

**VISVESVARAYA TECHNOLOGICAL UNIVERSITY
BELAGAVI, KARNATAKA, INDIA**



A MINI-PROJECT REPORT

ON

“COVID19 STATUS MANAGEMENT SYSTEM”

Submitted in partial fulfillment of the requirement for the VI semester BE in

Information Science and Engineering

FS mini-project-17ISL68

Submitted by

**ARPITHA BHANDARI [1SG17IS011]
DEEPAK K [1SG17IS027]**

Under the guidance of

**Prof. CHAITHRA B M
Assistant Professor**



DEPARTMENT OF INFORMATION SCIENCE & ENGINEERING

SAPTHAGIRI COLLEGE OF ENGINEERING

Bengaluru-57

2019-20

SAPTHAGIRI COLLEGE OF ENGINEERING
14/5, Chikkasandra, Hesaraghatta Main Road, Bengaluru-560057

[AFFILIATED TO VISVESVARAYA TECHNOLOGICAL UNIVERSITY, BELAGAVI]



Department of Information Science & Engineering

CERTIFICATE

Certified that the project work entitled **“COVID19 STATUS MANAGEMENT SYSTEM”** carried out by **ARPITA BHANDARI [1SG17IS011], DEEPAK K [1SG17IS027]** bonafide students of 6th semester, department of **Information Science & Engineering** carried out at our college **Sapthagiri College of Engineering**, Bengaluru in partial fulfillment for the 6th Semester BE, FILE STRUCTURES Mini-Project-17ISL68 by **Visvesvaraya Technological University**, Belagavi during the year 2019-20. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements in respect of Project work prescribed for the said Degree.

Signature of the Guide
Chaithra B M
Assistant Professor

Signature of the HOD
Dr. H R. Ranganatha
Head of Dept

External Viva

Name of the Examiners

1.
2.

Signature of the Examiners with date

- 1.....
2.....

ABSTRACT

The objective of COVID19 STATUS MANAGEMENT SYSTEM is to automate the record of corona cases from various states of India. This system typically enables the government or private medical department to keep track of the corona cases in the nation. Covid19 status management system is used to keep track the records in a much simpler way. It allows easy management and tracking of various states in the country in fields such as active cases, death, recovered, negative cases on day to day basis. The case data and details are easily maintainable and accessible by every authenticate person . It helps to keep track, recognize and plan for reducing cases effectively . It is an effective system to help manage these activities in this pandemic time.

ACKNOWLEDGEMENT

Any achievement doesn't depend solely on the individual efforts but on the guidance, encouragement and co-operation of intellectuals, elders and friends. A number of personalities have helped us. We would like to take this opportunity to thank them all.

We would like to express our heart-felt gratitude to **Dr. ,H Ramakrishna** Principal, Sapthagiri College of Engineering, Bengaluru, for his help and inspiration during the tenure of the course.

It is great privilege to extend our deep sense of gratitude to **Dr. , H R Ranganatha** Head of the Department, Information Science and Engineering, Sapthagiri College of Engineering, Bengaluru, who patronized throughout our career, for his constant support and encouragement and for the facilities provided to carry out this work successfully.

We wish to express our sincere thanks to our guide Prof., Chaitra B M Assistant Professor, Department of Information Science and Engineering, Sapthagiri College of Engineering, Bengaluru for helping us throughout and guiding us from time to time. We also extend our sense of gratitude and sincere thanks to non-teaching staff members of Information Science and Engineering, Sapthagiri College of Engineering, Bengaluru for their views and encouraging ideas.

Finally, we also thank our family and friends for their co-operation and motivation.

ARPITA BHANDARI (1SG17IS011)

DEEPAK K (1SG17IS027)

TABLE OF CONTENTS

| CHAPTER NO. | CHAPTER NAME | PAGE NO |
|-------------|---|-----------|
| 1. | INTRODUCTION | 1 |
| 1.1 | Introduction to File Structures | 1 |
| 1.1.1 | History | 2 |
| 1.1.2 | About the File | 3 |
| 1.1.3 | Various Storage kind of Fields and records | 4 |
| 1.1.4 | Application of File Structure | 5 |
| 2. | SYSTEM ANALYSIS | 7 |
| 2.1 | Analysis of Application | 7 |
| 2.2 | Structure used to store Fields and records | 7 |
| 2.3 | Operation performed on File | 8 |
| 2.4 | File Seek Operation | 10 |
| 2.5 | Indexing Used | 10 |
| 3. | SYSTEM DESIGN | 11 |
| 3.1 | Design of Fields and Records | 11 |
| 3.2 | User Interface | 12 |
| 4. | IMPLEMENTATION | 15 |
| 4.1 | About C++ | 15 |
| 4.2 | Pseudocode | 16 |
| 4.2.1 | Insertion Modulo Pseudocode | 16 |
| 4.2.2 | Display Modulo Pseudocode | 16 |
| 4.2.3 | Deletion Modulo Pseudocode | 16 |
| 4.2.4 | Search Modulo Pseudocode | 17 |
| 4.2.5 | Modify Modulo Pseudocode | 17 |
| 4.2.6 | Indexing Pseudocode | 17 |
| 4.3 | Testing | 18 |
| 4.3.1 | Unit Testing | 18 |
| 4.3.1.1 | Record Insertion | 18 |
| 4.3.1.2 | Record Deletion | 19 |
| 4.3.2 | Integration Testing | 19 |
| 4.4 | Discussion of Results | 20 |
| 4.4.1 | Menu | 20 |

| | | |
|-----------|------------------------------|-----------|
| 4.4.2 | Insertion | 20 |
| 4.4.3 | Deletion | 21 |
| 4.4.4 | Modification | 22 |
| 4.4.5 | File Contents | 22 |
| 5. | CONCLUSION AND FUTURE | |
| | ENHANCEMENT | 23 |
| | REFERENCES | 24 |

LIST OF FIGURES

| FIGURE NO | NAME OF THE FIGURE | PAGE NO |
|------------------|---------------------------|----------------|
| 3.2.1 | Insertion of Records | 12 |
| 3.2.2 | Deletion of Records | 13 |
| 3.2.3 | Searching of Record | 13 |
| 3.2.4 | Modification of Records | 14 |
| 3.2.5 | Display of Records | 14 |
| 4.3.1.1 | Record Insertion | 18 |
| 4.3.1.2 | Record Deletion | 19 |
| 4.4.1 | Menu | 20 |
| 4.4.2 | Insertion | 20 |
| 4.4.3 | Deletion | 21 |
| 4.4.4 | Modification | 22 |
| 4.4.5 | File contents | 22 |

CHAPTER 1

INTRODUCTION

1.1. Introduction to File Structure

File Structures is the Organization of Data in Secondary Storage Device in such a way that minimizes the access time and the storage space. A File Structure is a combination of representations for data in files and of operations for accessing the data. File Structure allows applications to read, write and modify data. It also supports finding the data that matches some search criteria or reading through the data in some particular order.

The goal of File Structure is to get the information we need with one access to the disk. If it is not possible, then get the information with as few accesses as possible. Group information so that we are likely to get everything we need with only one trip of the disk. It is relatively easy to come up with File Structure designs that meet the general goals when the files never change. When files grow or shrink when information is added and deleted, it is much more difficult.

Goal of this course is with reference to time and space is to first minimize number of trips to the disk in order to get desired information. Ideally get what we need in one disk access or get it with a few disk access as possible. Secondly grouping related information so that we are likely to get everything we need with only one trip to the disk for example name, address, phone number, account balance.

Good File Structure design must have:

- Fast access to great capacity.
- Reduce the number of disk accesses.
- By collecting data into buffers, blocks or buckets.
- Manage growth by splitting these collections.

1.1.1. History

- History of File Structure design, In the beginning the file access was sequential, and the cost of access grew in direct proportional to the size of the file. So Indexes were added to files.
- Indexes made it possible to keep a list of keys and pointers in a smaller file that could be searched more quickly
- Simple indexes become difficult to manage for dynamic files in which the set of keys changes. Hence Tree Structures were introduced.
- Trees grew unevenly as records were added and deleted, resulting in long searches requiring multiple disk accesses to find a record. Hence an elegant, self-adjusting binary tree structure called an AVL Tree was developed for data in memory.
- Even with a balanced binary tree, dozens of accesses were required to find a record in moderate sized files.
- A method was needed to keep a tree balanced when each node of the tree was not a single record as in a binary tree, but a file block containing hundreds of records. Hence B-trees were introduced.
- AVL trees grow from top down as records are added, B-trees grow from the bottom up.
- B-trees provided excellent access performance but, a file could not be accessed sequentially with efficiency.
- The above problem was solved using B+ tree which is a combination of a B-tree and a sequential linked list added at the bottom level of the B-tree.
- To further reduce the number of disk accesses, Hashing was introduced for files that do not change size greatly overtime.

1.1.2. About the File

A File is an object on a computer that stores data, information, settings, or commands used with a computer program. In a graphical user interface such as Microsoft Windows, files display as icons that relate to the program that opens the file. For example, the picture is an icon associated with adobe acrobat PDF files, if the file was on your computer, double clicking the icon in Windows would open that file in adobe acrobat or the PDF reader installed on the computer. A File is created using a software program on the computer. For example, to create a text file we use text editor, to create an image file we use an image editor, and to create a document we use a word processor.

Files are not made for just reading the contents, we can also perform some operations on the Files.

- **Read operation:** Meant to Read the information which is stored into the files.
- **Write operation:** For inserting some new contents into a file.
- **Rename or Change the Name of the file.**
- **Copy the file from one location to the other.**
- **Sorting or arrange the contents of the file.**
- **Move or cut the file from one place to another.**
- **Delete a file.**
- **Execute Means to Run Means File Display Output.**

We can also link a file with any other File. These are also called Symbolic Links; in the symbolic links all the files are linked by using some text Alias.

1.1.3. Various storage kinds of fields and records

Field Structures:

There are many ways of adding structure to files to maintain the identity of fields.

Four most common methods follow:

- Method 1: Force the fields into a predictable length. Fix the length of fields. The fields of the file vary in length to make the fields fixed length we have to predict lengths.
- Method 2: Begin each field with a length indicator, store the field length just ahead of the field.
- Method 3: Place a delimiter at the end of each field to separate it from the next field.
- Method 4: Use a “keyword = Value” expression to identify each field and its contents.

Record Structures:

A record can be defined as a set of fields that belong together when the file is viewed in terms of a higher level organization. Five most common methods follow:

- Method 1: Make the records be a predictable number of bytes in length.
- Method 2: Make the record be a predictable number of fields in length.
- Method 3: Begin each record with a length indicator consisting of a count of the number of bytes that the record contains.
- Method 4: Use a second file to keep track of the beginning byte address for each record.
- Method 5: Place a delimiter at the end of each record to separate it from the next record.

1.1.4. Application of File Structure

Amaze file manager:

Amaze File Manager is a newer app comparatively speaking and it's a pretty good. It's open source and focuses on as lighter experience for those who just need to do some light file browsing. It features Material Design, SMB file sharing, a built-in app manager to uninstall apps, root explorer, and more. It manages to include the most important stuff without feeling bloated. It's free to download and use with optional in-app purchases in case you want to help fund development.

Asus file manager:

It's not every day we see on OEM app make an app list, but File Manager by ASUS is actually really good. It's compatible with most devices, even non-ASUS ones. You'll also get clean, simple interface with LAN and SMB support, cloud storage support, support for various types of files, archiving support, and more. It's entirely free with no in-app purchases and provides a great experience for a simple file browser. About the only negative part is the lack of root access.

ES file explorer pro:

ES File Explorer has been around as long as most Android nerds can remember and comes with pretty much every feature you can ask for in a file browser. A while back, it was purchased by another company. Since then, things haven't gone well. The free version of the app, while very capable, now has a ton of added bloatware that not only doesn't add to the experience, but activity subtracts from it. Thankfully, the pro version of the app doesn't have these features.

File manager:

File Manager is blandly named, but it's actually quite good. It's a newer file manager app that gives you one of the best sets of features without adding too much bloat. You'll get basic file management features along with cloud storage features, NAS support, and more. You can even browse your installed apps, music, and video with this and the player isn't half bad. Perhaps the best part is that the app is free with no in-app purchases and advertising.

MK explorer:

MK Explorer is another newer file manager option. It's a simple option that doesn't have a whole lot of flair. That is extremely preferable if you really just want something simple. It features a Material Design interface, the basic file management features (copy, paste, delete, SD Card support for Lollipop), and root access. There are also support for 20 languages and it has a built-in text editor, gallery, and music player. It doesn't have anything like cloud storage or network storage support, but that's not really what it's for. It's a good, cheap option.

X-Plore file manager:

X-Plore File Manager is one of the more unique options on the list. It's a forced dual-pane app which means you'll be managing two windows at once pretty much all the time. This is kind of cool if you're copy/pasting between folders or need to move files quickly. It also comes with support for various types of files, cloud storage, network storage, a built-in hex editor, root support, and plenty of other features. You can even view APK files as zips if you're into that kind of thing.

CHAPTER 2

SYSTEM ANALYSIS

2.1. Analysis of application

COVID19 STATUS MANAGEMENT SYSTEM is designed to help in managing and keeping track of corona cases across various states of a nation. It allows easy management and tracking of cases in fields such as number of active cases, negative cases, recovered, number of death cases. This project can also be modified to suit any nations database or even for private companies doing research on this disease. An administrator has the overall control of this system that allows him to add, modify and remove the cases of every state as per the requirement on daily basis. He can also add or delete information of states. It helps to keep track, recognize and used as data for future prediction. It is an effective system to help manage these activities in this pandemic situation. Thus the whole system is built to be user-friendly as well as help the department to recognize more more affected states and help in the projects working to reduce this cases growth. The techniques and the software used are also easily understandable for any user. Without any complications, the user can upload records into the database without fully understanding the system as to how it works. The technique used also simplifies the access and search function of the system. Hence the user can easily work it and access the records without difficulty.

2.2. Structure used to store the Fields and Records

- A field is an item of stored data. A field could be a name, an address, a description etc.
- A record is the collection of fields that relate to a single entity.
- For example here we have created a class called variable which contains a structure called state which has fields like Name, Confirm, Suspect, death, and recover.

```
class variable
{
    class state
    {
        char name[50];
        int confirm;
        int suspect;
        int death;
        int recover;
    };
};
```

2.3 Operations Performed On File

File Create Operation

- The file is created with no data.
- The file create operations is the first step of the file.
- Without creating any file, there is no any operation can be performed.

Opening a File

- The process must open the file before using it.

Insertion

- In file insertion we are inserting the records using indexing.
- Every field in the record is separated by using '|' as a delimiter and '\$' is used to indicate the end of a record.
- Each record is created by using the read_data() function that prompts the user to enter the required details.

Deletion

- In deletion of a record a team's record that is existing in the *state.txt* is deleted by using the delete_team() function.
- This function obtains the position of the record from the *state.txt* file by searching for the index of the record using the email id as a key.
- The search_index function is internally called by this function to and it returns the position of the required record.
- After receiving the position the delete_team() the record is deleted by using the concept of reclaiming the free spaces by the program.

File Closing operation

- The file must be closed to free up the internal table space, when all the accesses are finished and the attributes and the disk addresses are no longer needed.

File Read Operation

- The file read operation is performed just to read the data that are stored in the file.

File Write Operation

- The file write operation is used to write the data to the file, gain, generally at the cursor position.

2.4 File Seek Operation

- For random access files, a method is needed to justify to specify from where to take the data. Therefore, the file seek operation performs this task.

2.5 Indexing Used

An Index is a tool for finding records in a file. It consists of:

- Key field on which the index is searched.
- Reference field that tells where to find the data file record associated with particular key.

In this project primary indexing as well as secondary indexing is implemented as follows:

- This project allows to organize the file as variable length record with a size field preceding each record. The fields within each record are also of variable – length but are separated by delimiters.
- Here, it searches a particular record based on the **primary key**.
- In this program the concept of the **secondary index** is also used as the search are based on this concept only like if we ask for the modification or deletion it is done on the basis of the secondary index.

CHAPTER 3

SYSTEM DESIGN

3.1 Design of the Fields and Records

Covid19 status management system is a detailed system and a menu driven application that helps to manage the achievements of the students. It maintains the records of all the students' achievement in different fields.

COVID19 STATUS MANAGEMENT SYSTEM

1. COVID19 STATUS MANAGEMENT SYSTEM .
2. EXIT.

1. **Student database management:**

1. Add : Insert details of cases in a state.
2. Show : Display the cases in state details.
3. Modify : Modify the required field in the state database.
4. Search : Search the information about the required state.
5. Delete : Delete the state record.
6. Exit

2. **Exit:**

Exits the current screen/application.

3.2 User Interface

- The junction between a user and a computer program. An interface is a set of commands or menus through which a user communicates with a program.
- A menu driven interface is one in which you select command choices from various menus displayed on the screen.

3.2.1 Insertion of a Record

- A new record is being inserted into the *state.txt*.

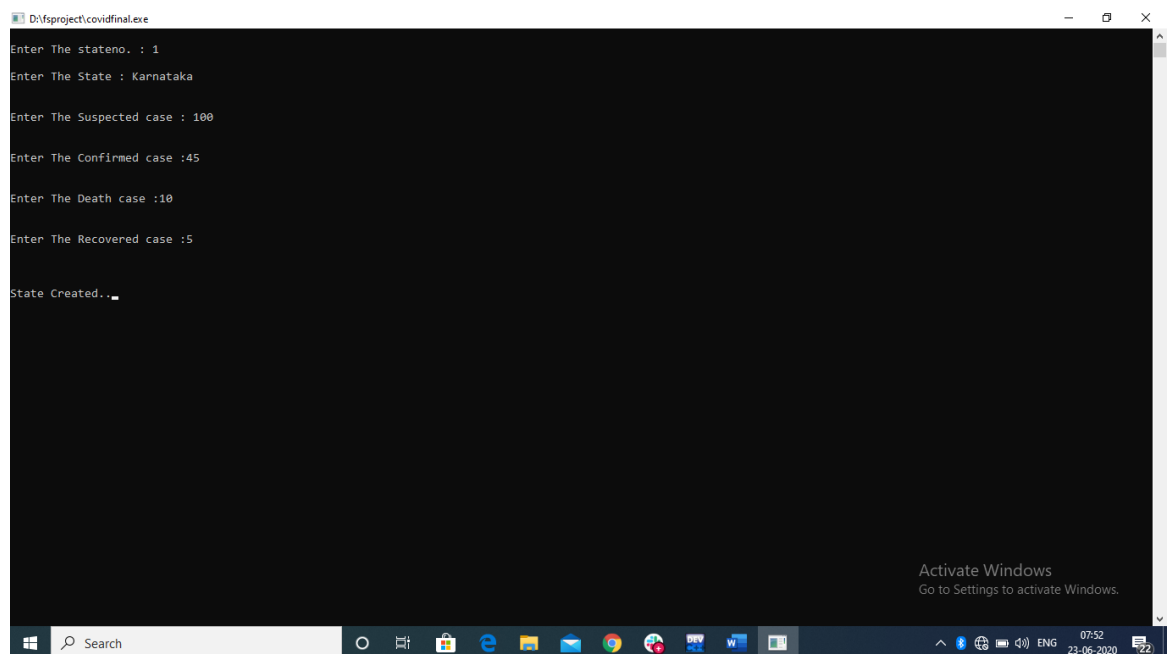


Fig 3.2.1 Insertion of a record

3.2.2 Deletion of a Record

- A record is deleted from the file *state.txt*.

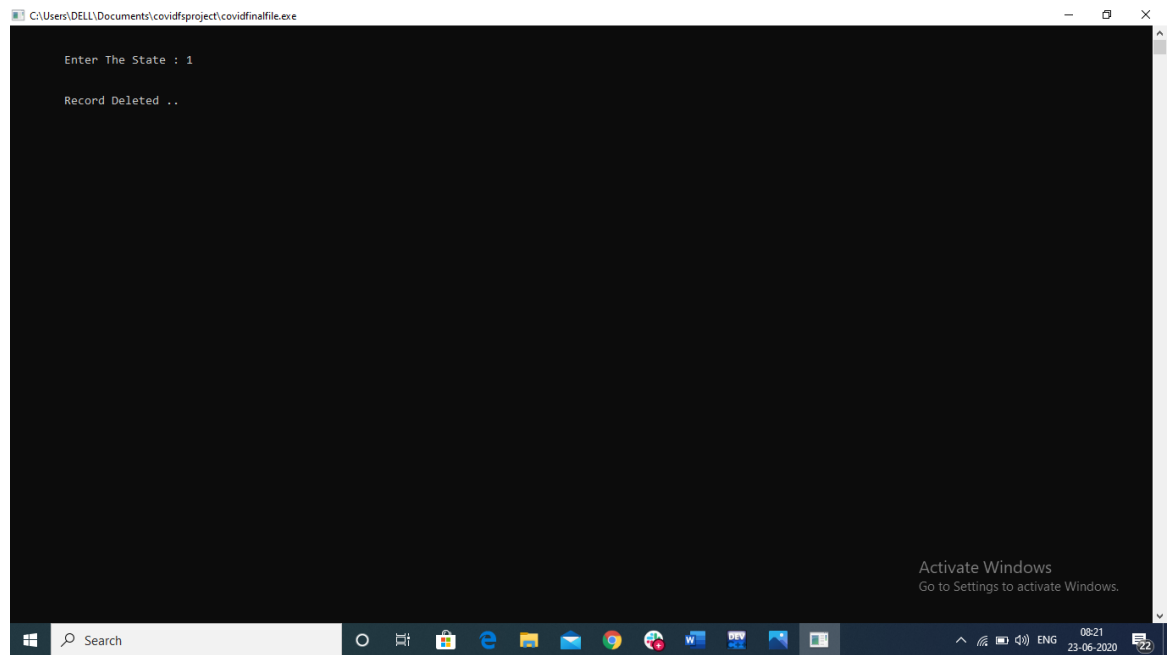


Fig 3.2.2 Record Deletion.

3.2.3 Searching of a Record

- A record is searched from the file *state.txt*.

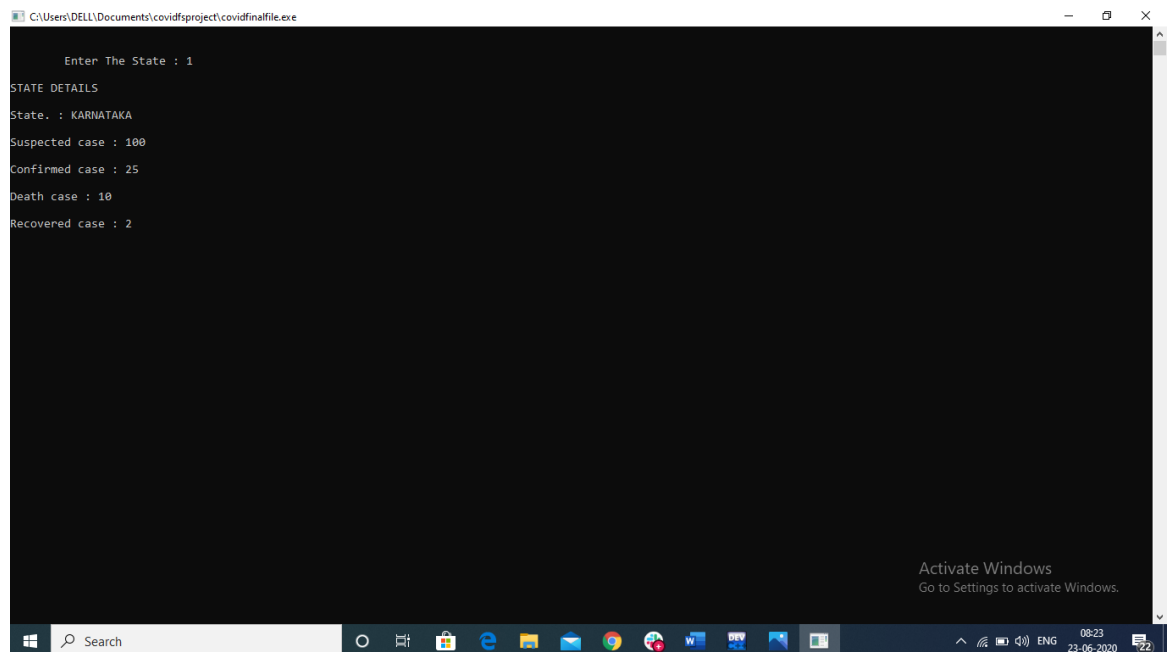


Fig 3.2.3 Record searching

3.2.4 Modify a record

- A record is modified from the file *state.txt*.

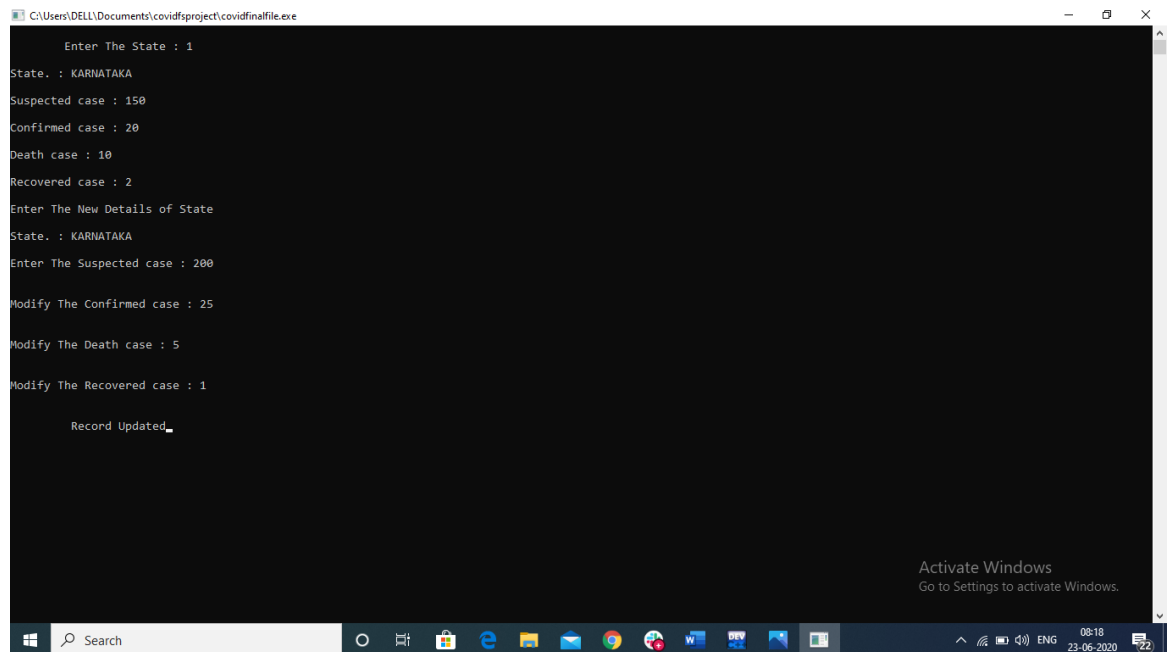


Fig 3.2.4 process of Record modification

3.2.5 Display a Record

- Details is displayed from *state.txt* file

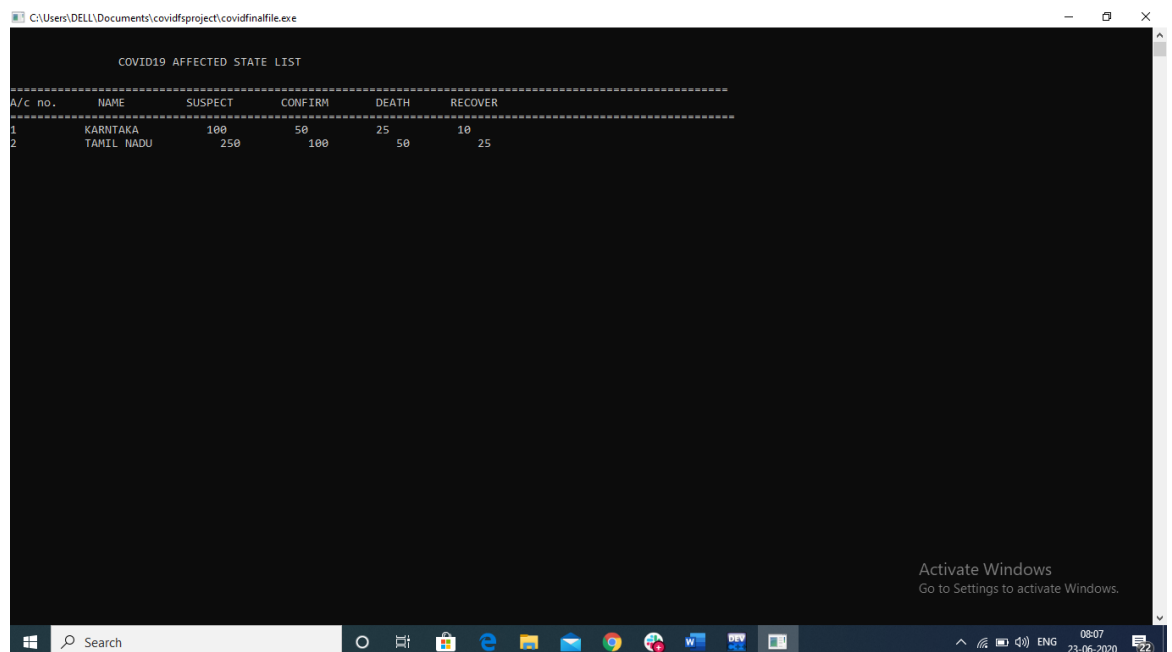


Fig 3.2.5 Record Display

CHAPTER 4

IMPLEMENTATION

4.1 About C++

- C++ is a general-purpose programming language. It has imperative, object-oriented and generic programming features, while also providing facilities for low-level memory manipulation.
- It was designed with a bias toward system programming and embedded, resource-constrained and large systems, with performance, efficiency and flexibility of use as its design highlights.
- C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications, including desktop applications, servers (e.g. e-commerce, web search or SQL servers), and performance-critical applications (e.g. telephone switches or space probes).
- C++ is a compiled language, with implementations of it available on many platforms. Many vendors provide C++ compilers, including the Free Software Foundation, Microsoft, Intel, and IBM.
- C++ is standardized by the International Organization for Standardization (ISO), with the latest standard version ratified and published by ISO in December 2017 as ISO/IEC 14882:2017 (informally known as C++17). The current C++17 standard supersedes these with new features and an enlarged standard library. Before the initial standardization in 1998, C++ was developed by Bjarne Stroustrup at Bell Labs since 1979, as an extension of the C language as he wanted an efficient and flexible language similar to C, which also provided high-level features for program organization.
- Many other programming languages have been influenced by C++, including C#, D, Java, and newer versions of C.

4.2 Pseudo Code

- Pseudo code is a simple way of writing programming code in English. Pseudo code is not actual programming languages.
- It uses short phrases to write code for programs before you actually create it in a specific language.
- Once the functionality of the program is known, then one can use pseudo code to create statements to achieve the required results for your program.

4.2.1 Insertion Module Pseudo code

Step1: Input the required variables into *state.txt* .

Step2: open the file and write the contents into *state.txt* until the condition is satisfied

Step3: Close the file.

4.2.2 Display Module Pseudo code

Step1: Open the file *state.txt* .

Step2: Unpack the contents.

Step3: Display the contents of *state.txt* .

Step4: Close the file.

4.2.3 Deletion Module Pseudo code

Step1: Get the position using search function.

Step2: If position is greater than zero

Then Open the file.

Go to position.

Delete the contents at that particular position.

Set flag to 1.

If flag is set to 1

then Return 1.

Else Return 0.

Step3: End that module.

4.2.4 Search Module Pseudo code

Step1: Open the file.

Step 2: While file is not equal to end of file

Erase the contents in buffer.

Get the position from file.

Copy the contents from file to buffer.

If key is equal to name

Display the contents of buffer.

Return position.

4.2.5 Modify Module Pseudo code

Step1: Open the file.

Step 2: While file is not equal to end of file

If key is equal to usn

Get the position from index file

Get the address from the file

Point to that address of the file

Remove the contents of that record

Modify any required field.

Return position.

4.2.6 Indexing Pseudo code

Step 1: Start the program.

Step 2: Open the file *state.txt*

Step 3: While file is not equal to end of file

Get the position from file.

Erase the contents from buffer.

Copy the contents from file to buffer.

If the contents of buffer are not deleted

then If buffer is empty

then **break**.

Extract the student

Get the address of the student

Step 4: Close the file.

Step 5: Erase the contents of buffer.

4.3 Testing

4.3.1 Unit Testing

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 whether they are fit for use.

4.3.1.1. Insert

- The details of a state and its cases are entered into the file.
- These details are stored into the file *state.txt* using a buffer.
- A message is displayed to indicate to the user the successful creation of the account.
- This module has been tested successfully.

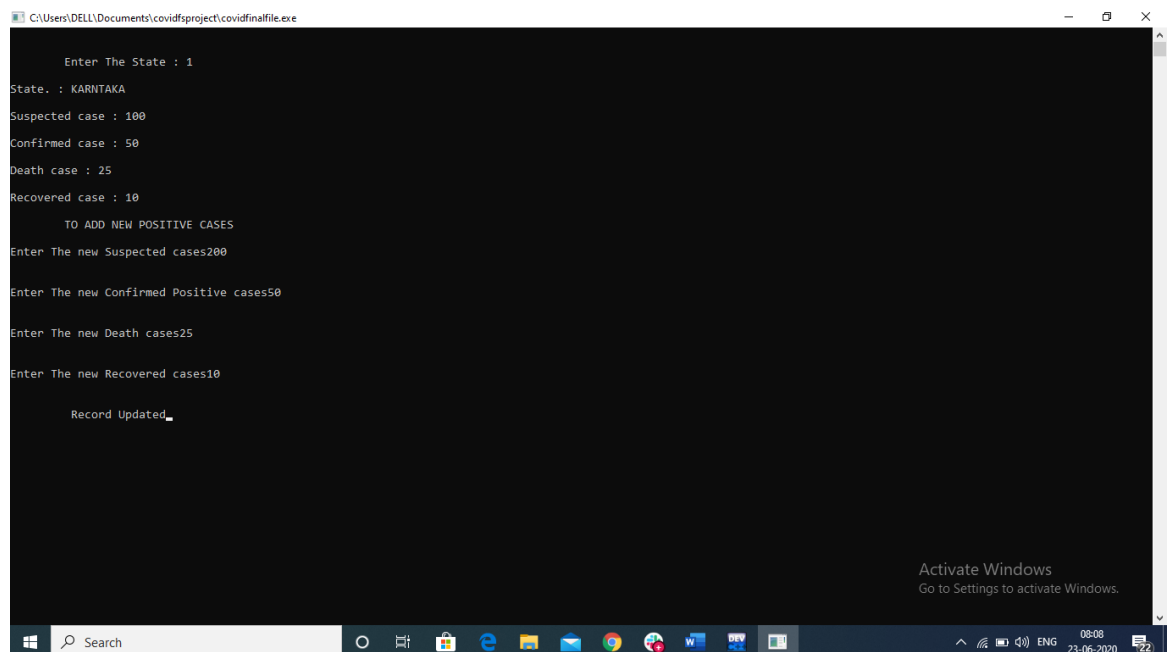


Figure 4.3.1.1 Record Insertion

4.3.1.2. Delete

- On choosing delete the state details from the site it is deleted and he/she needs to make a new registration in database.
- This module has been tested successfully.

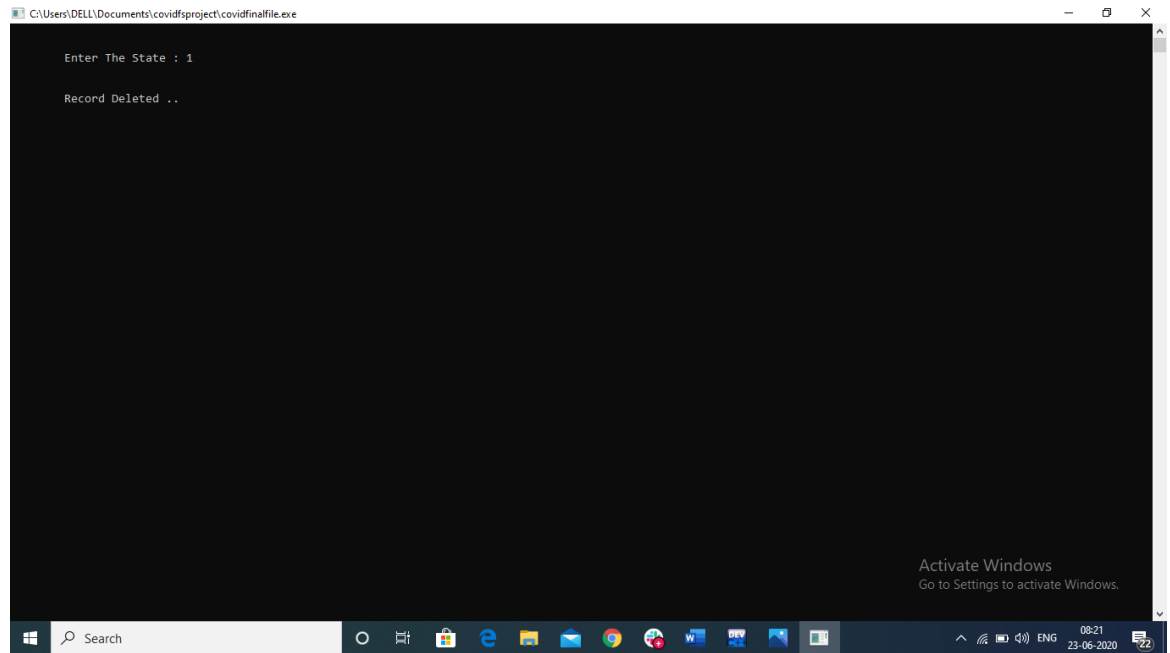


Figure 4.3.1.2 Record Deletion

4.3.2 Integration Testing

Integration testing is a phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. When state has been made for the users, then the user can list all the state and its details and can modify the details of the cases also.

4.4 Discussion of Results

4.4.1 Menu

After a new account is created it can be managed efficiently.

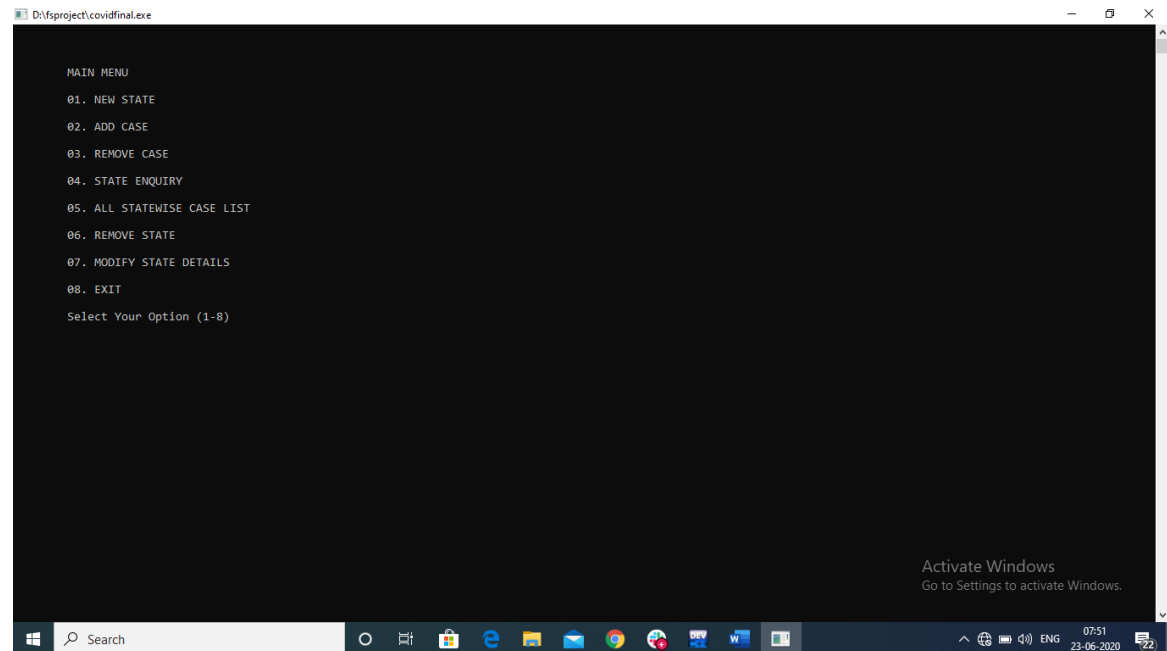


Figure 4.4.1 Menu

4.2 Insertion

The *state.txt* file after insertion of records.

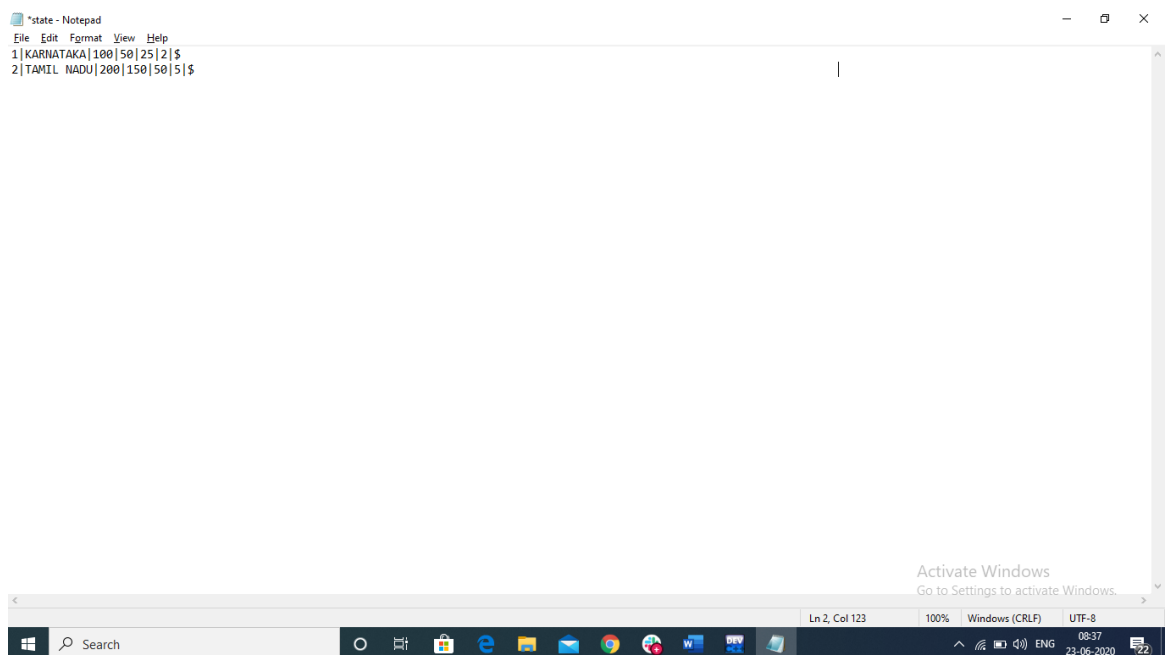


Figure 4.4.2 Insertion

4.4.3 Deletion

The *state.txt* file before deletion of a record.

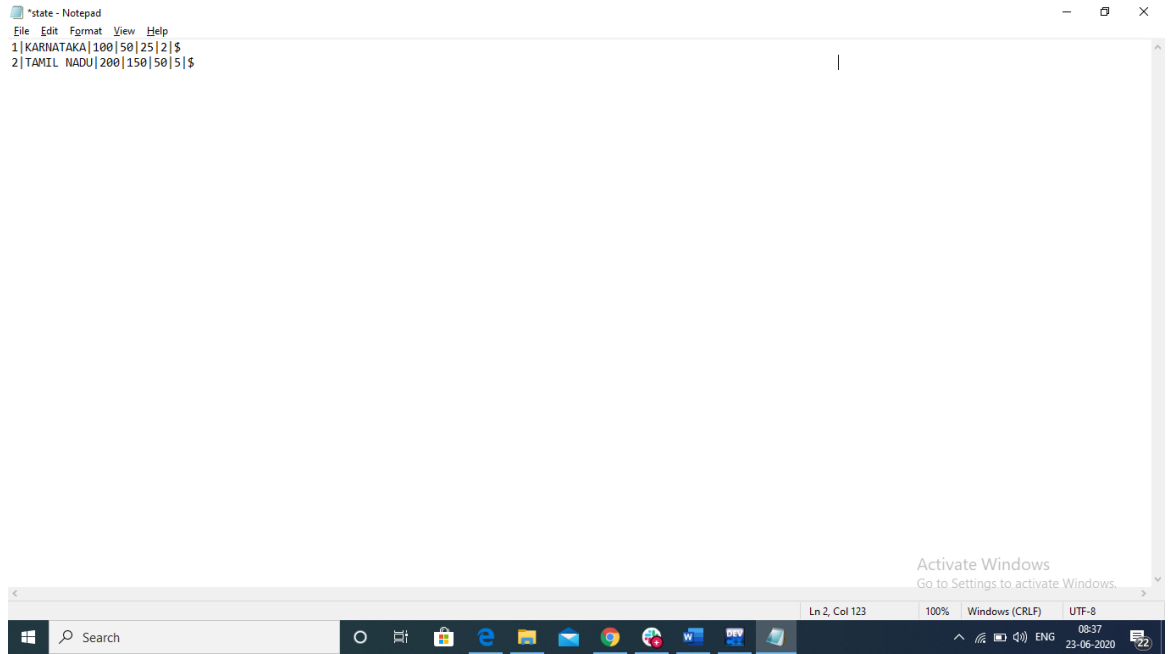


Figure 4.4.3.1 Before Deletion

The *state.txt* file after deletion of a record.

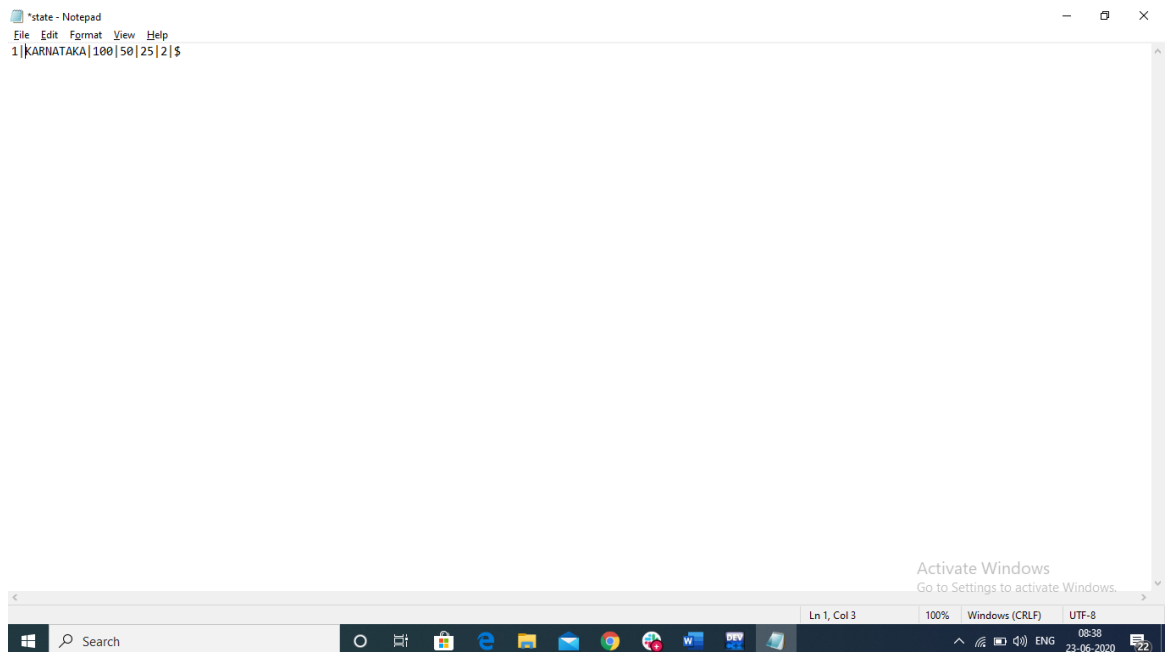
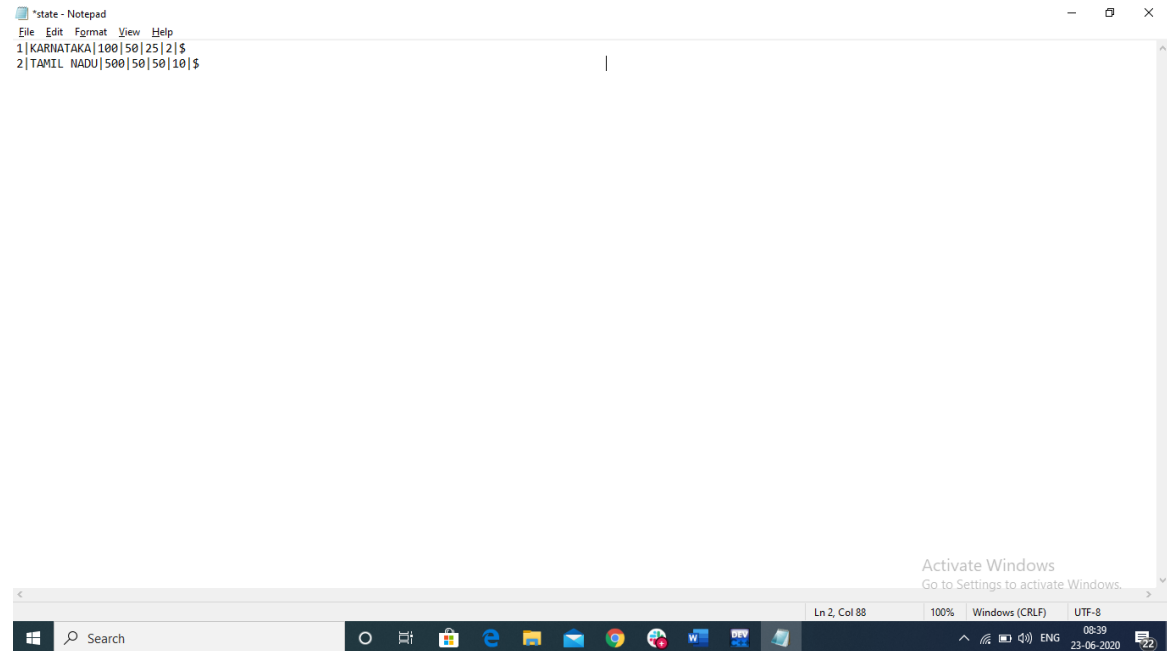


Figure 4.3.3.2 After Deletion

4.4.4 Modification

The *state.txt* file after modification.

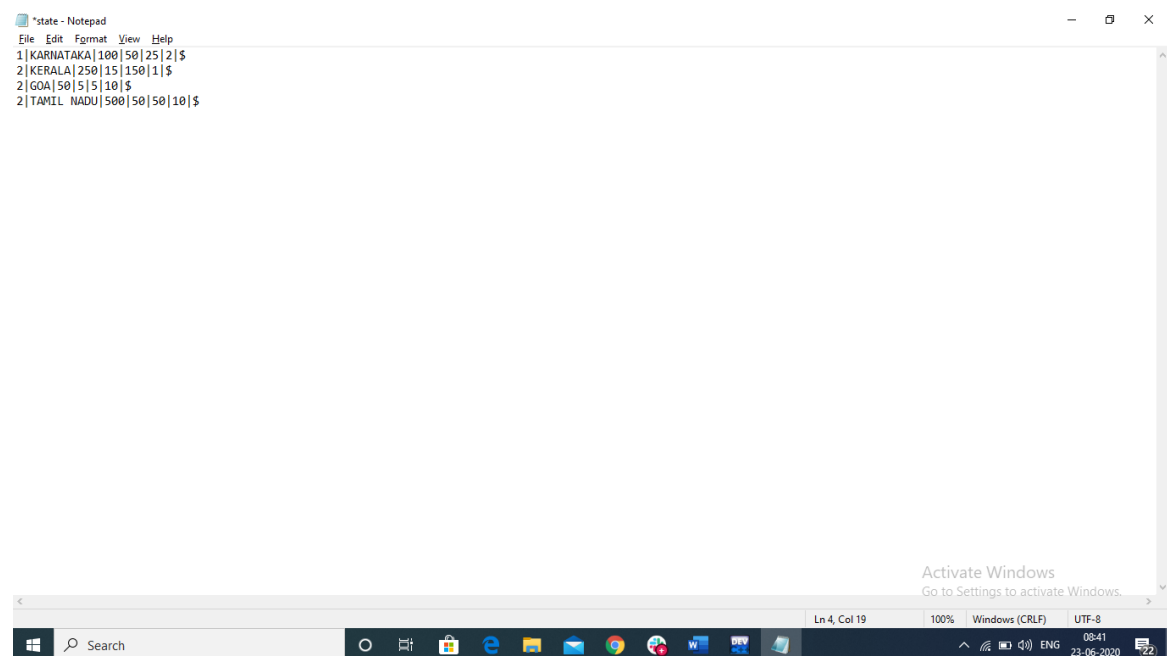


```

*state - Notepad
File Edit Format View Help
1|KARNATAKA|100|50|25|2|$
2|TAMIL NADU|500|50|50|10|$
    
```

Fig 4.4.4 After Modification.

4.4.5 File Contents



```

*state - Notepad
File Edit Format View Help
1|KARNATAKA|100|50|25|2|$
2|KERALA|250|15|150|1|$
2|GOA|50|5|5|10|$
2|TAMIL NADU|500|50|50|10|$
    
```

Fig 4.4.5 *state.txt*

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENTS

The Covid19 status management system has been developed to override the problems prevailing in the practicing manual system. This software is supported to eliminate and in some cases reduce the hardships faced by this existing system. Moreover this system is designed for the particular need of the government or institution to carry out operations in a smooth and effective manner. The application is reduced as much as possible to avoid errors while entering the data. It also provides error message while entering invalid data. No formal knowledge is needed for the user to use this system. Thus by this all it proves it is user-friendly. The developed Covid19 status management system is more versatile, flexible and updating can be done easily.

The Covid19 status management system can be upgraded with more advanced software that support search engines and also work along with the internet connection. It can also be dynamically linked to the internet connection. The records can be viewed easily and more upgrades can be done in terms of the achievement details and the students who have achieved it. It can be connected with other departments within the institution with servers and the records can be stored in the servers along with protection to it.

REFERENCES

- www.google.com
- www.stackoverflow.com
- www.youtube.com
- File Structures –An Object Oriented Approach with C++ By Michael J.Folk, Bill Zoellick, Greg Riccardi
- K.R. Venugopal, K.G. Srinivas, P.M. Krishnaraj: File Structures Using C++, Tata McGraw-Hill, 2008.
- Dev-C++ on SourceForge.net

