Vocabulary Learning Client with JavaFX

User Guide and Software Documentation

Bela Bothin, Maximilian Engelmann,
Lukas Radermacher, Sebastian Raßmann

Software Documentation as a partial fulfillment
of the module "Praktikum Objektorientierte
Softwareentwicklung" (BA-INF 025)
in the Bachelor degree Computer Science

University Bonn
Germany
July 2020

Contents

1 User Documentation **1**

1.1 Requirements . . . 1

1.2 Login . . . 1

1.3 Settings . . . 1

1.4 Vocabulary entry . . . 1

1.5 Managing Vocabulary . . . 2

1.5.1 Editing and Deleting . . . 2

1.5.2 Adding entries . . . 3

1.6 Learning . . . 3

1.6.1 Learning . . . 3

1.6.2 Goals . . . 3

2 System Documentation **5**

2.1 Software Structure and Components . . . 5

2.2 Tools, Frameworks, and Libraries . . . 6

3 Project Documentation **7**

3.1 Project organization and testing . . . 7

3.2 Project Responsibilities and Contributions . . . 7

1 User Documentation

The developed application implements the api.voc5 API offering a client-side graphical user interface (GUI) to the functionality of creating, managing, and learning user-specific vocabulary lists.

1.1 Requirements

This application is a java program, hence, requires an installation of java (version 13 or higher.) Also internet access is required while using the application.

1.2 Login

The application stores data remotely on a api.voc5 implementing server at the selected address. In order to manage the personal data an user-account needs to be created by inserting a valid email address (needs to contain a '@') and password, and pressing "Register". **User data is not stored safely - Make sure to not use real data!**
To access and existing account, email and password need to be inserted and "Login" pressed. Use the "Remember me" function to automatically login upon the start of the application.

1.3 Settings

The settings symbol allows the user to access the application settings. Currently, only changing the language of the application between English and German is possible.

1.4 Vocabulary entry

Each vocabulary consists of

- Question
  - The translation or description in the original language.

- Answer
  - The correct word in the target language, thus, the word to learn.
- Language
  - The language in which the word is learned.
- Phase
  - The phase describes the state of learning of each individual vocab: Newly added vocabulary start at the lowest phase and progress towards the maximum phase with correctly given answers.
    Note: Differing from the project requirements, the lowest phase was set to 0 and the highest to 4, as this is defined by assigning phase 0 to new entries by the API!
    Hence, new entries start at 0 and are assumed to be perfectly learned at phase 4.

1.5 Managing Vocabulary

The Vocabulary pane allows for adding new and editing existing vocables. Changes are automatically synchronized with the server and, hence, saved.

1.5.1 Editing and Deleting

It is possible to search for specific vocables by entering a search topic into the text field in the upper left corner and pressing the ”enter” key or the ”Search” button on the right hand side of the field. By pressing the ”Reset” button the filters will be reset and the whole vocabulary will be shown again. To edit the vocabulary, enable editing by pressing the ”Edit Mode” button: It is now possible to change the answer, the question, and the language of each entry in the table below by performing a double-click on the respective cell. Besides enabling editing, the previously hidden buttons ”Add” and ”Delete” are shown. Also, a column of check boxes (”Select Column”) will appear on the right side of the table. To delete selected vocables press the ”Delete” button after selecting the respective entries from the table. Pressing the ”Add” button opens the ”Add” dialog.

1.5.2 Adding entries

This Scene is responsible for adding new vocabulary. It consists of text fields for the answer, the question, and the language of the new vocable. By pressing the "Confirm" button in the lower left hand corner the input is converted, saved and added to the table. The entry will also be permanently stored on the server. The text fields are automatically cleared and can accept a new input. The "Cancel" button, located in the lower right hand corner, cancels all input and closes the "Add" dialog.

1.6 Learning

1.6.1 Learning

In order to start learning, switch to the learning screen. If accessed directly, all of the user's vocabulary will be learned (see section 1.6.2).
In the screen, a question fields from one of the user's vocables will appear together with a field for the answer. Enter the assumed translation and press solve, in order to show the right answer. The correctness of the user's answer is classified into 3 categories: Completely right, partially right (e.g. containing single typos), and wrong yielding a score of 3,1, and 0, respectively. A completely right answer will further increase the phase of the vocable by one, whereas a wrong answer will decrease it by one. A partially right answer will keep the vocable's phase unchanged. Phase changes will also be synchronized with the user's data stored on the server.
If the question was assumed wrong or partially wrong (e.g. due to ambiguity or synonymous translation), the user can also correct the scoring by pressing "Manual correction" and selecting the right scoring category.
After going through all vocables selected for learning, a overview of the learning results is shown.

1.6.2 Goals

In order to offer the option to only learn individual words or word in defined languages, the *Goals* screen can be used. This pane supplies a list of all of the user's vocabulary at a phase lower than four. By pressing on "Filter Languages", the shown vocabulary can be subset to only specific languages. To ignore vocabulary from learning, the checkbox of each vocab can

be unchecked, either for each entry manually, or by clicking on ”Deselect all”. The list of vocabulary can be sorted by any property by pressing on the respective column title.
Upon pressing ”Start Learning” and ”Start Learning in Random Order” all selected vocabulary can be learned in the shown or random order, respectively.

2 System Documentation

2.1 Software Structure and Components

The initial Scene of the program is set in the *Main* Class, which implements Scene launching and handles Scene switching.
The Login and Register Pane is encoded in *FXLogin.fxml* and controlled by the *ControllerLogWindow* Login or register requests triggered by the user's insertions of eMail and password to the respective text fields are handled via the *APICalls* class.
Upon successful log-in in or registration the controller *ControllerMainScene* is initialized and loads *MainScene.fxml*. Also, the global variables of the program are initialized in the *Variables* class, which is used for inter-scene data exchange and communication. I.a. the user's vocabulary is retrieved from the API and stored locally to allow for a faster access. Additionally, the user's password is encrypted using the *Crypt* class via AES encryption so eMail and the encrypted password are stored as *logInfo.json*. Hence, the password can be restored and decrypted using the set key, which is initially generated using the users Mac-Address.
The *MainScene* contains a navigation bar with the logo, a title bar and a *BorderPane* for the main content. It sets the stage for all the other components and windows of the program, by having the before mentioned *BorderPane* which is able to be set to different controllers extending *AnchorPane*. These are:

- **ControllerVocList**: Displays vocabulary and uses the *Variables* class to get the user's initial list of vocabulary by calling *APICalls*
  - **ControllerVocabAdd**: Displays a stage where the user can add new entries
- **ControllerGoals**: The user can select which vocables to learn. For an instant response it first uses the local copy of the vocabulary but also triggers a API call to the update the list
- **ControllerLearning**: Handles the learning processes by iterating through the entries selected in the "Goals" menu or alternatively all entries. Correctness of the inserted answer is calculated by the Levenshtein distance between inserted and correct answer: A distance of 0 is graded as fully correct, 1 as partially correct, and greater than 1 as wrong.

All answers are pushed to list to show at the end of learning process and the phase is edited by an asynchronous API request.

- **Controller settings**: It contains the user settings (currently only the menu language). It features an "Apply" button which uses the *SettingsChanger* class to store the settings as "Settings.json". This file can be loaded by the *LocalizationManager* class to restore the settings after the start of the application.

The *SettingsChanger* class needs be called to reload the whole current stage to apply Settings.
The *LocalizationManager* class is used to store all translations for all the elements of the program in each language, which in turn is stored in another HashMap containing all the languages. The HashMap can then be accessed with a method to get different entries and use their values to set labels and descriptions.

The *User* class is used to pass User information around, especially for the title bar in *MainScene*.
The *Vocab* class is used to provide a structure for formatted JSON bodies from the API. It is used as a (super) class for all objects representing vocables throughout the program. Only some of its fields are marked with *@Exposed*, since ID and Phase are not needed for posting vocables to the API.
Instances of *VocabSelection* are used in *Goals* and *VocabSelection*, in order to extend the *Vocab* class with a check box to select individual entries.

2.2 Tools, Frameworks, and Libraries

- The project is based on **Gradle** to manage dependencies and provide a structure
- The **JavaFX** was used to realize all GUI elements
- **OkHttpClient** is used to for connection to the API
- **Gson** is used to transfer objects between the client and the server as defined in the API
- **Jackson** is used to create lists from Json strings
- **Filewriter** to write Json to the file system
- **Lists** and **HashMaps** are used to store and pass Information and Objects

3 Project Documentation

3.1 Project organization and testing

As a starting point, the main Scene and Controller handling server selection and user login, as well as the functionality to set and switch between scenes was implemented.
On the top of these fundamental construct, the login screen was adapted from the previous exercises was adapted and the rest of the application function was split between the 3 main screens ("Vocabulary", "Learning", and "Goals") and style and design was added by extending the .css file and adding icons, designs etc.
In each developmental area a GUI implementing the required features was designed and the required internal data-structures, interfaces between screens, and API communication were determined and implemented as needed.
The generated code was manually tested by the specified use-cases.

3.2 Project Responsibilities and Contributions

The project was split amongst the frames within in the application. Responsibilities were distributed as following:

- Bela Bothin was responsible for
  - Setting up *Main* class and providing function to change scenes
  - Registration frame: Login and register screen (adapted from previous exercise assignments), server selection and connection, waiting animation, Remember-me function, and password encryption
  - Application design: Usage of css file and addition of the icons, with the structure and design of MainScene containing the NavigationBar and TitleBar
  - API calls (used for Vocabulary learning and management)
- Maximilian Engelmann was responsible for
  - Vocabulary learning: Learning core functionality and string operations like checking the users answer for mistakes

- Lukas Radermacher was responsible for
  - Vocabulary management: Accessing, adding, searching, editing, and deleting Vocabulary entries
- Sebastian Rassmann was responsible for:
  - Setting up Gradle project and gitignore
  - Selection of vocabulary to learn ("Goals"): Selecting vocabulary to learn and filter by language - the code for the vocabulary *TableView* with check-boxes (*VocabSelection* class and state accessing logic) was re-used for vocabulary management
  - Documentation

Usage of FXML and the respective controllers, internationalization, and asynchronous API calls were realized by each developer individually.
Remaining jointly used functionality, especially, the *Variables* class used for inter-screen communication was developed jointly.