How to Use this Template

1. Make a copy [File → Make a copy...]
2. Rename this file: "**Capstone_Stage1**"
3. Replace the text in green

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"

GitHub Username: Mahmoud-S-Ali

VoKeeper

Description

VoKeeper is a FREE translation and a well organized vocabs keeper app. VoKeeper uses your phone's Internet connection (4G/3G/2G/EDGE or Wi-Fi, as available) to enable you translate text from one language to another,
as well as the ability to create dictionaries where you can save any vocab in an organized and memorable way by

putting them into categories and adding a photo, a voice record or a note to any saved vocab.
WHY USE VOKEEPER:

Translation: VoKeeper uses your phone's Internet connection (4G/3G/2G/EDGE or Wi-Fi, as available) to enable you translate text from one language to another with no fees at all.

Saving Vocabs: VoKeeper creates a local database in your phone to enable you save any vocab you translate or manually add (No internet connection needed) in an organized way by inserting them in their specified dictionaries and categories which you can add manually or being added automatically.

Vocab Information: You can add some information to each vocab you save in a form of a photo you select from your gallery or using your camera, a voice record or a note. These information help you memorize your vocabs easily.

Notifications: As a practice to help you memorize your vocabs. VoKeeper chooses a random vocab from your saved vocabs and sends you a notification every period of time you specify asking the meaning of the chosen vocab. This feature can be disabled from the settings.

Intended User

This app is intended for anyone searching for translation, people learning new languages or those who face difficulties in memorizing new vocabs.

Features

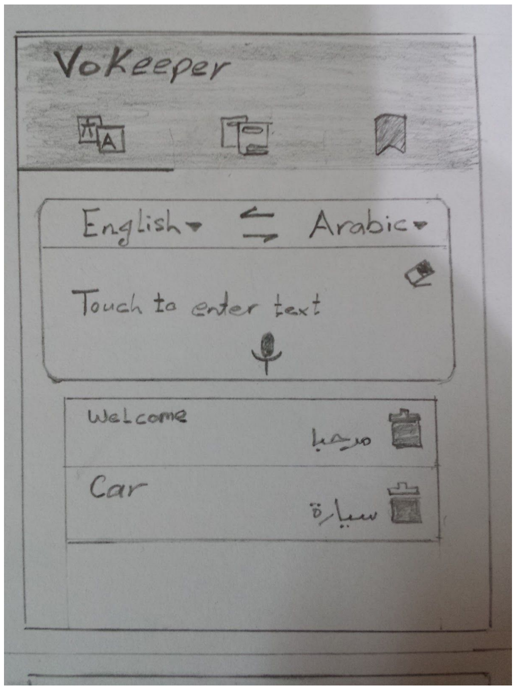
VoKeeper's Features:

- Text translation
- Saving translated text
- Attaching a photo to the saved text
- Adding a voice record to the saved text
- Adding a note to the saved text
- Sending notifications

User Interface Mocks

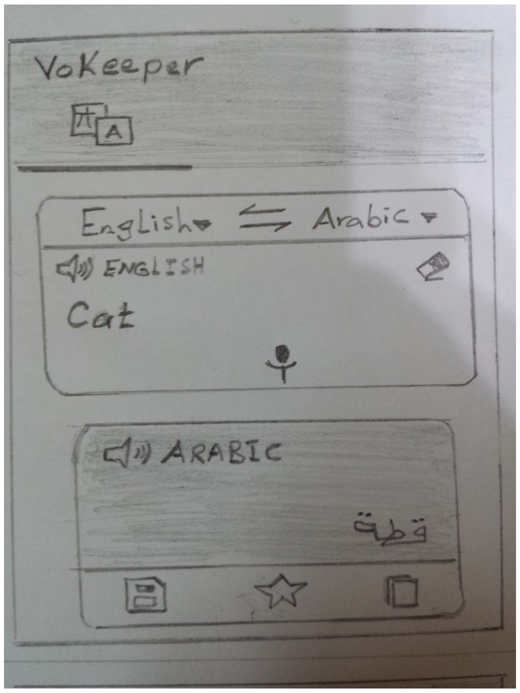
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

Screen 1



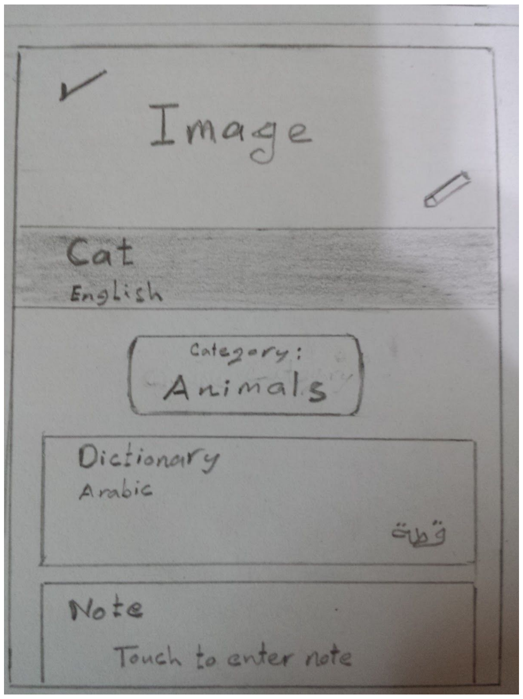
This is in Translation Fragment and this is where a user can enter a text to be translated or view the history of the recent entered texts.

Screen 2



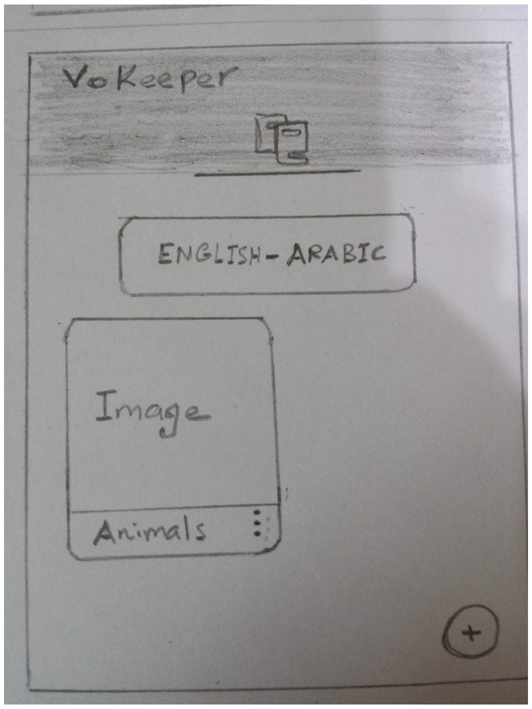
This is in Translation Fragment as well but when the enters a text to be translated

Screen 3



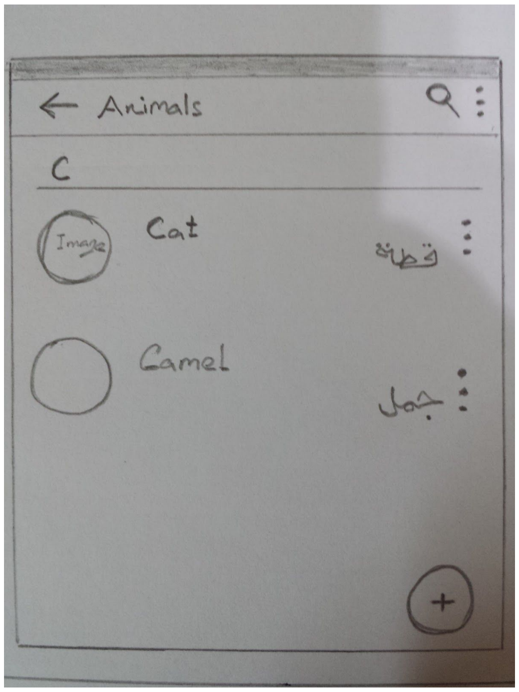
This is Save/Edit Activity and it shows up when the user presses the save button. Here is where the user can add information to the text entered and save it

Screen 4



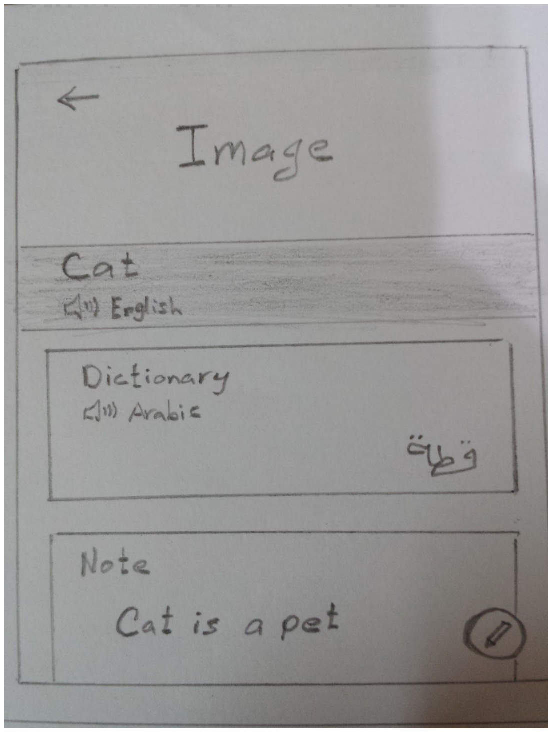
This is the Dictionary Fragment and it's where the user can select a dictionary. Each dictionary has a collection of categories created by the user.

Screen 5



Selecting any category will open this Vocabs Activity and it shows all the saved vocabs inside the selected category.

Screen 6



Selecting any vocab will open this View Activity in a form of a ViewPager where the user can move between vocabs by swiping left or right

Key Considerations

How will your app handle data persistence?

- App will use Content Providers to handle data fetched from microsoft translation api
- App will use SharedPreferences to save the last selected dictionary name and show it on app's start

Describe any corner cases in the UX.

For example, how does the user return to a Now Playing screen in a media player if they hit the back button?

- If user chose to save a vocab without creating a dictionary, a new one will be added automatically
- If user didn't choose a category for the vocab, it will be saved in a "unspecified" category and it will be created automatically if it's not there

- User can add a vocab manually. He doesn’t have to translate it first then save it
- For API versions >= 23, users will be asked for permissions at runtime, not when they install the app as in API versions before 23

Describe any libraries you’ll be using and share your reasoning for including them.

- Glide: to handle loading and caching images (https://github.com/bumptech/glide)

- Schematic: Automatically creates a ContentProvider backed by an SQLite database . It’s very simple to use over building the database manually (https://github.com/SimonVT/schematic)

- Microsoft java translator api: Provides a java wrapper around the translator microsoft api (https://github.com/boatmeme/microsoft-translator-java-api)

- EventBus: Send events across fragments and activities (https://github.com/greenrobot/EventBus)

- CircularImageView: To realize a circular imageview (https://github.com/lopspower/CircularImageView)

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

- Research on similar apps and learning from their experiences
- Setting up a feedback and verify the need of such an app as well as getting new ideas from users
- Creating the design architecture
- Research on existed translation APIs, comparing between them and choosing the most suitable
- Specifying minimum and target sdk versions
- Configure libraries
- Build a database scheme

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity
- Build UI for TranslationFragment
- Build UI for DictionaryFragment
- Build UI for BookmarksFragment
- Build UI for EditActivity
- Build UI for ViewActivity
- Build UI for VocabsActivity
- Build UI for SearchActivity
- Build UI for SettingsActivity

Task 3: Translation Fragment Implementation

In this fragment, user should be able to:

- Specify source and destination languages
- Enter a text to be translated either by typing or by voice
- View result translation
- Listen to the entered or the result text pronunciation
- View, delete or select history of a previously entered text

Subtasks:

- Create translation layout
- Create history list item layout
- Create history recyclerview
- Create TranslationFragment.java
- Create history recycler adapter
- Create history loader
- Create TranslationService that fetches data from the translation API
- Implement a method that uses tts to handle pronunciation
- Implement a method that uses RecognizerIntent to handle voice inputs
- Integration
- Testing

Task 4: Dictionary Fragment Implementation

Describe the next task. List the subtasks. For example:

- Create layout
- Something else

In this fragment, user should be able to:

- Create, delete or select a dictionary
- Create, delete, edit or select a category
- View categories in each dictionary
- Search for a specific vocab in the selected dictionary

Subtasks:

- Create dictionary layout
- Create category list item
- Create category recyclerview
- Create DictionaryFragment.java
- Create dictionary recycler adapter
- Create category recycler adapter
- Create dictionary loader
- Create category loader
- Create DictionaryService to handle adding or deleting dictionaries in database
- Create CategoryService to handle adding or deleting categories in database
- Integration
- Testing

Task 5: Edit or View Vocab Implementation

Editing and viewing a vocab implementation will be very similar except that in editing, user will be able to change the vocab information.

Through this, user will be able to:

- Attach a photo
- Record and save a voice record
- Modify the meaning
- Add a note

Subtasks:

- Create edit layout
- Create EditActivity
- Create VocabsService to handle adding or deleting vocabs in database
- Create view layout
- Create ViewActivity
- Integration

- Testing

Task 4: Vocabs Activity Implementation

This is where user will be able to view saved vocabs in a specific dictionary and category

Subtasks:
- Create vocabs layout
- Create vocabs recyclerview
- Create vocabs list item
- Create VocabsFragment.java
- Create vocabs loader
- Create vocabs recycler adapter
- Integration
- Testing

Task 5: Search Activity Implementation

In this activity, user should be able to search any vocab according to a specific dictionary or category

Subtasks:
- Create search layout
- Create search recyclerview
- Create SearchActivity.java
- Create search loader
- Integration
- Testing

Task 5: Handle Error Cases

This task will include both bug fixes and handling error cases such as input text limit characters, invalid connection, invalid server, trying to save empty vocab ...etc

Task 6: Add Animations

Add button animations, recyclerview animations, shared element transitions and activity and fragment transitions.

Task 7: Adding Widget

Create a widget that can be easily used either for translation or viewing saved vocabs

Task 8: Adding Notifications

Choosing a random vocab and sending a notification with only the phrase asking the user to try to remember its meaning.

Task 9: Google Play Services Implementation

In this task Google Play Services will implemented so as to integrate both:

- Adware services
- Google Analytics services

Task 10: Accessibility Support

Subtasks:

- Add content descriptions
- Navigation using D-pad

Task 11: Integration And Testing

Making sure that everything is working fine together on phone.

Task 12: Responsive Design Implementation

Designing for different sizes and tablets

Subtasks:

- Design for mobile landscape mode
- Design for 7” tablet in both portrait and landscape modes
- Design for 10” tablet in both portrait and landscape modes

Task 12: Build

Create a free signed apk version.

Submission Instructions

1. After you’ve completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it “**Capstone Project**”
3. Add this document to your repo. Make sure it’s named “**Capstone_Stage1.pdf**”