server Module

6.4.2. Unit Testing of Analysis Module

Execution of the Analysis Module is tested for various conditions and the test cases are tabulated as follows:

Test Case ID	1
Description	Test Case to check the seasonality bias of the patient count predictor module
Input	Patient data and statistics with deliberate increase of cases in June of every year
Expected Output	The forecasted output should predict more cases in the month of June
Actual Output	The actual output predicted a more than normal cases for the month of June
Remarks	Test Case Passed

Table 6.5. Unit Test Case 1 for Analysis Module

Test Case ID	2
Description	Test Case to check the robustness of the Location Vulnerability Index module
Input	Volatile (increasing and decreasing) data for last 5 days for a particular area
Expected Output	Catch the trend of the most recent day(s) - Upward or downward
Actual Output	The output could rightly identify the recent trend and compute VI

	accordingly
Remarks	Test Case Passed due to the weighted computation formula

Table 6.6. Unit Test Case 2 for Analysis Module

6.4.3. Unit Testing of Mobile Application

Execution of the Mobile Application is tested for various conditions and the test cases are tabulated as follows:

Test Case ID	1
Description	Negative Test Case for Login screen
Input	Invalid email address
Expected Output	Error logging in because email address is invalid
Actual Output	Error logging in because email address is invalid
Remarks	Test Case Passed

Table 6.7. Unit Test Case 1 for Mobile Application

Test Case ID	2
Description	Positive Test Case for Get notifications screen
Input	Empty notifications table
Expected Output	No notifications available at present
Actual Output	No notifications available at present
Remarks	Test Case Passed

Table 6.8. Unit Test Case 2 for Mobile Application

Test Case ID	3
Description	Positive Test Case for opening a survey
Input	User taps on a particular survey
Expected Output	The survey that was tappen should open up
Actual Output	The tapped survey opens
Remarks	Test Case Passed

Table 6.9. Unit Test Case 3 for Mobile Application

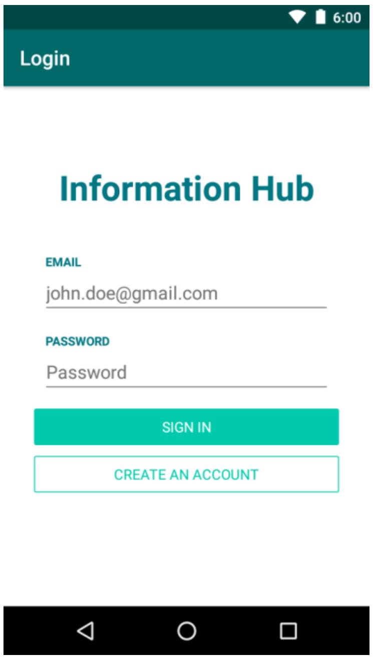
6.5. Integration Testing of Modules

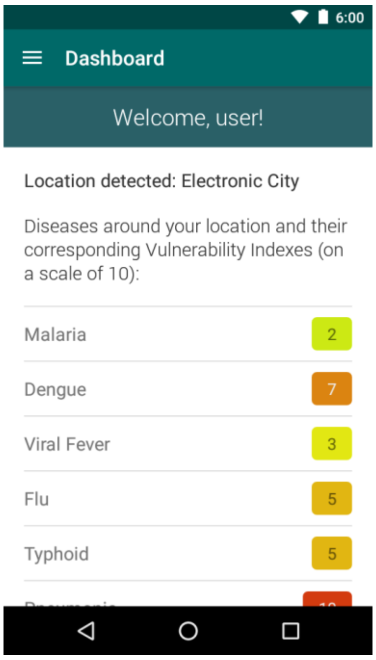
After successfully executing the unit tests, the three modules – Server Module, Analysis Module and Mobile Application – were integrated together and all operations were tested manually.

Dummy notifications were sent from the server and successfully received at the mobile application. However, the notifications failed to be delivered and appropriate error message displayed when there were internet issues/no internet connectivity.
Since the Analysis Module is processed in batch, the application would seek either the stored data or updated (new) data.

All test cases passed as expected, therefore integration testing results in a positive outcome. All modules work as expected after integrating them together.

6.6. Result Snapshots



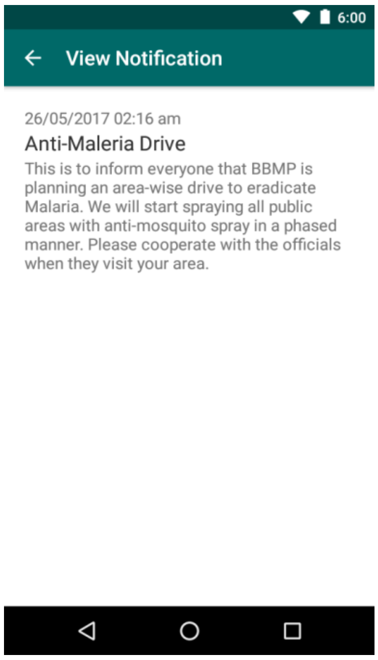


Welcome, user!

Location detected: Electronic City

Diseases around your location and their corresponding Vulnerability Indexes (on a scale of 10):

Malaria	2
Dengue	7
Viral Fever	3
Flu	5
Typhoid	5



26/05/2017 02:16 am
Anti-Maleria Drive
This is to inform everyone that BBMP is planning an area-wise drive to eradicate Malaria. We will start spraying all public areas with anti-mosquito spray in a phased manner. Please cooperate with the officials when they visit your area.

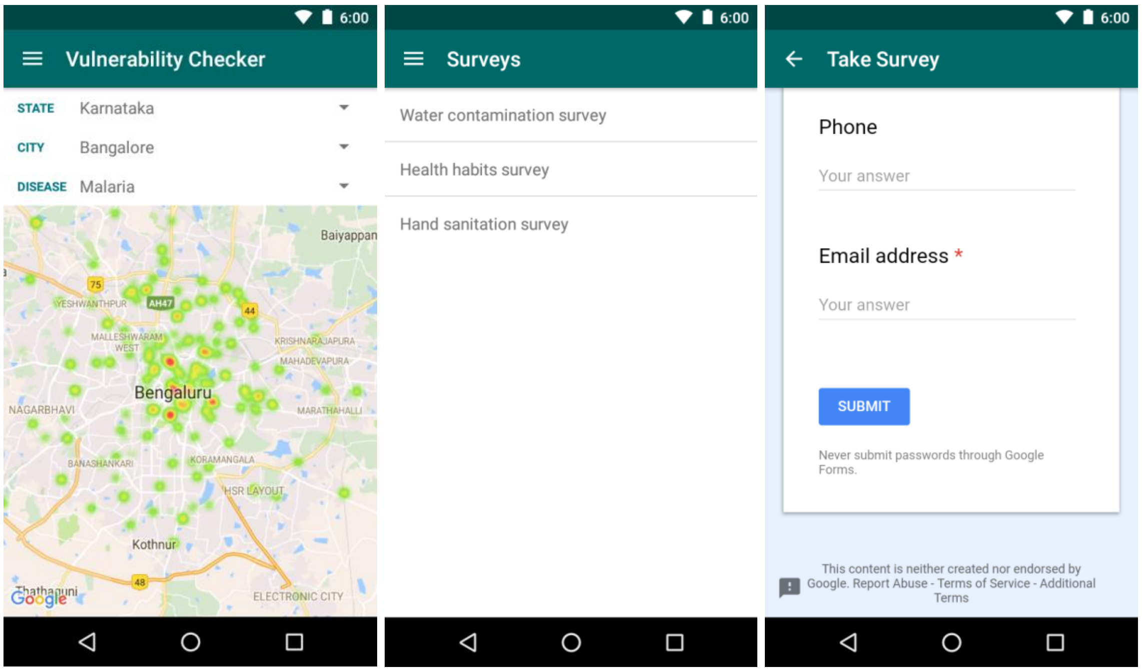


Fig. 6.1. Mobile application screens

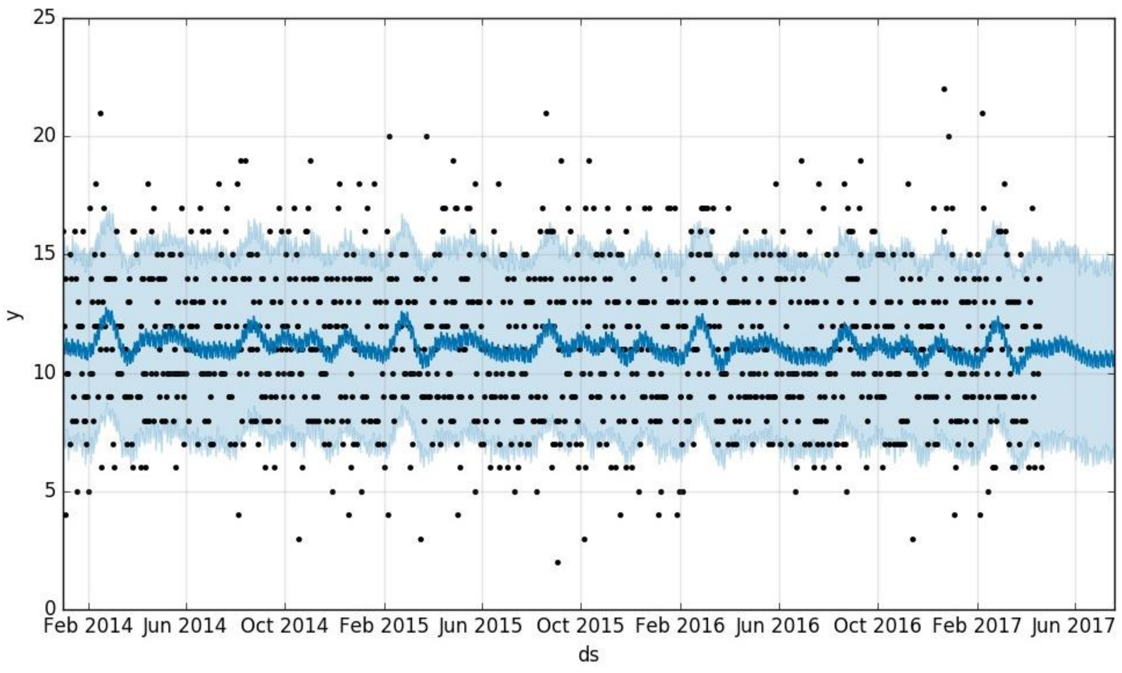


Fig. 6.2. Historical and predicted case volume for a disease at a hospital

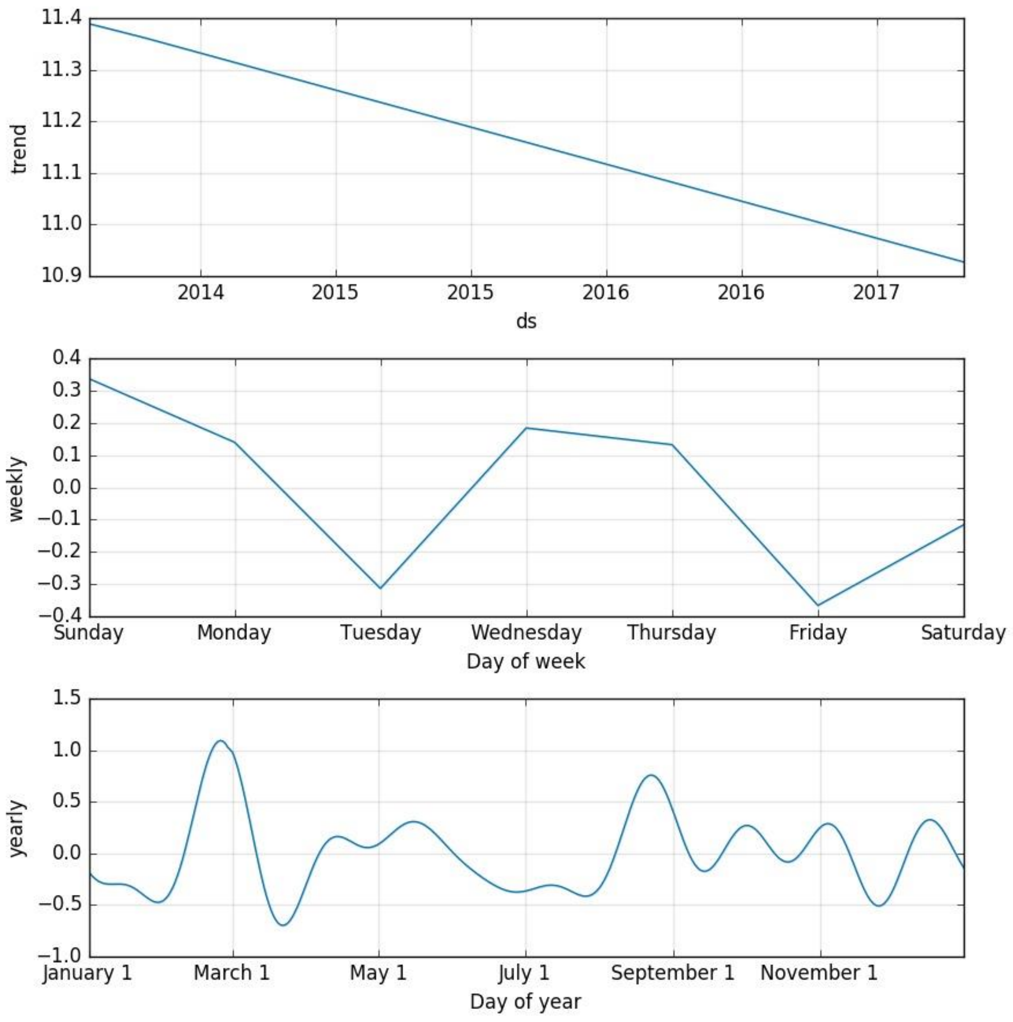


Fig. 6.3. Yearly, daily and monthly trends for a disease at a hospital

6.7. Summary

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Test techniques include, but are not limited to the process of executing a program or application with the intent of finding software bugs. Unit testing is a software testing method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. System testing is the phase in which the behavioral aspects of the system as a whole is tested. In this section a brief introduction to all the testing techniques are discussed and the unit test cases, integration test cases and system test case of the project are demonstrated.

7.1. Conclusion

The novel approach of using patient data and statistics to forecast the number of patients that are predicted to report of a specific disease has been outlined in this project. This project also help estimate the risk of disease spread in each area which is useful to alert both, the citizens and the municipal authorities. The project also pioneers in suggesting a platform for easy and direct communication between the Municipal Authority and the general public.

The three components work in tandem to solve the demand-supply problem of the hospital resources when an epidemic spreads unexpectedly. It also aims to fill the communication gap that usually exists between the municipal authorities and the citizens which prevents public participation and coordination in municipal awareness programs intended to educate people about good health habits.

7.2. Limitations of the System

- Though a state-of-the-art system has been proposed, the system is bound by the limitation of availability of patient data and statistics. Under data, following are the limitations:
    - Lack of prompt manual updation from hospitals
    - Lack of uniformity in the data provided to the server by the hospitals
- Since the system’s accuracy is untested due to lack of real world data, the effectiveness of the predictions cannot be categorically claimed.
- The system and the results are available only to those citizens who have access to a smartphone and internet.

7.3. Future Enhancements

In the future, the system needs to be improved on the following areas:

- Improving accuracy based on results from real world data
- Enhance usability by simplifying the UI further
- Add more features to the Information Hub.
  E.g., Connecting a citizen to a Municipal Authority representative through direct chat.

REFERENCES

1. *Peter Groves, Basel Kayyali, David Knott, Steve Van Kuiken, "The 'big data' Revolution in healthcare", McKinsey & Company, January 2013*
2. *Django documentation*: https://docs.djangoproject.com/en/1.10/
3. *NativeScript documentation*: http://docs.nativescript.org/
4. *PEP8*: https://www.python.org/dev/peps/pep-0008/
5. *Facebook Prophet*: https://research.fb.com/prophet-forecasting-at-scale/

TEAM INFORMATION

Name	Chaithra P	*Name*	Shreesha S
USN	1PE13IS401	*USN*	1PE13IS084
Email id	prakash22552@gmail.com	*Email id*	shreesha.suresh@gmail.com
Contact number	+91 96324 20781	*Contact number*	+91 96638 75790
Name	Nandini Prasanna B P	*Name*	Syed Amjad Ali
USN	1PE14IS405	*USN*	1PE13IS100
Email id	nandini1995prasanna@gmail.com	*Email id*	amjad.4@gmail.com
Contact number	+91 94492 31343	*Contact number*	+91 89043 62086