Olympia Dictionary

Capstone Project Final Technical Report

Sergii Opryshko 7759145 sopryshko9145@conestogac.on.ca
Brahmpreet Singh 7845159 brahmpreet.official@gmail.com
Sharfaraz Ahamed Bachinvalapan Hameed 7834120 sharfrazq17@gmail.com
Anju Mathew 7804396 10mathew10@gmail.com

School of Engineering & Information Technology
Conestoga College Institute of Technology and Advanced Learning
August 1, 2018

Abstract

More than 1 billion people [1] in the world learn English as their second language. Most of them face a challenge of improving the language skills. Among all the mobile applications on the market that give definitions of the word or two-way translations there is no single app for both Android and iOS platforms that allows the users to manipulate the words they search for. In most cases the history contains some 50-100 entries which disappear upon adding new words. The user has no possibilities to organize his vocabulary.

A solution is a smartphone application that uses public APIs of popular dictionaries (Oxford, Longman, Cambridge, Macmillan, Collins, American Heritage dictionary [2] etc.) and allows to add labels to words, putting them (words) into categories, and allows users to check their knowledge through self-tests. The application can support text-to-speech, speech-to-text, contrast themes and other accessibility features. User can be motivated by getting higher score for giving correct answers for more words and competing with friends. As no application found on the market allows to meet one's preferences and ease the process of learning new words to any degree, a personalized and customizable solution is the key advantage of the project.

This report overviews the solution developed and summarizes its technologies and implementation.

Introduction

As usage of smartphone has become habitual nowadays, the project relies on the fact that average person has short windows of time multiple times a day and can use their smartphone to improve their language skills. Therefore, the main goal of the project is to give the user the ability to make short but frequent sessions of learning. To make it efficient, quizzes are added for the user to check the progress. To make it easier and more fun, score points are accrued when user gives correct answers. Also there is a possibility to compete with friends and share the results on social networks. Application requires Internet connection and can synchronize the profile and progress with the cloud.

Definitions

Dictionary API – a predefined set of GET/POST requests to a dictionary's web service. Application can support multiple dictionaries APIs to give more rich experience.

Application – Android application for searching words and managing their lists. Stores words and categories in local database, allows text-to-speech and speech-to-text functions as well as optical recognition and accessibility features.

Web server – written in JavaScript + NodeJS and running on a host in global network to manage users and their keys. Is required to deliver requests from users to online dictionaries and back.

Keyword – a word usually in an infinitive form that represents the entry in a dictionary. Multiple languages can be supported if the dictionary API provides them.

Category – a user defined category with an arbitrary name that allows grouping words into them. Can be any user defined text and are created to their own needs or interests – sounds, emotions, animals, plants, verbs, books etc. Words can naturally belong to multiple categories. Category names can be changed at any time without affecting the words in them.

Word card – a view in the application that contains the word, its definition and categories it belongs to.

Vocabulary – a single storage in the application for selected by user words and their categories. Can be saved to a database and restored.

User – unique client of Olympia application. Must be registered, logged in and have a valid application key in order to use the app.

Login token – a unique key that is hold by application after successful login. If missing from the phone, the user needs to login on the server to continue using the app. Reflects the idea of a session in a web browser.

Olympia application key – a unique key that allows the user to use Olympia services. Unlike login token can be managed by administrators to only allow users who subscribed for paid services.

Administrator – a privileged user on the web server who can issue keys, assign them to users and delete the keys.

Local database – SQLite database (Room wrapper) on Android devices. Connected to application to allow backup and restore operations.

Server database – MongoDB database to store users, their keys and administrators. Connected to server part only.

Quiz – a sequence of tasks where the user has to spell correctly word to given definitions. The more answers are correct the higher is the score. The score +10 is given for every correct word and -10 for every incorrect answer. Therefore, in order to get better score, the user has to answer at least half of the questions correctly.

Technologies

Agile methodology [3] that was used in the project helped to develop the product in iterative and incremental way, so that the changes from customer feedback were received every two weeks and then implemented.

Concurrent versions system (CVS) [4] – collaborative work on the project would be impossible without tools that allow working on the code in parallel.

Software-as-a-service [5] model was chosen because it allows easier maintenance and more security of personal data. It becomes more and more popular nowadays as Internet coverage and speeds increase.

Client-server [6] relationship was used to provide the services in an organized and consistent way. This model allows to improve services by only updating the server.

RESTful API [7] – used in modern web technologies. Eases the development of interactive and fault-safe web applications.

Optical Character Recognition [8] – useful technique to transform text on images in actual text to add one more input type.

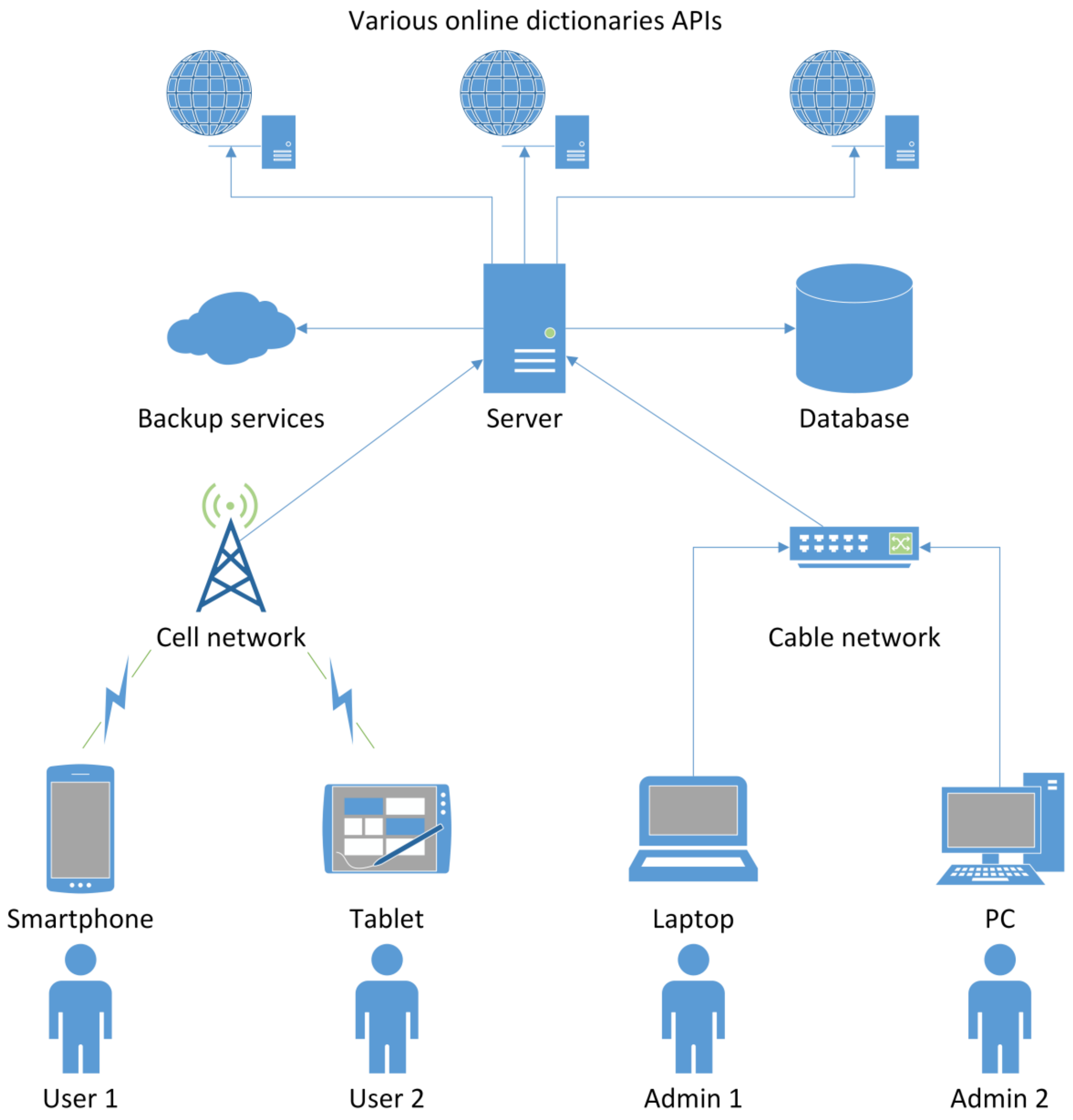
Text-to-Speech and Speech-to-Text [9] – are very advanced nowadays and do not require much computational power. Used in the project to add some accessibility.

High level design of the system

Olympia adds an extra layer between a customer (smartphone user) and online dictionaries. The application on the phone interacts with Olympia server to get the word definition. The server in its turn takes them from API of the dictionary. To get access to the APIs of third-party dictionaries Olympia must have valid keys. But caching on the server allows to save on numbers of requests. Olympia server also has connection to the cloud backup server and a database, where information about the users is stored. That allows to maintain decent level of security, flexibility and robustness. User can access his profile from different devices and have his vocabulary loaded on a new device in one click.

Full backup is made automatically every day with incremental backups every hour. A RAID 5/6 system is used as a storage. Olympia software is run on a dedicated server in a reliable datacenter, while backup is made to a cloud server AWS from Amazon.

An application must have access to the Internet via either Wi-Fi or cell network. Otherwise the request to the server will not be made and the user will see an error message. Administrators can access the administration panel only from browser using SSL encryption.



Server part

Server backend is written in JavaScript and NodeJS. It mostly handles POST and GET requests to the server, and actively uses Promises for data retrieval from the MongoDB database. Frontend for administration portal is powered by HTML 5 and CSS and uses the third-party Bootstrap and DataTable libraries for visualizing data in tables with sort, filter and select support.

List of packages used on the server side (from NPM repository [10]):

- Express 4.16.3 – fast, minimalist web framework for routing.
- Mongodb 3.1.0 – the official MongoDB driver for Node.js. Provides a high-level API on top of mongodb-core for end users.
- Mongoose 5.1.7 – a MongoDB object-modeling tool designed to work in an asynchronous environment.
- Bcrypt-nodejs 0.0.3 – library used for hashing password in the project.
- Body-parser 1.18.3 – Node.js body parsing middleware. Parses incoming request bodies in a middleware before the handlers, available under the req.body property.
- Connect-mongodb-session 2.0.3 – module that only exports one function that maintains the session with the MongoDB.
- Nodemailer 4.6.7 – allows to send emails directly from NodeJS. Used for password recovery and changing emails for the user.
- Passport-local 1.0.0 – this module allows to authenticate using a username and password in our administration portal.
- Smtp-connection 4.0.2 – SMTP client module. Connect to SMTP servers and send mail with it.

Application part

Application is written in Java v.1.8 for minimal API level 21 (Android 5.0 Lollipop and higher) and built with Gradle. Views layouts and all drawables are marked up in XML files. Custom font (Georgia) and themes are added and used.

Hierarchy of activities:

- SplashActivity
  - MainActivity
    - WordsListActivity
      - WordCardActivity
      - SettingsActivity
        - ChangeNameActivity
        - ChangeEmailActivity
        - ChangePasswordActivity
      - LegalActivity
      - AboutActivity
      - QuizActivity
      - StatisticsActivity
      - HowToActivity

Interaction with remote server is done asynchronously using GET and POST requests with the help of Retrofit library. For storing local parameters of the app shared preferences are used. Access to Room database on the phone is available through a RoomDbFacade interface. Application uses MVC and observer patterns and stores all the data in a singleton object. User's vocabulary is stored in a hash map. It allows to include words to multiple categories and the categories to have any number of words. Categories can be renamed without losing link with words.

The application requires several permissions for operation. Permissions are divided into several protection levels: normal, signature and dangerous. The protection level affects whether runtime permission requests are required. *Normal* permissions cover areas where the app needs to access data or resources outside the app's sandbox, but where there is very little risk to the user's privacy or the operation of other apps. The system grants *signature* app permissions at install time, but only when the app that attempts to use a permission is signed by the same certificate as the app that defines the permission. *Dangerous* permissions cover areas where the app wants data or resources that involve the user's private information, or could potentially affect the user's stored data or the operation of other apps. To use a dangerous permission, the app must prompt the user to grant permission at runtime.

The list of permissions used in the app:

Permission	Security level
ACCESS_NETWORK_STATE	normal
ACCESS_WIFI_STATE	normal
INTERNET	normal
CAMERA	dangerous
READ_EXTERNAL_STORAGE	dangerous
WRITE_EXTERNAL_STORAGE	dangerous

Online dictionaries APIs usage

All online dictionaries require the key to use their databases. For educational purposes it was possible to get a trial key to develop the sample app. However, to use APIs in a business application the key must be purchased. Usually there are limitations on the number of requests per day. In addition, there is a restriction in terms and agreements with the dictionary that forbids storing of data received from their servers for more than 24 hours. Considering these limitations, it is reasonable to cache requests on the Olympia server not to exceed the number of requests per day.

A valid key is required to send requests to the API. An application ID must be passed as a parameter in a request header. Additionally, a special token of the session may be required for some APIs. Therefore, the Olympia solution considers these differences and provides a generic interface for the mobile application, leaving these transformations under the hood of the server.

Examples of GET requests of Oxford dictionary:

- /entries/{source_lang}/{word_id}
- /search/{source_lang}
- /stats/frequency/word/{source_lang}/

In case of successful request a JSON object is returned and needs to be parsed. It has a complex structure that varies from dictionary to dictionary. For all the classes of responses Olympia server creates different objects inherited from one parent, so that the mobile application receives the response from the server in one standard format and does not need to convert anything. In most cases a word definition contain one or several entries for different parts of speech, pronunciation, transcription, audio file, synonyms and examples of usage. All this information is shown in what is known as word card.

Development tools

Android Studio [11] is a standard IDE for developing Android applications. Version 3.1.3 was used with Android SDK 27 and Java v.1.8 support. Minimal API is 21, so the application only works with Android 5.0 Lollipop and higher.

Cloud9 [12] is a popular tool for web development combined with some limited hosting capabilities. It allows to develop, run and debug applications in multiple languages. Olympia server is written in JavaScript with NodeJS libraries included. The libraries support ‘Promises’ which are asynchronous events. They allow to communicate with applications on dozens of client devices (smartphones) simultaneously.

LucidChart [13] – a simple tool with solid capabilities for creating diagrams and charts. The database diagrams were created in it.

Conclusions

The solution of the problem is elegant, extendible, robust and secure. Most advanced modern technologies as well as efficient and thoroughly tested third-party libraries are used to provide the best user experience, flexibility and accessibility. The application that was developed can be used in educational purposes by international students, teachers, linguists, tourists, businesspersons and all those who want to drastically improve their English language skills in short time. The server infrastructure is minimal to support all the features of the app, and does not require much funds to build up. The application can be polished to a point when it complies with Google UI guidelines and be published in Google Play store.

References

[1] How Many People Learn English? - https://www.thoughtco.com/how-many-people-learn-english-globally-1210367

[2] Dictionaries:

- Oxford - https://www.oxforddictionaries.com/
- Longman - https://www.ldoceonline.com/
- Cambridge - https://dictionary.cambridge.org/
- Macmillan - https://www.macmillandictionary.com/
- Collins - https://www.collinsdictionary.com/
- American Heritage - https://ahdictionary.com/

[3] Agile methodology - https://en.wikipedia.org/wiki/Agile_software_development

[4] Concurrent versions system (CVS) - https://en.wikipedia.org/wiki/Concurrent_Versions_System

[5] Software-as-a-service model - https://en.wikipedia.org/wiki/Software_as_a_service

[6] Client-server architecture - https://en.wikipedia.org/wiki/Client%E2%80%93server_model

[7] RESTful API - https://en.wikipedia.org/wiki/Representational_state_transfer

[8] Optical Character Recognition - https://en.wikipedia.org/wiki/Optical_character_recognition

[9] Text-to-Speech and Speech-to-Text - https://en.wikipedia.org/wiki/Speech_synthesis

[10] NPM repository - https://www.npmjs.com/

[11] Android studio - https://developer.android.com/studio/

[12] Cloud9 - https://c9.io/

[13] LucidChart - https://www.lucidchart.com/