Technological Basics II | Report

Lukas Mikulec

The idea of my app for Tech Basics II was already presented in Tech Basics I and pitched in December 2023, therefore, this report only provides a brief summary of the idea and focuses mainly on the minimum viable product (MVP) which is the main part of this assignment. My app, *Activities,* is trying to solve the problem of bubble creation among immigrants in Germany. The mechanism of the app is to connect people based on activities they like to do or would like to try. The connection always happens between German(s) and non-German(s) which should help with the integration process in Germany and spark genuine connections across cultural divides. As shown before, no solution like this exists yet on the market, with only more pragmatic apps available, such as job and accommodation searching or culture teaching for immigrants.

Although I did not use a professional methodology to develop my MVP per se, among water, agile, and rapid application development (RAD) methodologies, RAD is the closest to what I have done. I had an idea of the main features for the app from the previous assignment, design mock-ups, but the precise implementation was developing on the go. In the beginning, I mostly programmed the essential features according to the design mockups and then I moved toward more specialized features. Some functionalities changed as I programmed since being in the headspace of the code allowed me to rethink the flows of the app and improve them. Small improvements I only undertook after the app was already functional and the design improved from sufficient to proficient in the final stage as well. While I programmed in an organized way and commented on my code along the way, the final improvement of the readability of the code was the last step in the process.

My MVP is a fully functional app that could already connect people. In the beginning, the user is either asked to log in or register for the app, but regardless of the steps, they always land on the *Activities* page (home page). As the app uses a CSV file to store user data information, an account is a real thing in the MVP, as settings user sets get saved and the app opens in the same state as the user left it. An important point from the architectural side is that each time a user setting is saved, the CSV file is loaded, updated, and saved, so even if the app crashes down during the user session, the progress is never lost (crashing can be caused if the user clicks too fast between pages which contain the map widget, as the module needs some time to load the map).

After logging in, the user can see activities suggested to them. For ease of use, only two suggestions are shown at this time, each corresponding to the preferred activity type the user selected. If a user changes their preferences, they will see different suggestions for different categories. The actual shown activity for a category is determined by the closeness from the user: the app identifies the user location based on their IP address and only shows the closest activity to them in, let us say, the Cooking category. The user can click on a *Show more* button for each activity which will display the activity's location, description, and contact for people. On this details page for the activity, the user can also save the activity to their connections which will result in this activity showing in the *Connections* tab. There can be at most 4 connections at a time, so the user can keep track of a lot of activities (not just the current suggestions). Going back to the *Activities* page from the details of an activity, there is an *Add activity* button that allows the user to submit an activity of their own for other users to see. Data for the new activity is validated throughout the adding procedure, so that e.g. wrongly entered coordinates do not wreak havoc in the app. As user data is stored in a CVS file, activities are too. This CSV file of activities is updated once the user adds their own activity and the app results are updated for the other users, as the data are loaded from this activities file.

The *Map* page has two modes. The default one shows the same two activities that the user can see on the *Activities* page on a map (the nearby ones). However, to give the user more freedom and options, they can choose the second mode "All activities", which will display all the available activities in the categories that the user selected. This means that the user can also see activities that are far away. Activities have their respective icon on the map and the activity name for easy navigation. Once the user clicks on the icon, the same activity details page will open as on the homepage.

Connections page, which was already mentioned earlier, shows activities that the user saved through the activity details page before. The same image as on the *Activities* page is used for captivating design and the phone number of the activity's organizers is displayed. The user can click on the WhatsApp icon of each connection and open the phone number in WhatsApp directly. They can also click on the bin icon to remove the connection.

The last page is the *Settings* page. In the *Accounts settings* tab, the user can change their activity preferences, looking for group/another person preference, and password. Status, name, or username cannot be changed because they are fundamental attributes of the user account and should not need to change. Data is validated (eg. password must contain some characters and have a minimum length) and the user can save the changes by both filling out

only one change or more. They can also log out of the page, if someone else wanted to log in or register. The *App settings and info* provide a toggle to turn dark mode on or off for the user (this setting gets saved and the next time the user logs in, their color mode gets loaded automatically) and short information about the app and attribution for images.

Despite the app being fully functional, there is always room for improvement. The first aspect I would mention is that for now, the app cannot be used by multiple people because the data is stored locally in a CSV file. Should the app become available on the market, a new way of storing data through a central server would have to be developed. This would have to include an advanced way of communication between an app and the server, as information like passwords cannot be sent over as plain text. For people to use the app, it would also have to be made into an executable file, preferably on the phone, so it could be easily installed. The app could also support scrolling in the future so that more than just a few activities or connections can be displayed. In my MVP, I did not implement this ability to keep the design reasonable and feasible. Another thing that could be improved would be the loading of offline maps instead of online ones, which would avoid the loading of map data when moving around the map. Additionally, the map should save its position so that once a user comes back from activity details to which they got through the map, the map will be on the same spot. This is, however, a limitation of the tkinter map module, as it does not allow to store the current map position. Activities could also display more information, such as age range, which could be also used to filter them. This feature was not developed due to the scrolling non-inclusion decision and also to keep the app reasonably complex for the assignment. Lastly, the app should have a framework for reporting activities, but I did not develop this in my MVP as this would have to assume another interface for the app support team, which there is none currently.

I based the design of my app on the mockups from the Tech Basics I assignment and developed it further. Fonts and colors are unified and I use Material Design icons to give the app a modern look. I also used the customtkinter library to use modern widgets in my app. The activity images were taken from Freepik and are attributed in the app and in the readme file. As the app is larger and more complex than the last time, code references are made in a separate file on GitHub called code_referencing.txt. Include references here would lead to half of the report being just a reference list. Details on how to run the app are at the beginning of my app code in the app.py file. Enjoy connecting!