GitHub Username: BBarisKilic

what is...

Description

Search words in **"what is…"** and find definitions for many languages like "Spanish, English, Latvian, Hindi, Swahili, Tamil and Gujarati" as Oxford Dictionaries supports. Oxford Dictionaries is at the forefront of lexical research.

Search history will be recorded and you can easily access. Also home widget is provided with latest research history entries.

In App's "More" screen, language selection can be done easily.

Your **"what is…"** Free version includes:

- Spanish Language
- Definitions
- Example Sentences

Your **"what is…"** Premium version includes:

- Access to all languages which Oxford API supports
- Supported Languages: Spanish, English, Latvian, Hindi, Swahili, Tamil and Gujarati
- Definitions
- Example Sentences
- No advertisement

Intended User

Free Version:

- Free version of this app is for non-native or native Spanish speakers who like to improve their vocabulary knowledge.

Premium Version:

- Premium version of this app is for non-native or native speakers who like to improve their vocabulary knowledge in Spanish, English, Latvian, Hindi, Swahili, Tamil or Gujarati.

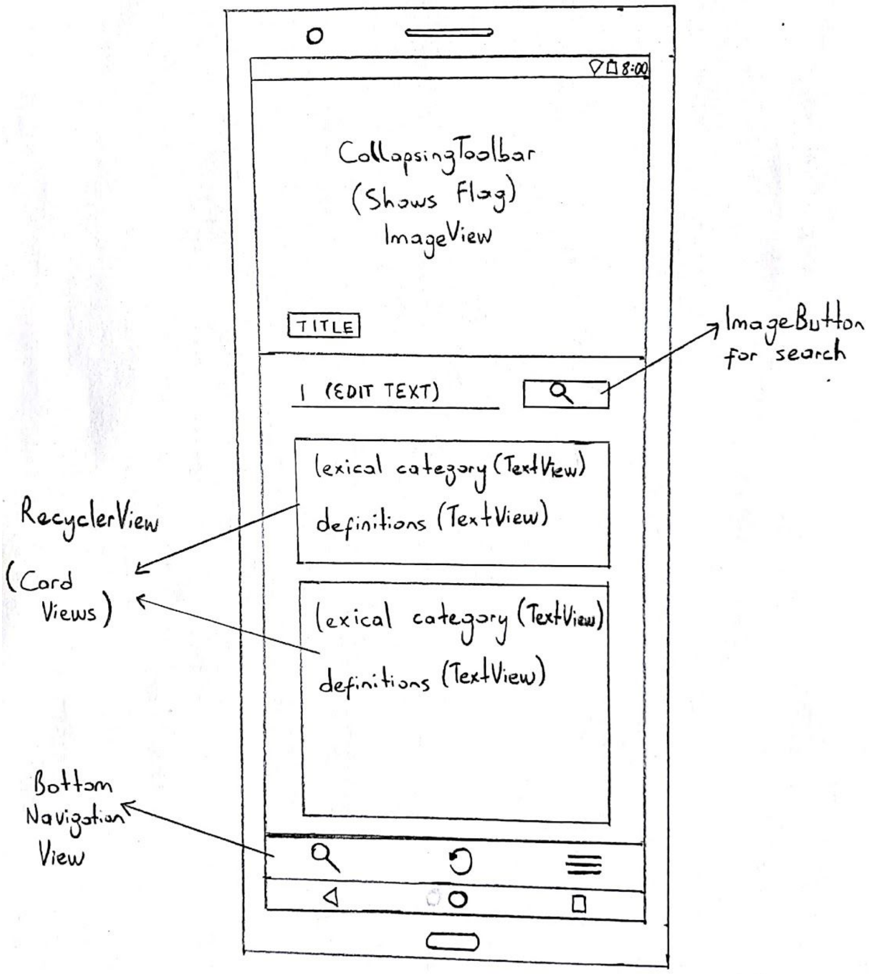
Features

- Searching words
- Viewing definitions
- Viewing example sentences
- Viewing history records
- Viewing past research records in home widget
- Clearing all history with one click
- Selecting language

User Interface Mocks

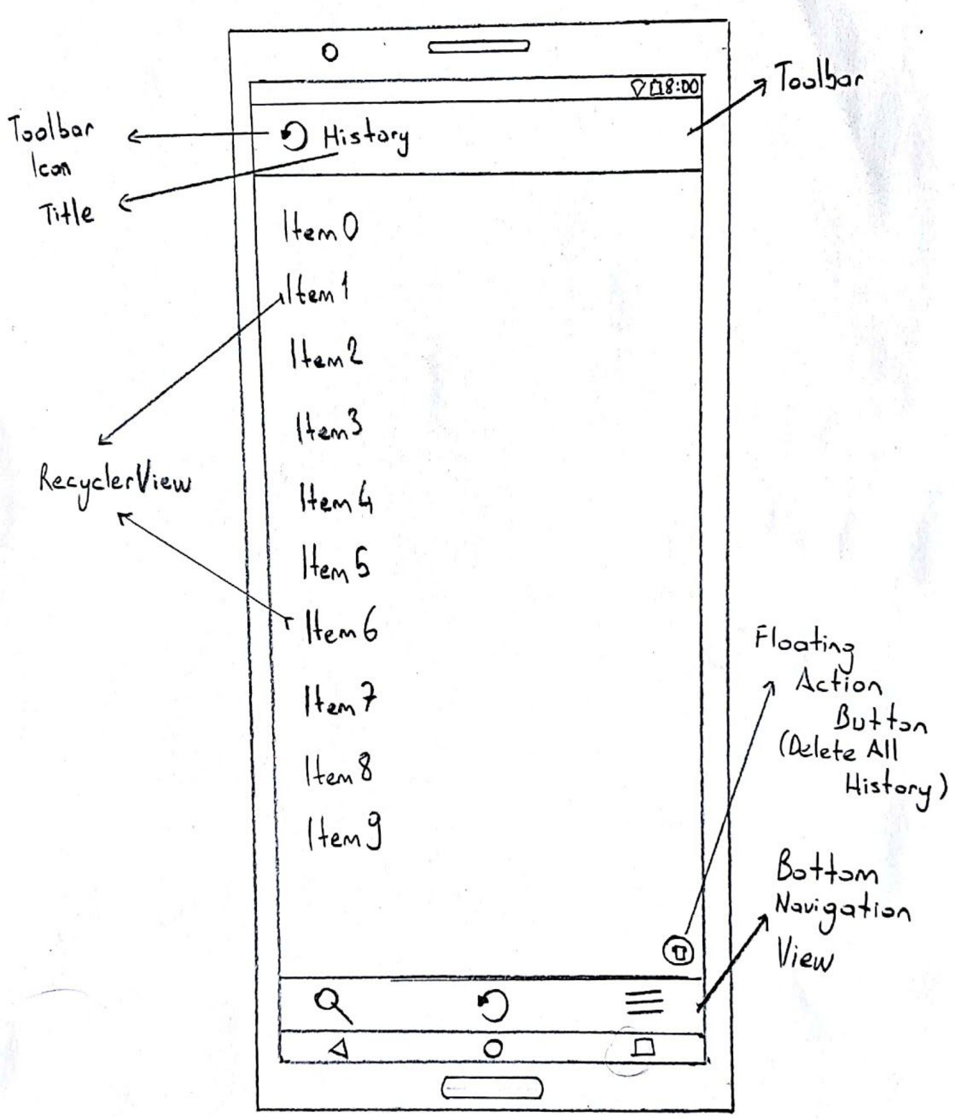
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Google Drawings, www.ninjamock.com, Paper by 53, Photoshop or Balsamiq.

Screen 1



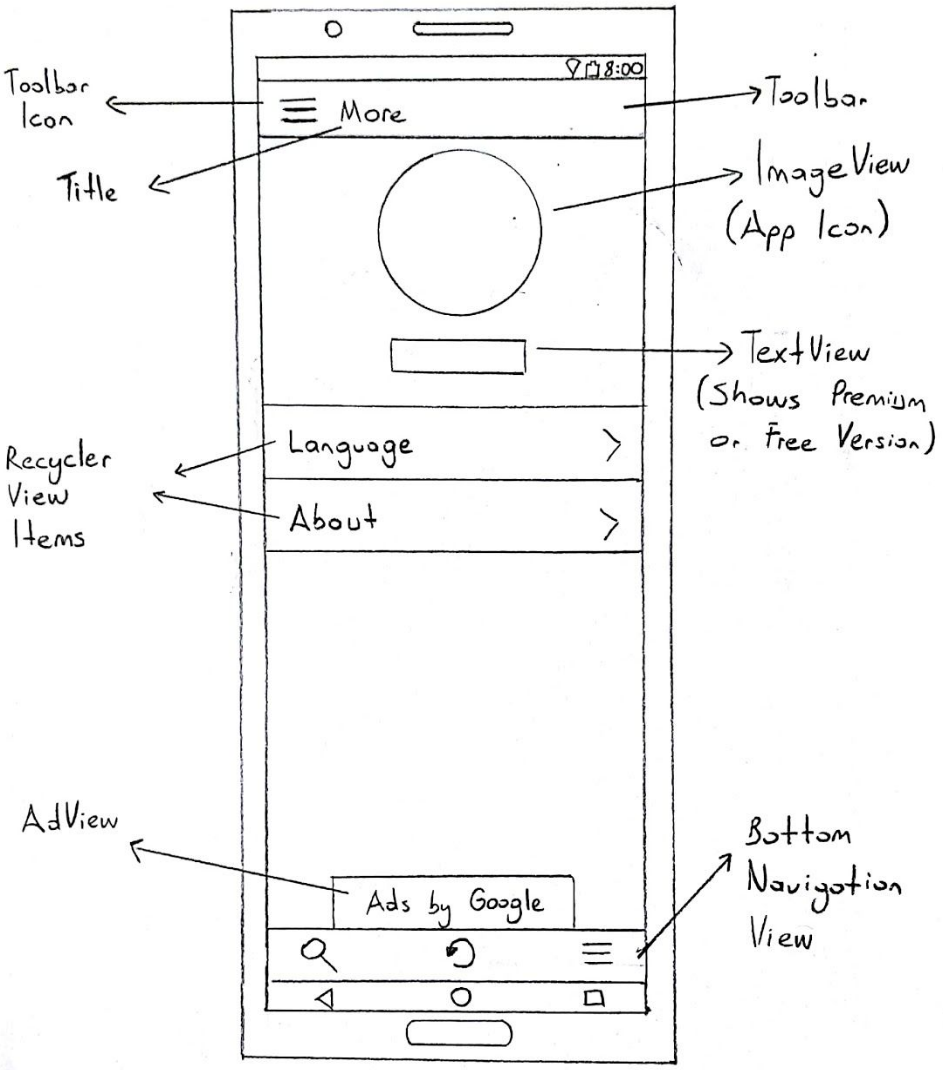
Main screen will include “Collapsing Toolbar” and inside of it there will be “ImageView” and “TextView”. “ImageView” will show national flag of selected language and “TextView” stands for showing title of app. At the bottom of screen there will be “BottomNavigationView” for direction to other screens. As in plan 1, “EditText” button will be in use for search entry and “ImageButton” for resulting search. Results will be shown in “CardView” and it will contain two “TextView”; one for showing lexical categories and one for showing definitions and example sentences.

Screen 2



History screen will stand for showing latest searches. As in the plan, at the top of screen there will be “Toolbar” and inside, icon and title of current screen will exist. At the bottom of the screen there will still be “BottomNavigationView” for direction to other screens. Between “Toolbar” and “BottomNavigationView” there will be “RecyclerView” for showing latest searches. When search entry is clicked, “MainActivity” will open with that entry search. Also at the right bottom side of the screen there will be a “FloatingActionButton” for deleting all history when clicked.

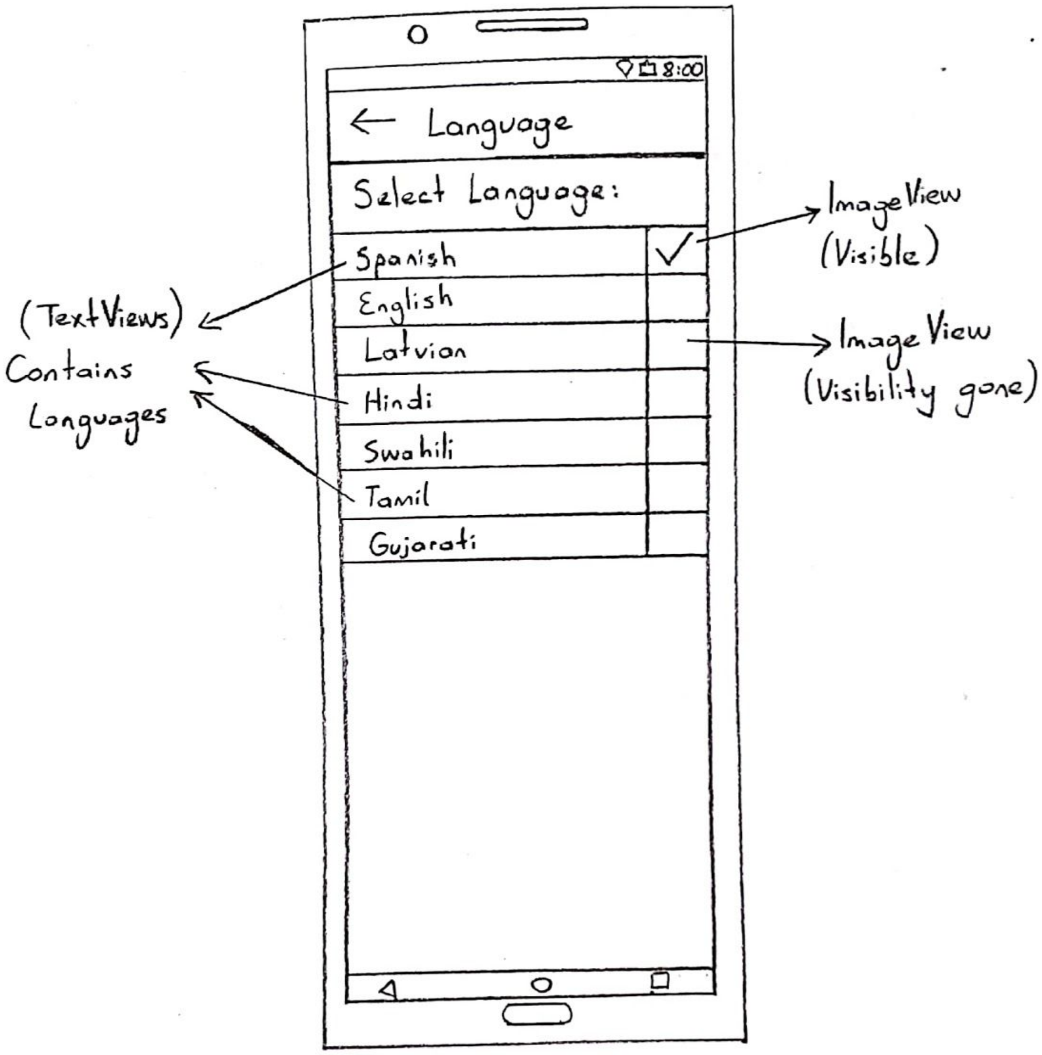
Screen 3



More screen will include “Toolbar” at the top of the screen, “ImageView” will be under “Toolbar”, “TextView” under “ImageView”, “RecyclerView” and its items under “TextView” and at the bottom of the screen there will still be “BottomNavigationView” for direction to other screens(Main, History) and lastly “AdView” above “BottomNavigationView”.

“Toolbar” will contain icon and will show title of screen. “ImageView” will show icon of the app. “TextView” will show version of app(Free or Premium). First item of “RecyclerView(Language)” will open select language screen and second item of “RecyclerView(About)” will open “about screen” when clicked.

Screen 4

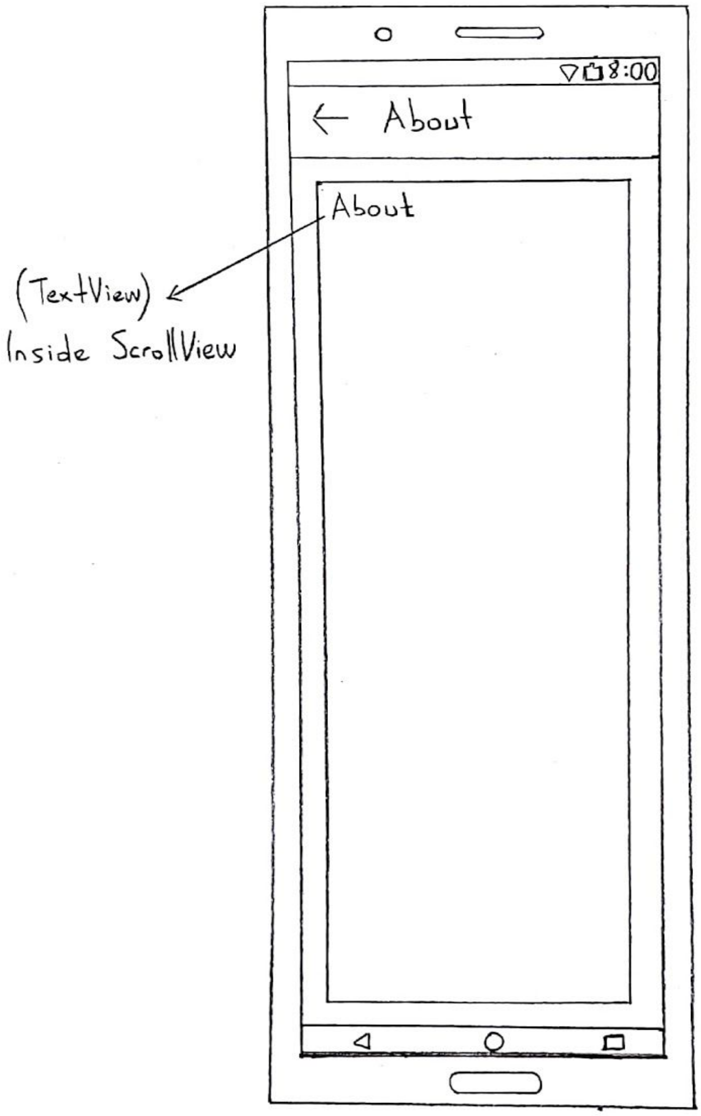


Language screen stands for changing language of dictionary. Parent view is “ScrollView” and inside of it as in plan there will be eight “ConstraintLayout”. Seven “ConstraintLayout” will contain “TextViews” and “ImageViews”.

In premium version when user change language, it will be saved in file for next start of app also selected language’s “ImageView” will be visible and “Snackbar” will show “X language selected”

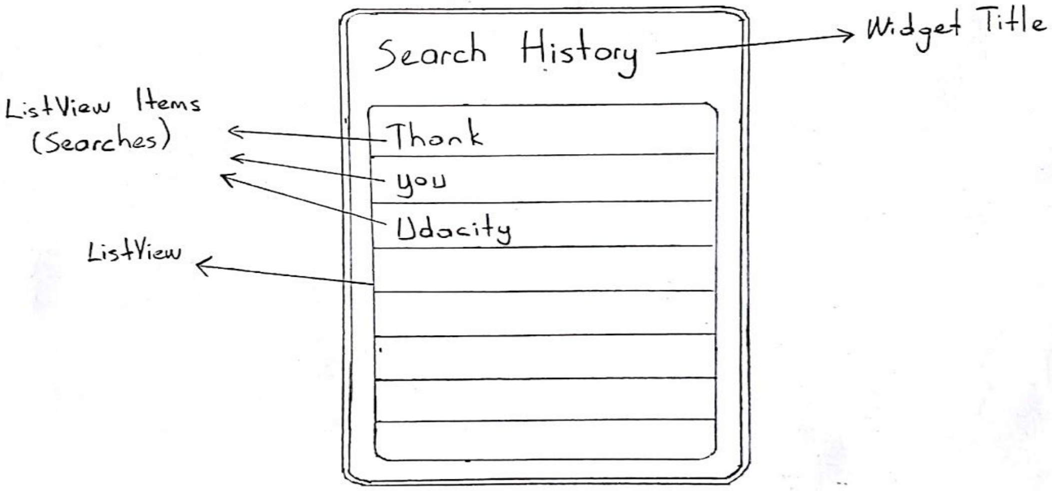
In free version when user try to change language “Toast” will show “Get premium for access to other languages!”

Screen 5



About screen will contain single “TextView” in “LinearLayout” and “LinearLayout” will be inside of “ScrollView”. This “TextView” will be used for showing my text including my thinkings about this journey in last 9 months.

Screen 6



App Widget will be displayed last ten history records. When search entry is clicked, “MainActivity” will open with that entry search.

Key Considerations

How will your app handle data persistence?

Room persistence library will be used for saving all history. If searching is successful, search entry will be saved to history database. And user will be able to see all history via clicking “History” in “BottomNavigationView”. In History screen, user will easily delete all entries via clicking “FloatingActionButton”.

Describe any edge or corner cases in the UX.

- The app will be written solely in the Java Programming Language.
- In “MainActivity” when user search something until search ends user will see “LoadingIndicator” and when search done “LoadingIndicator” will be gone.
- In “MainActivity” when user search something and if there is no result or error; “TextView” will be visible and show “Unable to resolve host…” or “There is no data for this sarch!” (ViewModel “observe” will be used)
- App keeps all strings in a “strings.xml” file.
- App keeps app’s icon in mipmap folder.
- App keeps all other images in drawable folder.
- App will have 2 AppThemes in “styles.xml” (e.g, “MainActivity” will use “AppTheme” and “HistoryActivity” will use “AppThemeAddition”)
- Resources like colors and dimensions will be stored in resource xml files.
- Accessibility will be addressed using content descriptions and RTL support.
- “Live Data” and “View Model” libraries are helps to design robust, testable, and maintainable apps. Start with classes for managing UI component lifecycle and handling data persistence.
- AsyncTask will be used for database operations.
- Android Studio 3.1.3, google-services plugin 4.0.0
- Search results show definitions and example sentences for each lexical category inside of CardView in MainActivity.
- Java will be used for app implementation.
- All libraries are stable versions.

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

Butterknife -> Eliminate findViewById calls by using @BindView on fields. Eliminate anonymous inner-classes for listeners by annotating methods with @OnClick and others. Eliminate resource lookups by using resource annotations on fields.

Retrofit -> Retrofit is a REST Client for Android and Java by Square. It makes it relatively easy to retrieve and upload JSON (or other structured data) via a REST based web service. In Retrofit you configure which converter is used for the data serialization.

Picasso -> To handle the loading and caching of images.

Android Architecture Components(LiveData, ViewModel) -> A collection of libraries that help you design robust, testable, and maintainable apps. Start with classes for managing your UI component lifecycle and handling data persistence.

Stetho -> Stetho is a sophisticated debug bridge for Android applications. When enabled, developers have access to the Chrome Developer Tools feature natively part of the Chrome desktop browser.

Room -> Room is a persistence library, part of the Android Architecture Components. It makes it easier to work with SQLiteDatabase objects in your app, decreasing the amount of boilerplate code and verifying SQL queries at compile time.

Describe how you will implement Google Play Services or other external services.

- AdMob : By displaying an Ad banner in More Activity
- Analytics : Log events to see which features are used in the app.

Stable Versions of used libraries:

- Version of constraintLayout = '1.1.2'
- Version of supportLibrary = '27.1.1'
- Version of retrofit = '2.4.0'
- Version of picasso = '2.5.2'
- Version of archComponents = '1.1.1'
- Version of okHttp3 = '3.9.1'
- Version of room = '1.1.0'
- Version of lifecycle = '1.1.1'
- Version of firebaseAdmob = '15.0.1'
- Version of firebaseAnalytics = '16.0.1
- Version of butterknife = '8.8.1'
- Version of stetho = '1.5.0'

Next Steps: Required Tasks

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

Task 1: Project Setup

- Create new project
- Configure the required dependencies in the build.gradle file and ensure that everything is compatible and there is no error
- Add vector support

Task 2: Implement UI for Each Activity and Fragment

- Build UI for MainActivity
    - Implement CollapsingToolbarLayout
    - Implement EditText for user entry
    - Implement ImageButton for start search
    - Implement RecyclerView that displays results
    - Implement BottomNavigationView

- Build UI for HistoryActivity
    - Implement Toolbar
    - Implement FloatingActionButton which deletes all history when click
    - Implement RecyclerView that displays latest searches.
    - Implement BottomNavigationView

- Build UI for MoreActivity
    - Implement Toolbar
    - Implement ImageView which shows App’s icon
    - Implement TextView which shows App’s version(Free or Premium)
    - Implement RecyclerView that contains “Language and About” items
    - Implement AdView in free Version
    - Implement BottomNavigationView

- Build UI for LanguageActivity
    - Implement TextView which shows “Select Language”
    - Implement TextViews which shows languages
    - Implement ImageView which visible when language selected

- Build UI for AboutActivity
    - Implement TextView that displays my feels about this journey in last 9 months.

- Build UI for app widget

Task 3: Handling Network Connection

- Create Manager of Data and add methods to use Oxford API
- Define API Key and APP ID
- Create models for JSON Data
- Create API service

Task 4: Build Data persistence Support

- Create App DataBase
- Create DataAccessObject model and interface
- Create TypeConverter for Date

Task 5: Implement ViewModels and Adapters

- Create ViewModelFactory for MainActivity
- Create ViewModel for MainActivity
- Create ViewModelFactory for HistoryActivity
- Create ViewModel for HistoryActivity
- Create Adapter for HistoryActivity
- Create Adapter for MoreActivity

Task 6: Build the app widget

- Build widget layout
- Build RemoteViewsFactory and Service
- Build AppWidgetProvider
- Add update codes when database change

Task 7: Create flavor

- Create two flavor - free and paid
- Change UI for each version

Task 8: Implement Google Play Services

- Implement Ads Support
- Create Firebase project and add google-services.json to app
- Implement Analytics support for send log events to see which features are used in the app

Task 9: Configure App Building

- Create a keystore
- Add Sign in Configuration

Submission Instructions

- After you've completed all the sections, download this document as a PDF [File → Download as PDF]
    - Make sure the PDF is named "**Capstone_Stage1.pdf**"
- Submit the PDF as a zip or in a GitHub project repo using the project submission portal

If using GitHub:

- Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
- Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"