1. INTRODUCTION
   1. **Introduction to project**

**What is PED?**

PED is an acronym for PERSONAL ENGLISH DICTIONARY according to us, it is an application for both desktop and mobile users where a user can search for definitions of various words, define new words, redefine existing words if they find much better fitting definitions and even delete a word if they find that it’s not necessary and all these features are independent on the internet.

**Why we are interested in doing this project?**

Generally, a dictionary is a daily usage, application for any user. As language plays a very important role in this communicative world, one should maintain a dictionary to know about the word’s definition that he/she finds difficult to understand. We agree that there are tons of applications that provide such a service, but none are perfect according to our observation as we mentioned in our abstract most of the dictionary applications are restricted to service over the internet, even some applications are providing service offline but lacking features like modifiability and expandability. So by taking all of this into consideration, we thought that this work allows us to make a good use of our knowledge that we had gained till now through our technical education to make this world a better place.

**What are our project objectives?**

Following are the series of activities or the objectives of our project that we want to achieve which are sorted in a priority based order

* As we observed most of the people are using Oxford as their standard dictionary so we thought to gather the dictionary data that is required for our project from Oxford through web scraping.
* As the time is limited and we have to perform a lot of tasks so we can’t waste time on database creation by hardcoding each and every query so we thought to automate the database creation process programmatically i.e., to develop a code that will just take the data that we got through web scraping as input and produce a database as output.
* We want to spend maximum time to produce similar, simple and efficient enough UI (User Interface Design) and UX (User Experience Design) for both desktop and mobile users by making use of prototype testing methodology to make sure that our application will be enhancing the performance of its user in this communicative world.
* Last thing that we want to achieve is to design our application in such a way that our application users will be able to share any updates provided to our application’s database among themselves.
  1. **Problem Definition**

Just for word processing there is no need of a server because these days’ electronic devices are being designed with sufficient enough resources which will be more than enough to maintain a dictionary.

Modifiability is really a worthy feature and comes handy because if a user can’t understand a word’s definition he/she will try to find that word’s definition from some other dictionary and then he/she will be confused to decide which is best to use since it’s really space consuming to maintain all the available dictionary applications they will end up with confusion. Even though this feature has a big advantage, none of the dictionary applications are providing such a feature.

Expandability is another worthy feature which allows users to define new word in their dictionary as soon as they find one so that they need not wait till the company which is providing the application as a service to add it. But none of the dictionary application are providing this feature too.

A problem affects the users who use the application that’s causing that problem. If a user is using an application which provides service only over internet he/she will not get the service in remote areas or the areas where internet is not available, if a user is using an application which does not provide features like modifiability and expandability it depends on the situations faced by the user if he/she can’t understand the definition provided then they need to go and search for another application which provides a definition that they can understand and finally they will end up in a confusion state to decide which application is best to use where as if a user finds a new word he/she will not be able to add that to his dictionary due to lack of expandability.

* 1. **Solution for Problem Definition**

To perform word processing offline we need an embedded database like SQLite so if the database is within the device there will be no need of a server hence the problem caused due to service over internet will be solved by this but here rises a problem if the database is quite big the application might get struck so to overcome that we need to use some kind of object oriented feature like multithreading to show some sort of loading animation when the backend process goes on and quit the loading animation as soon as the backend process gets completed successfully and proceed with an operation of users choice.

Modifiability can be provided by giving some privileges to the application users which allows modification of a word ‘s definition in the database. But this will rise a new problem like one user may choose to have some definition and the other may have something else so if both are making use of a same database instance then it will create deadlocks in the database or collisions in the database or even if everything goes well one user may not be satisfied with other user’s definition so to solve this we need to make sure that each and every user is having a separate and unique database instance.

Expandability can be provided by giving data insertion privileges to the application user so that the user who finds a new word will add it to his/her dictionary database instance. As discussed above in the case of modifiability this will also rise that same problem so to solve this we need to make sure that each and every user is having a separate and unique database instance.

* 1. **Process Design**

Design UI, UX and integrate everything

create a database

Correct error prone code and re-design

Analyse the data collected through web scraping if no errors are found

Perform web scraping on oxford site to collect words definitions

Get a list of existing words from internet

No

Yes

Test working if no errors are found

If

No

Yes

Publish Application

1. LITERATURE REVIEW

* English words collection:

To perform web scraping we need a list of words to find their definitions from Oxford site the below link is the place where we had found that list.

<https://github.com/dwyl/english-words>

* Oxford’s online dictionary:

To perform web scraping we need a website from which we can get a webpage of our choice through URL.

<https://en.oxforddictionaries.com/>

* Web Scraping:

Since the data we want to create an offline database can only be accessed as a part of a web page we need to use a technique called web scraping to get the data from the web page into a format that we can work with. The below given link is the place where we had found a working tutorial of web scraping using python 3, requests (A Python Library to make a get request to a web server to download the HTML content of a web page) and Beautiful Soup (A Python Library to parse a document)

<https://www.dataquest.io/blog/web-scraping-tutorial-python/>

* Working with database:

To work with a database programmatically we need a programming language and a driver the below link is the place where we had found a working tutorial to work with a SQLite database using java for desktop environment by importing an external jar file into our working environment.

<https://www.tutorialspoint.com/sqlite/sqlite_java.htm>

Normally most of the people use SQLite open helper but since it’s not suitable for our requirement we choose to work with SQLite Asset helper the below link is the place where we had found class named SQLite Asset Helper which provides developers with a simple way to ship their Android app with an existing SQLite database (which may be pre-populated with data) and to manage its initial creation and any upgrades required with subsequent version releases.

<https://github.com/jgilfelt/android-sqlite-asset-helper>

Working with this class is similar to working with SQLite Open Helper the below link is the place where we had found a working tutorial to work with a SQLite database.

<https://www.tutorialspoint.com/android/android_sqlite_database.htm>

* Loading Animation:

Since our application deals with a large data set we need to show some kind of loading animation while a heavy backend work is taking place so we created a few PNG files in photo shop and the below link is the place where we were able to convert all those files into a GIF.

<http://gifmaker.me/>

In desktop environment we are able to load a gif directly into an image view using JavaFX but in android we are not able to load so we are making use of an API which is available through the below link

<https://github.com/koral--/android-gif-drawable>

* Thread Synchronization:

To show loading animation and perform a backend task we need to work with two different threads since our application is GUI (Graphical User Interface) based we can’t perform a backend task on UI thread so we need to create a new worker thread to work with the backend process like handling database tasks etc. The below link is the place where we had found a working tutorial which demonstrates how to synchronize two threads using java.

<https://www.javatpoint.com/synchronized-block-example>

* UI design:

For android we can build UI by drag and drop but there is no proper facility in any IDE for desktop so we found an application that helps us to build UI for desktop just through drag and drop from the link given below.

<http://gluonhq.com/products/scene-builder/>

* Desktop UI Elements:

Since we want to design a similar UI for both desktop and mobile we need the UI elements that are available for mobile in desktop too so we had found them in the link given below which can be used with java FX.

<http://jfoenix.com/>

* Auto fill suggestions for textbox:

Auto fill is a common feature found in most of the online dictionary sites so we thought to provide that feature in our application though its offline. We found AutoCompleteTextView for android which will satisfy needs for mobile application but there is no feature provided by java FX for desktop application so made use of this open source project which is available through the link given below to simulate autocomplete text view in our desktop application.

<http://fxexperience.com/controlsfx/>

* Text to speech:

We are making use of text to speech since we want to provide the basic pronunciation of each and every word that is present in the database of our application in android we have an inbuilt class to do that but in java there is no such inbuilt class for desktop environment so we made use of an open source project known as Mary TTS which provides a custom Speech synthesis interface which is available through the link given below

<http://mary.dfki.de/index.html>

* To access file system in Android:

For desktop environment we are able to access the file system using native java libraries but in android we have no such feature so we made use of an API which is available through the below link

<https://github.com/nbsp-team/MaterialFilePicker>

1. SYSTEM ANALYSIS
   1. **Existing System**

Dictionary services are mostly provided over internet, i.e., a user with an electronic device which can access the internet can search for a word and find its details and users will not be allowed to modify or expand the data provided.

**Limitations of Existing System**

Every Individual has their own electronic gadgets with high cost and good features, which are over dependent on the internet. If the user moved to any remote areas, he/she cannot access the services provided over the internet, which means no internet results no solution. And also takes more effort to get a response data from the server. At some times a user may not get satisfied with the response provided by some site “A”, then he/she should search that word in another site B/C/… After a long time if the user forgets the meaning of that word he/she should start from “A” again. These are the limitations of existing system

* 1. **Proposed System**

PED will be an offline Dictionary service for Desktop and Android users. It will be useful anywhere and anytime so that it even satisfies travelers too. PED do not need to wait for server response over internet, as the device itself has the response in its respective storage. If the user is not satisfied with the response given by the PED, then the user can customize that response as he/she wants. So for the next time PED satisfies that particular user for sure.

**Advantages of Proposed System**

🡪 Word search with suggestions and auto fill.

🡪 Can add new words.

🡪 Can update the existing words details.

🡪 Deletion of unwanted words is also possible.

🡪 Bulk insertion of ‘n’ number of words is even possible.

🡪 Uses an embedded database.

🡪 Comes with predefined words based on the Oxford which are collected through web scraping.

* 1. **Analysis Model**

We had chosen waterfall model as our analysis model.

**Requirements**

**Design**

**Implementation**

**Verification**

**Deployment**

**Maintenance**

**3.4. Modules**

**3.4.1. Find**

**3.4.1.1. Pronunciation**

**3.4.1.2. Previous**

**3.4.1.3. Next**

**3.4.2. Add**

**3.4.3. Update**

**3.4.4. Delete**

**3.4.5. Load**

**3.4.1. Find:**

Retrieve and display details of the word

If word exists in the dictionary database

word = given Input

Show a message to the user saying that word not found

No

Yes

**3.4.1.1. Pronunciation**

Process text to speech on the word

Yes

If Find is successful

Say that the word doesn’t exist in the dictionary to the user.

No

**3.4.1.2. Previous**

Retrieve and display details of the previous word

Yes

If Find is successful

Show that the word doesn’t exist in the dictionary to the user.

No

**3.4.1.3. Next**

Retrieve and display details of the next word

Yes

If Find is successful

Show that the word doesn’t exist in the dictionary to the user.

No

**3.4.2. Add:**

Get all the details and insert into dictionary

If word exist in the dictionary database

word = given Input

Show a message to the user saying that word already exists

Yes

No

**3.4.3. Update:**

Get all the details and update the dictionary

If word exists in the dictionary database

word = given Input

Show a message to the user saying that word not found

No

Yes

**3.4.4. Delete:**

Delete the word from the dictionary

If word exist in the dictionary database

word = given Input

Show a message to the user saying that word not found

No

Yes

**3.4.5. Load:**

If loadable has next

Show to user that loading has been completed successfully

No

yes

If the word exists in the dictionary

Yes

No

Insert the word and its details into the dictionary

1. FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase we are going to describe our feasibility study mainly in 3 different categories they are technical feasibility, economic feasibility and operational feasibility.

* 1. **Technical Feasibility**

This study is carried out to check the technical feasibility, i.e., the technical requirements of the system. According our analysis, we don’t need any high end technology to do this we just need basic free technologies like android studio, eclipse ide, Java FX, Beautiful Soup, etc. Since we believe that the problem solution proposed by us is practically possible without any need of any cost efficient technology but the process to solve may be quite time consuming since we need to perform web scraping which is a combination of two processes namely web crawling and data filtering but not a big problem since we have pretty much enough time and enough number of systems to work with and a good enough technical team to support so we can say that its technically feasible

* 1. **Economic Feasibility**

This study is carried out to check the economic feasibility, i.e., the economical requirements of the system. Since we are willing to hardcode most of the functionality by ourselves and even if a need arises we are planning to make use of some open source libraries. And we don’t even need high bandwidth to perform web scraping the basic internet is just enough to satisfy our needs. The amount of fund that our group can pour into the research and development of the system is limited but according to our knowledge this is just achievable with what we have since we are efficient enough to develop things based on our previous experience and we might need to spend some amount in our projects deployment phase to maintain a website and if possible to publish our application into google play store and for further maintenance except that we think the rest of the process doesn’t need any big amount so by taking all of things that are described till now into consideration we can assure that our project is economically feasible.

* 1. **Operational Feasibility**

This aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently as we observed a user can grab things just by using it once since we made our UI to be as simple as possible we didn’t allow any operation to be complicated for a user to handle since we want each and every basic person to use our application with their own free will and not to get confused as they did with lot of other dictionary applications which restricted them to some boundaries. So we can assure that each and every operation can be easily performed by the user without any doubt.

1. SYSTEM REQUIREMENT SPECIFICATION
   1. **Introduction**

## **Purpose:**

This SRS defines Functional and non-functional requirements of PED version1.0. This document is intended for the following group of people: -

* Developers for the purpose of maintenance and new releases of the software.
* Documentation writers.
* Testers

## **Scope:**

This document applies to Personal English Dictionary software PED version1.0. This software facilitates the user to perform various operations in his/her dictionary without connecting to internet. This software offers beneficial features such as modifiability, expandability for users.

* 1. **Functional Requirements**

` Functional requirements provide highly specific details that allow designers, developers and testers to create a system to meets precise specificationsbased on our observation following are the functional requirements of our project:

F1:User should be allowed to add any word that’s not existing in the dictionary

F2:User should be allowed to update any word its details that are existing in the dictionary

F3:User should be allowed to delete any word and its details that are existing in the dictionary

F4:User should be allowed to work with other applications when our application performs a time taking background process

F5:User should be allowed to navigate to previous word or the next word of the current word if that word exists in the dictionary

F6: User should be allowed to know the pronunciation of a word if that word exists in the dictionary

* 1. **Non-Functional Requirements**

Non-Functional requirements provide qualities and characteristics of the system that are not features. Based on our observation following are the functional requirements of our project:

N1: A word should be added to auto fill words list as soon as its successfully added to the dictionary

N2: A word should be removed from the auto fill words list as soon as its successfully removed from the dictionary

N3: A loading animation should be displayed when a time consuming backend process takes place

N4: A pop up kind of message should be shown to the user to notify the state of a process completion either its successful or unsuccessful due to an error

N5: A user should be allowed to know what’s happening in the background when a time consuming process is taking place for example when a loadable file is being processed into the dictionary he/she should be notified how much percentage of the loadable file is already processed into database of the dictionary etc.

* 1. **System Requirements**
     1. **Software Requirements**

|  |  |
| --- | --- |
| Operating System: | Windows7 or above |
| IDEs: | Android Studio & Eclipse |
| Programming Environments: | Java 8.0+& Python 3.0+ |
| Editors: | Visual Studio Code or any alternative |
| External Java Libraries: | JDBC, JFoenix, JavaFX, ControlsFX & MarryTTS |
| APIs: | requests, Beautiful Soup 4, nbsp MaterialFilePicker, SqliteAssetHelper & pl.droidsonroids.gif |
| Browser: | Google Chrome or any alternative |
| Other software: | Photo Shop CS6, Scene builder 8, Launcher 4j, inno setup. |

* + 1. **Hardware Requirements**

|  |  |
| --- | --- |
| Network Interface Card: | To access Internet over Ethernet or Wi-Fi |
| RAM: | 4GB or above |
| Storage: | 250GB or above |
| Input Devices: | Basic Mouse & Basic Keyboard |
| Output Devices: | Monitor & Speaker |

1. SYSTEM DESIGN

**6.1. UML Modeling**

**6.1.1. Importance of UML in Modeling:**

In the field of software engineering, the Unified Modeling Language (UML) is a standardized visual specification language for object modeling. UML is a general-purpose modeling language that includes a graphical notation used to create an abstract model of a system, referred to as a UML model The model also contains a "Semantic backplane” documentation such as written use cases that drive the model elements and diagrams.

**6.2. Class diagram:**

Classes are the most important building block of any object oriented system. A class is a description of set of objects that share the same attributes, operations, relationships and semantics. A class implements one or more interfaces. It is graphically rendered as a rectangle

Classes may contain:

*Names:* Every class must have a name that distinguishes it from other classes

*Attributes:* It is name property of a class that describes a range of values that instances of the property may hold

*Operations:* An operation is an implementation of a service that can be requested from any object of the class to affect behavior

A class diagram shows a set of classes, interfaces, collaborations and their relationships. The class diagrams are used to model the static design view of the system. Graphically, it is a collection of vertices and arcs.

Class diagram commonly contain the following things:

*Classes:* A class is a description of set of objects that share the same attributes, operations, relationships and semantics

*Interfaces:* An interface is a collection of operations that are used to specify a service of a class or component

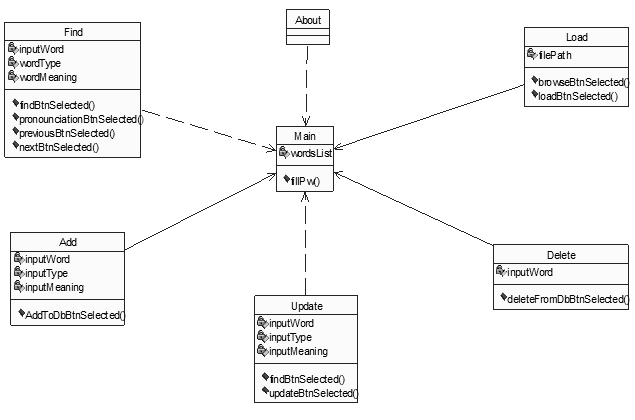
*Collaborations:* It is a society of classes, interfaces and other elements that work together to provide some Co-operative behavior that’s bigger than some of all its parts. It is rendered as an ellipse with dashed lines

**Identifying Relationships:**

In UML the ways that things can connect to each other wither logically or physically are modeled as relationships. In object-oriented modeling there are three kinds of relationships that are most important, they are

* Dependencies: It is a using relationship that states that a change in specification of one thing may affect another thing that uses each. Graphically dependency is rendered as a dashed directed line.
* Generalization: It is relationship between a general thing (called parent) and a more specific kind of that thing (called the child).Generalization is sometimes called ‘is-a-kind-of’ relationship. It means that objects of the child may be used anywhere that parent may appear. Graphically it is rendered as a solid directed line with a large open arrow head pointing to the parent.
* Association: It is a structural relationship that specifies that objects of one thing are connected to the objects of another. The associations that connect more than two classes are called n-array associations. Graphically it is rendered as solid line connecting the same or different classes.

The class diagram of our project is represented as follows



Class diagram

**6.3. Use case Diagram**

Use case diagram: A use case diagram shows a set of use cases and actors and their relationships. We apply the use case diagram to illustrate the static use case view of a system. Use case diagram are especially important in organizing and modeling the behaviors of a system.

Use case: A use case describes what a system (or a subsystem, class or interface) does but it does not specify how it does it. For dynamic aspects of systems, use case diagram is used to model the behavior of the system, sub-system or classes. Each one shows a set of use case, actors and their relationship with role names

Actor: An actor is the one plays an active role in the system. Typically an actor represents a role that a human, a hardware device, or even another system plays with a system

The relationships or stereotype in this use case are below:

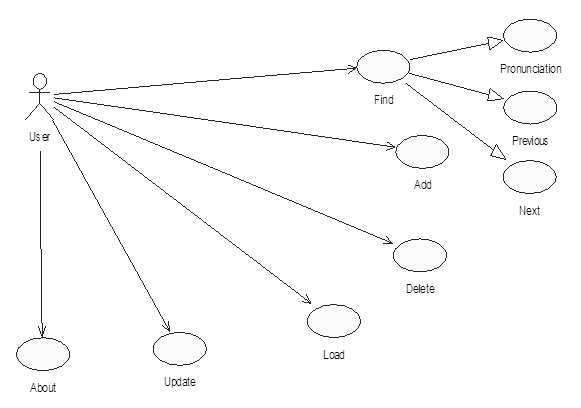
Include:

You can use the «include» stereotype to indicate that base use case “includes” the contents or behavior of common use case. This enables you to factor out frequent common behavior.

Extends:

You can use the «extend» stereotype to indicate that base use case may include the behavior specified by the “extending” use case.

The use case diagram of our project is as follows:



Use-case Diagram

**6.4. Activity Diagram**

The purpose of an activity diagram is to provide a view of flows and what is *going on inside a use case or among several classes. An activity diagram is just to explain* the internal operations performed and also the transitions that are triggered by the completion of the particular operations. At the abstract level it explains the sequence of the activities. This focus on the events occurring to a single object as it responds to messages, an activity diagram can be used to model an entire process.

Activity diagram commonly contains

Activity states and action states

Calling an operation on an object sending a signal to an object or even creating or destroying an object, these executable computations are called action states because they are the states of the system, representing the execution of an action.

Activity states can be further decomposed, their activity being represented by other activity diagrams these may be interrupted and take some duration to complete.

Transitions

When the action or activity of a state completes flow of control passes immediately to the next action or activity state. You specify this flow by using transitions to show the path from one action or activity state to next action or activity state. You represent this as simple directed line.

Object Flow

Objects may be involved in the flow of control associated with an activity diagram. The use of dependency relationships and objects is called an object flow because it represents participation of an object in the flow of control.

**Activity diagram – Find**

****

Activity diagram – Find

**Activity diagram – Add**

****

Activity diagram – add

**Activity Diagram – Update**

****

Activity diagram – update

**Activity Diagram – Delete**

****

Activity diagram – delete

**Activity Diagram – Load**

****

Activity diagram – load

**Activity diagram - About**

****

Activity diagram – about

**6.5. Sequence Diagram**

A sequence diagram emphasizes the time ordering of messages. These are used to model the dynamic aspects of the system. A sequence diagram shows a set of objects and messages that are dispatched between those objects based on time-ordering.

Sequence-diagram has two features

1. The object life line: An object life line is the vertical dashed line that represents the existence of an object over a period of time. Most objects that appear in an interaction diagram will be in existence for the duration of the interaction, so these objects are all aligned on the top of the diagram, with their lifelines drawn from the top of the diagram to the bottom.

2. The focus of control is a tall, thin rectangle that shows the period of time during which an object is performing an action, either directly or a sub ordinate procedure. The top of the rectangle is aligned with the start of the action the bottom is aligned with its completion.

The below sequence diagrams has indicated two

The objects considered according to the available details. The lifeline consists of messages as per the sequence of actions occurred.

The messages on the life lines are based on the actions performed by the objects. The letter details will be maintained. This diagram shows the flow of actions or the sequence of actions while inserting the details into the database.

**Sequence diagram – Find**

****

**Sequence diagram – Add**

****

**6.5.3. Sequence diagram – Update**

****

**6.5.4. Sequence diagram – delete**

****

**6.5.5. Sequence diagram – load**

****

**6.5.6. Sequence diagram – About**

****

**6.6. Component**

Component diagrams are different in terms of nature and behaviour. Component diagrams are used to model the physical aspects of a system. Now the question is, what are these physical aspects? Physical aspects are the elements such as executables, libraries, files, documents, etc. which reside in a node.

Component diagrams are used to visualize the organization and relationships among components in a system. These diagrams are also used to make executable systems.

****

Component diagram

**6.7. Deployment**

Deployment diagrams are used to visualize the topology of the physical components of a system, where the software components are deployed.

Deployment diagrams are used to describe the static deployment view of a system. Deployment diagrams consist of nodes and their relationships.

Before drawing a deployment diagram, the following artifacts should be identified −

* Nodes
* Relationships among nodes

The following is the deployment diagram of our project:

****

Deployment diagram

1. CODING

**7.1. Software Description**

**Web Scraping:**

Web scraping, web harvesting, or web data extraction is [data scraping](https://en.wikipedia.org/wiki/Data_scraping) used for [extracting data](https://en.wikipedia.org/wiki/Data_extraction) from [websites](https://en.wikipedia.org/wiki/Website). Web scraping software may access the World Wide Web directly using the [Hypertext Transfer Protocol](https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol), or through a web browser. While web scraping can be done manually by a software user, the term typically refers to automate processes implemented using a [bot](https://en.wikipedia.org/wiki/Internet_bot) or [web crawler](https://en.wikipedia.org/wiki/Web_crawler). It is a form of copying, in which specific data is gathered and copied from the web, typically into a central local [database](https://en.wikipedia.org/wiki/Database) or spreadsheet, for later [retrieval](https://en.wikipedia.org/wiki/Data_retrieval) or [analysis](https://en.wikipedia.org/wiki/Data_analysis).

Web scraping a web page involves fetching it and extracting from it. Fetching is the downloading of a page (which a browser does when you view the page). Therefore, web crawling is a main component of web scraping, to fetch pages for later processing. Once fetched, then extraction can take place. The content of a page may be [parsed](https://en.wikipedia.org/wiki/Parsing), searched, reformatted, its data copied into a spreadsheet, and so on. Web scrapers typically take something out of a page, to make use of it for another purpose somewhere else. An example would be to find and copy names and phone numbers, or companies and their URLs, to a list (contact scraping).

Web scraping is used for [contact scraping](https://en.wikipedia.org/wiki/Contact_scraping), and as a component of applications used for [web indexing](https://en.wikipedia.org/wiki/Web_indexing), [web mining](https://en.wikipedia.org/wiki/Web_mining) and [data mining](https://en.wikipedia.org/wiki/Data_mining), online price change monitoring and [price comparison](https://en.wikipedia.org/wiki/Comparison_shopping_website), product review scraping (to watch the competition), gathering real estate listings, weather data monitoring, [website change detection](https://en.wikipedia.org/wiki/Change_detection_and_notification), research, tracking online presence and reputation, [web mashup](https://en.wikipedia.org/wiki/Web_mashup) and, web data integration.

[Web pages](https://en.wikipedia.org/wiki/Web_page) are built using text-based mark-up languages ([HTML](https://en.wikipedia.org/wiki/HTML) and [XHTML](https://en.wikipedia.org/wiki/XHTML)), and frequently contain a wealth of useful data in text form. However, most web pages are designed for human [end-users](https://en.wikipedia.org/wiki/End-user_(computer_science)) and not for ease of automated use. Because of this, tool kits that scrape web content were created. A web scraper is an [Application Programming Interface](https://en.wikipedia.org/wiki/Application_Programming_Interface) (API) to extract data from a web site. Companies like [Amazon AWS](https://en.wikipedia.org/wiki/Amazon_AWS) and [Google](https://en.wikipedia.org/wiki/Google) provide web scraping tools, services and public data available free of cost to end users.

Newer forms of web scraping involve listening to data feeds from web servers. For example, [JSON](https://en.wikipedia.org/wiki/JSON) is commonly used as a transport storage mechanism between the client and the web server.

There are methods that some websites use to prevent web scraping, such as detecting and disallowing bots from crawling (viewing) their pages. In response, there are web scraping systems that rely on using techniques in [DOM](https://en.wikipedia.org/wiki/Document_Object_Model) parsing, [computer vision](https://en.wikipedia.org/wiki/Computer_vision) and [natural language processing](https://en.wikipedia.org/wiki/Natural_language_processing) to simulate human browsing to enable gathering web page content for offline parsing.

We got a list of words from GitHub and then we had chosen python to perform web scraping since it’s a scripting language we thought that it will be quite efficient based on our research to know about which programming language is better to perform scrapping we came to know that there is an efficient API written in python which is known as Beautiful Soup 4 which can be added to our programming environment by using pip module through a simple command “pip install bs4” which provides an HTML parser which we had used to parse the web page content which we get through a get request using another API which is known as requests which sends a get request by appending a word from our collected list to the URL of Oxford website which transforms into an URL of a web page so that we can get details of a particular word in a raw HTML format, then we decode the retrieved data into UTF8 format to make it readable then we filtered the readable data using re module in python to get the required part of readable data and then we had send the filtered data to beautiful soup html parser to get the details like type, definition and examples of collected word if the parser retrieves details successfully then we are storing those details into a text file then we perform this process again and again till all the words in our collected list gets completed.

**Database Creation:**

After performing web scrapping our next objective was to create a database based on our requirements we had chosen to make use of SQLite as our application’s database since our application requires portability and also it’s a single user local application so obviously SQLite will serve as a best embedded database for our needs to satisfy all the functionality requirements.

There are many advantages to using SQLite as an application file format, including:

1. **Better performance**
   * Reading and writing from an SQLite database is often faster than reading and writing individual files from disk. See [35% Faster Than The File system](https://www.sqlite.org/fasterthanfs.html) and [Internal Versus External BLOBs](https://www.sqlite.org/intern-v-extern-blob.html).
   * The application only has to load the data it needs, rather than reading the entire file and holding a complete parse in memory.
   * Small edits only overwrite the parts of the file that change, reducing write time and wear on SSD drives.
2. **Reduced application cost and complexity**
   * No application file I/O code to write and debug.
   * Content can be accessed and updated using concise SQL queries instead of lengthy and error-prone procedural routines.
   * The file format can be extended in future releases simply by adding new tables and/or column, preserving backwards compatibility.
   * Applications can leverage the [full-text search](https://www.sqlite.org/fts3.html) and [RTREE](https://www.sqlite.org/rtree.html) indexes and use triggers to implement an [automated undo/redo stack](https://www.sqlite.org/undoredo.html).
   * Performance problems can often be resolved, even late in the development cycle, using [CREATE INDEX](https://www.sqlite.org/lang_createindex.html), avoiding costly redesign, rewrite, and retest efforts.
3. **Portability**
   * The application file is portable across all operating systems, 32-bit and 64-bit and big- and little-endian architectures.
   * A federation of programs, perhaps all written in different programming languages, can access the same application file with no compatibility concerns.
   * Multiple processes can attach to the same application file and can read and write without interfering with each another.
   * Diverse content which might otherwise be stored as a "pile-of-files" is encapsulated into a single disk file for simpler transport via scp/ftp, USB stick, and/or email attachment.
4. **Reliability**
   * Content can be updated continuously and atomically so that little or no work is lost in a power failure or crash.
   * Bugs are far less likely in SQLite than in custom-written file I/O code.
   * SQL queries are many times smaller than the equivalent procedural code, and since the number of bugs per line of code is roughly constant, this means fewer bugs overall.
5. **Accessibility**
   * SQLite database content can be viewed using a wide variety third-party tools.
   * Content stored in an SQLite database is more likely to be recoverable decades in the future, long after all traces of the original application have been lost. Data lives longer than code.

Since time was limited we thought that we can’t waste our time by performing hard coding so we planned to automate the entire database creation process using java in which we took the output of web scrapping and performed file operations to transform retrieved data into SQL query format and then we processed those queries to insert details into database. This process was iterated till all the information that was gathered through web scrapping is completely processed.

**User Interface and User Experience:**

**Why did we choose Java FX for our desktop application development?**

JavaFX is a set of graphics and media packages that enables developers to design, create, test, debug, and deploy rich client applications that operate consistently across diverse platforms. The cross-platform compatibility enables a consistent runtime experience for JavaFX applications developers and users.

There are many advantageous features provided by javaFX which are listed as follows

JavaFX 2.2 and later releases have the following features:

* **Java APIs**. JavaFX is a Java library that consists of classes and interfaces that are written in native Java code. The APIs are designed to be a friendly alternative to Java Virtual Machine (Java VM) languages, such as JRuby and Scala.
* **FXML and Scene Builder**. FXML is an XML-based declarative mark-up language for constructing a JavaFX application user interface. A designer can code in FXML or use JavaFX Scene Builder to interactively design the graphical user interface (GUI). Scene Builder generates FXML mark-up that can be ported to an IDE where a developer can add the business logic.
* **Web View**. A web component that uses WebKitHTML technology to make it possible to embed web pages within a JavaFX application. JavaScript running in Web View can call Java APIs, and Java APIs can call JavaScript running in Web View.
* **Swing interoperability**. Existing Swing applications can be updated with new JavaFX features, such as rich graphics media playback and embedded Web content.
* **Built-in UI controls** **and CSS**. JavaFX provides all the major UI controls required to develop a full-featured application. Components can be skinned with standard Web technologies such as CSS
* **Canvas API**. The Canvas API enables drawing directly within an area of the JavaFX scene that consists of one graphical element (node).
* **Multitouch Support**. JavaFX provides support for multitouch operations, based on the capabilities of the underlying platform.
* **Hardware-accelerated graphics pipeline**. JavaFX graphics are based on the graphics rendering pipeline (Prism). JavaFX offers smooth graphics that render quickly through Prism when it is used with a supported graphics card or graphics processing unit (GPU). If a system does not feature one of the recommended GPUs supported by JavaFX, then Prism defaults to the Java 2D software stack.
* **High-performance media engine**. The media pipeline supports the playback of web multimedia content. It provides a stable, low-latency media framework that is based on the GStreamer multimedia framework.
* **Self-contained application deployment** **model**. Self-contained application packages have all of the application resources and a private copy of the Java and JavaFX runtimes. They are distributed as native installable packages and provide the same installation and launch experience as native applications for that operating system.

**Why did we choose native android for our android mobile application development?**

The basic benefit of developing apps in its native language like Android is saving of time and money due to coding in one language. The native language is the best choice due to better service and the best performance.

Advantages of native language over others are as follows

* The basic requirements can be converted into reality easily.
* Use of one coding language is better to understand and implement rather than using different languages.
* Easiest integration of the advanced features (AR/VR, face recognition, machine learning)
* Modification can be done easily
* Code runs faster and better
* Integrating new features is quick and easy
* Runs inside the operating system
* Flexibility exist and compatibility is high.
* Best user experience
* More fragmented in use and integration
* Hardware accessibility is maximum.
* Fine-tuned to run on a specific platform
* Quality and performance can be tested easily.
* Interactive and intuitive

As we thought to provide a couple of features to both desktop and mobile users so we made an analysis on how the features should be designed to satisfy user functionality, the following are the conclusions that we had drawn based on our analysis

* A user should be provided an options list which should allow the user to perform multiple tasks based on his/her choice since we thought to provide each and every feature in an explicit manner we thought to provide the options list as a navigation drawer which will allow a user to navigate from one scene to another based on his or her choice to perform a required operation explicitly.
* Since we are working with a large data set we thought to make the other things as less resource hungry as possible so for desktop we made use of a border pane as our main stage, each and every option will be provided as a separate scene where as in android we made use of an activity as our main layout and made each and every option as a fragment programming to keep it faster and smooth to operate.
* When a high end background process takes place a user might feel disgusted if his/her screen gets froze so to avoid that kind of situation for our application users we planned to design few loading animations which will be displayed when a high end background process takes place through thread synchronization.

**7.2. Sample Code**

**Web Scraping:**

**Dictionary.py**

import re

import requests

import urllib

from bs4 import BeautifulSoup

def findDef(word):

    url = 'https://en.oxforddictionaries.com/definition/'+word

    data = urllib.request.urlopen(url).read()

    data0 = data.decode("utf-8")

    data1 = re.search('Phrases</strong></h3>',data0)

    try:

     data1 = data0[0:data1.start()]

    except Exception:

     data1 = data0

     pass

    result=""

    soup = BeautifulSoup(data1,'html.parser')

    title = soup.find("meta", property="og:title")

    title = title

    fetchedWord = soup.find('span',{'class':'hw'})

    try:

     t = title["content"]

     index = t.index("|")

     t = t[:index]

     print(t)

     result = "--Word--\n"+t

    except Exception:

     result = "--Word--\n"+word

     pass

    data2 = soup.find\_all('span',{'class':'pos'})

    result=result+"\n--Type--"

    count = 0

    for i in data2:

        try:

            count = count+1

            if count>1:

             result = result+", "+i.string

            else:

             result = result+"\n"+i.string

        except Exception:

         result = result+"\n"+"UnKnownKind"

         pass

    if(count==0):

        return None

    count = 0

    data3 = soup.find\_all('span',{'class':'ind'})

    result=result+"\n"+"--Meanings--"

    for i in data3:

        try:

            result = result+"\n"+i.string

        except Exception:

         result = result+"\n"+"UnKnownKind"

         pass

    data4 = soup.find\_all('div',{'class':'ex'})

    result=result+"\n"+"--Examples--"

    for i in data4:

        try:

            result = result+"\n"+i.em.string

        except Exception:

         result = result+"\n"+"UnKnownKind"

         pass

    return(result)

with open('words.txt','r', encoding='utf-8') as wordsFile:

    for w in wordsFile:

        try:

         str = findDef(w)

        except Exception as e:

         print(e)

         continue

        if(str == None):

            continue

        count = str.count("\n")

        if(count>4):

            destinyPath = "Dictionary/dictionary.txt"

            destinyFile = open(destinyPath,'a',encoding='utf-8')

            destinyFile.write(str+"\n")

            destinyFile.close()

**Transforming Scraped data:**

**IntermediateFileCreation.java**

import java.util.Scanner;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

public class IntermediateFileCreation

{

    public static void main(String args[]) throws IOException {

        Scanner s = new Scanner(System.in);

        String inputFile = s.next();

        FileInputStream fis = null;

        BufferedReader reader = null;

         PrintWriter printWriter = new PrintWriter(new FileWriter(new File("C:\\Users\\k\\Desktop\\Dictionary\\"+inputFile+"\\Dictionary\\intermediate.txt")));

        try

        {

            String filePath = "C:\\Users\\k\\Desktop\\Dictionary\\"+inputFile+"\\Dictionary\\Dictionary.txt";

            fis = new FileInputStream(filePath);

            reader = new BufferedReader(new InputStreamReader(fis));

            String line = reader.readLine();

            while(line != null)

            {

                line = line.replaceAll("\"","");

                line =  line.replaceAll("‘", "");

                line =  line.replaceAll("’", "");

                line =  line.replaceAll("|", "");

                line =  line.replaceAll("UnKnownKind, ", "");

                line =  line.replaceAll("UnKnownKind", "");

                if(line.length()!=0)

                {

                    if(line.equals("--Word--"))

                        printWriter.println();

                    printWriter.append(line+"|");

                }

                line = reader.readLine();

            }

        }

        catch(Exception e)

        {

        }

        finally

        {

            printWriter.close();

            reader.close();

        }

    }

}

**Database Creation:**

**MainTableCreation.java**

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.Statement;

public class MainTableCreation

{

    public static void main( String args[] )

    {

        Connection c = null;

        Statement stmt = null;

        try

        {

            Class.forName("org.sqlite.JDBC");

            c = DriverManager.getConnection("jdbc:sqlite:MASK.db");

            c.setAutoCommit(false);

            System.out.println("Opened database successfully");

            stmt = c.createStatement();

            String sql = "CREATE TABLE DICTIONARY "+

                              "(\_id INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,"+

                              "WORD TEXT UNIQUE NOT NULL,"+

                              "TYPE TEXT NOT NULL,"+

                              "DEFINITION TEXT)";

             stmt.executeUpdate(sql);

            stmt.close();

            c.commit();

            c.close();

     }

    catch ( Exception e )

    {

        System.err.println( e.getClass().getName() + ": " + e.getMessage() );

        System.exit(0);

    }

    System.out.println("Table created successfully");

    }

}

**Desktop Application:**

**Find.fxml**

<?xml version="1.0" encoding="UTF-8"?>

<?import com.jfoenix.controls.JFXButton?>

<?import com.jfoenix.controls.JFXTextArea?>

<?import com.jfoenix.controls.JFXTextField?>

<?import javafx.scene.image.ImageView?>

<?import javafx.scene.layout.AnchorPane?>

<?import javafx.scene.layout.HBox?>

<?import javafx.scene.text.Font?>

<AnchorPane fx:id="mainPane" prefHeight="660.0" prefWidth="864.0" style="-fx-background-color: white;" xmlns="http://javafx.com/javafx/8.0.141" xmlns:fx="http://javafx.com/fxml/1" fx:controller="application.FindController">

<children>

<JFXTextField fx:id="inputWord" alignment="TOP\_CENTER" focusColor="#517abd" labelFloat="true" layoutX="60.0" layoutY="125.0" prefHeight="51.0" prefWidth="624.0" promptText="Word" unFocusColor="#323232" AnchorPane.leftAnchor="60.0" AnchorPane.rightAnchor="60.0" AnchorPane.topAnchor="85.0">

<font>

<Font name="Segoe UI Semibold" size="24.0" />

</font>

</JFXTextField>

<JFXButton fx:id="findBtn" buttonType="RAISED" layoutX="655.0" layoutY="136.0" onAction="#findBtnSelected" ripplerFill="BLACK" style="-fx-background-color: #517abd;" text="Find" textFill="WHITE" AnchorPane.rightAnchor="60.0" AnchorPane.topAnchor="150.0">

<font>

<Font name="Segoe UI Semibold" size="24.0" />

</font>

</JFXButton>

<JFXButton fx:id="pronounciationBtn" layoutX="60.0" layoutY="234.0" onAction="#pronounciationBtnSelected" ripplerFill="WHITE" text=" " AnchorPane.leftAnchor="60.0" AnchorPane.topAnchor="140.0">

<graphic>

<ImageView fx:id="pronounceIcn" fitHeight="32.0" fitWidth="32.0" pickOnBounds="true" preserveRatio="true">

<image>

</image>

</ImageView>

</graphic>

</JFXButton>

<HBox alignment="TOP\_CENTER" prefHeight="51.0" prefWidth="584.0" AnchorPane.leftAnchor="140.0" AnchorPane.rightAnchor="140.0" AnchorPane.topAnchor="140.0">

<children>

<JFXButton fx:id="previousBtn" layoutX="671.0" layoutY="30.0" onAction="#previousBtnSelected" ripplerFill="WHITE" text=" ">

<graphic>

<ImageView fx:id="previousIcn" fitHeight="32.0" fitWidth="32.0" pickOnBounds="true" preserveRatio="true">

<image>

</image>

</ImageView>

</graphic>

</JFXButton>

<JFXButton fx:id="nextBtn" layoutX="736.0" layoutY="30.0" onAction="#nextBtnSelected" ripplerFill="WHITE" text=" ">

<graphic>

<ImageView fx:id="nextIcn" fitHeight="32.0" fitWidth="32.0" pickOnBounds="true" preserveRatio="true">

<image>

</image>

</ImageView>

</graphic>

</JFXButton>

</children>

</HBox>

<JFXTextField fx:id="wordType" focusColor="#517abd" labelFloat="true" layoutX="60.0" layoutY="342.0" promptText="Type" unFocusColor="#323232" AnchorPane.leftAnchor="60.0" AnchorPane.rightAnchor="60.0" AnchorPane.topAnchor="300.0">

<font>

<Font name="Segoe UI Semibold" size="24.0" />

</font>

</JFXTextField>

<JFXTextArea fx:id="wordMeaning" focusColor="#517abd" labelFloat="true" layoutX="60.0" layoutY="456.0" prefHeight="194.0" prefWidth="744.0" promptText="Definition" unFocusColor="#323232" AnchorPane.bottomAnchor="40.0" AnchorPane.leftAnchor="60.0" AnchorPane.rightAnchor="60.0" AnchorPane.topAnchor="400.0">

<font>

<Font name="Segoe UI Semibold" size="24.0" />

</font>

</JFXTextArea>

</children>

</AnchorPane>

**FindController.java**

package application;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.Statement;

import java.util.ArrayList;

import org.controlsfx.control.textfield.TextFields;

import com.jfoenix.controls.JFXButton;

import com.jfoenix.controls.JFXSnackbar;

import com.jfoenix.controls.JFXTextArea;

import com.jfoenix.controls.JFXTextField;

import javafx.beans.binding.Bindings;

import javafx.fxml.FXML;

import javafx.scene.image.Image;

import javafx.scene.image.ImageView;

import javafx.scene.layout.AnchorPane;

public class FindController

{

String input;

@FXML

public AnchorPane mainPane;

@FXML

private JFXButton pronounciationBtn;

@FXML

private JFXButton findBtn;

@FXML

private JFXButton previousBtn;

@FXML

private JFXButton nextBtn;

@FXML

private JFXTextArea wordMeaning;

@FXML

private JFXTextField inputWord;

@FXML

private JFXTextField wordType;

@FXML

private ImageView previousIcn;

@FXML

private ImageView nextIcn;

@FXML

private ImageView pronounceIcn;

@FXML

private JFXSnackbar snackbar;

TextToSpeech obj = new TextToSpeech();

public void initialize()

{

snackbar = new JFXSnackbar(mainPane);

TextFields.bindAutoCompletion(inputWord, Main.pw);

pronounceIcn.setImage(new Image("img-files/pronounce.png"));

previousIcn.setImage(new Image("img-files/previous.png"));

nextIcn.setImage(new Image("img-files/next.png"));

//pronounceBtn hover effect

ImageView pronounceHoverIcn = new ImageView(

new Image("img-files/pronounce\_hover.png")

);

pronounciationBtn.graphicProperty().bind(

Bindings.when(

pronounciationBtn.hoverProperty()

)

.then(pronounceHoverIcn)

.otherwise(pronounceIcn)

);

//previousBtn hover effect

ImageView previousHoverIcn = new ImageView(

new Image("img-files/previoushover.png")

);

previousBtn.graphicProperty().bind(

Bindings.when(

previousBtn.hoverProperty()

)

.then(previousHoverIcn)

.otherwise(previousIcn)

);

//nextBtn hover effect

ImageView nextHoverIcn = new ImageView(

new Image("img-files/nexthover.png")

);

nextBtn.graphicProperty().bind(

Bindings.when(

nextBtn.hoverProperty()

)

.then(nextHoverIcn)

.otherwise(nextIcn)

);

obj.setVoice("cmu-bdl-hsmm");

//Removing editable property for type and meaning

wordType.setEditable(false);

wordMeaning.setEditable(false);

}

@FXML

public void pronounciationBtnSelected()

{

String s="";

if(Main.pw.indexOf(inputWord.getText())>=0)

{

s = inputWord.getText();

}

else

{

s="SORRY! word not found in the dictionary";

}

obj.speak(" "+s+" ", 2.0f, false, true);

}

@FXML

public void findBtnSelected()

{

String word=containsIgnoreCase(inputWord.getText().toString(), Main.pw);

if(word.length()==0)

{

wordType.setText("");

wordMeaning.setText("");

snackbar.show("No input given",3000);

}

else if ( Main.pw.indexOf(word)<0)

{

wordType.setText("");

wordMeaning.setText("");

snackbar.show("Sorry word not found",3000);

}

else

{

Connection c = null;

Statement stmt = null;

try

{

Class.forName("org.sqlite.JDBC");

c = DriverManager.getConnection("jdbc:sqlite:MASK.db");

c.setAutoCommit(false);

stmt = c.createStatement();

ResultSet rs = stmt.executeQuery( "SELECT \* FROM DICTIONARY WHERE WORD = \""+word+"\";" );

if( rs.next() )

{

inputWord.setText(word);

wordType.setText(rs.getString("type"));

wordMeaning.setText(rs.getString("definition").replaceAll("Examples :- ","\nExamples :- "));

}

else

{

snackbar.show("Sorry word not found",3000);

}

rs.close();

stmt.close();

c.close();

}

catch(Exception e)

{

wordType.setText("");

wordMeaning.setText("");

snackbar.show("Sorry unable to find",3000);

}

}

}

@FXML

public void findBtnSelectedSimulation()

{

String word=input;

System.out.println(word);

if(word.length()==0)

{

wordType.setText("");

wordMeaning.setText("");

snackbar.show("No input given",3000);

}

else if ( Main.pw.indexOf(word)<0)

{

wordType.setText("");

wordMeaning.setText("");

System.out.println(input);

snackbar.show("Sorry word not found",3000);

}

else

{

Connection c = null;

Statement stmt = null;

try

{

Class.forName("org.sqlite.JDBC");

c = DriverManager.getConnection("jdbc:sqlite:MASK.db");

c.setAutoCommit(false);

stmt = c.createStatement();

ResultSet rs = stmt.executeQuery( "SELECT \* FROM DICTIONARY WHERE WORD = \""+word+"\";" );

if( rs.next() )

{

inputWord.setText(word);

wordType.setText(rs.getString("type"));

wordMeaning.setText((((rs.getString("definition")).replaceAll("Examples :- ","\nExamples :- "))));

}

else

{

System.out.println(input);

snackbar.show("Sorry word not found", 3000);

}

rs.close();

stmt.close();

c.close();

}

catch(Exception e)

{

e.printStackTrace();

wordType.setText("");

wordMeaning.setText("");

snackbar.show("Sorry unable to find", 3000);

}

}

}

@FXML

public void previousBtnSelected()

{

String word=containsIgnoreCase(inputWord.getText().toString(), Main.pw);

if(word.length()==0)

{

wordType.setText("");

wordMeaning.setText("");

snackbar.show("No input given",3000);

}

else if ( Main.pw.indexOf(word)<0)

{

wordType.setText("");

wordMeaning.setText("");

snackbar.show("Sorry word not found",3000);

}

else if ( Main.pw.indexOf(word)==0)

{

input = Main.pw.get(Main.pw.size()-1);

findBtnSelectedSimulation();

}

else

{

try

{

if(Main.pw.indexOf(word)>0)

{

int id = Main.pw.indexOf(word);

if(id == 0)

id = Main.pw.size()-1;

else

id = id-1;

input = Main.pw.get(id);

findBtnSelectedSimulation();

}

}

catch(Exception e)

{

e.printStackTrace();

wordType.setText("");

wordMeaning.setText("");

snackbar.show("Sorry unable to find", 3000);

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

}

}

}

@FXML

public void nextBtnSelected()

{

String word=containsIgnoreCase(inputWord.getText().toString(), Main.pw);

if(word.length()==0)

{

wordType.setText("");

wordMeaning.setText("");

snackbar.show("No input given",3000);

}

else if ( Main.pw.indexOf(word)<0)

{

wordType.setText("");

wordMeaning.setText("");

snackbar.show("Sorry word not found",3000);

}

else if ( Main.pw.indexOf(word)==Main.pw.size()-1)

{

input = Main.pw.get(0);

findBtnSelectedSimulation();

}

else

{

try

{

int id = Main.pw.indexOf(word);

id = id+1;

input = Main.pw.get(id);

System.out.println(input);

findBtnSelectedSimulation();

}

catch(Exception e)

{

e.printStackTrace();

wordType.setText("");

wordMeaning.setText("");

snackbar.show("Sorry unable to find", 3000);

System.err.println( e.getClass().getName() + ": " + e.getMessage() );

}

}

}

public String containsIgnoreCase(String str, ArrayList<String> list){

for(String i : list){

if(i.equalsIgnoreCase(str))

return i;

}

return str;

}

}

**Android Application:**

**fragment\_find.xml**

<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context="project.nriit.ped.FindFragment">

<RelativeLayout

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:padding="@dimen/activity\_horizontal\_margin">

<android.support.design.widget.TextInputLayout

android:id="@+id/label\_input\_word"

android:gravity="center"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<AutoCompleteTextView

android:id="@+id/inputWord"

android:inputType="textNoSuggestions"

android:gravity="center"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/label\_input\_word"

android:layout\_centerHorizontal="true"

android:layout\_centerVertical="true"

android:hint="Word" />

</android.support.design.widget.TextInputLayout>

<RelativeLayout

android:id="@+id/innerLayout"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/label\_input\_word"

>

<ImageButton

android:id="@+id/pronounciation"

android:layout\_width="32dp"

android:layout\_height="32dp"

android:layout\_alignParentStart="true"

android:background="#00ffffff"

android:scaleType="fitStart"

android:src="@drawable/ic\_voice\_black\_24dp" />

<ImageButton

android:id="@+id/previous"

android:layout\_width="32dp"

android:layout\_height="32dp"

android:layout\_toStartOf="@+id/spacer"

android:background="#00ffffff"

android:scaleType="fitStart"

android:src="@drawable/ic\_previous" />

<Space

android:id="@+id/spacer"

android:layout\_centerHorizontal="true"

android:layout\_height="10dp"

android:layout\_width="10dp"

android:layout\_centerInParent="true"

android:layout\_weight="1.0" />

<ImageButton

android:id="@+id/next"

android:layout\_width="32dp"

android:layout\_height="32dp"

android:layout\_toEndOf="@+id/spacer"

android:layout\_weight="1.0"

android:background="#00ffffff"

android:scaleType="fitStart"

android:src="@drawable/ic\_next"

/>

<Button

android:id="@+id/findBtn"

style="@style/Widget.AppCompat.Button.Colored"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_alignParentEnd="true"

android:layout\_alignParentTop="true"

android:background="#517abd"

android:text="Find"

android:textColor="@color/white" />

</RelativeLayout>

<android.support.design.widget.TextInputLayout

android:id="@+id/label\_type"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/innerLayout">

<EditText

android:id="@+id/type"

android:inputType="textNoSuggestions"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/label\_input\_type"

android:layout\_centerHorizontal="true"

android:layout\_centerVertical="true"

android:clickable="false"

android:cursorVisible="false"

android:focusable="false"

android:focusableInTouchMode="false"

android:hint="Type" />

</android.support.design.widget.TextInputLayout>

<android.support.design.widget.TextInputLayout

android:id="@+id/label\_definition"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/label\_type">

<EditText

android:id="@+id/definition"

android:inputType="textMultiLine|textNoSuggestions"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/label\_input\_type"

android:layout\_centerHorizontal="true"

android:layout\_centerVertical="true"

android:clickable="false"

android:cursorVisible="false"

android:focusable="false"

android:focusableInTouchMode="false"

android:hint="Definition" />

</android.support.design.widget.TextInputLayout>

</RelativeLayout>

</FrameLayout>

**FindFragment.java**

package project.nriit.ped;

import android.database.Cursor;

import android.os.Bundle;

import android.speech.tts.TextToSpeech;

import android.support.annotation.Nullable;

import android.support.v4.app.Fragment;

import android.util.Log;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.view.inputmethod.InputMethodManager;

import android.widget.AdapterView;

import android.widget.AutoCompleteTextView;

import android.widget.Button;

import android.widget.EditText;

import android.widget.ImageButton;

import android.widget.Toast;

import java.util.ArrayList;

import java.util.Locale;

import static android.content.ContentValues.TAG;

import static android.content.Context.INPUT\_METHOD\_SERVICE;

public class FindFragment extends Fragment {

String input;

String wordoutput,type,definition;

AutoCompleteTextView inputWord;

Button findBtn;

ImageButton nextBtn,previousBtn,pronounciationBtn;

EditText inputType,inputDefinition;

protected TextToSpeech ttobj;

public FindFragment() {

// Required empty public constructor

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

// Inflate the layout for this fragment

((MainActivity)getActivity()).setActionBarTitle("Find");

return inflater.inflate(R.layout.fragment\_find, container, false);

}

@Override

public void onDestroy() {

//Close the Text to Speech Library

if(ttobj != null) {

ttobj.stop();

ttobj.shutdown();

Log.d(TAG, "TTS Destroyed");

}

super.onDestroy();

}

@Override

public void onViewCreated(View view, @Nullable Bundle savedInstanceState){

ttobj=new TextToSpeech(getContext(), new TextToSpeech.OnInitListener() {

@Override

public void onInit(int status) {

}

});

ttobj.setLanguage(Locale.UK);

inputWord = (AutoCompleteTextView) view.findViewById(R.id.inputWord);

inputType = (EditText) view.findViewById(R.id.type);

inputDefinition = (EditText) view.findViewById(R.id.definition);

inputWord.setAdapter(MainActivity.adapter);

inputWord.setOnItemClickListener(new AdapterView.OnItemClickListener() {

@Override

public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,

long arg3) {

String selected = (String) arg0.getAdapter().getItem(arg2);

}

});

pronounciationBtn = (ImageButton) view.findViewById(R.id.pronounciation);

pronounciationBtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

String utteranceId=this.hashCode() + "";

String word="";

try{

word = containsIgnoreCase(inputWord.getText().toString(),MainActivity.wordsList);

}

catch(Exception ex)

{

}

if(word.length()==0)

ttobj.speak("SORRY! word not found in the dictionary", TextToSpeech.QUEUE\_FLUSH, null, utteranceId);

else if(MainActivity.wordsList.indexOf(word)>=0)

{

ttobj.speak(word, TextToSpeech.QUEUE\_FLUSH, null, utteranceId);

}

else

{

ttobj.speak("SORRY! wordo not found in the dictionary", TextToSpeech.QUEUE\_FLUSH, null, utteranceId);

}

}

});

findBtn = (Button) view.findViewById(R.id.findBtn);

findBtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

String word="";

try{

word = containsIgnoreCase(inputWord.getText().toString(),MainActivity.wordsList);

}

catch(Exception ex)

{

}

hideSoftKeyBoard();

if (word.length() == 0) {

inputType.setText("");

inputDefinition.setText("");

Toast.makeText(getActivity(), "No input given", Toast.LENGTH\_SHORT).show();

}

else if (MainActivity.wordsList.indexOf(word)<0) {

inputType.setText("");

inputDefinition.setText("");

Toast.makeText(getActivity(), "Sorry word not found", Toast.LENGTH\_SHORT).show();

wordoutput = "";

type = "";

definition = "";

}

else {

try {

DatabaseAccess databaseAccess = DatabaseAccess.getInstance(getContext());

databaseAccess.open();

Cursor cursor = databaseAccess.findDetails(word);

if (cursor == null) {

Toast.makeText(getActivity(), "Sorry word not found", Toast.LENGTH\_SHORT).show();

wordoutput = "";

type = "";

definition = "";

} else {

wordoutput = cursor.getString(1);

type = cursor.getString(2);

definition = cursor.getString(3).replaceAll("Examples :- ", "\nExamples :- ");

inputWord.setText(wordoutput);

inputWord.dismissDropDown();

inputType.setText(type);

inputDefinition.setText(definition);

cursor.close();

}

databaseAccess.close();

}

catch(Exception e)

{

inputType.setText("");

inputDefinition.setText("");

Toast.makeText(getActivity(), "Sorry unable to find", Toast.LENGTH\_SHORT).show();

}

}

}

});

previousBtn = (ImageButton) view.findViewById(R.id.previous);

previousBtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

String word="";

try{

word = containsIgnoreCase(inputWord.getText().toString(),MainActivity.wordsList);

}

catch(Exception ex)

{

}

hideSoftKeyBoard();

DatabaseAccess databaseAccess = DatabaseAccess.getInstance(getContext());

databaseAccess.open();

if (word.length() == 0) {

inputType.setText("");

inputDefinition.setText("");

Toast.makeText(getActivity(), "No input given", Toast.LENGTH\_SHORT).show();

}

else if(MainActivity.wordsList.indexOf(word)<0)

{

inputType.setText("");

inputDefinition.setText("");

Toast.makeText(getActivity(), "Sorry word not found", Toast.LENGTH\_SHORT).show();

wordoutput = "";

type = "";

definition = "";

}

else if(MainActivity.wordsList.indexOf(word)==0) {

try{

Cursor cursor = databaseAccess.findDetails(MainActivity.wordsList.get(MainActivity.wordsList.size()-1));

if(cursor!=null) {

wordoutput = cursor.getString(1);

type = cursor.getString(2);

definition = cursor.getString(3).replaceAll("Examples :- ", "\nExamples :- ");

inputWord.setText(wordoutput);

inputWord.dismissDropDown();

inputType.setText(type);

inputDefinition.setText(definition);

cursor.close();

}

else

{

inputType.setText("");

inputDefinition.setText("");

Toast.makeText(getActivity(), "Sorry word not found", Toast.LENGTH\_SHORT).show();

wordoutput = "";

type = "";

definition = "";

}

}

catch(Exception e)

{

inputType.setText("");

inputDefinition.setText("");

Toast.makeText(getActivity(), "Sorry unable to find", Toast.LENGTH\_SHORT).show();

}

}

else {

try {

Cursor cursor = databaseAccess.findDetails(MainActivity.wordsList.get(MainActivity.wordsList.indexOf(word) - 1));

if (cursor != null) {

wordoutput = cursor.getString(1);

type = cursor.getString(2);

definition = cursor.getString(3).replaceAll("Examples :- ", "\nExamples :- ");

inputWord.setText(wordoutput);

inputWord.dismissDropDown();

inputType.setText(type);

inputDefinition.setText(definition);

cursor.close();

}

}

catch(Exception e)

{

inputType.setText("");

inputDefinition.setText("");

Toast.makeText(getActivity(), "Sorry unable to find", Toast.LENGTH\_SHORT).show();

}

}

databaseAccess.close();

}

});

nextBtn = (ImageButton) view.findViewById(R.id.next);

nextBtn.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View view) {

String word="";

try{

word = containsIgnoreCase(inputWord.getText().toString(),MainActivity.wordsList);

}

catch(Exception ex)

{

}

hideSoftKeyBoard();

DatabaseAccess databaseAccess = DatabaseAccess.getInstance(getContext());

databaseAccess.open();

if (word.length() == 0) {

inputType.setText("");

inputDefinition.setText("");

Toast.makeText(getActivity(), "No input given", Toast.LENGTH\_SHORT).show();

}

else if(MainActivity.wordsList.indexOf(word)<0) {

inputType.setText("");

inputDefinition.setText("");

Toast.makeText(getActivity(), "Sorry word not found", Toast.LENGTH\_SHORT).show();

wordoutput = "";

type = "";

definition = "";

}

else if(MainActivity.wordsList.indexOf(word)==MainActivity.wordsList.size()-1) {

Cursor cursor = databaseAccess.findDetails(MainActivity.wordsList.get(0));

wordoutput = cursor.getString(1);

type = cursor.getString(2);

definition = cursor.getString(3).replaceAll("Examples :- ","\nExamples :- ");

inputWord.setText(wordoutput);

inputWord.dismissDropDown();

inputType.setText(type);

inputDefinition.setText(definition);

cursor.close();

}

else {

Cursor cursor = databaseAccess.findDetails(MainActivity.wordsList.get(MainActivity.wordsList.indexOf(word)+1));

wordoutput = cursor.getString(1);

type = cursor.getString(2);

definition = cursor.getString(3).replaceAll("Examples :- ","\nExamples :- ");

inputWord.setText(wordoutput);

inputWord.dismissDropDown();

inputType.setText(type);

inputDefinition.setText(definition);

cursor.close();

}

databaseAccess.close();

}

});

}

private void hideSoftKeyBoard() {

InputMethodManager imm = (InputMethodManager) getActivity().getSystemService(INPUT\_METHOD\_SERVICE);

if(imm.isAcceptingText()) { // verify if the soft keyboard is open

imm.hideSoftInputFromWindow(getActivity().getCurrentFocus().getWindowToken(), 0);

}

}

public String containsIgnoreCase(String str, ArrayList<String> list){

for(String i : list){

if(i.equalsIgnoreCase(str))

return i;

}

return str;

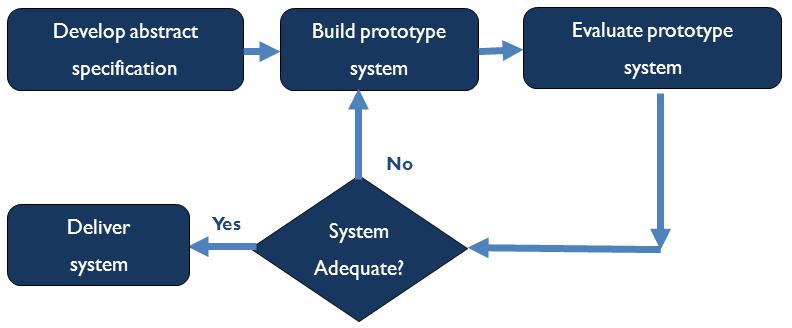
}

}

1. TESTING

Based on our analysis we thought that prototype testing would be better compared with other models of testing since our application deals with GUI taking reviews would improve our product’s quality.

## **What is Prototyping Model?**



prototype-model diagram

During Prototyping model, the software development team, clarify requirements and/or design elements, that generate mock-ups and prototypes of screens, reports, and processes. Although some of the prototypes may appear to be very substantial, they’re generally similar to a movie set: everything looks good from the front but there’s nothing in the back.

When a prototype is generated, the developer produces the minimum amount of code necessary to clarify the requirements or design elements under consideration. No effort is made to comply with coding standards, provide robust error management, or integrate with other database tables or modules. As a result, it is generally more expensive to retrofit a prototype with the necessary elements to produce a production module then it is to develop the module from scratch  
using the final system design document.

For these reasons, prototypes are never intended for business use, and are generally crippled in one way or another to prevent them from being mistakenly used as production modules by end-users.

### ****Advantages of Prototyping****

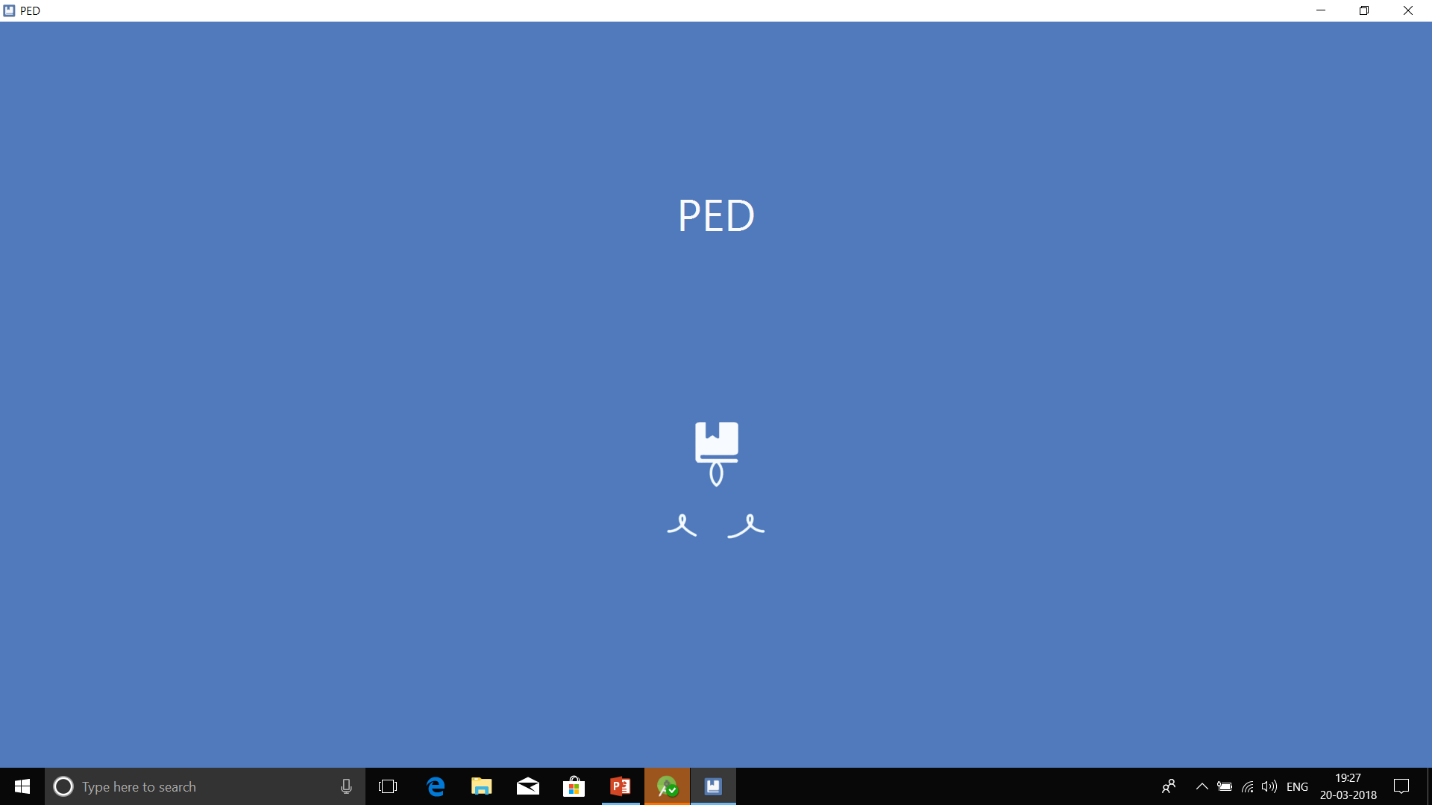
* The software designer and implementer can obtain feedback from the users early in the project.
* The client and the contractor can compare if the software made matches the software specification, according to which the software program is built.
* It also allows the software engineer some insight into the accuracy of initial project estimates and whether the deadlines and milestones proposed can be successfully met.

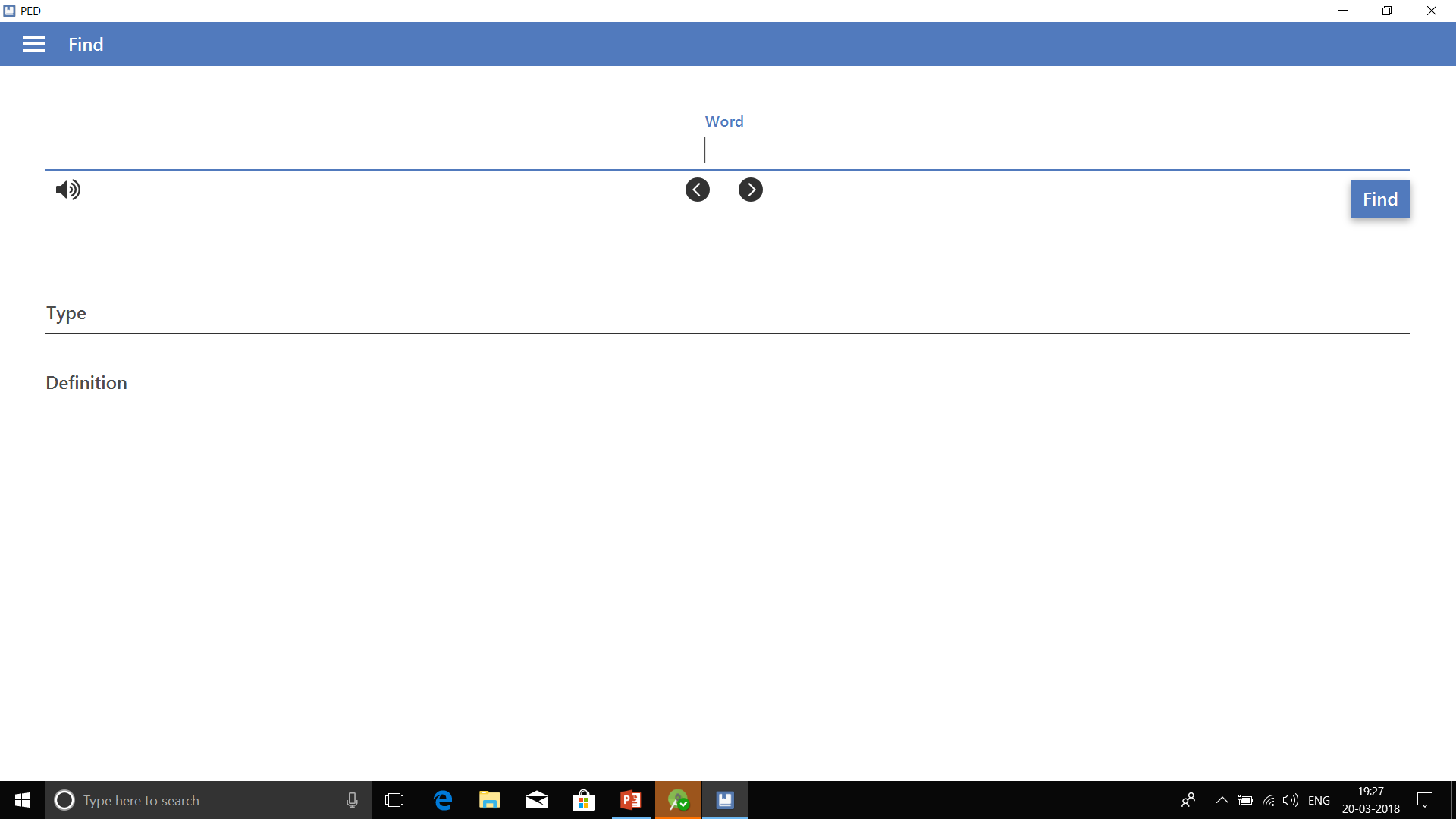
**Following are some of the valuable feedbacks which we got from people who had tested our application's previous prototypes which are taken care of in our current build:**

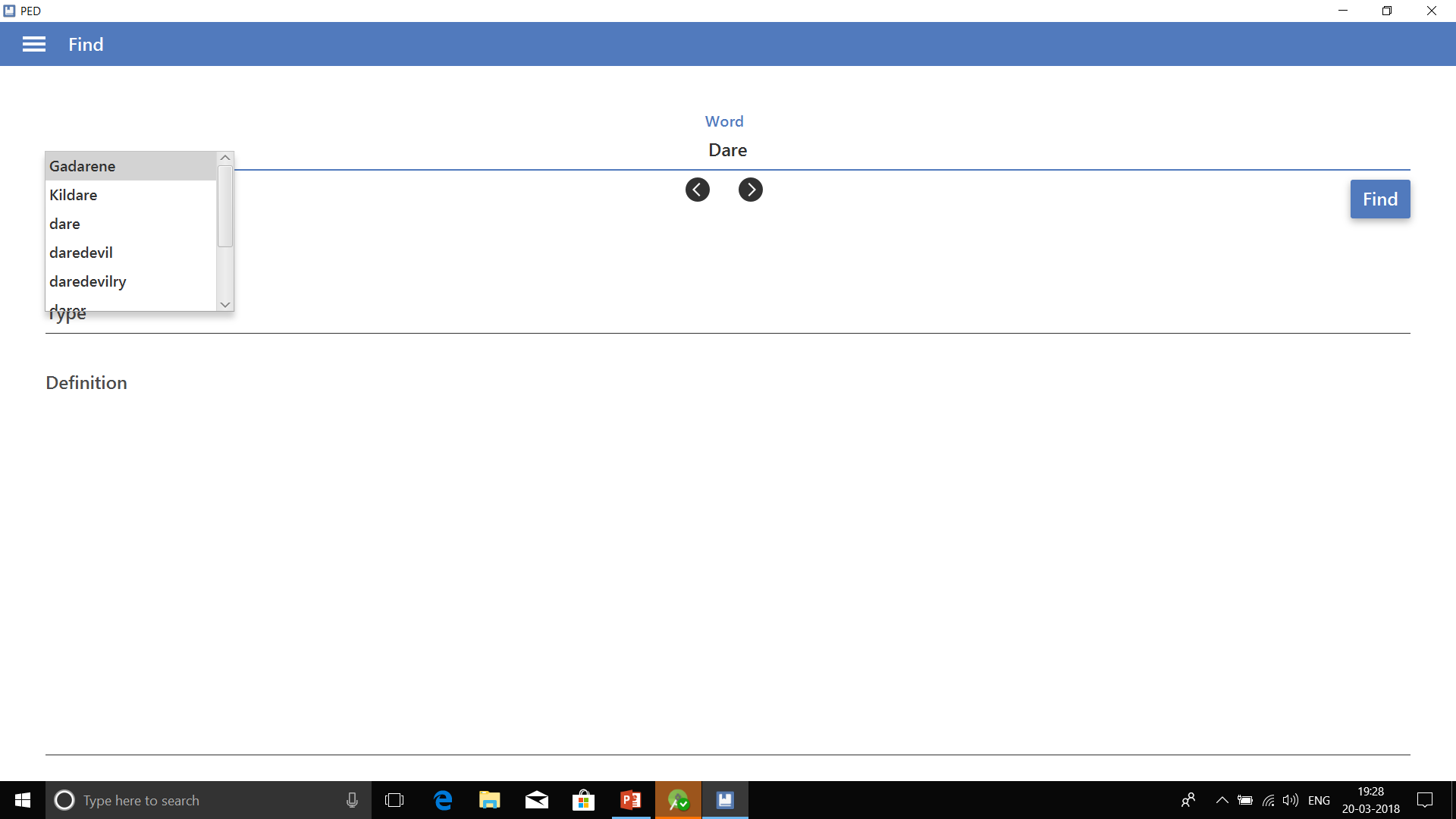
* Autofill suggestions would be handy.
* The application’s responsive time is high.
* Application is not responding while performing a backend task till that task gets completed.
* Application is not resuming from its pause state instead it’s restarting (for android app only).
* Application’s keyboard is not closing after processing a function (for android app only).
* Application is only providing single type for words with more than one type.

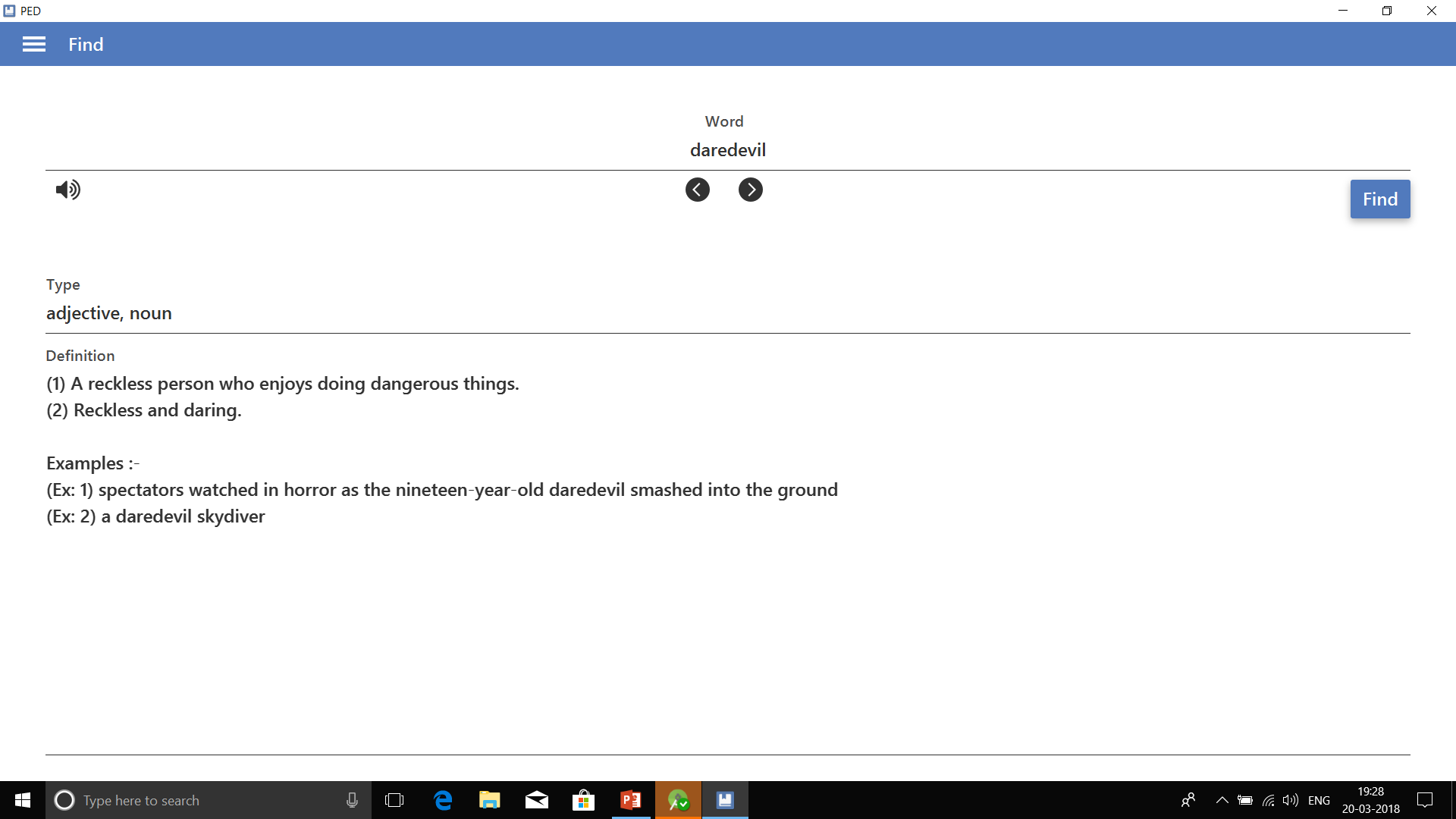
1. SCREEN SHOTS

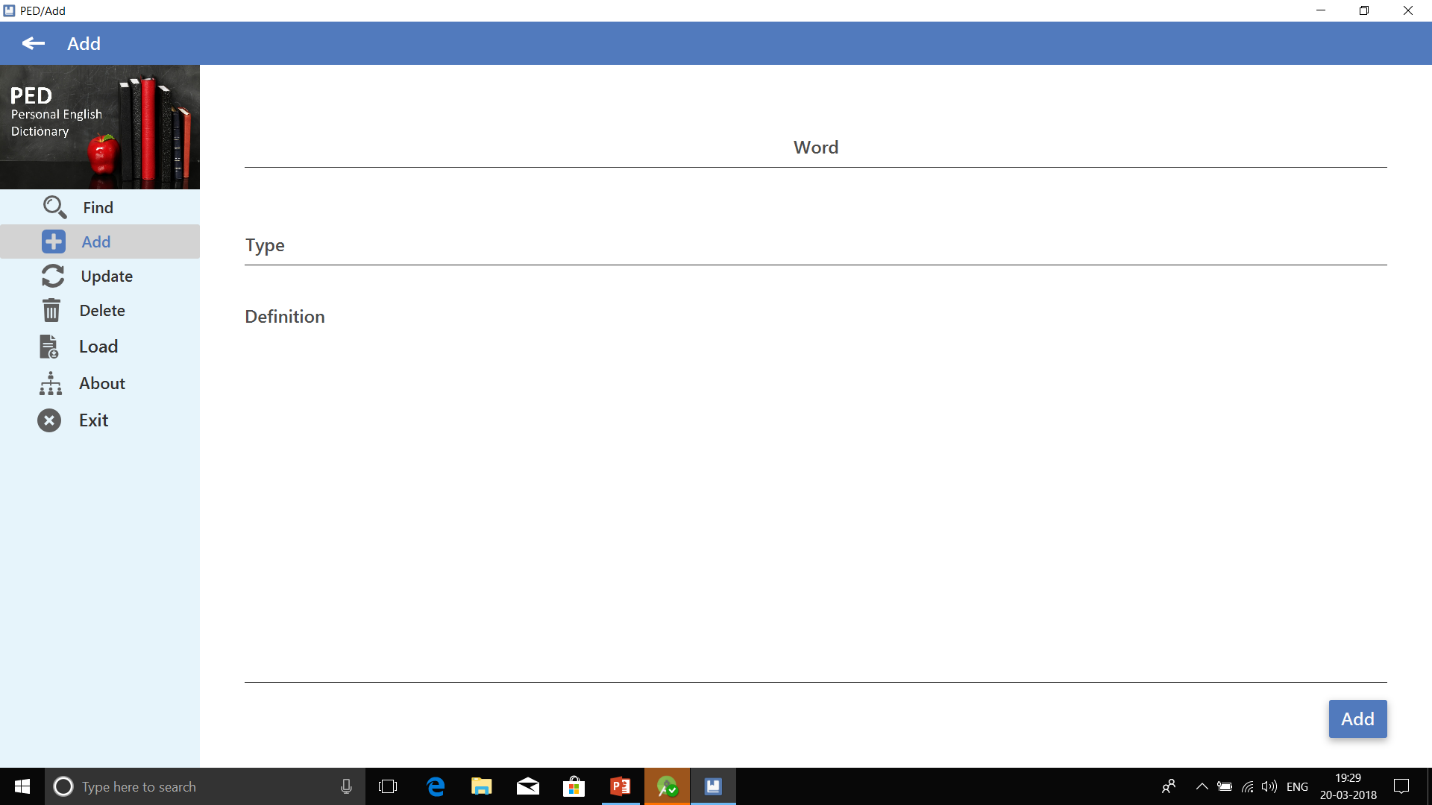
Desktop:

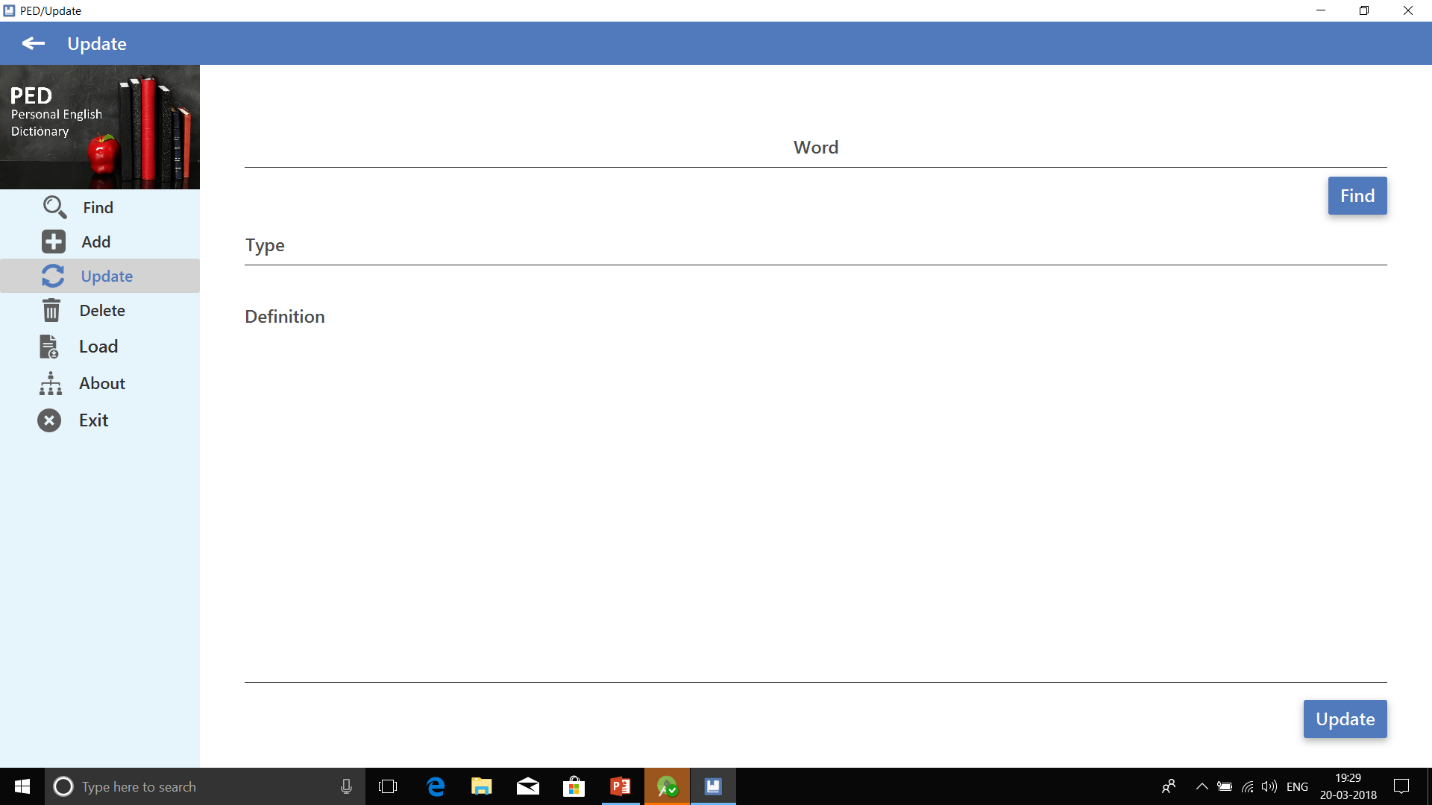


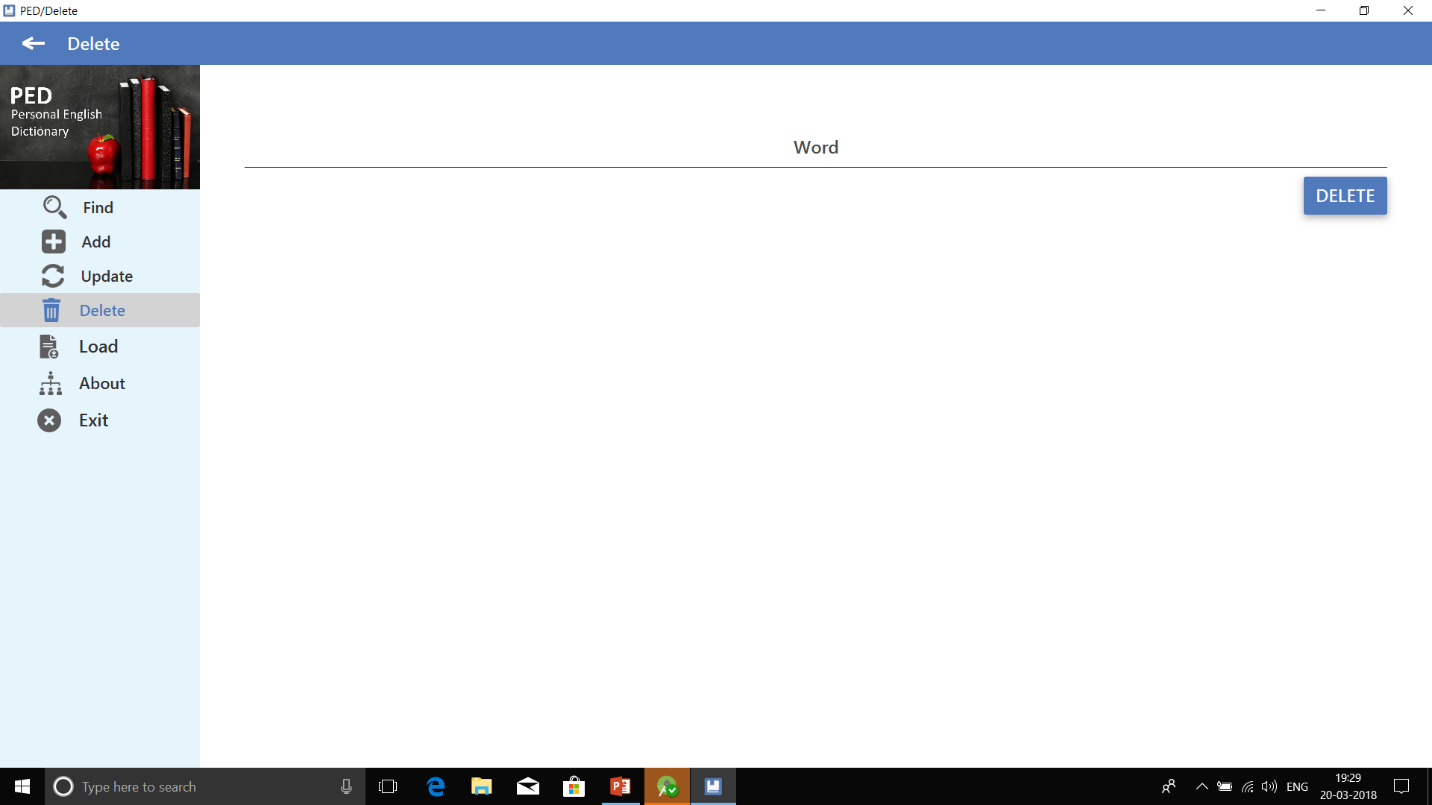


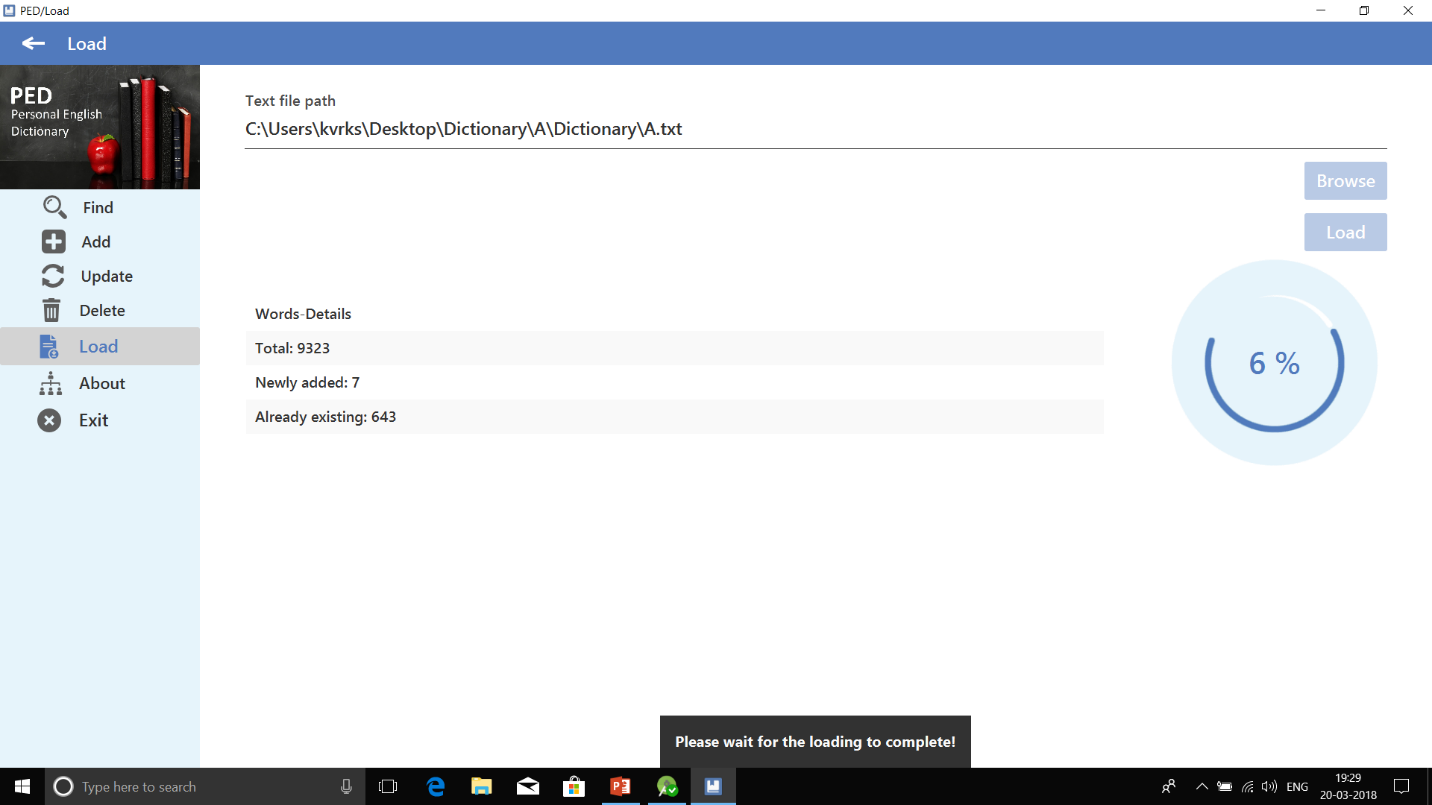




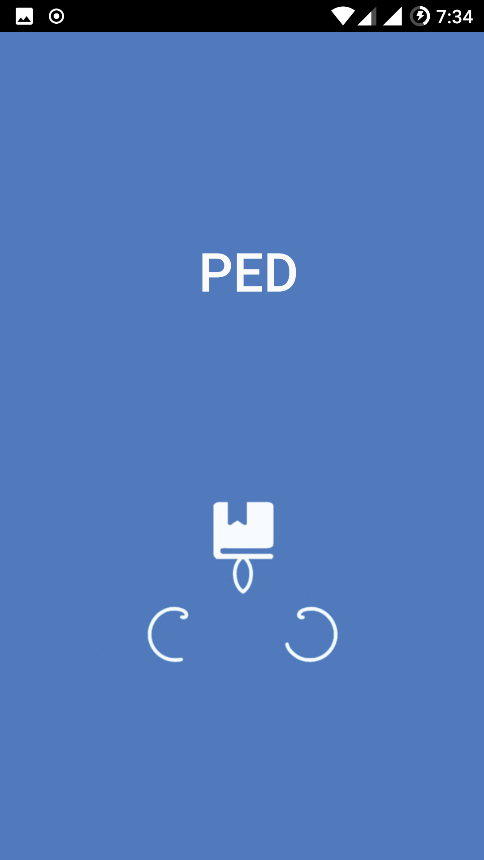


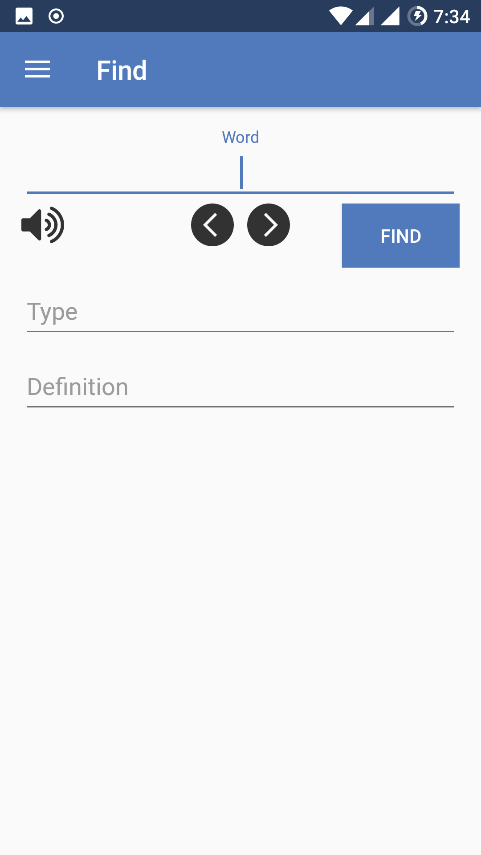


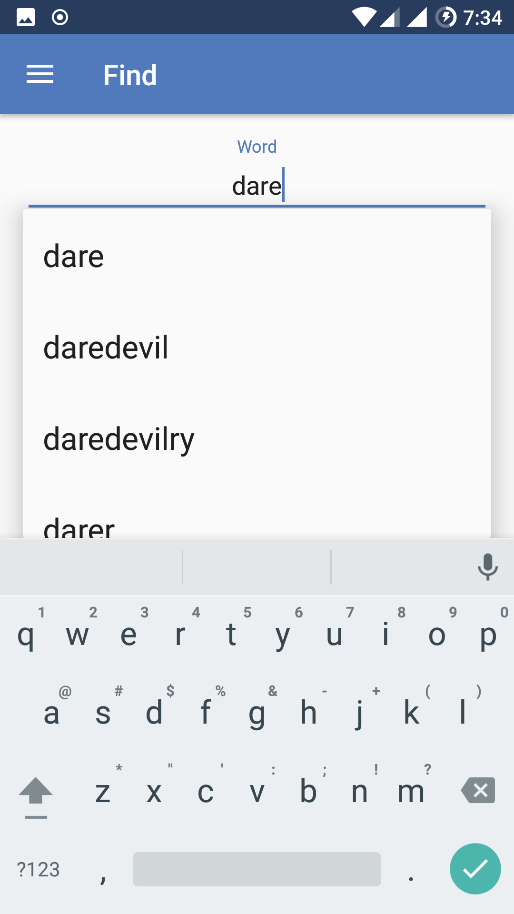


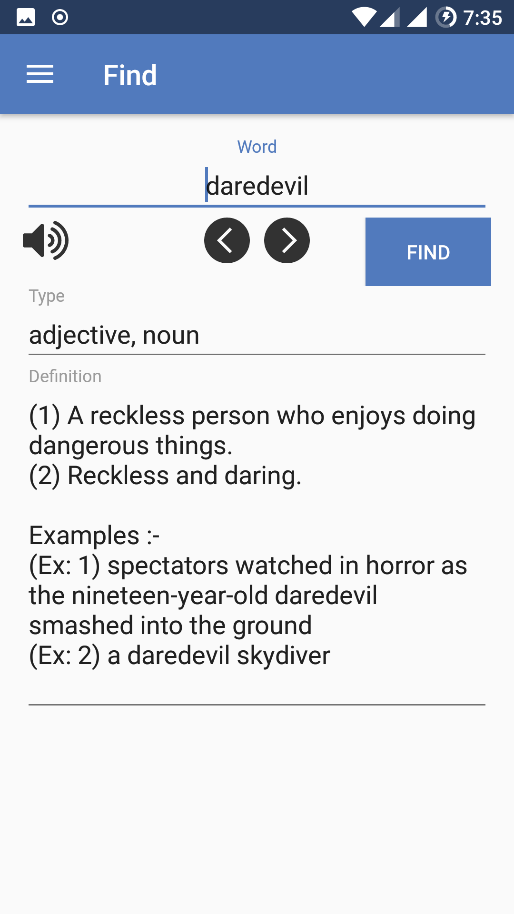


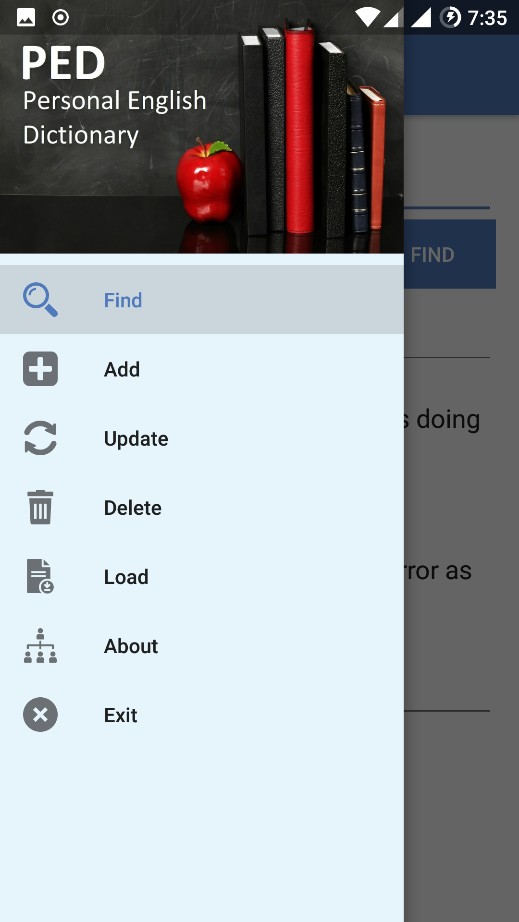
Mobile:

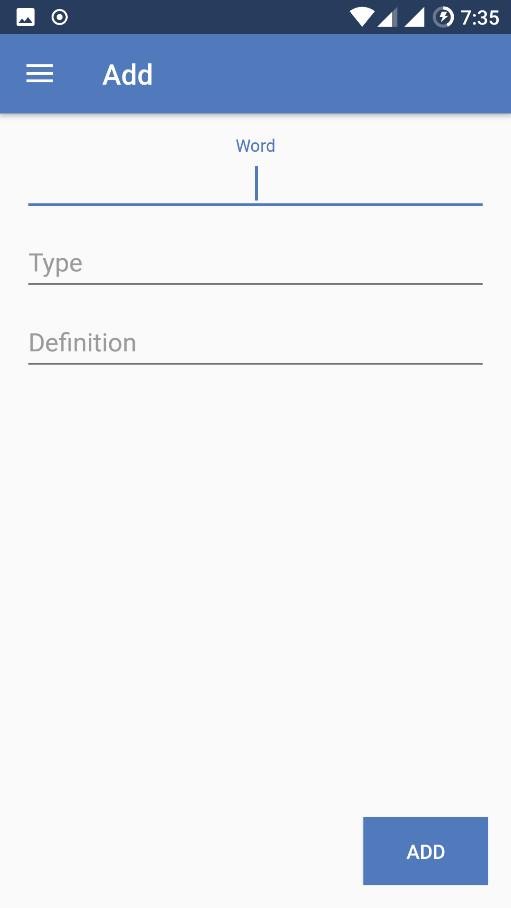


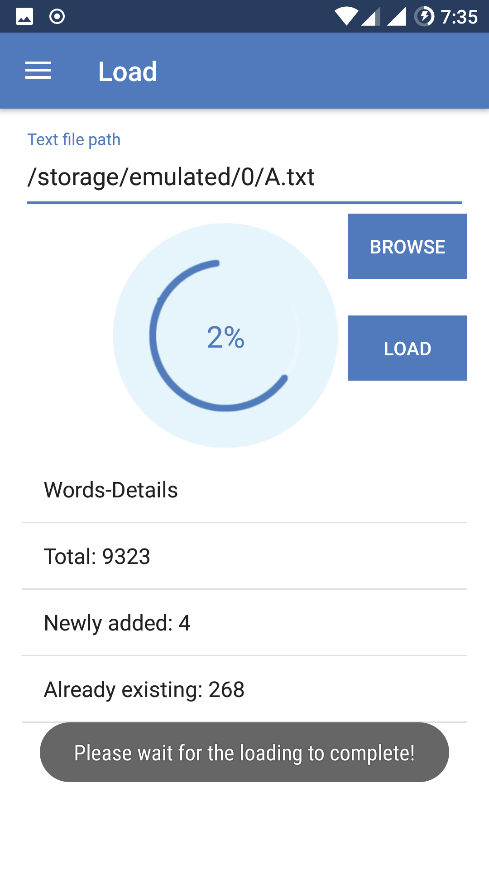












1. CONCLUSION

Based on our knowledge we had achieved all our objectives, satisfied all the requirements and processed a couple of feedbacks so we consider that we had met our design criteria.

We think the final tests we did went smoothly and we had no problems with any functionality of our application even during deployment. So we proudly say that we had successfully developed the world’s first dictionary application which allows its user to modify and expand his/her dictionary data at any place, at any time and independent on the internet.

This journey made us to learn a lot of new things by working with a couple of technologies, allowed us to practically implement most of the topics that we had learnt in our technical education and we proudly conclude by saying that we are able to achieve this outcome only because of our team work and good guidance.

1. FUTURE ENHANCEMENTS

The project has covered almost all the requirements. Further requirements and improvements can easily be done since the coding is mainly structured and modular in nature.

Improvements can be appended by changing the existing modules or by adding new modules like a detailed history of all the tasks that are performed by a user which we are planning on to provide from our application’s version 2.0 onwards.

We are even planning on to perform datamining to collect words and their details from a couple of different languages through web scraping in future if time permits and if some person or a group provides sufficient funding.

1. BIBILIOGRAPHY

## PED: Personal English Dictionary

## DB: Data Base

## SQL: Structured Query Language

## XML: Extensible Markup Language

## FXML: XML-based user interface markup language created by Oracle

## JavaFX: JavaFX is a scripting language based on Java

## JDBC: The Java Database Connectivity API

## PIP: The package management system for python

## Web Scraping:  [Data scraping](https://en.wikipedia.org/wiki/Data_scraping) used for [extracting data](https://en.wikipedia.org/wiki/Data_extraction) from [websites](https://en.wikipedia.org/wiki/Website)